[EspressReport 7.0]
ユーザガイド

Copyright(C) 20

株式会社クライム

Climb Inc.
Growing to Meet Your Needs

作成日: 2018/3/20(火)
更新日: 2018/3/20(火)
バージョン: 1.0

目次

1 はじめに 10
1.1 範囲 10
1.2 対象バージョン 10
2 クイックスタート 11
2.1 概要 11
2.1.1 EspressReport アーキテクチャ 11
2.2 主な特徴 13
2.3 Designer のクイックスタート 15
2.3.1 Report Designer の起動 15
2.3.2 データソースを設定する 16
2.3.3 レポートマッピング 32
2.3.4 基本的なレポート書式設定 45
2.3.5 数式とスクリプティング 56
2.3.6 ドリルダウン 63
2.3.7 サブレポート 69
2.3.8 チャートの操作 74
2.4 API のクイックスタート 88
2.4.1 環境を設定する 88
2.4.2 レポートを実行する 90
2.4.3 プログラムでレポートを作成する 101
2.4.4 レポートのデータソースの変更 106
2.4.5 レポート要素の変更 120
2.4.6 パラメータ化されたレポート 127
2.4.7 ドリルダウンの展開/エクスポート 136
2.4.8 サーブレットの詳細 140
2.4.9 Report Designer の起動 145
3 デザイナーガイド 147
3.1 概要 147
3.1.1 EspressReport ドキュメント 147
3.2 アーキテクチャとインストール 148
3.2.1 EspressReport アーキテクチャ 149
3.2.2 インストール 150
3.2.3 設定 154
3.2.4 EspressManager の起動 156
3.2.5 Report Designer の起動 166
3.2.6 下位互換性パッチ 170
3.3 Data Source の動作 173
3.3.1 Data Source Manager 173
3.3.2 Database からのデータ 175

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.3.3 XML および XBRL ファイルのデータ 217
3.3.4 テキストファイルのデータ 222
3.3.5 クラスファイルのデータ 224
3.3.6 EJB のデータ 225
3.3.7 WSDL をサポートする SOAP のデータ 228
3.3.8 Salesforce のデータ 231
3.3.9 Excel ファイルのデータ 236
3.3.10 レポートの使用 238
3.3.11 データソースの更新 240
3.4 レポートの種類とデータマッピング 243
3.4.1 シンプルカラムナレポート 243
3.4.2 サマリブレークレポート 245
3.4.3 クロス集計レポート 248
3.4.4 マスタ&詳細レポート 254
3.4.5 メーリングラベルレポート 258
3.4.6 フォーマットオプションの追加 260
3.4.7 列をページ幅に合わせる 265
3.4.8 データマッピングの変更 265
3.4.9 データソースの順序を維持 266
3.5 デザイナーインタフェース 268
3.5.1 レポートセクション 268
3.5.2 セクションのリサイズ 269
3.5.3 セクションオプション 270
3.5.4 ルーラー 273
3.5.5 デザイナーメニュー 274
3.5.6 デザイナーツールバー 282
3.5.7 レポート要素の挿入と操作 285
3.5.8 Report Explorer 337
3.5.9 グローバル書式設定、グループ書式設定、テンプレート 338
3.5.10 Mac OS X での Report Designer 342
3.6 プレビューウィンドウ 343
3.6.1 レポートのナビゲート 343
3.6.2 その他のプレビューウィンドウオプション 344
3.6.3 ページプロパティの設定 350
3.7 レポートの保存とエクスポート 352
3.7.1 レポートの保存 352
3.7.2 レポートのエクスポート 354
3.8 数式および式ビルダの使用 361
3.8.1 数式の作成 361
3.8.2 数式構文 363
3.8.3 サブレポートの数式アクセス 377

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.9 スクリプト 380
3.9.1 スクリプトの作成 380
3.9.2 スクリプト構文 382
3.9.3 スクリプトの管理 393
3.10 ドリルダウン 395
3.10.1 ドリルダウンレポートの作成 397
3.10.2 複数値ドリルダウン 399
3.10.3 クロス集計ドリルダウン 399
3.10.4 ドリルダウンレポートの表示 401
3.10.5 ドリルダウンのマッピング解除 403
3.11 サブレポート 406
3.11.1 サブレポートの追加 406
3.11.2 サブレポートの編集 406
3.11.3 サブレポートの削除 408
3.11.4 リンクされたサブレポート 409
3.11.5 パラメータのリンク 410
3.12 テンプレートセキュリティ 412
3.12.1 セキュリティレベル 414
3.12.2 セルレベルのセキュリティ 414
3.12.3 サブレポート＆ドリルダウンレイヤのセキュリティ 417
3.12.4 保護されたテンプレートの表示 418
3.13 スケジューリング 419
3.13.1 スケジューラインタフェース 419
3.13.2 スケジュールの設定 421
4 プログラミングガイド 433
4.1 Report Viewer の概要 433
4.1.1 前書き 433
4.1.2 Report Viewer の起動 433
4.1.3 Report Viewer のパラメータ 434
4.1.4 Report Viewer のデータソースの指定 436
4.1.5 Report Viewer の使用 438
4.1.6 パラメータサーバ 440
4.1.7 EspressManager に接続する 440
4.1.8 Swing バージョン 441
4.1.9 ビューアからのエクスポート 441
4.2 Page Viewer の概要 443
4.2.1 前書き 443
4.2.2 Page Viewer の起動 443
4.2.3 Page Viewer のパラメータ 444
4.2.4 Page Viewer の使用 447
4.2.5 バッファ時間 449

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.2.6 EspressManager に接続する 450
4.2.7 Swing バージョン 450
4.2.8 ビューアからのエクスポート 450
4.3 EspressReport レポート API 452
4.3.1 導入とセットアップ 452
4.3.2 レポート API を使用するための推奨アプローチ 453
4.3.3 EspressManager とのインタラクション 453
4.3.4 EspressManager に接続する 454
4.3.5 API の使用 455
4.4 スケジューラ 536
4.4.1 前書き 536
4.4.2 EspressManager に接続する 537
4.4.3 スケジューラグラフィカルインタフェースの呼び出し 539
4.4.4 エクスポートのスケジューリング 540
4.4.5 コマンドのスケジューリング 547
4.4.6 スケジュールの削除 547
4.4.7 失敗したスケジュールの詳細の取得 548
4.4.8 ICallBackScheduler インタフェース 550
4.4.9 スケジューラリスナー 551
4.4.10 レポートのバースト 553
4.4.11 Javadoc 554
4.4.12 概要 554
4.5 サーブレットと Java Server Pages 555
4.5.1 サーブレット 555
4.5.2 Java Server Pages（JSP） 579
4.5.3 ファイルへのレポートの保存とブラウザへのレポートの送信 581
4.6 WebObjects を使った作業 588
4.6.1 前書き 588
4.6.2 基本 API 588
4.6.3 WebObjects アプリケーションのデプロイ 596
4.6.4 ノート 617
4.7 デプロイメント 618
4.7.1 前書き 618
4.7.2 EspressManager でデプロイする 620
4.7.3 EspressManager なしでのデプロイ 625
4.7.4 非 Windows 環境での展開 629
4.7.5 プラットフォーム固有の問題 630
4.8 付録 A. パラメータサーバ 632
4.8.1 パラメータサーバの作成 632
4.9 付録 B. レポートデータの取得 639
4.9.1 データベースのデータ 639

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.9.2 データファイルのデータ（TXT/DAT/XML） 641
4.9.3 XML データソースのデータ 643
4.9.4 メモリ内の配列に渡されたデータ 644
4.9.5 カスタム実装で渡されるデータ 646
4.9.6 Enterprise Java Bean（EJB）のデータ 651
4.9.7 SOAP データソースのデータ 651
4.9.8 複数のデータソースからのデータ 652
4.10 付録 C. レポートの作成 654
4.10.1 シンプルカラムナ 655
4.10.2 サマリブレーク 657
4.10.3 クロス集計 660
4.10.4 マスタ&詳細 665
4.10.5 メーリングリスト 667
4.10.6 式 670
4.10.7 パラメータ化されたレポート 671
4.10.8 サブレポート 674
4.10.9 ドリルダウン 680
5 チャートガイド 684
5.1 概要 684
5.2 チャート作成の基礎 684
5.2.1 チャートとは 684
5.2.2 基本的データマッピング 687
5.3 チャートの作成 690
5.3.1 データソースオプション 690
5.3.2 複数のデータソース 691
5.4 チャートの種類ごとのデータマッピング 692
5.4.1 データマッピング 694
5.4.2 縦棒グラフ 696
5.4.3 横棒グラフ 699
5.4.4 XY(Z)散布図 701
5.4.5 折れ線グラフ 702
5.4.6 スタック縦棒グラフ 703
5.4.7 スタック横棒グラフ 706
5.4.8 円グラフ 707
5.4.9 エリアチャート 709
5.4.10 スタックエリアチャート 711
5.4.11 High-Low チャート 712
5.4.12 HLCO チャート 713
5.4.13 100%スタック横棒グラフ 715
5.4.14 ドーナツチャート 716
5.4.15 サーフェスチャート 717

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.4.16 バブルチャート 718
5.4.17 オーバーレイチャート 719
5.4.18 ボックスチャート 721
5.4.19 レーダーチャート 723
5.4.20 ダイヤルチャート 725
5.4.21 ガントチャート 728
5.4.22 極座標 729
5.4.23 データマッピング/データソースの変更 730
5.5 デザイナーインタフェース 731
5.5.1 デザイナーメニュー 731
5.5.2 Designer のツールバー 742
5.5.3 カラー/カラーセット/パターン/フォントパネル 743
5.5.4 ナビゲーションパネル 749
5.5.5 ビューポート 751
5.5.6 チャート要素の追加 755
5.5.7 グラフ軸の書式設定 782
5.5.8 プロット/データ要素の書式設定 791
5.5.9 グラフ固有のオプション 803
5.6 ドリルダウン 827
5.6.1 データドリルダウン 827
5.6.2 動的データドリルダウン 831
5.6.3 パラメータドリルダウン 833
5.7 チャートの保存およびエクスポート 835
5.7.1 レポートデータによるチャートの保存 835
5.7.2 レポートデータなしのチャートの保存 835
5.7.3 チャートのエクスポート 841
5.8 チャートビューア 845
5.8.1 チャートビューアのパラメータ 846
5.8.2 チャートビューアのデータソースの指定 850
5.8.3 チャートビューアの使用 853
5.8.4 軸ルーラー 855
5.8.5 パラメータサーバ 855
5.8.6 ポップアップメニュー 855
5.8.7 利用可能な Swing バージョン 856
5.9 EspressReport Chart API 857
5.9.1 概要およびセットアップ 857
5.9.2 Chart API を使用するために推奨されるアプローチ 858
5.9.3 EspressManager とのインタラクション 859
5.9.4 EspressManager に接続する 860
5.9.5 API の使用 861
5.9.6 API のみの機能 880

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.9.7 チャートビューアオプションの変更 900
5.9.8 Javadoc 900
5.9.9 スイングバージョン 900
5.9.10 サマリ 901
5.10 付録 A. チャートレイアウトのカスタマイズ 902
5.10.1 キャンバスエリアの変更 902
5.10.2 キャンバス領域の変更 903
5.10.3 チャートにフィットする要素 904
5.10.4 凡例の位置の変更 904
5.10.5 データポイントへのラベルの添付 905
5.11 付録 B. チャートデータの取得 909
5.11.1 データベースのデータ 910
5.11.2 データファイル（TXT/DAT/XML）のデータ 911
5.11.3 XML データソースのデータ 913
5.11.4 メモリ内の配列に渡されたデータ 914
5.11.5 カスタム実装で渡されたデータ 915
5.11.6 スプレッドシートモデルのデータ 920
5.11.7 Enterprise Java Beans (EJBs) のデータ 922
5.11.8 複数のデータソースのデータ 922
5.11.9 スプレッドシート形式のデータ 923
5.11.10 データ変換 925
5.12 付録 C. チャートの作成 928
5.12.1 縦棒/横棒/折れ線/エリア/円/オーバーレイチャート 929
5.12.2 レーダーチャート 931
5.12.3 XY(Z)散布図 933
5.12.4 スタック縦棒/割合縦棒/スタック棒/スタックエリアチャート 935
5.12.5 ダイヤルチャート 938
5.12.6 ボックスチャート 940
5.12.7 バブルチャート 942
5.12.8 High-Low/HLCO チャート 945
5.12.9 サーフェスチャート 948
5.12.10 ガントチャート 950
5.12.11 極座標 952
5.12.12 日付・時間ベースのズームチャート 954
5.12.13 パラメータ化されたグラフ 956
5.12.14 ドリルダウンチャート 960
5.12.15 レポートへのチャートの追加 970
6 ローカライズ 974
6.1 EspressReport のローカライズ 974
6.1.1 ローカライズの指定 974
6.1.2 言語とエンコーディング 975

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

7 更新履歴.....982

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

1 はじめに

- 本ドキュメントに記載されたイラスト、写真、文章の一部またはすべてを無断で複製、転載することを禁止します。
- 本ドキュメントは製品を購入されたお客様、評価版をご使用のお客様向けに株式会社クライムが提供しております。

1.1 範囲

本ドキュメントは、[EspressReport]のインストール手順について記載しております。

1.2 対象バージョン

本ドキュメントは、以下の製品のバージョンに対応しております。

- [EspressReport] Ver. 7.0

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2 クイックスタート

2.1 概要

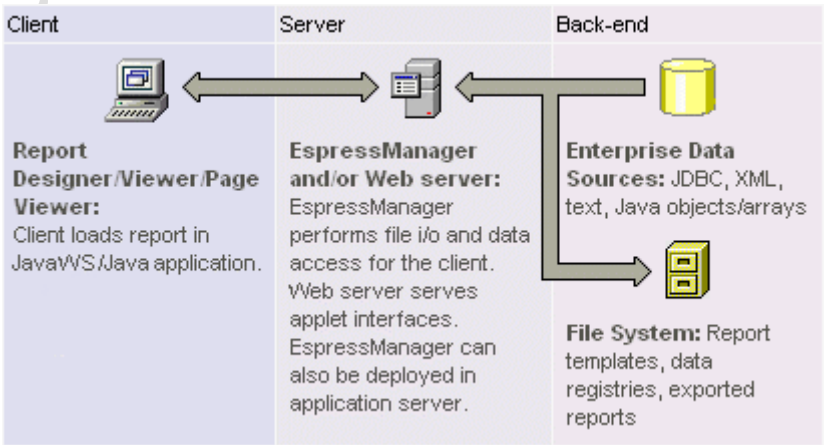
EspressReport は、インターネットやイントラネットアプリケーションで情報豊富なレポートを簡単に設計して展開できる強力な Java レポートツールです。EspressReport は事実上すべてのデータソースに接続し、DHTML スタイルシート、印刷品質の PDF、Microsoft Excel、またはリッチテキストの情報を簡単にフォーマットしてレンダリングできます。またテキスト、CSV、および XML 形式にデータを抽出することもできます。強力な Java API により、アプリケーション、サーブレット、JSP、および JavaWS にレポート機能を簡単に組み込むことができます。

これは、EspressReport クイックスタートガイドです。EspressReport を初めてご使用になり、迅速にスピードアップしたい場合、または製品の機能を評価したい場合は、クイックスタートガイドを参照することをお勧めします。クイックスタートガイドでは、EspressReport の最も一般的に使用されている機能のいくつかを示すサンプルレポートとコードだけでなく、レポートの設計と実行の例も提供しています。

2.1.1 EspressReport アーキテクチャ

EspressReport には、設計時と実行時の両方で実行できるさまざまな構成があります。設計時に、データアクセスツールとチャートツールを含む Report Designer の GUI インタフェースは、クライアントマシン上のアプリケーションとして、またはクライアントサーバアーキテクチャ内のアプレットとしてロードできます。

Report Designer が実行されているとき、EspressManager コンポーネントはサーバ側で実行されます。EspressManager は、セキュリティ制限のためにクライアントアプレットによって防止されるデータアクセスとファイル I/O を実行します。EspressManager は、データベース接続のための接続とデータのバッファリングと、マルチユーザ開発環境の同時実行制御も提供します。EspressManager は Report Designer と連携して実行する必要があります。



設計時の EspressReport アーキテクチャ

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

実行時に、EspressManager を実行する必要はありません。EspressReport は、他のアプリケーション環境に組み込まれて動作するように設計されており、API クラスとレポートテンプレートファイルの配置を最小限に抑えることができます。アプリケーションサーバで実行する場合、Report API を使用してサーブレットと JSP 内でレポートを生成し、生成されたレポートをクライアントブラウザにストリーミングできます。クライアントは、レポートビューアアプレットを読み込んでレポートを表示することもできます。

レポート生成は、オンデマンドで処理したり、アプリケーションプロセスによってトリガされたり、スケジュールされたりすることがあります。EspressReport には、インタフェースも用意されています。スケジューラを実行するには、EspressManager も実行している必要があります。

EspressReport のアーキテクチャとデプロイメントオプション/コンフィグレーションの詳細については、プログラミングガイドのアーキテクチャとインストール、およびサーブレットと Java Server Pages を参照してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2.2 主な特徴

このセクションでは、EspressReport パッケージに含まれる主要な特徴と機能のいくつかについて説明します。

Pure Java Architecture

Windows、Solaris、Linux、Mac OS X、HP UX、その他の Unix プラットフォームを含むほとんどのプラットフォームで簡単にデプロイできます。

フル機能の Report Designer

Report Designer のインタフェースは、ユーザがレポートを視覚的にデザインして配置できる、使いやすい GUI を提供します。全てのレポート書式設定コマンドは、フルレポートを事前に設計し、レポートを実行するのに必要な配備コードの量制限する Designer 内で使用できます。Report Designer は、アプリケーションとして実行することができ、ゼロクライアントインストールの Web ブラウザを使用してリモートからロードでき、他のアプリケーション内に統合して API を介して様々な構成で起動することもできます。

任意のソースからデータを描画

リレーショナルデータベース(JDBC)、XML ファイル、テキストファイル、および EJB から直接データを取得できます。Java オブジェクト/配列からのデータは、クラスファイルを介して取得することも、実行時に API を介してレポートに渡すこともできます。

強力なクエリツール

リレーショナルデータベースに対して実行されるレポートに対していくつかの異なるクエリツールを提供します。上級ユーザ向けに、EspressReport はクエリを完全に制御して SQL 文を書く機能を提供します。中間ユーザのために、グラフィカルクエリビルダーが用意されています。これにより、ユーザはあるポイントでクエリを作成し、QBE インタフェースをクリックすることができます。基礎となるデータベース構造を知らないユーザのために、EspressReport は、事前に配列されたフィールドと関数によるクエリを簡単に定義できる Data Views インタフェースを提供します。データビューを使用すると、管理者は、複雑なデータベース名/構造を、ユーザの好みに応じて編成されたフォルダ/フィールド階層に翻訳できます。これらのローカルの **catalogs** により、ユーザは簡単にフィールドを選択して簡単なフィルタリング条件を書くことができます。

広範なデータの可視化

豊富なビルトインチャートライブラリを提供しています。ユーザは、30 種類以上の異なる 2D および 3D チャートのいずれかにレポートデータを簡単にプロットできます。3 次元チャートは真 3D でレンダリングされ、Z 軸にデータ系列をプロットするだけでなく、パン/ズーム、回転、光源の変更も可能です。埋め込まれた Chart Designer を使用すると、チャートプロパティを視覚的に編集してからレポートに挿入できます。

柔軟な展開モデル

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

EspressReport は純粋な API ベースのデプロイメント構成により、事実上すべてのアプリケーション環境に容易に統合できます。レポートを配信するために専用サーバを必要としないため、レポートエンジンをアプリケーションサーバに直接組み込むことができます。これにより、ユーザは既存のプラットフォームのスケーラビリティを活用できます。EspressReport は、API クラスとレポートテンプレートのような簡単なデプロイメントパッケージを使用して、サードパーティアプリケーション内に完全に統合することができます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

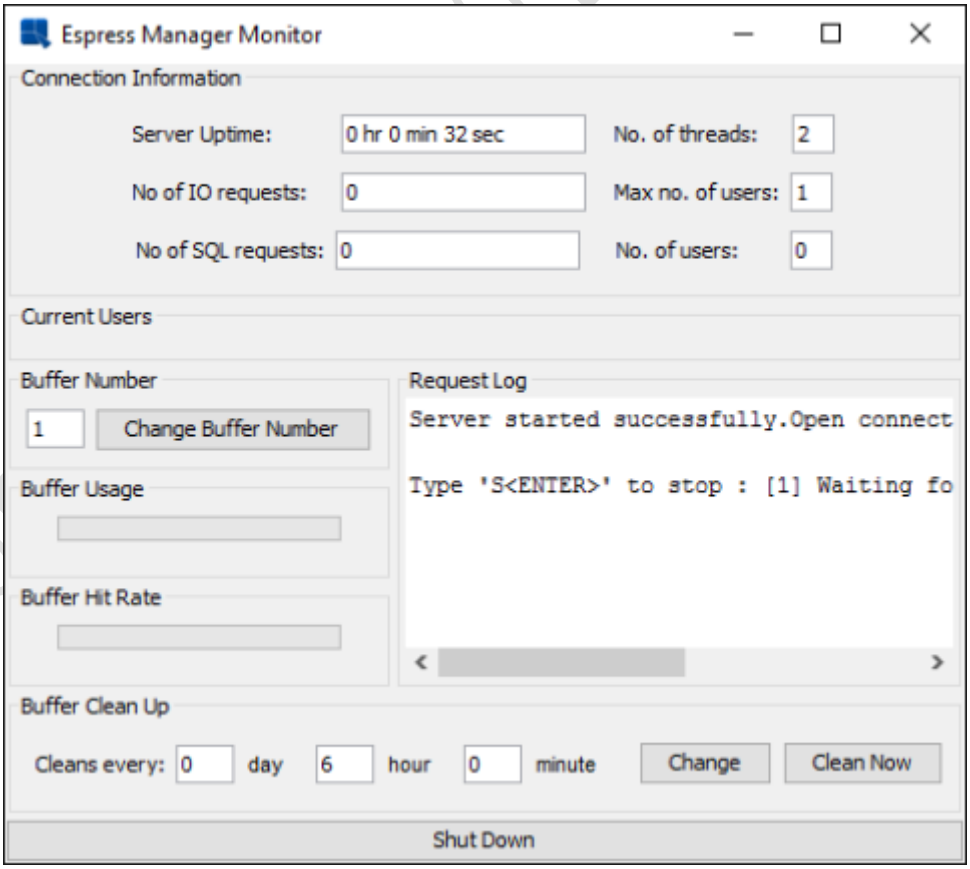
2.3 Designer のクイックスタート

このセクションでは、Report Designer の基本機能の一部を説明するための一連の短いチュートリアルが含まれています。このセクションで説明されている機能の詳細は、Designer ガイドを参照してください。

2.3.1 Report Designer の起動

この例では、Report Designer がインストールされているマシン上でローカルに Report Designer を実行することを前提としています。リモートでの実行の詳細については、Report Designer の起動を参照してください。

Report Designer を起動する前に、EspressManager が実行されている必要があります。EspressManager は、Report Designer のデータアクセスとファイル I/O を管理するバックエンドコンポーネントです（これにより、ローカルとリモートの両方で実行できます）。EspressManager を起動するには、インストールのルートディレクトリ（Unix インストールの場合は **EspressManager.sh**）で **EspressManager.bat** ファイルを実行します。デフォルトでは、EspressManager の起動時にモニタが新しいウィンドウで開きます。

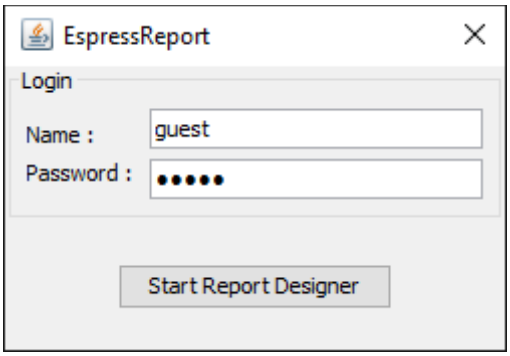


EspressManager モニタ

EspressManager を実行すると、インストールのルートディレクトリにある **ReportDesigner.bat** ファ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

イル(Unix インストールの場合は **ReportDesigner.sh**)を実行して、Report Designer を起動できます。デフォルトのユーザとしてログインするには、パスワードなしでユーザ名に **guest** と入力して使用します(ユーザの設定の詳細は、Designer ガイドの設定を参照してください)。

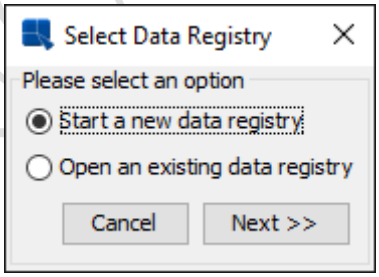


Designer ログインウィンドウ

ユーザ名とパスワードを入力して、**Start Report Designer** ボタンをクリックすると、Report Designer が新しいウィンドウで開きます。

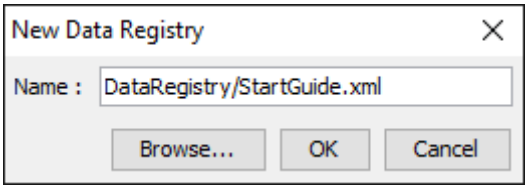
2.3.2 データソースを設定する

データソースは、データレジストリ内で維持されます。新しいデータレジストリを作成するには、まずナビゲーションバーの **File** メニューから、**New** を選択します。これにより、既存のレジストリを使用するか、新しいレジストリを作成するかを確認するダイアログが表示されます。



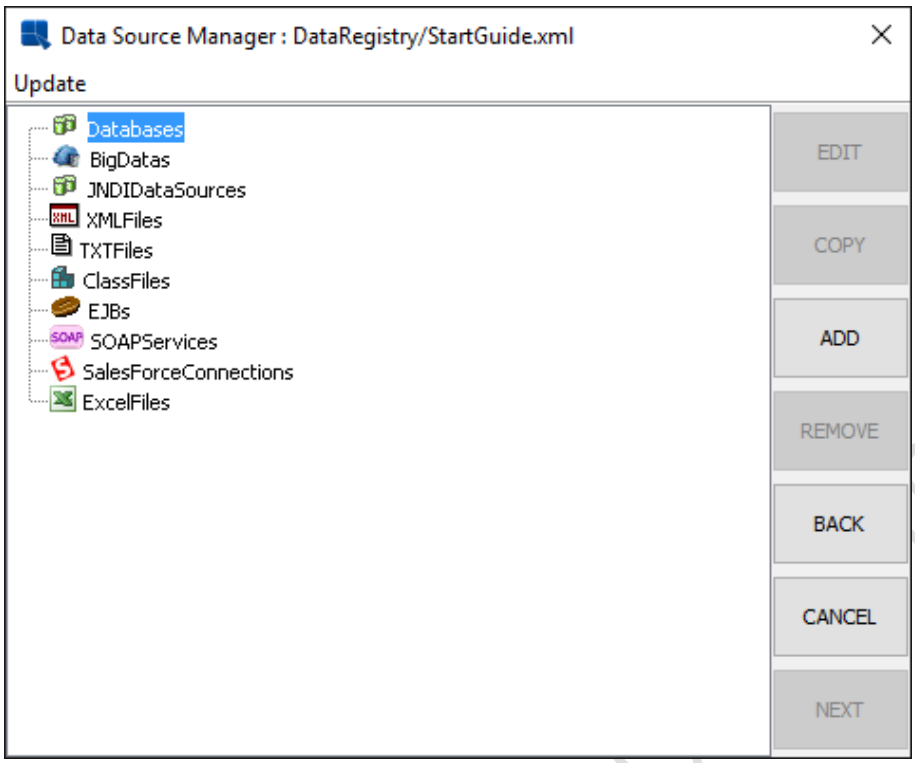
Select Data Registry ダイアログ

新しいデータレジストリを開始し、**Next** ボタンをクリックします。これにより、レジストリの名前を指定するダイアログが開きます。データレジストリの名前を入力し、**OK** ボタンをクリックします。データレジストリが新しいウィンドウで開きます。



New Data Registry ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



データレジストリウィンドウ

2.3.2.1 データベース接続のセットアップ

JDBC 準拠のデータソースに接続することができます。インストールガイドに例が記載されています。

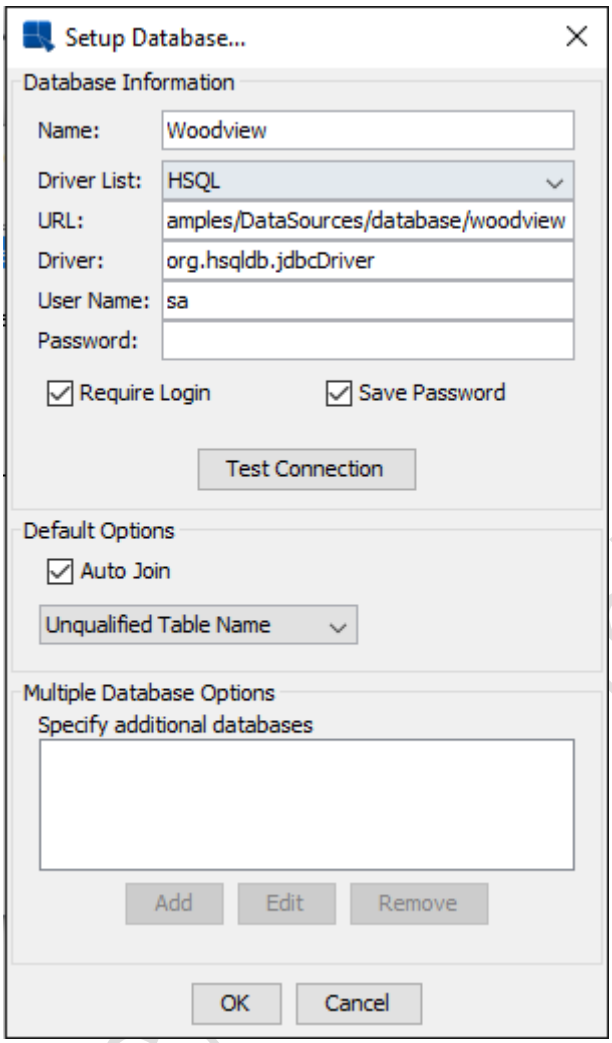
2.3.2.1.1 JDBC 接続のセットアップ

このセクションでは、EspressReport インストールに付属の Woodview HSQL データベースへの JDBC 接続を設定します。(HSQL はオープンソースの Java アプリケーションデータベースです)。

データソースマネージャから、左側のフレームの **Databases** ノードをクリックし、**Add** ボタンをクリックします。新しいデータベースの接続情報を入力するダイアログが表示されます。データベースの名前として Woodview を入力し、ドライバリストから HSQL を選択し、URL に **jdbc:hsqldb:help/examples/DataSources/database/woodview** と入力し、ドライバとして **org.hsqldb.jdbcDriver** と入力します。**Require Login** と **Save Password** の両方のボックスをクリックします。次に、ユーザ名に **sa** を入力し、パスワードを空白のままにします。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Add Database ダイアログ

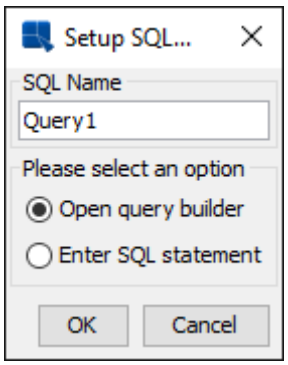
Auto Join とテーブル名のプロパティをそのままにして、**Test Connection** ボタンをクリックして、情報が正しく入力されていることを確認します。次に、**OK** ボタンをクリックして、このデータベース接続をデータソースマネージャウィンドウに追加します。そこには、Woodview 用の **Databases** の下に新しいノードがあります。

2.3.2.2 クエリを作成する

EspressReport は、データベースに照会してレポートデータを取り出すためのさまざまなインタフェースを提供します。SQL 文を実行したり、クエリビルダを使用したり、データビューを使用してエンドユーザをデータベース構造から隔離するクエリインタフェースを作成することができます。この例では、クエリビルダを使用してクエリを作成します。

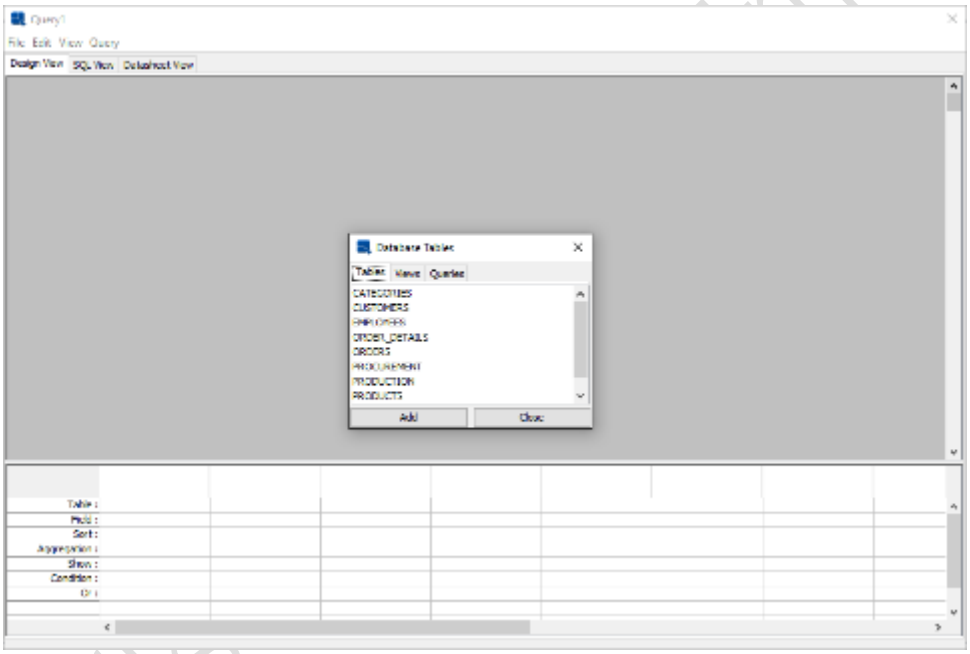
新しいクエリを作成するには、データソースマネージャの左フレームにある **Woodview** ノードをクリックして展開します。2 つのサブノードが表示されます。1 つは **Queries** と呼ばれ、もう 1 つは **Data Views** と呼ばれます。**Queries** ノードを選択し、**Add** ボタンをクリックします。クエリの名前を指定し、クエリビルダを起動するか SQL 文を入力するかを選択するダイアログが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Query Name ダイアログ

任意の名前を入力し、**Open Query Builder** を選択して、**OK** ボタンをクリックします。Query Builder が起動します。メインの Query Builder ウィンドウの上部にある Woodview のすべてのテーブルを含む別々のウィンドウが表示されます。



Query Builder ダイアログ

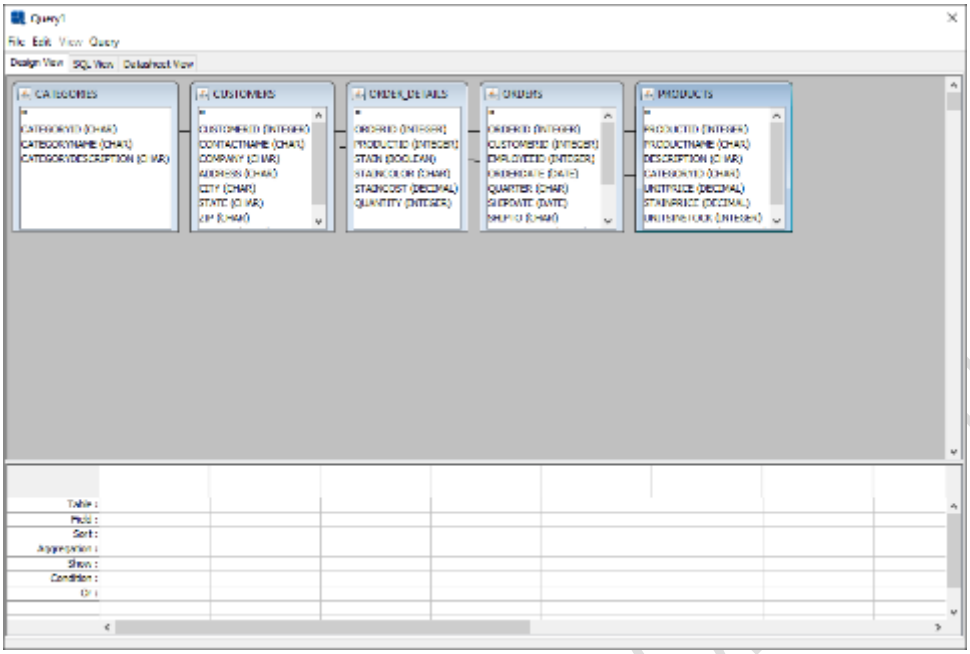
クエリにテーブルを追加するには、Tables ウィンドウでテーブルを選択し、**Add** ボタンをクリックします。また、テーブル名をダブルクリックすることもできます。2 つの方法のいずれかを使用して、クエリに以下のテーブルを追加します：

CATEGORIES
CUSTOMERS
ORDER_DETAILS
ORDERS
PRODUCTS

作業が完了したら、**Close** ボタンをクリックして **Database Tables** ウィンドウを閉じます。テーブルは、Query Builder ウィンドウの上半分に表示されます。テーブルのさまざまなフィールドをつなぐ結

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

合線が表示されます。



テーブルを持つ Query Builder

クエリにフィールドを追加するには、テーブルウィンドウのフィールドをダブルクリックします。また、クエリビルダウィンドウの下部(QBE)部分の **Table** フィールドと、**Field** フィールドをダブルクリックし、ドロップダウンメニューからテーブルとフィールドを選択することもできます。どちらの方法を使用しても、以下のフィールドをクエリに追加します：

ORDERID (INTEGER) from ORDERS

COMPANY (CHAR) from CUSTOMERS

REGION (CHAR) from CUSTOMERS

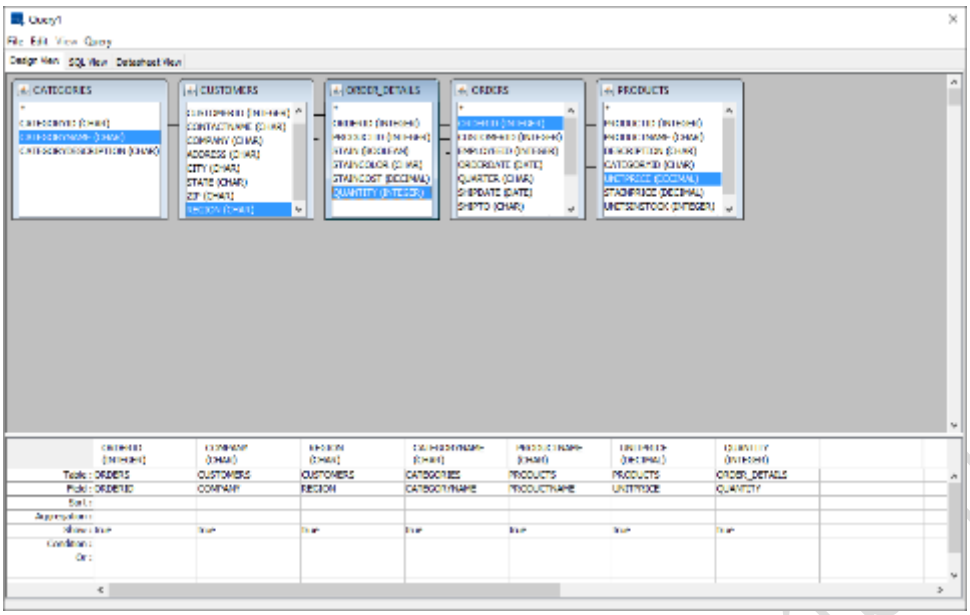
CATEGORYNAME (CHAR) from CATEGORIES

PRODUCTNAME (CHAR) from PRODUCTS

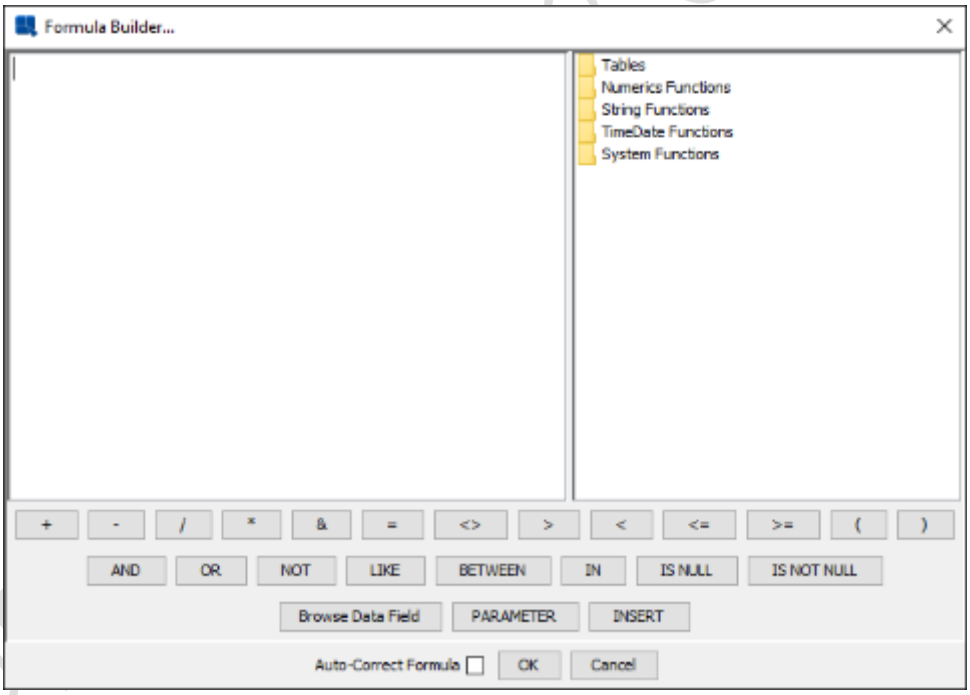
UNITPRICE (DECIMAL) from PRODUCTS

QUANTITY (INTEGER) from ORDER_DETAILS

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



空白にする必要がある 8 番目の列で、**Field** フィールドを右クリックし、ポップアップメニューから **Build** を選択します。計算された列を作成できるようにする、数式ビルダインタフェースが開きます。

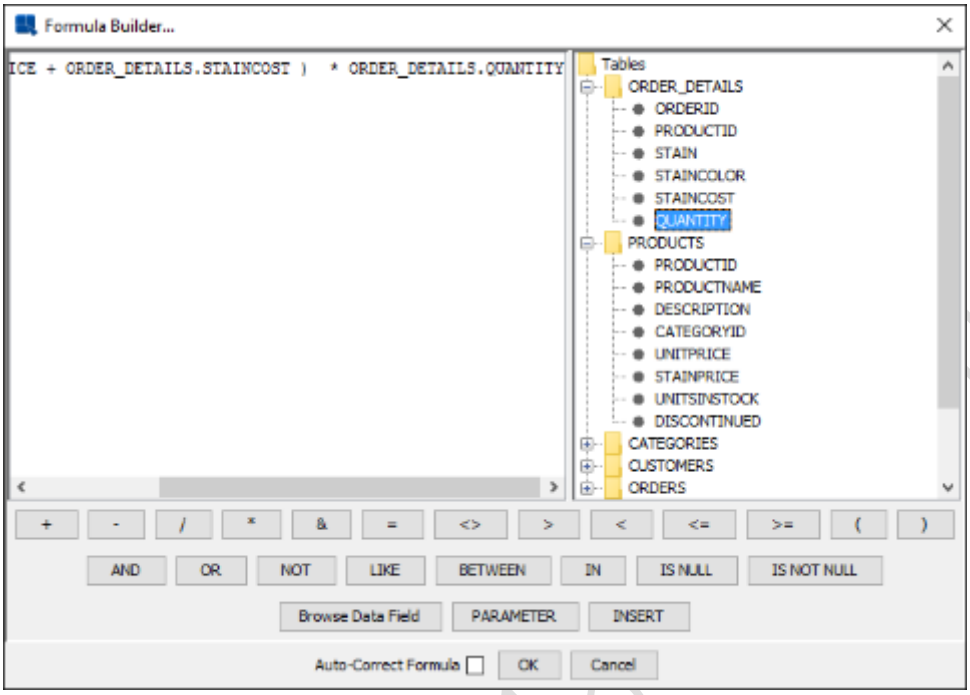


数式ビルダウィンドウ

まず、**left parenthesis** ボタンをクリックして列を作成します。次に、**Tables** フォルダをダブルクリックして、クエリ用に選択した各テーブルに 5 つの node が展開します。テーブルフォルダを開くと、そのテーブルのすべての列フィールドが一覧表示されます。**PRODUCTS** フォルダを開き、**UNIRPEICE** を選択して **INSERT** ボタンをクリックします。次に、追加(+)ボタンをクリックします。次に、**ORDER_DETAILS** 表から **STAINCOST** を挿入します。次に、**right parenthesis** ボタンをクリックしてから、乗算（*）ボタンをクリックします。最後に、**ORDER_DETAILS** 表から **QUANTITY** を挿入します。完成した式は、以下のようになります。

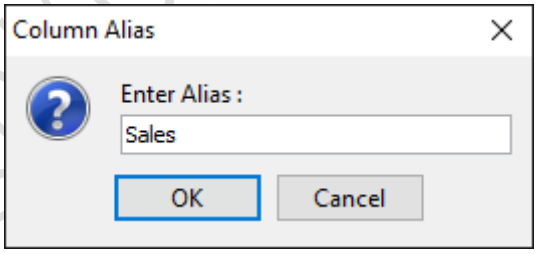
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

(PRODUCTS.UNITPRICE + ORDER_DETAILS.STAINCOST) * ORDER_DETAILS.QUANTITY



数式を含む数式ビルダウィンドウ

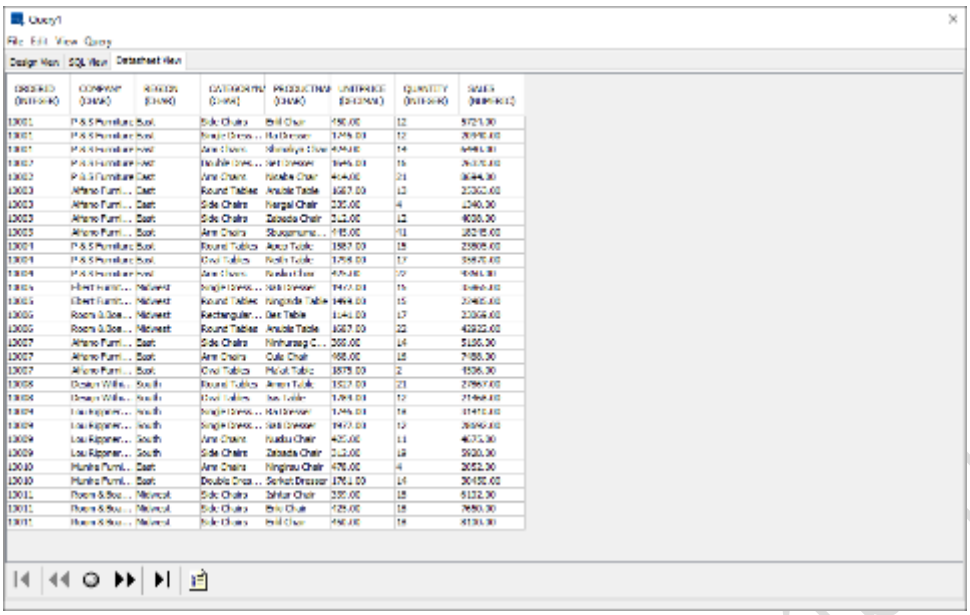
OK ボタンをクリックすると、作成された列がクエリに追加されます。次に、列に別名を付けます。列を右クリックし、ポップアップメニューから **Alias** を選択します。列の別名の入力を求めるウィンドウが表示されます。**SALES**（引用符は不要）を入力し、**OK** ボタンをクリックします。



Column Alias ダイアログ

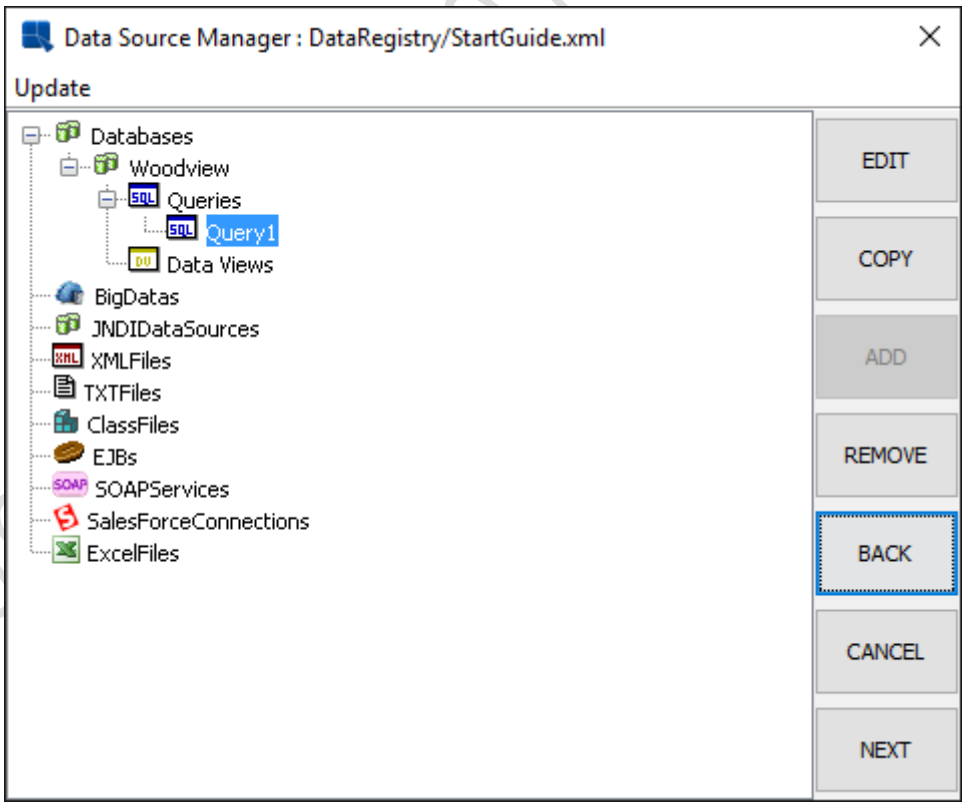
OK ボタンをクリックすると、Query Builder に列名の変更が表示されます。Query Builder の **Datasheet View** タブをクリックします。クエリが実行され、クエリ結果の最初の 30 個のレコードが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Query Builder のデータシートビュー

これで、クエリの設計が完了したので、File メニューから **Done** を選択して変更を保存します。これにより、Query Builder ウィンドウが閉じ、Data Registry Manager ウィンドウに戻ります。今作成したクエリの **Queries** の下にノードがあります。



クエリ付きデータソースマネージャ

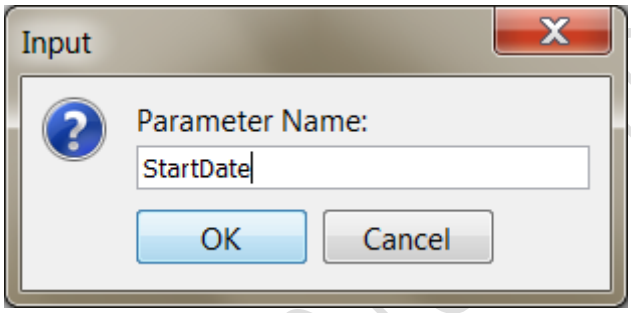
2.3.2.2.1 クエリパラメータの追加

レポートクエリを簡単にパラメータ化でき、実行時にレポートデータを動的にフィルタリングすることができます。今セクションでは、クエリを作成するで作成したクエリに、パラメータを追加します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

作成したクエリを開くには、そのクエリを選択し、データソースマネージャの **Edit** ボタンをクリックします。次に、**OK** ボタンをクリックして名前を確定します。クエリは、Query Builder で再度開きます。**Close** ボタンをクリックして、Database Tables ウィンドウを閉じ、Query Builder の下の ORDERID 列の **Condition** フィールドを右クリックし、ポップアップメニューから **Build** を選択します。これにより、式ビルダが表示され、クエリの条件を作成出来ます。

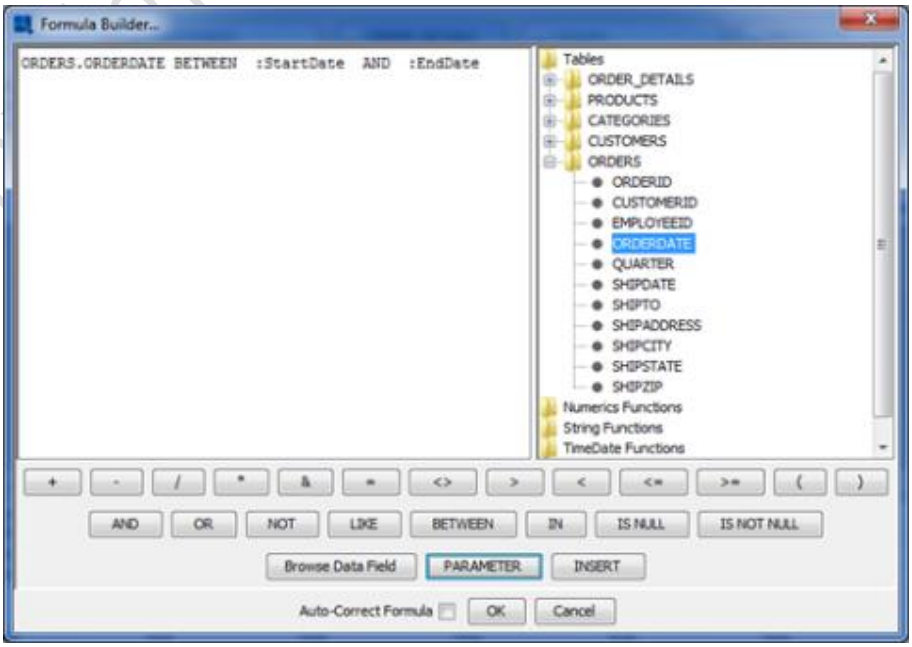
数式ビルダ内で、**Tables** フォルダをダブルクリックして展開します。**ORDERS** ノードを展開し、**ORDERDATE** フィールドをダブルクリックします。次に、**BETWEEN** ボタンをクリックし、**PARAMETER** ボタンをクリックします。これにより、クエリパラメータの名前を指定するダイアログが表示されます。



Parameter Name ダイアログ

パラメータ名に **StartDate** と入力し、**OK** ボタンをクリックします。パラメータがクエリに追加されます。次に、**AND** ボタンを押してから、**PARAMETER** ボタンをもう一度クリックします。2 番目のパラメータ名として **EndDate** を入力します。完成した状態は、以下のようになります。

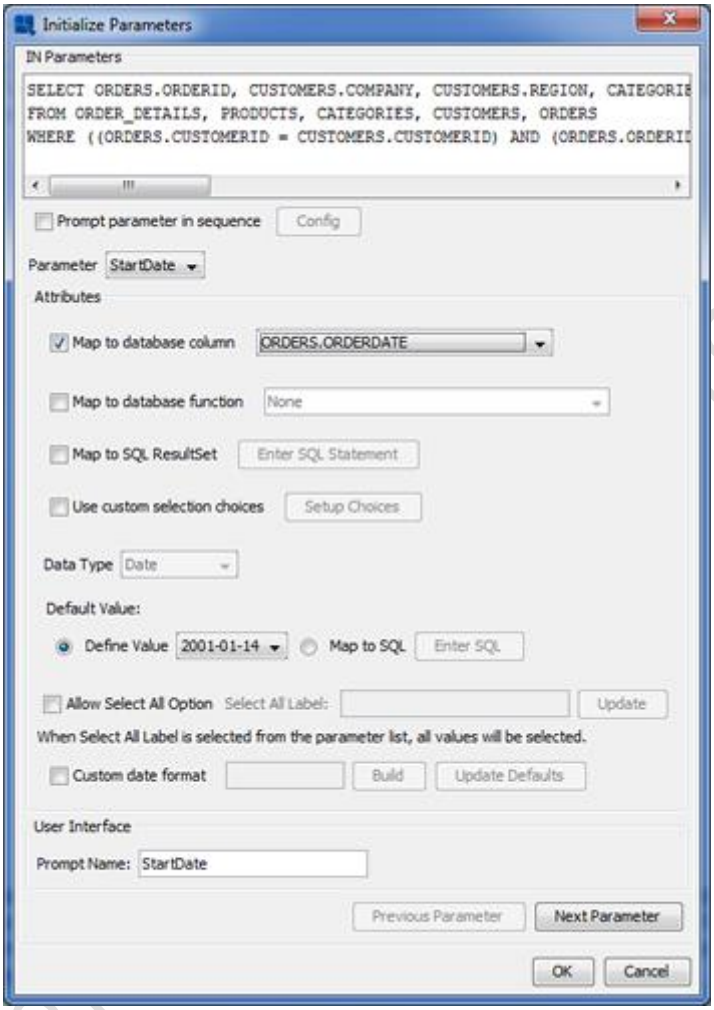
ORDERS.ORDERDATE BETWEEN :StartDate AND :EndDate



条件付き数式ビルダ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

OK ボタンをクリックして数式ビルダを閉じ、Query Builder ウィンドウに戻ります。次に、**Datasheet View** タブをクリックします。クエリに 2 つのパラメータを追加したばかりなので、初期化ダイアログが表示され、クエリパラメータにいくつかのプロパティを指定するように求められます。



Initialize Parameter ダイアログ

このウィンドウで、**Map to a database column** をクリックし、ドロップダウンメニューから、**ORDERS.ORDERDATE** を選択します。これにより、**Default Value** と **Data Type** オプションが自動的に設定されます。次に、**Prompt Name** に **Start Date** と入力し、**Next Parameter** ボタンをクリックして、**EndDate** パラメータを同じ列にマップします。**Define Value** ドロップダウンメニューをクリックして、終了日を選択できます。開始日から十分離れた日付を選択すると、既定では複数のレコードを操作できます（これにより、レポートデザインが簡単になります）。**Prompt Name** を **End Date** に変更します。

全てのオプションを指定したら、**OK** ボタンをクリックして初期化ウィンドウを閉じます。新しいダイアログが表示され、結果セットをフィルタリングする日付範囲を選択するように指示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Parameter Selection ダイアログ

必要な開始日と終了日を選択し、**OK** ボタンをクリックします。フィルタされた結果がデータシートウィンドウに表示されます。ここで、File メニューの **Done** をクリックして、クエリに加えた変更を保存します。

2.3.2.3 データビューを作成する

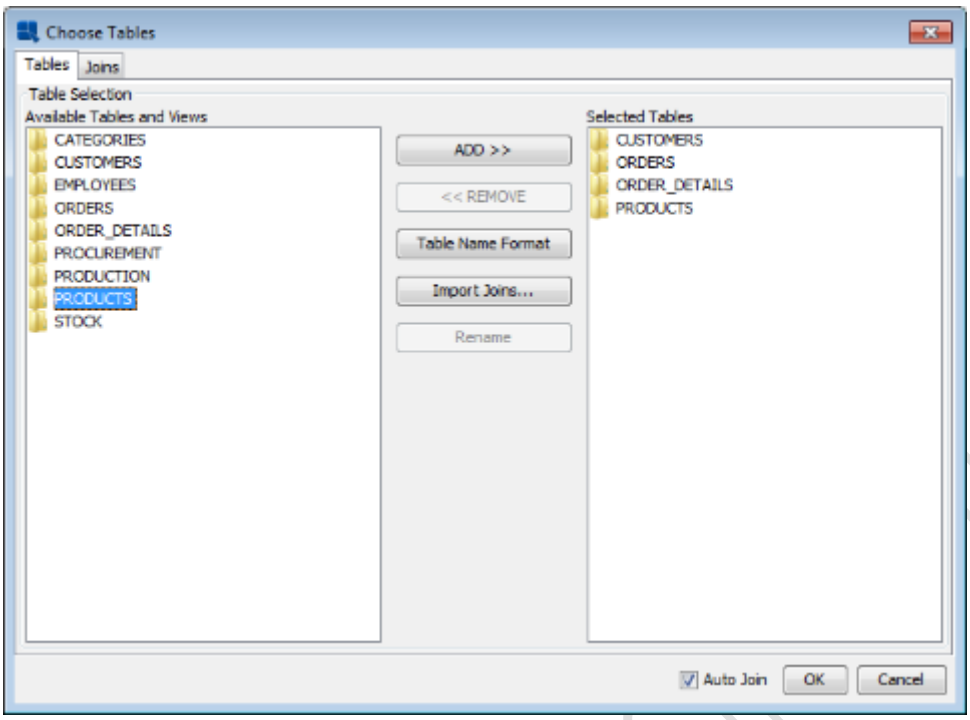
EspressReport のユニークな機能は、データビューを作成する機能です。データビューは、管理者がテーブルとフィールドのグループを事前に構成できるローカルスキーマ/ビューです。これにより、エンドユーザはフィールドを選択し、シンプルな条件を定義してクエリを作成することができます。データビューを作成するには、**Woodview** の下にある **Data Views** ノードを選択し、**Add** をクリックします。

これにより、使用するデータベーステーブルを選択するように求める、新しいダイアログボックスが開きます。以下の表を選択し、**ADD**>>ボタンをクリックして、**Selected Tables** ウィンドウに追加します：

CUSTOMERS
ORDERS
ORDER_DETAILS
PRODUCTS

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Data View Table ダイアログ

次に、**Joins** タブをクリックします。Query Builder のようなテーブルの表現が表示されます。テーブル間の自動結合線が表示されます。このウィンドウは、必要に応じてテーブルを結合したり、自動結合を変更するために使用できます。**OK** ボタンをクリックして、テーブルの選択を確定します。次のウィンドウでは、ビューのフィールドを選択してグループ化できます。ウィンドウの上部には、ビューの名前を指定できます。**Invoicing** という名前を付けます。

次に、**CUSTOMERS** フォルダをダブルクリックして、そのテーブルのフィールドを表示します。次のフィールドを選択し、**ADD>>**ボタンをクリックして追加します：

CONTACTNAME
COMPANY
ADDRESS
CITY
STATE
ZIP

以下のように、他のテーブルのフィールドを追加します。

ORDERS:
- **ORDERDATE**
- **SHIPDATE**
- **SHIPTO**
- **SHIPADDRESS**
- **SHIPCITY**

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

SHIPSTATE
SHIPZIP

ORDER DETAILS:
ORDERID
STAIN
STAINCOLOR
QUANTITY

PRODUCTS:
PRODUCTNAME
UNITPRICE
STAINPRICE

Add Heading ボタンをクリックします。プロンプトで、**Customer Info** という名前を指定します。同様に 2 つの見出しを追加します。1 つは **Shipping Info**、もう 1 つは **Order Info** です。それらが作成されたら以下のフィールドを選択します（**CTRL + クリック**、複数選択の場合は **SHIFT + クリック**）：

CONTACTNAME
COMPANY
ADDRESS
CITY
STATE
ZIP

フィールドが選択されたら、**Group Fields** ボタンをクリックし、ドロップダウンリストから **Customer Info** を選択します。フィールドは、その見出しの下に移動します。次に、以下のフィールドを同じ方法で選択します：

SHIPDATE
SHIPTO
SHIPADDRESS
SHIPCITY
SHIPSTATE
SHIPZIP

これまでと同じ方法で、これらの項目を **Shipping Info** グループに追加します。次に、以下のフィールドを選択します：

ORDERDATE
ORDERID

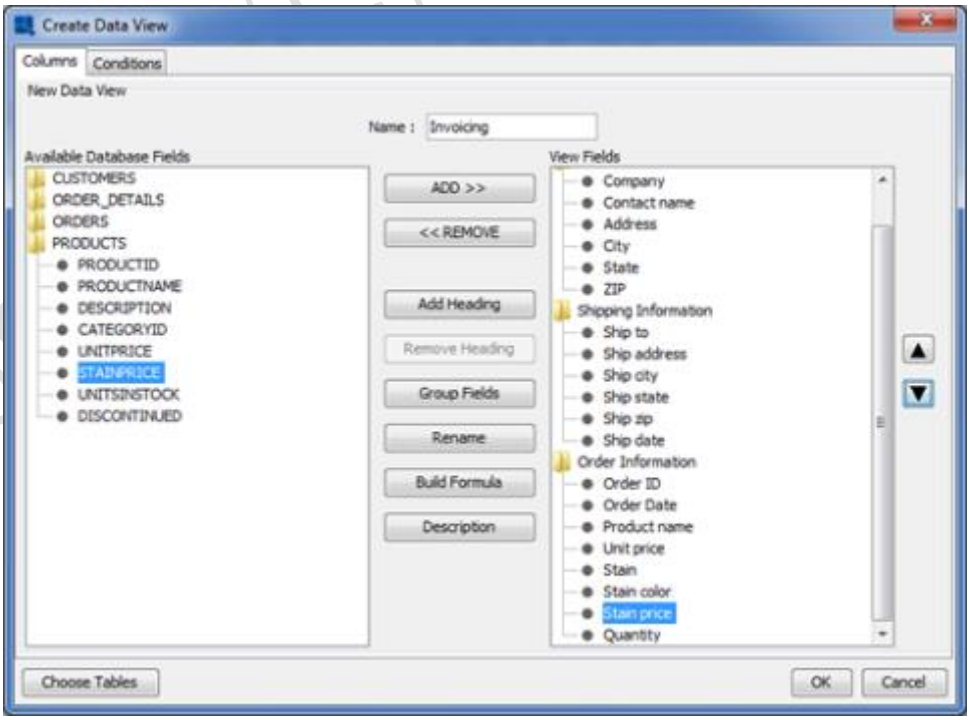
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

STAIN
STAINCOLOR
QUANTITY
PRODUCTNAME
UNITPRICE
STAINPRICE

これらのフィールドを、**Order Info Info** グループに追加します。次に、右側の **CONTACTNAME** フィールドを選択し、**Rename** ボタンをクリックします。ダイアログで、**Contact name** を指定します。適切な名前を付けるために、すべてのフィールドでこれを繰り返します。

次に、右側の **Order ID** フィールドを選択し、**up arrow** ボタンをクリックして、フィールドを **Order Info** 見出しの一番上に移動します。矢印を使用して、**Order Info** 見出しの項目を以下の順序で並び替えます：

Order ID
Order Date
Product name
Unit Price
Stain
Stain Color
Stain Price
Quantity



データビューフィールドウィンドウ

次に、フィールドウィンドウの **OK** ボタンをクリックしてビューを保存します。データソースマネージ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

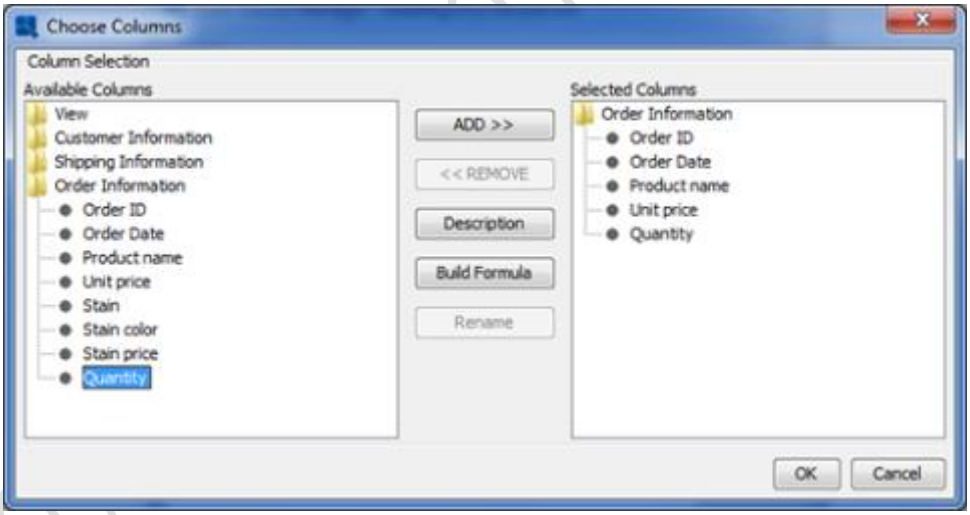
ャのデータビューの下に、新しいノードとして保存されます。

2.3.2.3.1 データビューのクエリ

データビューが作成されたので、ビューに対してクエリを記述することができます。これにより、ユーザはデータベースの基本構造を知らなくても照会を作成することができます。管理者は、ユーザがアクセスできるデータベース要素を制限することもできます。このセクションでは、データビューを作成するで作成したデータビューのクエリを作成します。

データソースマネージャで、**Invoicing** データビューを選択します。次に、**Next** ボタンをクリックします。これは、ビューからフィールドを選択するように促すダイアログを開きます。フィールドを選択するには、見出しをダブルクリックして展開します。それらを選択し、**Add** ボタンをクリックして、以下のフィールドをクエリに追加します：

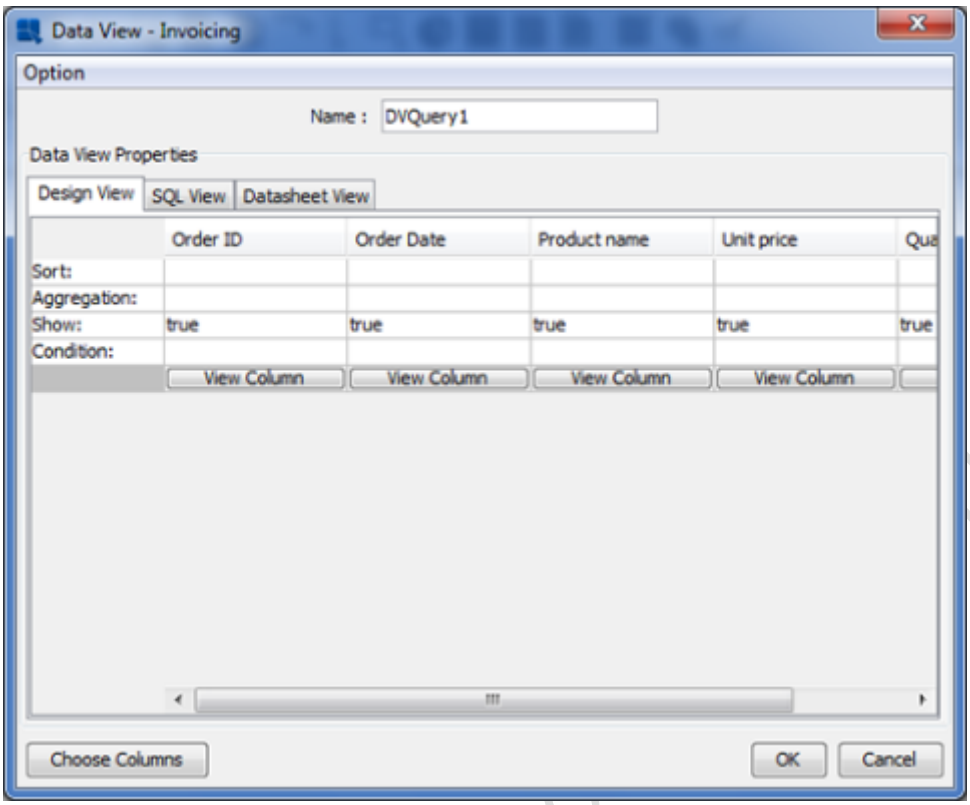
Order ID
Order Date
Product name
Unit Price
Quantity



Data View Query Field Selection ダイアログ

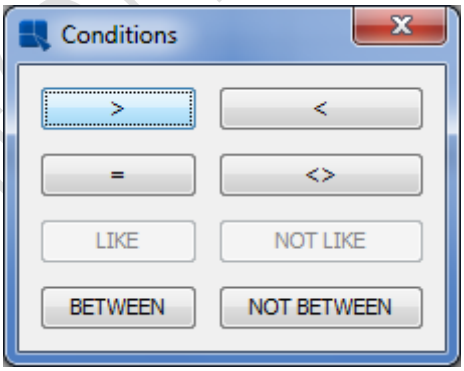
フィールドの追加が終了したら、**OK** ボタンをクリックします。これにより、条件の設定、グループ化、およびクエリの順序付けを可能にする、新しいウィンドウが表示されます。Query Builder 同様に、このウィンドウでは、**Datasheet View** タブでクエリ結果をプレビューすることもできます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Data View Conditions ダイアログ

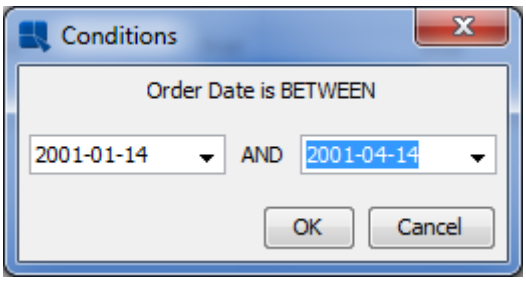
まず、上部にあるスペースにクエリの名前を指定します。Order Date 列の **Condition** フィールドをダブルクリックします。これにより、フィールドの条件を指定できるダイアログが表示されます。



Specify Condition ダイアログ

Between ボタンをクリックします。新しいダイアログが表示され、結果をフィルタリングする開始日と終了日を指定するように求められます。最初の日付として **2001-01-14** を、第 2 の日付として **2003-12-09** を指定します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Specify Condition Fields ダイアログ

OK をクリックしてダイアログを閉じ、条件を追加します。条件ウィンドウに戻ります。**Datasheet View** タブをクリックすると、クエリをプレビューできます。

メインウィンドウで **OK** をクリックすると、指定した名前でクエリが保存されます。レポートウィザードの最初の手順に進みます。このダイアログで **Cancel** をクリックしてインタフェースを閉じ、**New File** ボタンをクリックして、データソースマネージャに戻ります。これで、**Invoicing** データビューの下にクエリ用の新しいノードが作成されます。

2.3.3 レポートマッピング

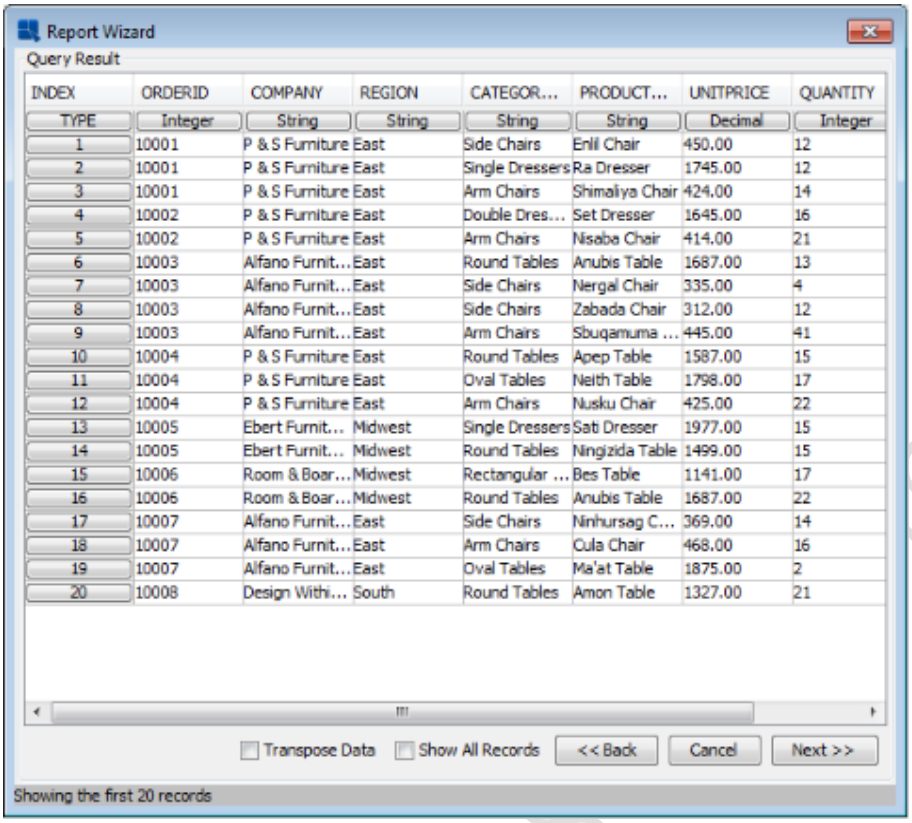
このセクションでは、データソースのデータをレポート構造にマップするさまざまな方法について説明します。このセクションでは、クエリを作成するで作成したクエリから結果セットを取り出し、それを使用して EspressReport がサポートするさまざまなレイアウトのレポートを生成します。さまざまなレイアウトとマッピングオプションの詳細については、Designer ガイドのレポートタイプとデータマッピングを参照してください。

2.3.3.1 シンプルカラムナレイアウト

シンプルカラムナレイアウトは、レポートマッピングの最も簡単な種類です。データソースの列は、グループ化や分割なしでレポート内の直線的な表に描画されます。

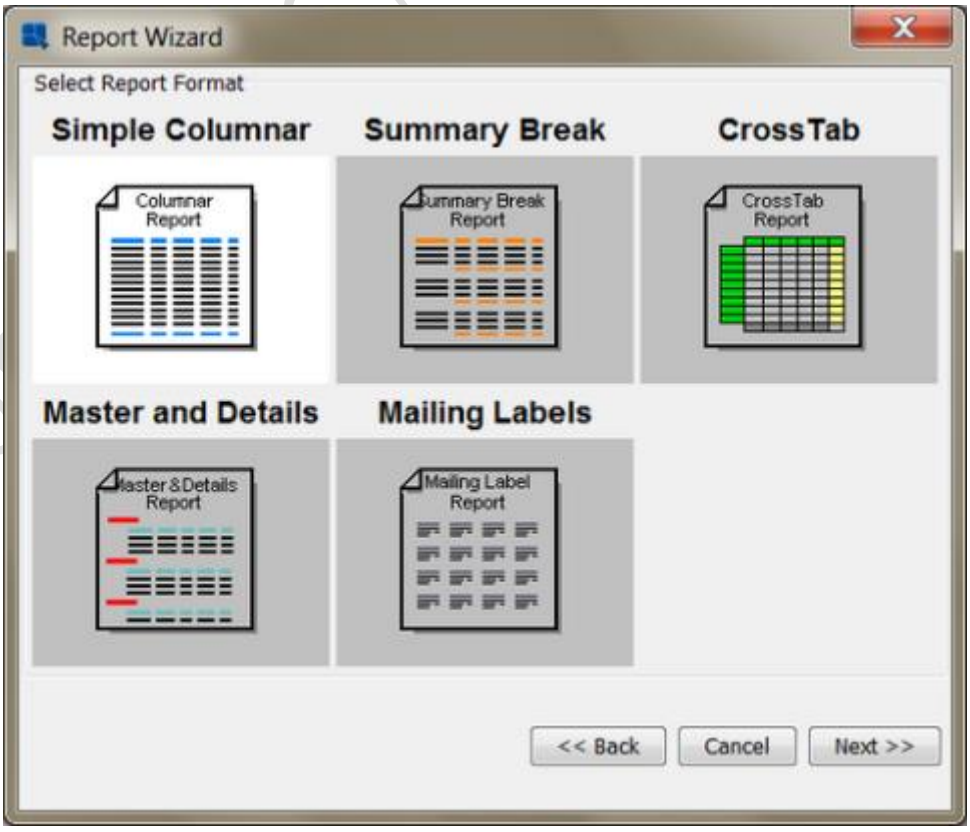
レポートの作成を開始するには、データレジストリを開き、**Queries** ノードを選択します。**Next** ボタンをクリックします。新しいウィンドウが開き、クエリの結果を含むテーブルが表示されます（最初の 20 レコードのみ）。クエリにはパラメータが含まれているため、最初はパラメータが初期化されたときに指定された既定値で実行されます

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



クエリ結果画面

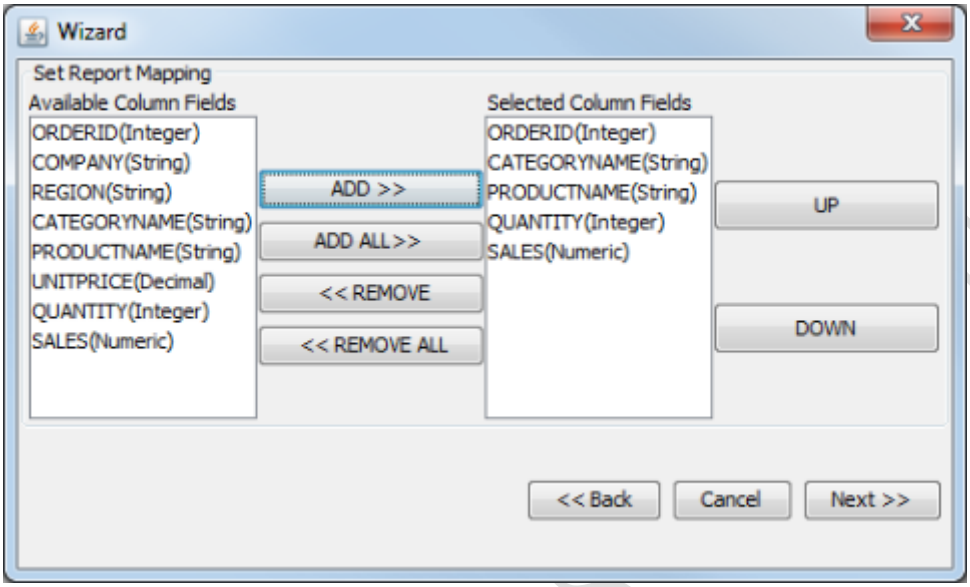
レポートウィザードを実行するには、**Next** ボタンをクリックします。これにより、どのレポートレイアウトオプションを使用するかを尋ねるダイアログが表示されます。



Select Report Layout ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

このダイアログで、レイアウトとして **Simple Columnar** を選択し、**Next** ボタンをクリックします。レポートウィザードの次のステップでは、レポートで使用するクエリの列を選択、また列の順序を並べ替えることができます。(列の順序は、シンプルカラムナレイアウトにとって特に重要ではありませんが、選択されたマッピングオプションに応じて、他のレイアウトに大きな影響を与える可能性があります)。



Column Selection/Ordering ダイアログ

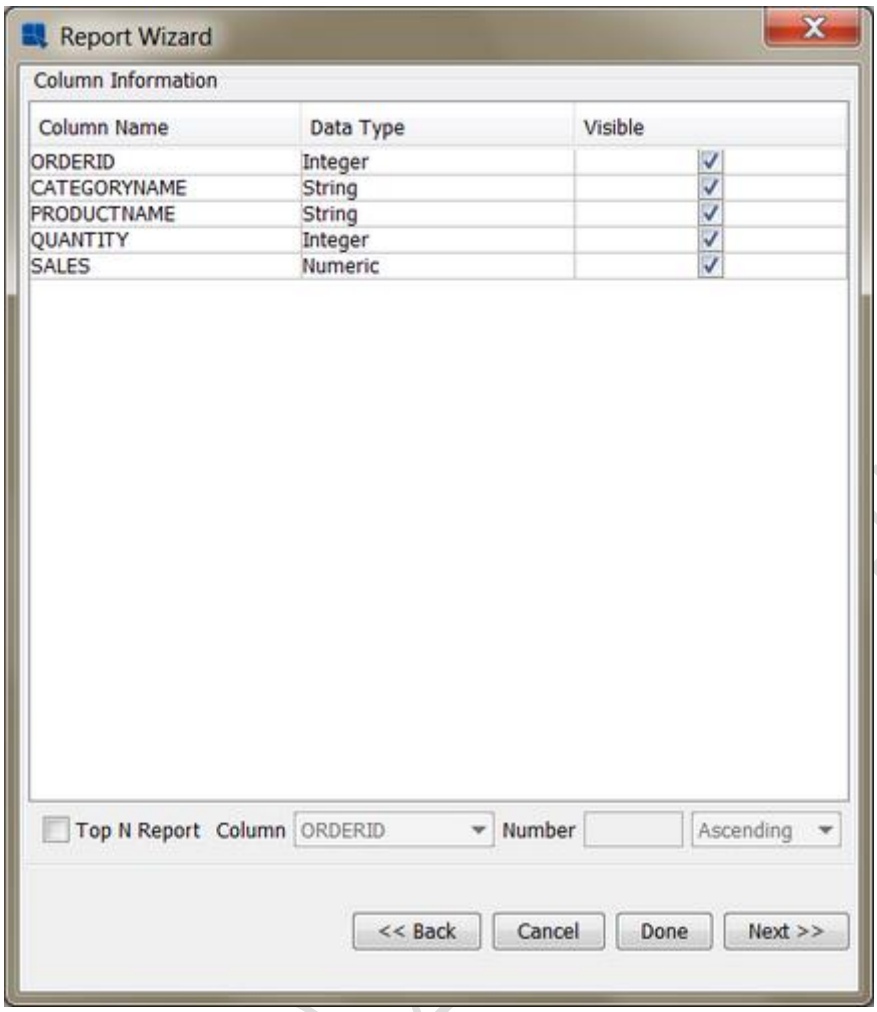
このダイアログで、レポートの以下のフィールドを選択します。

ORDERID
CATEGORYNAME
PRODUCTNAME
QUANTITY
SALES

フィールドを選択したら、**Next** をクリックしてウィザードを続行します。次のダイアログは、データマッピングダイアログです。ここで、データソースのフィールドを選択したレポートレイアウトにマップする方法を選択できます。これはシンプルカラムナレイアウトであるため、オプションは、列を表示するかどうか、上位 N プレゼンテーションを生成するかどうかのみです。

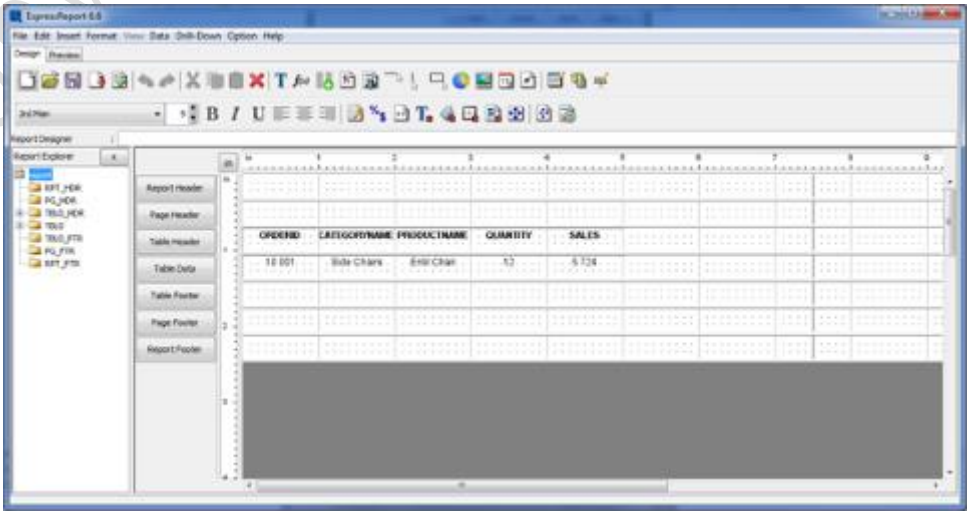
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Data Mapping ダイアログ

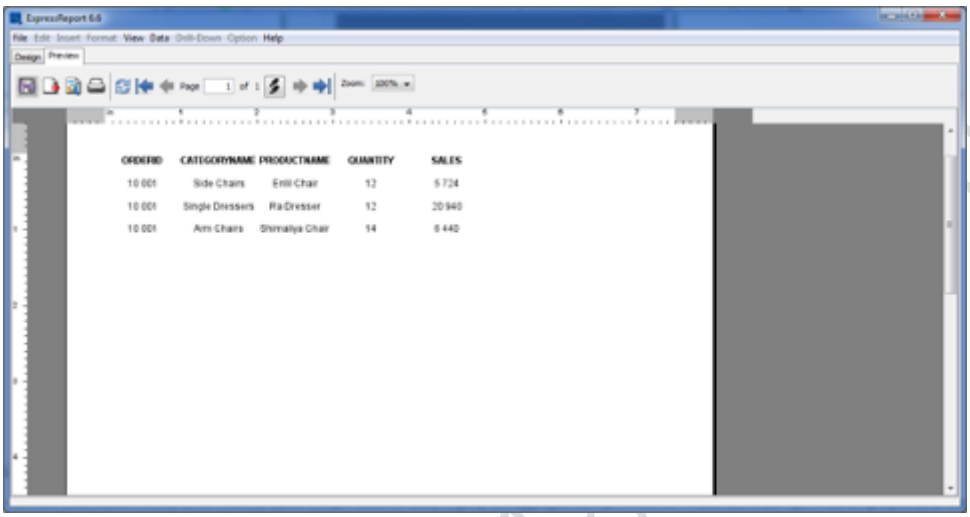
このダイアログで、すべての列を表示したままにし、上位 N オプションを選択しないように選択します。（上位 N オプションのプレゼンテーションの詳細については、Designer ガイドの上位 N レポートを参照してください）。次に、**Done** ボタンをクリックします。（このセクションの後半で説明するウィザードには、オプションの追加手順がいくつかあります）。これにより、Report Designer のインタフェースが、レポートの書式なしのバージョンとともに表示されます。



デザインウィンドウでのシンプルカラムナレポート

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ウィンドウの左隅にある **Preview** タブをクリックし、**Use Live Data** を選択します。パラメータ選択ダイアログが表示され、レポートをフィルタする開始日と終了日を選択するように求められます。2 つ以上のレコードを取得するのに十分な範囲を指定し、**OK** ボタンをクリックします。レポートの出力が表示されます。シンプルカラムナレイアウトがどのようにグループ化、並び替え、またはサマリを使わずに列をレポートに直接配置するかに注目してください。



シンプルカラムナレポートのプレビュー

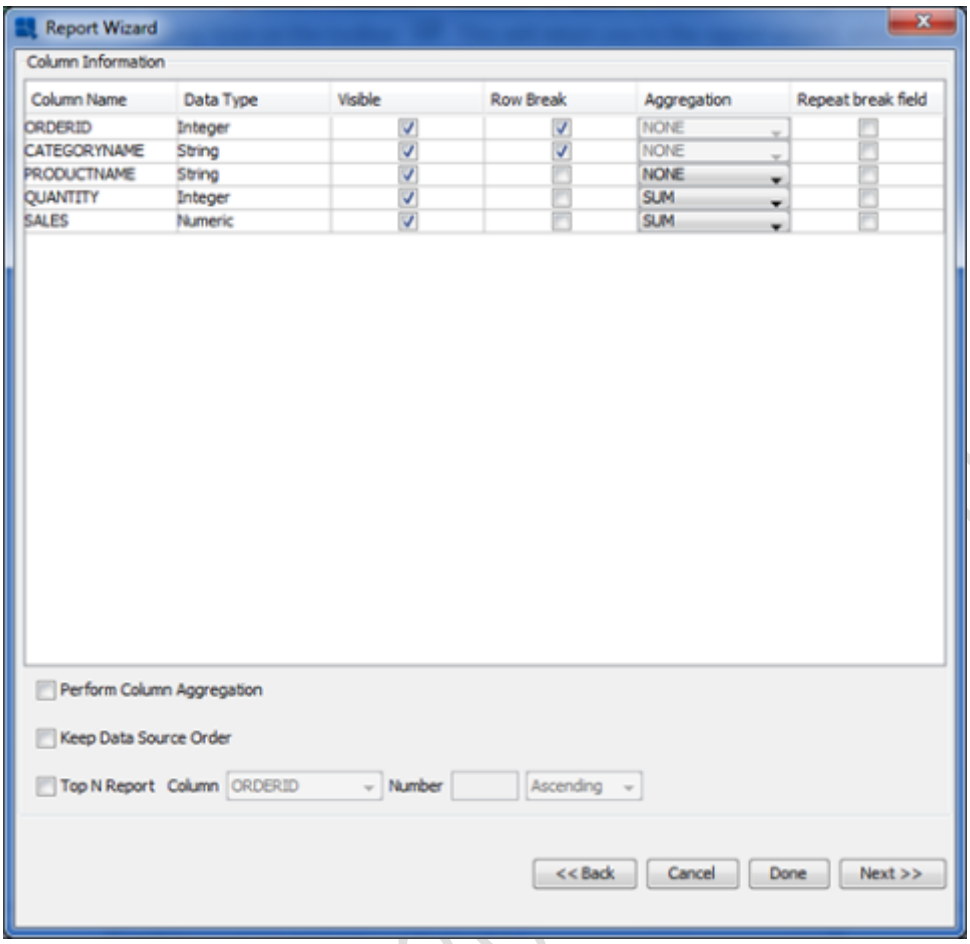
2.3.3.2 サマリブレークレイアウト

サマリブレークレイアウトは、レポート列をグループ化して集計する機能を追加する点を除いて、カラムナレイアウトに似ています。サマリブレークレポートは、少なくとも 1 つの列でグループ化する必要があります。シンプルカラムナレイアウトを使用して、それをサマリブレークレイアウトに変換します。

design タブに移動し、ツールバーの **Change Data Mapping** アイコン を選択します。これにより、レポートウィザードが表示され、別のレポートレイアウトが選択されます。このダイアログから、**Select Report Format** ウィンドウに戻るまで **Back** ボタンを 2 回押します。

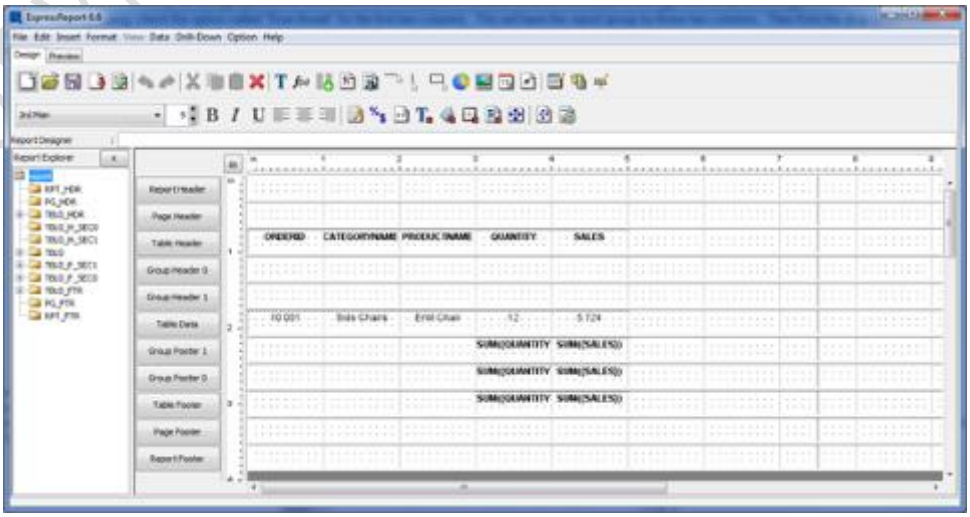
レポートのレイアウトの種類を **Summary Break** に変更し、**Next** ボタンをクリックします。次の画面では、同じ列の選択を維持し、**Next** ボタンをもう一度クリックしてデータマッピングウィンドウに移動します。このウィンドウには、カラムナレイアウトよりも多くのオプションがあることがわかります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



サマリブレークレイアウトのデータマッピング画面

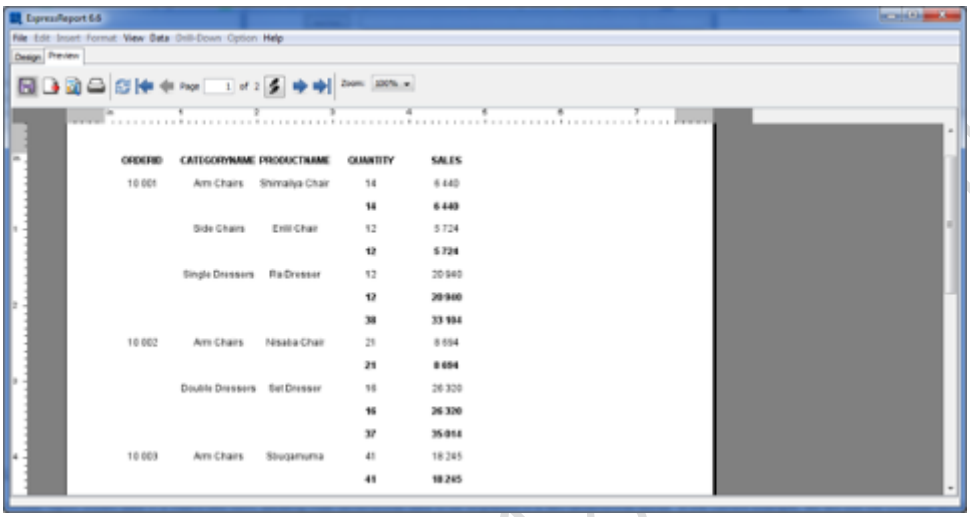
データマッピングダイアログで、最初の 2 つの列に対して **Row Break** というオプションをチェックします。これにより、2 つの列でレポートがグループ化されます。**Aggregation** のドロップダウンメニューから、**QUANTITY** 列と **SALES** 列の **SUM** を選択します。また、シンプルカラムナレイアウトから書式を引き継ぐ必要がないので、**Apply Template** オプションのチェックを外します。オプションの指定が終わったら、**Done** ボタンをクリックし、次の警告メッセージで **Yes** をクリックして続行します。新しいマッピングが有効になった Report Designer に戻ります。



デザインウィンドウでのサマリブレークレポートの中断

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Designer では、レポートにネストされたグループ化のレベルが 2 つあります。したがって、それぞれに対応するグループヘッダーとフッターセクションが存在するようになりました。レポートセクションとその動作の詳細については、デザイナーガイドのレポートセクションを参照してください。**Preview** タブをクリックして、レポートをプレビューします。繰り返しますが、パラメータ値を指定するよう求められます。プレビューウィンドウでレポートが表示されたら、データがカテゴリ名とオーダーID でどのようにグループ化されているかを確認し、グループごとに中間集計を計算します。



サマリブレークレポートのプレビューを中断する

2.3.3.3 クロス集計レイアウト

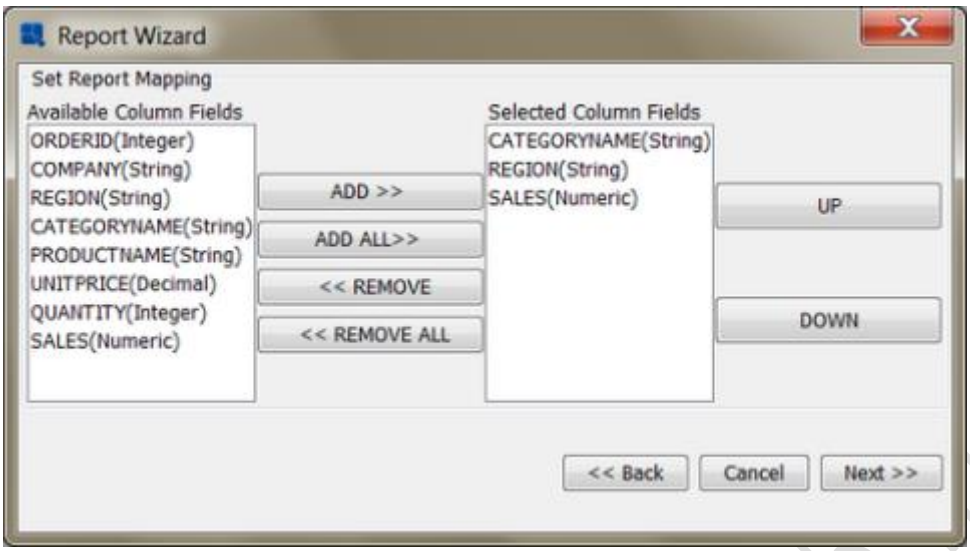
クロス集計レポートでは、マトリックス状のデータが表示され、多次元データを 2 次元レイアウトで表示することができます。現在の例では、クロス集計レイアウトを使用して、販売カテゴリを製品カテゴリと地域別に分類します。

Design タブに移動し、**Change Data Mapping** アイコンを選択します。レポートウィザードが再度開いたら、レイアウト選択画面に戻るまで **Back** ボタンをクリックします。この画面で、**CrossTab** レイアウトを使用するように選択し、**Next** をクリックします。列選択/注文画面（ウィザードの次に）で、選択内容を以下のように変更します：

CATEGORYNAME
REGION
SALES

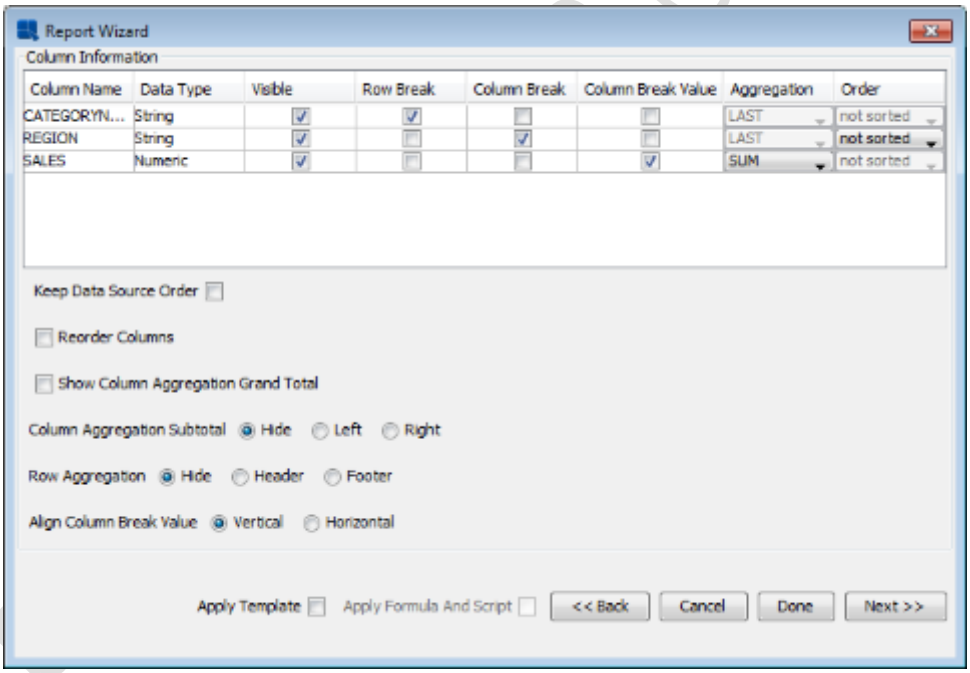
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



クロス集計レポートの列選択

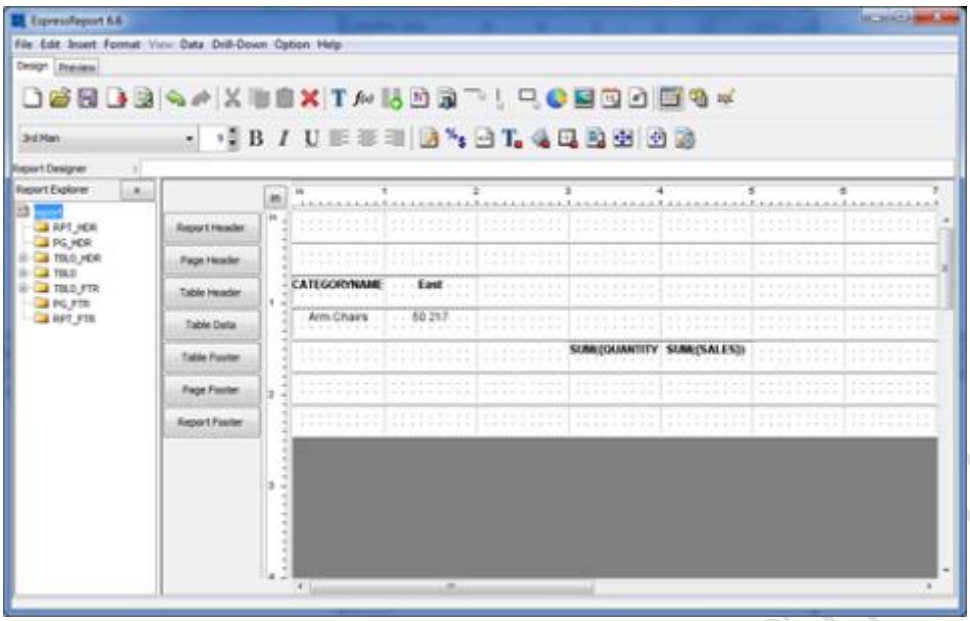
正しい順序で列を指定したら、**Next** ボタンをクリックして、クロス集計レイアウトのデータマッピングオプションを表示します。



クロス集計レイアウトのデータマッピング画面

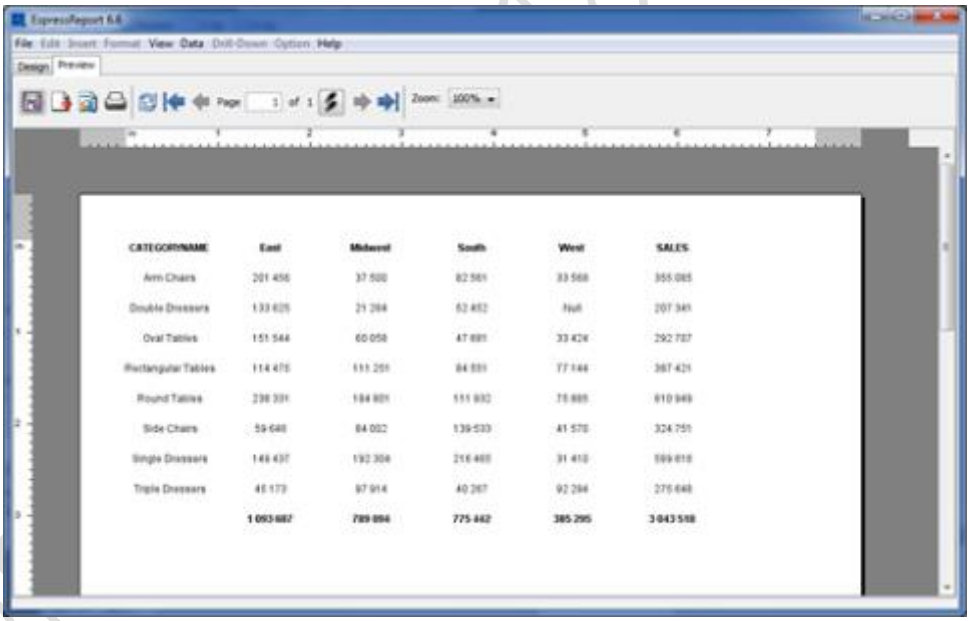
CATEGORYNAME 列の場合は、**Row Break** とマークされているオプションを選択します。これにより、各別のカテゴリ名ごとにレポートデータに行が作成されます。次に、**Region** 列の **Column Break** オプションを選択します。その後、各別の領域ごとにレポートデータに列が作成されます。**Order** オプションを **not sorted** として残します。最後に、**Sales** 列の **Column Break Value** オプションを選択し、**Aggregation** メニューをクリックして **SUM** を選択します。これにより、レポートの各カテゴリと地域の合計売上が得られます。オプションの指定が完了したら、**Done** ボタンをクリックし、警告メッセージで **Yes** をクリックして Report Designer に戻ります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



デザインウィンドウでのクロス集計レポート

ご覧のとおり、レポート列は各地域ごとに生成され、縦（列）横（行）の両方で自動的に合計されています。次に、**Preview** タブをクリックしてクロス集計レイアウトをプレビューします。



クロス集計レポートのプレビュー

プレビューをもう一度試してみてください（デザインウィンドウに移動し、もう一度プレビューをクリックします）。今回は、小さな時間範囲を指定します。値がデータに存在しなくなったため、列の数がどのように減少するかに注目してください。日付範囲を再び拡大すると、列が再び表示されます。

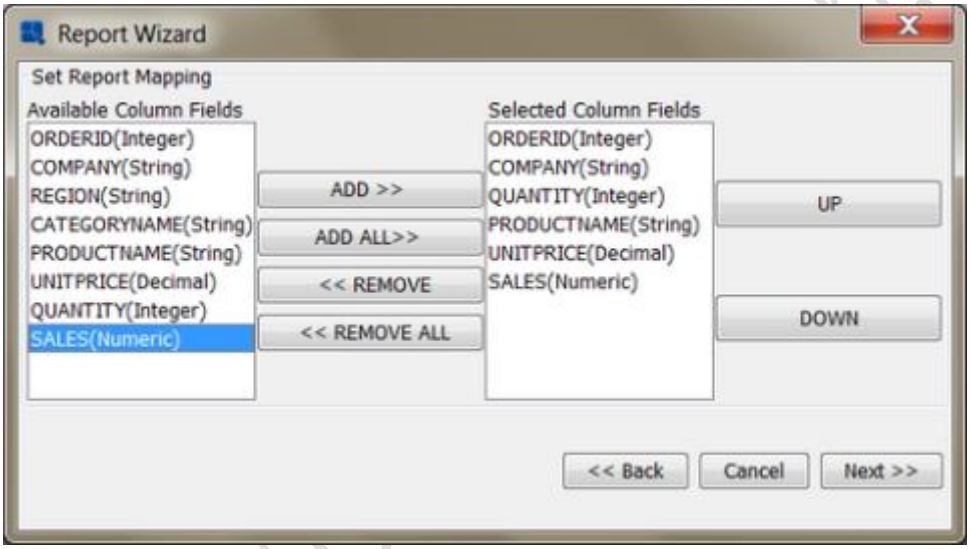
2.3.3.4 マスタ&詳細レイアウト

サマリブレークレイアウトと同様に、マスタ&詳細レイアウトでは、データをグループ化できます。また、グループヘッダーセクションに列フィールドを自動的に追加して、複数の一意のレイアウトを作成して、並べて配置することもできます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Design タブに戻り、ツールバーから **Change Data Mapping** を選択します。再度、レイアウト選択画面に戻ります。今回は、Master & Details レイアウトを選択し、**Next** ボタンをクリックして列選択画面に進みます。列選択画面で、以下の列を含むように選択を変更します：

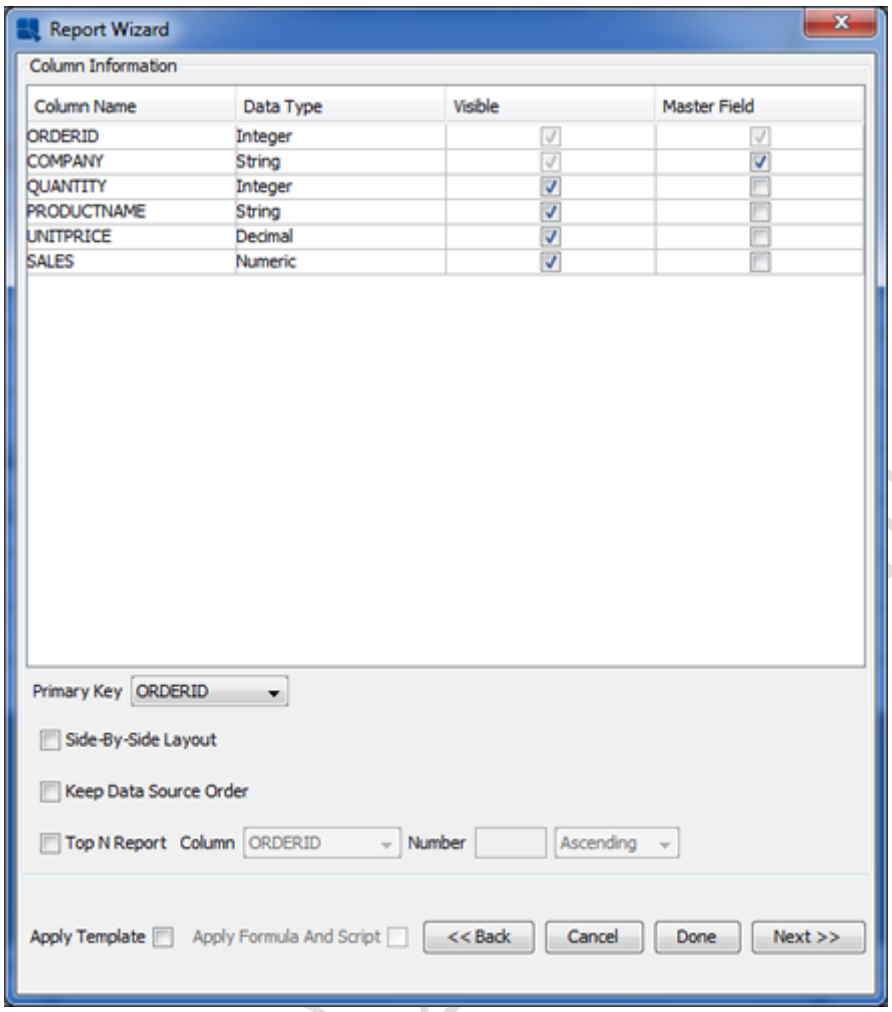
ORDERID
COMPANY
QUANTITY
PRODUCTNAME
UNITPRICE
SALES



マスタ&詳細レイアウトの列選択

列を正しい順序で指定したら、**Next** ボタンをクリックして、マスタ&詳細レイアウトのデータマッピングダイアログを表示します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

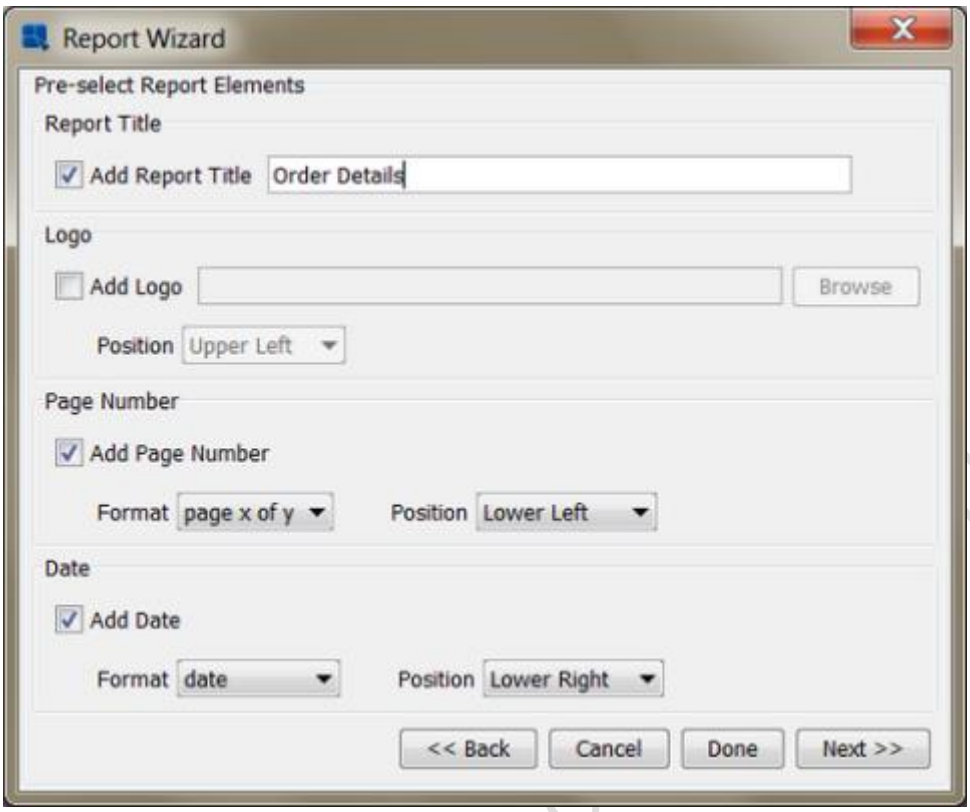


マスタ&詳細レイアウトのデータマッピング画面

画面の左下部分のドロップダウンメニューから、**ORDERID** フィールドを **Primary Key** として選択します。これにより、データが **OrderID** フィールドでグループ化されます。次に、**Company** 列の **Master Field** オプションをチェックします。これにより、**Company** フィールドがレポートの Group Header セクションに配置されます。

Done をクリックしてこの画面から Report Designer にアクセスするのではなく、**Next** ボタンをクリックして、追加のプレフォーマッティングオプションを呼び出します。もう一度、警告メッセージの **Yes** ボタンをクリックすると、いくつかの要素をレポートに追加できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

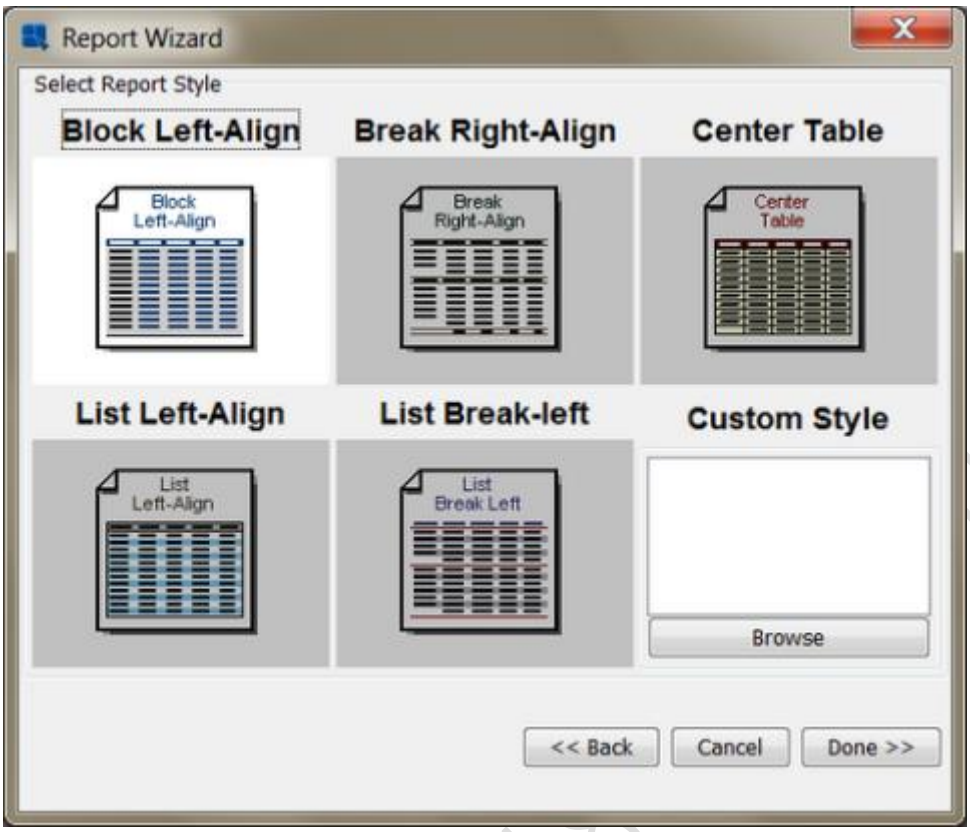


Add Report Elements ダイアログ

レポートのタイトル、ページ番号、および日付を追加するには、チェックボックスをオンにします。レポートタイトルとして使用するテキストを入力し、各オプションのドロップダウンボックスを使用して、日付と時刻の形式と場所を指定します。オプションの設定が終了したら、**Next** をクリックします。新しいダイアログのスタイルを指定するための新しいダイアログが開きます。

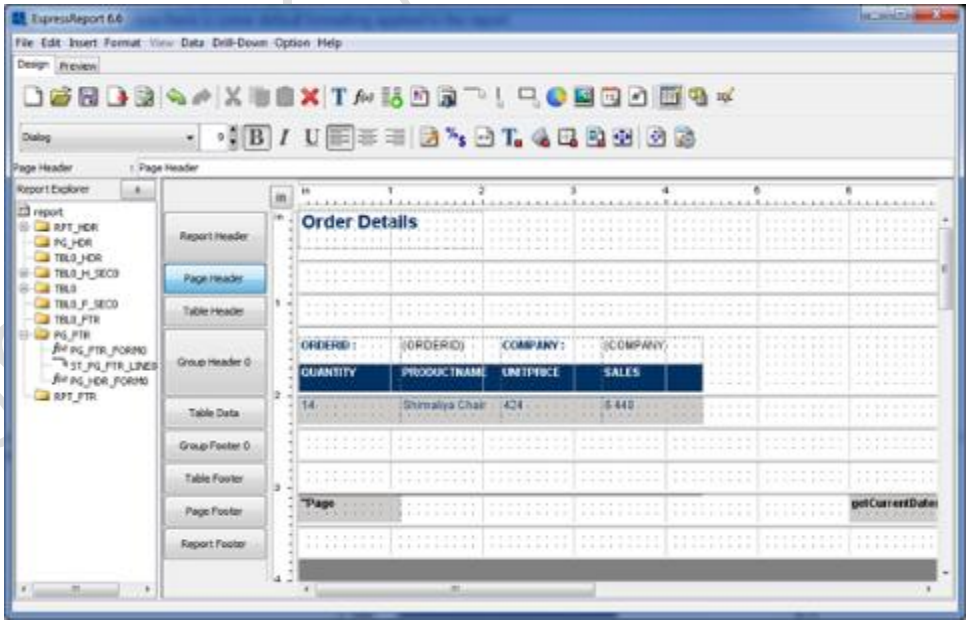
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



レポートスタイルの選択

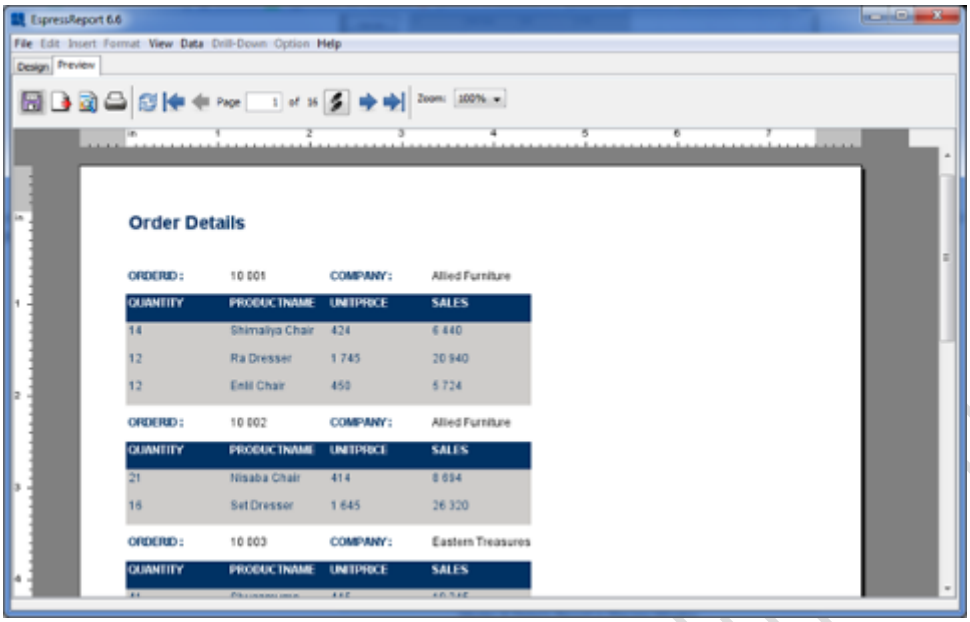
Block Left-Align スタイルを選択し、**Done** ボタンをクリックします。新しい要素が追加された Report Designer ウィンドウに戻り、レポートにもデフォルトの書式設定が適用されます。



デザインウィンドウのマスタ&詳細レポート（プリフォーマット）

ご覧のとおり、**OrderID** フィールドと **Company** フィールドは、Group Header セクションに生成されています。レポートをプレビューすると、各注文のグループと、ページ番号と日付が配置されたセクション（ヘッダーまたはフッター）が表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



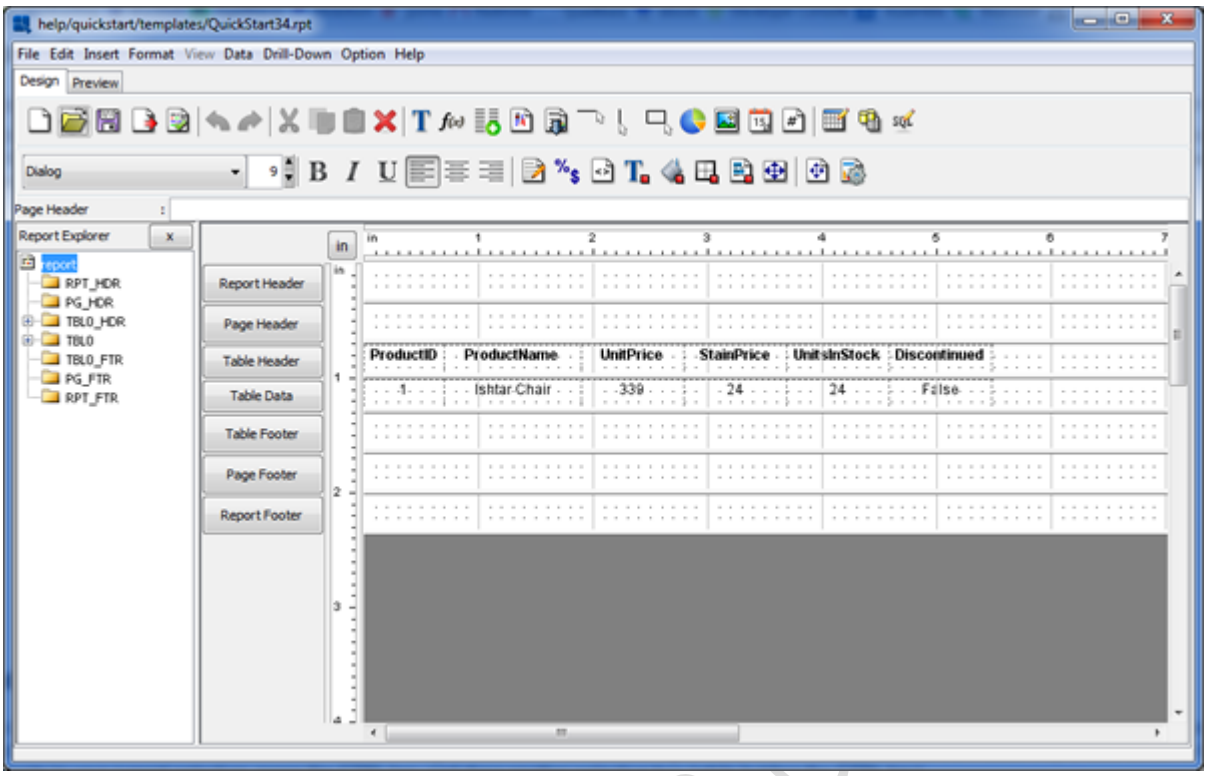
プレビューウィンドウのマスタ&詳細レポート

2.3.4 基本的なレポート書式設定

このセクションでは、書式なしのテンプレートを開き、EspressReport の基本的な書式設定機能を使用して洗練されたプレゼンテーションを作成します。Report Designer のデザインウィンドウで、**Open** ボタン をクリックします。これにより、開くテンプレートのファイル名を指定するダイアログが表示されます。**Browse** ボタンをクリックし、**<InstallDir>/help/quickstart/templates** ディレクトリに移動します。一度そこにいたら、**QuickStart34.rpt** ファイルを選択して開きます。レポートウィンドウが開きます。ご覧のとおり、レポート要素にはほとんどフォーマットがありません。

Copyright(C) 2018 Climb.Inc.

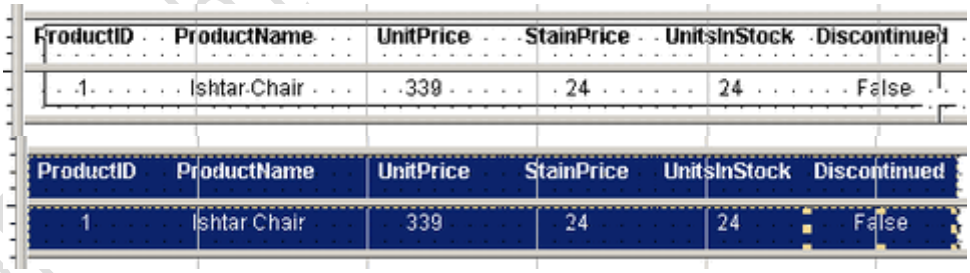
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



デザインウィンドウでの QuickStart34.rpt

2.3.4.1 レポート要素の移動と整列

レポート要素は、セルをクリックしてドラッグすることで、1 つずつ移動できます。レポート要素をグループとして移動することもできます。要素のグループを移動するには、まず選択ボックスを使用してグループを選択する必要があります。選択ボックスをアクティブにするには、クリックしてドラッグしてレポート列の周囲にボックスを描きます。マウスを放すと、それらが強調表示されます。現在の選択範囲に要素を追加するには、**Ctrl** キーを押しながら別の選択ボックスを描画します。



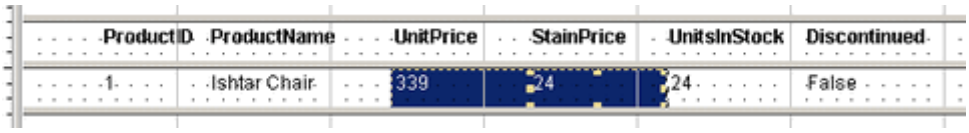
Report Designer の選択ボックスの描画

レポート内のフィールドが選択されたら、クリックしてドラッグして約 1/2 インチ分インデントします。次に、ツールバーの左 **alignment** ボタン：ツールバーの **selection box** ボタン：をクリックします。これにより、セルテキストのすべてがセルの左端に揃えられます。

2.3.4.2 データフォーマット

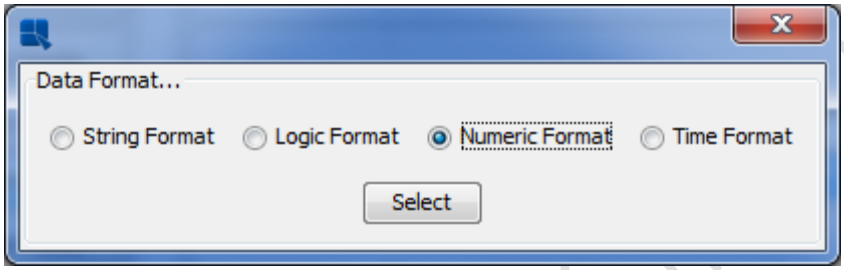
データの表示方法（日付形式、小数点以下の桁数、丸めなど）を制御するためのオプションが用意されています。選択ボックスを再度使用して、**UnitPrice** 列と **StainPrice** 列（ヘッダーではなく列フィールドのみ）を選択します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



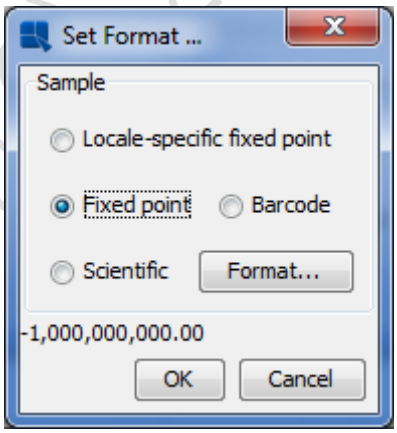
2 列の選択

列が選択されたら、**Format** メニューから **Data Format** オプションを選択します。これにより、フォーマットを設定するデータタイプを選択するダイアログが表示されます。**Numeric Format** を選択し、**Select** をクリックします。



Data Type for Formatting ダイアログ

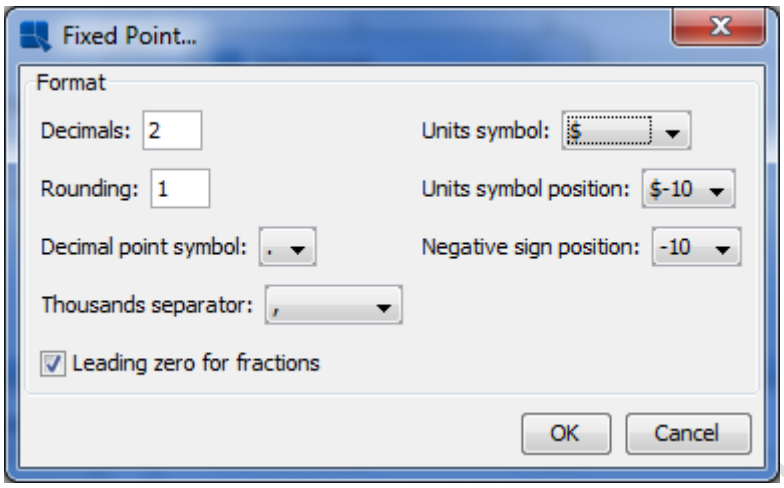
これにより、新しいダイアログが開きます。このダイアログから、**Fixed Point** を選択し、フォーマットボタンをクリックします。次のダイアログで、小数点以下 2 桁を指定し、単位記号としてドル記号を選択します。



Set Format ダイアログ

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



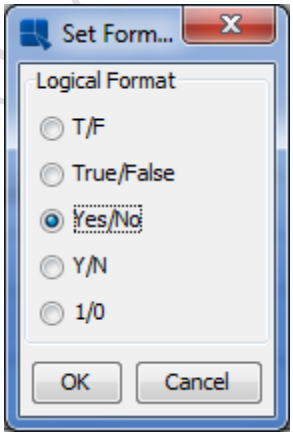
Numeric Format ダイアログ

OK を二回クリックすると、選択したフィールドが Currency Format に変換されます。

ProductID	ProductName	UnitPrice	StainPrice	UnitsInStock	Discontinued
1	Ishtar Chair	$339.00	$24.00	24	False

Currency Format の数値データ

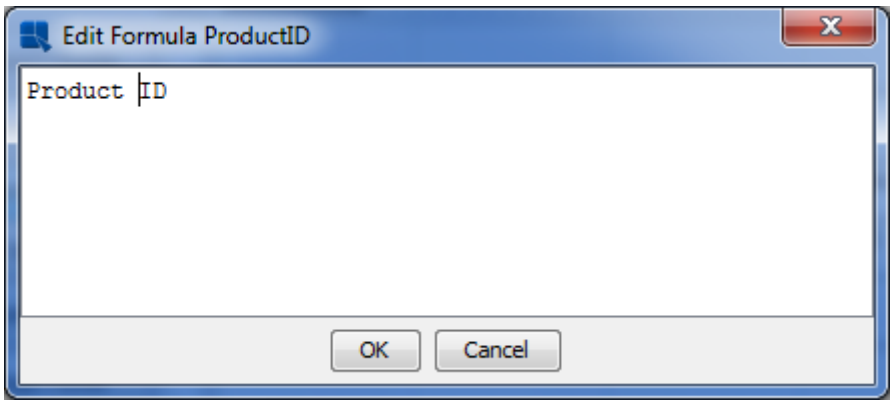
次に、Discontinued 列をクリックして選択します（境界線の輪郭が表示されます）。ここでも、**Data Format** ボタンをクリックします。今回は、ブール列の書式を選択できるダイアログが表示されます。**Yes/No** を選択し、**OK** をクリックします。列のデータが No に変更されます。



Data Format for Boolean ダイアログ

ラベルテキストを編集します。デフォルトでは、列見出しにはデータベースの列名が表示されます。ただし、これらのヘッダーをカスタムヘッダーでオーバーライドできます。これを行うには、ProductID セルをダブルクリックすると、列ヘッダーを変更できるダイアログが表示されます。「Product」と「ID」の間にスペースを挿入し、**OK** をクリックします。変更がデザインウィンドウに表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



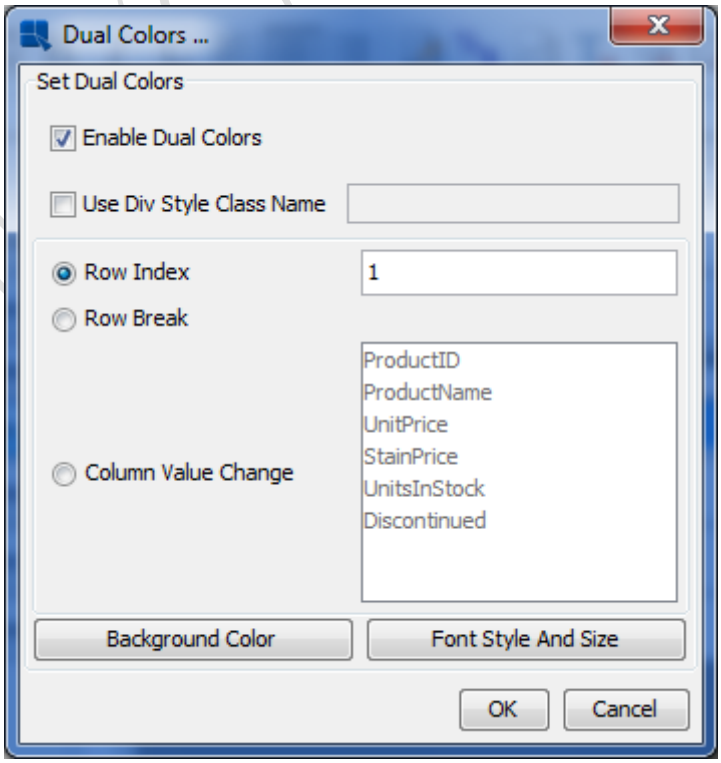
Edit Column Header ダイアログ

すべての列ラベル（「Discontinued」を除く）に対してこれを繰り返して、すべての列に適切な名前を付けます。

2.3.4.3 デュアルカラーを設定する

EspressReport のデュアルカラー機能により、ユーザは背景色やフォントを変更することで、異なる行やグループのデータを区別することができます。デュアルカラーをオンにするには、グループ選択ツールを使用して、ヘッダーではなくレポートのすべての列を選択します。

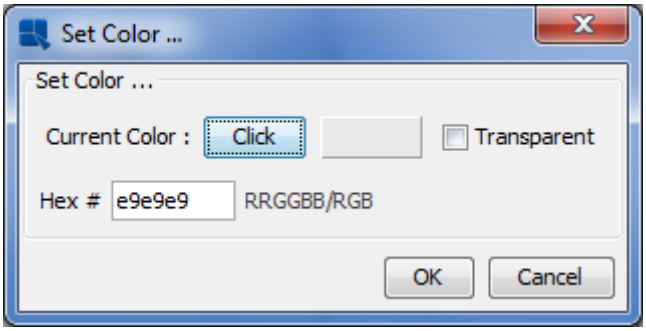
次に、ツールバーの **Dual Colors** ボタン を選択します。列の二色を設定するダイアログが表示されます。**Enable Dual Colors** チェックボックスをクリックします。次に、**Row Index** ラジオボタンを選択して、行の値に基づいて色を変更することを示します。行インデックス値に **1** を入力します。



Dual Colors ダイアログ

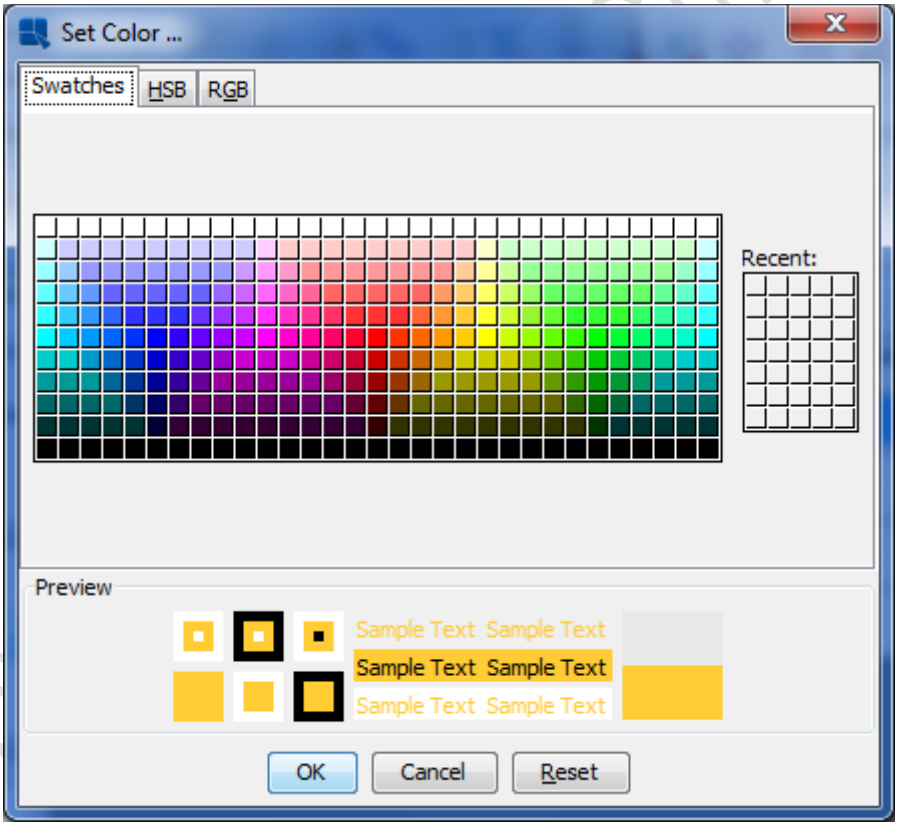
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次に、**Background Color** ボタンをクリックします。ダイアログが表示され、背景を透明に設定し、現在の背景色を表示するオプションが表示されます。



Background Transparency ダイアログ

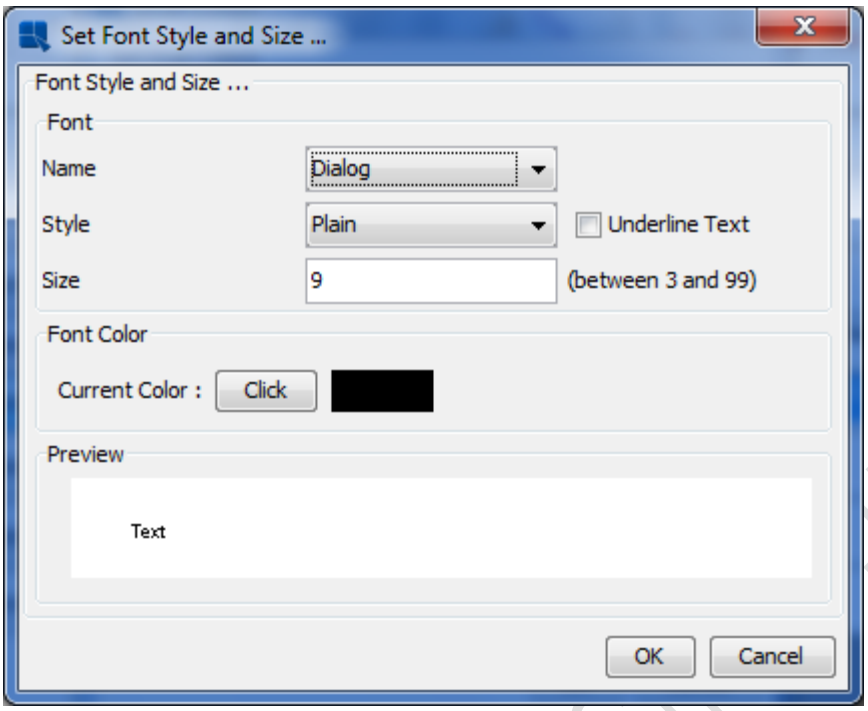
Click ボタンをクリックすると、背景色を変更できるカラースウォッチを含む新しいダイアログが表示されます。



Choose Color ダイアログ

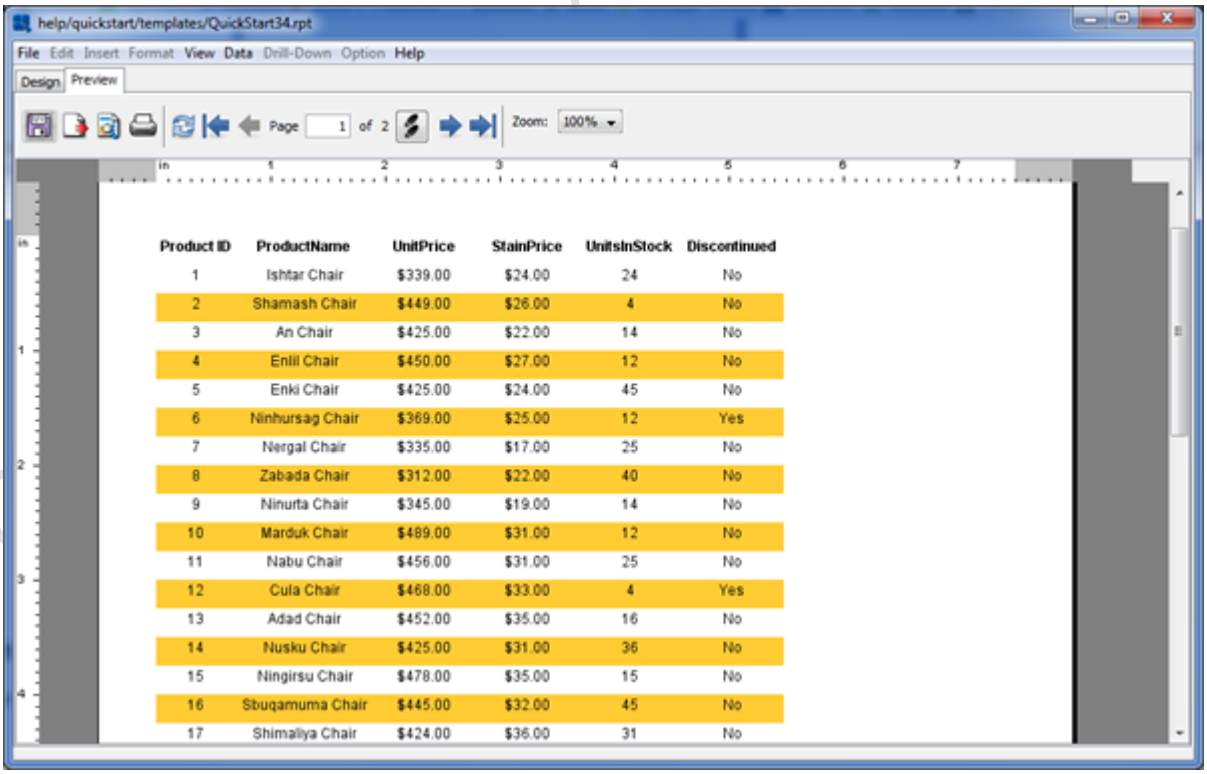
使用する新しい背景色を選択し、**OK** ボタンをクリックします。最初のダイアログに戻り、色の選択が反映されます。**OK** ボタンをもう一度クリックすると、Dual Colors ダイアログに戻ります。Dual Colors ダイアログに戻ったら、**Font Style and Size** ボタンをクリックします。これにより、交互行のフォント、フォントサイズ、フォントスタイルを指定できるダイアログが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Font Style and Size ダイアログ

このダイアログで、**Name** を **Dialog** に、**Style** を **Plain** に、**Size** を **9** に設定します。これは交互の行のフォントと一致します。**OK** ボタンをクリックすると、**Dual Color** ダイアログに戻り、もう一度 Report Designer に戻ります。レポートをプレビューすると、各行に色の帯が交互に表示されます。



Product ID	ProductName	UnitPrice	StainPrice	UnitsInStock	Discontinued
1	Ishtar Chair	$339.00	$24.00	24	No
2	Shamash Chair	$449.00	$26.00	4	No
3	An Chair	$425.00	$22.00	14	No
4	Enlil Chair	$450.00	$27.00	12	No
5	Enki Chair	$425.00	$24.00	45	No
6	Ninhursag Chair	$369.00	$25.00	12	Yes
7	Nergal Chair	$335.00	$17.00	25	No
8	Zabada Chair	$312.00	$22.00	40	No
9	Ninurta Chair	$345.00	$19.00	14	No
10	Marduk Chair	$489.00	$31.00	12	No
11	Nabu Chair	$456.00	$31.00	25	No
12	Cula Chair	$468.00	$33.00	4	Yes
13	Adad Chair	$452.00	$35.00	16	No
14	Nusku Chair	$425.00	$31.00	36	No
15	Ningirsu Chair	$478.00	$35.00	15	No
16	Sbuqamuma Chair	$445.00	$32.00	45	No
17	Shimaliya Chair	$424.00	$36.00	31	No

Dual Colors のプレビュー

2.3.4.4 要素を挿入する

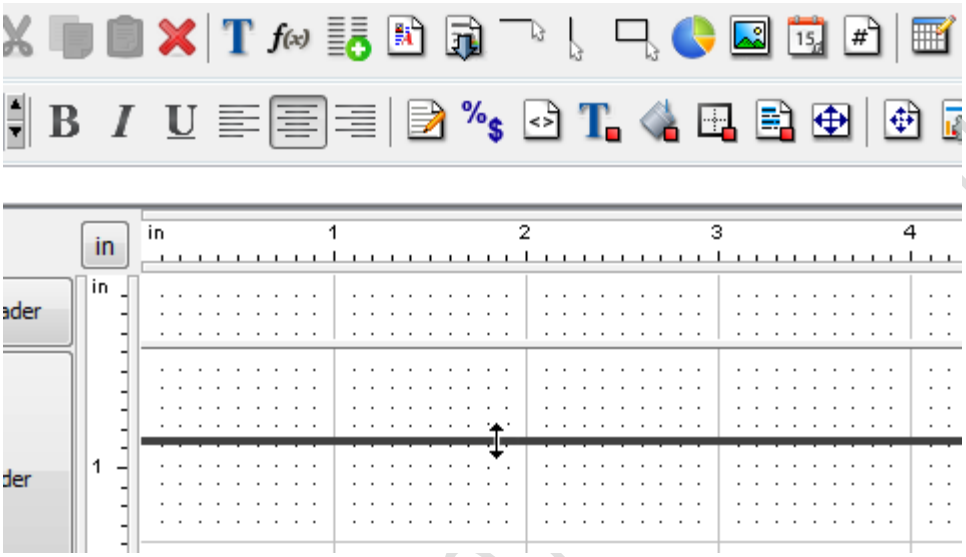
レポートをさらにカスタマイズするには、さまざまな種類の要素をレポートテンプレートに追加できま

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

す

2.3.4.4.1 イメージを挿入する

イメージをレポートヘッダーに挿入するには、まずイメージに合わせてレポートセクションのサイズを変更する必要があります。デザインウィンドウで、レポートヘッダーセクションの下部のディバイダーにマウスを置き、クリックしてドラッグして、セクションの高さを約 1 インチ高くします。

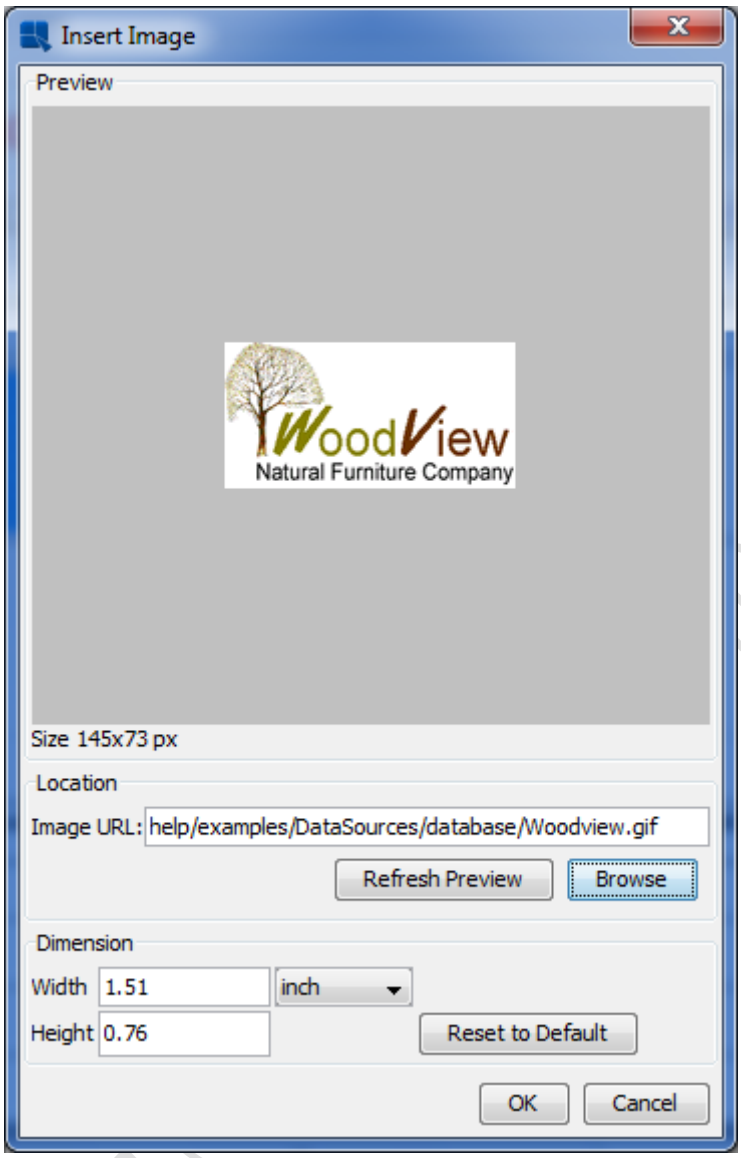


レポートセクションのサイズ変更

セクションのサイズを変更したら、ツールバーの **Image** ボタン をクリックします。デザインウィンドウの周りに小さな四角形がマウスポインタの後に表示されます。レポートヘッダーセクションの左上隅に四角形を置き、クリックします。挿入する画像を選択するようダイアログが表示されます。**Browse** をクリックし、**help/examples/DataSources/database/Woodview.gif** に移動します。プレビューパネルに画像が表示されるはずです。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



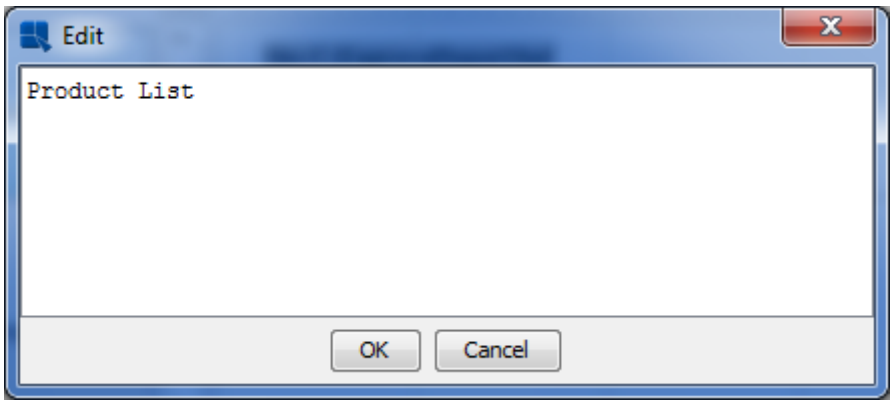
Insert Image ダイアログ

OK をクリックすると、イメージがレポートに挿入されます。イメージはデザインビューでグレーの矩形で表されます。レポートをプレビューすると、イメージが表示されます。

2.3.4.4.2 タイトルを挿入する

タイトルを挿入するには、ツールバーの **insert label** ボタン T をクリックします。デザインウィンドウの周りに小さな四角形がマウスポインタの後に表示されます。挿入した画像の横に長方形を置き、クリックします。ラベルテキストの入力を求めるダイアログが表示されます。目的のタイトルのテキストを入力します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



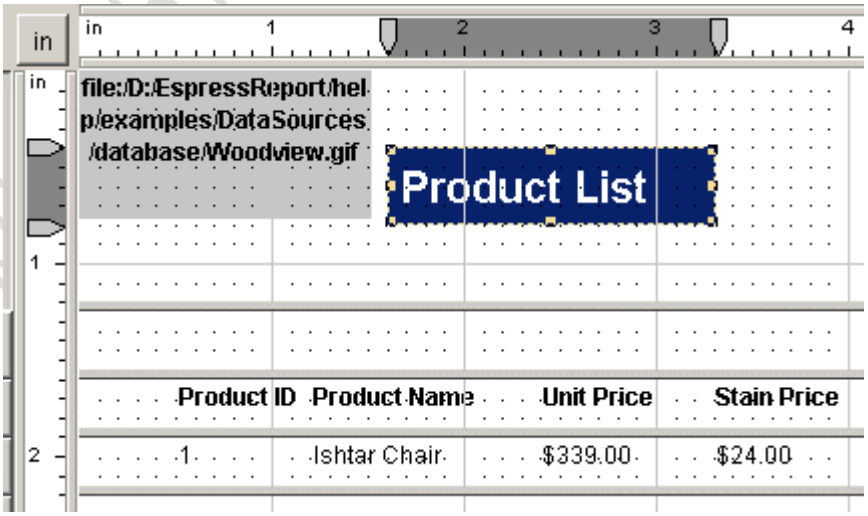
Insert Label ダイアログ

OK をクリックすると、タイトルがレポートに追加されます。ご覧のように、デフォルトではテキストはかなり小さいです。これを変更するには、ツールバーのフォントサイズダイアログを 18pt フォントに変更します。

ツールバーのフォントオプション

これを行うと、タイトルセルのテキストの一部が消えていることがわかります。これは、テキストが定義されたスペースより大きくなったためです。セルのサイズを変更するには、レポートタイトルを再度表示できるようになるまで、サイズ変更ハンドルをクリックしてドラッグします。

次に、ツールバーの **Left Alignment** ボタンをクリックして、他のレポート要素の場合と同様にテキストの配置を左に設定します。挿入した画像の横にタイトルセルを移動して配置します。



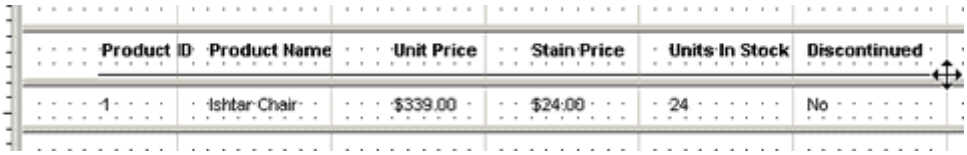
画像とタイトルを含むレポートヘッダー

2.3.4.4.3 ラインを挿入する

次に、列見出しの下に水平線を追加します。これを行うには、最初にテーブルヘッダーセクションのサイズを少し変更して、列ヘッダーの下にさらにスペースを確保します。次に、ツールバーの **Insert**

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

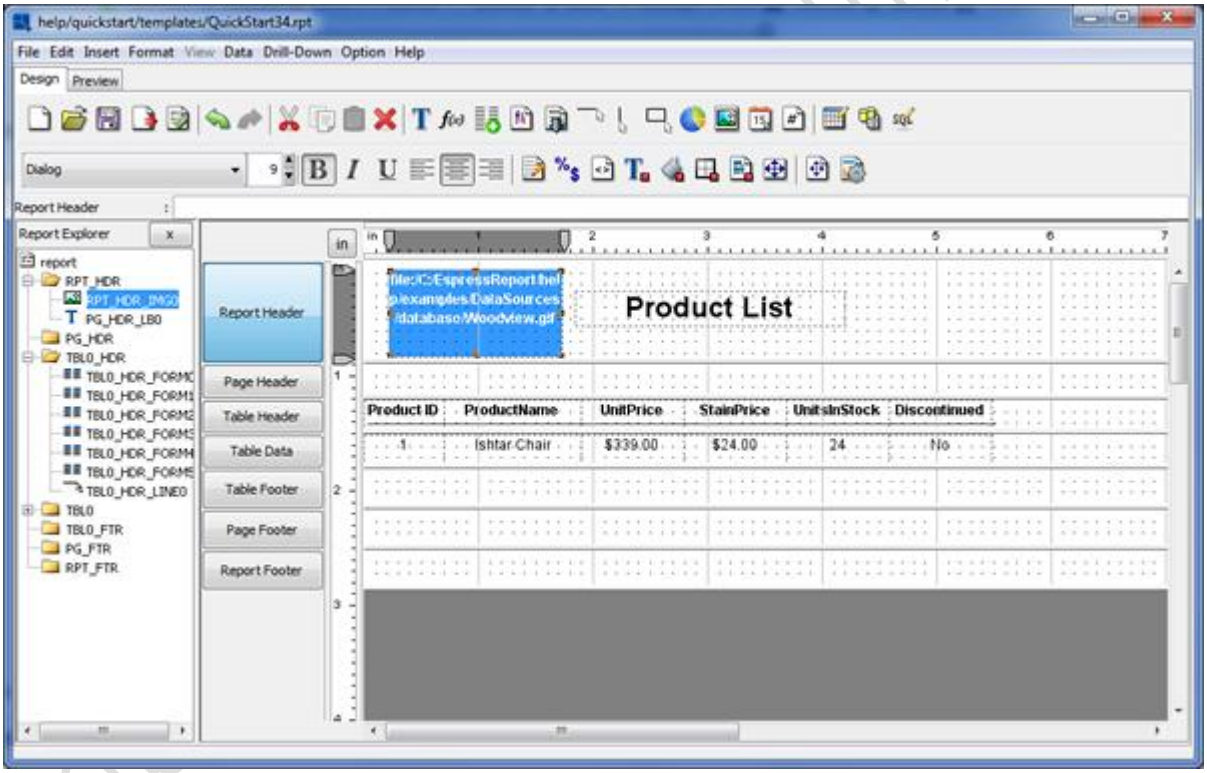
Horizontal Line ボタン を選択します。カーソルが十字に変わります。ProductID ヘッダーの下をクリックし、最後の列にドラッグして線を描画します。



Report Designer で線を描く

2.3.4.5 レポートエクスプローラでのレポート要素の表示

レポート要素を選択する際、Report Designer の左側にある **Report Explorer** パネルに注目して、レポート要素をツリー形式で表示することができます。クリックするとセクションのいくつかが展開され、ツリーに表示されたレポートのすべての要素が表示されます。ツリー内の要素のいずれかを選択すると、レポート内の対応する要素が選択されます（逆も同様です）。



エクスプローラを開いた Designer

Option > **Report Explorer** を選択するか、**Report Explorer** パネルの右上隅にある **X** ボタンをクリックして、Report Explorer を閉じることができます。

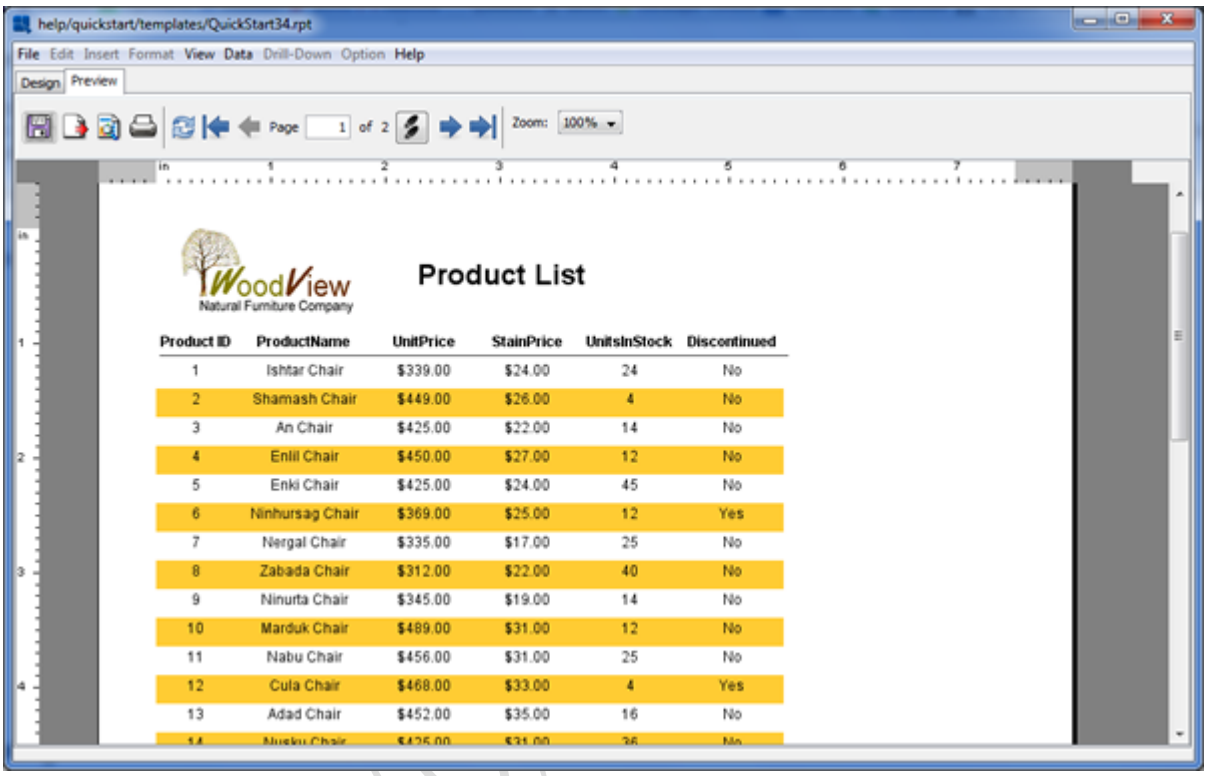
2.3.4.6 セクションオプションを設定する

EspressReport では、各レポートセクションにいくつかの設定可能なオプションがあり、さまざまな方法でセクションにデータを表示できます。セクションのオプションメニューを呼び出すには、デザインウィンドウの左側にあるそのセクションのボタンをクリックします。この例では、対応するボタンをクリックして、テーブルヘッダーセクションのオプションメニューを表示します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ポップアップメニューから **Repeat On Every Page** を選択します。これにより、テーブルヘッダーが各レポートページで毎回描画されます（これがデフォルトです）。レポートセクションオプションの詳細については、デザイナーガイドのセクションオプションを参照してください。

レポートの書式を設定したら、プレビューして結果を確認します。



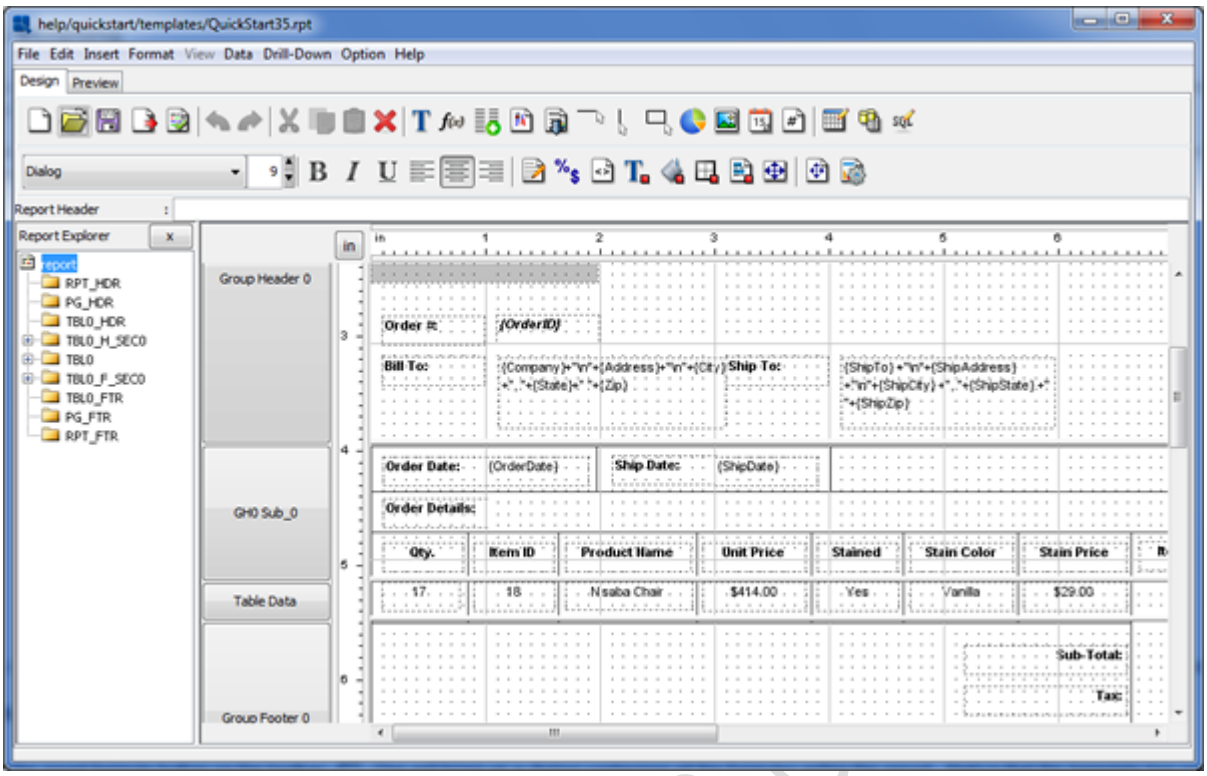
完成したレポート

2.3.5 数式とスクリプティング

EspressReport は、組み込み式の大規模なスクリプトライブラリを提供しており、レポートデータを操作および分析するさまざまな方法を提供します。次のセクションでは、テンプレートを使用して数式とスクリプトを使用して、レポートにいくつかの値を計算、追加します。

デザインウィンドウから、**Open** ボタン をクリックします。次に、**<InstallDir>/help/quickstart/templates** ディレクトリを参照し、**QuickStart35.rpt** ファイルを選択して開きます。レポートがデザインウィンドウに表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



デザインウィンドウでの QuickStart35.rpt

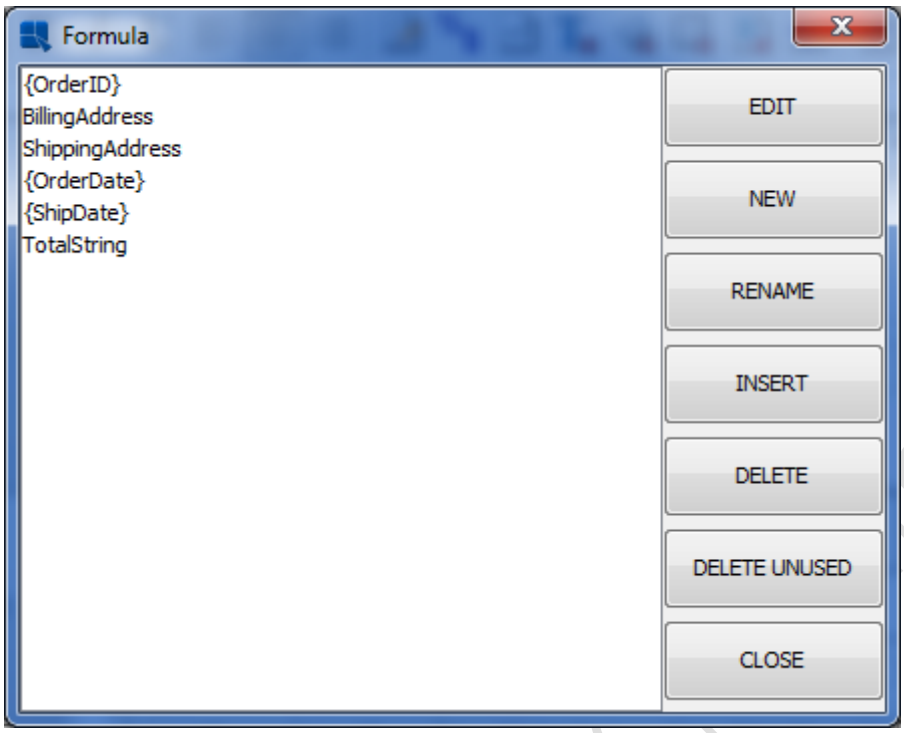
レポートは、マスタ&詳細レイアウトを使用して作成された請求書です。Item Total、および小計は空白であることに注意してください。これらの値を計算する数式を追加します。

2.3.5.1 数式を追加する

数式を挿入するには、ツールバーの **Insert Formula** ボタン $f(x)$ をクリックします。これにより、レポート内のすべての数式を含むダイアログが表示されます。テンプレートにはいくつかの既存の数式があることに注意してください。**NEW** ボタンをクリックして新しい数式を作成し、プロンプトに **ItemTotal** という名前を入力します。

Copyright(C) 2018

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

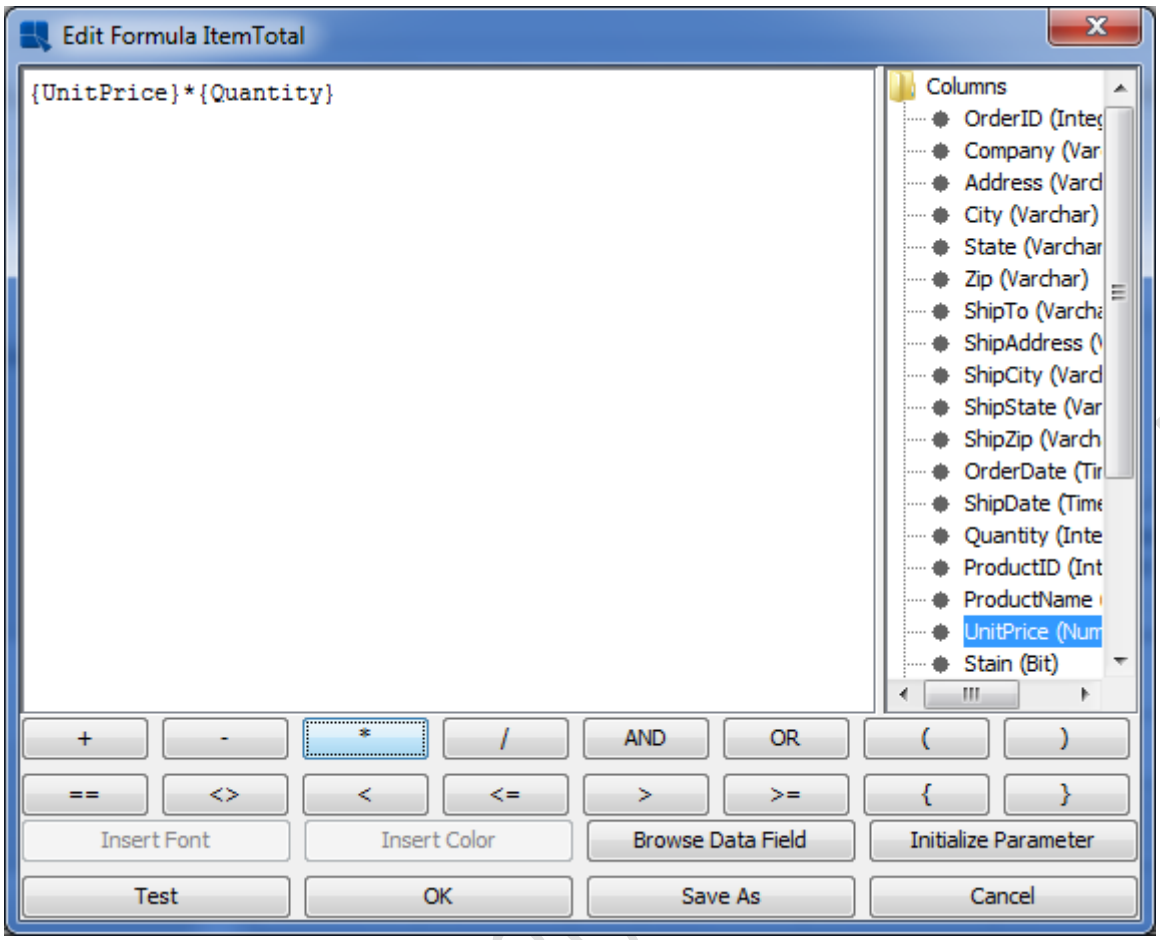


Report Formula List

Formula Builder ウィンドウが開きます。右側の **Columns** フォルダをダブルクリックして展開します。**UnitPrice** 列をダブルクリックして数式に追加します。次に、掛け算*ボタンをクリックし、**Quantity** 列をダブルクリックして追加します。完成した式は次のようになります。

```
{UnitPrice}*{Quantity}
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Formula Builder ウィンドウ

Test ボタンをクリックして、数式が正しく入力されていることを確認します。次に **OK** をクリックします。新しい数式が追加された数式リストに戻ります。数式リストから、作成した **ItemTotal** 数式を選択し、**Insert** ボタンをクリックします。ダイアログが閉じて、小さな点線の四角形がデザインウィンドウの周りのポインタの後に表示されます。**Item Total** ラベルの下および Table Data セクションの行間に数式を配置し、クリックします。計算式がレポートに追加されます。

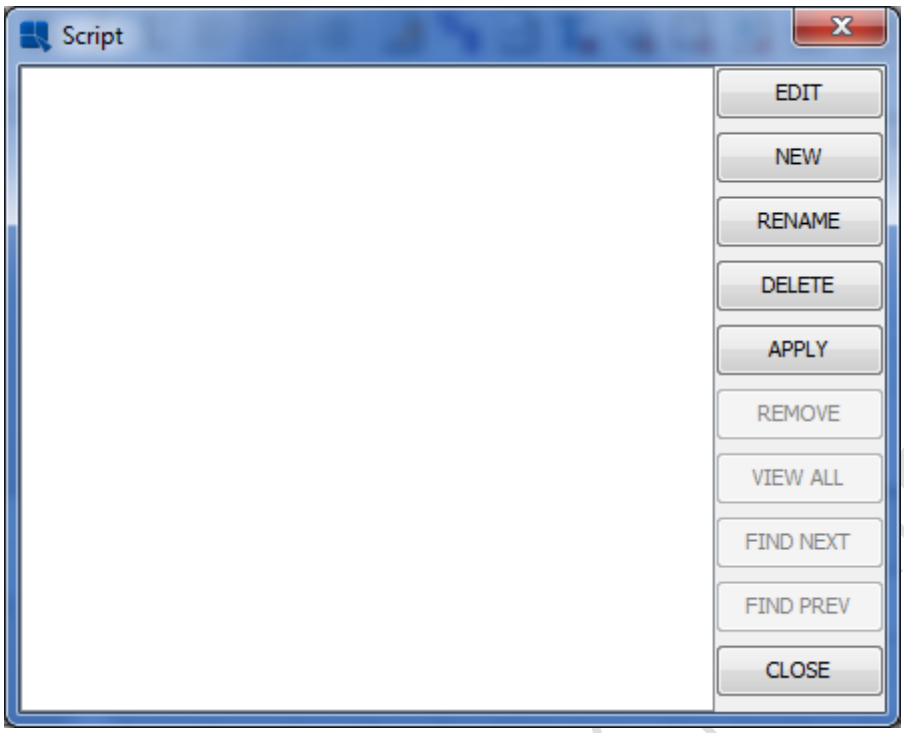
レポートをプレビューし、次のダイアログで **Use Live Data** を選択します。テーブルデータセクションに数式が追加されたので、データの各行を計算するようになりました。また、数式データ形式が自動的に通貨に設定されていることに気づいたこともあります。

2.3.5.2 スクリプトを追加する

前のセクションで追加した数式が請求書の行合計を正しく計算していないことに注意してください。単価と数量を掛けるだけで、品目が染色されたかどうか（追加コストがかかります）は無視されます。これを処理するために、セルスクリプトを使用します。

スクリプトを追加するには、数式を追加するで作成した **ItemTotal** 式を選択し、ツールバーのスクリプトボタン をクリックします。これにより、レポート内のすべてのスクリプトを含むダイアログが表示されます。以前にスクリプトが追加されていないので、ダイアログは空白になります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Script List ウィンドウ

NEW をクリックして新しいスクリプトを作成します。プロンプトで、スクリプトの **StainCheck** という名前を入力します。**OK** をクリックすると、数式ビルダが開き、スクリプトを追加できます。次のスクリプトを入力します。

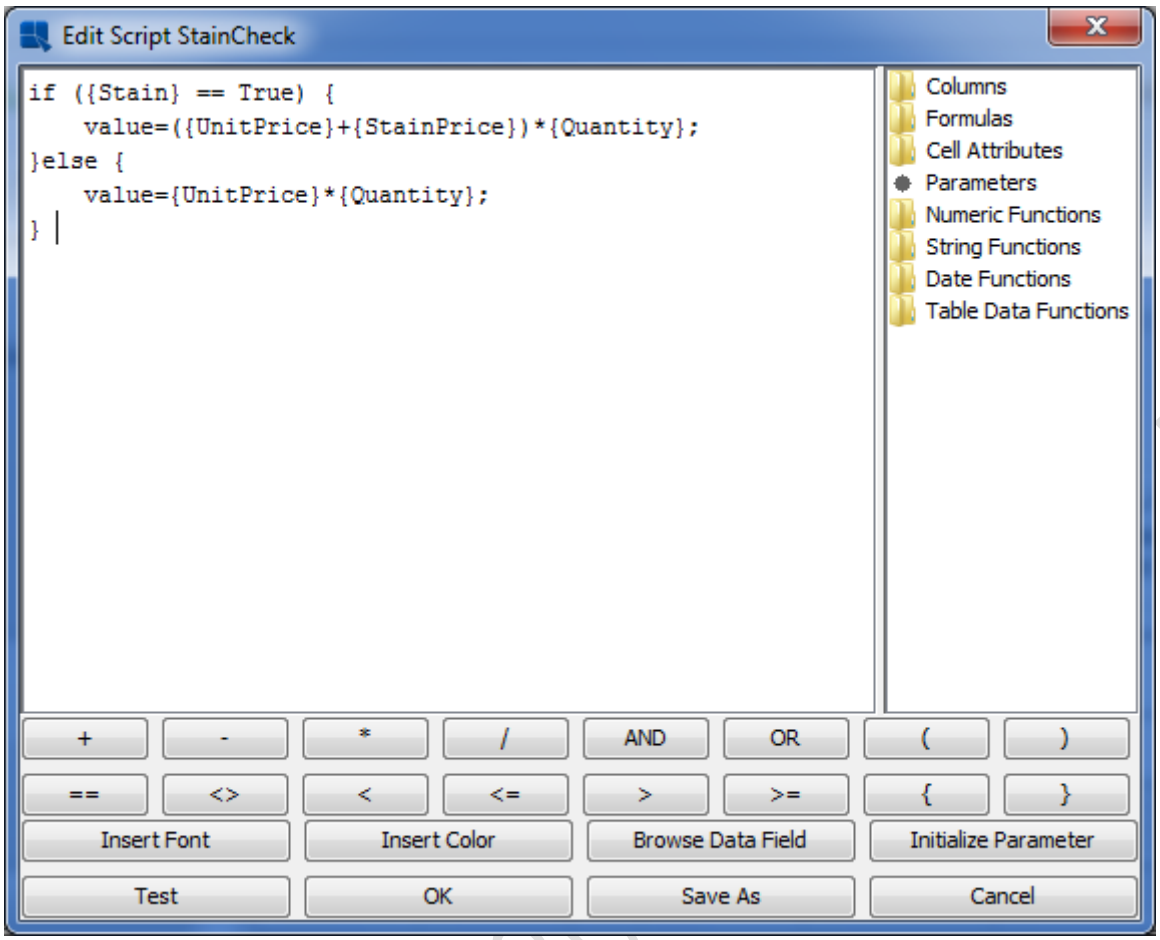
```
if ({Stain} == True) {
value=({UnitPrice}+{StainPrice})*{Quantity};
 }else {
value={UnitPrice}*{Quantity};
 }
```

このスクリプトは、商品が染色されているかどうかによって **ItemTotal** 価格を動的に変更します。アイテムが汚れている場合

{Stain} == True

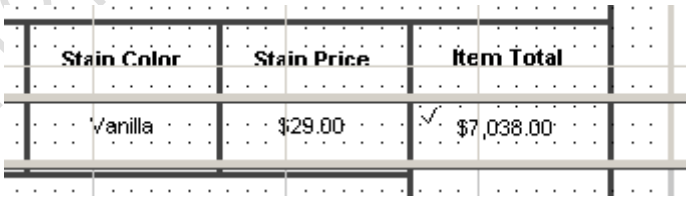
ItemTotal 値には汚れの価格が含まれます。 商品が汚れていない場合、価格は以前と同じ方法で計算されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



セルスクリプトの Formula Builder

Test をクリックして、スクリプトが正しく入力されていることを確認します。次に **OK** をクリックします。新しいスクリプトが追加されたスクリプトリストに戻ります。スクリプトリストで、作成したスクリプトを選択し、**APPLY** ボタンをクリックします。スクリプトが列に適用されます。**Item Total** セルの左上隅にチェックマークが表示されるようになりました。



デザインウィンドウでスクリプトを適用したセル

2.3.5.3 集計を追加する

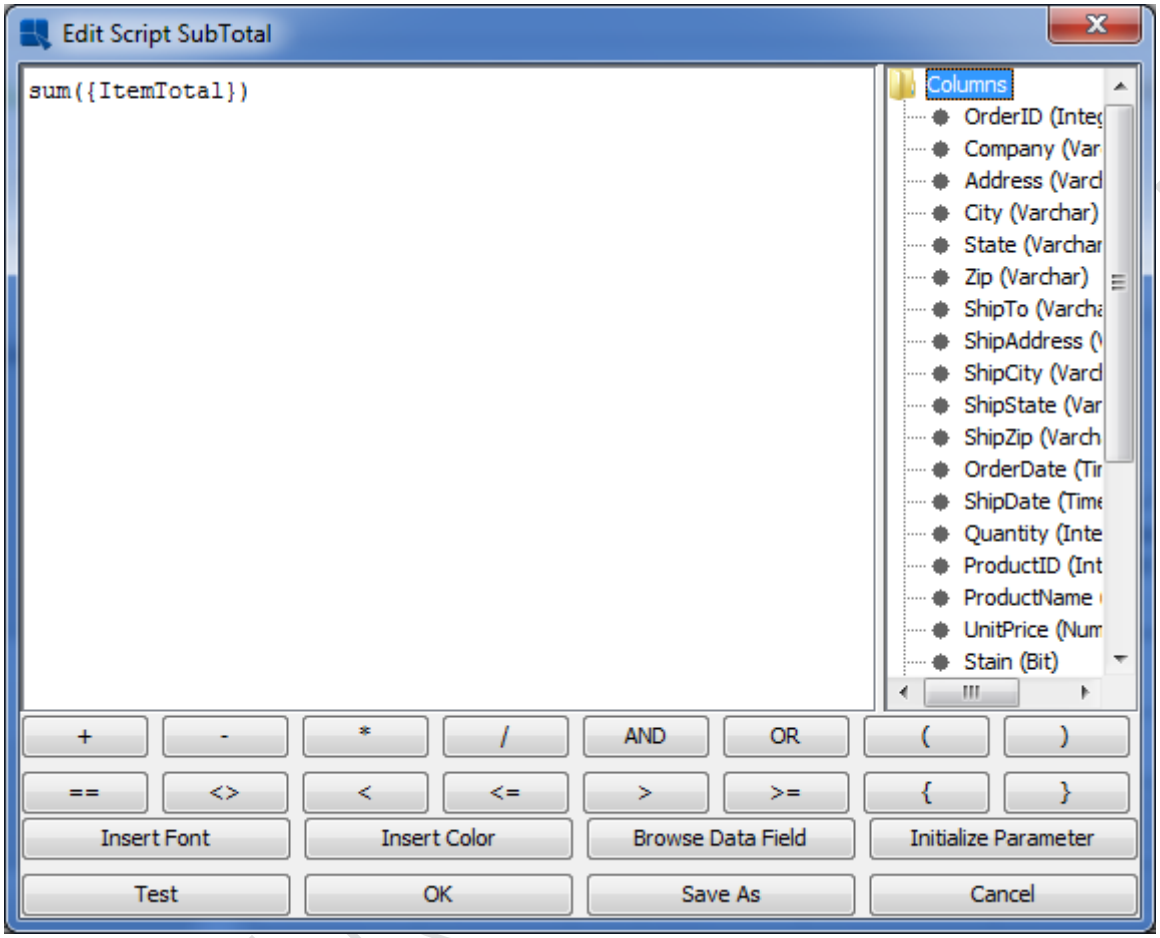
EspressReport では、数式を使用してレポート列を集約することもできます。集計を追加することは、数式を追加するで作成したような式を追加するのと同じです。新しい数式を追加するには、ツールバーの **Insert Formula** ボタン $f(x)$ をクリックします。この式に **SubTotal** という名前を付けます。

数式ビルダで、**Numeric Functions** フォルダをダブルクリックして展開します。**Sum** 関数をダブルクリックして数式に挿入します。次に、カーソルを使用して、sum 関数の"**field**"部分を強調表示します。次に、列フォルダをダブルクリックして展開します。リストの最後には、数式を追加するで作成した

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

「**ItemTotal**」があります。これをダブルクリックして数式に追加します。完成した式は次のようになります。

sum({ItemTotal})

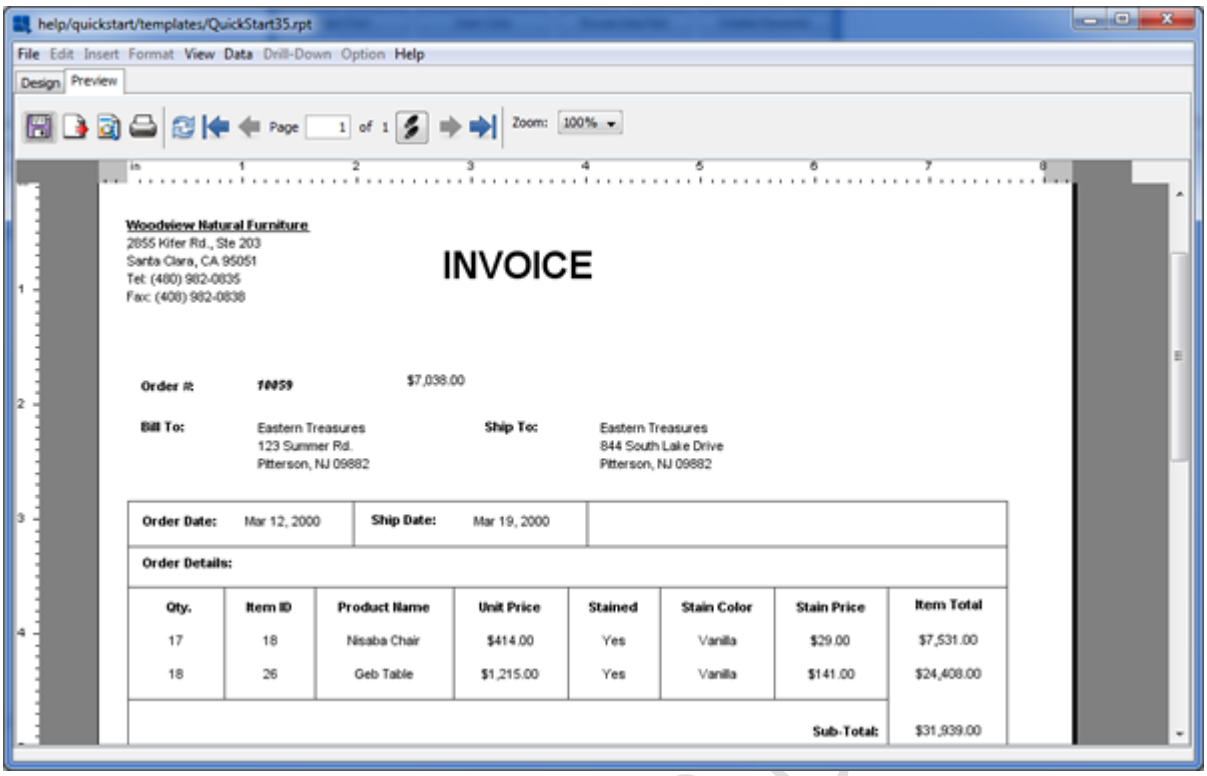


集計式を含む数式ビルダ

Test をクリックして、数式が正しく入力されていることを確認します。次に、**OK** をクリックして数式リストに戻ります。数式リストで、作成した数式を選択し、**INSERT** ボタンをクリックします。ダイアログが閉じて、小さな点線の四角形がデザインウィンドウの周りのポインタの後に表示されます。レポートのグループフッターセクションの **ItemTotal** 列の下にあり、**Sub-Total** の横に数式を配置し、クリックして追加します。

数式はグループフッターセクションにあるため、レポートが実行されて数式のテキストのみが表示されるまで計算されません。レポートをプレビューします。集計には、セルスクリプトによって変更された値が反映されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



数式によるレポート

追加のチュートリアルでは、2 つの数式を Group Footer セクションに追加します。1 つは売上税を計算するもの、もう 1 つは注文の総計を計算するものです。

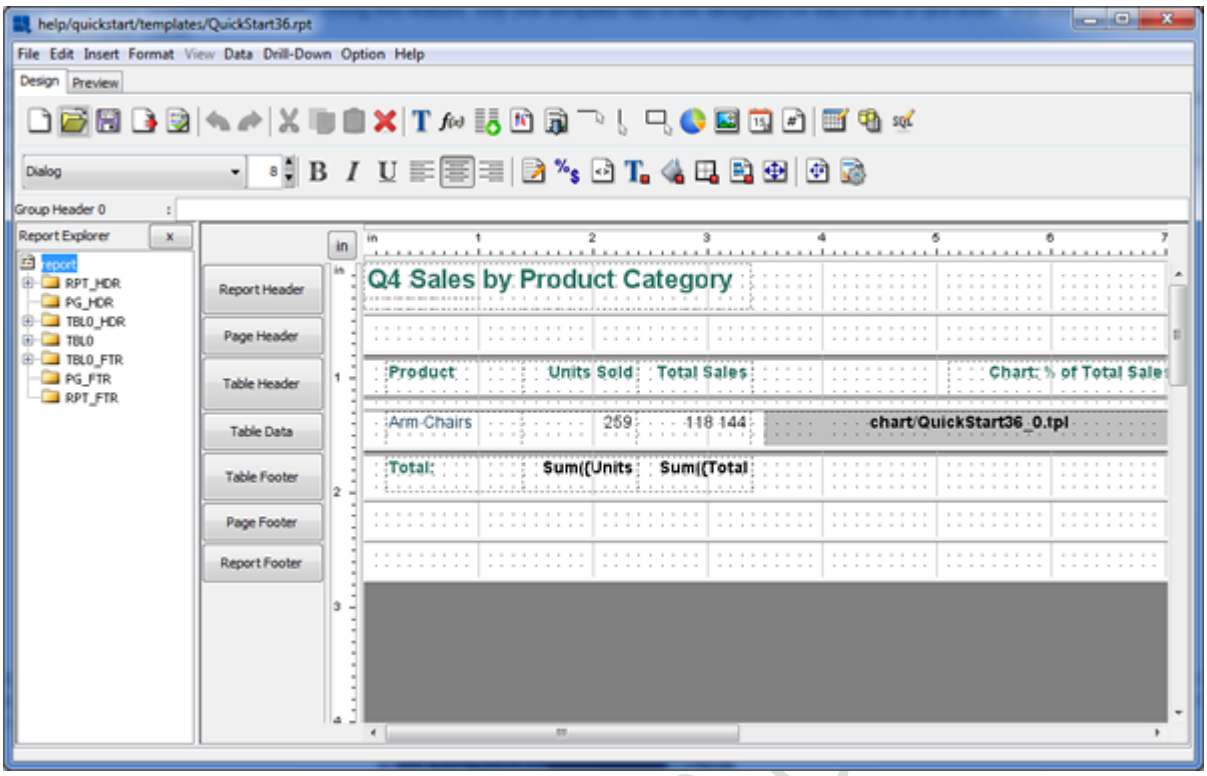
2.3.6 ドリルダウン

EspressReport のユニークな機能は、レポートを自動的にドリルダウンする機能です。ドリルダウンを使用すると、ユーザはサマリデータをトップレベルのレポートに簡単に表示したり、クリックして詳細な情報を表示することができます。この機能を使用すると、ドリルダウンの各レベルに 1 つのテンプレートのみを設計する必要があります。この機能の詳細については、ドリルダウンを参照してください。

ドリルダウン機能はパラメータ化されたクエリを使用するため、このチュートリアルを実行するには、Woodview データベースをセットアップする必要があります（詳細は JDBC 接続のセットアップを参照してください）。

デザインウィンドウから、**Open** ボタンをクリックします。次に、**<InstallDir>/help/quickstart/templates** ディレクトリを参照し、**QuickStart36.rpt** ファイルを選択して開きます。レポートがデザインウィンドウに表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

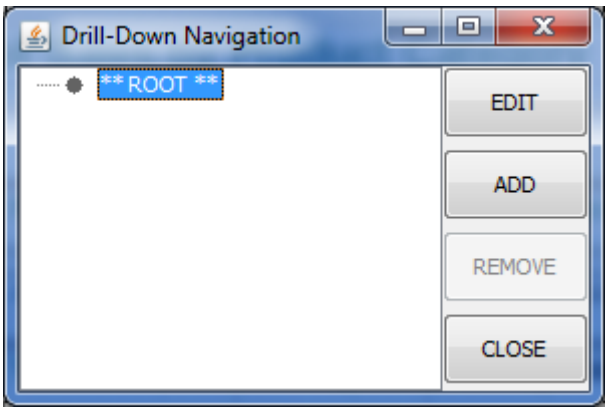


デザインウィンドウで QuickStart36.rpt

レポートには、製品カテゴリ別に集計された集計データが表示されます。次に、このテンプレートを別のパラメータ化されたテンプレートにリンクし、ユーザが各カテゴリにドリルダウンし、各カテゴリの製品の販売数量を確認できるようにします。ドリルダウンのレイヤーを追加するには、**Drill-Down** > **Navigate** を選択します。これにより、このレポートのすべてのドリルダウンレイヤの階層を示すダイアログが表示されます。定義されているレイヤーがないため、** **ROOT** **だけが表示されます。

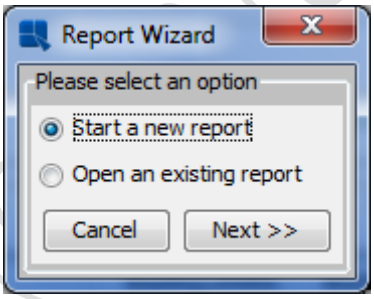
Copyright(C) 2018

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



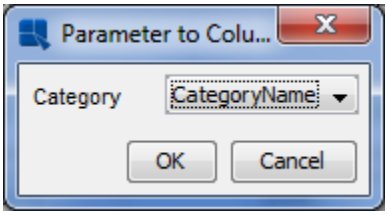
Drill-Down Navigation ダイアログ

ドリルダウンの新しいレイヤーを追加するには、**ADD** ボタンをクリックします。既存のテンプレートを使用するか新しいテンプレートを作成するかを尋ねるダイアログが表示されます。新しいテンプレートの作成を選択すると、データレジストリに戻り、そこでレポートの設計を開始できます。既存のレポートを使用することもできます。いずれの場合も、使用するレポートには、データソースとしてパラメータ化されたクエリまたはクラスが必要です。詳細は、パラメータ化されたクエリおよび Designer ガイドのパラメータ化されたクラスファイルを参照してください。この場合、既存のレポートを開き、**Next >>**ボタンをクリックします。



Report Options ダイアログ

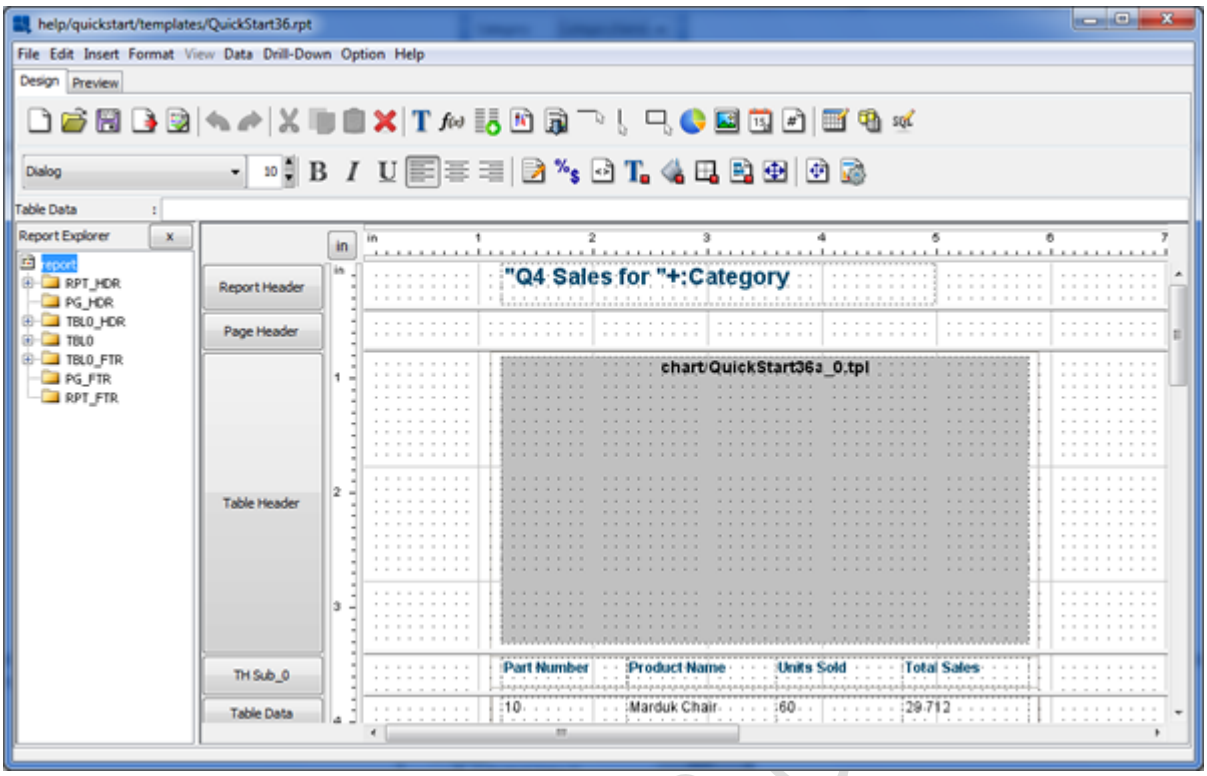
プロンプトで、**<InstallDir>/help/quickstart/templates** を参照し、**QuickStart36a.rpt** ファイルを選択します。(**/Access/**ディレクトリからの **QuickStart36a_Acc.rpt**)。ダイアログが開き、プライマリレポートからドリルダウンレイヤにマップする列を選択するように求められます。



Column - Parameter Mapping ダイアログ

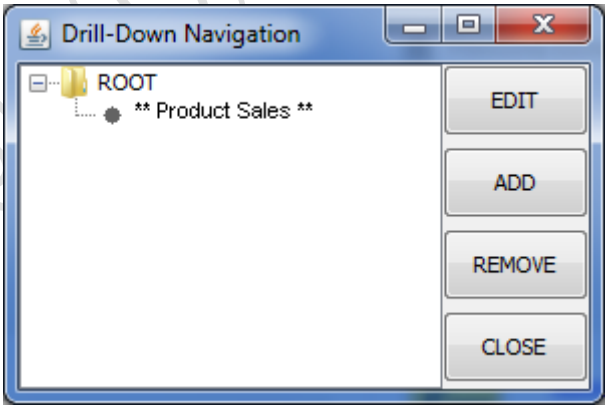
Categoryname 列をパラメータにマップし、**OK** をクリックします。レポートの表示名を指定するダイアログが開きます。任意の名前を入力し、**OK** をクリックします。ドリルダウンレイヤがデザインウィンドウで開きます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Report Designer の最初のドリルダウンレベル

レポートをプレビューすると、カテゴリ名に基づいてレポートがどのようにパラメータ化されているかがわかります。設計ウィンドウに戻り、**Drill-Down > Navigate** を再度選択します。追加されたレベルの新しいノードが ROOT ノードの下に表示されます。星印「**」は、現在どのレベルがデザイナで開いているかを示します。



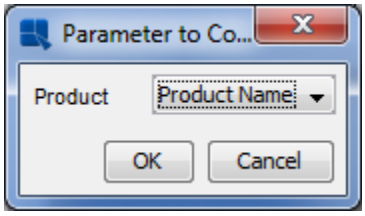
追加レイヤーを含む Drill-Down Navigation ダイアログ

次に、このレポートにドリルダウンのレイヤーを 1 つ追加します。これにより、ユーザは製品販売を掘り下げ、特定の製品の各注文の記録を見ることができます。これを行うには、作成したレベルのノードを選択し、**ADD** をクリックします。

再度、新しいドリルダウンレイヤの既存のテンプレートを開く場合に選択します。ダイアログで、**<InstallDir>/help/quickstart/templates** を参照し、QuickStart36b.rpt ファイルを選択します。(または/ Access /ディレクトリからの QuickStart36b_Acc.rpt)。これにより、列マッピングダイアログへ

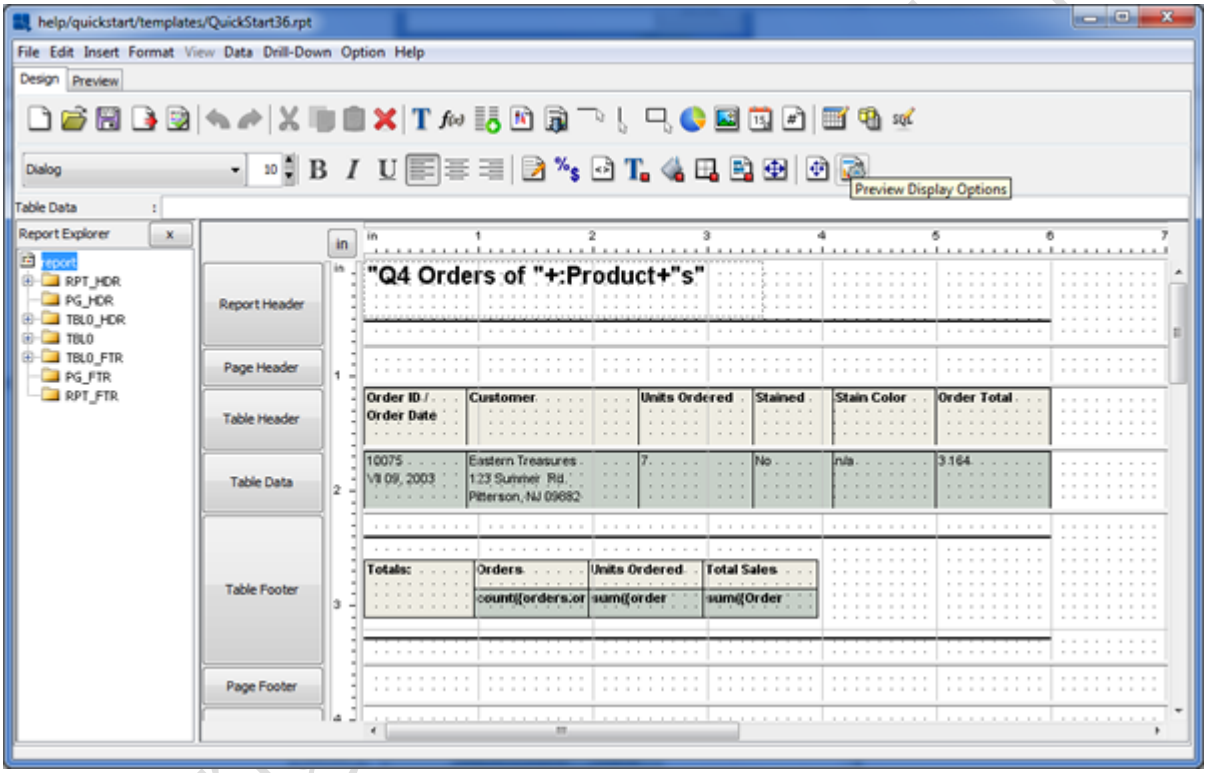
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

のパラメータが表示されます。



Second Parameter Mapping ダイアログ

Product Name 列をパラメータにマップする場合に選択し、**OK** をクリックします。新しいレイヤーの表示名を入力し、もう一度 **OK** をクリックしてデザインウィンドウで開きます。



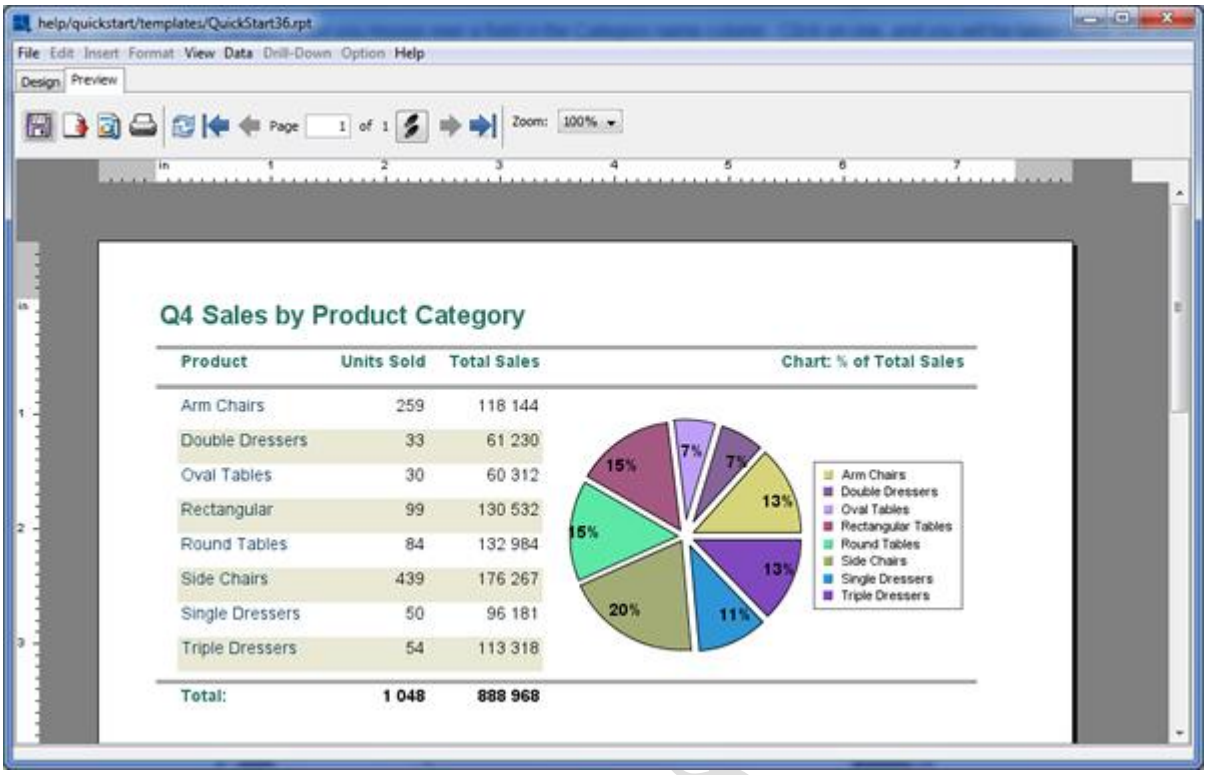
Report Designer の 2 番目のドリルダウンレベル

次に、**Drill-Down** > **Navigate** を選択します。今追加したレイヤーの新しいノードがあることに注目してください。ルートレポートを選択し、**EDIT** ボタンをクリックしてデザインウィンドウで開きます。最初のレポートがデザイナで表示されます。次に、**CLOSE** をクリックして、ナビゲーションダイアログを閉じます。

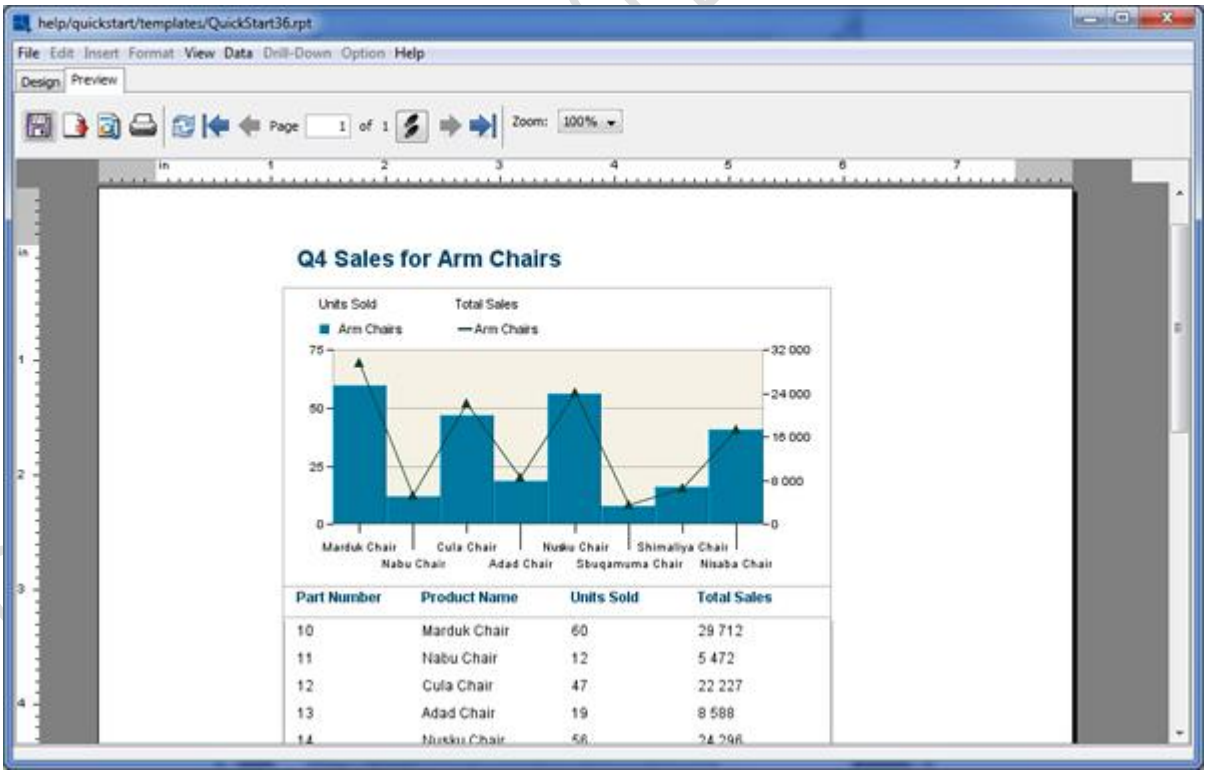
レポートをプレビューします。Category name 列のフィールドにマウスを重ねると、カーソルが変化することに注意してください。1 つをクリックすると、そのカテゴリの各製品の売上を示す次のレベルに移動します。

製品レポートでは、**Product name** 列のフィールドをクリックして、特定の製品の注文を示す第 3 レベルのレポートに移動できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

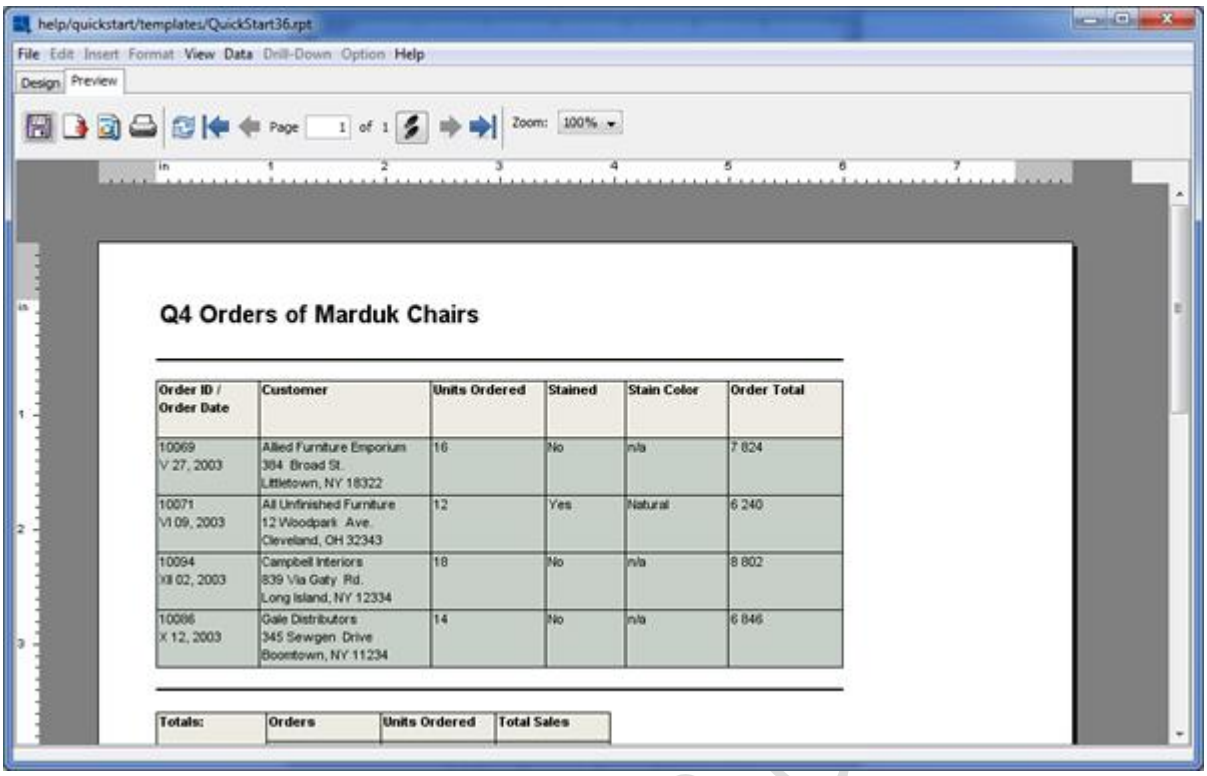


ドリルダウンレベル 1



ドリルダウンレベル 1

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



ドリルダウンレベル 1

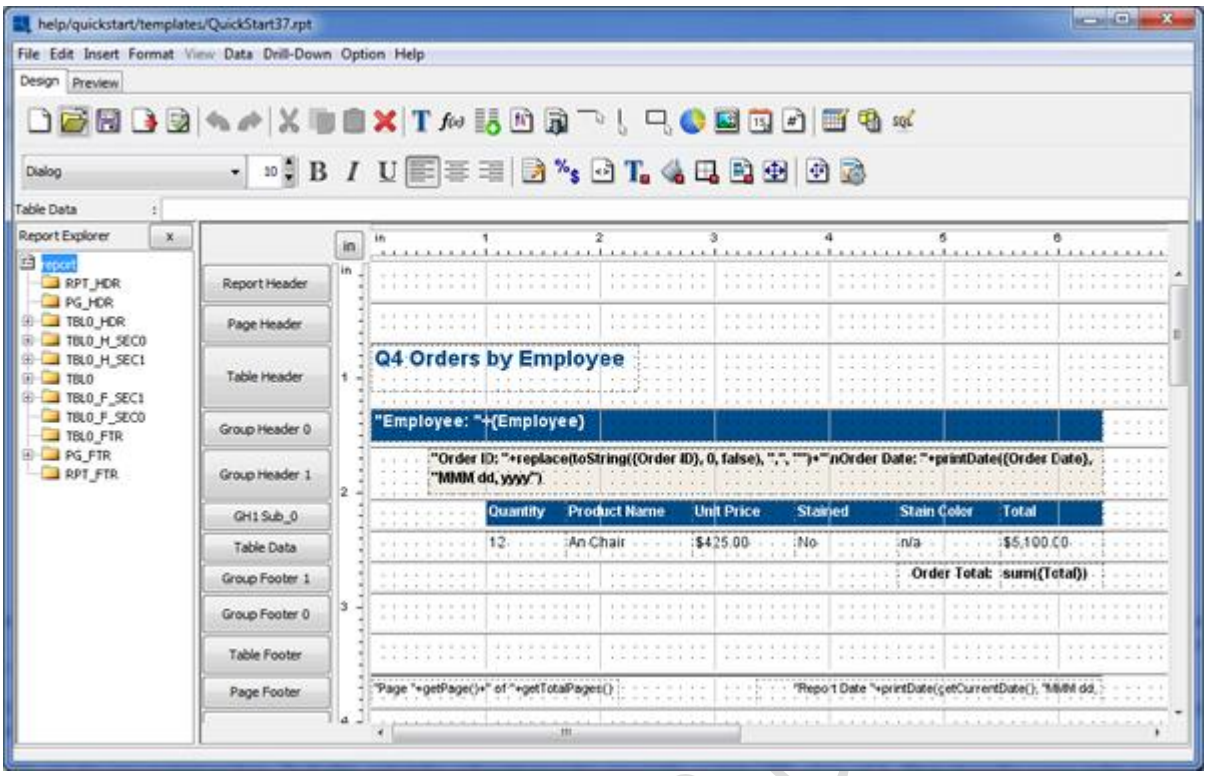
上位レイヤーに戻るには、右クリックし、ポップアップメニューから **Back** を選択します。ドリルダウンレポートの展開方法については、ドリルダウンの展開/エクスポートを参照してください。

2.3.7 サブレポート

EspressReport のもう一つの強力な機能は、サブレポートを使用してより複雑なレポートレイアウトを作成し、レポート内の複数のソースからのデータを結合する機能です。サブレポートの詳細については、デザイナーガイドのサブレポートを参照してください。

デザインウィンドウから、**Open** ボタンをクリックします。次に、**<InstallDir>/help/quickstart/templates** ディレクトリを参照します。一度そこに **QuickStart37.rpt** ファイルを選択して、それを開くように選択します。レポートがデザインウィンドウに表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



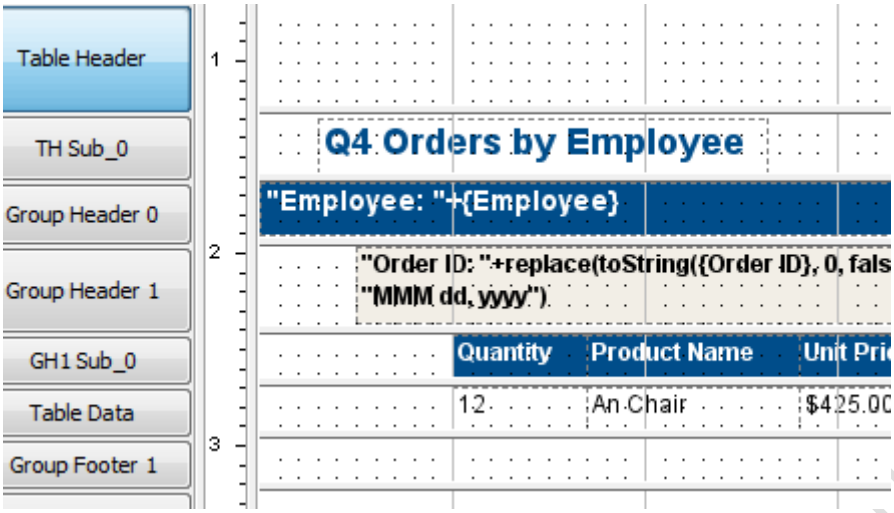
デザイナの QuickStart37.rpt

このレポートでは、ネストされたグループ化の 2 つのレベルを使用して各従業員の売上を示しています。ヘッダーにサブレポートを追加して、カテゴリと従業員による集計売上をクロス集計レイアウトで表示するようになりました。サブレポートを追加する前に、サブレポートを配置する新しいレポートセクションを作成します。サブレポートはレポートのどこにでも配置することができますが、サブレポートのサイズが固定されていない場合は特に、サブレポートに独自のセクションを付けるのが適切です。

ネストされたセクションを挿入するには、左側の **Table Header** ボタンをクリックして、セクションオプションメニューを表示します。**Insert Section.**を選択します。これは、テーブルヘッダーのネストされたセクションを生成します。ネストされたセクションの詳細については、デザイナガイドのネストされるセクションを参照してください。

次に、タイトルを新しいネストされたセクションに移動します。これを行うには、レポートタイトルを含むセルを選択し、**CTRL + X** を押してカットします。次に、新しいセクションにカーソルを置き、**CTRL + V** を押して貼り付けます。カーソルが十字に変わります。フィールドを希望する場所に配置し、クリックして追加します。セルに合わせて新しいセクションのサイズを変更することができます。

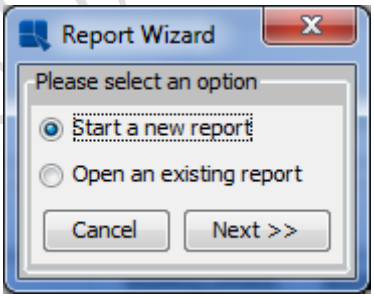
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



新しいレポートセクション

次に、ツールバーの **Insert Sub-Report** ボタン をクリックします。続行する前に、変更をレポートに保存するように求められます。これを行うと、マウスポインタが十字に変わります。SubReport を挿入するには、Table Header セクションの左上隅にあるマウスの左ボタンを押します。

ドリルダウンと同様に、サブレポートまたは既存のテンプレートを使用して新しいレポートを作成することもできます。ただし、ドリルダウンとは異なり、テンプレートはデータソースとしてパラメータ化されたクエリを持つ必要はありません（リンクされたサブレポートを作成する場合を除きます)。詳細については、デザイナガイドのリンクされたサブレポートを参照してください。)。既存のテンプレートを開き、**Next** >>をクリックします。

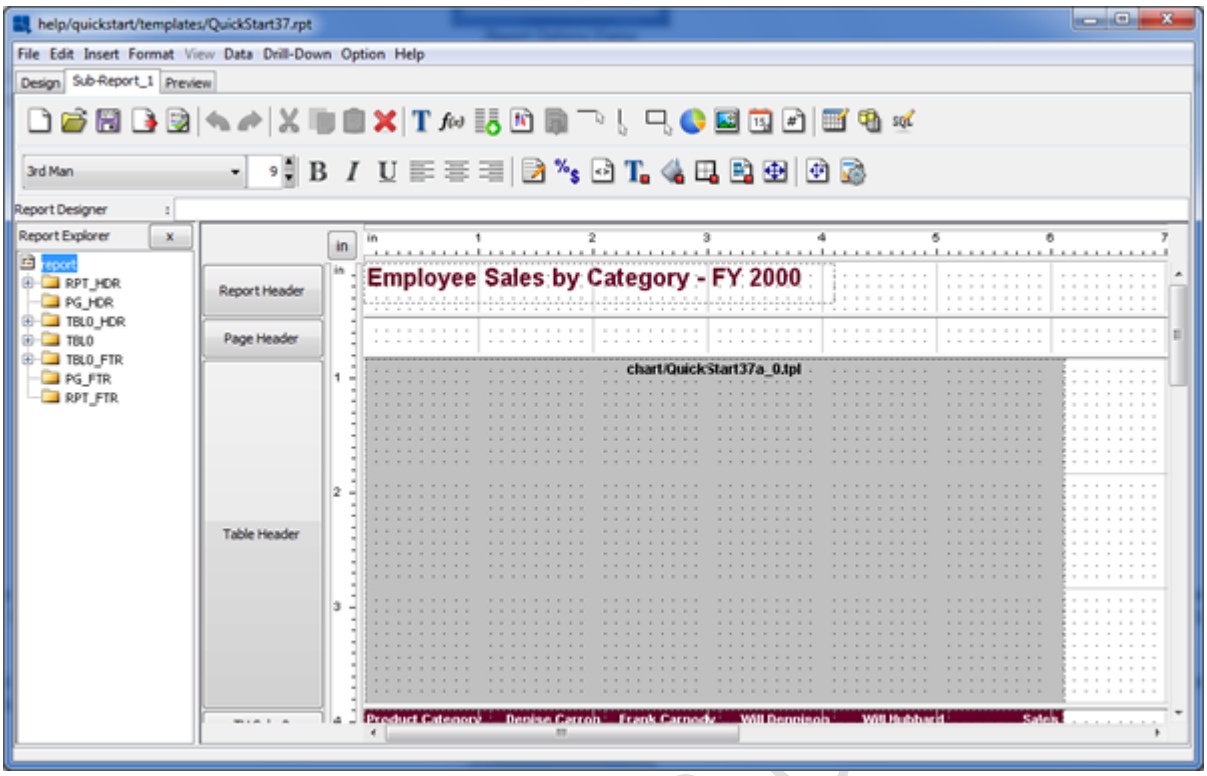


Report Options ダイアログ

プロンプトが表示されたら、**<InstallDir>/help/quickstart/templates** ディレクトリを参照します。そこに QuickStart37a.rpt ファイルを選択し、**Open** をクリックします。サブレポートは、Report Designer の Sub-Report_1 という新しいタブで開きます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



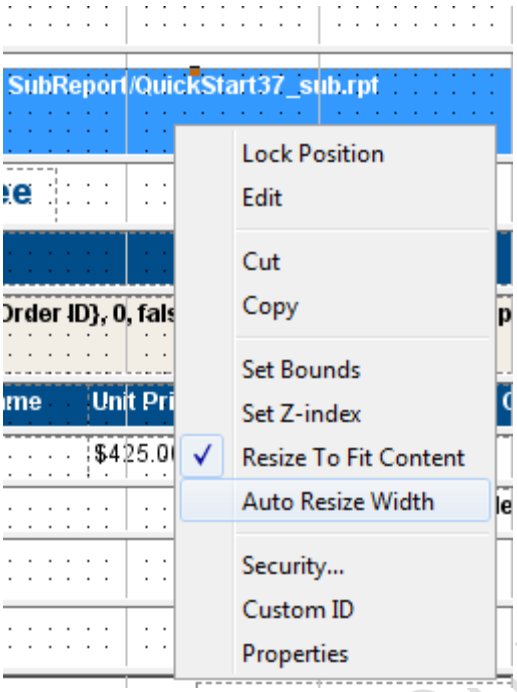
Designer のサブレポート

Preview タブと **Sub-Repor** タブを切り替えることで、サブレポートだけをプレビューできます。次に、**Design** タブをクリックしてメインレポートに戻ります。サブレポートは小さな灰色の四角形で表示されます。四角形をセクションの左上隅に移動し、横のルーラーをクリックしてドラッグすると、サブレポートの幅が約 7 インチに広がります。

次に、サブレポートを動的にサイズ変更するように設定します。サブレポートを右クリックし、ポップアップメニューの **Resize to fit Content** が選択されていることを確認します（オプションの横にティッカーが表示されます）。

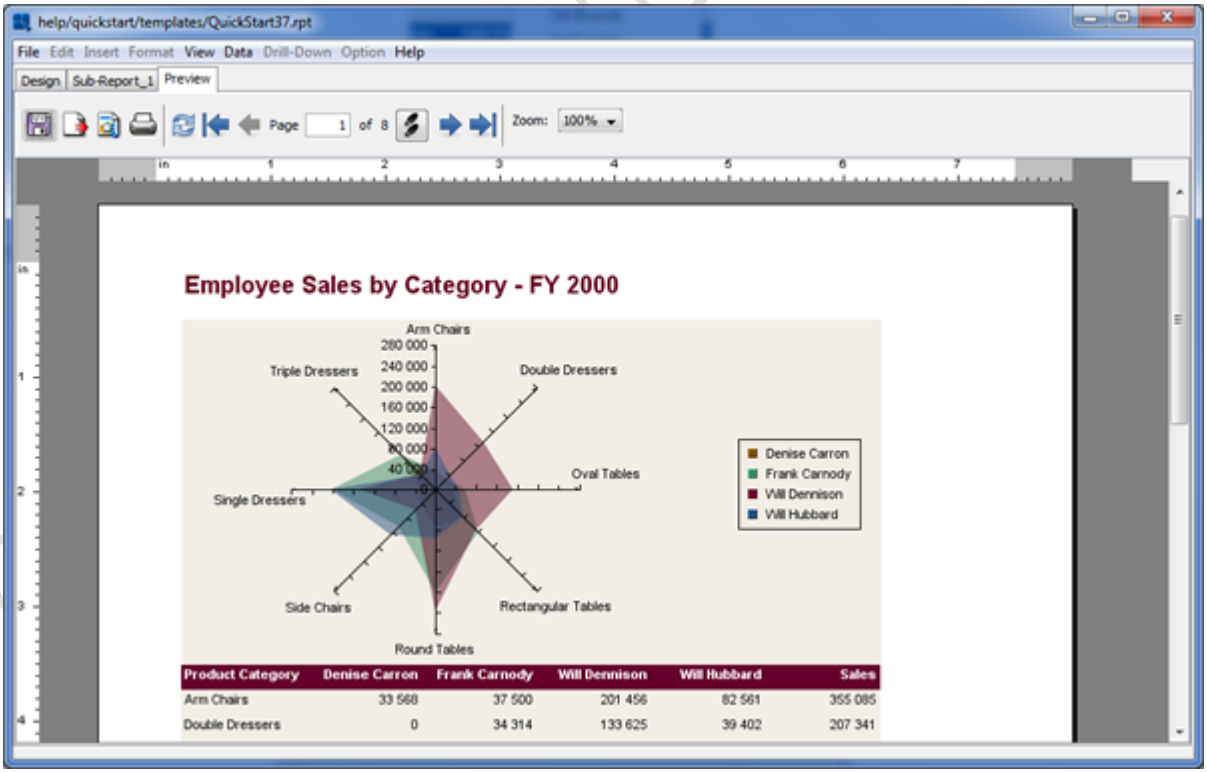
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



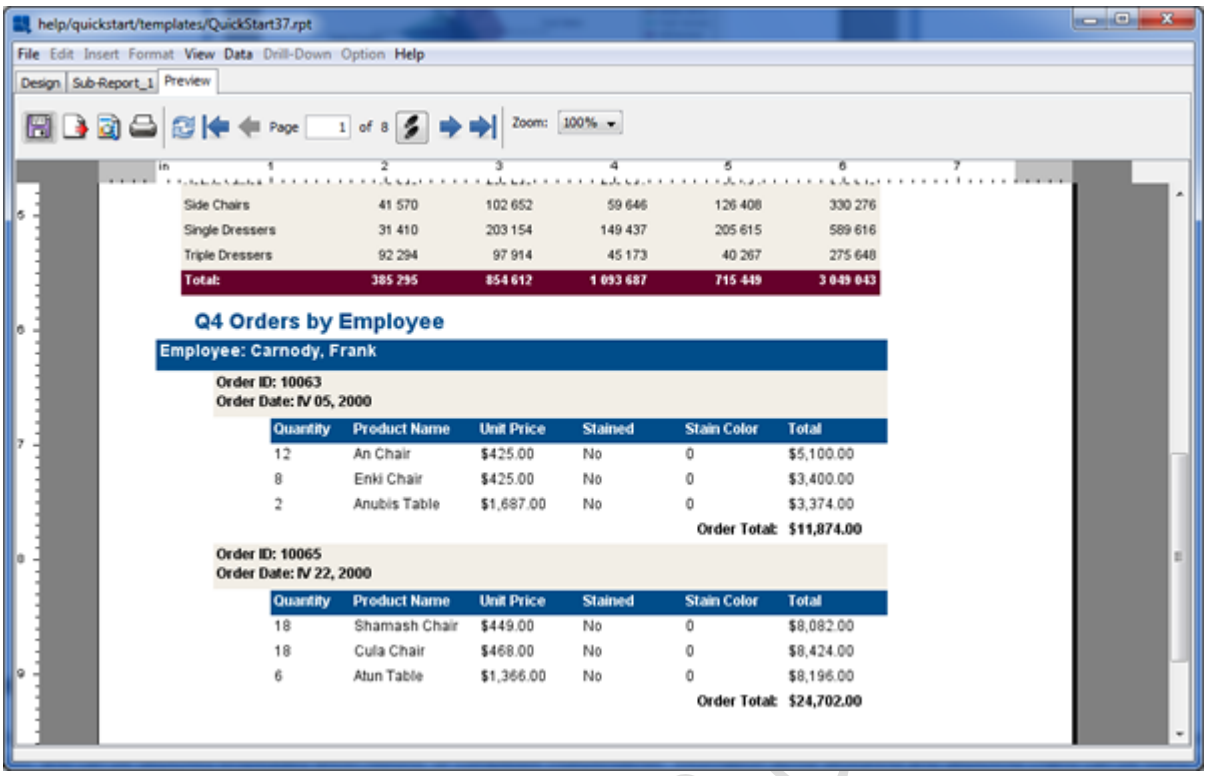
Resize to Fit Content ダイアログ

レポートをプレビューします。サブレポート全体がメインレポートの前に実行されていることがわかります。



プレビューのメインレポート

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



プレビューでメインレポートを含むサブレポート

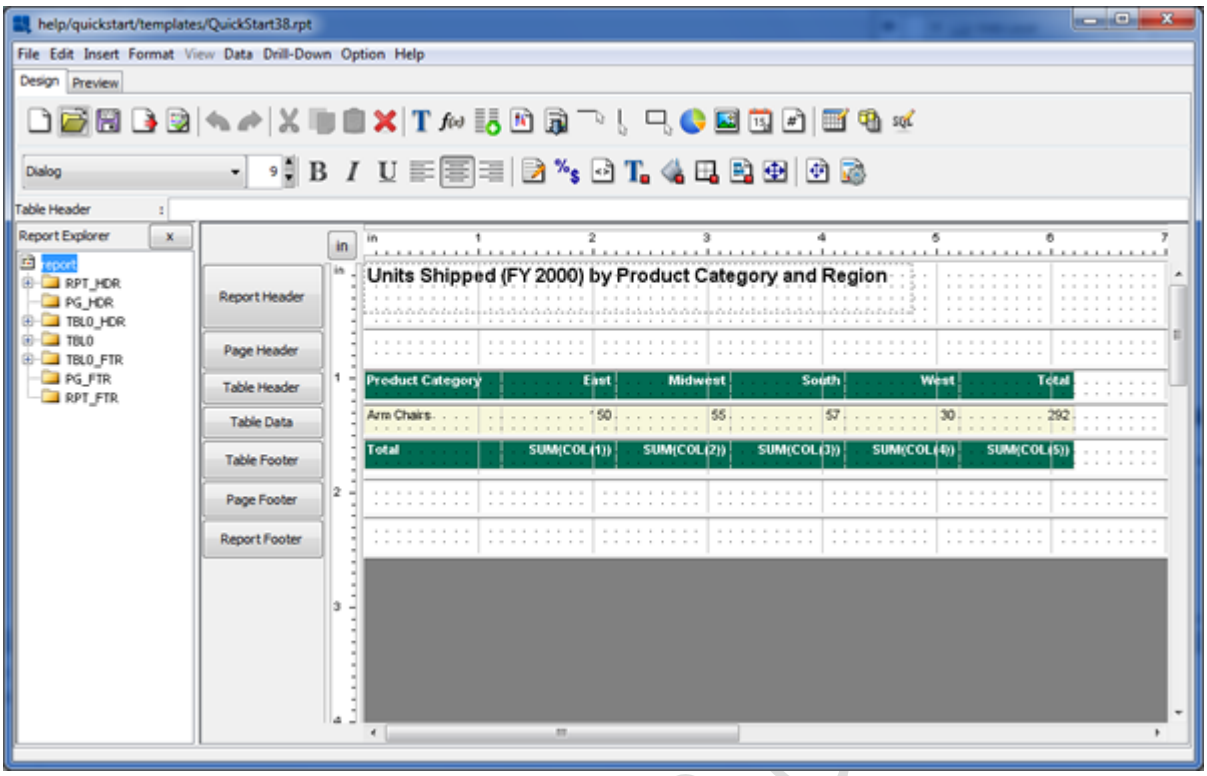
2.3.8 チャートの操作

EspressReport には、30 以上の異なる 2 次元および 3 次元チャートタイプでデータをプロットするための広範囲なチャートライブラリが含まれています。チャートは独立したデータソースを使用することができ、レポート内に配置/埋め込み、または完全に独立して配置できます。チャートの設計と配備に関する情報は、チャート作成ガイドを参照してください。

この例では、クロス集計レポートにグラフを追加します。デザインウィンドウで、ツールバーの **Open** ボタン をクリックします。次に、**<InstallDir>/help/quickstart/templates** ディレクトリを参照し、QuickStart38.rpt ファイルを選択して開きます。レポートウィンドウが開きます。

Copyright(C) 2018

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



デザインウィンドウで QuickStart38.rpt

2.3.8.1 グラフと地図データを挿入する

このレポートでは、テーブルフッターのサマリの下にグラフを追加します。まず、グラフのためのスペースを確保する必要があります。テーブルフッターを右クリックし、ポップアップメニューから **Insert Section** を選択して新しいセクションを挿入します。それからそれを拡大して約 4 インチのスペースを追加します。次に、ツールバーの **Insert Chart** ボタンをクリックします。小さな点線の四角形が、デザインウィンドウの周りのポインタの後に表示されます。Table Footer セクションの[Total]ラベルの下に配置し、をクリックします。チャートデザイナは、チャートを生成して、チャートを生成します。

表示される最初の画面では、チャートのデータソースに関する情報を指定するよう求められます。グラフはレポートからデータを取得することも、独立したデータソースを持つこともできます。

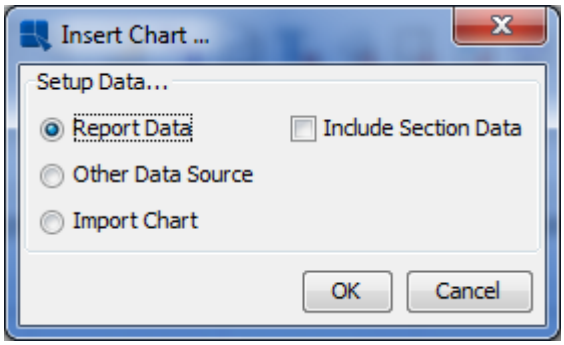


Chart Data Options

この場合、グラフのレポートデータを使用するように選択し、**OK** をクリックします。これにより、グラフが描画されるレポートデータを示す表が表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

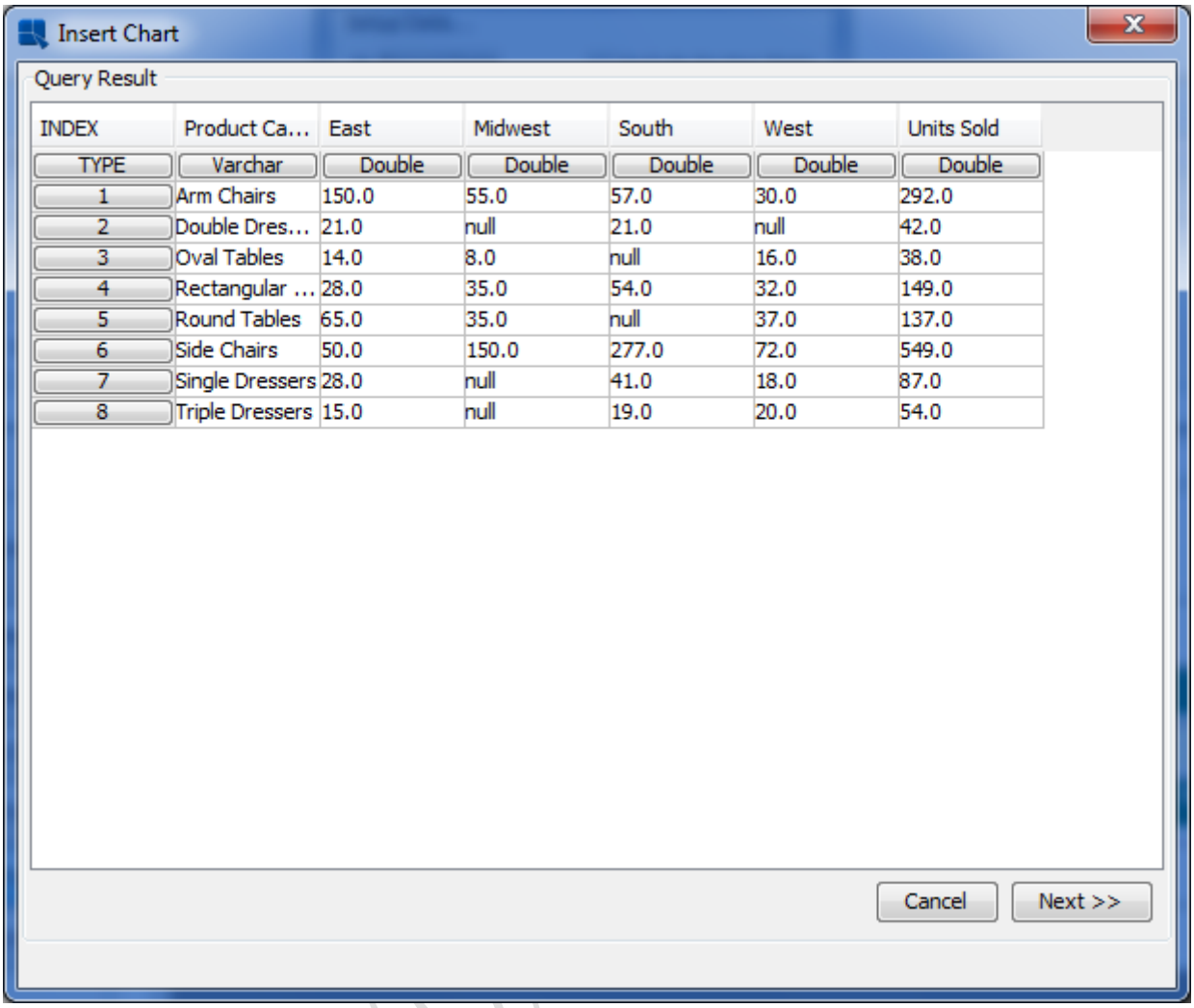


Chart Data ダイアログ

Next >>ボタンをクリックして、チャートウィザードの次のダイアログに進みます。次の画面では、グラフの種類を選択できます。ラジオボタンを使用して、2 次元と 3 次元のグラフタイプを切り替えることができます。2 次元の縦棒グラフを選択し、**Next** >>ボタンをクリックします。

Copyright(C)

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

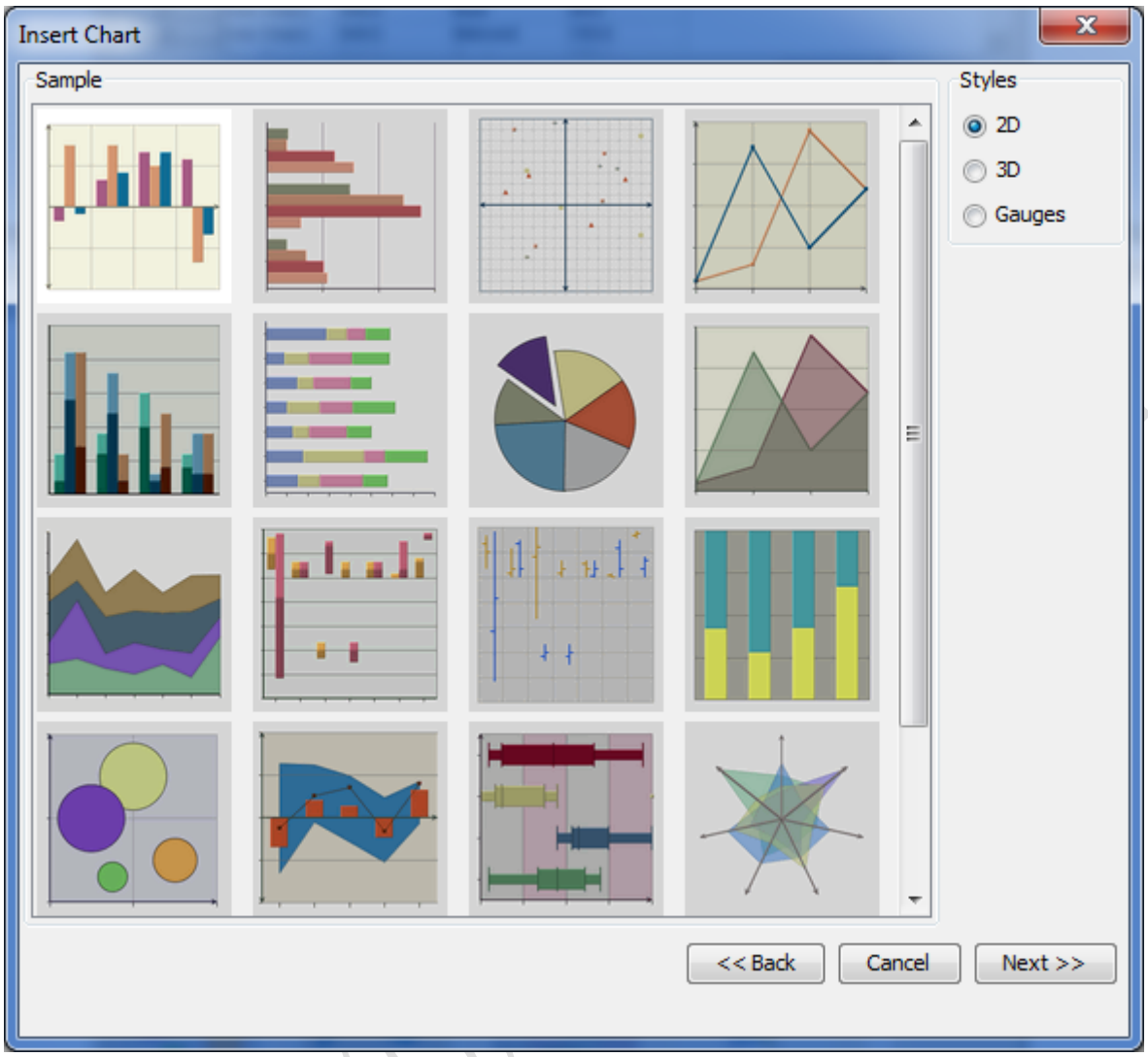


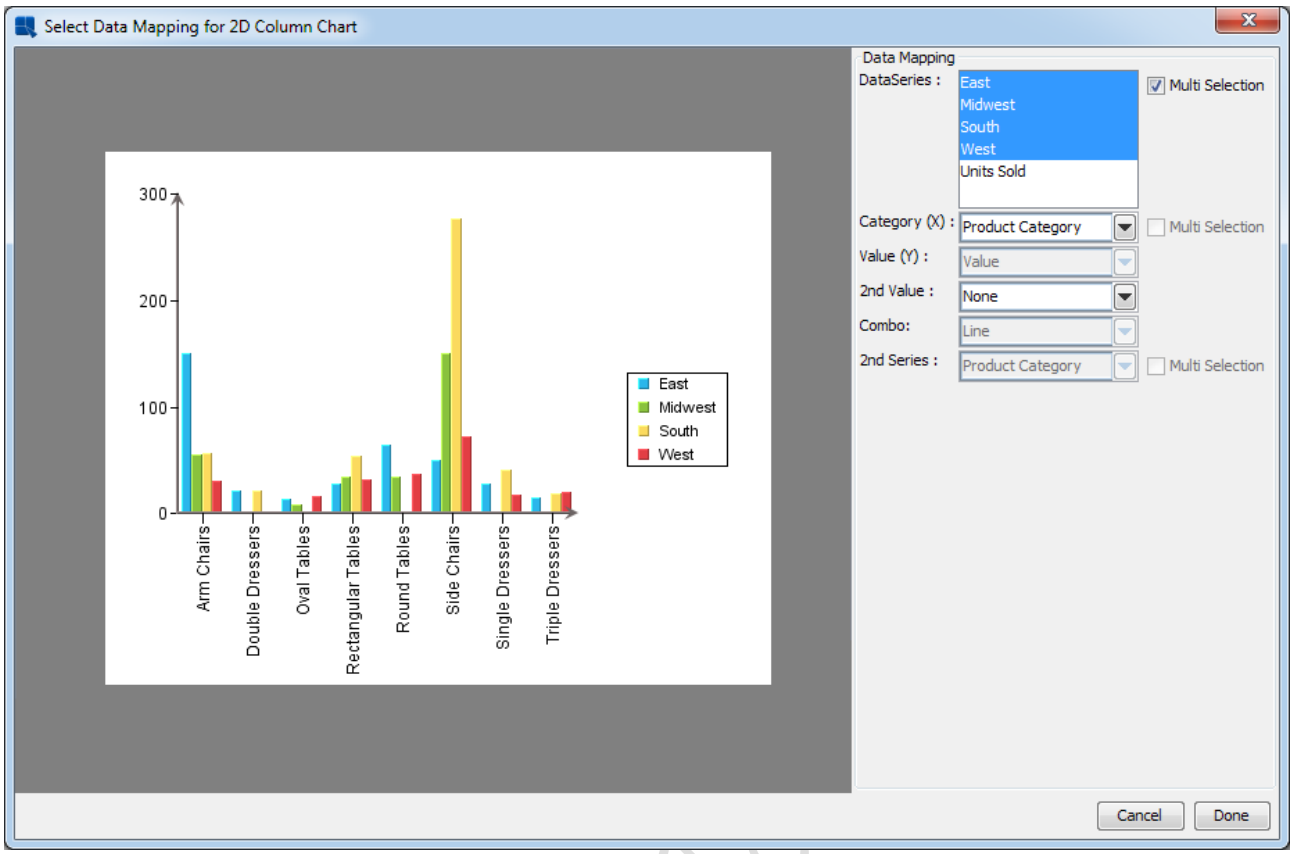
Chart Types ダイアログ

次の画面では、グラフのデータマッピングを設定できます。データマッピングは、データソースのデータの列がグラフの要素にマップされるプロセスです。**Data Series** フィールドの横にある **Multi Selection** オプションを選択します。フィールドは単一値から複数値選択ボックスに変更する必要があります。**Ctrl +クリック**で、次の 4 つの列をデータ系列として選択します。

East
Midwest
South
West

Category（X）オプションを「Product Category」に設定します。ダイアログは次のようになります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Data Mapping ダイアログ

マッピングオプションの設定が完了したら、**Done** ボタンをクリックすると、チャートをカスタマイズできる Chart Designer ウィンドウが表示されます。

2.3.8.2 チャートのプロパティをカスタマイズする

チャートで最初に修正するのは、キャンバスのサイズです。これは、完成したグラフのサイズを制御します。これを行うには、**Format** > **Canvas** を選択します。これにより、グラフキャンバスを変更できるダイアログが表示されます。

Copyright(C) 2018

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

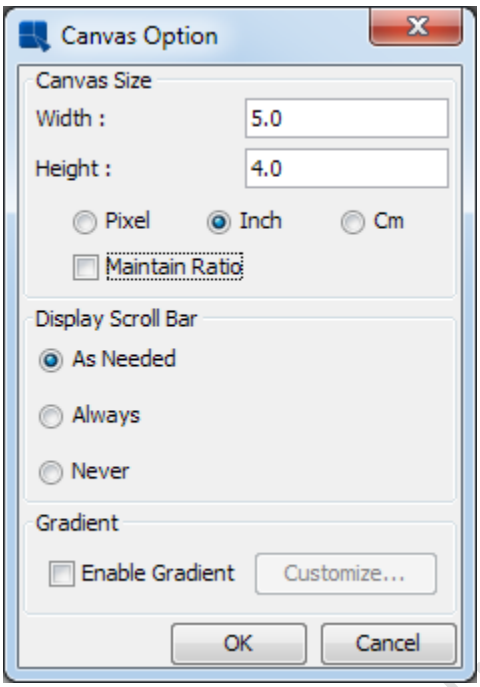


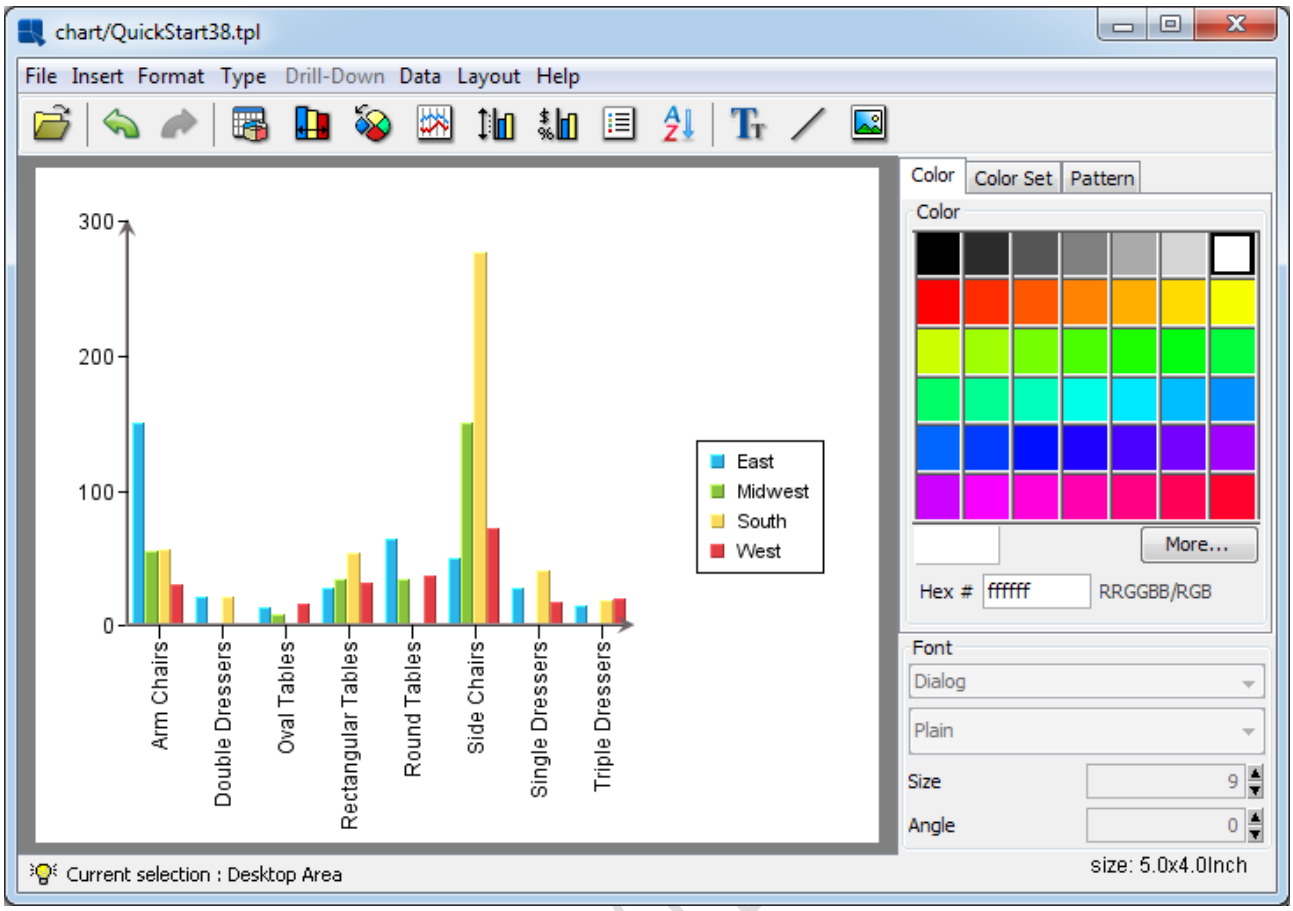
Chart Canvas ダイアログ

ボタンを選択して測定値をインチで指定し、**Maintain Ratio** オプションの選択を解除し、新しいチャートキャンバスサイズを 5x4 インチに設定します。**OK** をクリックします。これにより、チャートデザイナのビュー部分でキャンバスのサイズが変更されます。

チャートキャンバスのサイズが変更されたので、グラフプロットは小さく表示され、X 軸ラベルの一部は切り捨てられます。これは、チャートのサイズを変更することで調整できます。グラフのプロットをクリックしてドラッグすると移動したり、右クリックしてドラッグするとサイズを変更することができます。また、クリックしてドラッグすると、凡例を移動することもできます。これらのオプションを使用して、チャートプロットと凡例をキャンバス上に配置します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

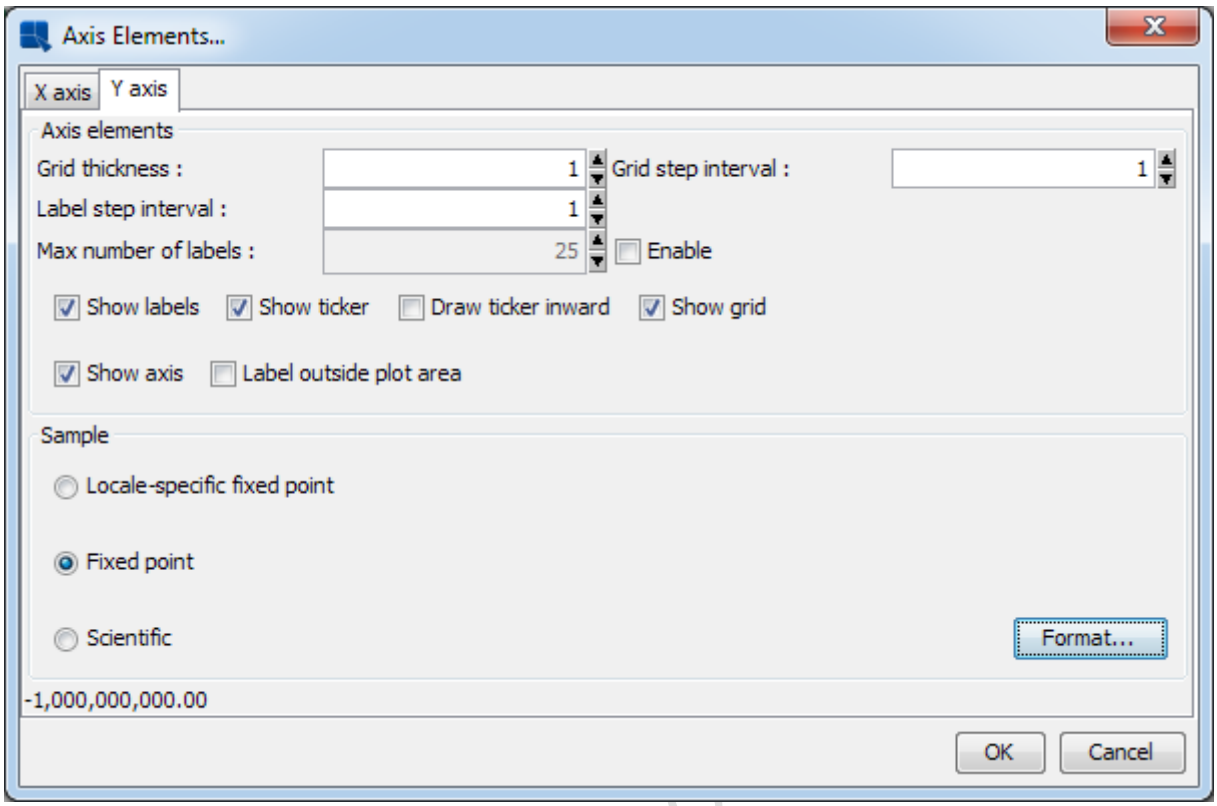


配置されたチャートと凡例を持つチャートデザイナ

次に、軸ラベルの書式を変更することができます。これを行うには、ツールバーの **Axis Elements** ボタン をクリックします。これにより、各チャート軸に異なるオプションを設定できるタブ付きのダイアログが表示されます。**Y Axis** タブをクリックすると、値軸のオプションが表示されます。

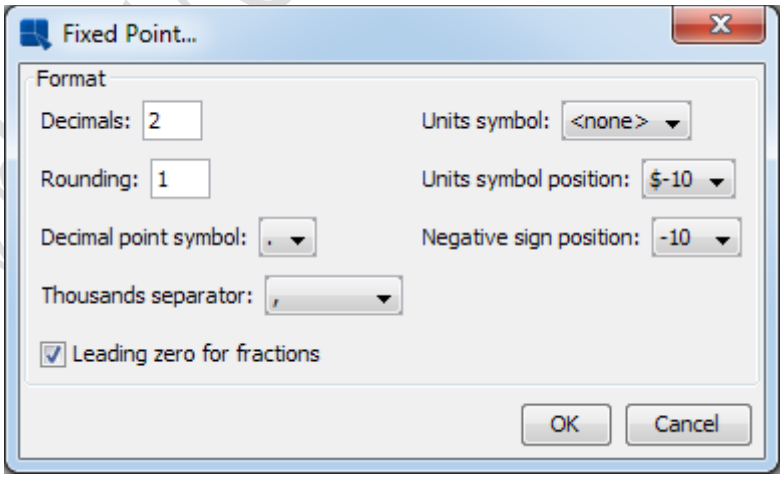
Copyright(C) 2018

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Axis Elements ダイアログ

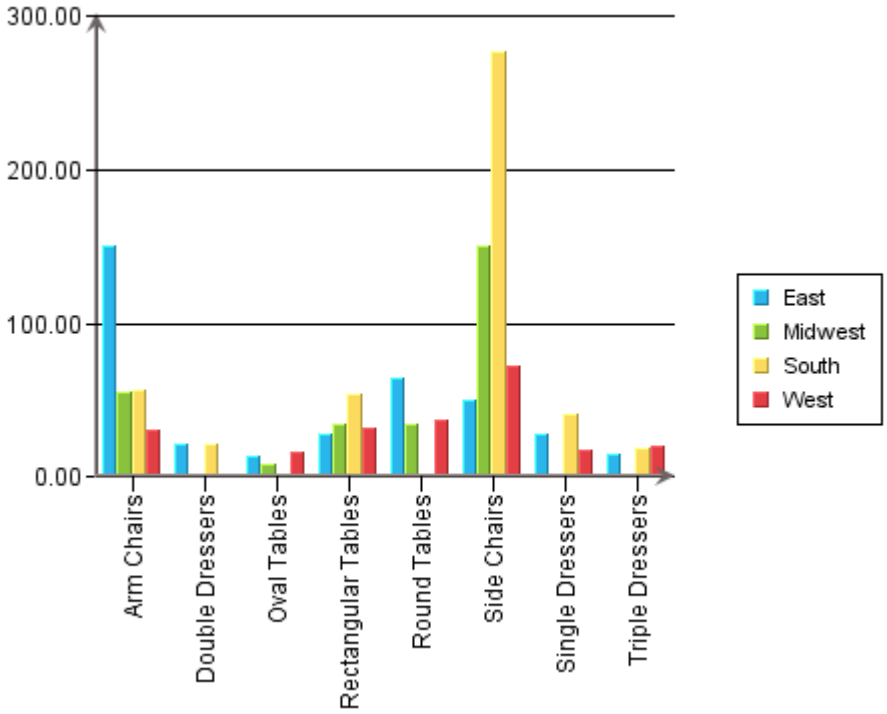
Y 軸にグリッド線を追加するには、**Show grid** のボックスをオンにします。次に、データフォーマットとして **Fixed point** を選択し、**Format** ボタンをクリックします。これにより、数値データの書式オプションを設定できる追加のダイアログが表示されます。小数点 2 として 2 を入力し、**OK** をクリックします。



Value Axis Fixed Point Formatting ダイアログ

もう一度 **OK** をクリックして軸要素ダイアログを閉じると、指定した変更がグラフに反映されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

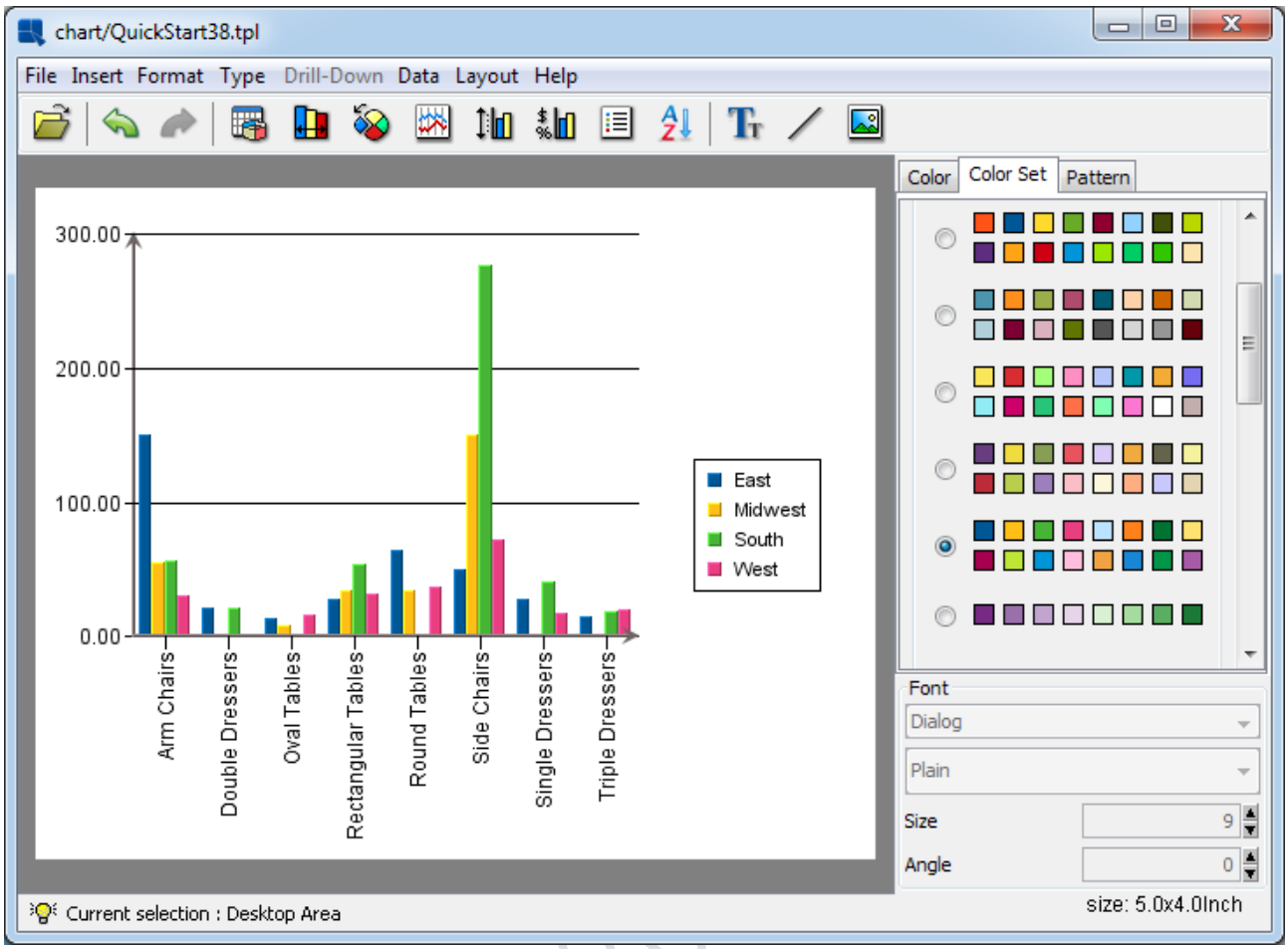


軸要素の書式設定後のグラフ

チャート要素の色を変更するには、プリセット **Color Set** の 1 つを使用します。Chart Designer ウィンドウの右側にある **Color Sets** タブをクリックし、任意のカラーセットを選択します。これにより、データ要素に設定された色が適用されます。任意の要素の色をクリックして選択することもできます（ヒント：現在選択されている要素が左下隅に表示されます)。カラーパネルから新しい色を選択します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



カラーセットをピックする

次に、チャートにタイトルを追加することができます。これを行うには、**Insert** > **Titles** を選択します。これにより、グラフのタイトルと各軸のタイトルを入力できるダイアログが表示されます。

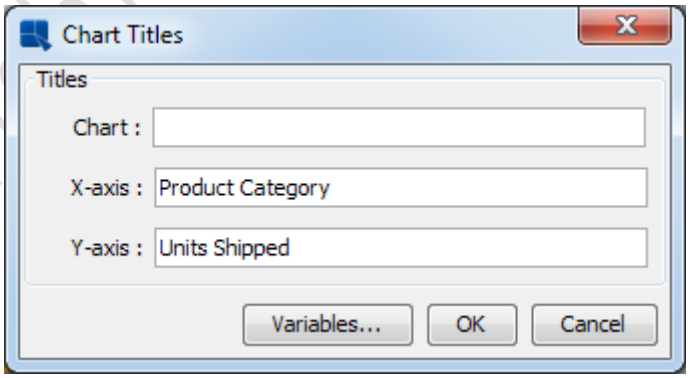
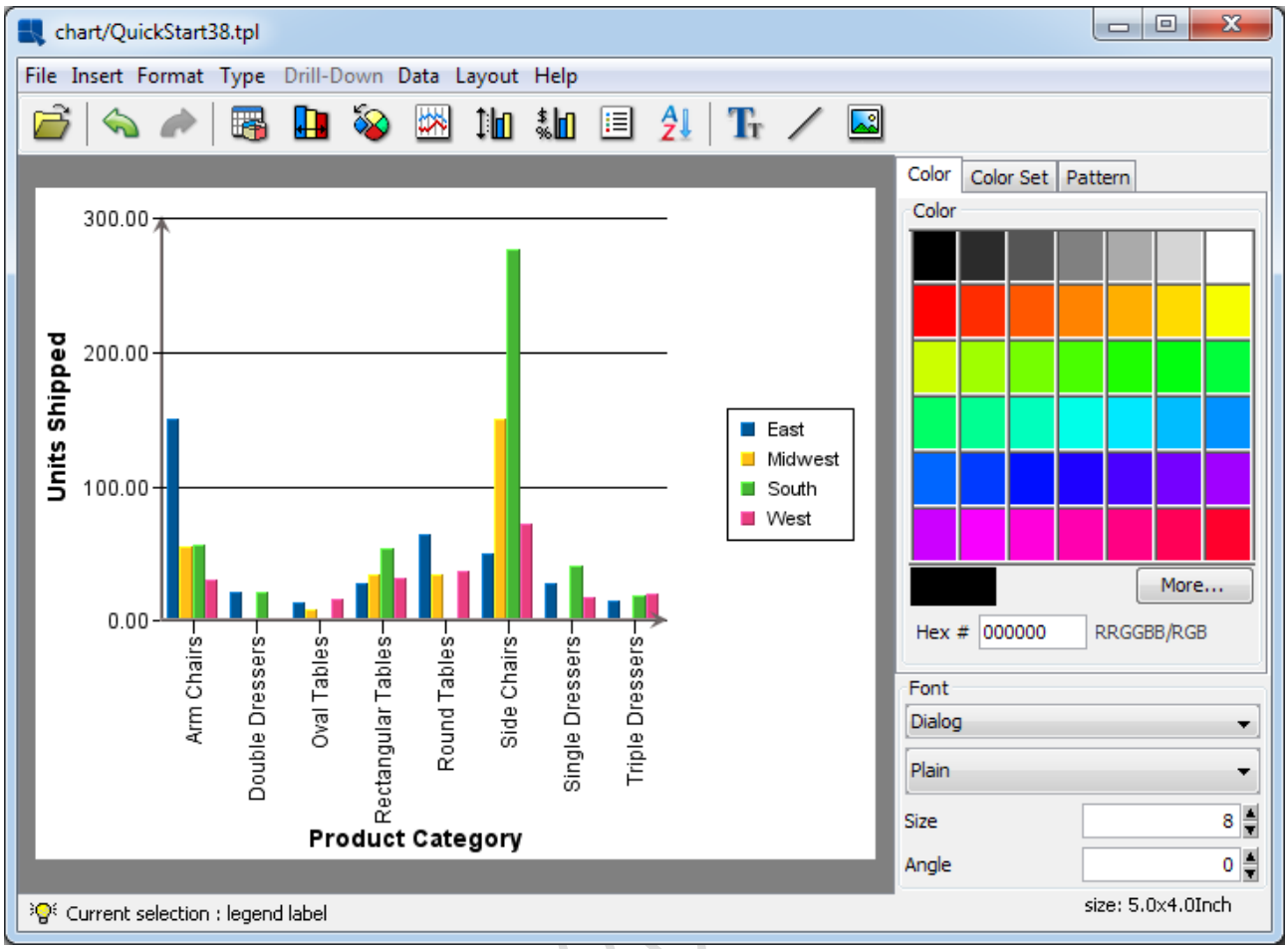


Chart Titles ダイアログ

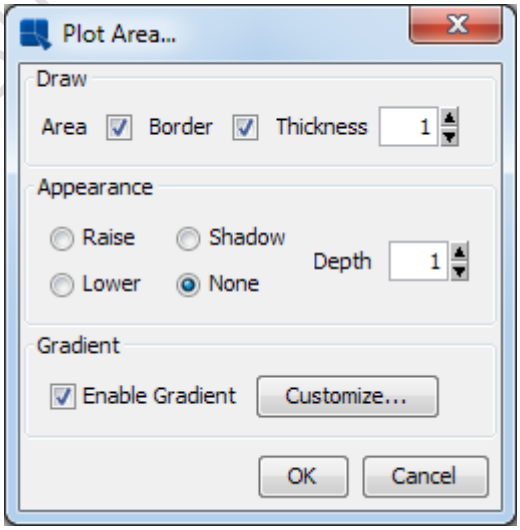
メインタイトルを空白のままにして、X 軸と Y 軸のタイトルを入力します。タイトルがチャートに追加されます。タイトルは自動的に配置されますが、チャートキャンバス上のテキストをクリックしてドラッグすることで、位置を手動で調整する必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



軸のタイトルがあるチャート

最後に、プロットの背景と境界線を追加して、プロット領域の外観をカスタマイズすることができます。これを行うには、**Format** > **Plot Area** を選択します。これにより、グラフプロットの表示オプションを設定できるダイアログが表示されます。

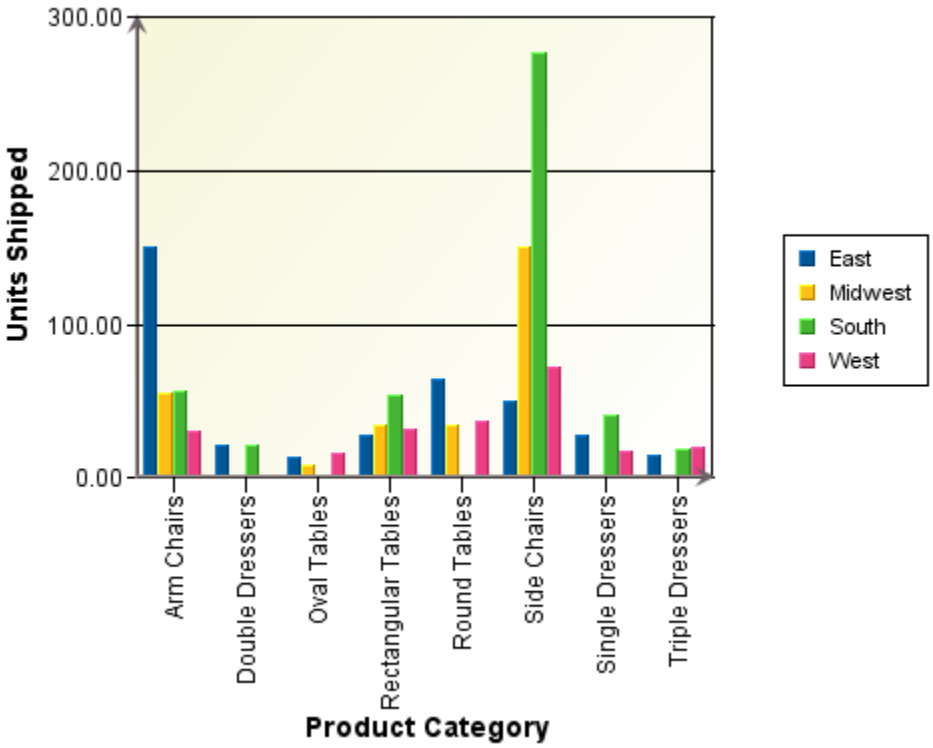


Plot Area ダイアログ

プロット領域と境界線の両方を 1 の厚さで描画する場合に選択します。外観に **None** を指定します。グラデーションを有効にするオプションを選択します。すべてのオプションを指定したら、**OK** をクリッ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

クすると、グラフのプロット領域が変更されます。プロット領域の背景色をクリックして選択し、カラーパネルで色を変更することで、プロット領域の背景色を変更することができます。

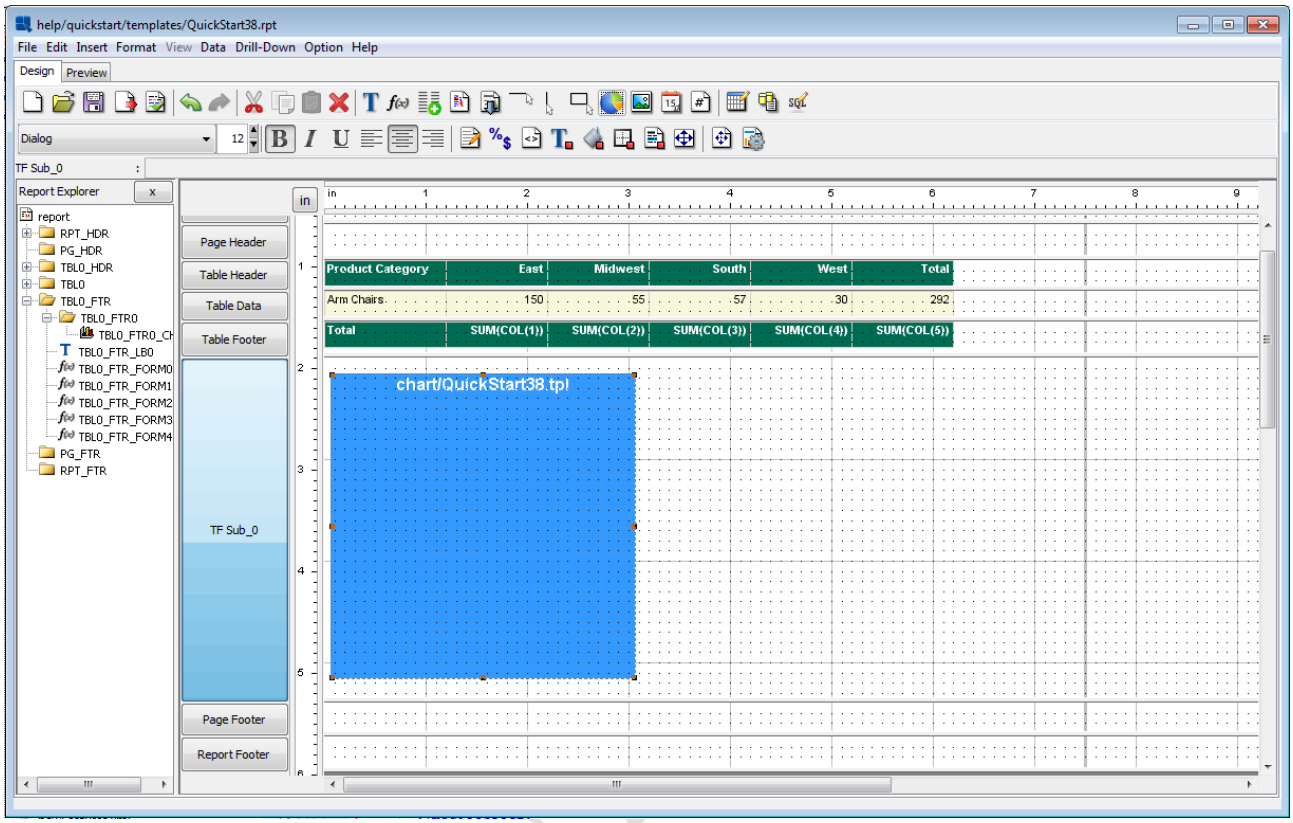


プロットエリアを含むチャート

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次に、ツールバーの **Save** ボタン をクリックして行った変更を保存します（グラフはレポートデータを使用するため、自動的に/**chart**/ディレクトリに保存されます）。次に、**File** > **Exit**.を選択してグラフデザイナを終了します。これにより、灰色の四角形がグラフを表すデザインウィンドウに戻ります。

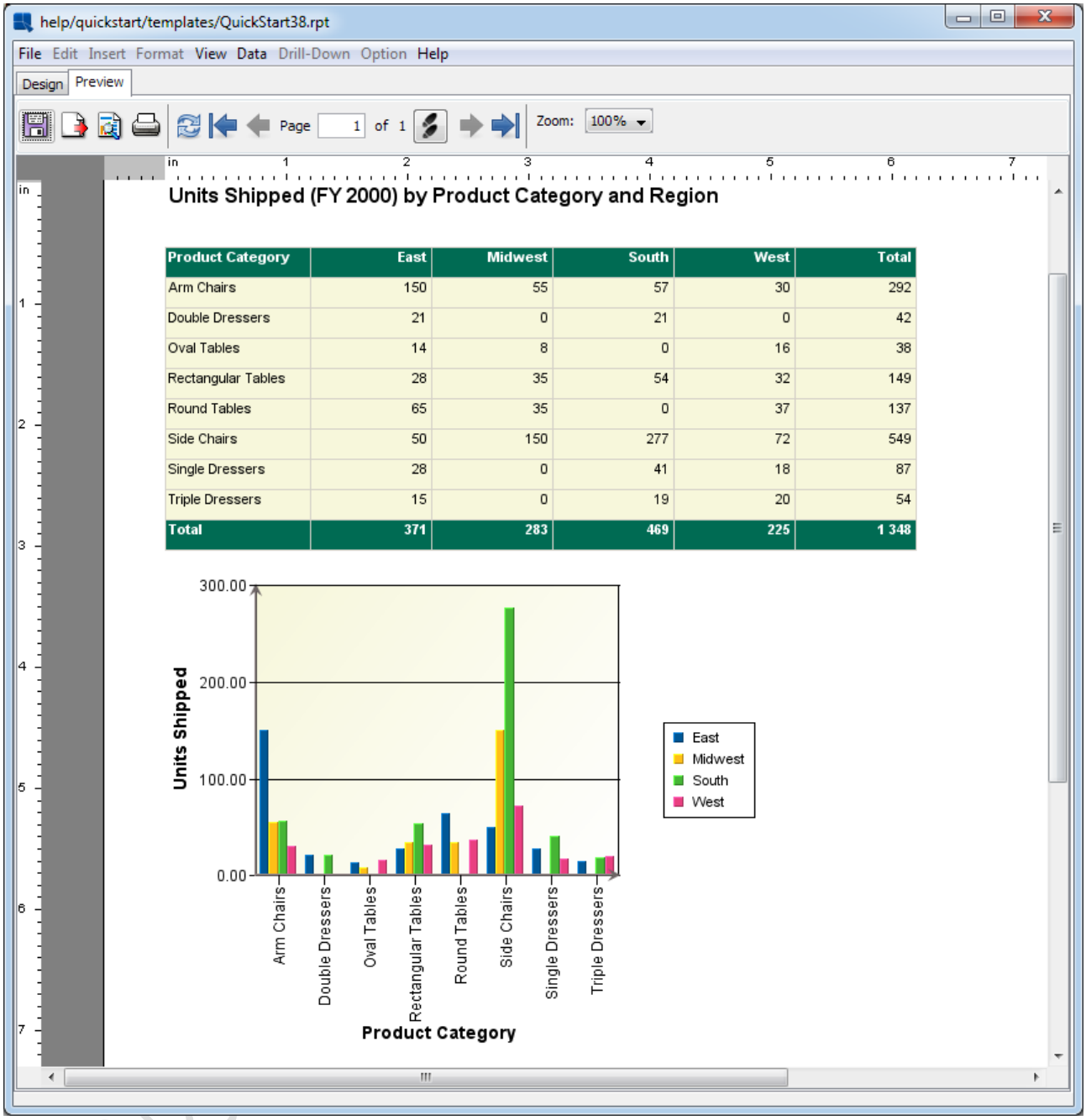


Report Designer のグラフ

Copyright(C)

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

レポートをプレビューします。表の下にグラフが表示されます。



チャートを使用したレポートのプレビュー

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2.4 API のクイックスタート

このセクションでは、Report API の基本機能の一部を説明するための一連の短いチュートリアルを示します。このセクションで説明されている機能の詳細については、ドキュメントの「プログラミングガイド」を参照してください。

2.4.1 環境を設定する

このガイドに記載されているコードは、次の点を考慮して作成されています：

- EspressManager が起動しています。
- Tomcat はサーブレットコンテナと Web サーバとして使用されています。
- MySQL Woodview データベースがセットアップされています。**<EspressReport InstallDir>/help/examples/DataSources/database** ディレクトリにある **Woodview_mysql.sql** という名前の Woodview MySQL ダンプファイル。ユーザはダンプファイルを再ロードして、環境内で MySQL Woodview データベースを作成することができます。これは、MySQL 接続でテンプレートを使用したいユーザにのみ必要です。

このガイドに記載されているコードは**<EspressReport InstallDir>/help/quickstart/src** ディレクトリにあり、Java アプリケーションクラスファイルは **<EspressReport InstallDir>/help/quickstart/classes** ディレクトリにあります。どの HTML ファイルも**<EspressReport InstallDir>/help/quickstart/html** ディレクトリにあります。同様に、どのテンプレートも **help/quickstart/templates** ディレクトリの下にあります。

チュートリアルを実行するために、ほとんどのファイルを移動する必要はありません。ファイルを移動および/または変更する必要があるいくつかの例で指示が与えられる。

クイックスタート API ガイドを正しく使用するには：

- EspressManager の CLASSPATH に **hsqldb.jar** を追加する必要があります(JDBC 接続のセットアップ)。
- **ReportAPIWithChart.jar**、**barbecue-1.5.jar**、**ReportDesigner.jar**、**qblicense.jar**、**ExportLib.jar**、および **hsqldb.jar**（**<EspressReport InstallDir>/lib** ディレクトリにあります）を CLASSPATH に追加する必要があります。さらに、CLASSPATH に **servlet.jar**（**<Tomcat InstallDir>/common/lib** ディレクトリにあります）も必要です。
- **ReportAPIWithChart.jar**、**ReportDesigner.jar**、**qblicense.jar**、**ExportLib.jar**、**hsqldb.jar** も、Tomcat サーバの CLASSPATH に追加する必要があります。

データベースをデータソースとして使用するテンプレート（本書で参照）は、HSQL Java データベースを使用します。MySQL Woodview データベースを使用するユーザには、代替テンプレートが用意されています。これらのテンプレートは、**<EspressReport installdir>/help/quickstart/templates/MySQL** ディレクトリにあります。テンプレートは、".rpt"（二重引用符なし）拡張の前に"_mysql"（二重引用符なし）とともに上記で指定した命

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

名規則に再び従います。

使用している Tomcat サーバのバージョンに応じて、サーブレットコンテキストはさまざまな方法で定義されます。

- Tomcat 6.x 以前のバージョンでは、サーブレットのコンテキスト/**servlet**/がコメントアウトされている可能性があります。<**Tomcat InstallDir**>/**conf**/**web.xml** にチェックを入れ、次の行のコメントを外してください：

```
<servlet-mapping>
    <servlet-name>invoker</servlet-name>
    <url-pattern>/servlet/*</url-pattern>
</servlet-mapping>
```

かつ

```
<servlet>
        <servlet-name>invoker</servlet-name>
        <servlet-class>
                org.apache.catalina.servlets.InvokerServlet
        </servlet-class>
        <init-param>
                <param-name>debug</param-name>
                <param-value>0</param-value>
        </init-param>
        <load-on-startup>2</load-on-startup>
</servlet>
```

文のブロックは、冒頭に<**!** **-**、最後に**->**があるとコメントになります。したがって、上記のブロックがコメントアウトされている場合は、コメントを外して Tomcat サーバを再起動してください。

- Tomcat 7.x 以降のバージョンでは、サーブレットのコンテキストは、<**Tomcat InstallDir**>/**webapps**/**ROOT**/**WEB-INF** ディレクトリの **web.xml** に明示的にマップする必要があります。以下の例では、**web.xml** でこれらの例で使用されるサーブレットを構成する方法を示します。

<**classes**>（二重引用符なし）は、<**Tomcat installDir**>/**webapps**/**ROOT**/**WEB-INF** ディレクトリにも存在する必要があります。ディレクトリが存在しない場合は、作成して Tomcat サーバを再起動してください。

コード実行中に、「Failed to read espressmanager.cfg. Use default host and port no. 22071.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

（espressmanager.cfg を読み込めませんでした。デフォルトホストとポート番号 22071 を使用してください。）」というメッセージが表示されることがあります。（二重引用符なし）。これはエラーメッセージではありません。このメッセージは、**espressmanager.cfg** が見つからなかったことと、コードと同じマシン上で実行されている EspressManager を検索し、ポート番号 22071 を使用することを示しています。

EspressReport API は EspressManager なしで、他のアプリケーションサーバでも使用できますが、このガイドのコードは EspressManager と Tomcat を念頭に置いて設計されています。他の設定に切り替えるには、プログラミングガイド（EspressManager なしでのデプロイとサーブレットと Java Server Pages）を参照してください。

このコードは、すべて（つまり、EspressManager、Tomcat サーバ、およびクライアント）が同じマシン上にあるという考えで実装されています。このコードはシングルスレッドであり、デモンストレーション目的でクライアントとサーバで同じマシンを使用するように設定されています。EspressReport API を使用して、マルチクライアントサーバ環境でマルチスレッドコードを生成できます。マルチマシン環境で使用するには、コード（および付随する HTML ファイル）を編集する必要があります。詳細については、プログラミングガイド（EspressManager でデプロイする）を参照してください。

2.4.2 レポートを実行する

以下のセクションでは、アプリケーション、アプレット、およびサーブレット内に既存のテンプレート（<**EspressReport InstallDir**>/**help/quickstart/templates** ディレクトリにある QuickStart42.rpt）を実行する方法を示します。

各セクションには、レポートを生成するためのコードと展開に必要な手順が示されています。

2.4.2.1 応用

次のコードは、既存のレポートテンプレート（この場合は QuickStart42.rpt）をアプリケーションに表示する方法を示しています。

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart421 extends Applet {
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

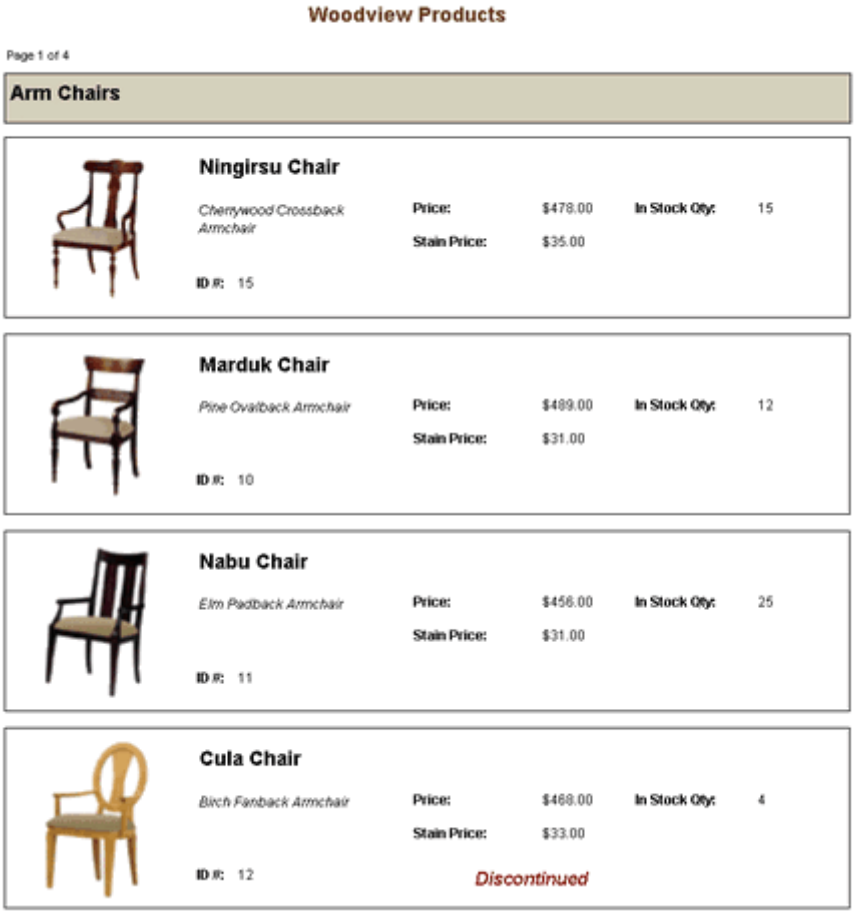
```
	public static void main(java.lang.String[] args) {
		try {
			QuickStart421 doReport = new QuickStart421();
			Frame frame = new Frame();
			frame.setLayout(new BorderLayout());
			frame.add("Center", doReport.doQuickStart421(frame));
			frame.setSize(600, 600);
			frame.setVisible(true);
		} catch (Exception ex) {
			ex.printStackTrace();
		}
	}

	public void init() {
		setLayout(new BorderLayout());
		add("Center", doQuickStart421(this));
	}

	Component doQuickStart421(Object object) {
		//Connect to EspressManager
		QbReport.setEspressManagerUsed(true);
		//Create new Report object using specified report
		QbReport report = new QbReport(object,
				"help/quickstart/templates/QuickStart42.rpt");
		//Show the Report
		return (new Viewer().getComponent(report));
	}
}
```

上記のソースのクラスファイルは、
<EspressReport InstallDir>/help/quickstart/classes ディレクトリにあります。

クラスファイルが実行されると、**java QuickStart421** コマンドを使用して次のレポートが表示されます：

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



生成されたレポート

コードの主要部分は doQuickStart421 コンポーネントにあります。そこで、QuickStart42.rpt テンプレートを使用して、**report** という **QbReport** オブジェクトが作成されます。次のコンストラクタが使用されます。

```
QbReport(Object parent, String reportTemplatename);
```

2.4.2.2 Java Web Start アプリケーション（JNLP）

次の JNLP（**<EspressReport InstallDir>/help/quickstart/html** ディレクトリの QuickStart422.jnlp）は、起動した JNLP ファイルを使用して既存のレポートテンプレート（この場合は QuickStart42.rpt）を表示する方法を示しています：

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.5" codebase="http://localhost:8080/" href="QuickStart422.jnlp">
    <information>
        <title>QuickStart422</title>
        <vendor>Quadbase Systems Inc.</vendor>
        <description>QuickStart422 JNLP</description>
        <offline-allowed/>
    </information>
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
    <security>
        <all-permissions/>
    </security>
    <resources>
          <j2se version="1.7+" max-heap-size="780m"/>
          <jar href="ReportAPIWithChart.jar"/>
          <jar href="QuickStart.jar" main="true"/>
    </resources>
    <applet-desc
        name="Espress Report QuickStart examples"
        main-class="QuickStart421"
        width="800"
        height="700">
            <param name="filename" value=""/>
      </applet-desc>
</jnlp>
```

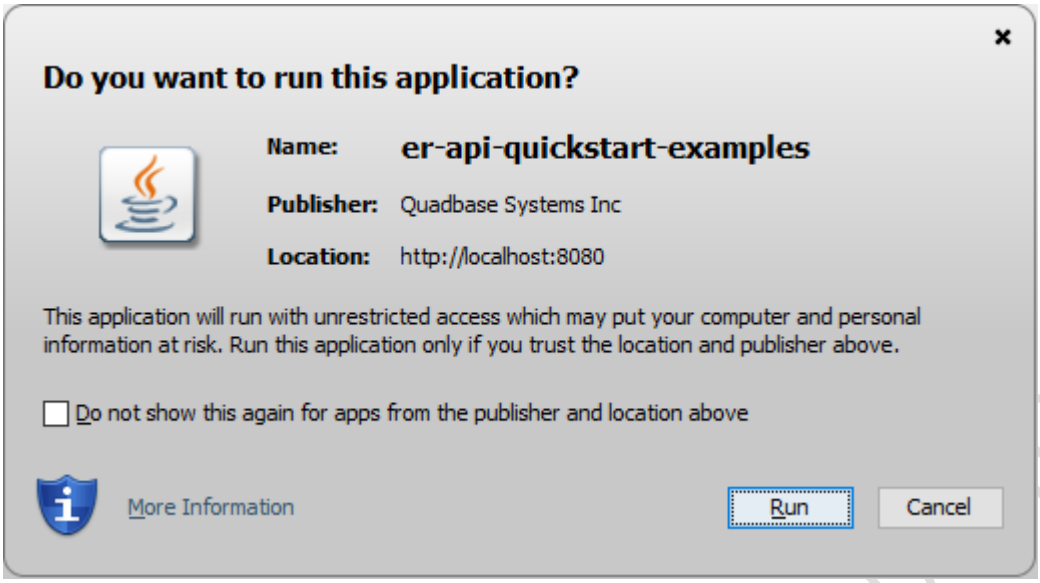
アプリケーションに使用されているのと同じコード（前のセクションで示した）を使用して、起動した JNLP ファイルにレポートを表示することもできます。JNLP ファイルをデプロイするには、次のようにコピーします。

- **QuickStart.jar** を**<EspressReport InstallDir>/help/quickstart/classes** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT** ディレクトリにコピーします。
- **ReportAPIWithChart.jar** と **qblicense.jar** を**<EspressReport InstallDir>/lib** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT** ディレクトリにコピーします。
- **QuickStart422.jnlp** を**<EspressReport InstallDir>/help/quickstart/html** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT** ディレクトリにコピーします。

URL「http://localhost:8080/QuickStart422.jnlp」（引用符なし）をブラウザに入力すると、このアプリケーションを実行するかどうかを尋ねるウィンドウがポップアップします。

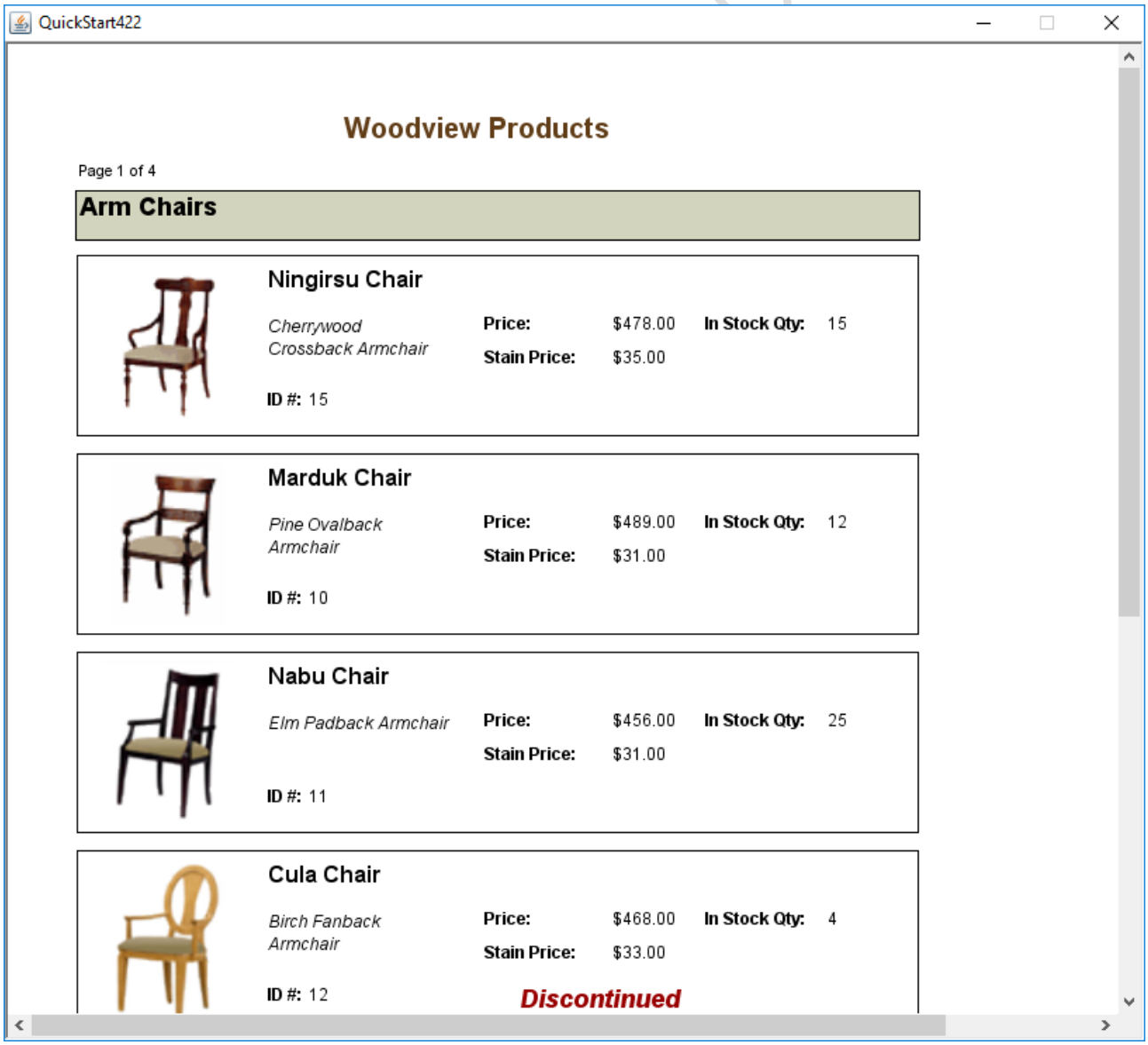
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



JNLP を実行するかどうか

Run ボタンをクリックすると、jnlp ファイルが実行され、次のレポートが表示されます。



Copyright(C) 2018 Climb.Inc. All Rights Reserved.

生成されたレポート

あなたのサーバが localhost でない場合は、**Quadbase**（QuickStart422.jnlp の 2 行目）の値を正しいサーバ IP で変更する必要があります。また、**QuickStart422.html** と **QuickStart422.jsp** を **<EspressReport InstallDir>/help/quickstart/html** ディレクトリから **<Tomcat InstallDir>/webapps/ROOT** ディレクトリにコピーすることもできます。その後、URL "http://localhost:8080/QuickStart422.html"（引用符なし）をブラウザに入力すると、jnlp がマシンにダウンロードされ、保存して開くことができます。

2.4.2.3 サーブレット

次のコードは、既存のレポートテンプレート（この場合は **QuickStart42.rpt**）をサーブレットに表示する方法を示しています。

```
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class QuickStart423 extends HttpServlet {

        public void doGet(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
                System.out.println("Calling QuickStart423....");
                //Set the "content type" header of the response
                res.setContentType("text/html");
                //Get the response's OutputStream to return content to the client.
                OutputStream toClient = res.getOutputStream();
                try {
                        //Use EspressManager
                        QbReport.setEspressManagerUsed(true);
                        //Open up specified Report
                        QbReport report = new QbReport((Applet) null,
                                        "help/quickstart/templates/QuickStart42.rpt");
                        //Export (Stream) the report to DHTML
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                report.export(QbReport.DHTML, toClient);
            } catch (Exception e) {
                e.printStackTrace();
            }
            //Flush the outputStream
            toClient.flush();
            //Close the writer; the response is done.
            toClient.close();
        }

        public String getServletInfo() {
            return "QuickStart423 servlet for EspressReport";
        }
}
```

サーブレットをデプロイするには：

- **QuickStart423.class** を<**EspressReport InstallDir**>/**help/quickstart/classes** ディレクトリから<**Tomcat InstallDir**>/**webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- Tomcat 7.x 以上のバージョンでは、サーブレット（この例では **QuickStart423**）を web.xml に明示的にマップする必要があります。

```
<servlet>
            <servlet-name>QuickStart423</servlet-name>
            <servlet-class>QuickStart423</servlet-class>
</servlet>
```

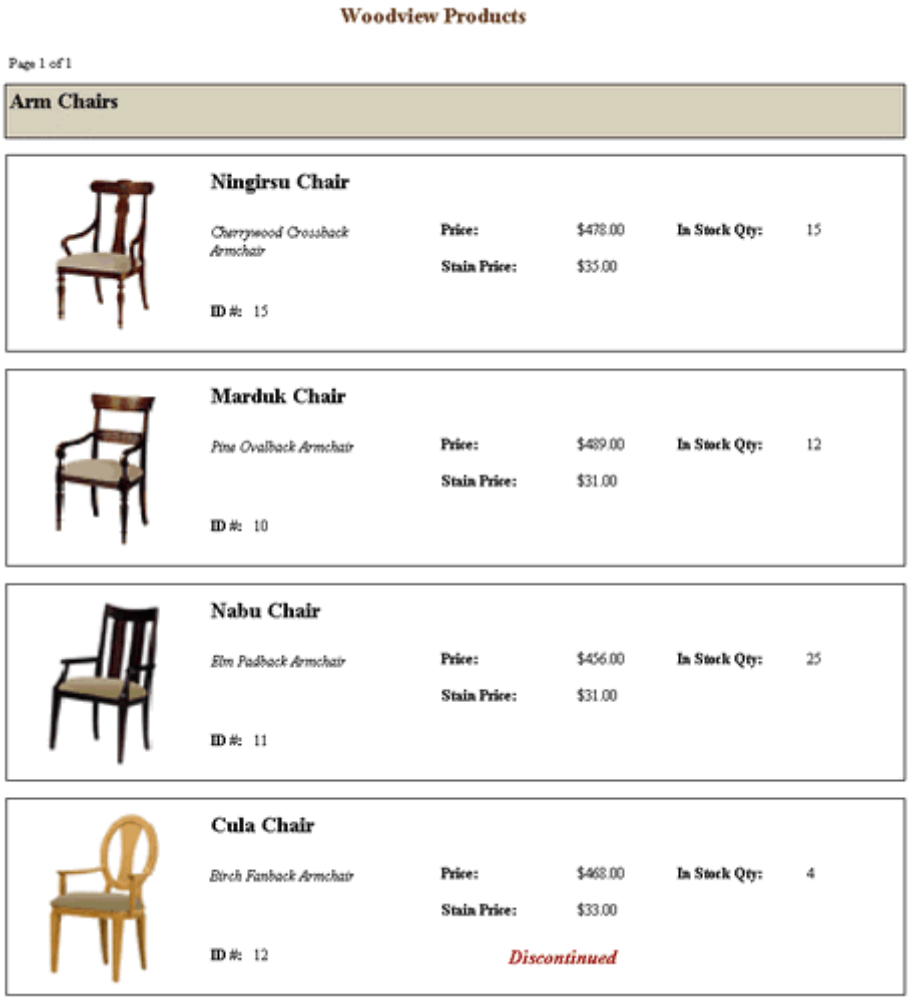
かつ

```
<servlet-mapping>
            <servlet-name>QuickStart423</servlet-name>
            <url-pattern>/servlet/QuickStart423</url-pattern>
</servlet-mapping>
```

URL「**http://localhost:8080/servlet/QuickStart423**」（引用符なし）を使用するサーブレットが実行されると、次のレポートが表示されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



生成されたレポート

コードの主要部分は **QuickStart423** にあります。そこで、**QuickStart42.rpt** テンプレートを使用して、**report** という **QbReport** オブジェクトが作成されます。次のコンストラクタが使用されます。

```
QbReport(Object parent, String reportTemplatename);
```

QbReport オブジェクト **report** は、**export** メソッドを使用してサーブレットのレスポンスストリームに DHTML ドキュメントとしてエクスポートされます。

```
<QbReport object>.export(int exportFormat, OutputStream out);
```

2.4.2.4 JSP

sevlets を使用してレポートをデプロイするほかに、JSP を使用してデプロイすることもできます。**JSP/Java Bean** テクノロジを使用してレポートをデプロイする例は、**<EspressReport InstallDir>/help/examples/jsp/deploy** フォルダにあります。同じフォルダには、このサンプルをサーブレットコンテナに設定する方法を説明する README.txt ファイルが含まれています。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2.4.2.5 Page Viewer

次のコードは、レポートビューアの代わりに Page Viewer を使用してアプリケーション/アプレットにレポートを表示する方法を示しています。

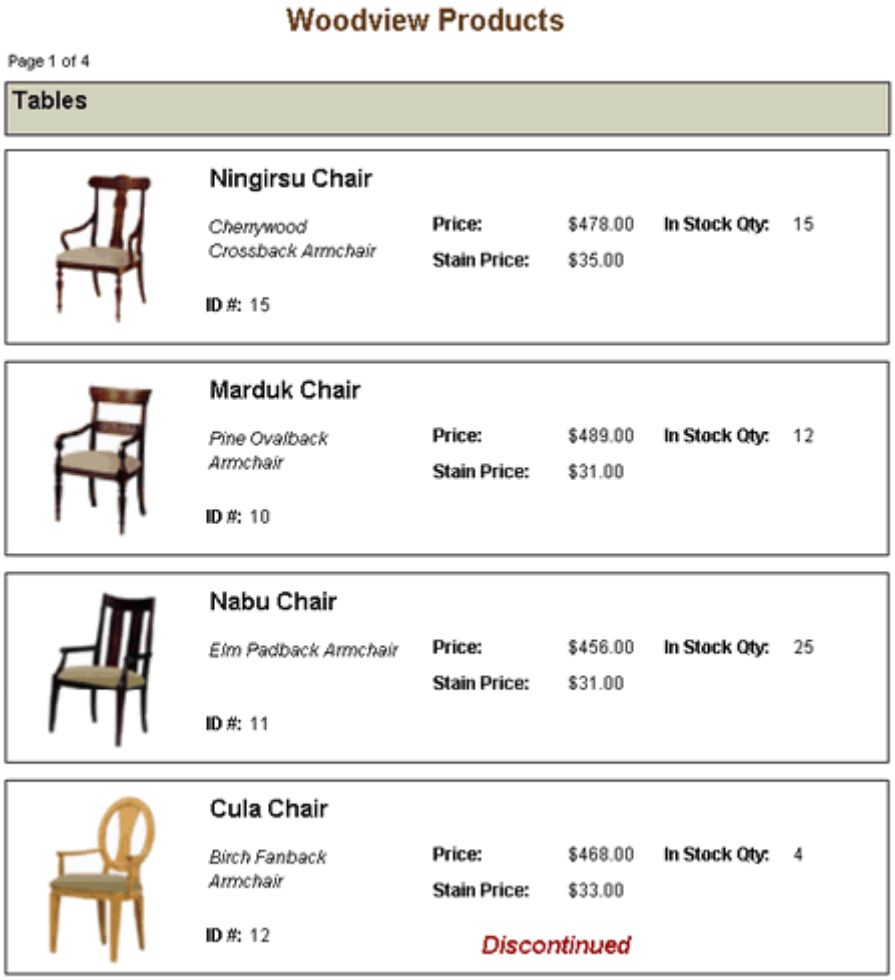
```
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.PageViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart424 extends Applet {

	public static void main(java.lang.String[] args) {
		try {
			QuickStart424 doReport = new QuickStart424();
			Frame frame = new Frame();
			frame.setLayout(new BorderLayout());
			frame.add("Center", doReport.doQuickStart424(frame));
			frame.setSize(600, 600);
			frame.setVisible(true);
		} catch (Exception ex) {
			ex.printStackTrace();
		}
	}

	public void init() {
		setLayout(new BorderLayout());
		add("Center", doQuickStart424(this));
	}

	Component doQuickStart424(Object parent) {
		//Connect to EspressManager
		QbReport.setEspressManagerUsed(true);

		//Location of the report
		String report = "help/quickstart/templates/QuickStart42.rpt";

		//Show the Report
		if (parent instanceof Applet)
			return (new Viewer().getComponent((Applet) parent, report, 0));
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                else
                        return (new Viewer().getComponent((Frame) parent, report, 0));
        }
}
```

上記のソースのクラスファイルは、
<EspressReport InstallDir>/help/quickstart/classes ディレクトリにあります。**java ファイル QuickStart424** を使用してクラスファイルを実行すると、次のレポートが表示されます。



生成されたレポート

QbReport オブジェクトは作成されません。代わりに**.rpt** ファイル名がコンポーネントに直接渡されます：

```
return (new Viewer().getComponent((Applet)parent, report, 0));
```

バックエンドでは**.rpt** は **VIEW** ファイルと **PAGE** ファイルにエクスポートされ、ビューアにロードされます。EspressManager を使用せずに実行すると、まずレポートを **VIEW** 形式でエクスポートする必要

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

があります。結果として得られる **VIEW** ファイルは、Page Viewer コンポーネントに渡すことができます。

Page Viewer コードは Java Web Start アプリケーションとしても実行できます。次の jnlp（**<EspressReport InstallDir>/help/quickstart/html** ディレクトリ内の **QuickStart424.jnlp**）は、Page Viewer JavaWS アプリケーションで既存のレポートテンプレート（この場合は **QuickStart42.rpt**）を表示する方法を示しています。

```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.5" codebase="http://localhost:8080/" href="QuickStart424.jnlp">
    <information>
        <title>QuickStart424</title>
        <vendor>Quadbase Systems Inc.</vendor>
        <description>QuickStart424 JNLP</description>
        <offline-allowed/>
    </information>
    <security>
        <all-permissions/>
    </security>
    <resources>
          <j2se version="1.7+" max-heap-size="780m"/>
          <jar href="ReportAPIWithChart.jar"/>
          <jar href="PageViewer.jar"/>
          <jar href="QuickStart.jar" main="true"/>
    </resources>
    <applet-desc
        name="Espress Report QuickStart examples"
        main-class="QuickStart424"
        width="800"
        height="700">
          <param name="filename" value=""/>
      </applet-desc>
</jnlp>
```

Copyright(C) 2018 Climb.Inc.

Java Web Start（JavaWS）アプリケーションでレポートを表示するために、アプリケーションに使用されているのと同じコード（前のセクションで示したもの）を使用することもできます。JavaWS アプリケーションをデプロイするには、次のようにコピーします。

- **QuickStart.jar** を**<EspressReport InstallDir>/help/quickstart/classes** ディレクトリから **<Tomcat InstallDir>/webapps/ROOT** ディレクトリにコピーします。
- **PageViewer.jar** を**<EspressReport InstallDir>/lib** ディレクトリから **<Tomcat InstallDir>/webapps/ROOT** ディレクトリにコピーします。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- **ReportAPIWithChart.jar** を**<EspressReport InstallDir>/lib** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT** ディレクトリにコピーします。
- **QuickStart424.jnlp** を**<EspressReport InstallDir>/help/quickstart/html** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT** ディレクトリにコピーします。

あなたのサーバが localhost でない場合は、**codebase**（**QuickStart424.jnlp** の 2 行目）の値を正しいサーバ IP で変更する必要があります。また、**QuickStart424.html** と **QuickStart424.jsp** を**<EspressReport InstallDir>/help/quickstart/html** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT** ディレクトリにコピーすることもできます。その後、URL 「http://localhost:8080/QuickStart422.html」（引用符なし）をブラウザに入力すると、jnlp がマシンにダウンロードされ、保存して開くことができます。

2.4.3 プログラムでレポートを作成する

次のセクションでは、レポートをプログラムで作成し、**QuickStart43.rpt** テンプレート（**<EspressReport InstallDir>/help/quickstart/templates** ディレクトリにあります）をアプリケーションのレポートに適用する方法を示します。

各セクションには、レポートを生成するためのコードと展開に必要な手順が示されています。

2.4.3.1 マップの概要 Break ColInfo

次のコードは、（データソースとして **QuickStart43.txt** を使用して）プログラムでサマリレポートを作成する方法を示しています。

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart431 extends Applet {

        //main method for application
        public static void main(java.lang.String[] args) {
                try {
                        QuickStart431 doReport = new QuickStart431();
                        Frame frame = new Frame();
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                frame.setLayout(new BorderLayout());
                frame.add("Center", doReport.doQuickStart431(frame));
                frame.setSize(600, 600);
                frame.setVisible(true);
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }

        //init method for applet
        public void init() {
            setLayout(new BorderLayout());
            add("Center", doQuickStart431(this));
        }

        //creates report and return it
        Component doQuickStart431(Object object) {
            //Connect to EspressManager
            QbReport.setEspressManagerUsed(true);
            //Specify Column Mapping
            ColInfo colInfo[] = new ColInfo[4];
            colInfo[0] = new ColInfo(0);
            colInfo[0].setRowBreak(true);
            colInfo[1] = new ColInfo(1);
            colInfo[1].setAggregation(false, ColInfo.NONE);
            colInfo[2] = new ColInfo(2);
            colInfo[2].setAggregation(false, ColInfo.SUM);
            colInfo[3] = new ColInfo(3);
            colInfo[3].setAggregation(false, ColInfo.SUM);
            //Create the QbReport object
            QbReport report = new QbReport(object, QbReport.SUMMARY,
                    "help/quickstart/templates/data/QuickStart43.txt", colInfo,
                    null);
            //Show the generated report in the Viewer
            return (new Viewer().getComponent(report));
        }
}
```

上記のソースのクラスファイルは、
<EspressReport InstallDir>/help/quickstart/classes ディレクトリにあります。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

クラスファイルが実行されると、**java QuickStart431** コマンドを使用して、次のレポートが表示されます。

Product Category	Product Name	Orders	Units Shipped
Arm Chairs	Adad Chair	4	50
	Cula Chair	10	129
	Marduk Chair	5	68
	Nabu Chair	4	53
	Ningirsu Chair	4	27
	Nisaba Chair	15	173
	Nusku Chair	10	124
	Sbuqamuma Chair	6	79
	Shimaliya Chair	7	81
		65	**784**
Double Dressers	Sekhmet Dresser	1	6
	Serket Dresser	4	36
	Set Dresser	2	22
	Shu Dresser	1	12
	Tefnut Dresser	1	9
	Thoth Dresser	2	22
		11	**107**

生成されたレポート

上記のコードを実行すると、生成されたレポートはデフォルトの外観になり、追加の書式設定は必要ありません。

コードの主要部分は **doQuickStart431** コンポーネントにあります。そこで、指定された列マッピング、指定されたレポートタイプ、および指定されたデータソースを使用して、**report** という **QbReport** オブジェクトが作成されます。次のコンストラクタが使用されます。

```
QbReport(Object parent, int reportType, String dataSource, ColInfo[] columnMapping, String reportTemplate);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2.4.3.2 テンプレートを適用する

QbReport オブジェクトの作成中にテンプレートを指定することによって、異なる書式を適用できます。

次のコードは、データソースとして **QuickStart43.txt** を使用し、テンプレートとして **QuickStart43.rpt** を使用して、プログラムでサマリブレークレポートを作成する方法を示しています。

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart432 extends Applet {

	//main method for application
	public static void main(java.lang.String[] args) {
		try {
			QuickStart432 doReport = new QuickStart432();
			Frame frame = new Frame();
			frame.setLayout(new BorderLayout());
			frame.add("Center", doReport.doQuickStart432(frame));
			frame.setSize(600, 600);
			frame.setVisible(true);
		} catch (Exception ex) {
			ex.printStackTrace();
		}
	}

	//init method for applet
	public void init() {
		setLayout(new BorderLayout());
		add("Center", doQuickStart432(this));
	}

	//creates report and return it
	Component doQuickStart432(Object object) {
		//Connect to EspressManager
		QbReport.setEspressManagerUsed(true);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            //Specify Column Mapping
            ColInfo colInfo[] = new ColInfo[4];
            colInfo[0] = new ColInfo(0);
            colInfo[0].setRowBreak(true);
            colInfo[1] = new ColInfo(1);
            colInfo[1].setAggregation(false, ColInfo.NONE);
            colInfo[2] = new ColInfo(2);
            colInfo[2].setAggregation(false, ColInfo.SUM);
            colInfo[3] = new ColInfo(3);
            colInfo[3].setAggregation(false, ColInfo.SUM);
            //Create the QbReport object
            QbReport report = new QbReport(object, QbReport.SUMMARY,
                                "help/quickstart/templates/data/QuickStart43.txt", colInfo,
                                "help/quickstart/templates/QuickStart43.rpt");
            //Show the generated report in the Viewer
            return (new Viewer().getComponent(report));
        }
}
```

上記のソースのクラスファイルは、
<EspressReport InstallDir>/help/quickstart/classes ディレクトリにあります。

クラスファイルが実行されると、**java QuickStart432** コマンドを使用して次のレポートが表示されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Product Sales Numbers

Product Category	Product Name	Orders	Units Shipped
Arm Chairs	Adad Chair	4	50
	Cula Chair	10	129
	Marduk Chair	5	68
	Nabu Chair	4	53
	Ningirsu Chair	4	27
	Nisaba Chair	15	173
	Nusku Chair	10	124
	Sbuqamuma Chair	6	79
	Shimaliya Chair	7	81
Total for Arm Chairs		**65**	**784**
Double Dressers	Sekhmet Dresser	1	6
	Serket Dresser	4	36
	Set Dresser	2	22
	Shu Dresser	1	12
	Tefnut Dresser	1	9
	Thoth Dresser	2	22
Total for Double Dressers		**11**	**107**

生成されたレポート

コードの主要部分は **doQuickStart432** コンポーネントにあります。そこで、指定された列マッピング、指定されたレポートタイプ、指定されたデータソース、および指定されたレポートテンプレートを使用して、**report** という **QbReport** オブジェクトが作成されます。次のコンストラクタが使用されます。

```
QbReport(Object parent, int reportType, String dataSource, ColInfo[] columnMapping, String reportTemplate);
```

2.4.4 レポートのデータソースの変更

次のコードは、新しい **QbReport** オブジェクトを作成することなく、レポート（およびそれに伴うサブレポート、ドリルダウンおよび独立チャート）データソースを変更する方法を示しています。**QbReport** オブジェクト（**QuickStart44.rpt** から作成）は、バックアップデータとともにオープンされます（データベースへの不必要なロードを避けるため）。その後、データソースは MySQL Woodview データベース

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

に変更されます。

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.common.util.*;
import quadbase.reportdesigner.lang.*;
import java.sql.*;

public class QuickStart44 extends Applet {

	public static void main(java.lang.String[] args) {

		try {

			QuickStart44 doReport = new QuickStart44();
			Frame frame = new Frame();
			frame.setLayout(new BorderLayout());
			frame.add("Center", doReport.doQuickStart44(frame));
			frame.setSize(600, 600);
			frame.setVisible(true);
		} catch (Exception ex) {
			ex.printStackTrace();
		}
	}

	public void init() {

		setLayout(new BorderLayout());
		add("Center", doQuickStart44(this));
	}

	Component doQuickStart44(Object object) {

		//Connect to EspressManager
		QbReport.setEspressManagerUsed(true);
```

Copyright(C) 2018 Climb.Inc.

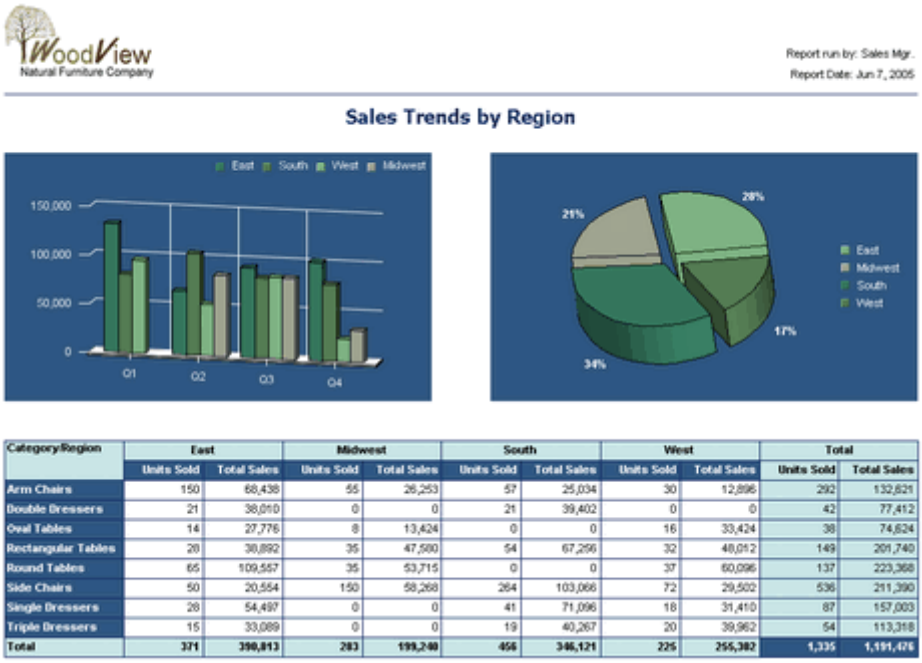
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                //Create the report using the two rows of back-up data
                QbReport report = new QbReport(object,
                                "help/quickstart/templates/QuickStart44.rpt", false, false,
                                false, true);

                //Begin Code : Specification for new Database
                String newDatabaseURL = "jdbc:mysql://localhost:3306/woodview";
                String newDatabaseDriver = "com.mysql.jdbc.Driver";
                String newDatabaseUserid = "root";
                String newDatabasePassword = "root";
                //End Code : Specification for new Database

                try {

                        //Begin Code : Change the data source of the main report and all
                        //ancillary templates
                        report.getInputData().setAllDatabaseInfo(newDatabaseURL,
                                        newDatabaseDriver,          newDatabaseUserid,
newDatabasePassword);
                        //End Code : Change the data source of the main report and all
                        //ancillary templates
                } catch (Exception ex) {

                        ex.printStackTrace();
                }

                return (new Viewer().getComponent(report));
        }
}
```

上記のソースのクラスファイルは、
<EspressReport InstallDir>/help/quickstart/classes ディレクトリにあります。

クラスファイルが実行されると、**java QuickStart44** コマンドを使用して次のレポートが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Category/Region	East		Midwest		South		West		Total	
	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales
Arm Chairs	150	68,438	55	26,253	57	25,034	30	12,896	292	132,621
Double Dressers	21	38,010	0	0	21	39,402	0	0	42	77,412
Oval Tables	14	27,776	8	13,424	0	0	16	33,424	38	74,624
Rectangular Tables	28	38,892	35	47,580	54	67,256	32	48,012	149	201,740
Round Tables	65	109,557	35	53,715	0	0	37	60,096	137	223,368
Side Chairs	50	20,554	150	58,268	264	103,066	72	29,502	536	211,390
Single Dressers	28	54,497	0	0	41	71,096	18	31,410	87	157,003
Triple Dressers	15	33,089	0	0	19	40,267	20	39,962	54	113,318
Total	**371**	**390,813**	**283**	**199,240**	**456**	**346,121**	**225**	**255,302**	**1,335**	**1,191,476**

生成されたレポート

コードの主要部分は **doQuickStart44** コンポーネントにあります。そこで、**report** という **QbReport** オブジェクトが作成されます。quadbase.reportdesigner.util.IInputData インタフェースの次のメソッドを使用して、レポート、サブレポート、ドリルダウン、およびチャート（独立したデータソースを使用）のデータソースを変更します。

```
setAllDatabaseInfo(String url, String driver, String userid, String password);
```

2.4.4.1 レポートのデータソースとクエリの変更（SubReport 付き）

次のコードは、新しい **QbReport** オブジェクトを作成することなく行う、レポート（それに付随するサブレポート）データソースを変更する方法を示します。データソースは、MySQL Woodview データベースから HSQLDB Woodview データベースに変更されます。HSQLDB データベースがヒットしなかった場合、**QbReport** オブジェクト（**QuickStart441_mysql.pak** から作成された）はバックアップデータと共に開かれます。

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.common.util.*;
import quadbase.reportdesigner.lang.*;
import java.sql.*;
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
public class QuickStart441 extends Applet {

        public static void main(java.lang.String[] args) {

            try {

                    QuickStart441 doReport = new QuickStart441();
                    Frame frame = new Frame();
                    frame.setLayout(new BorderLayout());
                    frame.add("Center", doReport.doQuickStart441(frame));
                    frame.setSize(600, 600);
                    frame.setVisible(true);
            } catch (Exception ex) {
                    ex.printStackTrace();
            }
        }

        public void init() {

            setLayout(new BorderLayout());
            add("Center", doQuickStart441(this));
        }

        Component doQuickStart441(Object object) {

            //Connect to EspressManager
            QbReport.setEspressManagerUsed(true);

            //Create the report using the two rows of back-up data
            QbReport report = new QbReport(object,
                            "help/quickstart/templates/MySQL/QuickStart441_mysql.pak",
                            false, false, false, true);

            //Begin Code : Specification for new Database
            String                          newDatabaseURL                          =
"jdbc:hsqldb:help/examples/DataSources/database/woodview";
            String newDatabaseDriver = "org.hsqldb.jdbcDriver";
            String newDatabaseUserid = "sa";
            String newDatabasePassword = "";
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
String newDatabaseReportQuery = "select year(o.orderdate) as ¥"Year¥", month(o.orderdate) as ¥"Month¥", count(o.orderid) as ¥"Orders¥", sum(od.quantity) as ¥"Units Sold¥", sum((p.unitprice + od.staincost) * od.quantity) as ¥"Total Sales¥" from orders o, order_details od, products p where o.orderid = od.orderid and p.productid = od.productid group by year(o.orderdate), month(o.orderdate) order by year(o.orderdate), month(o.orderdate);";

String newDatabaseSubReportQuery = "select c.region as ¥"Region¥", year(o.orderdate) as ¥"Year¥", sum((p.unitprice + od.staincost) * od.quantity) as ¥"Total Sales¥" from customers c, orders o, products p, order_details od where c.customerid = o.customerid and o.orderid = od.orderid and od.productid = p.productid group by c.region, year(o.orderdate);";
try {
    //Begin Code : Get a handle to the Sub-Report and change its data
    //source
    SubReportObject subReportObject = report.getSubReports()[0];
    QbReport subReport = (QbReport) subReportObject.getSubReport(false,
            false, true, report);

    //Get the query and pass in new database info
    DBInfo newSubReportDatabaseInfo = new DBInfo(newDatabaseURL,
            newDatabaseDriver, newDatabaseUserid,
newDatabasePassword,
            newDatabaseSubReportQuery);

    subReport.getInputData().setDatabaseInfo(newSubReportDatabaseInfo);
    //End Code : Get a handle to the Sub-Report and change its data
    //source

    //Begin Code : Change the data source of the main report
    //Get the query and pass in new database info
    DBInfo newReportDatabaseInfo = new DBInfo(newDatabaseURL,
            newDatabaseDriver, newDatabaseUserid,
newDatabasePassword,
            newDatabaseReportQuery);

    report.getInputData().setDatabaseInfo(newReportDatabaseInfo);
    //End Code : Change the data source of the main report
} catch (Exception ex) {

    ex.printStackTrace();
}
```

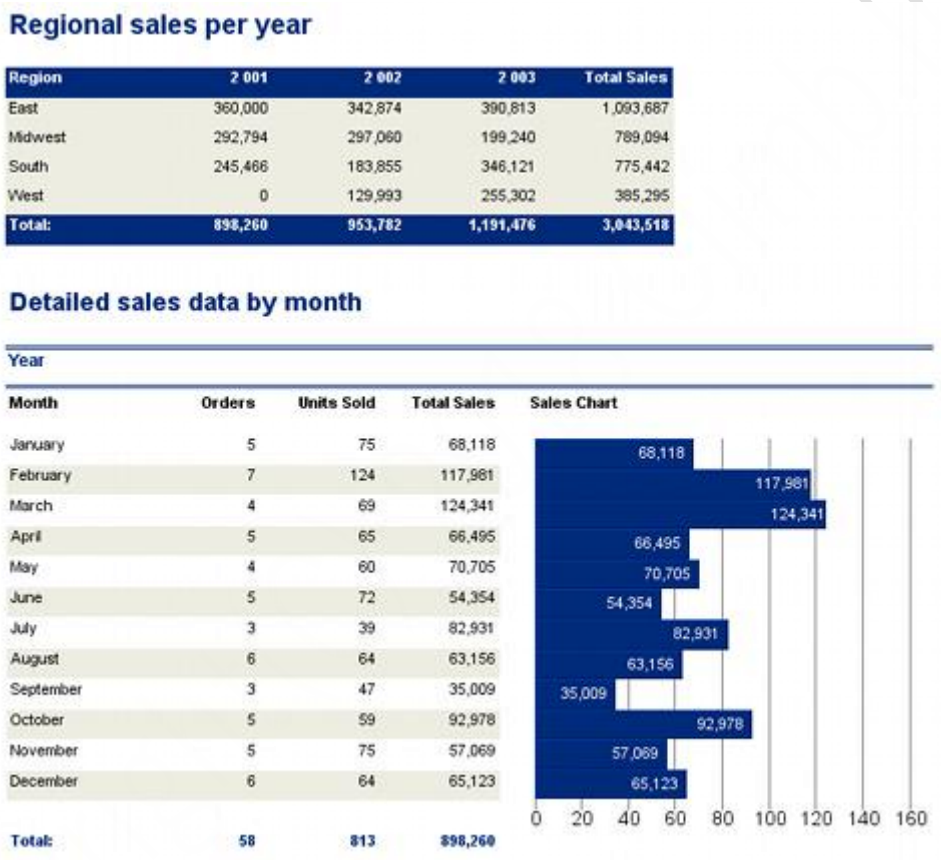
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            return (new Viewer().getComponent(report));
        }
}
```

上記のソースのクラスファイルは、
<EspressReport InstallDir>/help/quickstart/classes ディレクトリにあります。

クラスファイルが実行されると、**java QuickStart441** コマンドを使用して次のレポートが表示されます。



Regional sales per year

Region	2 001	2 002	2 003	Total Sales
East	360,000	342,874	390,813	1,093,687
Midwest	292,794	297,060	199,240	789,094
South	245,466	183,855	346,121	775,442
West	0	129,993	255,302	385,295
Total:	**898,260**	**953,782**	**1,191,476**	**3,043,518**

Detailed sales data by month

Year

Month	Orders	Units Sold	Total Sales	Sales Chart
January	5	75	68,118	
February	7	124	117,981	
March	4	69	124,341	
April	5	65	66,495	
May	4	60	70,705	
June	5	72	54,354	
July	3	39	82,931	
August	6	64	63,156	
September	3	47	35,009	
October	5	59	92,978	
November	5	75	57,069	
December	6	64	65,123	
Total:	**58**	**813**	**898,260**	

生成されたレポート

コードの主要部分は **doQuickStart441** コンポーネントにあります。そこで、**report** という **QbReport** オブジェクトが作成されます。レポートおよびサブレポートのデータソースは、quadbase.reportdesigner.util.IInputData インタフェースの次のメソッドを使用して変更されます：

```
setDatabaseInfo(IDatabaseInfo db);
```

サブレポートは、サブレポートを含むセルへのハンドルを取得し、サブレポートを呼び出すことによって取得されます。サブレポートは、そのバックアップデータを使用して開かれます。これは次のように行われます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
SubReportObject subReportObject = report.getSubReports()[index of particular SubReport];
(QbReport)subreport = (QbReport)subReportObject.getSubReport(boolean isEnterpriseServer, boolean optimizeMemory, boolean useBackupData, IReport report);
```

データソースを HSQLDB Woodview データベースから MySQL Woodview データベースに変更したい場合は、以下のコード行を変更する必要があります：

```
QbReport report = new QbReport(object, "help/quickstart/templates/MySQL/QuickStart441_.pak", false, false, false, true);

//Begin Code : Specification for new Database
String newDatabaseURL = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
String newDatabaseDriver = "org.hsqldb.jdbcDriver";
String newDatabaseUserid = "sa";
String newDatabasePassword = "";

String newDatabaseReportQuery = "select year(o.orderdate) as ¥"Year¥", month(o.orderdate) as ¥"Month¥", count(o.orderid) as ¥"Orders¥", sum(od.quantity) as ¥"Units Sold¥", sum((p.unitprice + od.staincost) * od.quantity) as ¥"Total Sales¥" from orders o, order_details od, products p where o.orderid = od.orderid and p.productid = od.productid group by year(o.orderdate), month(o.orderdate) order by year(o.orderdate), month(o.orderdate);";

String newDatabaseSubReportQuery = "select c.region as ¥"Region¥", year(o.orderdate) as ¥"Year¥", sum((p.unitprice + od.staincost) * od.quantity) as ¥"Total Sales¥" from customers c, orders o, products p, order_details od where c.customerid = o.customerid and o.orderid = od.orderid and od.productid = p.productid group by c.region, year(o.orderdate);";
```

から

```
QbReport report = new QbReport(object, "help/quickstart/templates/QuickStart441.pak", false, false, false, true);

//Begin Code : Specification for new Database
String newDatabaseURL = "jdbc:mysql://localhost:3306/woodview";
String newDatabaseDriver = "com.mysql.jdbc.Driver";
String newDatabaseUserid = "root";
String newDatabasePassword = "root";

String newDatabaseReportQuery = "select year(o.orderdate) as ¥"Year¥", month(o.orderdate) as ¥"Month¥", count(o.orderid) as ¥"Orders¥", sum(od.quantity) as ¥"Units Sold¥", sum((p.unitprice + od.staincost) * od.quantity) as ¥"Total Sales¥" from orders o, ¥"order_details¥" od, products p where
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
o.orderid = od.orderid and p.productid = od.productid group by year(o.orderdate), month(o.orderdate) order by year(o.orderdate), month(o.orderdate);";

String newDatabaseSubReportQuery = "select c.region as ¥"Region¥", year(o.orderdate) as ¥"Year¥", sum((p.unitprice + od.staincost) * od.quantity) as ¥"Total Sales¥" from customers c, orders o, products p, ¥"order_details¥" od where c.customerid = o.customerid and o.orderid = od.orderid and od.productid = p.productid group by c.region, year(o.orderdate);";
```

2.4.4.2 パラメータ化されたレポートのデータソースとクエリの変更（パラメータ化されたサブレポート付き）

次のコードは、新しい **QbReport** オブジェクトを作成することなく、レポートの付随するサブレポートとパラメータ化されたデータソースを変更する方法を示しています。**QbReport** オブジェクト（**QuickStart442_Acc.rpt** から作成）は、バックアップデータとともにオープンされます（データベースへの不必要なロードを避けるため）。その後、データソースは、Access Woodview データベースから HSQLDB Woodview データベースに変更されます。

```
import java.awt.*;
import java.util.*;
import java.io.*;
import java.applet.*;
import java.sql.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart442 extends Applet {

        public static void main(java.lang.String[] args) {

                try {
                        QuickStart442 doReport = new QuickStart442();
                        Frame frame = new Frame();
                        frame.setLayout(new BorderLayout());
                        frame.add("Center", doReport.doQuickStart442(frame));
                        frame.setSize(600, 600);
                        frame.setVisible(true);

                } catch (Exception ex) {
                        ex.printStackTrace();
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            }

        }

        public void init() {

            setLayout(new BorderLayout());
            add("Center", doQuickStart442(this));

        }

        Component doQuickStart442(Object parent) {

            //Connect to EspressManager
            QbReport.setEspressManagerUsed(true);

            //Open the report using the two rows of backup data
            QbReport report = new QbReport(parent,
                        "help/quickstart/templates/MySQL/QuickStart442_mysql.rpt",
                        false, false, false, true);

            //Begin Code : Specification for new Database
            String URL = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
            String driver = "org.hsqldb.jdbcDriver";
            String userid = "sa";
            String passwd = "";

            String newDatabaseSubReportQuery = "select c.categoryname as ¥"Category¥",
p.productname as ¥"Product¥", cu.region as ¥"Region¥", sum((od.staincost + p.unitprice) *
od.quantity) as ¥"Sales¥" from categories c, products p, customers cu, orders o, order_details od
where c.categoryid = p.categoryid and p.productid = od.productid and cu.customerid = o.customerid
and o.orderid = od.orderid and c.categoryname IN (:category) group by c.categoryname,
p.productname, cu.region";

            try {

                //Begin Code : Get a handle to the SubReport and change its
                //datasource
                SubReportObject subReportObject = report.getSubReports()[0];
                QbReport subReport = (QbReport) subReportObject.getSubReport(false,
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                                false, true, report);

                        //Begin Code : Get the parameter information of the subreport and
                        //pass in the new database information along with the parameter
                        //information
                        IQueryInParam[] subReportParameters = ((IQueryFileInfo) subReport
                                        .getInputData().getDatabaseInfo()).getInParam();
                        SimpleQueryFileInfo subReportInfo = new SimpleQueryFileInfo(URL,
                                        driver, userid, passwd, newDatabaseSubReportQuery);
                        subReportInfo.setInParam(subReportParameters);

                        subReport.getInputData().setDatabaseInfo(subReportInfo);
                        //End Code : Get the parameter information of the subreport and
                        //pass in the new database information along with the parameter
                        //information
                        //End Code : Get a handle to the SubReport and change its
                        //datasource

                        //Begin Code : Get the parameter information of the main report and
                        //pass in the new database information along with the parameter
                        //information
                        //Begin Code : Pass in the parameter values for the main report
                        //(which is then picked up by the subreport)
                        Vector<String> paramValues = new Vector<String>();
                        paramValues.addElement(new String("Arm Chairs"));
                        paramValues.addElement(new String("Double Dressers"));
                        paramValues.addElement(new String("Round Tables"));
                        //End Code : Pass in the parameter values for the main report
                        //(which is then picked up by the subreport)

                        //Begin Code : Get the parameter properties information and pass in
                        //the value of the parameter
                        IQueryInParam[] reportParameters = (IQueryInParam[]) ((IQueryFileInfo)
report
                                        .getInputData().getDatabaseInfo()).getInParam();
                        ((IQueryMultiValueInParam) reportParameters[0])
                                        .setValues(paramValues);
                        //End Code : Get the parameter properties information and pass in
                        //the value of the parameter

                        //Get the query for the main report from the report template itself
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

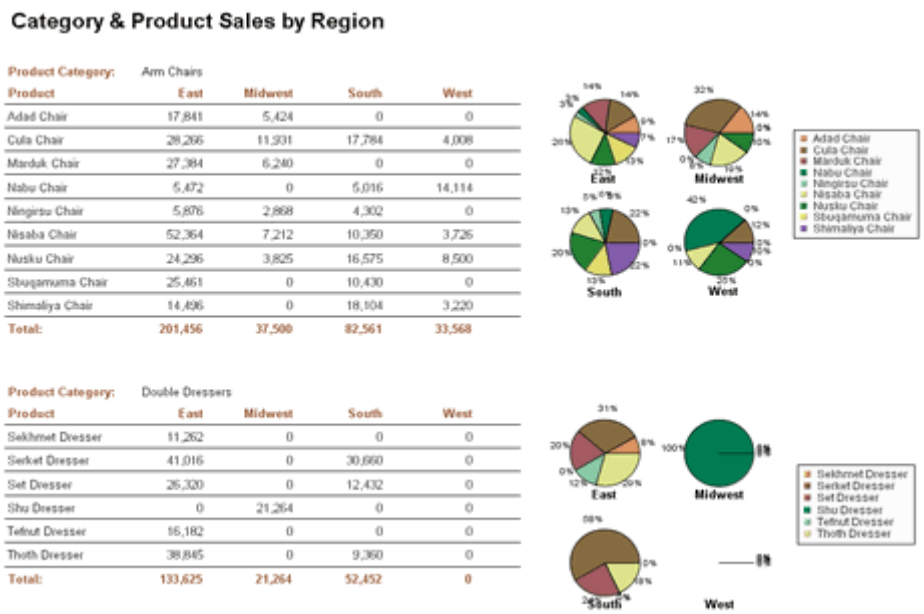
```
                    //instead of passing in a new query
                    SimpleQueryFileInfo reportInfo = new SimpleQueryFileInfo(URL,
                                driver, userid, passwd, report.getInputData()
                                                .getDatabaseInfo().getQuery());
                    reportInfo.setInParam(reportParameters);

                    report.getInputData().setDatabaseInfo(reportInfo);
                    //End Code : Get the parameter information of the main report and
                    //pass in the new database information along with the parameter
                    //information

            } catch (Exception ex) {
                    ex.printStackTrace();

            }

            return (new Viewer().getComponent(report));

        }

}
```

上記のソースのクラスファイルは、
<EspressReport InstallDir>/help/quickstart/classes ディレクトリにあります。

クラスファイルが実行されると、**Java QuickStart442** コマンドを使用して次のレポートが表示されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



生成されたレポート

コードの主要部分は **doQuickStart442** コンポーネントにあります。そこで、**report** という **QbReport** オブジェクトが作成されます。レポートおよびサブレポートのデータソースは、quadbase.reportdesigner.util.IInputData インタフェースの次のメソッドを使用して変更されます：

```
setDatabaseInfo(IDatabaseInfo db);
```

ただし、両方のレポートではパラメータ化されたレポートが使用されるため、データベース接続情報とパラメータプロパティ情報を渡すために、**quadbase.reportdesigner.util.IQueryFileInfo** インタフェースを使用する別のクラスファイル（**QueryFileInfo**）が作成されます。次に、**QueryFileInfo** クラスのオブジェクト（必要な情報を持つ）が **setDatabaseInfo()**メソッドの引数として渡されます。

パラメータ情報は、（**getDatabaseInfo()**メソッドを呼び出して）データベース接続情報を取得し、**IQueryFileInfo** にキャストし、**IQueryFileInfo** で次のメソッドを使用して取得します。

```
public IQueryInParam[] getInParam();
```

上記のメソッド呼び出しは、レポートに定義されているすべてのパラメータの完全なパラメータプロパティ情報を返します。

コードでは、値を指定するようにユーザに指示するのではなく、パラメータの値をレポートに直接渡します。値は、各パラメータに移動し、それらを指定してレポートオブジェクトに渡すことによって入力されます。値を指定するには、**IQueryInParam** の次のメソッドを使用します。

```
public void setValue(Object value);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

サブレポートは、サブレポートを含むセルへのハンドルを取得し、サブレポートを呼び出すことによって取得されます。サブレポートは、そのバックアップデータを使用して開かれます。これは次のように行われます。

```
SubReportObject subReportObject = report.getSubReports()[index of particular SubReport];

(QbReport)subreport = (QbReport)subReportObject.getSubReport(boolean isEnterpriseServer, boolean optimizeMemory, boolean useBackupData, IReport report);
```

データソースを HSQLDB Woodview データベースから MySQL Woodview データベースに変更したい場合は、以下のコード行を変更する必要があります。

```
QbReport report = new QbReport(object, "help/quickstart/templates/MySQL/QuickStart442_mysql.rpt", false, false, false, true);

//Begin Code : Specification for new Database
String newDatabaseURL = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
String newDatabaseDriver = "org.hsqldb.jdbcDriver";
String newDatabaseUserid = "sa";
String newDatabasePassword = "";

String newDatabaseSubReportQuery = "select c.categoryname as ¥"Category¥", p.productname as ¥"Product¥", cu.region as ¥"Region¥", sum((od.staincost + p.unitprice) * od.quantity) as ¥"Sales¥" from categories c, products p, customers cu, orders o, order_details od where c.categoryid = p.categoryid and p.productid = od.productid and cu.customerid = o.customerid and o.orderid = od.orderid and c.categoryname IN (:category) group by c.categoryname, p.productname, cu.region;";
```

から

```
QbReport report = new QbReport(object, "help/quickstart/templates/QuickStart442.rpt", false, false, false, true);

//Begin Code : Specification for new Database
String newDatabaseURL = "jdbc:mysql://localhost:3306/woodview";
String newDatabaseDriver = "com.mysql.jdbc.Driver";
String newDatabaseUserid = "root";
String newDatabasePassword = "root";

String newDatabaseSubReportQuery = "select c.categoryname as ¥"Category¥", p.productname as ¥"Product¥", cu.region as ¥"Region¥", sum((od.staincost + p.unitprice) * od.quantity) as ¥"Sales¥"
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
from categories c, products p, customers cu, orders o, order_details od where c.categoryid = p.categoryid and p.productid = od.productid and cu.customerid = o.customerid and o.orderid = od.orderid and c.categoryname IN (:category) group by c.categoryname, p.productname, cu.region;";
```

また、データベースの URL、ドライバ、ユーザ ID、およびパスワードを渡す代わりに **java.sql.Connection** オブジェクトを渡すこともできます。たとえば、**Connection** オブジェクト **conn** を渡す場合は、次のコード行を変更する必要があります。

```
SimpleQueryFileInfo subReportInfo = new SimpleQueryFileInfo(URL, driver, userid, passwd, newDatabaseSubReportQuery);
SimpleQueryFileInfo reportInfo = new SimpleQueryFileInfo(URL, driver, userid, passwd, report.getInputData().getDatabaseInfo().getQuery());
```

から

```
SimpleQueryFileInfo subReportInfo = new SimpleQueryFileInfo(conn, newDatabaseSubReportQuery);
SimpleQueryFileInfo reportInfo = new SimpleQueryFileInfo(conn, report.getInputData().getDatabaseInfo().getQuery());
```

2.4.5 レポート要素の変更

次のコードは、レポートの特定の要素をプログラムによって変更する方法を示しています。**QbReport** オブジェクトは、**QuickStart45.rpt** から作成されます。コードを実行すると、元のレポートが最初に表示され、**Change** ボタン（下部にある）がクリックされると、レポート要素の一部が変更されます。

```
import java.awt.*;
import java.io.*;
import java.applet.*;
import java.util.*;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;
import quadbase.reportdesigner.report.Formula;

//Set up Frame
public class QuickStart45 extends Frame {
        QbReport report;
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
Viewer viewer;
Component reportComponent;
Button b = new Button("Change");

//Start Frame
public QuickStart45() {
        start();
}

//Create Empty Report and set up initial data and wipe out empty report
//data
public void start() {

        //Connect to EspressManager
        QbReport.setEspressManagerUsed(true);
        setLayout(new BorderLayout());
        //Create QbReport object
        report = new QbReport((Applet) null,
                        "help/quickstart/templates/QuickStart45.rpt");
        viewer = new Viewer();
        reportComponent = viewer.getComponent(report);
        add("Center", reportComponent);
        add("South", b);
}

//What to do with the button
public boolean action(Event e, Object o) {
        //Begin Code : Dual color
        int numberOfColumns = report.getTable().getColumnCount();
        for (int i = 0; i < numberOfColumns; i++) {
                ReportColumn column = report.getTable().getColumn(i);
                column.setAlternateRow(1);
                column.setBgColor2(new Color(245, 245, 238));
                column.setFontColor2(new Color(0, 0, 0));
                column.setFont2(new Font("Dialog", Font.PLAIN, 8));
        }
        //End Code : Dual color

        //Begin Code : Add title to Report Header
        ReportCell title = new ReportCell();
        title.setText("Top 10 Customers");
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
title.setBgColor(new Color(255, 255, 255));
title.setFontColor(new Color(0, 54, 100));
title.setFont(new Font("Dialog", Font.BOLD, 14));
title.setAlign(IAlignConstants.ALIGN_LEFT);
title.setWidth(2.1);
title.setHeight(0.4);
title.setX(0);
title.setY(0);
report.getReportHeader().addData(title);
//End Code : Add title to Report Header

//Begin Code :Add a formula
ReportCell formulaCell = new ReportCell();
Formula formula = new Formula("totalSales", "sum({Total Sales})");
report.addFormula(formula);
formulaCell.setFormulaObj(formula);
formulaCell.setBgColor(new Color(255, 255, 255));
formulaCell.setFontColor(new Color(0, 0, 0));
formulaCell.setFont(new Font("Dialog", Font.BOLD, 8));
formulaCell.setAlign(IAlignConstants.ALIGN_LEFT);
formulaCell.setWidth(1.0);
formulaCell.setHeight(0.25);
formulaCell.setX(4.1);
formulaCell.setY(0);
report.getReportFooter().addData(formulaCell);
//End Code :Add a formula

//Begin Code : Add a label
ReportCell label = new ReportCell();
label.setText("Total sales for top 10 customers:");
label.setBgColor(new Color(255, 255, 255));
label.setFontColor(new Color(0, 54, 100));
label.setFont(new Font("Dialog", Font.BOLD, 8));
label.setAlign(IAlignConstants.ALIGN_RIGHT);
label.setWidth(2.1);
label.setHeight(0.4);
label.setX(1.9);
label.setY(0);
report.getReportFooter().addData(label);
//End Code : Add a label
remove(reportComponent);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                viewer = new Viewer();
                reportComponent = viewer.getComponent(report);
                add("Center", reportComponent);
                pack();
                return true;
        }

        public Dimension getPreferredSize() {
                return new Dimension(600, 600);
        }

        //Start
        public static void main(String[] args) {
                QuickStart45 t = new QuickStart45();
                t.setSize(600, 600);
                t.setVisible(true);
        }
}
```

上記のソースのクラスファイルは、
<EspressReport InstallDir>/help/quickstart/classes ディレクトリにあります。

クラスファイルが実行されると、**java QuickStart45** コマンドを使用して次のレポートが表示されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Rank	Company	Orders	Units Ordered	Total Sales
1	Imports & Leather Gallery 5 Baker's Lane Chicago, IL 39283	21	289	341,028
2	Sevilla Home & Garden 389 Jardim Rd. Sevilla, AK 82938	29	378	335,106
3	Allied Furniture Emporium 384 Broad St. Littletown, NY 18322	24	310	328,390
4	Woodworks Furniture 88 Rio Grand Ave. Akergen, PA 38923	19	196	301,834
5	Furniture Palace 1230 Hoes Lane Piscataway, NJ 08854	20	197	261,799
6	Benson Imports 23 Tasaga Ave. Aachen, TX 03887	18	240	227,318
7	Eastern Treasures 123 Summer Rd. Pitterson, NJ 09882	29	296	215,498
8	Specialty Retail 88 Spenser Drive Los Angles, CA 88938	14	132	161,689
9	Furniture Gallery 90 First Ave. New York, NY 11223	14	131	140,501
10	HomeWorld Furniture 71 Mulbery St. San Francisco, CA 98839	11	117	126,520

生成されたレポート

Change ボタンをクリックすると、次のレポートが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Top 10 Customers

Rank	Company	Orders	Units Ordered	Total Sales
1	Imports & Leather Gallery 5 Baker's Lane Chicago, IL 39283	21	289	341,028
2	Sevilla Home & Garden 389 Jardim Rd. Sevilla, AK 82938	29	378	335,106
3	Allied Furniture Emporium 384 Broad St. Littletown, NY 18322	24	310	328,390
4	Woodworks Furniture 88 Rio Grand Ave. Akergen, PA 38923	19	196	301,834
5	Furniture Palace 1230 Hoes Lane Piscataway, NJ 08854	20	197	261,799
6	Benson Imports 23 Tasaga Ave. Aachen, TX 03887	18	240	227,318
7	Eastern Treasures 123 Summer Rd. Pitterson, NJ 09882	29	296	215,498
8	Specialty Retail 88 Spenser Drive Los Angles, CA 88938	14	132	161,689
9	Furniture Gallery 90 First Ave. New York, NY 11223	14	131	140,501
10	HomeWorld Furniture 71 Mulbery St. San Francisco, CA 98839	11	117	126,520
			Total sales for top 10 customers:	2,439,683

変更ボタンがクリックされた後に生成されるレポート

コードの主要部分はアクションクラスにあります。デュアルカラーなどのレポートプロパティがオンになり、数式、ラベル、タイトルが **QbReport** オブジェクトに追加されます。

デュアルカラーは、次のコードを使用して設定されます。デュアルカラープロパティは各テーブルの列に設定され、代替の背景色、フォントの色、およびフォントが指定されます。

```
<Desired Report Column>.setAlternateRow(int numberOfRowsBeforeAlternateColor);
<Desired Report Column>.setBgColor2(Color alternateBackgroundColor);
<Desired Report Column>.setFontColor2(Color alternateFontCOlor);
<Desired Report Column>.setFont2(Font alternateFont);
```

レポートヘッダーにタイトルを追加するには、次のコードを使用します。**ReportCell** オブジェクトが最初に作成され、タイトルテキストが設定され、レポートヘッダーセクションに追加する前に、高さ、幅、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

x 位置、および y 位置などの **ReportCell** プロパティが指定されます。

```
ReportCell title = new ReportCell();
title.setText(String text);
title.setBgColor(Color backgroundColor);
title.setFontColor(Color fontColor);
title.setFont(Font font);
title.setAlign(int alignment);
title.setWidth(double width);
title.setHeight(double height);
title.setX(double xPosition);
title.setY(double yPosition);
<Handle to desired Report Section>.addData(title);
```

数式とラベルは同じ方法で指定されます。新しく作成された式またはラベルの **ReportCell** オブジェクトを適切なセクションに追加する前に、**ReportCell** オブジェクトが最初に作成され、式またはラベルセット、および **ReportCell** プロパティが指定されます。

```
ReportCell formulaCell = new ReportCell();
Formula formula = new Formula(String formulaname, String formulaText);
report.addFormula(formula);
formulaCell.setFormulaObj(formula);
formulaCell.setBgColor(Color backgroundColor);
formulaCell.setFontColor(Color fontColor);
formulaCell.setFont(Font font);
formulaCell.setAlign(int alignment);
formulaCell.setWidth(double width);
formulaCell.setHeight(double height);
formulaCell.setX(double xPosition);
formulaCell.setY(double yPosition);
<Handle to desired Report Section>.addData(formulaCell);

ReportCell label = new ReportCell();
label.setText(String text);
label.setBgColor(Color backgroundColor);
label.setFontColor(Color fontColor);
label.setFont(Font font);
label.setAlign(int alignment);
label.setWidth(double width);
label.setHeight(double height);
label.setX(double xPosition);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
label.setY(double yPosition);
<Handle to desired Report Section>.addData(label);
```

2.4.6 パラメータ化されたレポート

以下のセクションでは、既存のテンプレート（**<EspressReport InstallDir>/help/quickstart/templates** ディレクトリにある **QuickStart46.rpt**）を実行して、パラメータ化されたクエリをデータソースとして使用する方法を示します。

各セクションには、レポートを生成するためのコードと展開に必要な手順が示されています。

2.4.6.1 パラメータ値を渡す

次のコードは、アプリケーションでパラメータ化されたクエリを使用する既存のレポートテンプレート（この場合は **QuickStart46.rpt**）を表示する方法を示しています。パラメータ値は、レポートの作成時に渡されます。

```
import java.io.*;
import java.applet.*;
import java.awt.*;
import java.sql.*;
import java.util.Vector;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.swing.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart461 extends Applet {

        public QuickStart461() {
        };

        public static void main(java.lang.String[] args) {
                try {
                        QuickStart461 doReport = new QuickStart461();
                        Frame frame = new Frame();
                        frame.setLayout(new BorderLayout());
                        frame.add("Center", doReport.createReport(doReport));
                        frame.setSize(600, 600);
                        frame.setVisible(true);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            } catch (Exception ex) {
                ex.printStackTrace();
            }
        }

        Component createReport(Object parent) {
            //Connect to EspressManager
            QbReport.setEspressManagerUsed(true);
            //Begin Code : Set query parameters
            Vector vec = new Vector();
            vec.addElement("CA");
            vec.addElement("NH");
            Object queryParams[] = new Object[3];
            queryParams[0] = vec;
            queryParams[1] = new Date(99, 0, 4);
            queryParams[2] = new Date(101, 01, 12);
            //End Code : Set query parameters
            //Begin Code : Set formula parameter
            Object formulaParams[] = new Object[1];
            formulaParams[0] = "Ivan";
            //End Code : Set formula parameter
            //Create new Report object using specified report and parameter values
            QbReport report = new QbReport(parent,
                    "help/quickstart/templates/QuickStart46.rpt", queryParams,
                    formulaParams);

            //Show the Report
            Viewer viewer = new Viewer();
            Component comp = viewer.getComponent(report);
            return comp;

        }
}
```

上記のソースのクラスファイルは、
<EspressReport InstallDir>/help/quickstart/classes ディレクトリにあります。

クラスファイルが実行されると、**java QuickStart461** コマンドを使用して次のレポートが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Sales Report

Report Run By: Ivan

Orders From Jan 04, 1999 To Feb 12, 2001

Total Orders	Units Ordered	Total Sales	Average Sale
2	75	$68,118.00	$13,623.60

Order Details

Order ID: 10001 **Ordered by:** P & S Furniture
Order Date: Jan 14, 2001 49 Main Street
Littleton, NH 03561

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
12	Ra Dresser	$1,745.00	No	$425.00	$20,940.00
14	Shimaliya Chair	$424.00	Yes	$36.00	$6,440.00
12	Enlil Chair	$450.00	Yes	$27.00	$5,724.00
				Order Total:	**$33,104.00**

Order ID: 10002 **Ordered by:** P & S Furniture
Order Date: Jan 30, 2001 49 Main Street
Littleton, NH 03561

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
16	Set Dresser	$1,645.00	No	$427.00	$26,320.00
21	Nisaba Chair	$414.00	No	$29.00	$8,694.00
				Order Total:	**$35,014.00**

生成されたレポート

コードの主要部分は、**createReport** コンポーネントにあります。そこで、**QuickStart46.rpt** テンプレートを使用して **report** という **QbReport** オブジェクトが作成されます。パラメータ値は、次のコンストラクタを使用して渡されます。

```
QbReport(Object parent, String reportTemplatename, Object[] queryParameterValues, Qbject[] formulaParamterValues);
```

クエリパラメータと数式パラメータの値は、パラメータが定義された順序と同じ順序で配置されます。複数の値をとるクエリパラメータは、**Vector** オブジェクトとして宣言され、**Vector** オブジェクトには複数の異なる値が含まれます。

Access テンプレートを使用している場合、次のコード行を以下から変更する必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
queryParams[1] = new Date(99, 0, 4);
queryParams[2] = new Date(101, 01, 12);
```

から

```
queryParams[1] = new Timestamp(99, 0, 4, 0, 0, 0, 0);
queryParams[2] = new Timestamp(101, 01, 12, 0, 0, 0, 0);
```

2.4.6.2 getAllParameters を使用してパラメータ値を渡す

上記に加えて、**QbReport** の **getAllParameters** メソッドを使用してパラメータ値を渡すこともできます。次のコードは、アプリケーションでパラメータ化されたクエリを使用する既存のレポートテンプレート（この場合は **QuickStart46.rpt**）を表示する方法を示しています。バックアップデータを使用してレポートが開かれ（データベースに不必要な負荷がかからないようにするため）、パラメータ値が設定されます。その後、レポートがデータで更新されます。

```
import java.io.*;
import java.applet.*;
import java.awt.*;
import java.sql.*;
import java.util.Vector;
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.swing.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;

public class QuickStart462 extends Applet {

        public QuickStart462() {
        };

        public static void main(java.lang.String[] args) {
                try {
                        QuickStart462 doReport = new QuickStart462();
                        Frame frame = new Frame();
                        frame.setLayout(new BorderLayout());
                        frame.add("Center", doReport.createReport(doReport));
                        frame.setSize(600, 600);
                        frame.setVisible(true);
                } catch (Exception ex) {
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                ex.printStackTrace();
        }
}

Component createReport(Object parent) {
        //Connect to EspressManager
        QbReport.setEspressManagerUsed(true);
        //Begin Code : Set query parameters
        Vector vec = new Vector();
        vec.addElement("CA");
        vec.addElement("NH");
        Object queryParams[] = new Object[3];
        queryParams[0] = vec;
        queryParams[1] = new Date(99, 0, 4);
        queryParams[2] = new Date(101, 01, 12);
        //End Code : Set query parameters
        //Begin Code : Set formula parameter
        Object formulaParams[] = new Object[1];
        formulaParams[0] = "Ivan";
        //End Code : Set formula parameter
        //Create new Report object using backup data
        QbReport report = new QbReport(parent,
                        "help/quickstart/templates/QuickStart46.rpt", false, false,
                        false, true);

        try {

                report.getAllParameters().get(0).setValues((Vector) queryParams[0]);
                report.getAllParameters().get(1).setValue(queryParams[1]);
                report.getAllParameters().get(2).setValue(queryParams[2]);
                report.getAllParameters().get(3).setValue(formulaParams[0]);

                report.refreshWithOriginalData();

        } catch (Exception ex) {
                ex.printStackTrace();
        }

        //Show the Report
        Viewer viewer = new Viewer();
        Component comp = viewer.getComponent(report);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                return comp;
        }
}
```

上記のソースのクラスファイルは、**<EspressReport InstallDir>/help/quickstart/classes** ディレクトリにあります。

クラスファイルが実行されると、**java QuickStart462** コマンドを使用して、次のレポートが表示されます：

Sales Report

Report Run By: Ivan

Orders From Jan 04, 1999 To Feb 12, 2001

Total Orders	Units Ordered	Total Sales	Average Sale
2	**75**	**$68,118.00**	**$13,623.60**

Order Details

Order ID: 10001 **Ordered by:** P & S Furniture
Order Date: Jan 14, 2001 49 Main Street
Littleton, NH 03561

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
12	Ra Dresser	$1,745.00	No	$425.00	$20,940.00
14	Shimaliya Chair	$424.00	Yes	$36.00	$6,440.00
12	Enlil Chair	$450.00	Yes	$27.00	$5,724.00
				Order Total:	**$33,104.00**

Order ID: 10002 **Ordered by:** P & S Furniture
Order Date: Jan 30, 2001 49 Main Street
Littleton, NH 03561

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
16	Set Dresser	$1,645.00	No	$427.00	$26,320.00
21	Nisaba Chair	$414.00	No	$29.00	$8,694.00
				Order Total:	**$35,014.00**

生成されたレポート

コードの主要部分は、**createReport** コンポーネントにあります。そこで、**QuickStart46.rpt** テンプレートを使用して **report** という **QbReport** オブジェクトが作成されます。パラメータ値は、**QbReport** の以下のインタフェースを使用して渡されます：

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
getAllParameters();
```

クエリパラメータと数式パラメータの値は、パラメータが定義された順序と同じ順序で配置されます。複数の値をとるクエリパラメータは、**Vector** オブジェクトとして宣言され、異なる複数の値が含まれます。

Access テンプレートを使用している場合、次のコード行を以下から変更する必要があります：

```
queryParams[1] = new Date(99, 0, 4);
queryParams[2] = new Date(101, 01, 12);
```

から

```
queryParams[1] = new Timestamp(99, 0, 4, 0, 0, 0, 0);
queryParams[2] = new Timestamp(101, 01, 12, 0, 0, 0, 0);
```

2.4.6.3 getParameterPage()と ParamReportGeneratorServlet を使用する

次のコードは、サーブレットでパラメータ化されたクエリを使用する既存のレポートテンプレート（この場合は **QuickStart46.rpt**）を表示する方法を示しています。サーブレットは、バックアップデータを使用してテンプレートを開いて **QbReport** オブジェクトを作成します。次に、パラメータ値を要求する HTML ページがストリーミングされます。これらの値は別のサーブレット（**ParamReportGeneratorServlet** サーブレット）に渡され、**QbReport** オブジェクトは指定されたパラメータ値を持つ指定されたテンプレートから作成されます。

```
import quadbase.reportdesigner.ReportAPI.QbReport;
import quadbase.common.param.*;
import java.applet.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.awt.Font;

public class QuickStart463 extends HttpServlet {

        public void doGet(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {

                System.out.println("QuickStart463 Servlet: Do Get....");
                System.out
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

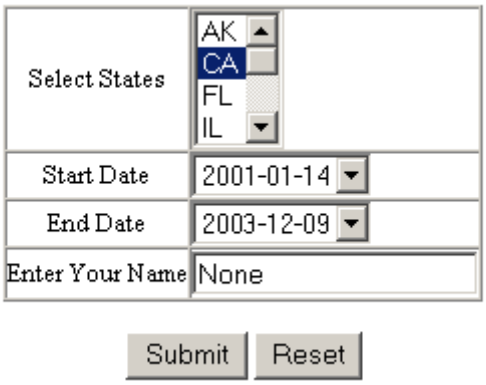
```
                .println("QuickStart463 Servlet: Generate paramter html
page...");
        res.setContentType("text/html");

        OutputStream toClient = res.getOutputStream();
        try {
            //Connect to EspressManager
            QbReport.setEspressManagerUsed(true);
            String reportLocation = "help/quickstart/templates/QuickStart46.rpt";
            //Create the ObReport object using back-up data
            QbReport report = new QbReport((Applet) null, reportLocation,
                    false, false, false, true, false);

            //Specify the parameters for connecting to
            //ParamReportGeneratorServlet
            report.setDynamicExport(true, "127.0.0.1", 8080);

            //Specify report template location and export format desired
            ParameterPage paramPage = report.getParameterPage(reportLocation,
                    null, QbReport.DHTML, null);
            Writer writer = new PrintWriter(toClient);
            HtmlParameterPageWriter paramPageWriter = new
HtmlParameterPageWriter(
                    paramPage, writer);
            paramPageWriter.writePage();
            writer.flush();
            writer.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        toClient.flush();
        toClient.close();

    }

    public String getServletInfo() {
        return "QuickStart463 servlet for EspressReport";
    }
}
```

サーブレットをデプロイするには：

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

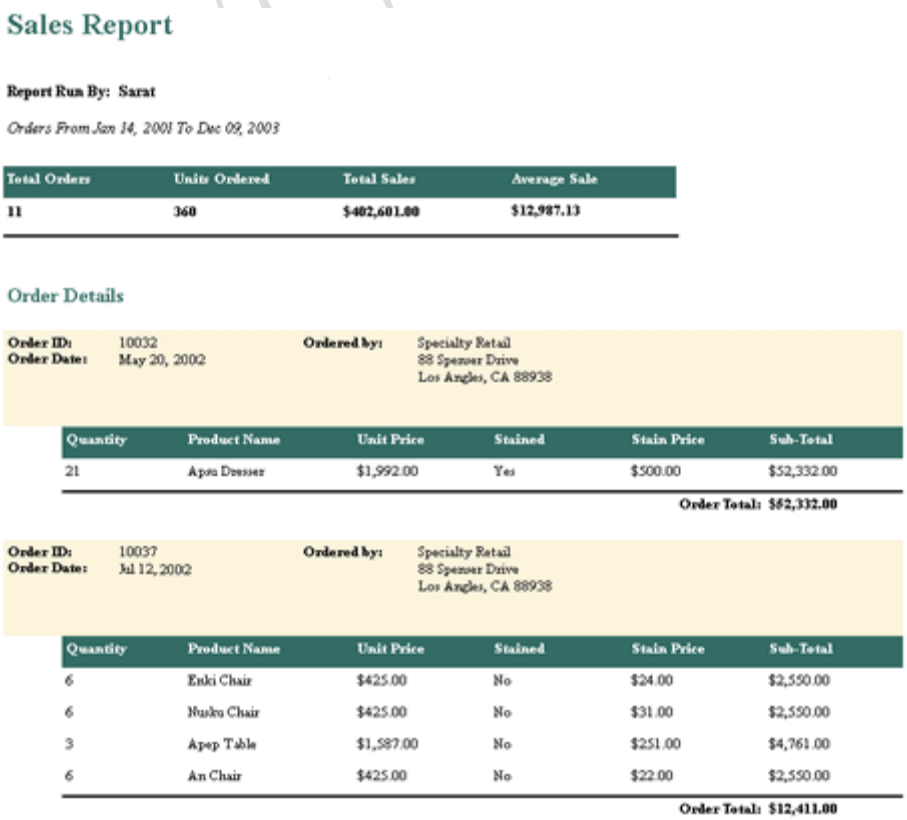
- **QuickStart463.class** を**<EspressReport InstallDir>/help/quickstart/classes** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- **ParamReportGeneratorServlet <EspressReport InstallDir>/ParamReportGenerator** をコンパイルしてください
- **ParamReportGeneratorServlet.class** を **<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。

URL **http://localhost:8080/servlet/QuickStart463** を使用してサーブレットを実行すると、次のHTML ページが表示されます。



生成された HTML プロンプトページ

どのパラメータ値を渡すかによって、次のようなレポートが表示されます。



Sales Report

Report Run By: Sarat

Orders From Jan 14, 2001 To Dec 09, 2003

Total Orders	Units Ordered	Total Sales	Average Sale
11	360	$402,601.00	$12,987.13

Order Details

Order ID: 10032 **Ordered by:** Specialty Retail
Order Date: May 20, 2002 88 Spenser Drive
Los Angles, CA 88938

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
21	Apsu Dresser	$1,992.00	Yes	$500.00	$52,332.00

Order Total: $52,332.00

Order ID: 10037 **Ordered by:** Specialty Retail
Order Date: Jul 12, 2002 88 Spenser Drive
Los Angles, CA 88938

Quantity	Product Name	Unit Price	Stained	Stain Price	Sub-Total
6	Enki Chair	$425.00	No	$24.00	$2,550.00
6	Nudru Chair	$425.00	No	$31.00	$2,550.00
3	Apsp Table	$1,587.00	No	$251.00	$4,761.00
6	An Chair	$425.00	No	$22.00	$2,550.00

Order Total: $12,411.00

パラメータ値を渡した後に生成されるレポート

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

コードの主要部分は **QuickStart463** にあります。そこで、**QuickStart46.rpt** テンプレートを使用して **report** という **QbReport** オブジェクトが作成されます。次に、動的エクスポートが呼び出され、（EspressReport で提供される）**ParamReportGeneratorServlet** が次の行を使用して使用されます。

```
<QbReport object>.setDynamicExport(boolean isDynamicExport, String servername, int servletRunnerPort);
String htmlParamPage = <QbReport object>.getHTMLParamPage(String reportLocation, int exportFormat);
```

パラメータ値を要求する HTML ファイルが生成され、OutputStream に渡されます。

パラメータページライタークラスの詳細は、HTML パラメータページの生成を参照してください。CSS を使用する **CssHtmlParameterPageWriter** を使用してパラメータページを生成する例については、**<EspressReport>/help/examples/servicelet/CssParamReport** にアクセスし、**javadoc** の指示に従ってください。

2.4.7 ドリルダウンの展開/エクスポート

次のコードは、ドリルダウンレポートである既存のレポートテンプレート（この場合は **QuickStart47.rpt**）をサーブレットに表示する方法を示しています。サーブレットはテンプレートを開いて **QbReport** オブジェクトを作成します。最初のレベルの内容を示す DHTML ページがストリーミングされます。次のレベルのレポートは、リンクをクリックすると得られます。これらのリンクは **DrillDownReportServlet** を指します。（クリックされた）リンクの値は **DrillDownReportServlet** に渡され、それらの値に基づく次のレベルのレポートが生成され、クライアントにストリーミングされます。

```
import quadbase.reportdesigner.ReportAPI.QbReport;
import java.applet.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.awt.Font;

public class QuickStart47 extends HttpServlet {

        public void doGet(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {

                System.out.println("QuickStart47 Servlet: Do Get....");
                res.setContentType("text/html");
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                OutputStream toClient = res.getOutputStream();
                try {
                        //Connect to EspressManager
                        QbReport.setEspressManagerUsed(true);
                        String reportLocation = "help/quickstart/templates/QuickStart47.rpt";
                        //Create the ObReport object using back-up data
                        QbReport report = new QbReport((Applet) null, reportLocation);
                        //Specify the parameters for connecting to DrillDownReportServlet
                        report.setDynamicExport(true, "127.0.0.1", 8080);
                        //Export the report to DHTML
                        report.export(QbReport.DHTML, toClient);
                } catch (Exception e) {
                        e.printStackTrace();
                }

                toClient.flush();
                toClient.close();

        }

        public String getServletInfo() {
                return "QuickStart47 servlet for EspressReport";
        }
}
```

サーブレットをデプロイする：

- **QuickStart47.class** を**<EspressReport InstallDir>/help/quickstart/classes** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- **DrillDownReportServlet<EspressReport InstallDir>/DrillDownLinkGenerator** をコンパイルします。
- **DrillDownReportServlet.class** を**<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- **RPTImageGenerator<EspressReportInstall Dir>/ImageGenerator** をコンパイルします。
- **RPTImageGenerator.class** を**<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- Tomcat 7.x 以上のバージョンでは、サーブレット（この例では **QuickStart482**）を web.xml に明示的にマップする必要があります。

```
<servlet>
                <servlet-name>QuickStart47</servlet-name>
                <servlet-class>QuickStart47</servlet-class>
```

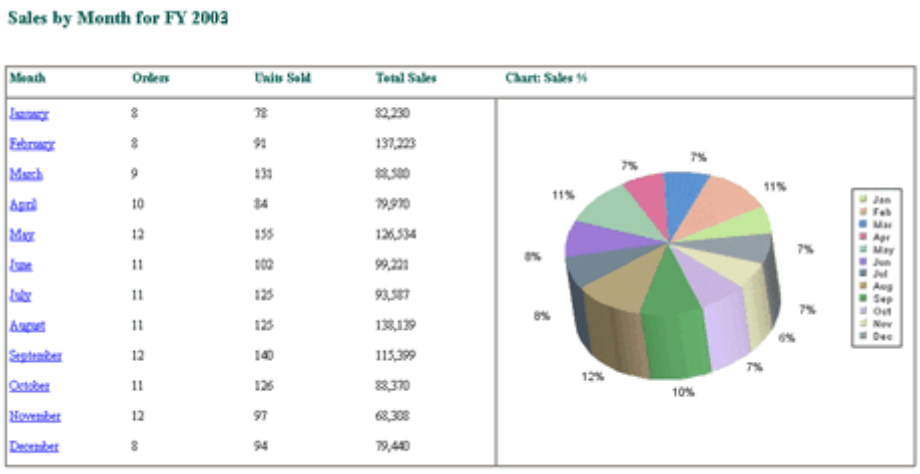

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
</servlet>
```

かつ

```
<servlet-mapping>
        <servlet-name>QuickStart47</servlet-name>
        <url-pattern>/servlet/QuickStart47</url-pattern>
</servlet-mapping>
```

URL **http://localhost:8080/servlet/QuickStart47** を使用してサーブレットを実行すると、次の HTML ページが表示されます。



Sales by Month for FY 2003

Month	Orders	Units Sold	Total Sales	Chart: Sales %
January	8	78	82,230	
February	8	91	137,223	
March	9	131	88,580	
April	10	84	79,970	
May	12	155	126,534	
June	11	102	99,221	
July	11	125	93,587	
August	11	125	138,139	
September	12	140	115,399	
October	11	126	88,370	
November	12	97	68,308	
December	8	94	79,440	

生成されたレポート

クリックしたリンクに応じて、次のようなレポートが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Orders for February 2003

Order ID	Customer	Quantity	ProductName	Sub-Total
10056	American Signature Furniture	12	Amon Table	18,468
		16	Anubis Table	31,216
			Order Total:	**49,684**
10057	Lou Rippners Compass	9	Ishtar Chair	3,051
		19	Nun Dresser	29,412
		12	Het Table	15,060
			Order Total:	**47,523**
10058	Munire Furniture	8	Isis Table	14,312
		9	Tefnut Dresser	16,182
		6	Atun Table	9,522
			Order Total:	**40,016**
			Total Sales for February:	**137,223**

リンクをクリックした後に生成されるレポート

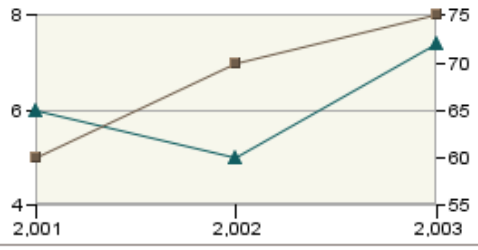
繰り返しますが、クリックしたリンクに応じて、次のようなレポートが表示されます。

Customer Details

Company: Munire Furniture **Address:** 91 New England Aveue
Contact: Muriel Fedder
Customer Since: Jun 12, 2001

Year	Orders	Units Ordered	Total Sales
2001	5	65	67,511
2002	7	60	97,528
2003	8	72	96,760

Chart: Orders / Units Ordered



リンクをクリックした後に生成されるレポート

コードの主要部分は **QuickStart47** にあります。そこで、**QuickStart47.rpt** テンプレートを使用して、**report** という **QbReport** オブジェクトが作成されます。次に、次の行を使用して、**DrillDownReportServlet**（EspressReport に付属）を使用するために動的エクスポートが呼び出されます。

```
<QbReport object>.setDynamicExport(boolean isDynamicExport, String servername, int servletRunnerPort);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

トップレベルレポートの DHTML ファイルが生成され、OutputStream に渡されます。

```
<QbReport object>.export(int exportFormat, OutputStream out);
```

2.4.8 サーブレットの詳細

以下のセクションでは、既存のテンプレート（**<EspressReport InstallDir>/help/quickstart/templates** ディレクトリにある **QuickStart48.rpt**）をサーブレットで実行し、生成されたレポートをファイルに保存したり、クライアントブラウザにストリーミングしたりする方法を示します。

各セクションには、レポートを生成するためのコードと展開に必要な手順が示されています。

2.4.8.1 ファイルへの書き込み（PDF）

次のコードは、既存のレポートテンプレート（この場合は **QuickStart48.rpt**）をサーブレットに表示する方法を示しています。サーブレットはレポートを PDF ファイルにエクスポートします。

```
import quadbase.reportdesigner.ReportAPI.*;
import quadbase.reportdesigner.ReportElements.*;
import quadbase.reportdesigner.ReportViewer.*;
import quadbase.reportdesigner.util.*;
import quadbase.reportdesigner.lang.*;
import java.awt.*;
import java.applet.*;
import java.io.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class QuickStart481 extends HttpServlet {

        public void doGet(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
                System.out.println("Calling QuickStart481....");
                //Set the "content type" header of the response
                res.setContentType("text/html");
                //Get the response's PrintWriter to return content to the client.
                PrintWriter toClient = res.getWriter();
                try {
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                        //Use EspressManager
                        QbReport.setEspressManagerUsed(true);
                        //Open up specified Report
                        QbReport report = new QbReport((Applet) null,
                                        "help/quickstart/templates/QuickStart48.rpt");
                        //Export the report to PDF file
                        report.export(QbReport.PDF, "generatedReport");
                        //Begin Code : Send message to client saying file has been exported
                        toClient.println("<html>");
                        toClient.println("<head>");
                        toClient.println("</head>");
                        toClient.println("<body>");
                        toClient.println("The report has been exported to generatedReport.pdf in
your EspressReport installation root directory.");
                        toClient.println("</body>");
                        toClient.println("</html>");
                        //End Code : Send message to client saying file has been exported
                } catch (Exception e) {
                        e.printStackTrace();
                }
                //Flush the outputStream
                toClient.flush();
                //Close the writer; the response is done.
                toClient.close();
        }

        public String getServletInfo() {
                return "QuickStart481 servlet for EspressReport";
        }
}
```

サーブレットをデプロイするには：

- **QuickStart481.class** を**<EspressReport InstallDir>/help/quickstart/classes** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。

http://localhost:8080/servlet/QuickStart481 という URL を使用してサーブレットを実行すると、**<EspressReport InstallDir>/generatedReport.pdf** にレポートが生成され、次のメッセージが表示されます。

The report has been exported to generatedReport.pdf in your EspressReport installation root directory.

レポートが生成された後に表示されるメッセージ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

コードの主要部分は **QuickStart481** にあります。そこで、**QuickStart48.rpt** テンプレートを使用して、**report** という **QbReport** オブジェクトが作成されます。次のコンストラクタが使用されます。**QbReport** オブジェクトレポートは、エクスポートメソッドを使用して PDF としてエクスポートされます。

<QbReport object>.export(**int** exportFormat, String exportFilename);

2.4.8.2 ストリームチャート（DHTML）

次のコードは、既存のレポートテンプレート（この場合は **QuickStart48.rpt**）をサーブレットに表示する方法を示しています。サーブレットは、DTHML ドキュメントとしてレポートを（チャートとともに）クライアントにストリームします。

```
import quadbase.reportdesigner.ReportAPI.QbReport;
import java.applet.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.awt.Font;

public class QuickStart482 extends HttpServlet {

        public void doGet(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
                System.out.println("QuickStart482 Servlet: Do Get....");
                res.setContentType("text/html");
                OutputStream toClient = res.getOutputStream();
                try {
                        //Connect to EspressManager
                        QbReport.setEspressManagerUsed(true);
                        String reportLocation = "help/quickstart/templates/QuickStart48.rpt";
                        //Create the ObReport object
                        QbReport report = new QbReport((Applet) null, reportLocation);

                        //Specify the parameters for connecting to RPTImageGenerator
                        report.setDynamicExport(true, "127.0.0.1", 8080);

                        //Stream report to client
                        report.export(QbReport.DHTML, toClient);
                } catch (Exception e) {
                        e.printStackTrace();
                }
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                toClient.flush();
                toClient.close();

        }

        public String getServletInfo() {
                return "QuickStart482 Servlet for EspressReport";
        }
}
```

サーブレットをデプロイするには：

- **QuickStart482.class** を<**Tomcat InstallDir**>/**webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- **RPTImageGenerator**<**EspressReport InstallDir** >/**ImageGenerator** をコンパイルします。
- **RPTImageGenerator.class** を<**Tomcat InstallDir**>/**webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- Tomcat 7.x 以上のバージョンでは、サーブレット（この例では QuickStart482）を web.xml に明示的にマップする必要があります：

```
<servlet>
        <servlet-name>QuickStart482</servlet-name>
        <servlet-class>QuickStart482</servlet-class>
</servlet>
```

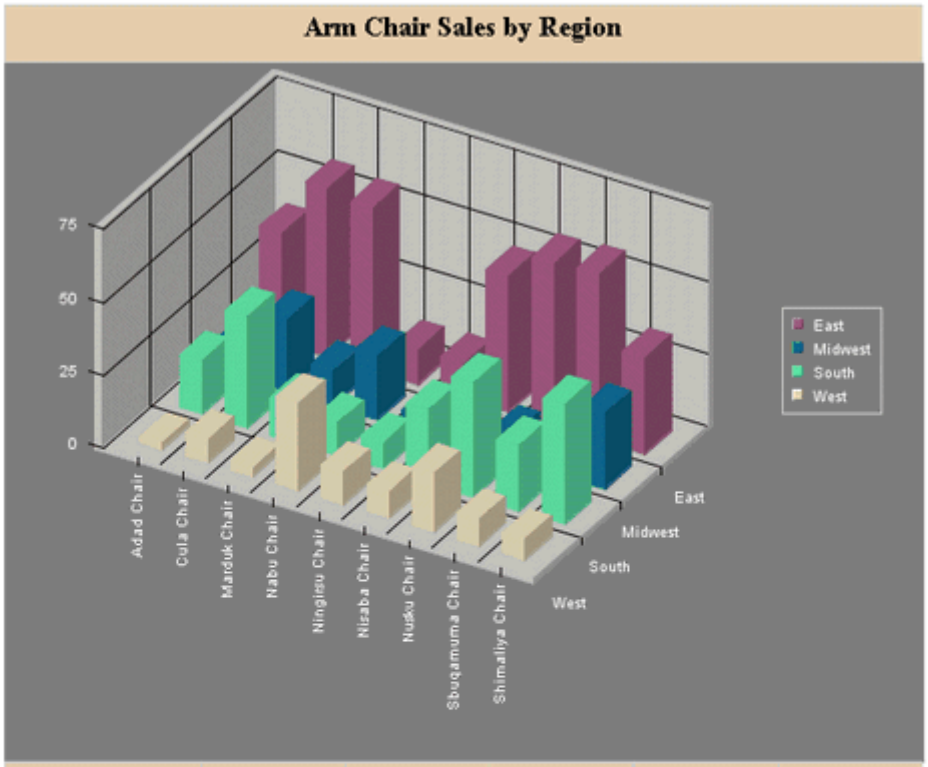
かつ

```
<servlet-mapping>
        <servlet-name>QuickStart482</servlet-name>
        <url-pattern>/servlet/QuickStart482</url-pattern>
</servlet-mapping>
```

URL **http://localhost:8080/servlet/QuickStart482** を使用してサーブレットを実行すると、次の HTML ページが表示されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Product Name	East	Midwest	South	West	Units Sold
Adad Chair	38	12	18	2	70
Cula Chair	58	25	38	8	129
Marduk Chair	56	12	13	3	84
Nabu Chair	12	22	11	30	75
Ningirsu Chair	12	6	9	11	38
Nisaba Chair	47	17	25	9	98
Nusku Chair	56	9	39	20	124
Shugamuma Chair	57	8	22	9	96
Shimaliya Chair	33	26	41	7	107
Total:	**369**	**137**	**216**	**99**	**821**

生成されたレポート

コードの主要部分は **QuickStart482** にあります。そこで、**QuickStart48.rpt** テンプレートを使用して、**report** という **QbReport** オブジェクトが作成されます。次に、次の行を使用して、**RPTImageGenerator**（EspressReport で提供）を使用するために動的エクスポートが呼び出されます。

```
<QbReport object>.setDynamicExport(boolean isDynamicExport, String servername, int servletRunnerPort);
```

トップレベルレポートの DHTML ファイルが生成され、OutputStream に渡されます。

```
<QbReport object>.export(int exportFormat, OutputStream out);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2.4.9 Report Designer の起動

次のコードは、Report API を使用して Report Designer を（デフォルトモードで）起動する方法を示しています：

```
import java.awt.*;
import javax.swing.*;
import java.io.*;
import quadbase.reportdesigner.designer.*;

public class QuickStart49 {

	public static void main(java.lang.String[] args) {

		try {
			QuickStart49 doReport = new QuickStart49();
		} catch (Exception ex) {
			ex.printStackTrace();
		}
	}

	public QuickStart49() {

		//Begin Code : Start Designer in default mode and show the Designer
		QbReportDesigner.setUseSysResourceImages(true);
		QbReportDesigner designer = new QbReportDesigner(null);
		designer.setVisible(true);
		//End Code : Start Designer in default mode and show the Designer
	}
}
```

上記のソースのクラスファイルは、**<EspressReport InstallDir>/help/quickstart/classes** ディレクトリにあります。

クラスファイルを実行するには、**QuickStart49.class** を**<EspressReport InstallDir>**に移動し、**java QuickStart49** コマンドを使用します。

コードの主要部分は **QuickStart49** にあります。そこで、designer という **QbReportDesigner** オブジェクトが作成され、表示されます。

```
QbReportDesigner(Object parent);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

<QbReportDesigner object>.setVisible(**boolean** b);

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3 デザイナーガイド

3.1 概要

100%Pure Java で書かれた EspressReport は、多くのデータソースからプロフェッショナルで情報豊富なレポートを簡単に作成してアプリケーションやウェブベースなどに組み込むことを可能にする一連のツールです。EspressReport には、Report Designer と堅牢なオブジェクト指向の API が付属しています。Report Designer は、強力で使いやすいグラフィカルユーザーインタフェースを提供し、コンテンツをすばやく編集してフォーマットすることができ、生データを数分で洗練されたレポートに変換できます。完成したレポートデザインは、最新のデータを使用して後で配布するためにテンプレートに保存することができます。Web 上でレポートを公開するには、API を使用して JSP/サーブレットコードを記述してテンプレートを参照し、目的の形式でレポートを生成します。サポートされているエクスポート形式には、PDF、DHTML、text、Excel（XLS）および Excel 2007（XLSX）スプレッドシートがあります。また、Report Viewer や EspressReport の API をアプレットとして実行し、後に WYSIWYG ハードコピーで印刷することで、レポートをブラウザで表示することもできます。

3.1.1 EspressReport ドキュメント

EspressReport のドキュメントは以下のセクションに分かれています。

クイックスタートガイド:

これは EspressReport をこれから始める初心者に良い出発点です。最も一般的に使用される機能をカバーし、Report Designer と API チュートリアルを提供します。

デザイナーガイド:

ここでは、Report Designer のすべての機能について説明し、クエリビルダでクエリを作成し、データを整形して操作して洗練されたレポートを作成する方法を教えています。

プログラミングガイド:

ここでは、Report API について説明し、レポートをプログラムで作成する方法と、サーブレット、JSP、およびアプリケーションにレポートを組み込む方法について説明します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.2 アーキテクチャとインストール

EspressReport は 7 つの主要コンポーネントで構成されています。

Report Designer:

Report Designer は、ユーザがポイントアンドクリック環境でレポートを作成できる GUI ツールです。Report Designer には、データ、クエリデータベース、およびデザインチャートにアクセスするためのインタフェースが含まれています。Report Designer は、クライアントアプリケーションとして、または Web ブラウザを介してアプレットとしてロードされたデザイナを使用してクライアント/サーバ構成で実行できます。

Chart Designer:

Chart Designer は GUI ツールで、ポイントアンドクリック環境でチャートを作成してデザインすることができます。Report Designer から起動し、レポート API またはレポート API を使用して実行できるスタンドアロンチャートに埋め込むことのできるチャートを作成できます。

Report API:

Report API は使いやすいアプリケーションプログラミングインタフェースで、ユーザはサーバ側またはクライアント側のいずれかのアプリケーション、サーブレット、または JSP にレポート機能を組み込むことができます。これは純粋な Java であるため、変更が少ないほとんどのプラットフォームで実行できます。レポートのすべての部分は API を使用してカスタマイズ可能で、ユーザは実行時にレポート書式を完全にコントロールできます。わずか数行のコードでレポートを作成し、展開することができます。

Report Viewer:

Report Viewer は、ユーザがレポートを表示して対話できるようにする統合されたアプレットです。アプレットでは、ページ単位のレポートが表示され、ユーザはポップアップメニューを使用してレポートをナビゲートできます。また、レポートテンプレートは、アプレットを使用して直接ハイパーリンクすることもできます。EspressReport は、コードを必要とせずに、埋め込まれたアプレットを含む HTML ページを生成することができます。

Page Viewer:

Page Viewer は、ページ表示技術を使用するレポートビューアのような統合アプレットです。Page Viewer では、レポートのページは要求されたときにのみクライアントに送信されます。これにより、ユーザは大きなレポートを表示/プレビューできます。

EspressManager:

EspressManager は、Report Designer と Scheduler のインタフェースの”バックエンド”として機能します。Report Designer はアプレットとして実行されます。EspressManager はサーバ側でデータアクセスとファイル I/O アクティビティを処理します。さらに、EspressManager はデータベース接続とデータバッファリングを提供します。EspressManager はまた、サーバ側

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

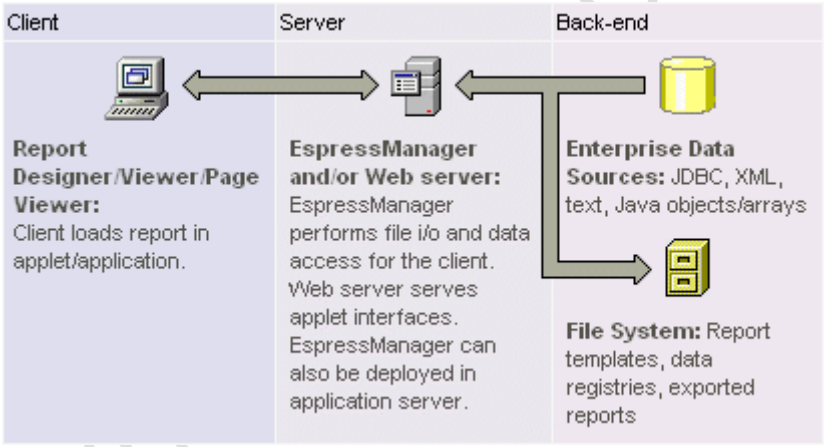
でスケジューリングプロセスを実行し、ユーザ定義のジョブに従ってレポートを実行します。サーバ上でアプリケーションプロセスとして実行することも、アプリケーションサーバ内にサーブレットとしてデプロイすることもできます。

Scheduler:

Scheduler は、EspressManager と連携して動作する小さなインタフェースです。ユーザは、レポートやその他のイベントのスケジュール設定、スケジュールされたタスクの管理を行うことができます。

3.2.1 EspressReport アーキテクチャ

EspressReport には、設計時と実行時の両方で実行できるさまざまな構成があります。設計時に、データアクセスツールとチャートツールを含む Report Designer の GUI インタフェースは、クライアントマシン上のアプリケーションとして、またはクライアントサーバアーキテクチャ内のアプレットとしてロードできます。次の図は、設計時に Report Designer で実行されている EspressReport を示しています。

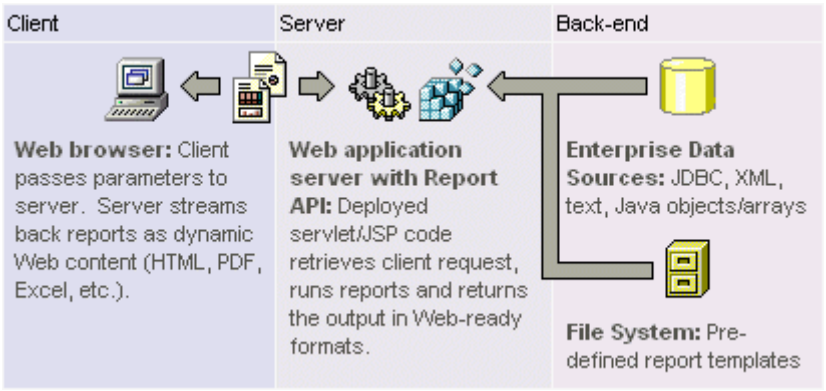


EspressReport デザイナアーキテクチャ

Report Designer が実行されているとき、EspressManager コンポーネントはサーバ側で実行されます。これは、アプリケーションプロセスとして、またはアプリケーションサーバ/サーブレットランナー内にデプロイされたサーブレットとして実行できます。EspressManager は、セキュリティ制限のためにクライアントアプレットによって防止されるデータアクセスとファイル I/O を実行します。
EspressManager は、データベース接続のための接続とデータのバッファリングと、マルチユーザ開発環境の同時実行制御も提供します。EspressManager は、Report Designer と連携して実行する必要があります。

実行時に、EspressManager を実行する必要はありません。EspressReport は、他のアプリケーション環境に組み込まれて動作するように設計されています。API クラスとレポートテンプレートファイルの配備を最小限に抑えることができます。次の図は、サーブレット/JSP 環境で実行されている EspressReport を示しています。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



サーブレット環境での EspressReport

アプリケーションサーバで実行する場合、Report API を使用してサーブレットと JSP 内でレポートを生成し、生成されたレポートをクライアントブラウザにストリーミングできます。クライアントは、Report Viewer アプレットまたは Page Viewer アプレットをロードして、レポートを表示することもできます。

レポート生成は、オンデマンドで処理したり、アプリケーションプロセスによってトリガされたり、スケジュールされたりすることがあります。EspressReport にはスケジューラーインタフェースも用意されています。スケジューラを実行するには、EspressManager が実行されている必要があります。

EspressReport は、クライアント/サーバ環境で実行するだけでなく、シッククライアントアプリケーションでも実行できます。Report API は、ビューアおよび Page Viewer と組み合わせて使用して、アプリケーションインタフェースでレポートを表示または生成することができます。この構成では、プロセス全体をクライアント側に含めることができるため、EspressManager を実行する必要はありません。これはアプレットの場合ではないことに注意してください。

3.2.2 インストール

EspressReport インストーラには、Windows、Unix、Mac OS X、Pure Java バージョンの 4 種類があります。

Windows

Windows インストーラを起動するには、**installERES.exe** ファイルを実行すると、インストーラが起動します。

Unix

Unix インストーラを起動するには、**installERES.bin** ファイルを実行すると、インストーラが起動します。

Mac OS X

Mac インストーラを起動するには、**InstallERES.zip** ファイルをダブルクリックして

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

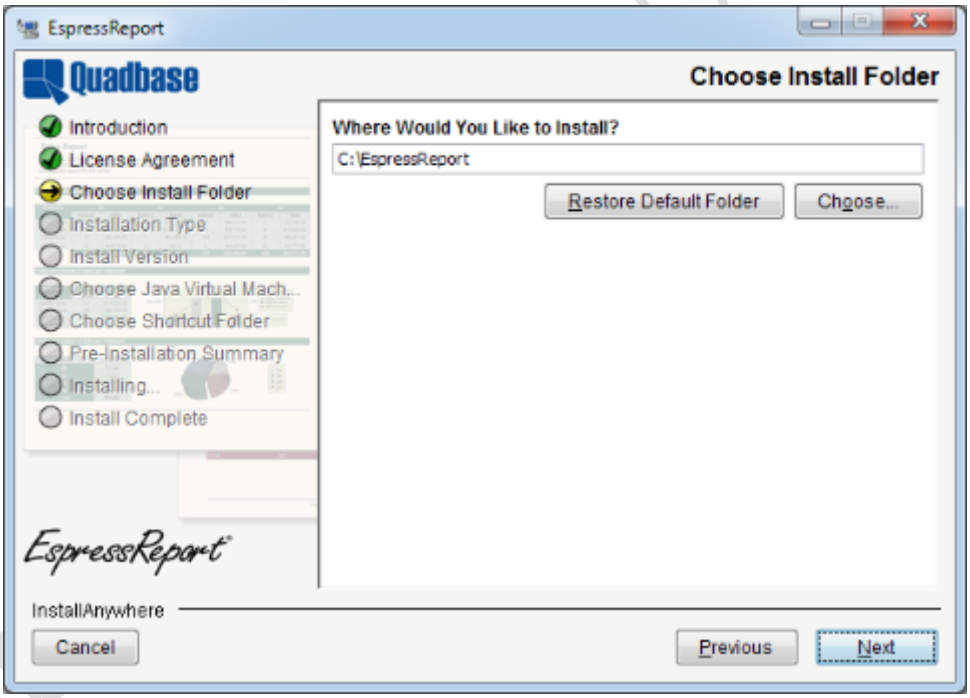
InstallERES.app ファイルを解凍します。**InstallERES.app** をダブルクリックすると、インストーラが起動します。

Pure Java:

Pure Java インストーラを開始するには、EspressReport がインストールされるマシンに、Java 1.2 以上の Java Virtual Machine がインストールされている必要があります。JVM がパスに含まれていることを確認します（または **installER.jar** ファイルを JVM と同じディレクトリに移動します）。コマンドプロンプトから、**installERES.jar** ファイルを配置したディレクトリに移動し、**java -jar installERES.jar** コマンドを入力します。インストーラが起動します。

インストールプログラムが起動し、使用許諾契約書に同意すると、最初のオプションで、EspressReport をインストールするディレクトリを指定するよう求められます。

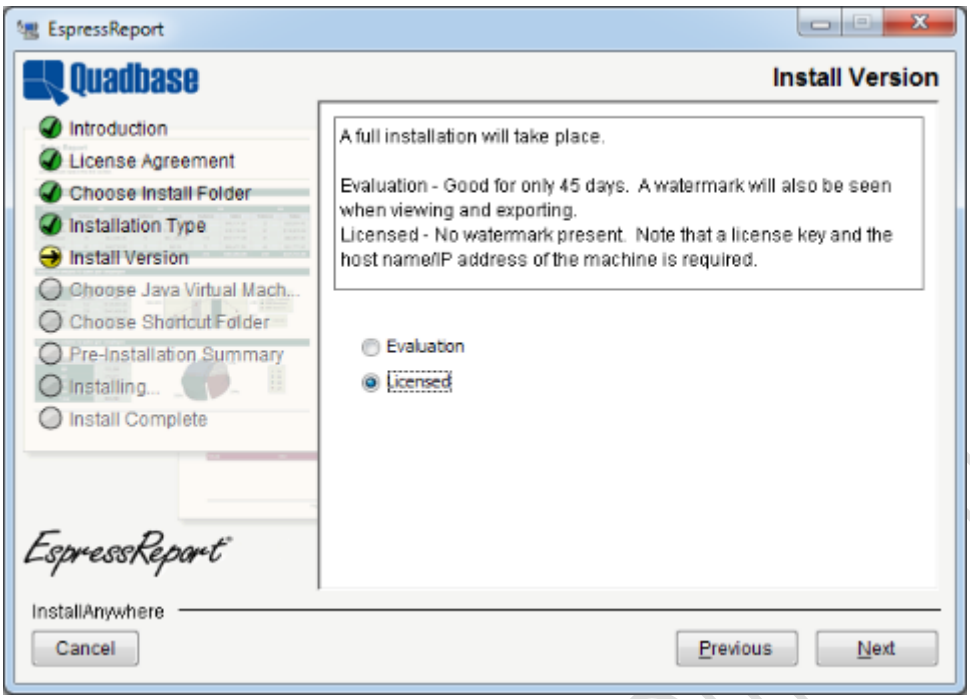
デフォルトの場所は**¥EspressReport¥**です。**Choose…**ボタンをクリックすると、新しいディレクトリを指定したり、ディレクトリを参照することができます。



Select Location ダイアログ

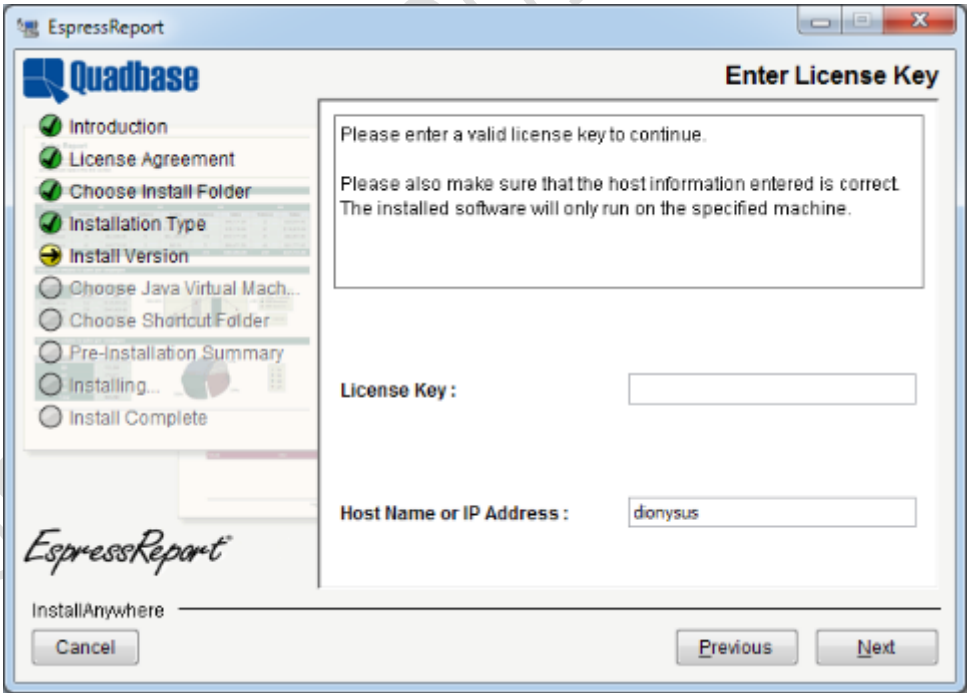
インストールディレクトリを選択すると、次のオプションで、評価版またはライセンス版のどちらをインストールするかを選択できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Version Select ダイアログ

ライセンス版をインストールすることを選択した場合は、ライセンスキーを入力して EspressReport をインストールするマシンのホスト名を確認するダイアログが表示されます。



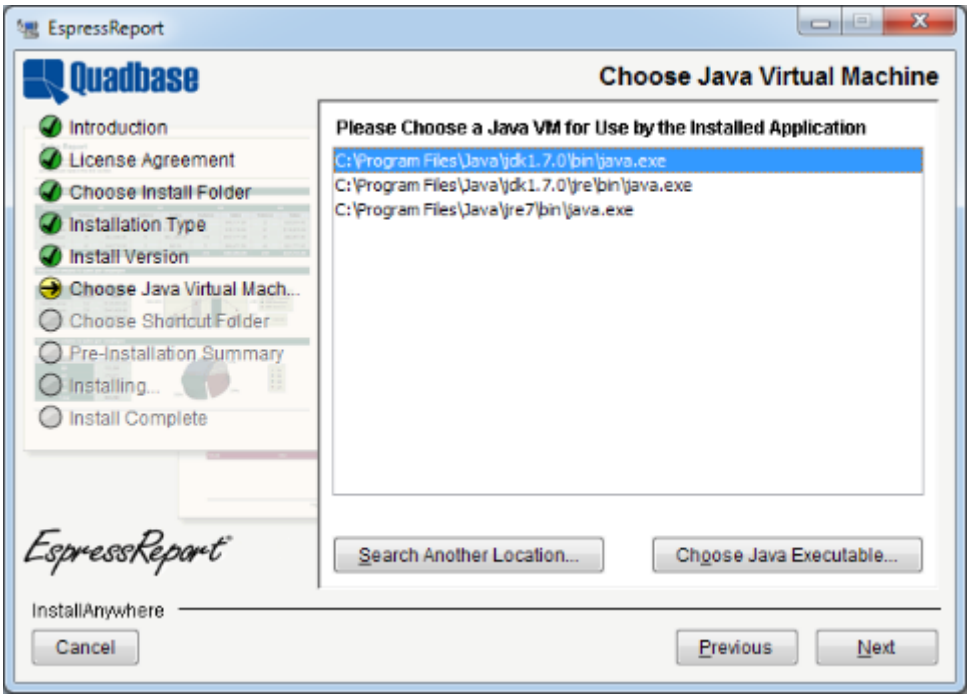
Enter License Key ダイアログ

すべての情報を入力すると、インストーラはライセンスキーを Quadbase に登録しようとします。登録に失敗すると、EspressReport のリリース版をインストールすることができなくなります。ただし、評価版をインストールすることはできます。インストールが完了したら、http://www.quadbase.com/register/でキーを登録するか、弊社まで問い合わせて追加のヘルプを入手してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

インストールが完了すると、リリースバージョンはこのダイアログで指定されたホスト名に対してのみ実行されるため、正しいことを確認するためにダブルチェックしてください。必要に応じて、マシンの IP アドレスを使用することもできます。

次のウィンドウでは、EspressReport の実行に使用する Java Virtual Machine のバージョンを指定できます。（これは Java プログラムなので、実行するには JVM が必要です。）



Choose JVM ダイアログ

このダイアログでは、使用する Java Virtual Machine を選択するように要求されます。システム上の互換性のある JVM のリストが表示され、選択することができます。使用する JVM がリストされていない場合は、**Search Another Location...**ボタンをクリックして参照することができます。互換性のある JVM が見つからない場合は、Windows インストーラを使用する際に、インストーラにバンドルされている JVM をインストールすることもできます。

インストーラにバンドルされている JVM は、Java Runtime Environment のみを提供します。EspressReport API を使用して Java コードを開発して実行するには、Java Development Kit または compiler/IDE が必要です。また、Pure Java または Mac OS X バージョンのインストーラを使用している場合、このオプションは表示されません。プログラムを実行する JVM でインストーラを実行していることを確認してください。

インストーラの最後のオプションは、Windows または Mac OS X 版のインストーラです。プログラムのショートカットの場所を指定することができます。デフォルトでは、ショートカットはスタートメニューの **EspressReport** というプログラムグループに追加されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Mac OS X の場合、デスクトップ、ドック、または任意のフォルダにエイリアスを作成できます。

最後のオプションを完了すると、選択したすべてのオプションの概要が表示されます。**Next** ボタンをクリックすると、プログラムがインストールされます。

3.2.3 設定

EspressReport の設定は既に設定されており、何も変更せずに実行できます。ただし以下の設定を変更することができます:

- EspressManager が使用するポート番号の変更
- Web ルートの変更
- ユーザの追加/削除/変更

これらの変更を行う場合構成設定を変更する必要があります。設定は **config.txt** というテキストファイルに保存されており次の場所にあります：**<EspressReport InstallDir>/userdb/config.txt**
設定ファイルの例を以下に示します。

```
[port]
22071

[webroot]
C:¥Intepub¥wwwroot

[users]
guest
Name1 Password1
Name2 Password2

[smtp host]
quadbase.com

[smtp secured]
true

[smtp ssl]
true

[smtp port]
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
587

[smtp username]
guest

[smtp password]
password
```

[**port**]セクションの下で、EspressManager が使用するポート番号を設定できます。この番号はデフォルトで **22071** に設定されています。EspressReport にマシンに IP アドレスを問い合わせさせる代わりに、EspressManager が使用する明示的な IP アドレスをここに与えることもできます。明示的な IP アドレスは、ポート番号の後にアドレスを追加することで指定できます。たとえば、次のように変更します。

```
[port]
22071
↓
[port]
22071, 204.147.182.31
```

EspressManager は起動時に常にその IP アドレスを使用します。EspressManager が config.txt ファイルの[**port**]セクションに接続を確立するための複数のドメインを指定することができます。ドメインは[**port**]セクションの IP アドレスの後に追加されます。

```
[port]
port number, ip address, domain1, domain2, ..., domain_n
```

検索順序は、IP アドレス、domain_n、domain1 です。

[**user**]セクションでは、各行はユーザ名とパスワードで構成されています。デフォルトのユーザ名はパスワードなしの **guest** です。この行を削除して、必要な数のユーザを追加することができます。各ユーザ名とパスワードは、1 つ以上のスペースで区切られています（ユーザ名とパスワードにはスペースは使用できません）。また、ユーザ名とパスワードで大文字と小文字が区別されます。

各開発ライセンスでは、1 人の同時ユーザしかログオンできません。複数の開発ライセンスを購入した場合は、開発キットを別々のマシンにセットアップするか、1 台のマシン上で複数ユーザをセットアップすることができます。

EspressManager へのログイン権限を持つユーザの数に制限はありません。このライセンス制限は、同時ユーザ数にのみ適用されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

[**smtp host**]セクションでは、スケジューラで電子メール通知またはレポートを送信するために使用される smtp サーバ名を指定できます。スケジューラの使用の詳細については、スケジューリングを参照してください。

SMTP サーバで認証が必要な場合は、次のセクションにも記入してください。SMTP サーバが SSL/TLS を使用している場合は、[**smtp ssl**]を **true** に設定します。

[smtp secured],[smtp ssl],[smtp port],[smtp username],[smtp password]

この設定を行わない場合、これらのセクションを空白のままにすることができます。

EspressManager がはじめて起動すると、起動されたディレクトリに EspressManager.cfg という設定ファイルが生成されます。このファイルには、config.txt ファイルで定義されているように、EspressManager がリッスンしている IP アドレスとポート番号が含まれています。Report Designer が起動すると、このファイルが EspressManager への接続情報を検索します。このファイルを見つけることができない場合は、デフォルト IP の 127.0.0.1（localhost）とポート 22071 を使用します。

Report API を使用する他の EspressReport コンポーネントとアプリケーション/サーブレット/JSP も（デフォルトでは）**EspressManager.cfg** ファイルを探して EspressManager に接続します。ただし、Report Viewer、Page Viewer、Report API を使用すると、コンポーネントが EspressManager に接続するために使用する IP とポートを明示的に設定できます。EspressReport コンポーネントのデプロイメントの詳細については、デプロイメントを参照してください。

3.2.3.1 アプレットの最大メモリヒープサイズの増加

すべてのアプレットは、デフォルトで最大メモリヒープサイズが 16 メガバイトです。Windows では、コントロールパネルで Java（または Java Plugin）を選択することで、これを増やすことができます。**Java** タブをクリックし、**View** ボタンをクリックします。Runtime Parameters の下に **Xmx128M** と入力し、**OK** ボタンをクリックします。

3.2.4 EspressManager の起動

ReportDesigner を起動する前に、EspressManager が実行されている必要があります。ほとんどの場合、Web サーバ上で EspressManager を起動する必要がありますが、EspressManager と ReportDesigner の両方をローカルで実行することにより、スタンドアロンマシンで EspressReport を使用することは可能です。割り当てられた IP は 127.0.0.1 になります。これにより、ログファイル **<EspressReport InstallDir>/EspressManager.log** が作成され、リソースの使用状況の監視と問題の診断に使用できます。各 ReportDesigner ユーザマシンに EspressManager をインストールまたは起動する必要はありません。

EspressManager を起動するには、EspressReport ルートディレクトリの **EspressManager.bat** または

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

EspressManager.sh ファイルを実行します(このファイルはインストール時に自動的に生成されます)。Windows インストールの場合、インストール中に指定されたオプションに応じて、スタートメニューまたはデスクトップに追加されたプログラムショートカットを使用できます。EspressManager がデフォルトのプロパティとして自動で起動します。

また、提供されているいくつかの引数を使用して独自の設定で EspressManager を設定することもできます。引数は dash で始まり、その後にコマンドが続きます。各引数にはスペースは必要ありませんが、異なる引数を区切るには少なくとも 1 つのスペースが必要です。

これらの引数は、EspressManager の起動時にコマンドラインで入力できます。また、**espressmanager.bat/espressmanager.sh** ファイルを編集して引数を追加することもできます。

EspressManager がサーブレットプロセスとして起動されると、テキストファイルの引数も取得されます。これについての詳細は、サーブレットでの EspressManager の起動を参照してください。

-help

-help と入力すると、EspressManager は利用可能な引数とその意味に関する情報を提供します。**-h** もしくは**-?** を同じオンラインヘルプ情報を取得するために使用することもできます。

-log

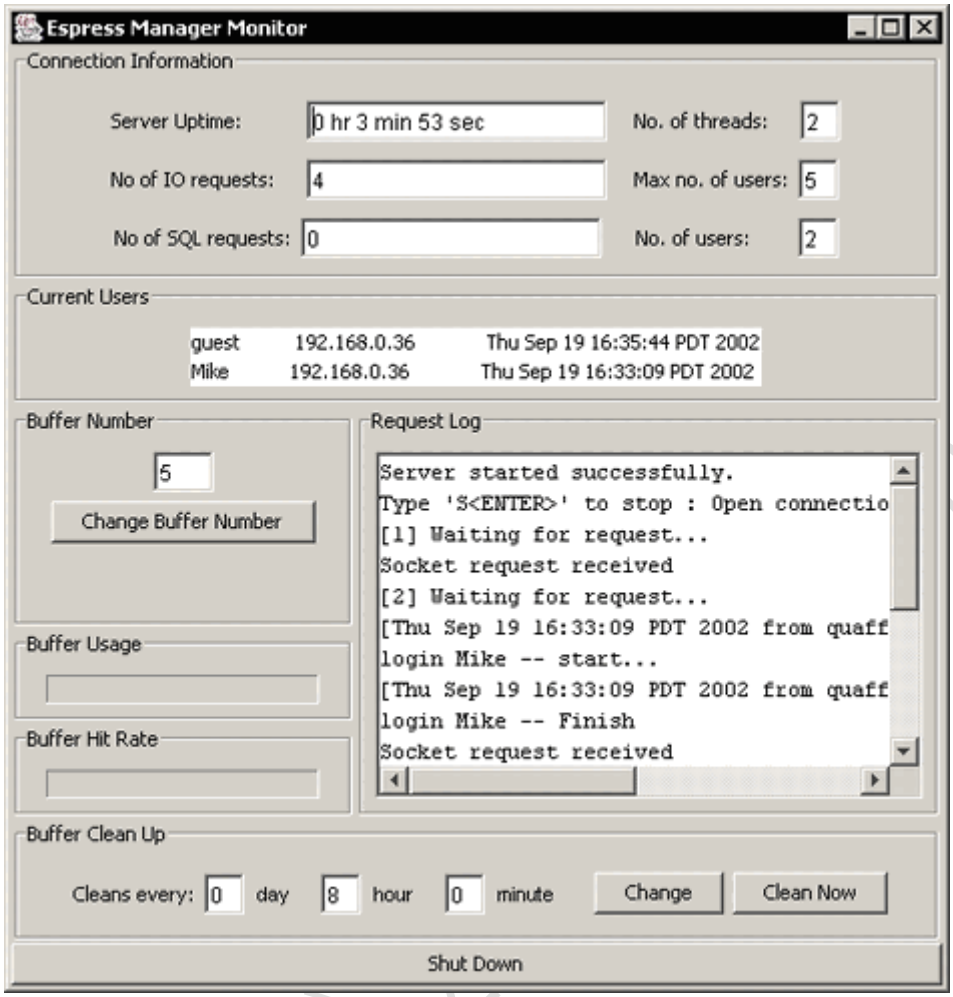
-log 引数を入力すると、ログファイルが作成され、すべてのクライアント/サーバネットワーク情報が保存されます。管理者はログファイルを開いて、各ユーザの要求を評価できます。ログは **EspressManager.log** として保存されます。

-monitor:ON/OFF

-monitor 引数を指定すると、EspressManager モニタが GUI として表示され、ステータスの表示や EspressManager の設定の変更に使用できます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



EspressManager ウィンドウ

-runInBackground:ON/OFF

この引数では、EspressManager をバックグラウンドプロセスとして実行するかどうかを指定できます。バックグラウンドプロセスとして実行すると、シャットダウンのためのユーザ入力が不要になり、スクリプトから EspressManager を実行することができます。この引数を正しく動作させるには、**-monitor：OFF** 引数と組み合わせて使用する必要があります。この方法で EspressManager を実行しているときは、プロセスを終了することだけをシャットダウンすることができます。

-recordLimit:nn

この引数を使用すると、クエリの実行時に EspressManager がデータベースから取得するレコード数の上限を設定できます。最大に達すると、EspressManager はクエリの実行を停止します。この機能により、ユーザはメモリに格納できる以上のレコードを取得できなくなり、メモリ不足のためにクラッシュすることがなくなります。

-queryTimeout:sss

この引数を使用すると、チャート照会のタイムアウト間隔を秒単位で指定できます。クエリ実行時間がタイムアウト引数を渡すと、EspressReport はクエリを中止します。この機能により、ユーザが誤って実行不能クエリを作成することを防止します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

-DBBuffer:nnn

データベースをグラフのデータソースとして使用している場合は、この引数を使用してデータベース接続とチャートに使用されるデータの両方をバッファリングすることができます。格納される接続とクエリの数は、**DBBuffer** 引数に指定された 1～999 の数によって異なります。

-DBCleanAll:ddhhmm

データソース内のデータは定期的に更新される場合があります。したがって、データバッファオプションが使用されている場合（DBBuffer の値がゼロ以外の場合）、最新のデータを取得するためにバッファをリフレッシュする必要があります。**-DBCleanAll** 引数は、データがバッファからクリアされ、データソースからフェッチされた後の期間を示すために使用できます。ddhhmm の値は、**dd** 日、**hh** 時間、および **mm** 分を意味します。EspressManager は省略されたフォーマットもサポートしています。例えば：

-DBCleanAll:101010：10 日、10 時間、および 10 分ごとにバッファを消去することを意味します。
-DBCleanAll:1010：10 時間、10 分ごとにバッファを消去することを意味します。
-DBCleanAll:10：10 分ごとにバッファを消去することを意味します。

この引数は、**-DBCleanAll** 引数が 0（つまり、常にメモリをクリーン）に設定されている場合は無効です。

EspressManager を起動すると、更新間隔が表示されます。

-RequireLogin

この引数は、ReportDesigner が API 経由で起動されたときに ReportDesigner にセキュリティを適用するために使用されます。通常、API 経由で ReportDesigner を呼び出すと、ユーザ認証はありません。そのためユーザはプログラムに独自の認証を適用できますが、許可されていないユーザがサーバにアクセスすることも許可されてしまいます。これを防ぐため、ユーザはこの引数を有効にして（値は true/false）、ReportDesigner が API から呼び出されたときに認証を強制させる事ができます。この引数をオンにすると、**QbReportDesigner** オブジェクトが表示されるたびに、API メソッド呼び出しを介して正しいユーザ名とパスワード（**config.txt** ファイルで定義）を入力する必要があります。API から ReportDesigner を呼び出す方法の詳細については、ReportAPI からの ReportDesigner の呼び出すを参照してください。

-QbDesignerPassword

この引数を使用すると、**-RequireLogin** 引数がオンのときにパスワードを設定できます。**config.txt** ファイルからのログインを使用する代わりに、**QbReportDesigner** オブジェクトを表示するときにメソッド呼び出しを介してサーバに提供する必要がある特定のパスワードを設定できます。API から ReportDesigner を呼び出す方法の詳細については、ReportAPI からの ReportDesigner の呼び出すを参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

-ScheduleCallBackClass:classfile:

この引数を使用すると、スケジューラリスナーメカニズムのクラスを指定できます。リスナーは、スケジュールが実行される前にスケジュールされたレポートへのハンドルを返します。この機能の詳細については、ICallBackScheduler インタフェースを参照してください。

-ListenerManagerClass:classfile:

この引数を使用すると、スケジューラリスナーメカニズムのクラスを指定できます。リスナーは、スケジュールが実行される前にスケジュールされたレポートへのハンドルを返します。この機能の詳細については、スケジューラリスナーを参照してください。

-PageCleanUpTime:ddhhmm:

この引数を指定すると、EspressReport の/pages/ディレクトリのクリーンアップ間隔を指定できます。レポートテンプレート（.pak/.rpt）または **QbReport** オブジェクトを使用して Page Viewer コンポーネントを呼び出すと、このディレクトリにファイルが生成されます。Page Viewer の詳細については、Page Viewer の概要を参照してください。この引数の時間引数は、**-DBCleanAll** 引数の場合と同じです。

-MaxRecordInMemory:nnn

この引数を使用すると、メモリ内で許可されるレコードの最大数を設定できます。クエリを停止するレコード制限とは異なり、この機能は、しきい値に達するとシステム内のテンポラリファイルにデータをページングし始めます。レポートの表示/エクスポートは続行されますが、データはページングファイルから読み込まれます。この機能を使用すると、レポートの実行中にシステムがメモリ不足になるのを防ぐことができます。

-MaxCharForRecordFile:nnn

この引数を使用すると、レコードファイルの格納時に生成されるページングファイルのフィールドあたりの最大文字数を設定できます。この引数よりも長いレコードのデータは、完成したレポートで切り捨てられます。

-FileRecordBufferSize:nnn

この引数を使用すると、レコードファイルの格納が呼び出されたときにページングする必要のあるレコードの数を設定できます。バッファのサイズはパフォーマンスに影響します。したがって、バッファサイズが大きいほど、レポートが高速に生成されます。

-globalFormat:xmlfile:

この引数を使用すると、グローバル形式の XML ファイルを指定して、レポート要素のデフォルトの外観を設定できます。ユーザが空のレポートを作成するたびに、このファイルの書式プロパティが適用されます。この引数には、EspressManager を基準にした XML ファイルへのファイルパスを指定する必要があります（**-globalFormat：Templates/FormatFile.xml**）。グローバル書式設定機能の詳細については、グローバル書式設定を参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

-fontMapping:xmlfile:

この引数を使用すると、フォントマッピング XML ファイルを指定して、PDF エクスポートのデフォルトフォントマッピングを設定できます。ユーザがレポートを作成するたびに、システムフォントマッピングが適用されます。この引数には、EspressManager を基準にした XML ファイルへのファイルパスを指定する必要があります。

（**-fontMapping:Templates/FontFile.xml**）フォントマッピング機能の詳細については、PDF フォントマッピングを参照してください。

-htmlDpi:nn:

この引数を使用すると、レポートを DHTML 形式または HTML 形式にエクスポートするときに使用する画面解像度をハードコードすることができます。デフォルトでは、システム解像度が使用されます。ただし、レポートが Linux または Unix サーバ上で生成され、Windows クライアントで表示される場合、これはいくつかの矛盾を引き起こす可能性があります。一般に、DPI を 96（**-htmlDpi：96**）に設定すると整合性のあるエクスポートが生成されます。

-singleTableForDistinctParamValue

-singleTableForDistinctParamValue 引数を使用すると、パラメータ化されたグラフのパラメータダイアログが異なる方法で描画されます。パラメータをデータベース列にマップすると、マップされたフィールドでのみ **select distinct** を実行することによって、別個のリストが描画されます。デフォルトの動作（この引数なし）では、元のクエリの結合と条件を使用して、別個のパラメータリストを制約します。パラメータ化されたクエリの詳細については、パラメータ化されたクエリを参照してください。

-schedulerBuffer:nnn:

この引数を使用すると、スケジューラが使用するレポートのプールを作成できます。この機能は、同じレポートを実行するスケジュールジョブが複数ある場合に役立ちます。ジョブごとに同じレポートを再ロードする代わりに、ジョブはバッファ内のレポートを使用できます。これにより、スケジュールの実行時間が短縮されます。複数の同時スケジュールジョブの場合、スケジュールバッファがスケジュールスレッド限度と同じサイズに設定されていると、最大のパフォーマンスが得られます。

-xmlEncoding:encoding:

この引数を使用すると、EspressReport が XML ファイルを記述するときに使用するエンコーディングを指定できます。これには、データレジストリファイル、XML レポートテンプレート、XML エクスポート、XML グローバルフォーマット、およびフォントマッピングファイルが含まれます。このパラメータは、EspressManager と Report Designer の両方で設定する必要があります。XML エンコーディングの詳細については、XML エンコーディングを参照してください。

-paperSize:LETTER/A4

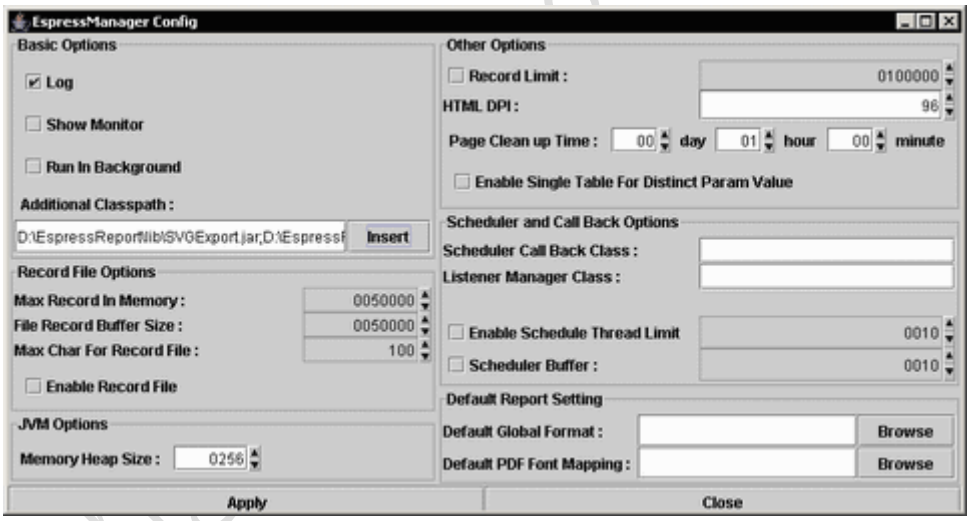
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

デフォルトの用紙サイズを定義するには、この引数を設定します。このパラメータは、**reportdesigner.bat/.sh** ファイルでも設定する必要があります。

Mac OS X で動作していて、インストール時にエイリアスを作成するように選択した場合は、EspressManager の設定を変更するために **espressmanager.app** パッケージを変更する必要があります。これを行うには、**espressmanager.app** を右クリック（Ctrl+クリック）し、ポップアップメニューから **Show Package Contents** を選択します。次に、**Info.plist** というファイルが表示される Contents フォルダに移動します。テキストエディタでこのファイルを開くと、Java プロパティの **lax.command.line.args** に引数が追加されます。

3.2.4.1 EspressManager 設定インタフェース

前のセクションで強調表示されている EspressManager 設定機能の多くは、EspressReport で提供されている EspressManager 設定インタフェースを使用して設定することもできます。インタフェースを起動するには、EspressManager が実行されていないことを確認してから、**ManagerConfig.bat/.sh** を実行します。設定ウィンドウが開きます。



EspressManager 設定ウィンドウ

以下の設定をインタフェースで設定できます：

- Log
- EspressManager Monitor on/off
- Run in Background on/off
- Record Limit
- HTML DPI
- Page Clean-up Time
- Single Table for Distinct Param Value
- Scheduler Callback Class
- Listener Manager Class
- Scheduler Thread Limit

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- Schedule Buffer
- Record File Settings
- Default Global Format
- Default PDF Font Mapping

上記のすべてのオプションの説明については、前のセクションを参照してください。設定オプションに加えて、ユーザはこのインタフェースを使用して EspressManager の CLASSPATH を追加することもできます。データベースドライバやその他の外部クラスを EspressManager に追加するには、**Insert** ボタンをクリックして目的のファイルを参照します。

Apply ボタンをクリックすると、すべての変更が **espressmanager.bat/.sh** に保存されます。

3.2.4.2 サーブレットでの EspressManager の起動

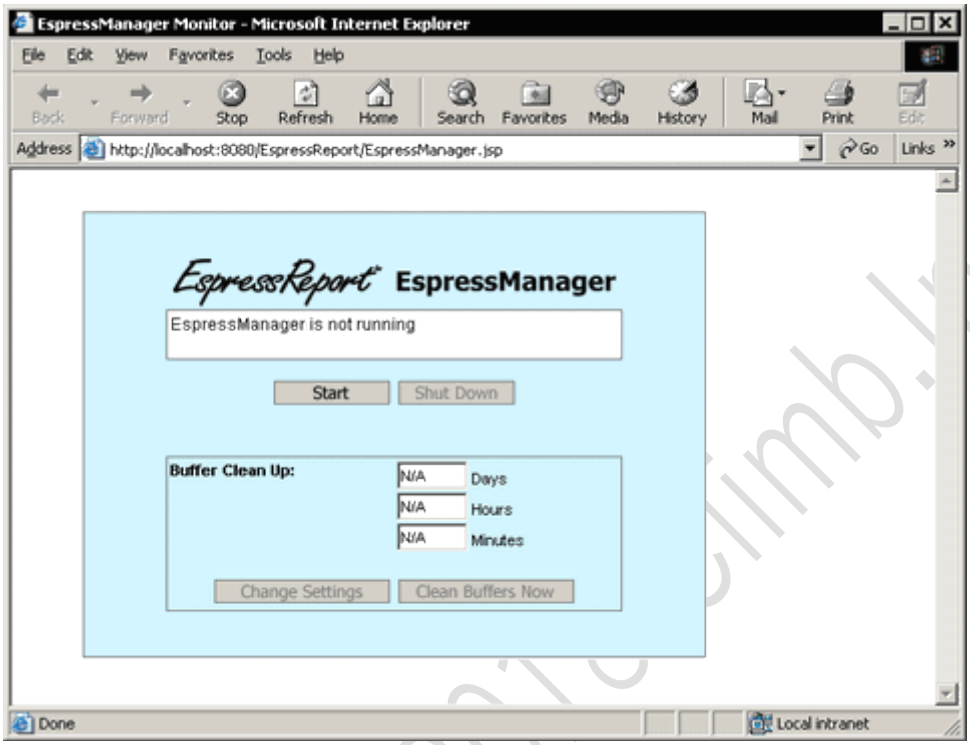
EspressManager は、アプリケーションプロセスとして実行するだけでなく、アプリケーションサーバ/サーブレットランナー内でサーブレットとして実行することもできます。この環境では、EspressManager は HTTP を使用してソケットの代わりにクライアントと通信します。この構成の利点は、EspressManager がアプリケーションサーバと同じポートを共有できるため、ファイアウォールの背後から簡単に展開できることです。さらに、この方法で EspressManager を実行すると、ユーザはリモート管理を実行できます。

EspressManager をサーブレットとしてデプロイするには、次の手順を実行します：

1. http 経由でアクセスできるように、アプリケーションサーバのルート（または仮想ディレクトリ）の下に **EspressManager.jsp**、**MenuError.jsp**、**/WebComponent**/ディレクトリをコピーします。
2. **EspressReport/classes** ディレクトリの内容をコピーし、サーブレットコンテキストを介してクラスを使用可能にします。
3. **QuadbaseDirectory.cfg** ファイルを EspressReport インストールディレクトリからアプリケーションサーバの作業ディレクトリにコピーします。作業ディレクトリを調べるには、EspressManager サーブレットクラスをコピーしたサーブレットコンテキストから **whatIsMyWorkingDirectory** を実行します。サーブレットは、呼び出されたときにブラウザに作業ディレクトリパスを出力します。
4. EspressReport インストールの **/lib/** ディレクトリにある **EspressManager.jar** と **qblicense.jar** をアプリケーションサーバの CLASSPATH に追加します。また、使用しているアプリケーションサーバに応じて、**/lib/**ディレクトリから **axercesImpl.jar** と **xml-apis.jar** を追加することもできます（たとえば、Tomcat にはすでに XML パーサがあります）。スケジューラを使用してレポートを電子メールで送信する場合は、**mail.jar** と **activiation.jar** を**/lib/**ディレクトリから CLASSPATH に追加する必要があります。最後に、EspressManager からデータベースに接続するために、JDBC ドライバのクラスを追加する必要があります。
5. アプリケーションサーバを再起動します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

これらのすべての手順が完了したら、Step1 で説明したように、Web ブラウザでアプリケーションサーバ上で使用できる **EspressManager.jsp** ページを指定して、EspressManager ページを読み込むことができます。



EspressManager JSP ウィンドウ

このページが読み込まれたら、**Start** ボタンをクリックして EspressManager を起動します。起動したら、バッファ設定を設定/変更し、このウィンドウからシャットダウンすることができます。EspressManager が起動したら、このウィンドウを閉じることができます。JSP を再度呼び出してシャットダウンするまで実行を続けます。

EspressManager をサーブレットとして実行する場合、EspressManager に接続するすべてのコンポーネントを考慮して変更する必要があります。これには、Report Designer、スケジューラ、レポートビューア、Page Viewer、および Report API を使用するすべてのアプリケーション/サーブレット/JSP が含まれます。EspressReport のデプロイメントの詳細については、デプロイメントを参照してください。

3.2.4.2.1 Tomcat 4.x/5.x でのサーブレットとしての EspressManager のデプロイ

次のセクションでは、Tomcat 4.x/5.x でサーブレットとして EspressManager を設定して実行する方法について説明します。手順は前のセクションと同じです。この例では、ポート 8080 で Tomcat をローカル（IP 127.0.0.1）で実行していることを前提としています。設定が異なる場合は、これらの手順の IP とポートを使用するポートと置き換えてください。

1. Tomcat のルートフォルダ（**<Tomcat InstallDir>/webapps/ROOT**）の下に **EspressReport** という名前のディレクトリを作成します。
   EspressReport インストール（<EspressReport InstallDir>）から、**EspressManager.jsp**、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

MenuError.jsp ファイル、および/**Web_Component**/ディレクトリを、Tomcat ルートフォルダの下の新しい/**EspressReport**/ディレクトリにコピーします。

2. <**EspressReport InstallDir**>/**classes** ディレクトリの内容を <**Tomcat InstallDir**>/**webapps**/**ROOT**/**WEB-INF**/**classes** ディレクトリにコピーします。実行者が<**Tomcat InstallDir**>/**conf**/**web.xml** ファイルでコメントアウトされていないことを確認します（Tomcat ではデフォルトでコメントアウトされていて、**/servlet/context** を利用できません）。

   Tomcat の呼び出し側サーブレットの詳細については、環境を設定するを参照してください。

3. **QuadbaseDirectory.cfg** ファイルを EspressReport ディレクトリから <**Tomcat InstallDir**>/**bin** ディレクトリにコピーします。（通常、これは Tomcat の作業ディレクトリですが、別の場所で Tomcat を起動すると作業ディレクトリが異なる場合があります）作業ディレクトリを確認するには、Web ブラウザを開いて Tomcat を実行し、**http://127.0.0.1:8080/servlet/whatIsMyWorkingDirectory** と入力します。サーブレットは作業ディレクトリへのパスを表示します）作業ディレクトリが/**bin**/以外の場合は、**QuadbaseDirectory.cfg** ファイルをそこに置きます。

4. **servlet-api.jar**（<**Tomcat InstallDir**>/**common**/**lib** の下）と同様に、**EspressManager.jar** と **qblicense.jar**（<**EspressReport InstallDir**>/**lib** の下）を Tomcat の CLASSPATH に追加します。一般に、これは<Tomcat InstallDir>/bin ディレクトリの **setclasspath.bat** または **setclasspath.sh** ファイルを編集することによって行われます。

5. Tomcat サーバを再起動します。

EspressManager を正しくデプロイする必要があります。EspressManager を起動するには、Web ブラウザを開き、次のアドレスに移動します：**http://127.0.0.1:8080/EspressReport/EspressManager.jsp** 画面がロードされ、**Start** ボタンをクリックして EspressManager を起動することができます。

3.2.4.2.2 Tomcat 8.x/9.x でのサーブレットとしての EspressManager のデプロイ

次のセクションでは、Tomcat 8.x/9.x でサーブレットとして EspressManager をセットアップして実行する方法について説明します。手順は前のセクションと同様です。この例では、ポート 8080 で Tomcat をローカル（IP 127.0.0.1）で実行していることを前提としています。設定が異なる場合は、これらの手順の IP とポートを使用するポートと置き換えてください。

1. Tomcat のルートフォルダ（<**Tomcat InstallDir**>/**webapps**/**ROOT**）の下に **EspressReport** という名前のディレクトリを作成します。EspressReport インストール（<**EspressReport InstallDir**>）から、**EspressManager.jsp**、**MenuError.jsp** ファイル、**/Web_Component/** ディレクトリを、Tomcat ルートフォルダの下の新しい/**EspressReport**/ディレクトリにコピーします。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2. **<EspressReport InstallDir>/classes** ディレクトリの内容を **<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリにコピーします。

3. **servlet-api.jar**（**<Tomcat InstallDir>/lib** の下）と同様に、**EspressManager.jar** と **qblicense.jar**（**<EspressReport InstallDir>/lib** の下）を Tomcat の CLASSPATH に追加します。一般的には、**<Tomcat InstallDir>/bin** ディレクトリに **setenv.bat** を作成することで行います。Tomcat の CLASSPATH に、**EspressManager.jar**、**qblicense.jar**、および **servlet-api.jar** の値を書き込みます。

   setenv.bat の例：
   CLASSPATH=C:¥EspressReport70¥lib¥EspressManager.jar;C:¥EspressReport70¥lib¥qblicense.jar;C:¥tomcat9¥lib¥servlet-api.jar

4. **<Tomcat InstallDir>/bin** ディレクトリに **html** という空のフォルダを 1 つ作成します。

5. **<Tomcat InstallDir>/bin** ディレクトリに **userdb** という名前のフォルダを 1 つ作成します。**<EspressReportInstallDir>/userdb** ディレクトリにある **config.txt** を新しく作成したフォルダにコピーします。

6. Tomcat サーバを再起動します。

今度は EspressManager を正しく配備する必要があります。EspressManager を起動するには、Web ブラウザを開き、次のアドレスに移動します：
http://127.0.0.1:8080/EspressReport/EspressManager.jsp
画面がロードされ、**Start** ボタンをクリックして EspressManager を起動することができます。

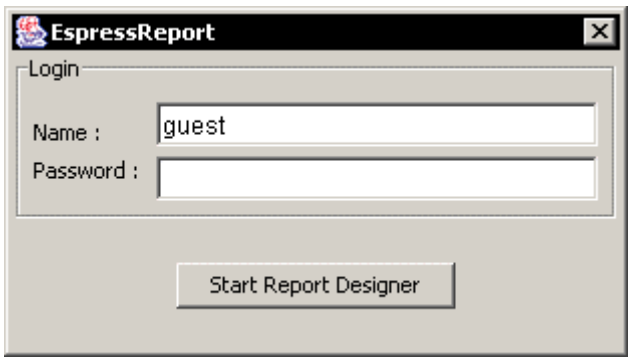
3.2.5 Report Designer の起動

スタンドアロン

Report Designer をローカルで実行している場合は、まず **espressmanager.bat** または **.sh** ファイルを実行して EspressManager を起動し、**reportdesigner.bat** または **.sh** ファイルを実行します（Windows または Mac のショートカット/エイリアスも実行できます）。EspressManager にログオンするように促すダイアログボックスが表示されます。**config.txt** ファイルでユーザを変更していない場合は、パスワードなしでユーザ名として **guest** を入力し、**Start Report Designer** ボタンをクリックします。これでアプリケーションが起動します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Report Designer ログインウィンドウ

ブラウザ

Report Designer をリモートで実行している場合は、EspressManager がリモートマシン上で実行されていることを確認し、ブラウザに EspressReport の URL（**http//machinename/espressreport/index.html**）を入力します。このページには、EspressManager にログオンするためのダイアログが表示されます。**config.txt** ファイルでユーザを変更していない場合は、パスワードなしでユーザ名として **guest** を入力し、**Start Report Designer** ボタンをクリックします。これにより、新しいウィンドウでアプリケーションが起動します。

EspressReport は JDK 1.2 以上を必要とするため、ブラウザ経由で Report Designer を実行するには、Java プラグインをダウンロードする必要があります。また、index.html ページのアプレットは、Windows クライアント上で実行するように設定されています。別のプラットフォームでアプレットを実行している場合は、ページの HTML ソースを変更する必要があります。HTML ソースには 2 つのアプレットがありますが、そのうちの 1 つはコメントアウトされています。Windows 以外のクライアントで実行するには、最初のアプレットをコメントアウトし、2 番目のアプレットをコメント解除します。

3.2.5.1 サーブレットとして実行する EspressManager への接続

サーブレットでの EspressManager の起動で説明されているように、EspressManager がサーブレットとして実行されている場合、ReportDesigner を正常に起動するには、いくつかの変更を加える必要があります。

スタンドアロン

.bat または.sh ファイルを使用して ReportDesigner を実行している場合は、EspressReport がサーブレットとして実行されていることを示すためにファイルを変更し、EspressManager サーブレットの場所をポイントする必要があります。これを行うには、次の引数を.bat ファイルまたは.sh ファイルに追加します。**-servlet:http://IP Address:Port/Context**
例：**-servlet:http://127.0.0.1:8080/servlet**

EspressManager が/**servlet/context** のポート 8080 でローカルホスト上で実行されていることを示します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ブラウザ

サーブレットとして実行されている EspressManager に接続している Java Web Start アプリケーションを使用して ReportDesigner を実行している場合は、次のものが必要です。

/lib/directory、**index.html**、**ReportDesigner.jsp** ファイル
（これらの 3 つはすべて**<EspressReport InstallDir>**にあります）を**<Tomcat InstallDir>/webapps/ROOT/EspressReport** ディレクトリにコピーします。

ReportDesigner.jsp で **jnlp** の内容を変更します。
要素 **jnlp** を **href** = **""**に設定します。

例：<jnlp spec="1.0+" codebase="<%=codebase%>" href="">

jnlp コンテンツの **applet-desc** の要素内に、次のパラメータタグを追加する必要があります。

```
<PARAM NAME="comm_protocol" VALUE="servlet">
<PARAM NAME="comm_url" VALUE="<%=codebase%>">
<PARAM NAME="servlet_context" VALUE="servlet">
```

最初のパラメータは、EspressManager がサーブレットとして実行されていることを示します。2 つ目は、アプリケーションサーバが使用している URL（JSP 式**<%codebase%>**に保存されている URL）で、3 つ目は EspressManager がデプロイされているサーブレットのコンテキストです。

/browserimages/ ディレクトリ（**<EspressReport InstallDir>** の下）を **<Tomcat InstallDir>/bin** ディレクトリにコピーします。

<web-app> と **</web-app>** の間に次のサーブレット定義を追加して、**web.xml** の ReportDesigner（**<Tomcat InstallDir>/webapps/ROOT/WEB-INF** ディレクトリ下）で使用するサーブレットを定義します。

```
<display-name>EspressReport</display-name>
  <description>EspressReport</description>

  <servlet>
                <servlet-name>whatIsMyWorkingDirectory</servlet-name>
                <servlet-class>whatIsMyWorkingDirectory</servlet-class>
                <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet>
                <servlet-name>ESMBaseServlet</servlet-name>
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        <servlet-class>ESMBaseServlet</servlet-class>
</servlet>

<servlet>
        <servlet-name>ESMMessageServlet</servlet-name>
        <servlet-class>ESMMessageServlet</servlet-class>
</servlet>

<servlet>
        <servlet-name>ESOSwitchServlet</servlet-name>
        <servlet-class>ESOSwitchServlet</servlet-class>
</servlet>

<servlet>
        <servlet-name>EspressManagerServlet</servlet-name>
        <servlet-class>quadbase.reportutil.EspressManager</servlet-class>
</servlet>


<servlet-mapping>
        <servlet-name>whatIsMyWorkingDirectory</servlet-name>
        <url-pattern>/servlet/whatIsMyWorkingDirectory</url-pattern>
</servlet-mapping>

<servlet-mapping>
        <servlet-name>ESMBaseServlet</servlet-name>
        <url-pattern>/servlet/ESMBaseServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
        <servlet-name>ESMMessageServlet</servlet-name>
        <url-pattern>/servlet/ESMMessageServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
        <servlet-name>ESOSwitchServlet</servlet-name>
        <url-pattern>/servlet/ESOSwitchServlet</url-pattern>
</servlet-mapping>

<servlet-mapping>
        <servlet-name>EspressManagerServlet</servlet-name>
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                <url-pattern>/servlet/EspressManagerServlet</url-pattern>
</servlet-mapping>
```

Tomcat サーバを再起動します。

ReportDesigner にアクセスするには、Web ブラウザを開き、次のアドレスに移動します：
http://127.0.0.1:8080/EspressReport/index.html

EspressReport.jnlp（Firefox）を開くか、ダウンロードディレクトリに保存してそこから実行して ReportDesigner（Chrome/Microsoft Edge）を開きます。

3.2.6 下位互換性パッチ

古いバージョンからアップグレードした場合、デフォルトの動作にいくつかの変更があることがあります。下位互換性は可能な限り保持されますが、新しい動作が適用される場合があります。新しい動作推奨がされますが、以前のバージョンのチャートやレポートが既にある場合は、古い動作を維持して、チャートとレポートがまったく同じに見えるようにすることができる下位互換性のパッチを提供しています。

パッチは上級ユーザ向けです。パッチを必要としている場合、**操作内容を理解している場合**にのみ、それらを適用してください。ご不明な点がある場合は、サポートに連絡してください。

パッチは**<EspressReportInstallDir>/lib/Patches** ディレクトリにあります。それらは JAR アーカイブに格納されます。パッチを適用するには、アプリケーションのクラスパスに適切な JAR ファイルを追加する必要があります。

すべてのデザイナ（レポートとグラフ用）とビューアにパッチを適用する場合は、EspressReport インストールディレクトリに **espressmanager.bat** ファイルと **reportdesigner.bat** ファイル（Windows を使用している場合）、**espressmanager.sh** ファイルと **reportdesigner.sh** ファイル（他の OS を使用している場合）を編集し、パッチ JAR ファイルへの相対パス（例：**./lib/Patches/patch1.jar**）を classpath パラメータに追加する必要があります。HTML ページから Report Designer を実行する場合は、EspressReport インストールディレクトリの **index.html** ファイルを編集し、パッチ JAR ファイルの相対パスをアプレットタグのアーカイブ属性に追加する必要があります。

API を使用している場合は、アプリケーションのクラスパスにパッチ JAR ファイルを含める必要があります。

以下は現在のバージョンで使用可能なすべてのパッチのリストです。

patch1.jar…グラフの軸のパディングをデフォルトで無効にします

- デフォルトの動作（パッチなし）…軸のパディングはデフォルトでオンです

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- パッチ適用後…軸のパディングはデフォルトではオフです
- 新しい動作はバージョン 4.0 で導入されました
- この機能は、API の **IAxis.setAxisPaddingAdded** メソッドまたは Chart Designer の Axis Scale ダイアログを使用して設定することもできます

patch2.jar…注釈テキストの左余白をチャートに追加します

- デフォルトの動作（パッチなし）…注釈テキストに余白が残りません
- パッチ適用後…注釈テキストに余白が残ります
- 新しい動作はバージョン 5.0 で導入されました
- この機能はパブリック API または UI で設定することはできません
- 注釈テキストは、凡例テキスト、チャートタイトル、および Chart Designer の **Insert** > **Text** を使用して挿入されたテキストです

patch3.jar…軸 pt のチャート軸オートスケールが 1 未満の場合、最小値/最大値として 0～1 を使用します

- デフォルトの動作（パッチなし）…自動スケールを使用すると、データに応じて最大値と最小値が常に設定されます
- パッチ適用後…max-min が 1 未満で autoscale が使用されている場合、min は 0 に設定され、max は 1 に設定されます
- 新しい動作はバージョン 5.4 で導入されました
- この機能はパブリック API または UI で設定することはできません
- **このパッチは使用しないことをお勧めします**

patch4.jar…新しい円グラフのラベル配置アルゴリズムをオフにします（円グラフの位置に基づいてラベルの配置を計算する）

- デフォルトの動作（パッチなし）…新しい円のラベル配置アルゴリズムが使用されます
- パッチ適用後…古いパイのラベル配置アルゴリズムが使用されます
- 新しい動作はバージョン 6.0 で導入されました
- この機能はパブリック API または UI で設定することはできません

patch5.jar…常にチャート軸の自動スケールに整数値を使用します

- デフォルトの動作（パッチなし）…整数は常に軸自動スケールに使用されます
- パッチ適用後…軸の自動スケールには軸の値のデータ型が使用されます
- 新しい動作はバージョン 6.0 で導入されました
- この機能はパブリック API または UI で設定することはできません

patch6.jar…グラフ軸スケールの最小および最大エラーチェックを無効にします

- デフォルトの動作（パッチなし）…エラーチェックが有効になります
- パッチ適用後…エラーチェックは無効になります（データセットに設定された最大値よりも低い最大値を設定できます）
- 新しい動作はバージョン 6.2 で導入されました

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- このパッチは API 専用です
- この機能はパブリック API または UI で設定することはできません

patch7jar…デフォルトでは、円柱チャートと棒グラフのカテゴリ機能のために単一色をオフにします

- デフォルトの動作（パッチなし）…カテゴリ機能の単一色は、デフォルトでオンになっています
- パッチ適用後…カテゴリ機能の単一色は、デフォルトでオフになっています
- 新しい動作はバージョン 6.3 で導入されました
- このパッチは API 専用です
- この機能はパブリック API または UI で設定することはできません
- この機能は、API の **IDataPointSet.setSingleColorForCategories** メソッドまたは Chart Designer の Chart Options ダイアログで設定することもできます

patch8jar…線グラフの端から端まで単一点データを左軸に表示します

patch9jar…スタックに十分なスペースがなくてもスタックラベルを表示します

Copyright(C) 2018

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.3 Data Source の動作

レポートを設計する最初のステップは、使用するデータを取得することです。Report Designer では、JDBC 準拠のデータベース、テキストファイル、XML ファイル、EJB からデータを取得したり、クラスファイルを使用してオブジェクト/配列データを取り込んだりすることもできます。すべてのデータソース情報は、データソースマネージャに格納されます。

3.3.1 Data Source Manager

データソースマネージャは、特定のユーザが使用するデータソースの場所と接続情報を格納する統合ユーティリティです。情報は XML レジストリファイルに保存されます。多くの異なるデータレジストリファイルを設定することができ、1 つのレジストリに格納できるデータソースの数に制限はありません。レジストリは設計時に補助として使用され、データまたはデータソース情報がレポートファイルに保存されるため、レポートを展開する必要はありません。

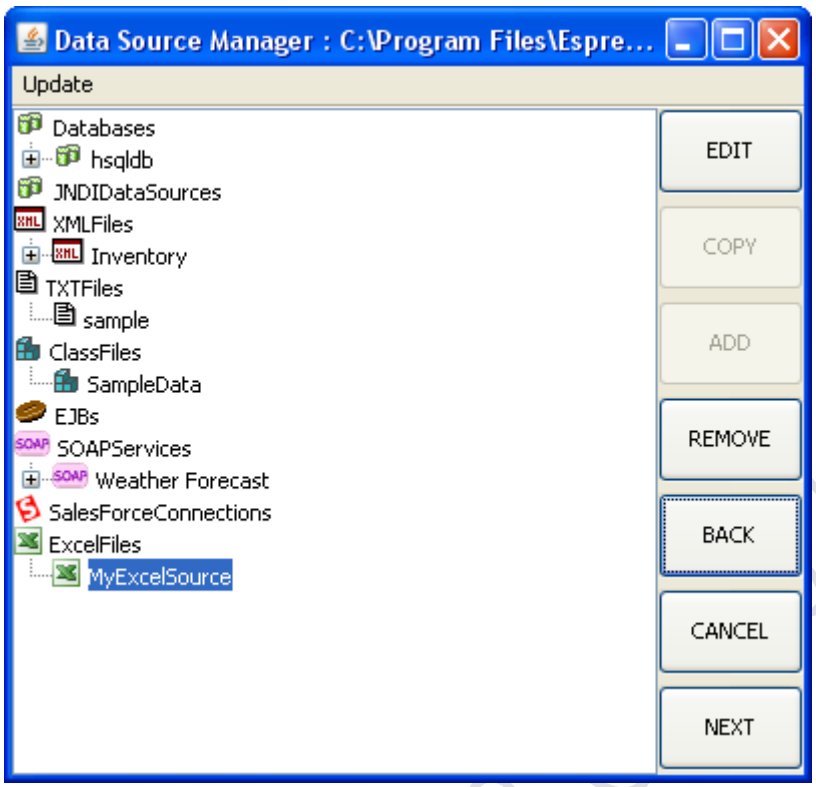
XML データソースリポジトリは、実際のデータではなく、場所と接続情報のみを格納します。これは、レポートで使用されるデータを含む XML ファイルではありません。

3.3.1.1 Data Source Manager の使用

新しいレポートを開始する（**File** メニューの **New** を選択するか、ツールバーの **New** ボタンをクリックして)、または最初に Report Designer を起動すると、レポートウィザードが起動します。ウィザードは、レポートの基本的な構成をガイドするステップバイステップのプロセスです。レポートウィザードの最初のダイアログで、使用するデータレジストリファイルを指定するか、新しいレジストリを作成するかどうかを確認するメッセージが表示されます。デフォルトでは、データソースレジストリファイルは DataRegistry ディレクトリに保存されます。既存のレジストリを開くか、新しいレジストリを起動すると、Data Source Manager ウィンドウが開きます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Data Source Manager

ウィンドウの左側には、レジストリファイルのすべてのデータソースを一覧表示するツリーがあります。グループ化された **Databases** には、個々のデータベースと関連するクエリとデータビューがあります。JDBC の代わりに JNDI（Java Naming and Directory Interface）名を使用して接続するデータベースソースは、**JNDIDataSources** の下にグループ化されています。**XMLFiles** の下にグループ化されているのは、すべての XML ファイルとその関連クエリです。すべての指定されたテキストファイルは、**TXTFiles** でグループ化されます。**ClassFiles** の下にグループ化されているのは、すべて指定されたクラスファイルです。**EJB** の下にグループ化されたすべての EJB 接続が指定され、**SOAP** の下にグループ化されているすべての SOAP データソースが指定されます。

ウィンドウの右側には、データソースマネージャのすべての機能を制御する一連のボタンがあります。各ボタンは次の機能を実行します。

Edit：

このオプションを使用すると、データソースの属性を変更できます。データベースの場合、接続情報を変更してクエリ/データビューを変更することができます。XML ファイルの場合は、ファイルとその場所を変更したり、XML クエリを変更することができます。テキストファイルの場合は、表示名とファイルの場所を変更できます。クラスファイルの場合は、表示名を変更して場所を変更することができます。また、EJB の場合は、表示名とパラメータ値を変更することができます。

Copy：

このオプションは、データベースノードおよびビッグデータノードだけでなく、指定したクエ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

リまたはデータビューに対しても使用できます。

Add：

このオプションを使用すると、データソースを追加できます。ウィンドウの左側でどのノードが選択されているかによって、新しいソースが作成されます。したがって、**TXTFiles** を選択して **Add** ボタンをクリックすると、新しいテキストファイルのデータソースを追加するよう求められます。

Remove：

このオプションは、選択したデータソースを削除します。

Back：

このオプションを使用すると、ウィザードのひとつ前に戻り、使用しているレジストリファイルを変更できます。

Cancel：

ウィザードの処理をキャンセルします。

Next：

チャートで現在選択されているデータソースが使用され、ウィザードの次のステップに進みます。

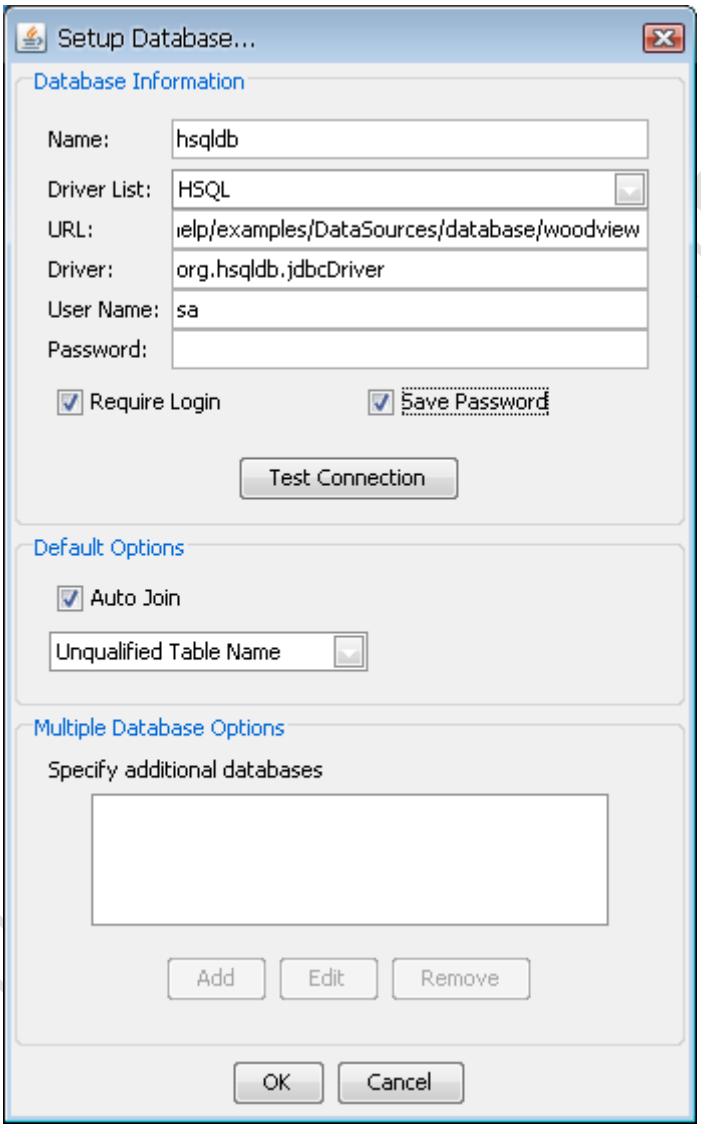
3.3.2 Database からのデータ

EspressReport は、JDBC 準拠のデータベースからデータを取得できます。サードパーティのドライバ（JDBC ブリッジ以外）を介してデータベースに接続するには、ドライバのクラスを取得するように EspressManager を変更する必要があります。設定インタフェース（EspressManager 設定インタフェースを参照）を使用するか、手動で **EspressManager.bat/.sh** ファイルを編集してクラスを追加できます。このファイルで、**-classpath** 引数に適切なクラスまたはアーカイブを追加します。Mac OS X で実行していてインストール中にエイリアスを作成することを選択した場合は、**espressmanager.app** パッケージを変更して、JDBC ドライバをクラスパスに追加する必要があります。これを行うには、**espressmanager.app** を右クリック（CTRL+クリック）し、ポップアップメニューから **Show Package Contents** を選択します。Contents フォルダに移動し、**Info.plist** というファイルを表示します。このファイルを開き、クラスパス引数に適切なクラスまたはアーカイブを追加します。MS SQL Server、MySQL、Oracle、Informix、PostgreSQL データベース用の JDBC ドライバは便利です。他のデータベースの JDBC jar ファイルは、ライセンス、複数のドライバ、および/またはその他の問題のために含まれていませんが、これらのデータベースのサポートが存在し、jar ファイルを明示的に追加できます。

データベースをデータソースとして使用するための最初の手順は、レジストリにデータベースを設定し、接続情報を指定することです。データベースを追加するには、**Databases** ノードをクリックし、**Add**

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ボタンをクリックします。これにより、そのデータベースの接続情報を指定するためのウィンドウが表示されます。ドライバリストから接続するデータベースを選択するか、手動で情報を指定することができます。入力するフィールドは、データベース名、URL、およびドライバです。データベースにログインが必要かどうか、またユーザ名とパスワードの情報を保存するかどうかを選択することもできます。ログイン情報とパスワード情報を保存することを選択した場合は、これらの情報を **User Name** および **Password** テキストボックスに入力できます。**OK** ボタンをクリックすると、新しいデータベースが Data Source Manager ウィンドウに追加されます。



Data Source Manager ウィンドウ

EspressReport がデータベースへの接続を作成するには、次の情報を入力する必要があります。

URL：

この JDBC URL は、使用するデータベースの場所を指定します。標準の JDBC URL には、3 つの部分があり、コロンで区切られています。

jdbc:<subprotocol>:<subname>

JDBC URL の 3 つの部分は、次のように分類されます：

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

1. jdbc…プロトコル。JDBC URL のプロトコルは常に jdbc です。
2. <**subprotocol**>…ドライバの名前または 1 つまたは複数のドライバによってサポートされるデータベース接続メカニズムの名前。subprotocol 名の顕著な例は、ODBC データソース名を指定する URL 用に予約されている odbc です。たとえば、JDBC-ODBC ブリッジを介してデータベースにアクセスするには、次の URL を使用する必要があります：

jdbc:odbc:Northwind

この例では、subprotocol は **odbc** であり、サブネーム **Northwind** はローカル ODBC データソースです。つまり、**Northwind** は ODBC でシステム DSN として指定されています。

3. <**subname**>…データベースの識別方法。subname は、subprotocol によって異なる場合があり、ドライバライターが選択した内部構文を含むサブサブネット名を持つことができます。subname の機能は、データベースを見つけるのに十分な情報を与えることです。前の例では、ODBC は残りの情報を提供するため、**Northwind** で十分です。

リモートマシン上のデータベースには、接続するための追加情報が必要です。たとえば、会社のイントラネット上でデータベースにアクセスする場合、ネットワークアドレスはサブネームの一部として JDBC URL に含める必要があります。また、標準の URL 命名規則に従う必要があります。

//hostname:port/subsubname

企業イントラネット上のマシンに接続するために VPN というプロトコルを使用すると仮定すると、使用する必要がある JDBC URL は次のようになります：

jdbc:vpn://dbserver:791/sales(jdbc:dbvendorname://machineName/SchemaName に似ています)

JDBC はデータベース自体ではなく、データベースドライバに接続することを覚えておくことが重要です。特定のドライバを識別する JDBC URL は、データベースドライバベンダによって決定されます。通常、データベースベンダーは適切なドライバを提供します。データベースドライバに接続するために必要な正しい JDBC URL については、データベースドライバベンダに問い合わせることを強くお勧めします。

Driver：

これは、データベースに接続するのに必要な適切な JDBC ドライバです。インストール（また

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

は Oracle の J2SE）に含まれている JVM を使用している場合は、次のドライバ仕様を使用して ODBC データソースに接続します。

sun.jdbc.odbc.JdbcOdbcDriver

JDBC-ODBC ブリッジを使用していない場合は、データベース固有の JDBC ドライバ名を指定することもできます。例として、Oracle データベースエンジンには **oracle.jdbc.driver.OracleDriver** が必要です。

User Name：

データベースへのログイン時に使用されます。

Password：

ユーザごとに設定するパスワードです。

接続情報を指定したら、**Test Connection** ボタンをクリックしてデータベース接続をテストできます。これにより、提供した情報を使用して接続がテストされ、問題が報告されます。

ダイアログのデフォルトオプション部分では、クエリビルダのインタフェースまたはデータビューで生成されたクエリのいくつかのプロパティを指定できます。選択したテーブルを自動結合するかどうかを指定できます。自動結合は、データベースに定義されている主キーと外部キーを結合しようとします。照会に使用する表名の形式を、修飾されていない（表名のみ）、または 2 部または 3 部修飾されたものとして指定することができます。ここで指定されたプロパティは、新しいクエリおよびデータビューのデフォルト設定になります。個々のクエリに対しても変更できます。

ダイアログの **Multiple Database Options** 部分では、クエリ内からデータを取得するために追加のデータベース（追加のデータベース URL）を指定できます。このオプションは、データベース（オリジナルおよび追加のデータベース）が MS SQL Server で、3-Part Qualified Table Name オプションが選択されている場合にのみ使用できます。指定された追加データベースへの接続にも同じログイン詳細と同じドライバ（元の接続で定義されているもの）を使用することに注意してください。このクエリは、3 つの表の命名法を使用して、追加のデータベースの列を参照することによってデータを取得できます。

EspressReport のインストールには 2 つのサンプルデータベースが含まれます。1 つは HSQL（pure Java アプリケーションデータベース）データベースで、もう 1 つは MS Access データベースです。両方とも同じデータを含み、**help/examples/DataSources/database** ディレクトリにあります。これらのサンプルデータベースへの接続を設定する方法の詳細は、データベース接続の設定を参照してください。

3.3.2.1 JNDI Data Sources

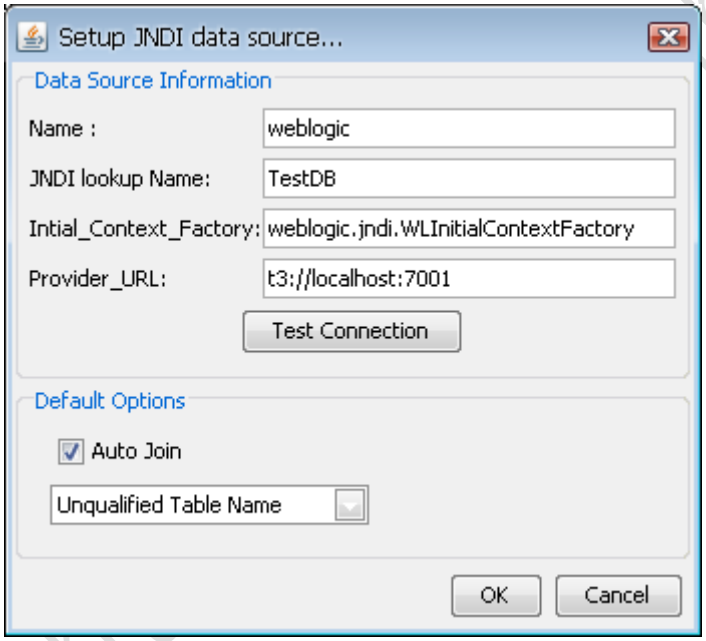
EspressReport では JDBC 経由でデータベースに接続するだけでなく、JNDI（Java Naming and Directory Interface）を使用してデータソースに接続することができます。EspressReport では、JNDI データソースはデータベースデータソースと同様に扱われ、同じ機能（クエリ、パラメータ、データビ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ューなど）をサポートします。JNDI データソースを使用する利点は、環境間でレポートを簡単に移行できるようにすることです。両方の環境のデータソースが同じルックアップ名で設定されている場合、レポートはそのまま移行できます。

EspressReport の JNDI データソースに接続するには、Web アプリケーション環境にデータソースをデプロイし、同じ環境内でサーブレットとして実行する EspressManager を持っている必要があります。サーブレットとしての EspressManager の実行の詳細については、サーブレットでの EspressManager の起動を参照してください。

JNDI データソースを設定するには、データソースマネージャの **JNDIDataSources** ノードを選択し、**Add** ボタンをクリックします。これにより、接続情報を指定できるダイアログが表示されます。



JNDI Setup ダイアログ

最初のオプションでは、データソースの表示名を指定できます。2 番目のオプションでは、データソースの JNDI ルックアップ名を指定できます。3 番目のオプションでは、データソースの初期コンテキストファクトリを指定できます。最後のオプションでは、プロバイダ URL を指定できます。この情報は、異なるベンダーが JNDI データソースを異なる方法で実装するため、使用しているアプリケーションサーバによって異なります。**Test Connection** ボタンをクリックすると、接続をテストできます。

3.3.2.2 クエリ

データベースを追加すると、データベースの新しいノードが Data Source Manager ウィンドウに表示されます。ノードを展開すると、さらに 2 つのノードが表示されます。1 つはクエリで、もう 1 つはデータビューです。これらは、データベースからデータを取得する 2 つの方法です。新しいクエリを作成するには、**Queries** ノードを選択して **Add** ボタンをクリックします。クエリ名を指定し、SQL 文を入力するかクエリビルダを起動するかを選択するダイアログが表示されます。

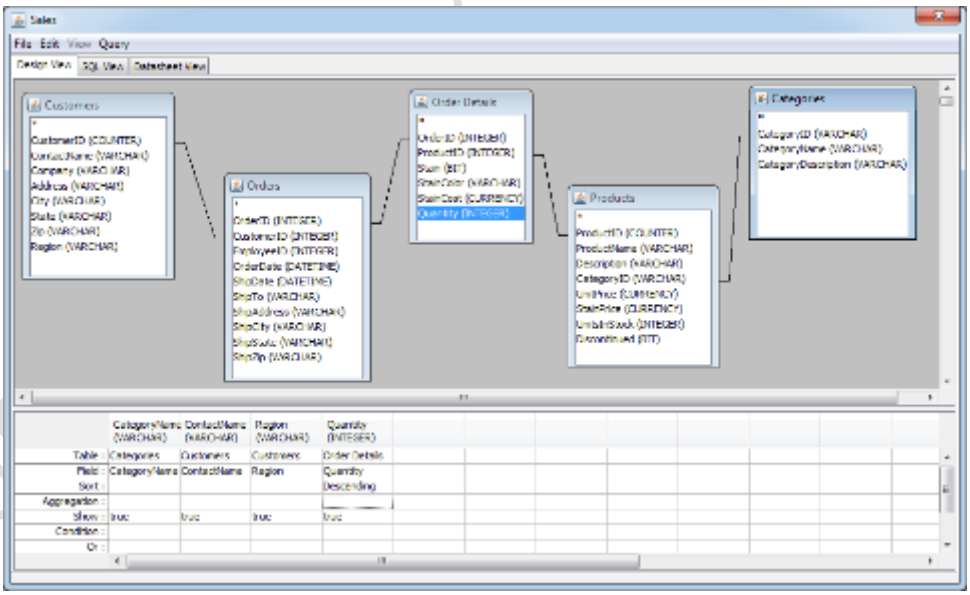
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

SQL 文の入力を選択すると、SQL 文を入力するためのダイアログボックスが表示されます。このダイアログでは、SQL テキストを含む QRY またはテキストファイルをロードしたり、ストアドプロシージャを実行したりすることもできます。クエリビルダの起動を選択すると、クエリビルダが新しいウィンドウで開き、クエリを視覚的に構築することができます。クエリの作成または入力が完了すると、Data Source Manager ウィンドウに戻り、クエリはデータベースのクエリノードの下に新しいエントリとして表示されます。

3.3.2.2.1 クエリビルダの使用

クエリビルダは、ビジュアル環境でリレーショナルデータベースに対してクエリを作成するための統合ユーティリティです。クエリビルダを起動するには、データソースマネージャ内に新しいクエリを追加し、**Open Query Builder** オプションを選択します。クエリビルダが新しいウィンドウで開きます。また、Data Source Manager のクエリ名をダブルクリックして、既存のクエリを変更するためにクエリビルダを起動することもできます。

メインのクエリビルダウィンドウは 2 つの部分で構成されています。ウィンドウの上半分には、クエリおよび関連する列に対して選択されたすべてのデータベーステーブルが含まれています。上のウィンドウには、列フィールド間にどのような結合が設定されているか示されます。メインウィンドウまたは QBE（例でクエリ）ウィンドウの下半分には、クエリまたは関連する条件に対して選択または作成された列が含まれています。



クエリビルダウィンドウ

クエリビルダウィンドウの上部には 3 つのタブがあります。これにより、さまざまなビューを切り替えることができます。**Design View** タブは、前述のメインデザイナーウィンドウです。**SQL View** タブには現在のクエリで生成された SQL 文が表示され、Datasheet View タブにはクエリ結果が表示されます。

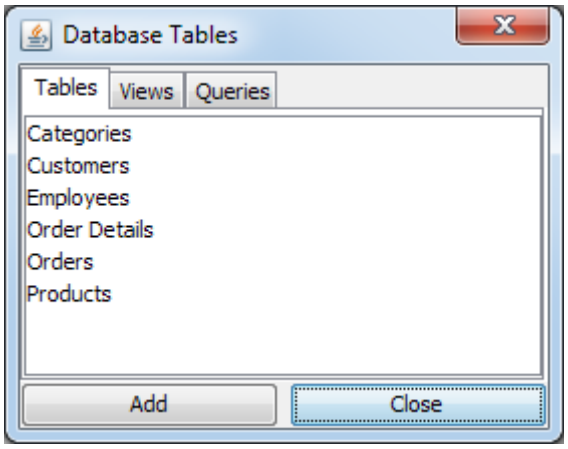
クエリの作成が完了したら、**File** メニューの **Done** を選択してデータソースマネージャに戻ります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Tables

クエリビルダが初めて起動すると、データベース内のすべてのテーブルのリストを含むタブ付きウィンドウが表示されます。2 番目のタブにはデータベース内のすべてのビューのリストが含まれ、3 番目のタブにはクエリという見出しの下にデータベース用に設計した他のクエリのリストが含まれています。このウィンドウから、クエリを作成するテーブル/ビュー/クエリを選択できます。以前に設計されたクエリをテーブルとしてロードすることもできます。テーブルを追加するには、テーブルを選択して **Add** ボタンをクリックするか、テーブル名をダブルクリックします。テーブルが追加されると、それはメインのクエリビルダウィンドウに表示され、そのテーブル内のすべてのカラムが表示されます。テーブルを削除するには、テーブル内を右クリックし、ポップアップメニューから **Delete** を選択します。テーブル領域を指定し、このメニューからアルファベット順にフィールドをソートすることもできます。**Close** ボタンをクリックすると、テーブルウィンドウを閉じることができます。再度開くには、**Query** メニューから **Show Tables** を選択します。

デフォルトでは、データベース接続を設定するときに指定した名前の形式を使用してテーブルが表示されます。**Query** メニューから **Name Format** を選択すると、名前を変更できます。



クエリビルダテーブルウィンドウ

Joins

クエリのデータベーステーブルを選択すると、クエリビルダはデータベースの主キーと外部キーの関係に基づいて列フィールド間の結合を自動検出できます。データベース接続を設定するときに選択したオプションに応じて、自動結合が追加され、表間の標準結合を作成します。結合は、デザインウィンドウの上半分の 2 つのフィールド間に描かれた線で表されます。結合を削除するか、そのプロパティを編集するには、その行を右クリックし、ポップアップメニューから選択します。結合を追加するには、1 つの列フィールドをクリックして別のテーブルの別の列フィールドにドラッグします。その後、参加が表示されます。自動結合設定を変更するには、**Queries** から **Auto Join** を選択します。

プロパティを結合する

ポップアップメニューから **Join Properties** を選択すると、列フィールド間で使用される結合のタイプを選択できる 3 つのオプションが表示されます。クエリビルダは、等結合だけをサポートしています。不等号結合は、**conditions** フィールドを使用して簡単に実行できます。内部結合、左外結合、および右外結合を指定できます。異なる結合タイプの説明については、以下の例を参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次の 2 つのテーブルがあるとします。上が Customer（顧客）テーブル、下が Orders（注文）テーブルです。

CustomerID	CustomerName
1	Bob
2	Ivan
3	Sarah
4	Randy
5	Jennifer

OrderID	CustomerID	Sales
1	4	$2,224
2	3	$1,224
3	4	$3,115
4	2	$1,221

2 つのテーブルの **CustomerID** の内部結合によって、**Customers** テーブルの各行が、一致する **CustomerID** 値を持つ **Orders** テーブルのすべての行と **Join** されるように **Customers** テーブルと **Orders** テーブルの行が結合されます。**Orders** テーブルの **CustomerID** フィールドが一致しない **Customers** テーブルの行は、クエリ結果セットに含まれません。

CustomerID フィールドの内部結合で **OrderID**、**CustomerName**、および **Sales** フィールドを選択して、クエリを作成するとします。クエリビルダによって生成される SELECT 文は、以下のようになります。

```
Select Orders.OrderID, Customers.CustomerName, Orders.Sales
From Customers, Orders
Where Customers.CustomerID = Orders.CustomerID
Order by Orders.OrderID;
```

クエリの結果を以下に示します。

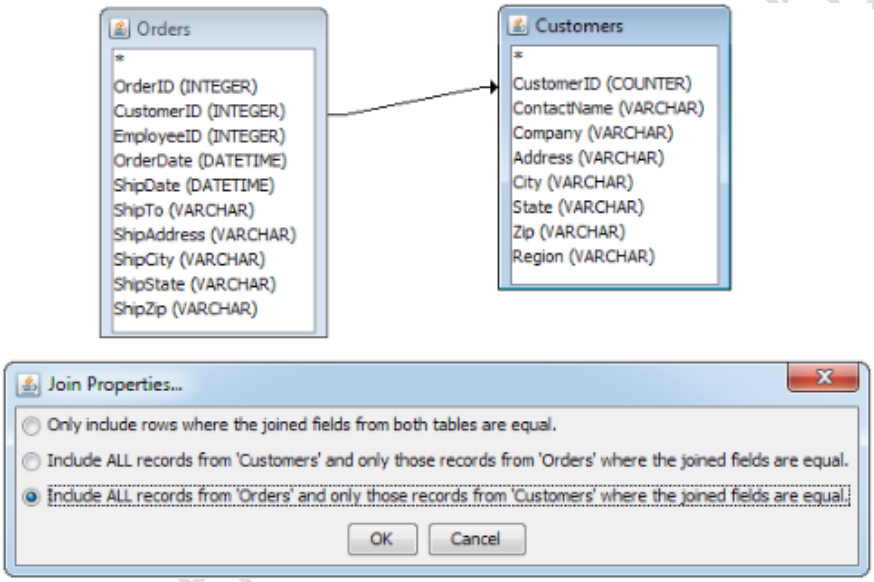
OrderID	CustomerName	Sales
1	Randy	$2,224
2	Sarah	$1,224
3	Randy	$3,115
4	Ivan	$1,221

ご覧のように、**CustomerName** エントリ"Bob"と"Jennifer"は結果セットに表示されません。どちらの

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

顧客も注文をしていないからです。関連するテーブル（この場合は **Orders** テーブル）に一致するレコードが存在するかどうかにかかわらず、すべてのレコード（この例では顧客名）を含めることができます。外部結合を使用してこの結果を達成できます。

クエリビルダには、右または左外部結合のいずれかのオプションが用意されています。キーワード「右」と「左」は重要ではありません。クエリビルダでテーブルが選択された順序で決定されます。外部テーブル（一致する結合条件に関係なくすべてのレコードを含むもの）が最初に選択された場合、クエリビルダは右外部結合を使用します。外部テーブルが他の結合テーブルの後で選択されている場合は、左外部結合が使用されます。この例では、**Orders** テーブルの前に **Customers** テーブルが選択されているため、**CustomerName** フィールドのすべてのレコードを選択するため、クエリビルダは **CustomerID** フィールドに対して右外部結合を使用します。



Join Properties ダイアログ

今、前の例を使用して、**Customers** テーブルのすべてのレコードを含めるように指定した場合を除いて、以前と同じクエリを作成するとします。クエリビルダによって生成される SELECT 文は、以下のようになります。

```
Select Orders.OrderID, Customers.CustomerName, Orders.Sales
From Orders right outer join Customers on Orders.CustomerID = Customers.CustomerID
Order by Orders.OrderID;
```

新しいクエリの結果を以下に示します。

OrderID	CustomerName	Sales
	Jennifer	
	Bob	
1	Randy	$2,224

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2	Sarah	$1,224
3	Randy	$3,115
4	Ivan	$1,221

表示されているように、すべての顧客名が選択され、対応するレコードが存在しない結果セットに NULL 値が挿入されています。外部結合を指定すると、クエリビルダの 2 つのテーブルを結ぶ結合線は結合方向の矢印になります。

Columns

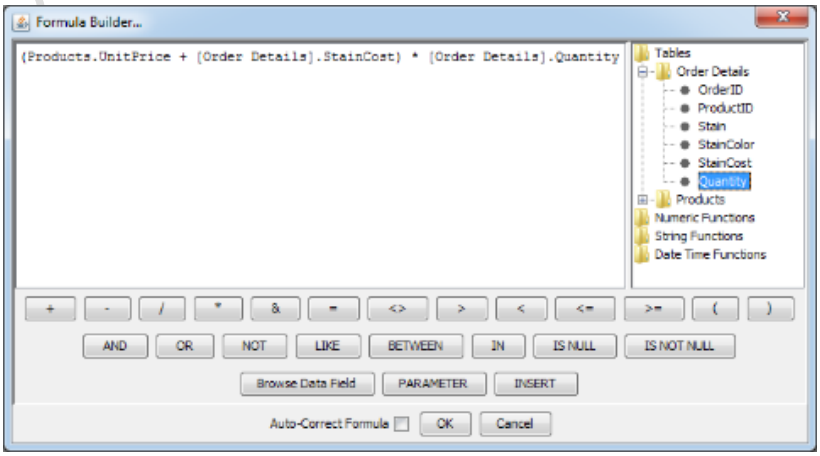
QBE ウィンドウには、照会のために選択された列フィールドに関する情報と選択の条件が含まれています。

列フィールドの選択

次の 2 つの方法のいずれかで選択された任意のテーブルから列フィールドをクエリに追加できます。テーブル内のフィールド名をダブルクリックしてクエリに追加するか、**Table** または **Field** をダブルクリックすると、フィールドの選択肢を含むドロップダウンメニューが表示されます。下のウィンドウを右クリックし、ポップアップメニューから **Delete Column** を選択するか、**Edit** > **Delete Column** を選択して、列から列を削除できます。列フィールドを選択すると、並べ替えフィールドをダブルクリックすることで、昇順または降順に並べ替える方法を指定できます。**Aggregation** フィールドをダブルクリックすることで、グループまたは列の集約を指定することもできます。集計オプションには、**Group By**、**Sum**、**Average**、**Min**、**Max**、**Count**、**Standard Deviation**、**Variance**、**First**、**Last** および **Where** が含まれます。1 つの列に対して group by を選択する場合は、他のすべての列に対して group by（または aggregation）を指定する必要があります。列エイリアスを指定するには、列を右クリックし、ポップアップメニューから **Alias** を選択します。

列の作成

独自の列を作成するには、QBE ウィンドウの空白の列を右クリックし、ポップアップメニューから **Build** を選択します。式ビルダが起動します。式ビルダでは、選択したテーブルと使用しているデータベースの数式ライブラリを使用して、視覚的に列を作成できます。



数式ビルダウィンドウ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

条件

Conditions フィールドまたは **Or** フィールドに条件を入力すると、クエリ選択に条件を配置できます。条件フィールドに置かれた条件は、生成された SQL 内に **AND** 句を作成します。**Or** フィールドに置かれた条件は、SQL 内に **OR** 句を作成します。いずれかのフィールドを右クリックし、ポップアップメニューから **Build** を選択すると、式ビルダが表示されます。式ビルダでは、標準条件（=、<、>、**BETWEEN**、**LIKE**、**NOT** など）を指定するだけでなく、クエリをフィルタ処理する式を作成することもできます。ここでクエリパラメータを指定することもできます。

EspressReport は、フィールド名と囲み文字列の引数を引用符で正しく付加することによって、クエリ条件として入力された項目を自動修正できます。たとえば、= **ARC** と入力すると、EspressReport はクエリ条件を **Categories.CategoryName** = **'ARC**'に変更します。複雑な関数（複数の文字列引数を取るデータベース関数）を使用している場合、EspressReport は関数を適切に解析できないことがあります。式ビルダウィンドウの下部にあるチェックボックスをオフにすることで、自動修正機能をオフにすることができます。

データベース関数の使用

クエリビルダの式ビルダコンポーネントを使用すると、クエリの列または条件を作成するときにデータベース固有の関数を使用できます。提供されている機能を使用することも、独自の機能をインタフェースに追加することもできます。

EspressReport には、Oracle、Access、MS SQL、および DB2 用の関数ライブラリがプリロードされています。これらは、**userdb** ディレクトリの **DatabaseFunctions.xml** ファイルに XML 形式で格納されます。XML に格納されていない機能を持つデータベースの場合、EspressReport はデフォルトのものを使用します。XML ファイルを編集するか、**userdb** ディレクトリの **DatabaseFunctions.dtd** ファイルに基づいて新しいデータベース関数を作成することによって、異なるデータベース関数を指定できます。サンプルのデータベース関数ファイルは、以下のようになります。

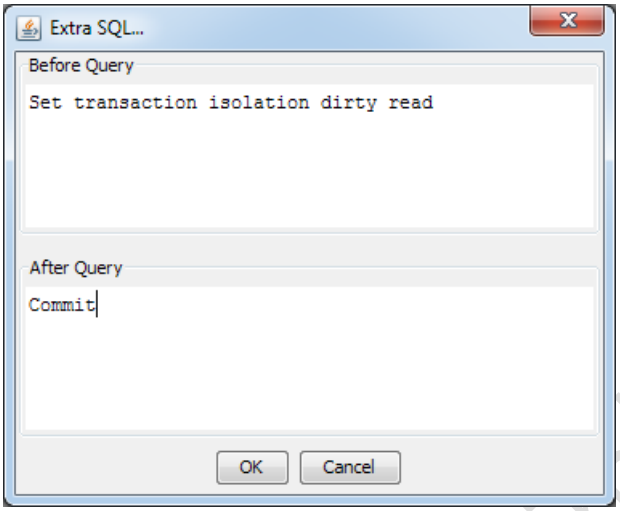
```
<DatabaseFunctions>
        <Database ProductName="ACCESS">
            <FunctionSet Name="Numeric Functions">
                    <Function>Abs(number)</Function>
                    <Function>Atn(number)</Function>
            </FunctionSet>
        </Database>
</DatabaseFunctions>
```

SQL にオプションを追加

場合によっては、クエリの前後に実行するオプションの SQL 文を追加する必要があることもあります。たとえば、クエリを実行する前にトランザクションレベルを設定したり、ストアドプロシージャを呼び出したり、クエリを実行した後にトランザクションをコミットしたり、一時テーブルを削除したりする

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

必要があります。クエリビルダでは、クエリメニューから **Extra SQL** を選択して、これらの余分な SQL 文を指定することができます。これにより、クエリの前後に実行する文を記述できるウィンドウが表示されます。



Extra SQL ダイアログ

ボックスにクエリの前後に実行したい適切な SQL 文を入力することができます。終了したら **OK** をクリックすると、クエリに文が追加されます。

クエリ出力

SQL View タブと **Datasheet View** では、クエリの 2 つの異なるビューを表示できます。

SQL View

SQL View には、デザインビューでのクエリによって生成された SQL 文が表示されます。これにより、クエリビルダでの、さまざまな操作が SQL へどのように変換されたかを確認できます。必要に応じて生成された SQL を編集できますが、SQL を変更して **Design View** に戻すと、行った変更はすべて失われます。SQL を変更した後にクエリを保存した場合、そのクエリの編集を選択すると、**SQL View** で再度開きます。

Datasheet View

Datasheet View タブには、クエリ結果がデータテーブル形式で表示されます（このタブは **Enter SQL** ダイアログでも使用できます）。データシートビューには、クエリを実行した結果として描画されるすべてのデータが表示されます。データシートビューに行くと、クエリをテストして設計エラーをチェックできます。ウィンドウの下部にあるツールバーを使用して、クエリの結果をナビゲートできます。

データテーブルの最初のページに移動します。

データテーブルの前のページに移動します。

特定のデータ行に移動する

データテーブルの次のページに移動します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

データテーブルの最後のページに移動します。

ページごとに表示する行数を設定します（デフォルトは 30)。

クエリのエクスポート

クエリは 2 つの方法のいずれかでエクスポートできます。SQL 文をテキストとして出力することも、クエリ結果を CSV ファイルとして出力することもできます。クエリをエクスポートするには、**File** > **Export** を選択します。2 番目のメニューが表示され、**Generate SQL** オプションまたは **Generate CSV** オプションが表示されます。目的のオプションを選択すると、ファイル名と場所を指定するダイアログボックスが表示されます。

クエリを保存してクエリビルダを終了するには、**File** > **Done** を選択します。

3.3.2.2.2 パラメータ化されたクエリ

また、クエリビルダを使用してパラメータ化されたクエリを設計することもできます。この機能を使用すると、実行時にデータをフィルタリングできます。パラメータ化されたクエリは、チャート/レポートのパラメータドリルダウンにも使用されます。ドリルダウンチャートの詳細については、ドリルダウンを参照してください。

クエリパラメータは、SQL 文の入力時またはクエリビルダの使用時に定義できます。データビューを実行するときに定義することもできます（次のセクションで説明します)。パラメータは、SQL 文内で**":"** 文字で指定されます。一般に、このパラメータは SQL Select 文の WHERE 句に配置されます。たとえば、次の SQL 文では **Name** というパラメータを指定します。

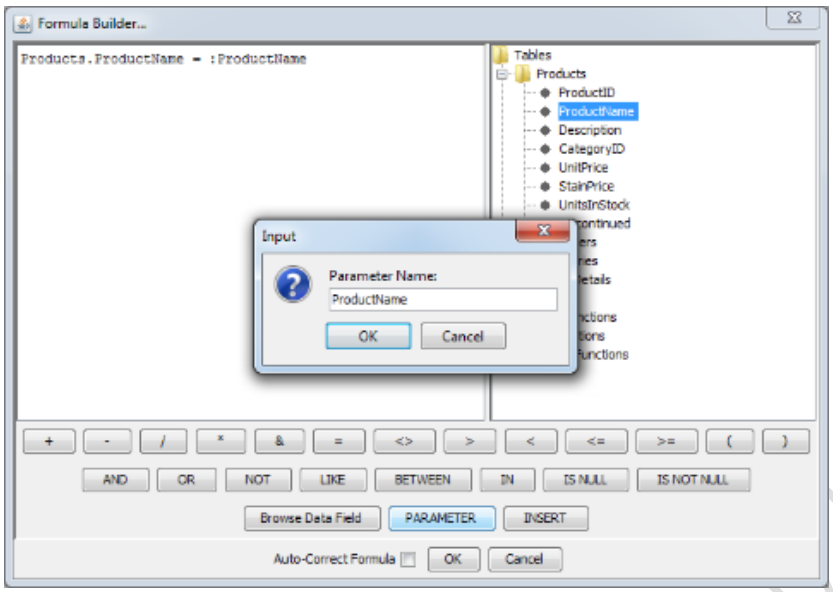
```
Select * From Products Where ProductName = :Name
```

実行時に製品名を入力し、その製品のデータのみを取り出すことができます。

クエリビルダ内では、**Condition** フィールドを右クリックし、ポップアップメニューから **Build** を選択することで、クエリパラメータを指定できます。式ビルダが開き、条件を列に配置できます。

Copyright(C) 2018 Climb Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



数式ビルダでのパラメータの指定

PARAMETER ボタンをクリックすると、パラメータを挿入できます。2 番目のダイアログが表示され、パラメータの名前を指定するよう求められます。パラメータ名を入力し、**OK** をクリックしてもう一度 **OK** をクリックして式ビルダを閉じます。クエリには、好きなだけ多くの異なるパラメータを指定できます。

複数値パラメータ

EspressReport は、単一の値ではなく配列を持つ種類のパラメータをサポートしています。複数値のパラメータは、パラメータ数が不明な値に基づいて結果セットをフィルタ処理する場合に便利です。たとえば、特定の都道府県の顧客リストを返すためのチャートが実行されているとします。ユーザは必要な数の異なる州/県を選択して関連情報を返すことができます。

複数値のパラメータを作成するには、SQL 文の **IN** 句内にパラメータを配置します。たとえば、以下のクエリでは **State** という名前の複数値パラメータが作成されます。

```
Select Customers.Company,Customers.Address,Customers.City,Customers.State,Customers.Zip
From Customers
Where Customers.State IN (:State);
```

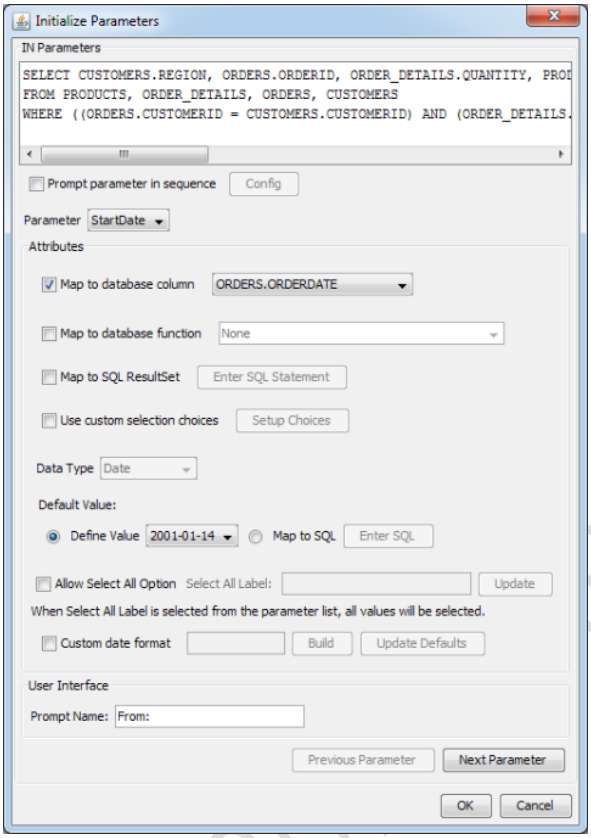
複数値のパラメータは、IN 句に 1 つのパラメータしかない場合にのみ作成されます。**IN** 句、つまり **Customers.State IN**（**:State1 :State2, :State3**）に複数のパラメータを配置すると、代わりに 3 つの単一値パラメータが作成されます。

クエリパラメータの初期化

パラメータ化されたクエリを保存する（**File > Done** を選択して）か、または **Datasheet View** タブをクリックしてプレビューすると、パラメータの初期化が求められます。また、クエリメニューから **Intialize Parameter** を選択するか、SQL の入力ダイアログで **Intialize Parameter** ボタンをクリックし

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

て初期化することもできます。



Initialize Parameters ダイアログ

このダイアログでは、以下のオプションを指定できます。

Map to database column:

このオプションを使用すると、パラメータ入力に値が使用されるデータベースの列を指定できます。このオプションを選択すると、Report Viewer でレポートをプレビューまたは実行するときにエンドユーザが取得するパラメータプロンプトが変更されます。パラメータをデータベース列にマップすると、パラメータ値を選択するための個別の値のドロップダウンリストが表示されます。マップしないと、ユーザは特定のパラメータ値を入力する必要があります。

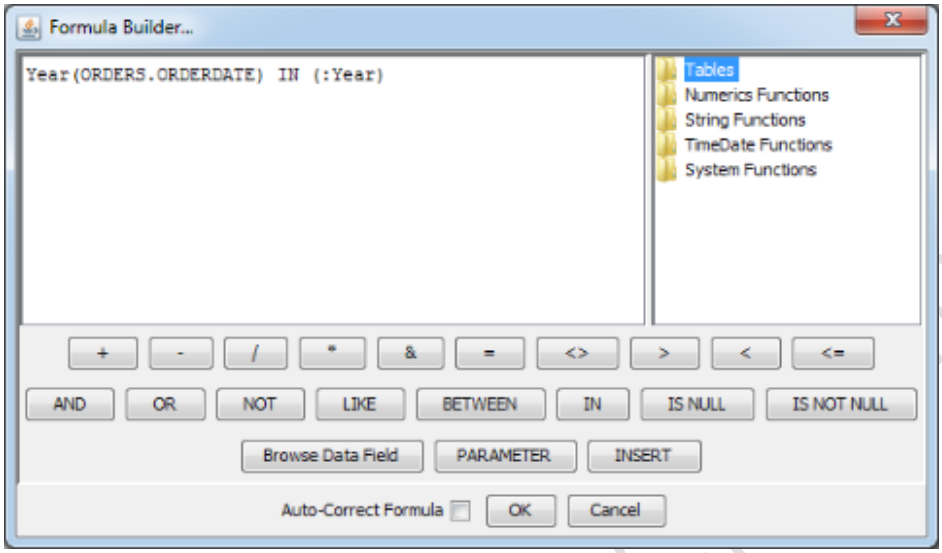
通常、このドロップダウンリストは、クエリから結合および条件を適用しながら、列に SELECT DISTINCT 文を実行することによって移入されます。制約なしでカラムからすべてのデータを取得したい場合（パラメータプロンプトのパフォーマンスを改善することがある）、EspressManager の起動時に **singleTableForDistinctParamValue** 引数を設定できます。EspressManager の設定オプションの詳細については、EspressManager の起動を参照してください。

Map to database function:

リストボックスのパラメータに有効な値を入力する場合は、データベース列のマップ機能は非常に便利ですが、データベース列の計算値または派生値が必要な場合があります。たとえば、2007 年のすべての注文を検索したいとします。ただし、OrderDate は日付です。必要なのは、OrderDate 列に Year 関

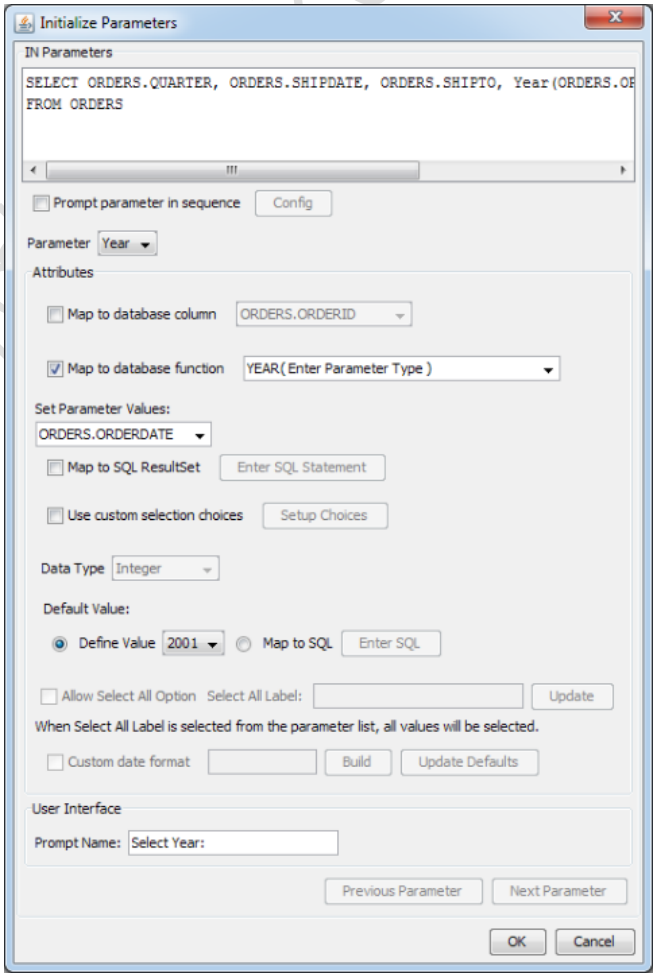
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

数を適用することです。この機能はバックグランドで動作させることができます。パラメータをデータベース機能にマッピングすることは、カラムにマッピングすることと非常によく似ています。式ビルダで、関数の結果と以下のようなパラメータを比較する条件を入力します。



データベース関数のマッピングの条件

Initialize Parameters ダイアログで、**Map to database function** チェックボックスをチェックすると、値が自動的に入力されます。



Copyright(C) 2018 Climb.Inc. All Rights Reserved.

パラメータをデータベース関数にマップする

カスタム関数のリストは、**/userdb/**ディレクトリにある **DatabaseFunctions.xml** ファイルから抽出されます。新しいデータベースまたはカスタム関数を追加する場合は**.xml** ファイルを変更します。新しい機能は、プログラムを再起動するとこのリストに表示されます。

データベースが**.xml** ファイルにリストされていない場合、関数リストには JDBC ドライバにリストされている関数が取り込まれます。ただし、関数パラメータは提供されていません。たとえば、HSQL データベースは**.xml** ファイルには表示されません。

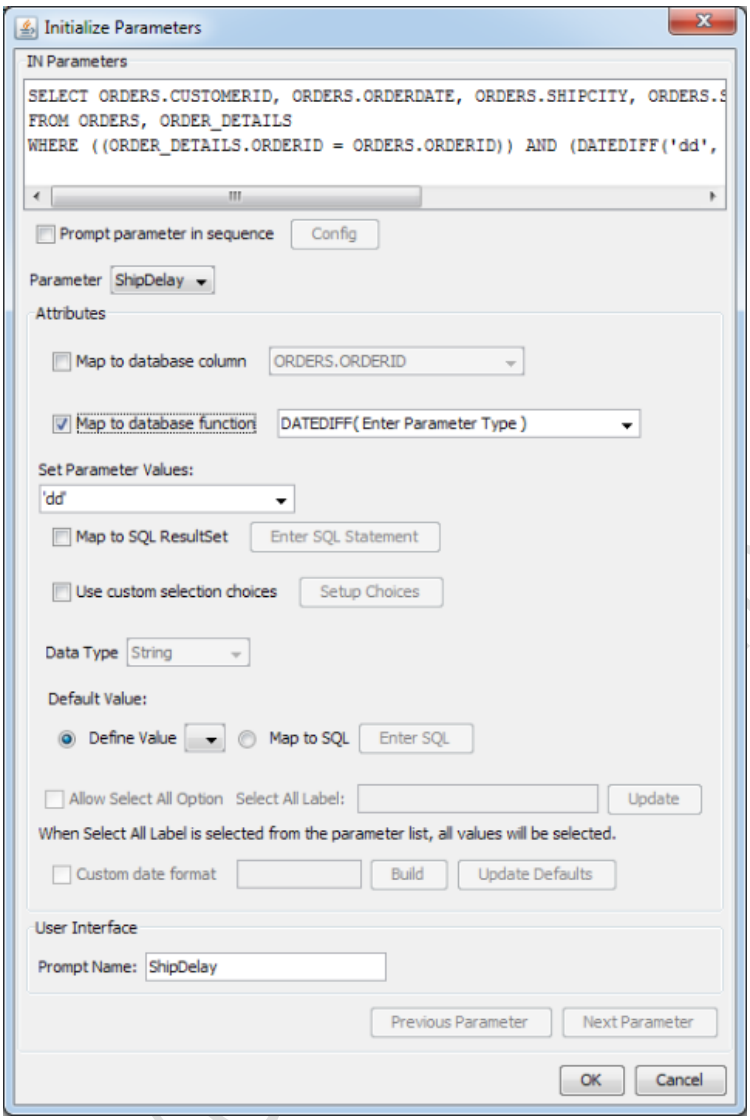
HSQL データベースを使用した興味深い例は次のとおりです。遅延した注文のレポートを作成するとします。HSQL の DateDiff 機能を利用して、発注日数を確認することができます。

```
DATEDIFF('dd'、ORDERS.ORDERDATE、ORDERS.SHIPDATE)>=：ShipDelay
```

この関数は、受注日と出荷日の差を求め、その結果を日数で表示します。パラメータを初期化してマップをデータベース関数にチェックインすると、以下のウィンドウが表示されます。

Copyright(C) 2018 Climb.Inc.

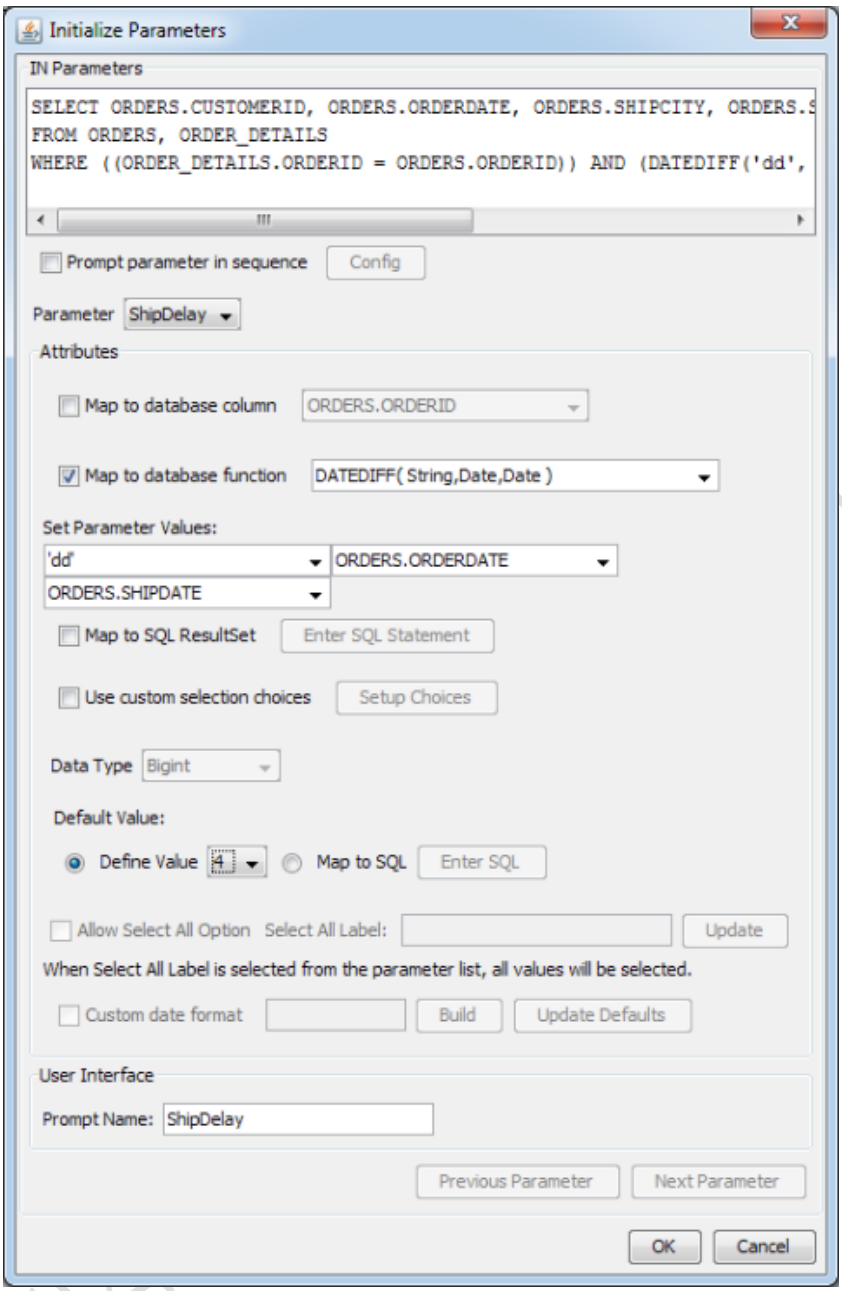
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



パラメータタイプの存在しない HSQL 関数

DateDiff 関数には、パラメータの文字列と 2 つの日付値が必要です。カッコ内にこれらのパラメータタイプを入力します。これにより、3 つの設定パラメータ値リストが表示されます。最初のパラメータに **dd**（日）を入力し、2 番目のパラメータのリストから **Orders.OrderDate** を選択し、3 番目のパラメータのリストから **Orders.ShipDate** を選択します。デフォルト値は関数の結果で更新されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



HSQL 関数へのパラメータのマップ

Map to SQL ResultSet

データベース列にマップされたパラメータは、ユーザがグラフを実行するときに選択できるドロップダウンリストボックス内の個別の値のリストを提供します。ただし、値のリストを生成するには、クエリの結合条件を使用して列の SELECT DISTINCT 文を実行します。場合によっては、これは時間のかかるプロセスになる可能性があります。この問題を解決し、実際にドロップダウンリストボックスに何を埋め込むかを完全に制御するために、独自の SELECT 文を記述してドロップダウンリストを作成することができます。さらに、クエリに含まれるパラメータをこのクエリに含めることができるということです。適切な結合とパラメータを使用すると、この機能を使用してカスケードパラメータ（詳細はカスケーディングパラメータを参照）を容易にすることができます。例は以下のとおりです。

クエリに 2 つのパラメータがあるとします。したがって、クエリは以下のようになります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
SELECT CATEGORIES.CATEGORYID, PRODUCTS.PRODUCTNAME, PRODUCTS.UNITPRICE,
PRODUCTS.UNITSINSTOCK
FROM PRODUCTS, CATEGORIES
WHERE ((PRODUCTS.CATEGORYID = CATEGORIES.CATEGORYID))
AND (((CATEGORIES.CATEGORYID =:category) AND (PRODUCTS.PRODUCTNAME =:product)))
```

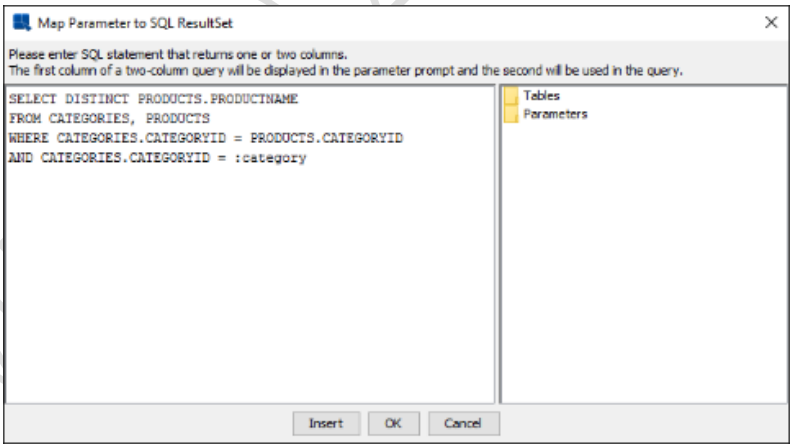
初期化パラメータの **config** プロンプトで、パラメータのプロンプトの順序を**カテゴリ（category）**に、次に**プロダクト（Product）**に設定します。

パラメータカテゴリの SELECT 文は、以下のとおりです。

```
SELECT DISTINCT CATEGORIES.CATEGORYID
FROM CATEGORIES
```

パラメータ **product** の SELECT 文は以下のようになります。

```
SELECT DISTINCT PRODUCTS.PRODUCTNAME
FROM CATEGORIES, PRODUCTS
WHERE CATEGORIES.CATEGORYID = PRODUCTS.CATEGORYID
AND CATEGORIES.CATEGORYID = :category
```



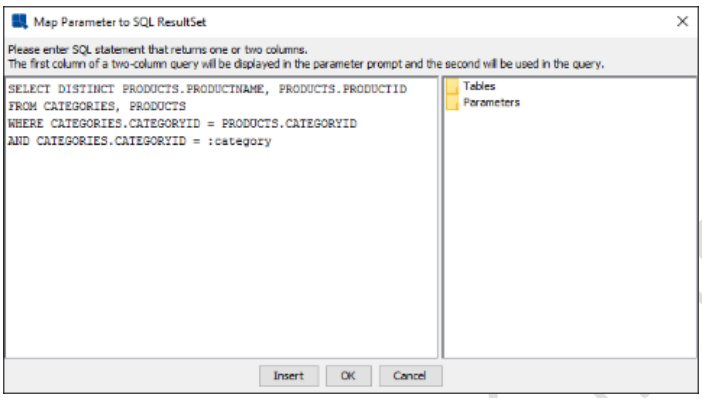
プロダクトの文の選択

ユーザがテンプレートを実行すると、カテゴリが最初に表示されます。次に、選択されたカテゴリの値が製品のフィルタリングに使用されます。

パラメータにマップされた SELECT 文は、SELECT リストに 1 つまたは 2 つのカラムを持つことができます。1 つの列が選択リスト内にある場合は、その列が、そのパラメータの個別の値のリストを提供する列でなければならないことは明らかです。ここで提供されるもう 1 つの便利な機能は、実際には、列の最初の列がドロップダウンリストの値を提供し、2 番目の列がフィルタ条件の実際のパラメータ値になるように、選択リストで 2 つの列を選択できることです。次の例を考えてみましょう。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

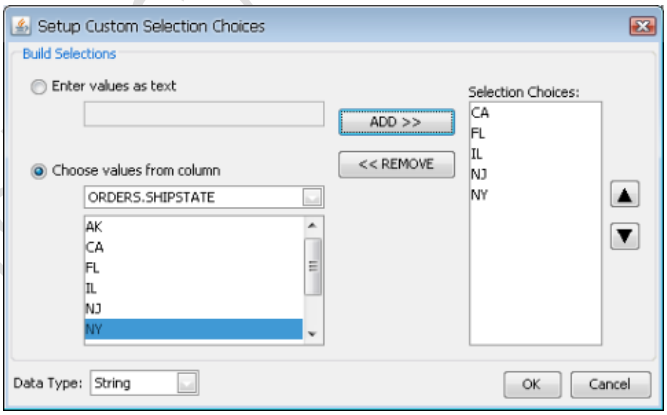
データベースにプロダクト ID が主キーとして含まれているテーブルがあるとします。エンドユーザがデータベースから製品を検索する場合、プロダクト ID は単なる暗黙のコードである可能性があるため、パラメータとして製品名を使用したい状況にあります（製品名が最初にクエリにリストされ、プロダクト ID が 2 番目にリストされます)。この機能を使用すると、ドロップダウンリストの値の製品名と実際の値フィルタ条件としての製品 ID を選択できます。



2 つのカラムを持つステートメントの選択

Use custom selection choices

個別の列の値をすべて含むドロップダウンメニューを表示するのではなく、パラメータ値のカスタムリストを作成することもできます。このリストを作成するには、このオプションを選択し、**Setup Choices** をクリックします。これにより、新しいダイアログが開き、選択肢のリストを作成することができます。



Custom Parameter List ダイアログ

このダイアログでは、カスタム値を入力するか、データベースの列の個別の値から値を選択することができます。リストの値の指定が終了したら、**OK** ボタンをクリックすると選択内容が保存されます。

Default Value

これにより、パラメータのデフォルト値を指定することができます。デフォルト値を指定する必要はありませんが、指定することをお勧めします。デフォルト値を指定しないと、データソースが存在しない場合でもグラフテンプレートを開いたり、操作したりすることはできません。

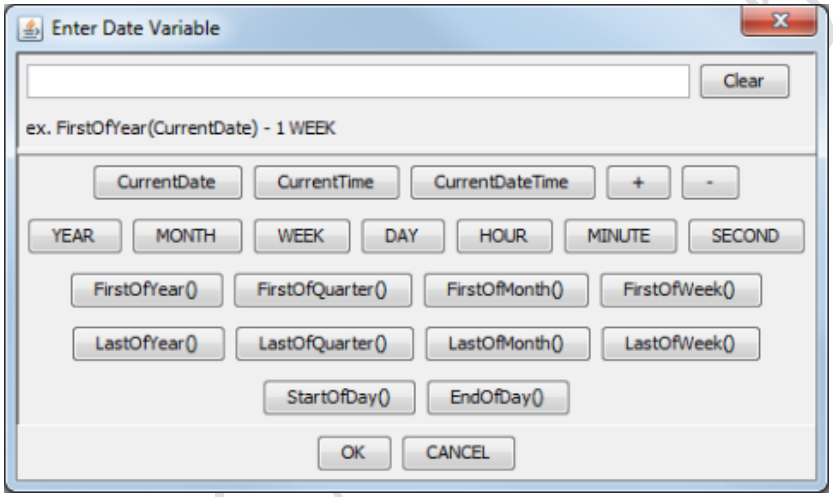
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

手動で 1 つの値を選択するか（リストから選択するか手動で入力するか、選択したマッピング方法に依存します)、デフォルト値を SQL クエリにマップします。

複数値のパラメータ(複数値パラメータを参照)の場合、SQL クエリは複数の値を返すことができます。このような場合、デフォルトのパラメータ値としていくつかの値が選択されます。

Date Variable

このオプションは、パラメータがデータベースの列または関数にマップされていない場合、または SQL 結果セットにマップされ、カスタム選択項目に設定されていない場合にのみ使用できます。このオプションは、日付/時刻が可変なパラメータの場合にのみ使用します。このボタンをクリックすると、以下のパネルが表示され、サポートされているすべてのキーワードが一覧表示されます。



Enter Date Variable ダイアログ

このダイアログでは、**CurrentDate**、**CurrentTime**、および **CurrentDateTime** という 3 つのキーワードのいずれかを選択できます。現在の日付/時刻から時間単位を加算または減算することができ、動的な日付範囲を持つことができます。たとえば、グラフには次のデフォルト値を設定できます。

```
StartDate: CurrentDate - 1 WEEK
EndDate: CurrentDate
```

これは、レポートが実行されるたびに、デフォルトのプロンプトが 1 週間前から現在の日付になることを示します。サポートされている他の時間単位は、**YEAR、MONTH、DAY、HOUR、MINUTE、SECOND** です。この機能では、1 つの加算または減算しかサポートされず、複数値のパラメータはサポートされません。

関数を使用してパラメータ値を定義することもできます。

FirstOfYear()

引数の形式:**CurrentDate、CurrentDateTime、 例: FirstOfYear(CurrentDate)**

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

この関数は、引数から年の最初の日の日付を返します。たとえば、引数が **2012-08-14** に評価されると、関数は **2012-01-01** を返します。

LastOfYear()
引数の形式: **CurrentDate、CurrentDateTime、 例: LastOfYear(CurrentDate)**

この関数は、引数から年の最終日の日付を返します。たとえば、引数が **2012-08-14** に評価されると、関数は **2012-12-31** を返します。

FirstOfQuarter()
引数の形式: **CurrentDate、CurrentDateTime、 例: FirstOfQuarter(CurrentDate)**

この関数は、引数の日付を含む四半期の最初の日の日付を返します。たとえば、引数が **2012-08-14** に評価されると、関数は **2012-07-01** を返します。

LastOfQuarter()
引数の形式: **CurrentDate、CurrentDateTime、 例: LastOfQuarter(CurrentDate)**

この関数は、引数の日付を含む四半期の最終日の日付を返します。たとえば、引数が **2012-08-14** に評価されると、関数は **2012-09-30** を返します。

FirstOfMonth()
引数の形式: **CurrentDate、CurrentDateTime、 例: FirstOfMonth(CurrentDate)**

この関数は、引数から月の最初の日の日付を返します。たとえば、引数が **2012-08-14** に評価されると、この関数は **2012-08-01** を返します。

LastOfMonth()
引数の形式: **CurrentDate、CurrentDateTime、 例: LastOfMonth(CurrentDate)**

この関数は、引数から月の最終日の日付を返します。たとえば、引数が **2012-08-14** に評価されると、関数は **2012-08-31** を返します。

FirstOfWeek()
引数の形式: **CurrentDate、CurrentDateTime、 例: FirstOfWeek(CurrentDate)**

この関数は、引数の日付を含む週の最初の曜日の日付を返します。たとえば、引数が **2012-08-14** に評価されると、関数は **2012-08-12** を返します（日曜日を週の初めとみなします）。

LastOfWeek()

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

引数の形式: **CurrentDate、CurrentDateTime、 例: LastOfWeek(CurrentDate)**

この関数は、引数の日付を含む最終曜日の日付を返します。たとえば、引数が **2012-08-14** に評価されると、関数は **2012-08-18** を返します（土曜日を週末とみなします)。

StartOfDay()
引数の形式: **CurrentTime、CurrentDateTime、 例: StartOfDay(CurrentDateTime)**

この関数は、引数から日の開始時刻を返します。たとえば、引数が **2012-08-14 12:15:03** に評価されると、関数は **2012-08-14 00:00:00.0** を返します。

EndOfDay()
引数の形式: **CurrentTime、CurrentDateTime、 例: EndOfDay(CurrentDateTime)**

この関数は、引数からの 1 日の終了時刻を返します。たとえば、引数が **2012-08-14 12:15:03** に評価されると、関数は **2012-08-14 23:59:59.999** を返します。

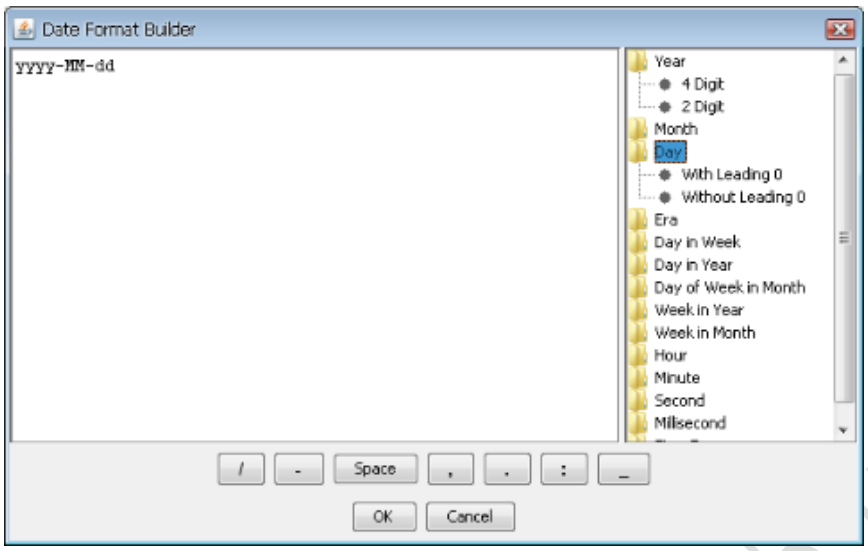
Data Type
これにより、パラメータ値のデータ型を指定することができます。パラメータを列にマップした場合、データ型は自動的に設定されます。

Allow Select All Option
これを使用して、パラメータプロンプトダイアログにオプションを追加します。このオプションを使用すると、単一値のパラメータに対してもすべてのパラメータ値を選択できます。詳細はすべてのパラメータを参照してください。

Custom Date Format
これにより、日付パラメータを入力する形式を設定することができます。このオプションは、パラメータを列にマップしなかった場合、またはカスタム選択肢を入力した場合（つまり、エンドユーザが日付値を入力する場合）にのみ使用できます。このオプションをオンにすると、時刻要素を表す文字の組み合わせで日付形式を入力できます。ビルドボタンをクリックすると、日付書式ビルダを使用して書式を簡単に作成できます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Date/Time Format Builder

ビルダには、右側に利用可能な要素のリストが含まれています。要素の上にマウスを置くと、各プレゼンテーションの例が表示されます。下部には、使用可能なセパレータのセットが含まれています。

文字とそのフォーマットの詳細については、日付/時刻関数の printDate()関数のドキュメントを参照してください。このオプションの書式設定は、関数の書式引数と同じです。

プロンプト名

これにより、パラメータダイアログでユーザに与えられるプロンプトを指定することができます。

パラメータをマップすると、ドロップダウンボックス(単一値パラメータ)または様々なオプションを含むリストボックス(複数値パラメータ)のいずれかが表示されます。パラメータをマップしないことを選択した場合、ユーザは独自の値を入力するためのテキストボックスを表示します。複数値のパラメータの場合、このパラメータが複数の値を受け入れることをユーザがパラメータプロンプトに通知することをお勧めします。ユーザは複数の値をカンマ(例：**ARC**, **DOD**, **TRD**)で区切ることができます。テキストにコンマを使用する必要がある場合は、引用符を使用してコンマをギル他文字列に含めることができます(**"Doe, John", "Smith, Mike"**)。

Previous Parameter と **Next Parameter** をクリックすると、クエリで定義されている各パラメータを初期化できます。

パラメータ化されたクエリを使用してグラフを設計するか、パラメータ化されたクエリを使用するグラフを開くかを選択すると、グラフは規定値で読み込み、開始されます。グラフをプレビューするときに、パラメータ値を入力するよう求められます。

カスケーディングパラメータ

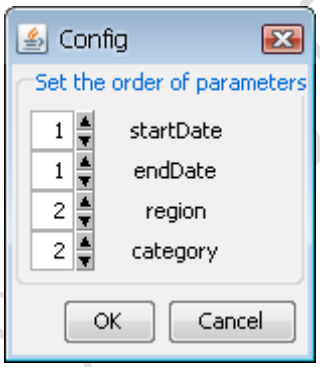
デフォルトでは、プロンプトダイアログにすべてのレポートパラメータを一度に入力するように求められます。ただし、一部のパラメータが多数の異なる値を持つデータベース列にマップされている場合、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

この構成は最善の方法ではない場合があります。非常に大きなリストから選択するのは難しい場合があり、パラメータの組み合わせによっては、ユーザがデータを返さないパラメータを選択できる場合があります。

これらの問題を解決するために、EspressReport には、パラメータの入力順序をユーザが設定できる機能があります。この機能を有効にすると、ユーザは事前定義された順序でダイアログにパラメータを入力します。したがって、各選択は、次のパラメータプロンプトにフィルタとして適用されます。カスケーディングパラメータを使用すると、ユーザに提示される個別の値の数が制限され、ユーザが無効なパラメータの組み合わせを選択できなくなる可能性があります。

カスケードパラメータを有効にするには、パラメータ初期化ダイアログで **Prompt Parameter in sequence** と表示されているオプションを順番にチェックします。次に、**Config** ボタンをクリックして、パラメータプロンプトの順序を設定します。クエリに定義されているすべてのパラメータを示すダイアログが開きます。



Parameter Sequence ダイアログ

スピンボックスを使用して、クエリパラメータのシーケンスを設定できます。ユーザは、最も低い番号のパラメータから始め、最高のものまで自分のやり方で作業するよう促されます。2 つ以上のパラメータが同じ番号を共有する場合、ユーザは同時に（同じダイアログ内で）それらのパラメータを入力するように求められます。

デフォルトでは、次のレベルのパラメータ値は、以前にプロンプトされたパラメータを入力してクエリ全体を再実行することによって生成されます。元のクエリの実行が遅い場合は、高次パラメータを SQL クエリにマッピングしてパフォーマンスを向上できます。以前に選択したパラメータ値をマップされたクエリに含めることもできます。詳細は、クエリパラメータの初期化を参照してください。

すべてのパラメータ

時々、すべてのパラメータ値を一度に選択したいことがあります。すべてのパラメータ機能を使用すると、以下のようにすることができます。

単一パラメータ

複数値選択ができないパラメータの場合でも、一度にすべてのパラメータ値を選択することができます。

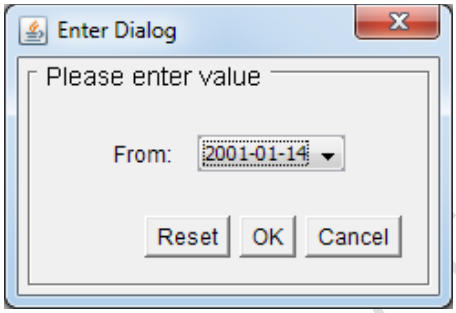
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

複数値選択と全値選択には違いがあります。詳細は、内部動作の章を参照してください。

たとえば、以下のような条件があるとします。

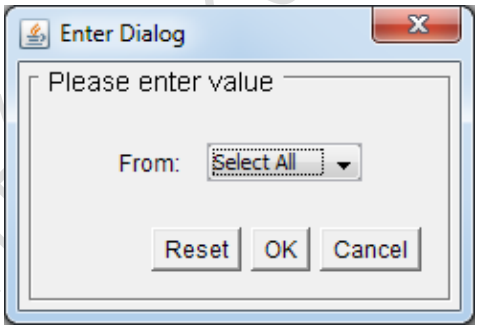
```
WHERE column = :Parameter
```

このような場合、パラメータプロンプトダイアログでは複数の値を選択することはできません。



典型的な単一値パラメータ

ただし、**Select All Values** を使用してパラメータ値リストにオプションを追加すると、(パラメータが複数値選択を許可していなくても)視聴者がすべてのパラメータ値を一度に選択できるようになります。

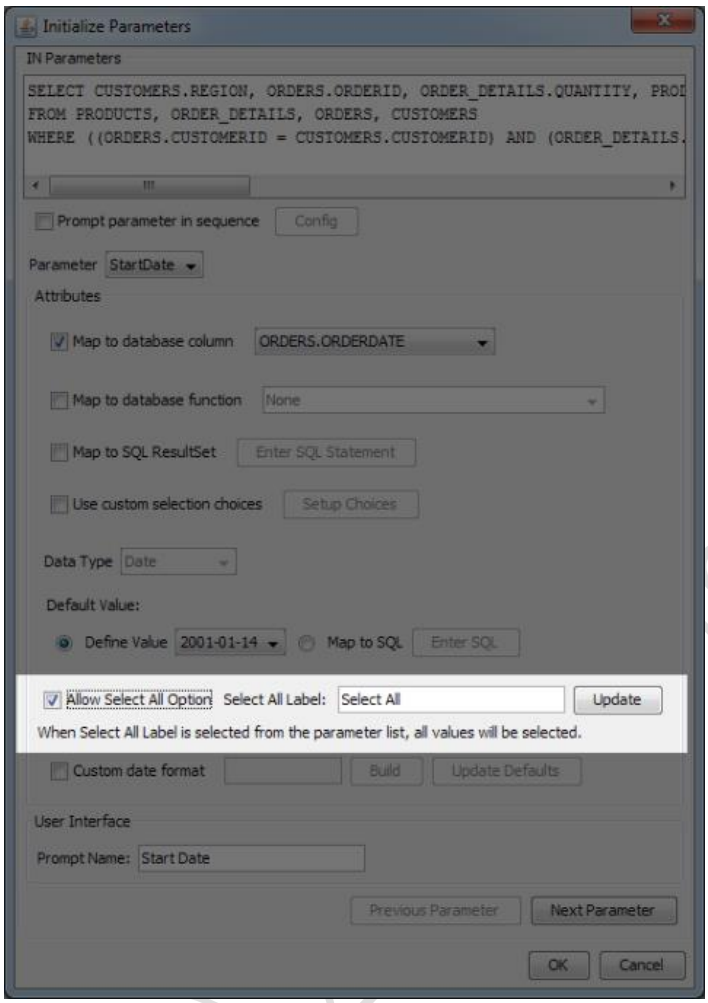


Select All 機能を有効にした単一値パラメータ

Select All Option チェックボックスを選択すると、**Intialize Parameters** ダイアログ(クエリパラメータの初期化を参照)で **Allow Select All Option** を有効にすることができます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



このオプションは、以下の要件を満たすパラメータでのみ使用できます。

1. このパラメータは、以下の演算子のいずれかを使用します。クエリにこのパラメータが複数存在する場合は、すべてのパラメータ比較演算子は以下いずれかでならなければなりません。

演算子	説明
<	未満
<=	以下
>	超過
>=	以上
=	等しい

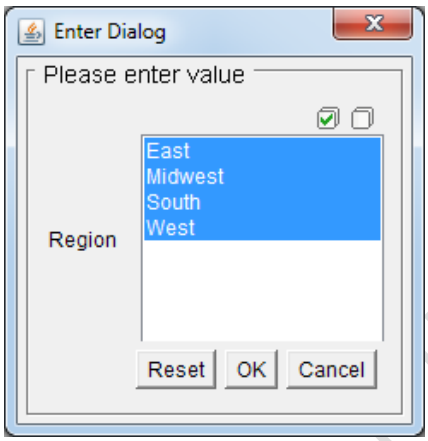
2. パラメータは、パラメータ条件から列と同じ列にマップされるか、パラメータは何にもマップされません。

Allow Select All Option を選択すると、**Select All Label** フィールドがアクティブになり、クエリのすべてのデータを選択するためのテキストを入力できます列にマッピングされたパラメータの場合、このテキストは 1 番目のパラメータ値リストに表示されます。何にもマッピングされていないパラメータ値リストに表示されます。何にもマッピングされていないパラメータの場合、このテキストをパラメータ値として入力すると、すべてのデータが選択されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Multi-value parameters

単一値パラメータとは異なり、**Select All** 機能はデフォルトですべての複数値パラメータに対して有効です(実際には無効にすることは、できません。複数値パラメータに対して無効にすると意味がありません)。この機能を使用するには、パラメータプロンプトで **Select All** アイコンをクリックするだけです。



ただし、複数値のパラメータは 2 つのモードで動作します。

1. パラメータが前の段落の条件を満たし、パラメータが最初のカスケーディングレベルにある場合(つまり、パラメータカスケードが無効になっている場合、またはそのパラメータが最初のカスケードレベルにある場合)、SQL パーサによって解析され、パラメータ条件は無効になります。パラメータを無効にするとクエリが最適化され、パフォーマンスの問題やエラーが発生することはありません。それがどのように機能するかについては、内部動作を参照してください。

2. パラメータが前の段落の条件を満たしていない場合、または最初のカスケーディングレベルにない場合、選択肢たちがコンマで区切られた値のリストとしてクエリに挿入されます。クエリに大量の値をリストとして注入すると、クエリがかなり長くなる可能性があります。クエリが長くなるとパフォーマンスの問題やエラーが発生する可能性があるため、このオプションは**多くの値を持つパラメータには使用しないでください。**

内部動作

チャートビュアーが、単一値パラメータまたはパラメータ無効化の条件を満たす複数値パラメータのすべてのパラメータ値を選択すると、クエリは自動的に解析され、基本的にパラメータを無効にする特別な条件がパラメータに追加されます。

例：以下のクエリ

```
select *
from table
where column > parameter_value
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

解析され、以下のようにデータベースに渡されます。

```
select *
from table
where ((column > parameter_value) OR (1 = 1))
```

また、<(より小さい)または>(より大きい)演算子のすべての値を選択すると、データを全く返さないのではなく、テーブルからすべての値を返します(条件が WHERE＜日付の例のすべてのデータ＞＞日付はデータを返しません)。

EspressReport では多くのデータベースシステムを使用できますので、特定のデータベースで複雑なクエリが発生すると解析が失敗することがあります。このような場合、警告ダイアログが表示されます。

このような状況では、以下の 3 つのオプションがあります。

1. パーザによって解析できるようにクエリを修正してみてください。
2. 自分自身を追加するすべてのパラメータ条件をクエリに選択します。

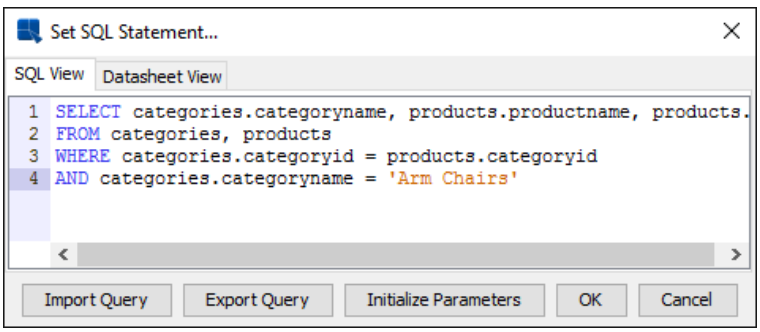
例えば

```
WHERE ((column = :Parameter) OR (:Parameter LIKE 'selectall'))
```

すべてのパラメータをクエリに直接埋め込む場合は、**Allow Select All Option** を無効にしたままにします。

3.3.2.2.3 SQL 文の入力

通常、クエリビルダをクエリの作成に使用します。ただし、SQL 文を直接入力する必要がある場合があります。たとえば、クエリが QRY ファイルで既に作成されている場合、クエリがストアドプロシージャ／関数に組み込まれている場合、またはクエリがクエリビルドでサポートされていないコマンドを必要とする場合などです。このような状況では、**Enter SQL Statements** を選択して、**Set SQL Statement** ウィンドウを開きます。ここでは、以下のように SQL 文をテキストエリアに直接入力するか、既存の QRY ファイルを読み込むことができます。



Enter SQL Statement ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

結果をプレビューするには、**Datasheet View** をクリックします。

Oracle ストアドプロシージャの呼び出し

Oracle は、ほかのデータベースシステムと比較して、ストアドプロシージャおよび関数に関して異なるアプローチを採用しています。たとえば、MS SOL Server では、EXEC コマンドを強いようすると結果セットが返されます。ただし、結果セットを戻すには、**REF CURSOR** 型の **OUT** パラメータを使用する必要があります。さらに、Oracle は単一のクエリから複数の分を受け入れません。したがって、ストアドファンクション内にクエリを格納し、既存の Oracle ストアドプロシージャにアクセスするために、特殊な構文を使用する必要があります。

Oracle ストアドプロシージャにアクセスするには、最初に、以下の PL/SQL 文を使用して弱く型付された **REF CURSOR** を定義します。

```
CREATE OR REPLACE PACKAGE types
AS

        TYPE ref_cursor IS REF CURSOR;

END;
```

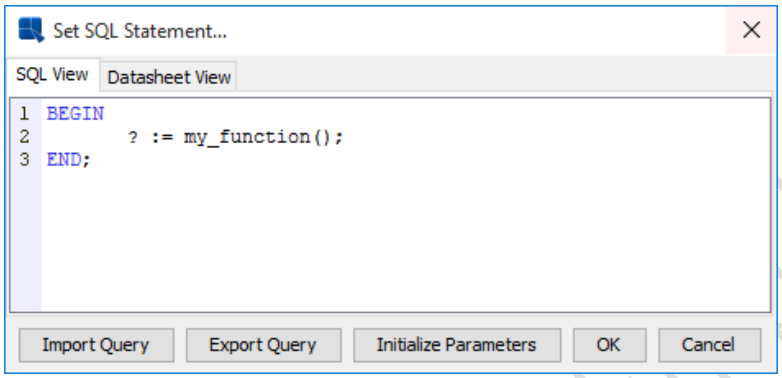
この **ref_cursor** 型は、クエリ結果セットを格納し、**OUT** パラメータとして返すために使用されます。次に、ストアドプロシージャを呼び出してクエリを実行する関数を作成します。以下のスケルトンコードは、**ref_cursor** 型を使用して簡単なクエリを返します。

```
CREATE OR REPLACE FUNCTION my_function()

        RETURN types.ref_cursor

AS

        result_cursor types.ref_cursor;

BEGIN

        do_stored_procedure();
        OPEN result_cursor FOR
                SELECT * FROM Categories

        RETURN result_cursor;
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
END;
```

Oracle のストアドファンクションが設定されたので、Chart Designer から PL/SQL のような特殊な構文を使用して簡単に呼び出すことができます。Set SQL Statement ウィンドウで、以下の構文を入力して、Oracle ストアドファンクションを呼び出します。



Oracle ストアドファンクションの呼び出し

BEGIN…END;構文は、ユーザが Oracle ストアドファンクションにアクセスしようとしていることをシステムに警告します。そして**"?"**は、変数が **OUT** パラメータのために予約されていることを ReportDesgner に通知します。Oracle ストアドプロシージャを呼び出すための JDBC 構文は以下の通りです。

```
( call ? := my_function() )
```

ただし、EspressReport はこの形式をサポートしていません。**DateSheet View** をクリックして、結果をプレビューします。

EspressReport でストアドプロシージャを使用して有用なソリューションを開発する方法を説明する、より具体的な例を示します。以下に示すような組織のロケーション改装を格納する **employee_table** というテーブルがあるとします。

ID	NAME	PARENT	EMPLOYEE
1	All	NULL	0
2	America	1	0
3	Europe	1	0
4	New York	2	20
5	Santa Clara	2	30
6	Dallas	2	12
7	London	3	14
8	Paris	3	11

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

この表には、企業の様々な場所がツリー構造で一覧表示されています。従業員の数は、葉ノード(たとえば、ニューヨーク、ロンドンなど)に格納され、各ノードは、その直接の親に関する情報を含む。ある地域の従業員数と、その地域内の別々の枝に関する情報を表示するチャートを作成するとします。たとえば、ユーザが **ID = 2(アメリカ)**を入力した場合は、支店の所在地とともに米国内の従業員の総数をグラフに表示する必要があります。Oracle の **CONNECT BY** 句と **START WITH** 句を使用すると、2 つのシンプルな Oracle ストアドファンクションを使用して問題を解決できます。

```
CREATE OR REPLACE FUNCTION sum_employees(locID IN NUMBER)

        RETURN NUMBER

AS

        sum_emp NUMBER;

BEGIN

        SELECT sum(employee) INTO sum_emp
        FROM employee_table
        CONNECT BY PRIOR id = parent
        START WITH id = locID;

        RETURN sum_emp;

END;

CREATE OR REPLACE FUNCTION regional_employees (locID IN NUMBER)

        RETURN types.ref_cursor

AS

        result_cursor types.ref_cursor;

BEGIN

        OPEN result_cursor FOR
                SELECT id, name, sumEmployees(id) AS Employees
                FROM employee_table
                CONNECT BY prior id = parent
                START WITH id = locID;
```

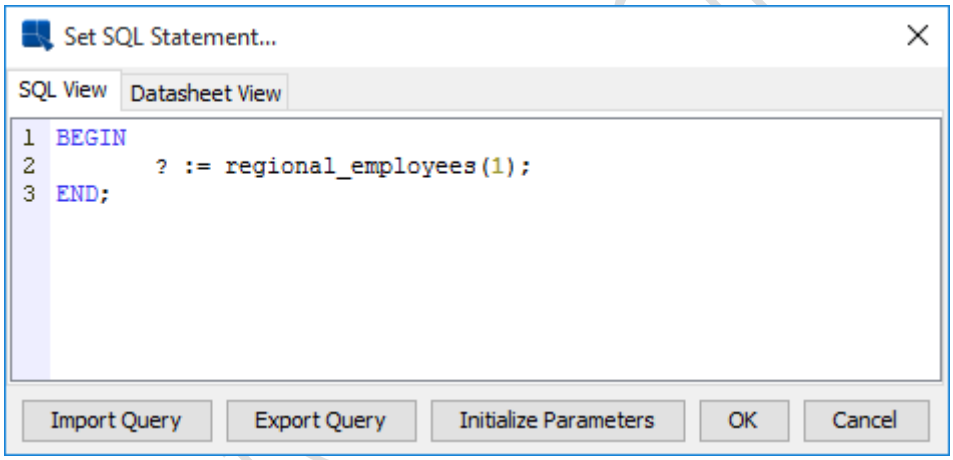
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        RETURN result_cursor;

END;
```

関数 **sum_employees** は開始ノードを引数としてとり、そのノードの子孫であるすべてのリーフノードの合計を求めます。例えば、**sum_employees(3)**は、ヨーロッパに 25 人の従業員がいるため（ロンドンでは 14 人、パリでは 11 人）、25 を返します。2 番目の関数、**regional_employees** は、**locID** で始まるツリー構造を走査し、**sum_employees** 関数の **ID**、**Name**、および結果から結果セットを構築します。結果セットは **REF CURSOR** を介して返されます。

引数を必要とするストアドファンクションを呼び出すには、Set SQL Statement ウィンドウに次の文を入力します。



regional_employees 関数の呼び出し

Datasheet View タブをクリックして結果をプレビューします。

SQL View | Datasheet View

ID	NAME	EMPLOYEES
1	All	87
2	America	62
4	New York	20
5	Santa Clara	30
6	Dallas	12
3	Europe	25
7	London	14
8	Paris	11

regional_employees の結果セット

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

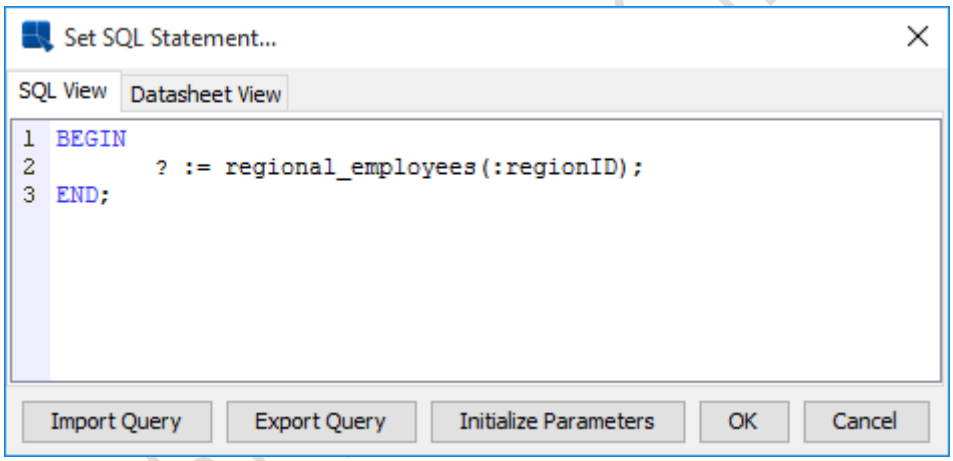
結果に見られるように、**CONNECT BY** 項は、ヨーロッパのノードをリストする前に、アメリカのノードを再帰的にリストするツリーを横断します。ユーザがヨーロッパの場所のみに注目する場合は、パラメータに **3** を入力すると、次の結果セットが返されます。

SQL View | Datasheet View

ID	NAME	EMPLOYEES
3	Europe	25
7	London	14
8	Paris	11

ヨーロッパの regional_employees の結果セット

パラメータ化されたレポートを作成するには、**:param_name** 構文を使用します。EspressReport の SQL パーサは、パラメータに使用されるコロンと代入演算子（**:=**)に使用されるコロンを区別することができます。パラメータの使用例を次に示します。



パラメータを使用した Oracle ストアドファンクションの呼び出し

IN パラメータを使用する場合は、クエリを実行する前にパラメータを初期化する必要があります。パラメータを既存の列にマップするため、ストアドプロシージャを実行するための正しいデフォルトデータ型を設定することは特に重要です。パラメータの初期化の詳細については、パラメータ化されたクエリを参照してください。この例を試すために、**<EspressReportInstallDir>/help/examples/data/locationHierarchyExample.sql** には、**employee_table** と 2 つのストアドファンクションを作成する SQL コマンドが含まれています。

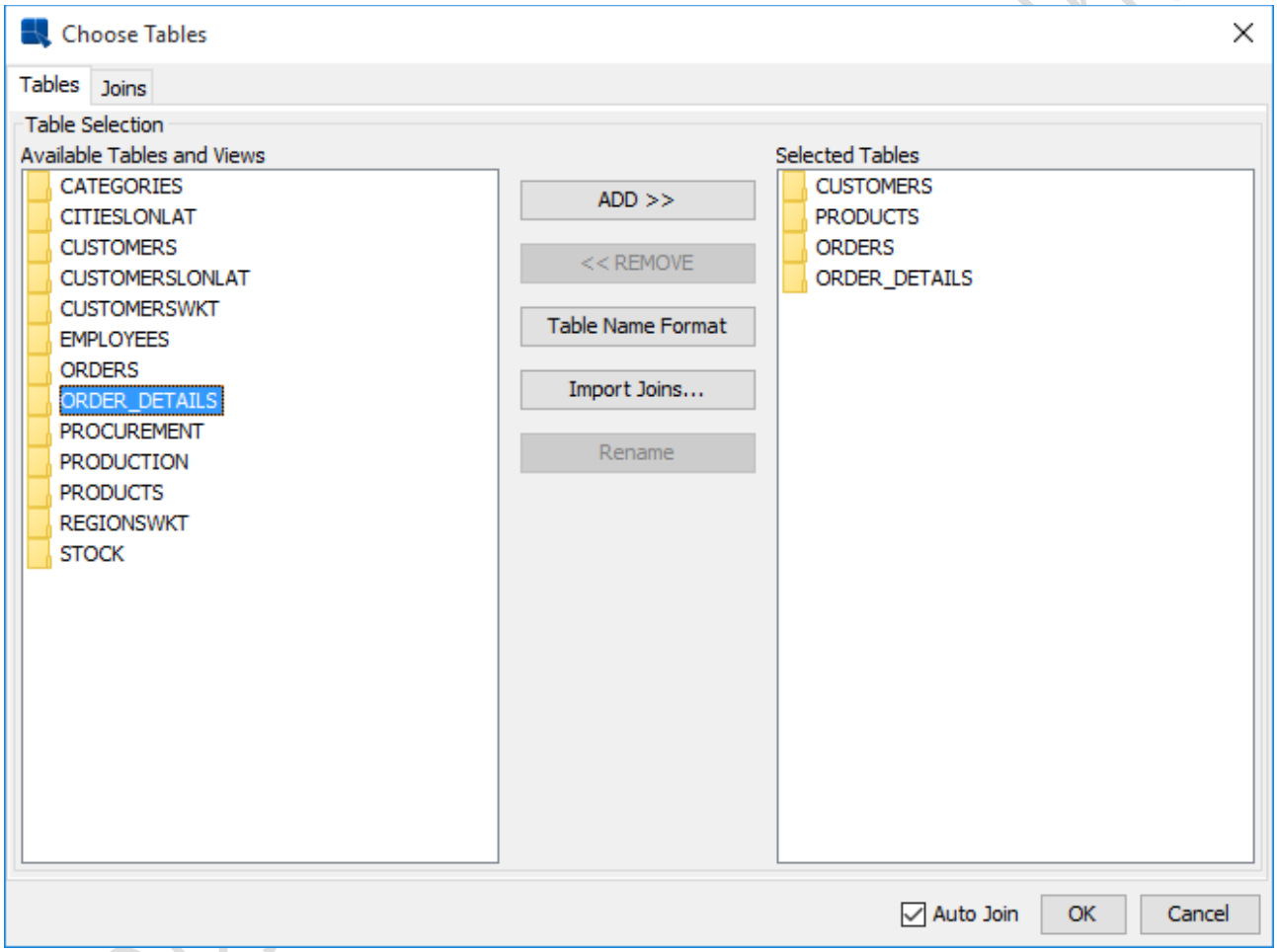
3.3.2.3 データビュー

EspressReport には、クエリインタフェースに加えて、データベースのデータ（データビュー）を取得する別の方法があります。データビューでは、クエリビルダを使用せずにフィールドを選択するか、基礎となるデータベース構造についての知識がある場合に、ユーザがクエリを設計できるデータベースの簡略化されたビューが提供されます。管理者は、データビューを使用して、テーブル、結合およびフィールドを事前定義し、ユーザが選択するためのローカルスキーマを作成できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

たとえば、管理者は営業部門のデータビューを設定できます。適切なデータベースのテーブルおよびフィールドは、ビジネスユーザのロジックと一致するようにあらかじめ選択し、グループ化できます。たとえば、invoices というグループには、適切な Customers フィールドと Order フィールドがあります。エンドユーザはこのデータビューを選択し、関連するフィールドを選択し、日付範囲を指定してから、レポートの設計を開始します。

データビューを作成するには、Data Source Manager ウィンドウで **Data Views** ノードを選択し、**Add** ボタンをクリックします。新しいウィンドウが開き、データビューに使用するデータベーステーブルを選択できます。

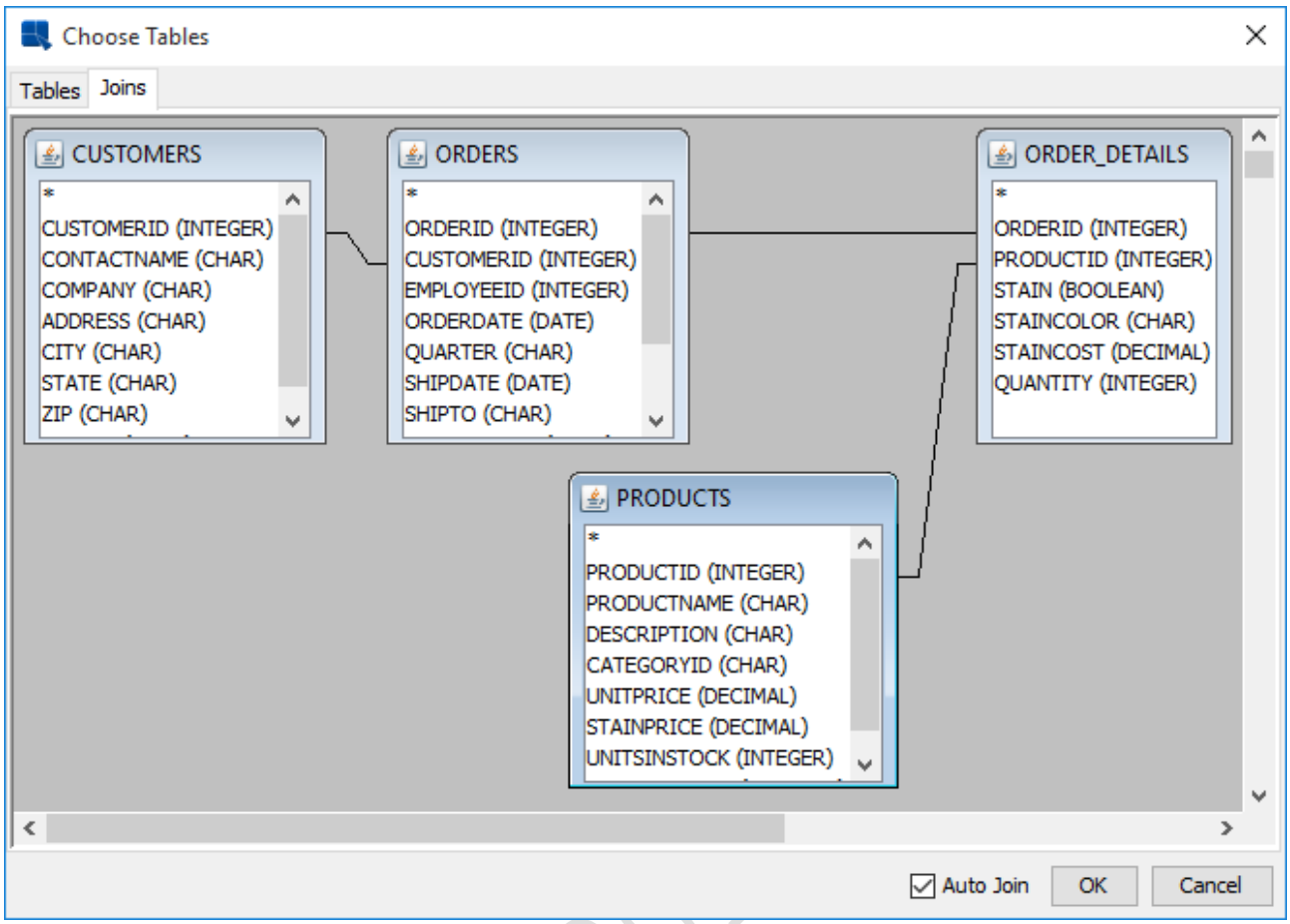


Data View Choose Tables ダイアログ

左ウィンドウには、使用可能なすべてのデータベーステーブルとビューが含まれています。左側のウィンドウでテーブルを選択し、**ADD** >>ボタンをクリックしてテーブルを追加できます。デフォルトでは、データビューは、データベース接続の設定時に指定した名前形式を使用します。**Table Name Format** ボタンをクリックして名前を変更するか、**Rename** ボタンをクリックしてテーブルのエイリアスを指定します。**Import Joins...**ボタンをクリックして、選択したテーブルと結合を別のデータビューからインポートすることもできます。

このウィンドウの **Joins** タブでは、選択したテーブル間の結合を指定できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



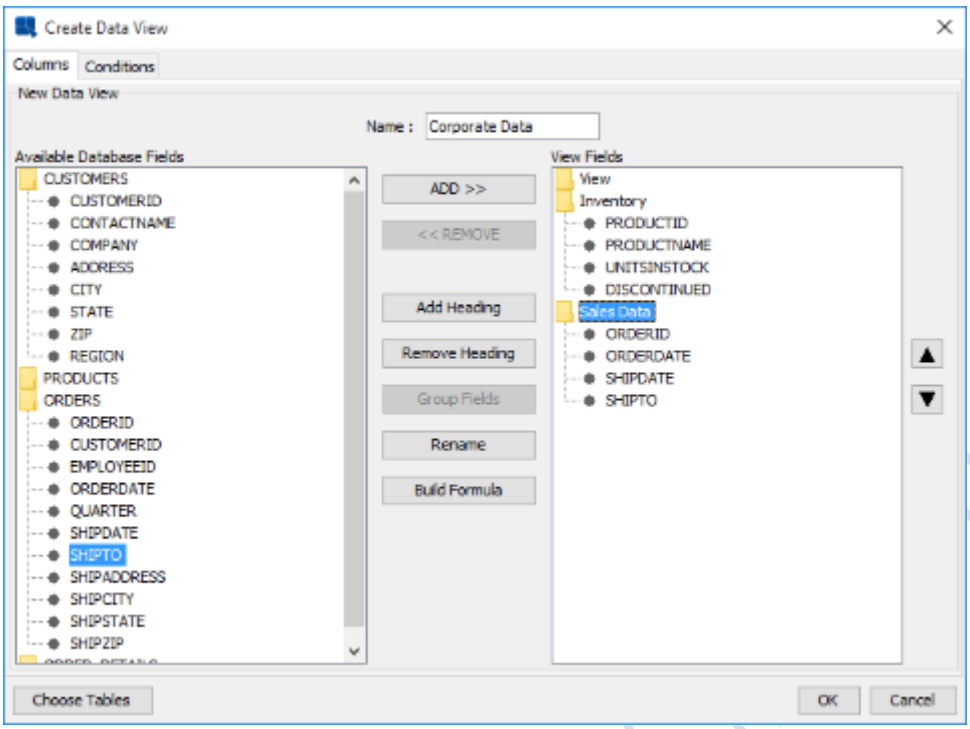
Data View Joins ダイアログ

Joins タブには、選択したすべてのテーブルとその関連フィールドが表示されます。テーブルは、データベース接続の設定時に選択したオプションに応じて自動結合されます。これらの自動結合は、行で表される表間の標準結合を作成します。結合を削除するには、その **Join** を右クリックし、ポップアップメニューから選択します。結合を追加するには、1 つの列フィールドをクリックして別のテーブルの別の列フィールドにドラッグします。その後、結合が表示されます。データビューはクエリビルダと同じ結合プロパティを使用します。結合プロパティの詳細については、Joins を参照してください。

テーブルの選択と結合が完了したら、**OK** ボタンをクリックすると、新しいウィンドウが開き、データビューを作成できます。

Copyright(C) 2018

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Create Data View ダイアログ

左側のウィンドウには、選択したテーブルのリストとその関連フィールドが表示されます。各フォルダはテーブルを表し、ダブルクリックで開閉できます。右側のウィンドウには、データビュー用に選択されたフィールドが含まれています。フィールドをデータビューに追加するには、左側のウィンドウでフィールドを選択し、**ADD** >>ボタンをクリックします。フィールドは、右ウィンドウのフィールドを選択し、<< **REMOVE** ボタンをクリックすることで、同じ方法でデータビューから削除することができます。計算式の列を作成するには、**Build Formula** ボタンをクリックします。式ビルダが開き、列を作成できます。別名を定義するには、右側のウィンドウのいずれかのビューフィールドを選択し、**Rename** ボタンをクリックします。

ヘッダーを追加することで、データビュー内のフィールドをグループ化することもできます。これにより、1 つの見出しの下にある異なるデータベーステーブルのデータをグループ化する、独自の仮想テーブルの組織構造を作成できます。見出しを作成するには、**Add Heading** ボタンをクリックします。見出しの名前を指定するよう求められます。新しい見出しは、右のウィンドウにフォルダとして表示されます。見出しの下にフィールドを追加するには、まず右側のウィンドウから追加するフィールドを選択し、**Group Fields** ボタンをクリックします。ドロップダウンメニューが表示され、フィールドを追加する見出しを選択することができます。

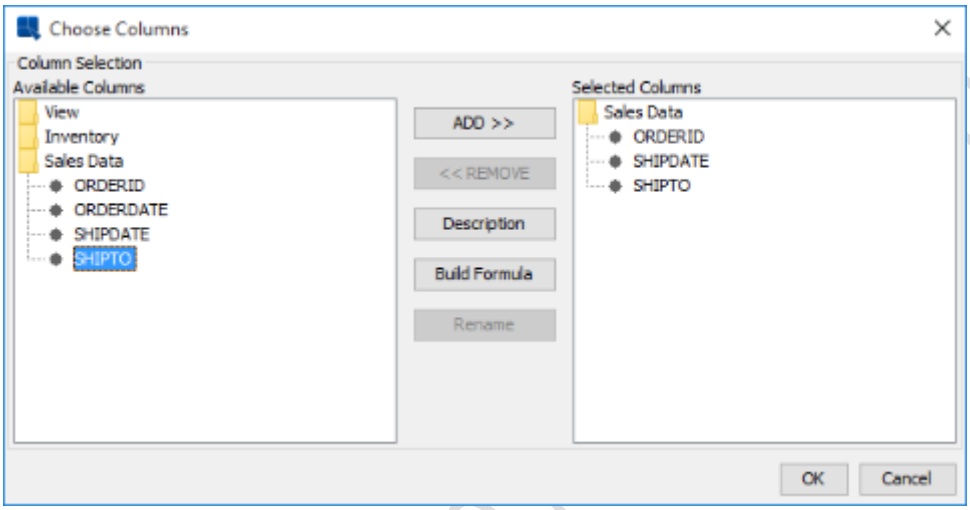
Conditions タブには、エンドユーザの特定のフィルタリング基準を指定できる式ビルダウィンドウが含まれています。このウィンドウに追加されたものは、生成された SQL の **Where** 項に追加されます。式ビルダの使用の詳細については、Columns を参照してください。

データビューの作成が終了したら、**OK** ボタンをクリックすると、データビューが Data Source Manager に追加されます。ユーザはこのビューを使用してアドホックなクエリを作成できるようになります。

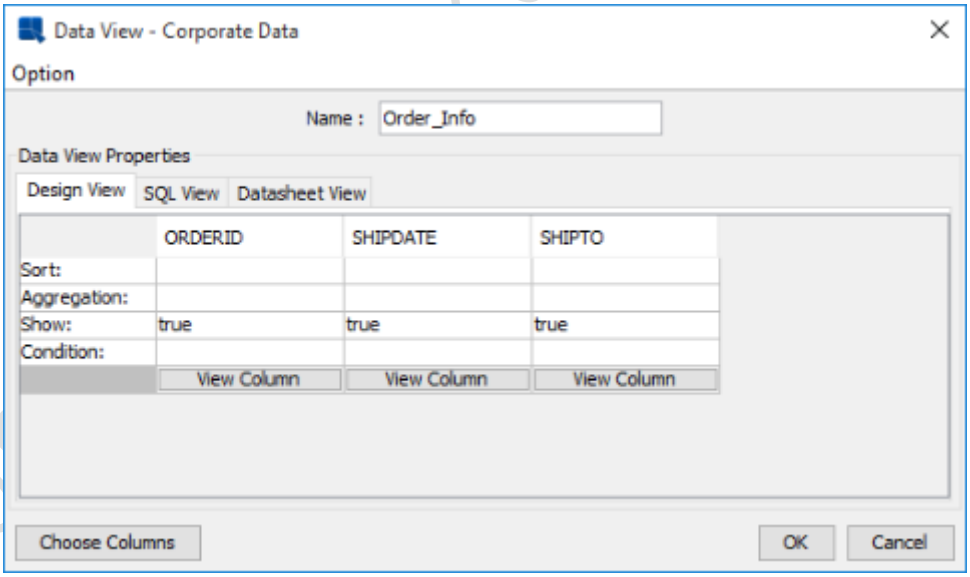
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

データビューをデータソースとして使用してグラフをデザインする場合（データビューを選択して **Next** ボタンをクリックすると)、ビューに使用するフィールドをチャートに選択できるウィンドウが開きます。このダイアログでは、使用可能なビュー列に基づいて計算フィールドを作成することもできます。

フィールドを選択したら、**OK** ボタンをクリックすると、新しいウィンドウが開き、ソート、集約、およびデータビューのフィルタ条件を指定できます。



Data View Choose Fields ダイアログ



Data View Conditions ウィンドウ

それぞれのフィールドをダブルクリックすることにより、データビューの各フィールドのソート、集約、および条件を指定することができます。ソートと集約はドロップダウンメニューから選択できます。**Conditions** フィールドをダブルクリックすると、**>、<、=、between** のような単純な選択基準を指定できる新しいウィンドウが表示されます。**Conditions** フィールドを右クリックし、ポップアップメニューから **Build** を選択すると、より高度なフィルタリング基準を作成できます。式ビルダウィンドウが開き、条件を作成できます。**View Column** ボタンをダブルクリックして、カラム内のすべての固有値

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

を表示することもできます。

条件ウィンドウの左上隅にある Option メニューでは、条件ウィンドウの垂直/水平ビューを選択したり、データビューでパラメータを初期化したり、クエリを保存したりできます。

選択セットと指定した条件は、名前フィールドに指定した名前のデータビュークエリとして保存されます。データビューのクエリは、データビューのノードの下に保存されます。データビューから作成されたチャートは、更新/変更のためのデータビュークエリを参照します。

3.3.2.3.1 データビューのパラメータ

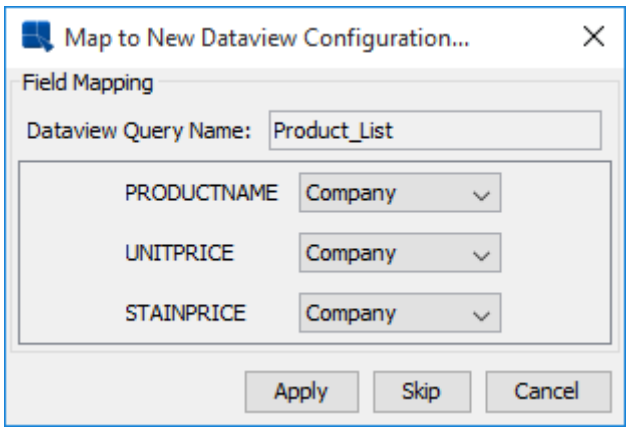
クエリビルダと同様に、ユーザはデータビューでクエリパラメータを指定できます。データビューにパラメータを追加するには、Data Source Manager でデータビューを選択し、**View** をクリックしてデータビューを実行します。データビューのフィールドを選択し、条件ウィンドウに入ったら、列の **Condition** フィールドを右クリックし、ポップアップメニューから **Build** を選択します。これにより、式ビルダが表示され、クエリビルダと同じ方法でパラメータを指定できます。詳細は、パラメータ化されたクエリを参照してください。

パラメータを入力すると、**Datasheet View** タブに移動してから **OK** をクリックしてチャートウィザードを続行するか、クエリとして選択内容を保存するかどうかを選択するプロンプトが表示されます。また、Option メニューから **Initialize Parameters** を選択して、パラメータを初期化することもできます。

3.3.2.3.2 データビューのクエリ更新

場合によっては、データモデルまたは要件の変更に応じて、データビューの構造/構成を変更する必要が生じることがあります。変更には、フィールドの追加/削除、名前の変更などがあります。Data Source Manager で変更を選択し、Update メニューから **Data View Queries** を選択すると、データビューから関連するクエリに変更を反映できます。

ビューに関連付けられたすべてのクエリがスキャンされ、矛盾した更新するフィールドまたはフィールド名が表示されます。



Update Query Fields ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

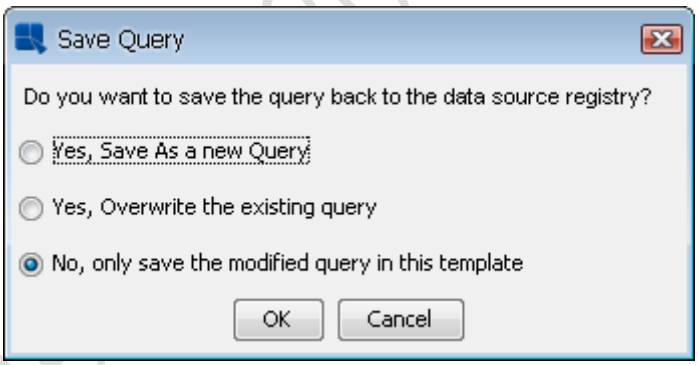
各クエリについて、データビュー構造と一致しなくなったフィールドを変更するかどうかを確認するメッセージが表示されます。各フィールドについて、データビューからフィールドを選択してクエリにマップしたり、クエリからフィールドを削除したりすることができます。クエリで何も変更しない場合は、**Skip** ボタンをクリックします。クエリは引き続き実行されますが、古いデータビュー構造を参照します。**Apply** ボタンをクリックして、データビューのクエリに変更を保存します。

3.3.2.4 クエリの編集

クエリビルダでクエリを設計したり、SQL 文を作成したり、データビューを実行したりして、データベースデータを使用してレポートを作成することを選択した場合は、Data Source Manager に戻らずに ReportDesigner から直接クエリを変更できます。

レポートのクエリを変更するには、Report Designer の **Data** メニューから **Modify Query** を選択するか、ツールバーの **Modify Query** ボタンをクリックします。クエリビルダでクエリを設計した場合は、クエリビルダインタフェースが再度開き、クエリを変更できます。SQL 文を入力すると、SQL を変更できるテキストボックスが開きます。データビューを使用した場合は、データビューの条件ウィンドウが再び開き、フィルタの変更や追加フィールドの選択が可能になります。

すべての変更を指定すると、データレジストリでクエリを変更したり、データレジストリに新しいクエリを保存したり、テンプレート内のクエリのみを変更したりするオプションが提供されます。



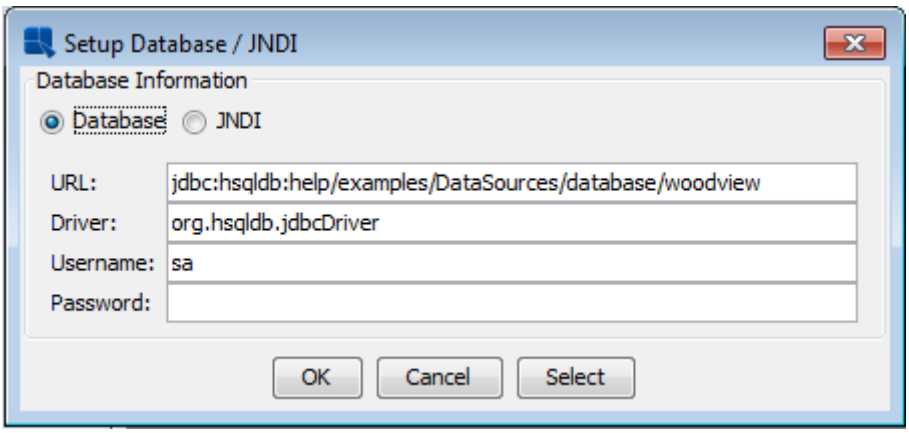
Saving Query Options

保存オプションを指定すると、変更されたクエリがレポートに適用されます。クエリを大幅に変更した場合は、レポートのデータマッピングを再度実行する必要があります。データマッピングの詳細については、レポートの種類とデータのマッピングを参照してください。

3.3.2.5 データベース接続の編集

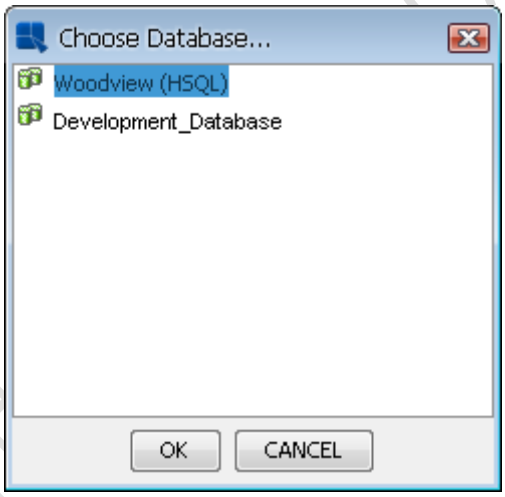
データベースデータを使用してチャートを作成することを選択した場合は、チャートデザイナ内からテンプレートのデータベース接続情報を直接編集することもできます。チャートの接続情報を変更するには、**Data** > **Modify Database** を選択します。これにより、データベースに接続するときに使用するレポートの別の URL、ドライバ、ユーザ名、およびパスワードを指定できるダイアログが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



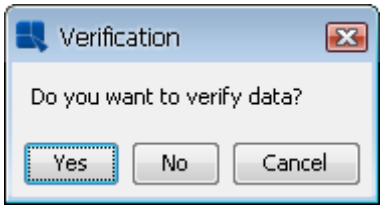
Change Database Connection ダイアログ

データベース情報を手動で入力するだけでなく、データ接続からデータベース接続情報を取得することもできます。これを行うには、Database Connection ダイアログで **Select** ボタンをクリックします。これにより、データベース接続をプルする XML レジストリファイルを参照することができます。レジストリファイルを選択すると、レジストリに定義されているデータベースの一覧が表示されます。



レジストリからデータベースを選択

使用するデータベースを選択し、**OK** ボタンをクリックします。そのデータベースの接続情報が自動的に接続ダイアログに適用されます。テンプレートの接続情報を設定したら、**OK** ボタンをクリックして変更を適用します。新しい接続情報を確認するかどうかを尋ねるダイアログが表示されます。



Verify Connection ダイアログ

データソースが存在しないことが分かっていない限り、提供された接続情報が正しいことを確認するのが一般的です。レポート内のデータベース接続情報を変更する場合、同じ接続情報を使用するレポート

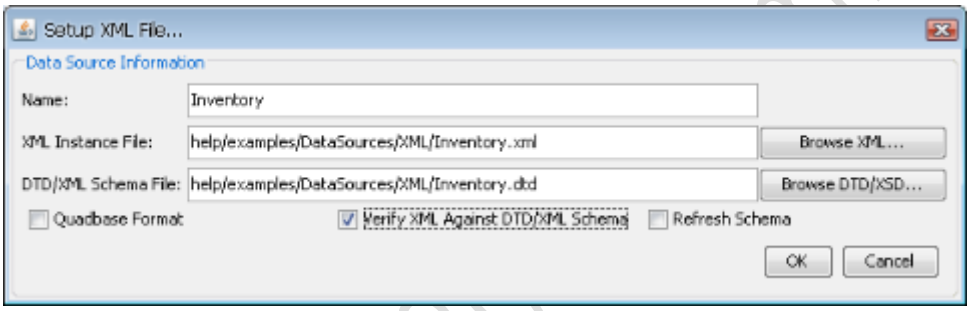
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

内のすべてのサブレポート、チャート、およびドリルダウンレイヤーも自動的に更新されます。

クエリの変更機能とは異なり、テンプレートのデータベース接続の変更はテンプレートにのみ保存されます。データレジストリには保存されません。

3.3.3 XML および XBRL ファイルのデータ

リレーショナルデータベースに加えて、EspressReport ではデータの検索や XML ファイルのクエリが可能です。XML データは事実上どんなフォーマットでも構いませんが、XML データとともに DTD ファイルまたは XML スキーマ（XSD）を指定する必要があります。XML データソースを設定するには、データソースマネージャで XMLFiles ノードを選択し、**Add** ボタンをクリックします。新しい XML ソースのオプションを指定するダイアログが表示されます。



Setup XML Data Source ダイアログ

Name では、XML データソースの表示名を指定できます。**XML Instance File** では、データを取得する XML ファイルの場所を指定できます。このフィールドに XBRL ファイルを指定することもできます。ファイルの場所として適切な URL を追加することで、ここでも HTTP サーバから XML データを取得するデータソースを設定できます。**DTD/XML Schema File** では、XML ファイルの有効な DTD または XML スキーマファイルの場所を指定できます。

Quadbase Format チェックボックスを使用すると、XML ファイルが EspressReport からの XML エクスポートの形式であるかどうかを指定できます。たとえば、チャートのデータを XML 形式でエクスポートすることを選択した場合、この形式を使用してチャートのデータを読み込むことができます。XML へのエクスポートの詳細については、チャートのエクスポートを参照してください。この形式でファイルを使用する場合は、DTD/XML スキーマを指定する必要はありません。

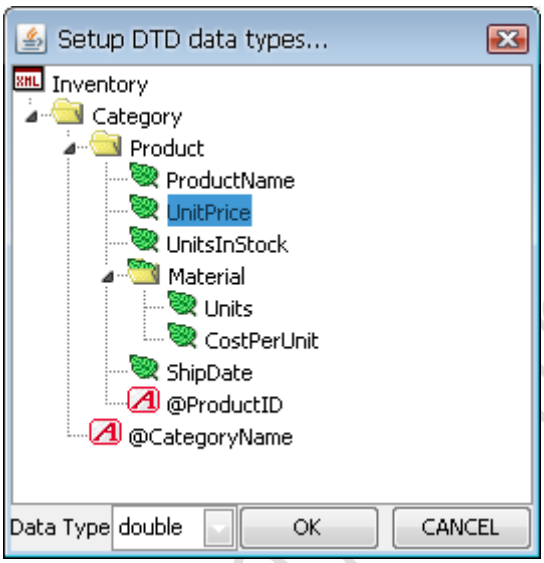
Verify XML against DTD/XML Schema チェックボックスは、指定された XML ファイル/ソースが DTD/XML スキーマファイルで指定されたレイアウトに準拠していることを確認します。クエリは DTD/XML スキーマファイルの構造に基づいて設計されているため、非準拠の XML ソースは予期しない結果を招く可能性があります。XML が DTD/XML スキーマに準拠していない場合は、警告が表示されます。ただし、データソースの設定は続行できます。

Refresh Schema チェックボックスは、スキーマまたは DTD 定義をリロードして、XML データソース

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

内の構造の変更をレジストリに組み込みます。このオプションは、データソースが最初に作成されてから DTD またはスキーマ定義が変更された場合にのみ必要です。

XML ファイルと DTD/XML スキーマの設定が完了すると、XML データソース内の選択可能なすべての要素のデータ型を指定できる新しいダイアログが開きます。ダイアログは、DTD ファイルまたは XML スキーマのどちらを使用しているかによって異なります。



DTD Data Type Selection ダイアログ

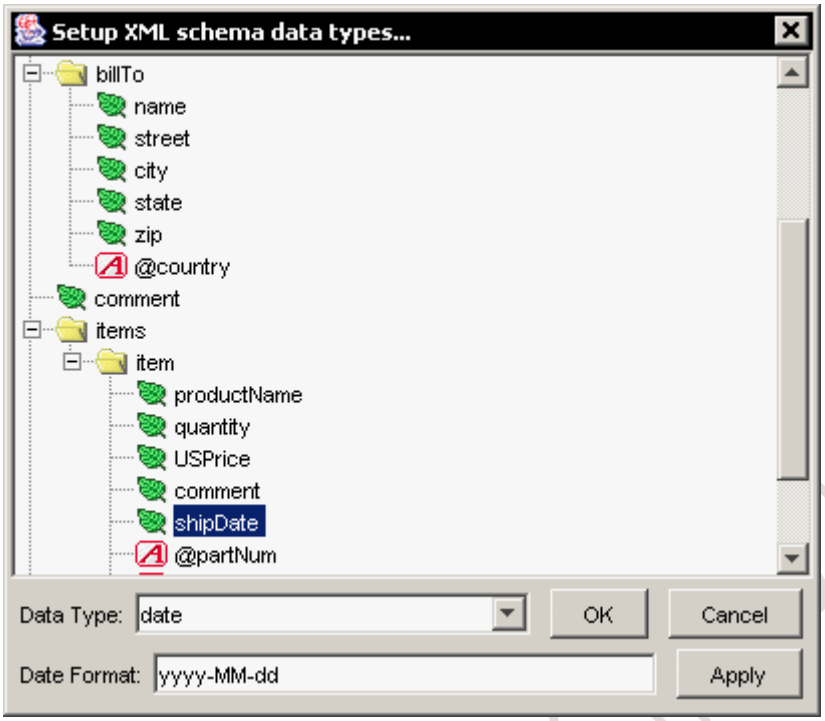
DTD ファイルでは要素の正しいデータ型が指定されていないため、すべての要素はデフォルトで String と見なされます。要素のデータ型を変更するには、要素を選択し、ダイアログの下部にあるドロップダウンウィンドウからデータ型を選択する必要があります。XML ファイルを照会するときに適切な結果を得るには、選択可能なすべての要素のデータタイプを設定する必要があります。これには、葉ノード、データを含む親ノード、および属性要素が含まれます。サポートされているデータ型は次のとおりです。

- String
- Integer
- Double
- Date(日付をデータ型として指定する場合は、日付形式も指定する必要があります。)
- Boolean

データタイプの指定が完了したら、**OK** ボタンをクリックします。XML ソースが Data Source Manager に追加されます。

XML スキーマを使用している場合は、別のデータ型ダイアログが開きます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



XML Schema Data Type Selection ダイアログ

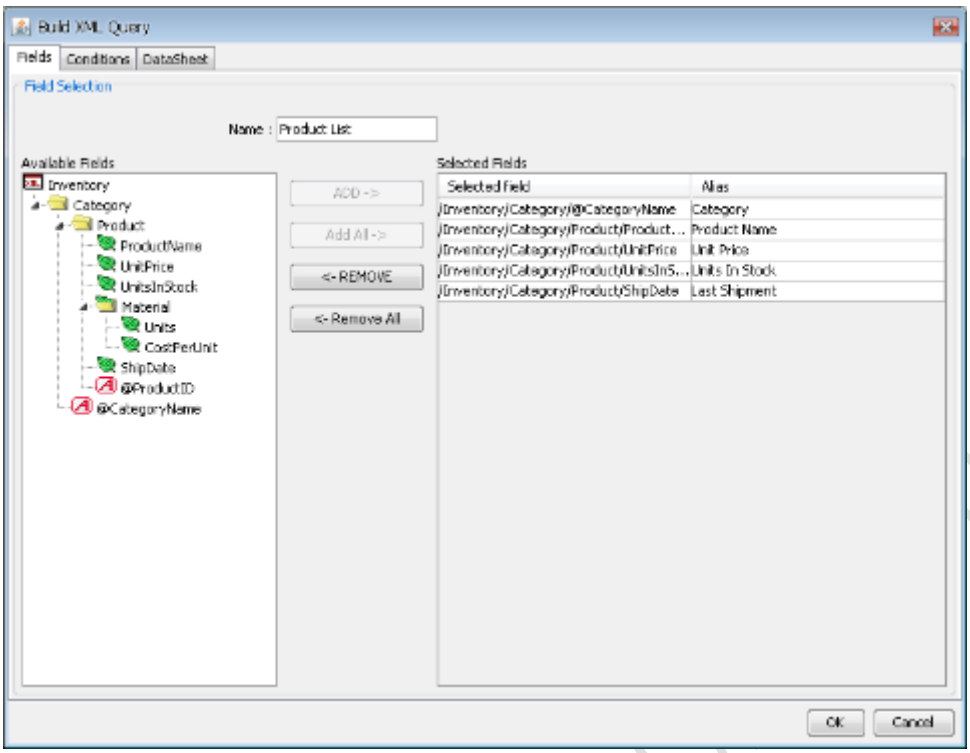
一般に、データ型は XML スキーマファイルで既に定義されているはずですが、このダイアログでは変更を加えることができます。データタイプの指定が完了したら、**OK** ボタンをクリックします。XML ソースが Data Source Manager に追加されます。

3.3.3.1 XML クエリ

XML データソースを設定したら、ノードを選択するクエリを作成し、フィルタ条件を指定し、ツリー構造を EspressReport で使用される表形式に変換することができます。クエリを追加するには、XML ソースのノードを選択し、**Add** ボタンをクリックします。これにより、クエリを作成するための XML クエリビルダインタフェースが起動します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



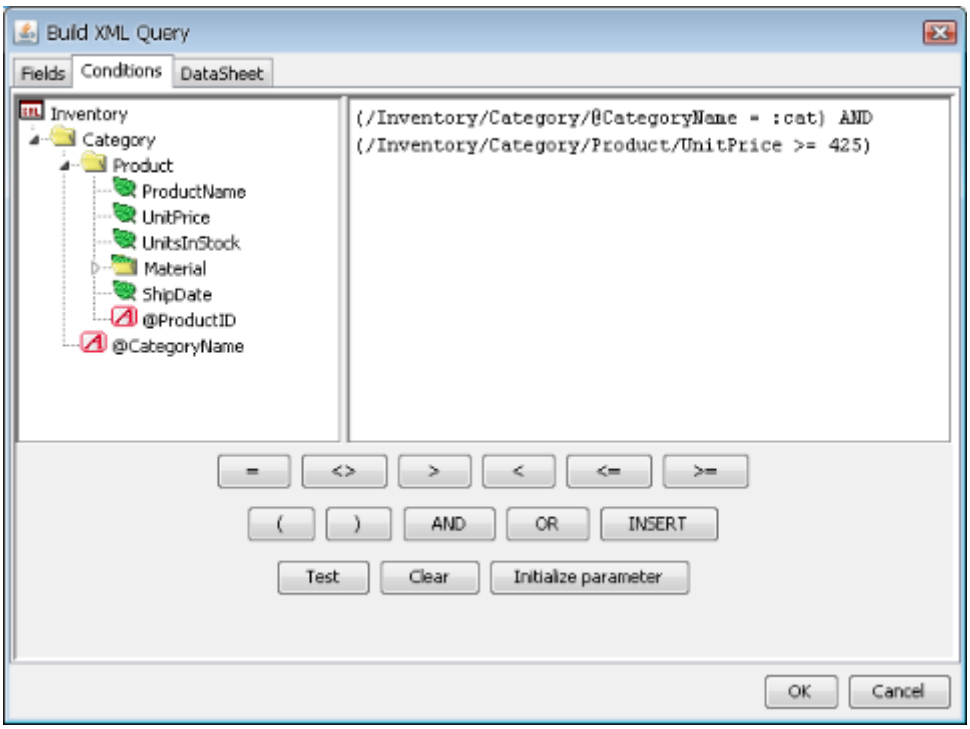
XML Query Field Selection タブ

XML クエリビルダの **Fields** タブでは、チャートで使用する XML ファイルからフィールド/ノードを選択できます。ウィンドウの左側には、DTD または XML スキーマファイルのツリー構造が含まれています。**Add** ボタンをクリックすると、選択可能な要素を選択してクエリに追加できます。選択したフィールドがウィンドウの右側に表示されます。列の **Alias** フィールドをダブルクリックし、新しい列エイリアスを入力することで、任意のフィールドのエイリアスを指定できます。

選択された各要素はチャート内の列になります。1 対 1 の関係を判別できない結果の場合、表構造はデータ内の使用可能なすべての順列を使用して作成されます（SQL のクロス結合に似ています）。最良の結果を得るには、明確な階層関係が存在するクエリのフィールドを選択することをお勧めします。

XML クエリビルダの **Conditions** タブでは、選択にいくつかの基本的なフィルタリング基準を指定できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



XML Query Conditions タブ

XML ファイル内の任意の選択可能な要素に対して、等しい、等しくない、超過、未満、以下、以上の条件を指定できます。AND 演算子または OR 演算子を使用して複合条件を作成することもできます。フィールドは、XML ツリーの下の直接パスを使用して指定されます。現在、ダイレクトパスのみがサポートされています。複雑な XPath 式は使用できません。フィールドを追加するには、左側のフィールドをダブルクリックするか、フィールドを選択して **Insert** ボタンをクリックします。

条件の記述が終了したら、**Test** ボタンをクリックして構文が正しいことを確認します。

DataSheet タブでは、クエリ結果をプレビューして、XML データがどのように表形式に変換されるかを確認できます。クエリビルダと同じ方法で結果セットをナビゲートすることができます (クエリ出力)。

フィールドを選択し、適切な条件を指定したら、**OK** ボタンをクリックします。クエリが、Data Source Manager の XML ソースの下に新しいノードとして追加され、チャートを作成するために使用できるようになりました。

EspressReport のインストールにはサンプルの XML ファイルとサンプルの **DTD/XSD** が含まれています。これらのファイルは、インストールの **help/examples/DataSources/XML** ディレクトリにあり、**Inventory.xml** と **Inventory.dtd** と呼ばれます。XML データを Chart Designer にストリームするためのサンプルサーブレットもあります。サーブレットのコードと説明は、**help/examples/DataSources/XML/servlet** ディレクトリにあります。

3.3.3.1.1 XML パラメータ

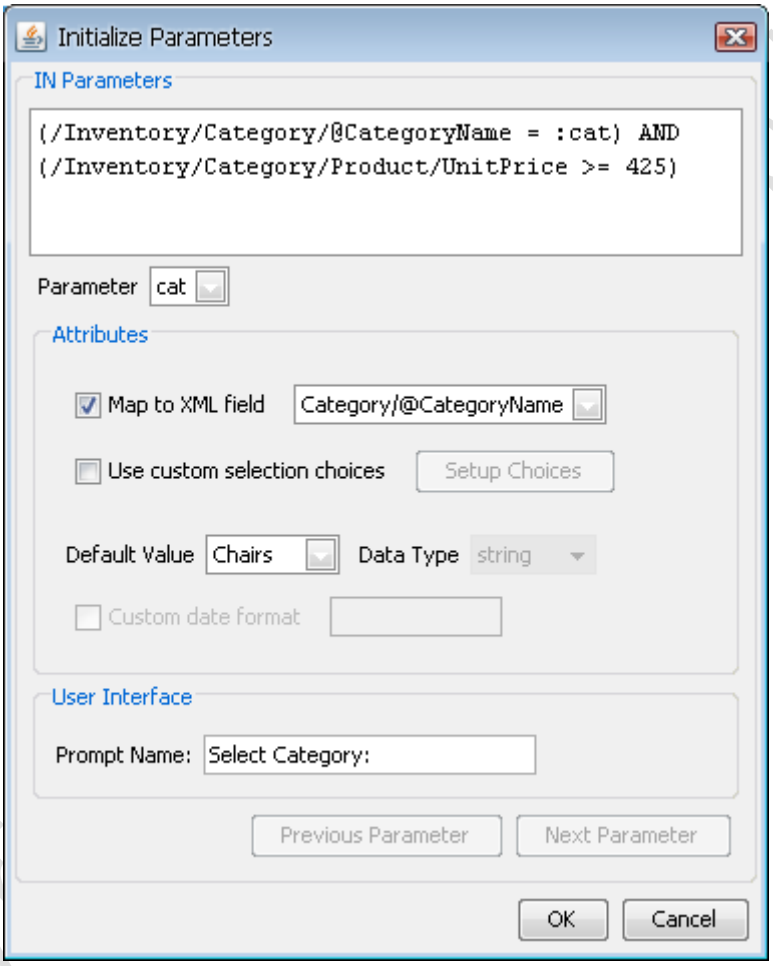
データベースクエリと同様に、XML クエリのパラメータを指定することもできます。同じ構文**":"**は、XML 条件のパラメータをクエリ条件の中に表示するために使用されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

したがって、次の XML 条件では CategoryName 属性のクエリに動的フィルタが配置されます。

```
/Inventory/Category/@CategoryName = :category
```

XML パラメータは、クエリパラメータと同じ方法で初期化されます。クエリをプレビューまたは閉じる、または **Initialize Parameters** ボタンをクリックしてトリガすると、初期化ダイアログが表示されます。唯一の違いは、データベースフィールドにマッピングする代わりに、パラメータプロンプトを XML ファイルのノードにマップできることです。

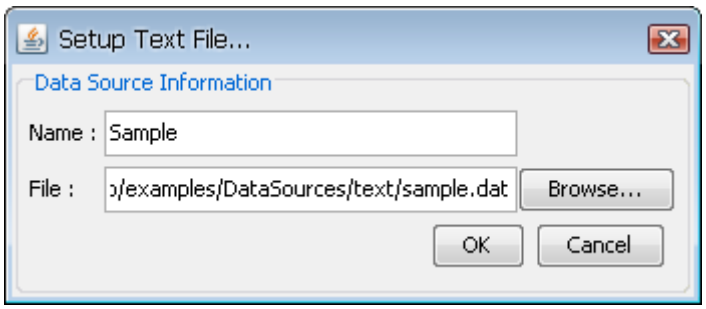


XML Parameter Initialization ダイアログ

3.3.4 テキストファイルのデータ

EspressReport では、フラットテキストファイルからデータを取得することもできます。テキストファイルをデータソースとして追加するには、Data Source Manager で **TXTFiles** ノードを選択し、**Add** ボタンをクリックします。表示名と使用するテキストファイルの場所を指定するダイアログが開きます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Add Text Data Source ダイアログ

テキストファイルは URL から取得することもできます。これを行うには、**File** テキストフィールドに URL を入力します。プロトコルなどを含む完全な URL を入力する必要があります（例：http://www.quadbase.com/textfile.txt）。

情報を指定したら、**OK** ボタンをクリックすると、Data Source Manager ウィンドウの **TXTFiles** ノードの下にテキストソースが表示されます。

3.3.4.1 テキストファイルのフォーマット要件

EspressReport で読むことができるように、テキストファイル内のデータには特定のフォーマット要件があります。一般的に、データは次のような形式になります。

```
String,date,decimal
Name,day,Volume
"John","1997-10-3",32.3
"John","1997-4-3",20.2
"Mary","1997-9-3",10.2
"Mary","1997-10-04",18.6
```

上記のデータファイルはプレーンテキストファイルです。最初の行はデータ型を指定し、2 行目はフィールド名を指定します。3 行目以降がレコードです。すべてのテキストファイルは、これら 3 つの部分で構成されている必要があります。サンプル・データ・ファイルにはそれぞれ 3 つのフィールドがある 4 つのレコードがあります。フィールド間の区切り文字は、",",";",または" "（カンマ、セミコロン、またはスペース）のいずれかの文字です。各フィールドは引用符で囲むことができます(1 つまたは 2 つ)。

テキストデータファイルでは、キーワードを使用してデータ型が指定されます。次に、認識されるキーワードと、対応する JDBC タイプと Java タイプのリストを示します。

Data File キーワード (Not Case Sensitive)	JDBC の種類	Java Type in EspressReport
Boolean, logical, bit	BIT	Boolean
tinyint	TINYINT	byte
smallint, short	SMALLINT	short

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

int, integer	INTEGER	int
long, bigint	BIGINT	long
float	FLOAT	double
real	REAL	float
double	DOUBLE	double
numeric	NUMERIC	java.math.BigDecimal
decimal	DECIMAL	java.math.BigDecimal
date	DATE	java,sql,Date
time	TIME	java,sql,Time
timestamp	TIMESTAMP	java,sql,Timestamp
string	CHAR	String
varchar	VARCHAR	String
longvarchar	LONGVARCHAR	String

特定のデータ型では、テキストファイル内のデータを特定の形式で表示する必要があります。特定の書式設定が必要なデータ型の一覧を次に示します:

データの種類	形式	例
Date	yyyy-mm-dd or yyyy-mm	2001-06-12 or 2000-06
Time	hh:mm:ss	12:17:34
Timestamp	yyyy-mm-dd hh:mm:ss	2001-06-12 12:17:34
Boolean	true/false, t/f, 1/0 (case insensitive)	true

EspressReport インストールにはサンプルテキストファイルが含まれています。このファイルは、インストールの **help/examples/DataSources/text** ディレクトリにあり、**Sample.dat** と呼ばれています。

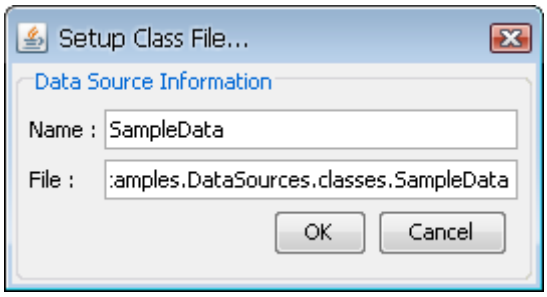
3.3.5 クラスファイルのデータ

柔軟性を最大限に高めるために、EspressReport では、Chart Designer に引数として渡すためのインタフェースを提供することにより、オブジェクトデータまたは配列データを使用してチャートをデザインできます。API を使用すると、**IDataSource** を実装して、JDBC 結果セットに使用される **java.sql.ResultSet** インタフェースと同様の **IResultSet** オブジェクトを返すことができます。ユーザは **IResultSet** の独自の実装を提供するか、EspressReport が提供するものを使用できます。

この機能の詳細については、カスタム実装で渡されたデータを参照してください。

クラスファイルをデータソースとして追加するには、Data Source Manager で **ClassFiles** ノードを選択し、**Add** ボタンをクリックします。表示名と、使用するクラスファイルの場所を指定するダイアログが開きます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Add Class File ダイアログ

インストールの **help/examples/DataSources/classes** ディレクトリにサンプルコードがあります。コンパイルされたコードは、データ配列を Chart Designer に渡すクラスファイルを生成します。クラスファイルをデータソースとして使用するには、クラスパスにパッケージを含むファイルまたはディレクトリ（Chart Designer を使用する場合は **EspressManager.bat** または **EspressManager.sh** ファイルの**-classpath** 引数）が必要です。

3.3.5.1 パラメータ化されたクラスファイル

EspressReport には、クラスファイルデータソースのコンテキスト内でチャートパラメータを定義することを可能にする API インタフェース **IParameterizedDataSource** が追加されています。

クラスファイルのコンテキスト内で定義されるパラメータは、クエリパラメータと同じ方法で動作します。パラメータ化されたクラスファイルのデータソースの設定の詳細については、カスタム実装で渡されたデータを参照してください。

3.3.6 EJB のデータ

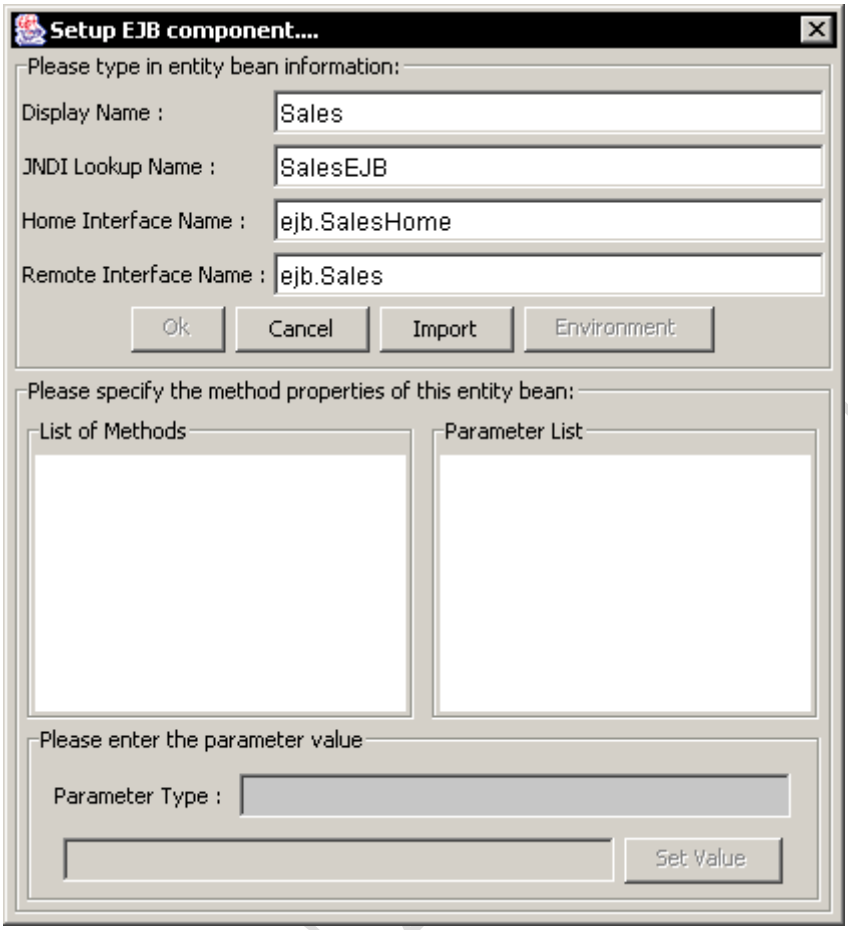
EnterpriseJavaBeans™テクノロジを使用することで、開発者は大規模なエンタープライズアプリケーションの開発を簡素化できます。EJB テクノロジでは、開発者はビジネスロジックを構築し、アプリケーションサーバ（EJB コンテナ）がすべてのシステムレベルのサービスを管理できるようにすることができます。

Java EE™環境で作業する場合、永続データインタフェースは Entity Bean によって提供されます。基礎となるストレージメカニズムはリレーショナルデータベースでも構いませんが、アプリケーションデータモデルは EJB であり、チャーティングツールで冗長なデータベース接続を行うことは望ましくない場合があります。このような状況では、EspressReport を使用して Entity Bean から直接データをクエリできます。

EJB をデータソースとして追加するには、まず EJB をアプリケーションサーバにデプロイし、適切なスタブクラスを含むクライアント JAR ファイルをクラスパス（または Chart Designer を使用するときは **EspressManager.bat** または **EspressManager.sh** ファイルの**-classpath** 引数）に追加する必要があります。Data Source Manager で **EJB** ノードを選択し、**Add** ボタンをクリックします。これにより、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

表示名と Bean の接続情報を指定できるダイアログが表示されます。

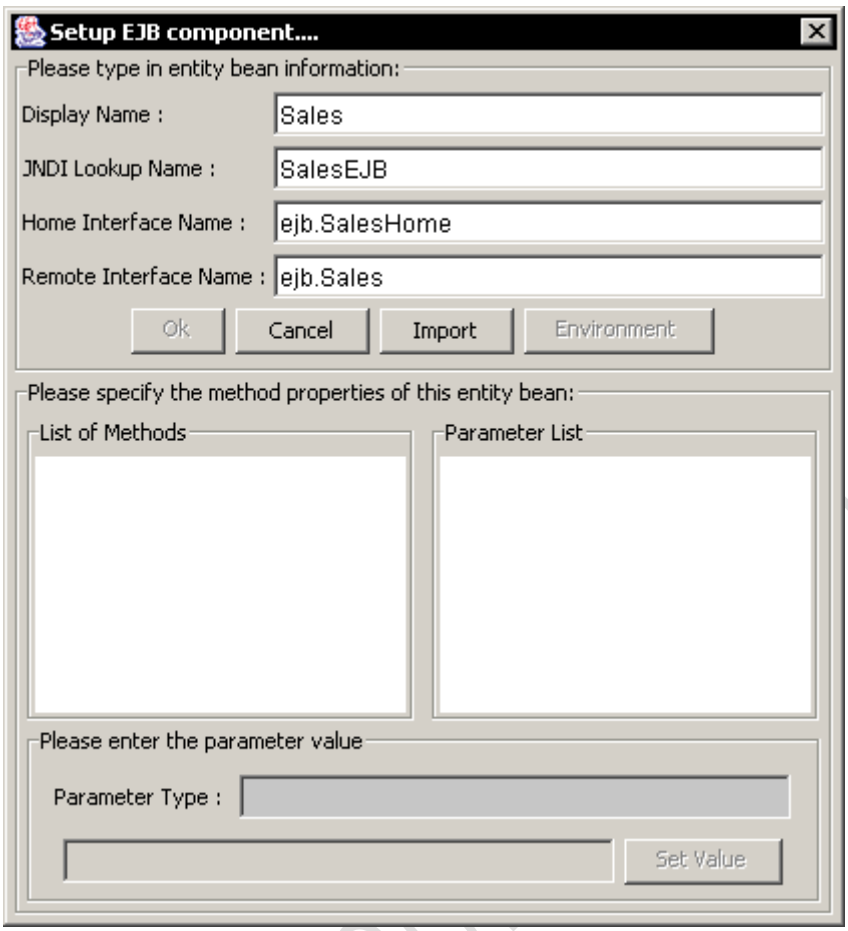


Add EJB ダイアログ

EJB データソースに接続するには、Bean の JNDI ルックアップ名を指定する必要があります（これは、Bean をデプロイするときに指定されます）。EJB 1.1 ユーザの場合は、ホームインタフェースとリモートインタフェースの名前を指定します。EJB 2.0 ユーザの場合は、ローカルホームインタフェースとローカルインタフェースを指定します。すべての情報を指定したら、**Import** ボタンをクリックして、ホームインタフェースを分析し、すべてのファインダーメソッドを取得します。見つかったメソッドはすべて、ダイアログの **List of Methods** セクションに表示されます。

同じダイアログを使用して、ファインダーメソッドに存在するパラメータに基づいて取得されるデータをフィルタリングすることができます。左側のダイアログでメソッドを選択すると、パラメータリストセクションにすべてのパラメータが表示されます。次に、パラメータをクリックして値を設定します。

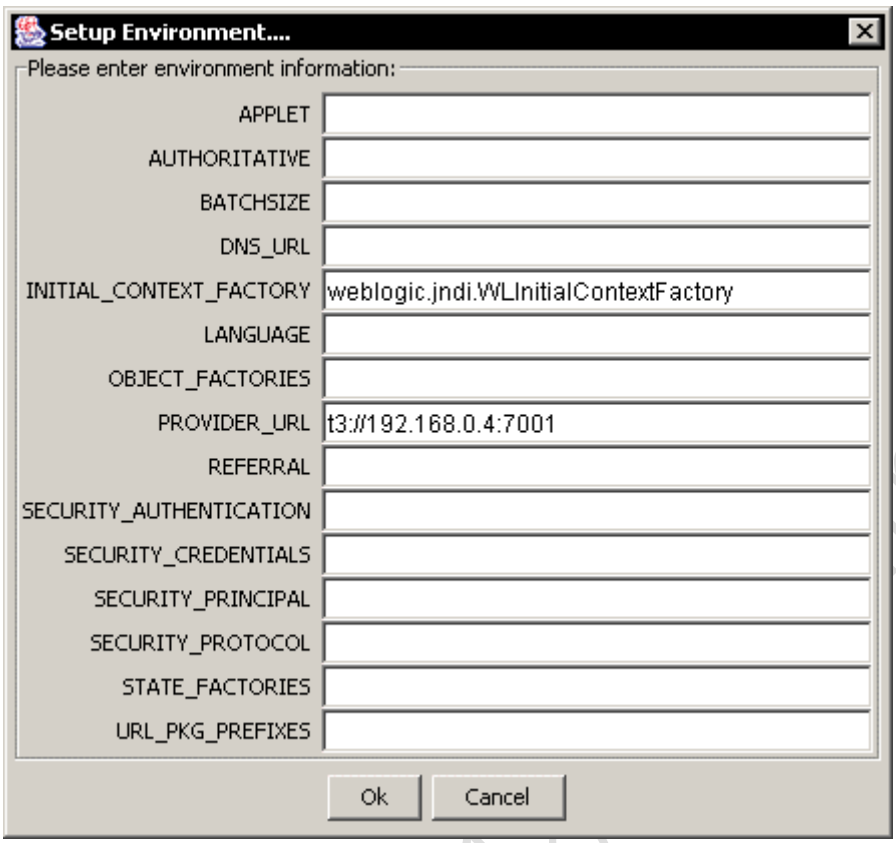
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Specifying EJB Parameter Values ダイアログ

パラメータを選択すると、パラメータのデータ型がウィンドウの下部に表示されます。以下では、パラメータの値を指定することができます。データ型に正しい値を入力し、**Set Value** ボタンをクリックしてください。これにより、パラメータ値が修正されます。すべてのパラメータ値の設定が完了したら、**Environment** ボタンをクリックします。これにより、アプリケーションサーバの環境プロパティを指定できる新しいダイアログが表示されます。この情報は、EspressManager が EJB に接続するために必要です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



EJB 環境設定

ここにあるフィールドは、JNDI コンテキストインタフェースの使用可能な環境プロパティです。すべての値を指定する必要はなく、環境（アプリケーションサーバ）に必要な情報のみを指定する必要があります。環境変数の指定が終わったら、**OK** ボタンをクリックします。EJB 設定ウィンドウに戻ります。**OK** をもう一度クリックして、EJB データソースの設定を完了します。**EJBs** を持つ EJB の下に新しいノードが表示されます。

EJB ソースを選択して **Edit** ボタンをクリックすると、パラメータ値を変更できます。

help/examples/DataSources/EJB ディレクトリにインストールに含まれるサンプル EJB データソースがあります。ディレクトリには **sales.ear** ファイルと **salesClient.jar** ファイル、および Sales Entity Bean のソースコードが含まれています。**sales.ear** ファイルは、Java 2 Reference Implementation にデプロイされるように設計されており、基盤となるストレージメカニズムとして Cloudscape データベースを使用します。**SalesClient.java** プログラムを使用して、Cloudscape データベースにデータを取り込むことができます。**salesClient.jar** ファイルには、デプロイされた Sales EJB に接続するためのスタブクラスが含まれており、EspressManager クラスパスに存在する必要があります。

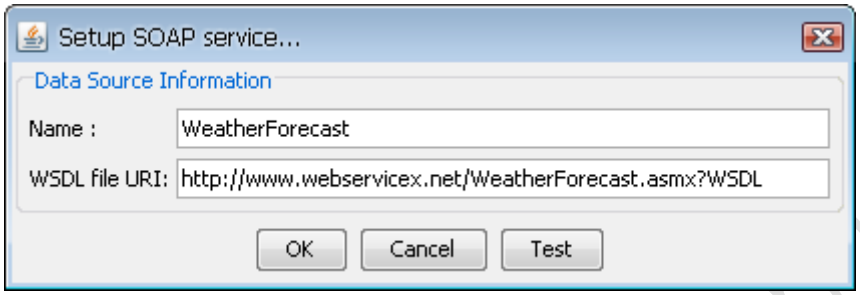
3.3.7 WSDL をサポートする SOAP のデータ

EspressReport では、SOAP(Service Oriented Architecture Protocol)を使用してデータを取得することもできます。WSDL を使用して SOAP データソースに接続するには、サービス、SOAP アクション、操作名またはパラメータの URL を知る必要はありません。WSDL ファイルに必要な情報がすべて含まれ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ています。

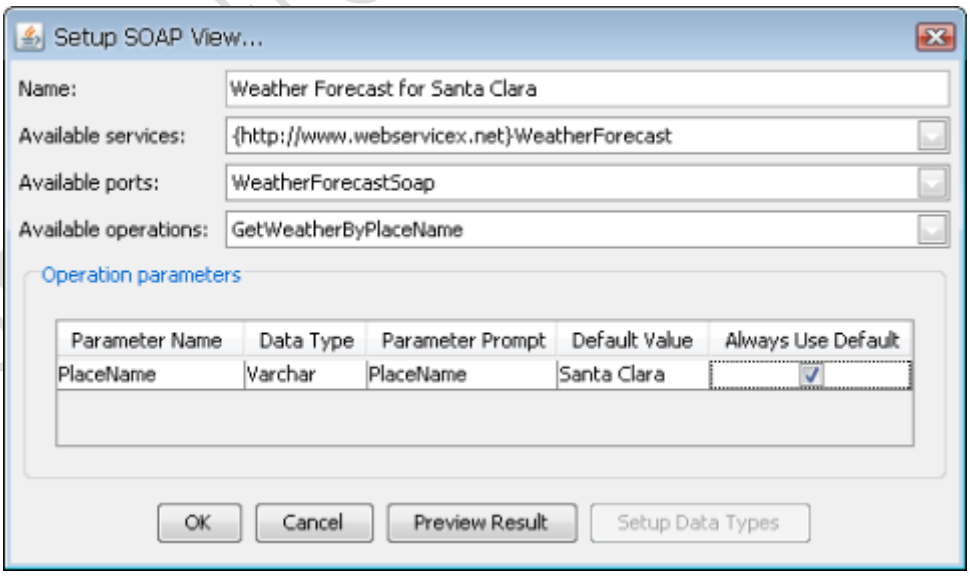
SOAP データソースを設定するには、Data Source Manager で **SOAPServices** ノードを選択し、**Add** ボタンをクリックします。新しい SOAP データソースのオプションを指定するダイアログが表示されます。



SOAP データソースのセットアップ

Name では、データソースの表示名を指定できます。**WSDL file URI** では、WSDL ファイルの場所を指定できます。場所は、サーバ上の絶対パスか、EspressReport のインストールディレクトリまたは URL からの相対パスです。接続情報を指定したら、**Test** ボタンをクリックして WSDL ファイルへの接続をテストできます。これにより、指定した URL から WSDL ファイルが取得されるかどうかがテストされ、サポートされている SOAP 操作があるかどうかチェックされ、問題が報告されます。**OK** をクリックすると、新しい SOAP データソースノードがデータレジストリに追加されます。

新しい SOAP ビューを追加するには、既存の SOAP データソースを選択し、**Add** ボタンをクリックします。ダイアログが開き、SOAP クエリを作成するために必要なすべてのパラメータが表示されます。



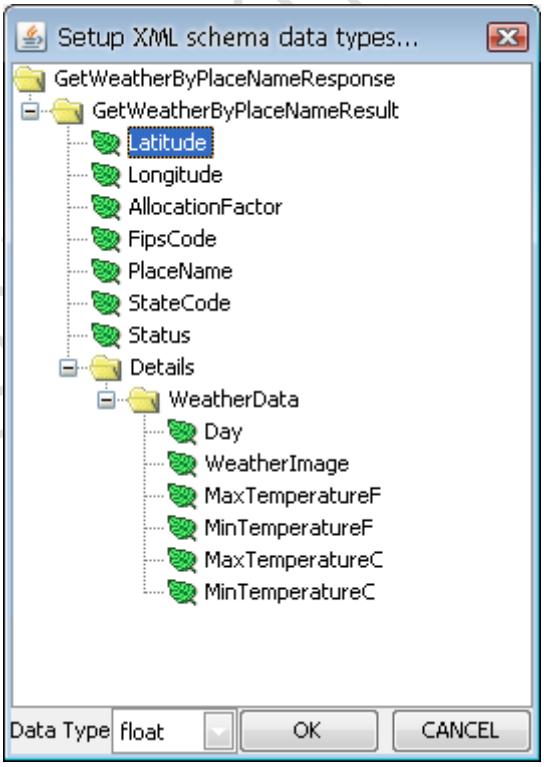
Setup SOAP View ダイアログ

ダイアログの最初のオプションでは、データソースマネージャで表示名を指定できます。次に、ダイアログに 3 つのドロップダウンメニューがあります。最初のドロップダウンメニューには、WSDL ファイルに記述されているすべての SOAP サービスが含まれています。サービスを指定すると、このサービス

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

のすべてのポートが 2 番目のドロップダウンリストに表示されます。ポートを選択すると、最後のドロップダウンリストにこのポートのすべての操作が入力されます。任意のドロップダウンリストの上にマウスを移動すると、サービス/ポート/操作のドキュメント (WSDL ファイルにドキュメントがある場合) のヒントが表示されます。サービス、ポート、および操作を指定すると、操作のすべてのパラメータが読み取られます。いくつかのパラメータがある場合、それらはテーブルに表示されます。表の最初の 2 つの列は編集できません。それらは WSDL ファイルから読み込まれます。次の 2 列は編集可能で、パラメータプロンプトとデフォルト値を指定できます。最後の列には、デフォルトのパラメータ値を常に使用するかどうかを選択できるチェックボックスがあります。これは、このパラメータの値が固定され、プロンプトが表示されないことを意味します。このチェックボックスがチェックされていないすべてのパラメータは、レポート/チャートパラメータとして使用されます。

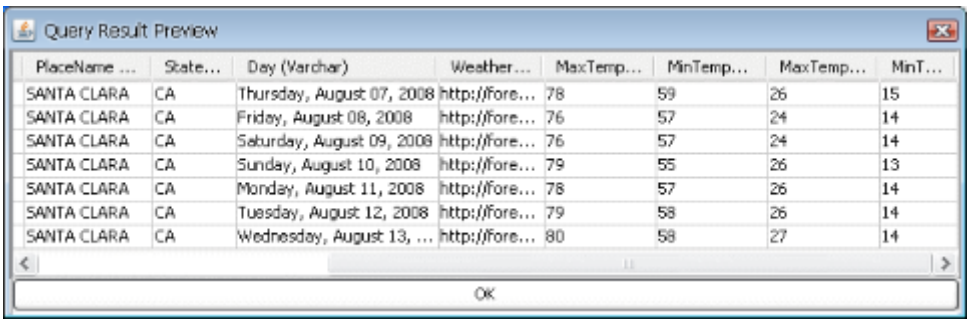
Setup Data Types ボタンは、Data Source Manager から SOAP ビューを編集する場合にのみ使用でき、必要に応じてデータ型を調整できます。SOAP 応答の結果を確認するには、**Preview Result** ボタンをクリックします。すべてのデフォルト値がテストされ、適切なデータ型があるかどうかが確認されます。一致しない場合は、調整するよう指示されます。その後、セットアップデータタイプダイアログが表示されます（XML データソースと同じです)。データソースがパラメータ化されている場合は、データ型を指定する前にパラメータダイアログボックスが表示されます。XML スキーマを適切に生成するためには、既存のパラメータ値を指定する必要があります。



Setup XML schema data types ダイアログ

このダイアログからデータ型を設定できます。この振る舞いは、DTD スキーマを持つ XML データソースとまったく同じです(XML および XBRL ファイルのデータを参照)。データタイプを指定したら、**OK** ボタンをクリックします。ダイアログボックスが開き、結果のプレビューが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Query Result Preview ダイアログ

このダイアログで **OK** ボタンをクリックすると、Setup SOAP View ダイアログに戻ります。必要な情報をすべて指定したら、ダイアログの **OK** ボタンをクリックします。新しいノードが Data Source Manager の SOAP データソースの下に追加され、レポートまたはグラフの作成に使用できるようになります。

いくつか Web サービスで試すこともできます。

米国天気予報(USA Weather Forecast)
この Web サービスは、米国内の有効な郵便番号または地名の天気予報を取得します。使用する WSDL ファイルの場所はこちらです。

米国住所確認(US Address Verification)
このサービスは、指定された米国の住所を確認するだけです。WSDL ファイルの場所はこちらです。

通貨変換器(Currency Convertor)
この Web サービスは、通貨を別の Web サービスに変換します。WSDL ファイルの場所はこちらです。

3.3.8 Salesforce のデータ

Salesforce データソースは、Salesforce データを EspressReport に表示する既存の Salesforce ユーザ向けに設計されています。Salesforce サーバへの接続は、Salesforce Partner WSDL(バージョン 13.0)を使用して SOAP 経由で確立されます。ユーザは SOQL（Salesforce Object Query Language）クエリによって Salesforce サーバと通信します。このデータソースで作業するには、ユーザ名とパスワードの有効な Salesforce アカウントが必要です。さらに、EspressReport Salesforce データソースを使用するユーザは、信頼できるネットワークの Salesforce アカウントにアクセスできる必要があります。あなたの IP アドレスを信頼できる IP リストに追加するには、以下に説明するようにコンピュータをアクティブにする必要があります。

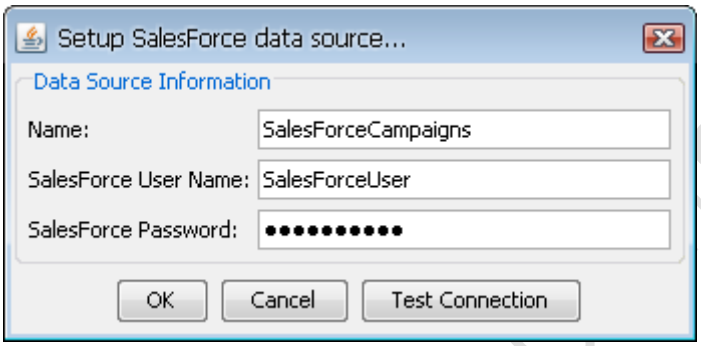
SOQL クエリと Salesforce ユーザのアカウントを信頼できるネットワークからアクティブ化する方法の詳細については、以下の Salesforce サイトを参照してください。

SOQL クエリ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Salesforce ユーザのアカウントを有効にする

Salesforce データソースを設定するには、データソースマネージャで **SalesForce** ノードを選択し、**Add** ボタンをクリックします。ダイアログボックスが開き、Salesforce アカウントへのデータソースの表示名、ユーザ名、およびパスワードを指定するよう求められます。接続情報を指定したら、**Test Connection** ボタンをクリックして、Salesforce アカウントへの接続をテストできます。これにより、提供した情報を使用して接続がテストされ、問題が報告されます。

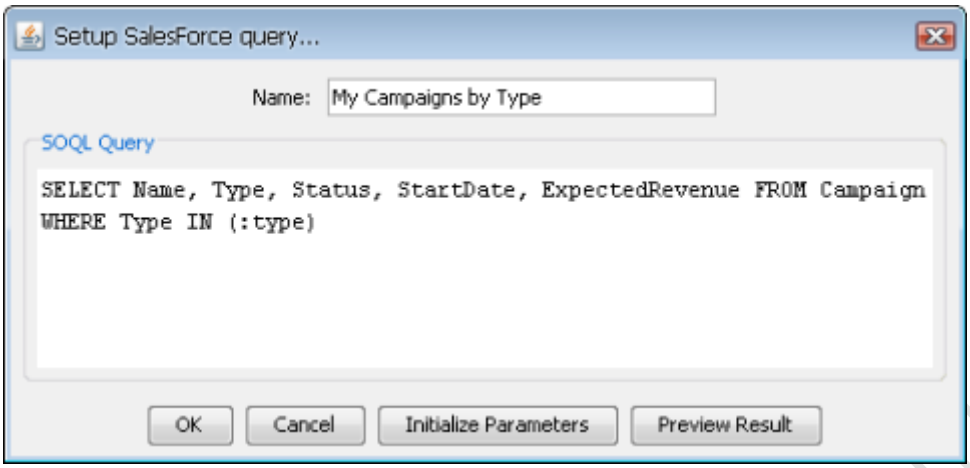


Setup SalesForce Data Source ダイアログ

Salesforce データソースを追加すると、新しいノードが Data Source Manager ウィンドウに表示されます。新しい Salesforce クエリを追加するには、**Add** ボタンをクリックします。新しいダイアログが開き、クエリ名と SOQL クエリを指定するように求められます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Setup SalesForce Query ダイアログ

現在の EspressReport バージョンでは、子から親への関係クエリのみがサポートされています。親から子へのクエリ（ネストされた SOQL クエリを使用）は使用できません。Salesforce リレーションシップクエリとその構文の詳細については、Salesforce サイトを参照してください。

さらに、このダイアログでは、クエリに単一値または複数値のパラメータが含まれている場合にクエリパラメータを初期化できます。パラメータは、":"文字を使用して SOQL 文内で指定されます。一般に、パラメータは SOQL Select 文の WHERE 句に配置されます。
たとえば、以下の SOQL 文では **CampaignName** という単一の値パラメータを指定します。

```
Select Name, Type, Status, ActualCost From Campaign Where Name = :CampaignName
```

実行時にキャンペーン名を入力し、そのキャンペーンのデータのみを取得することができます。

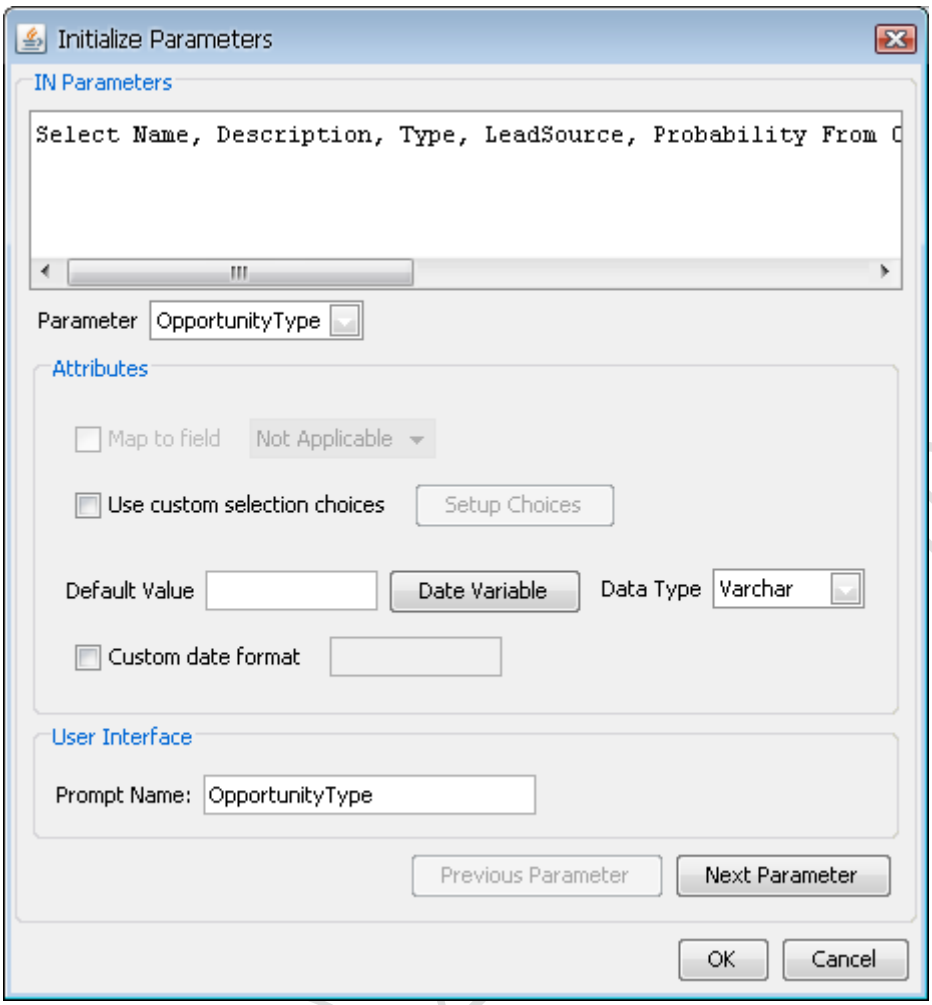
SOQL 文の別の例では、単一の値ではなく値の配列を入力として使用する複数値のパラメータを使用しています。

```
Select Name, Description, Type, LeadSource, Probability From Opportunity Where Type IN (:OpportunityType) And LeadSource IN (:OppLeadSource)
```

この文は、**OpportunityType** と **OppLeadSource** という 2 つの複数値パラメータを指定します。ランタイムに opportunity types と lead source を指定することができ、指定されたパラメータ値に従ってデータを取得するだけです。

SOQL クエリパラメータを初期化するには、**Initialize Parameters** ボタンをクリックします。パラメータの初期化ダイアログが表示され、パラメータのマッピングを指定できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Initialize Parameters ダイアログ

このダイアログでは、以下のオプションを指定できます。

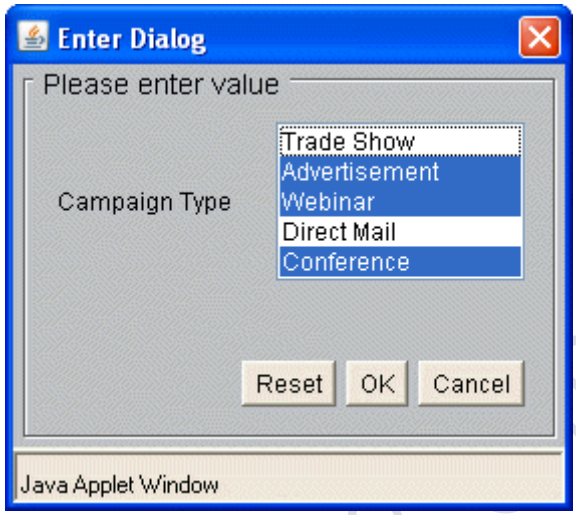
Map to field:
これにより、値がパラメータ入力に使用される Salesforce データソースのフィールドを指定できます。このオプションを選択すると、レポート/チャートのプレビューまたは実行時に表示されるパラメータプロンプトが変更されます。パラメータを Salesforce フィールドにマップすると、パラメータ値を選択できる個別の値のドロップダウンリストが表示されます。マップしないと、特定のパラメータ値を入力する必要があります。

カスタム選択項目を使用する
個別の列の値をすべて含むドロップダウンメニューを表示するのではなく、パラメータ値のカスタムリストを作成できます。リストを設定するには、このオプションを選択し、**Setup Choices** ボタンをクリックします。これにより、新しいダイアログが開き、選択肢のリストを作成することができます。

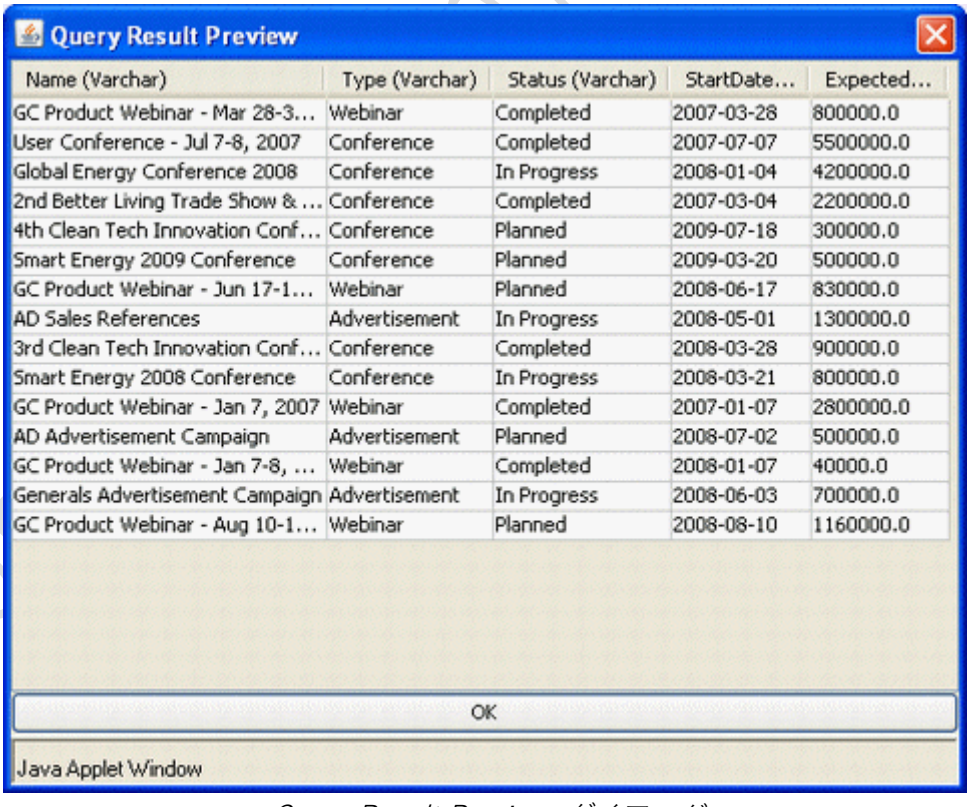
残りのオプションは基本的にデータベースクエリパラメータと同じです。データベースクエリパラメータの初期化の詳細は、クエリパラメータの初期化を参照してください。使用可能なすべてのパラメータのマッピングを指定したら、**OK** ボタンをクリックして、Salesforce クエリの設定ダイアログに戻ります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Setup SalesForce Query ダイアログでは、**Preview Results** ボタンを使用してクエリ結果をプレビューして、クエリの出力を確認することもできます。パラメータ化されたクエリがある場合、パラメータのプロンプトダイアログが表示され、パラメータ値を指定するよう求められます。パラメータ値を指定したら、**OK** ボタンをクリックすると、クエリ結果のプレビューダイアログが表示されます。



Parameter Prompt



Query Result Preview ダイアログ

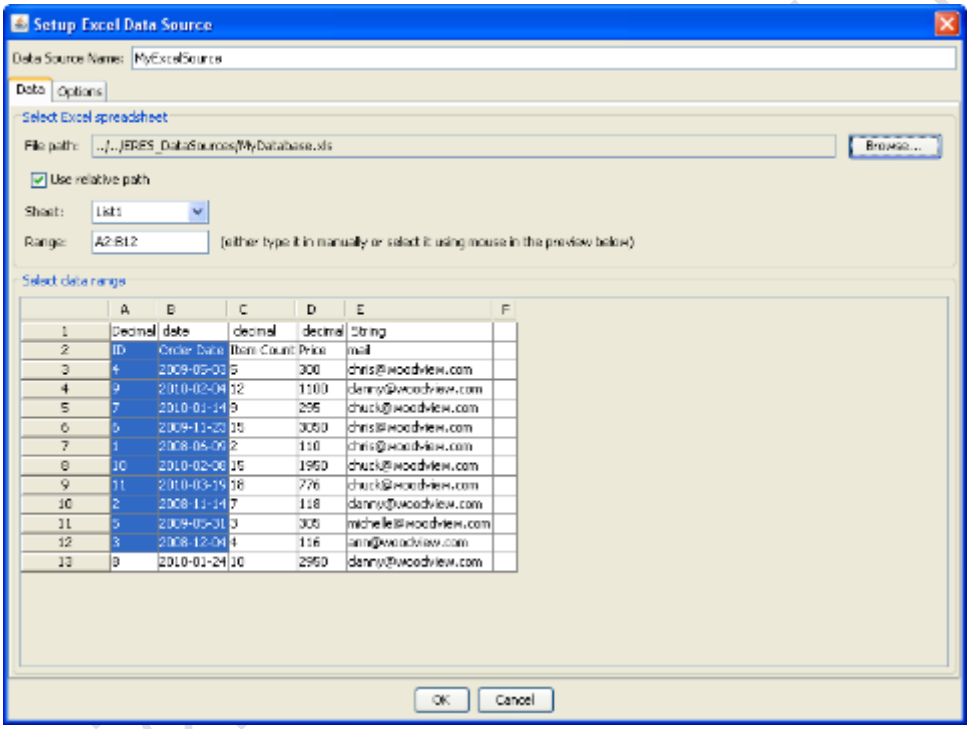
このダイアログでは、クエリ出力を確認できます。 **OK** ボタンをクリックすると、クエリ結果プレビューダイアログが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

クエリを指定したら、**OK** ボタンをクリックします。クエリが、Data Source Manager の Salesforce データソースの下に新しいノードとして追加され、レポートまたはチャートの作成に使用できるようになります。

3.3.9 Excel ファイルのデータ

EspressReport では、Excel ファイルから取得したデータを使用してグラフをデザインすることもできます。Excel ファイルをデータソースとして追加するには、Data Source Manager で **ExcelFiles** ノードを選択して、**Add** ボタンをクリックします。ダイアログボックスが開き、データソース名を指定し、データをインポートする Excel ファイルを選択するよう求められます。



Setup Excel Data Soure ダイアログ - データ

Excel ファイルを選択すると、インポートされたデータがダイアログでプレビューされます。**Use Relative Path** チェックボックスをオンにすると、選択した Excel ファイルへのファイルパスが EspressReport インストールディレクトリからの相対パスとして設定されます。それ以外の場合は、フルパスが使用されます。Excel ファイルに EspressReport がインストールされているディスクドライブとは別のディスクドライブに格納されている場合、このオプションは使用できません。シートを選択することができます（ファイルに複数のファイルがある場合）。また、マウスを使用して設計するデータソースに関連するセルを選択するか、**Range** ボックスに範囲を指定します（たとえば、ソース Range ボックスに **A2:B12** と入力すると、列 A と B、および行 2～12 のデータが使用されます（これは MS Excel の範囲でも使用されます）。行ヘッダーまたは列ヘッダーをクリックして、データを選択することもできます。**Ctrl** キーまたは **Shift** キーを押しながら、さらに行または列を選択します。すべてのセルを選択するには、左上隅をクリックします。ここでも、この動作は MS Excel に似ています。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

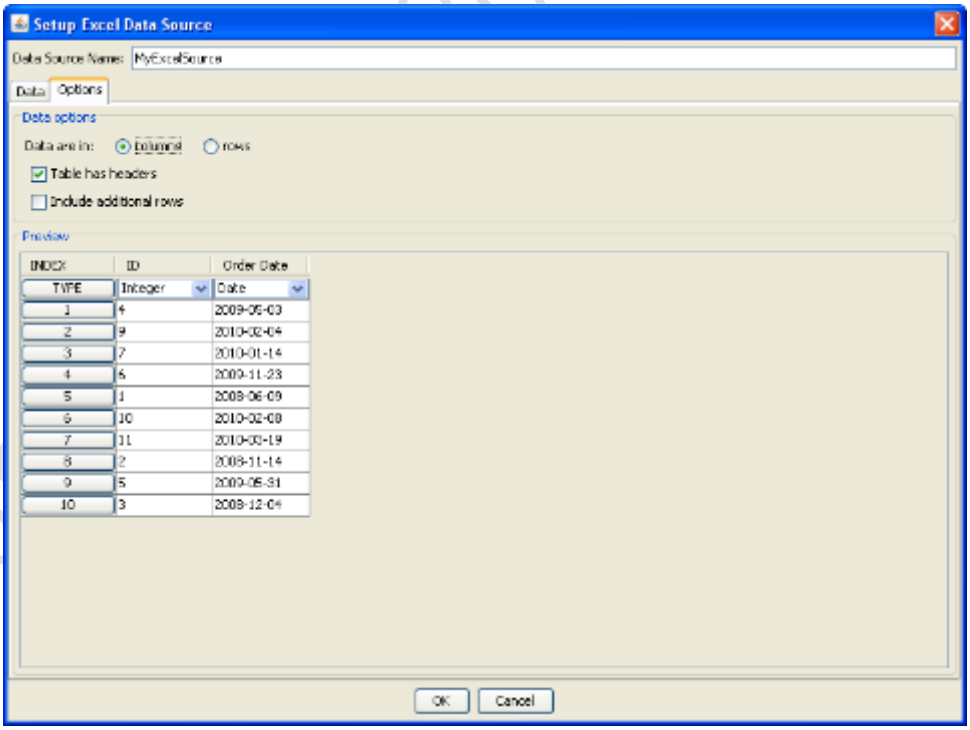
EspressReport は、*** .xls** ファイルと*** .xlsx** ファイルの両方を処理できます。*** .xlsx** ファイルは Microsoft Excel 2007 以降で使用され、Open XML に基づいています。

EspressReport では、基本的な Excel 式も処理できます。入力した数式を処理できない場合は、エラーメッセージが表示されます。

シートの最後の空の行（データが列にある場合）または列（データが行にある場合）は、選択されている場合でも、データソースから自動的に削除されます。

Options タブをクリックして、データ構造を正しく取得するためにデータが列にあるのか行にあるのかを指定します。テーブルヘッダーをデータ選択に含めるかどうかを指定することもできます。 **Include additional rows** オプションまたは **Include additional columns** オプション（データが列または行にあるかどうかに応じて）を使用すると、データソースを変更せずにデータソースを作成した後に自動的に Excel ファイルにデータ行または列を含めることができます。Date Source Manager を手動で起動します。

Options タブのダイアログの下部には、現在の設定に従ったデータソースの内容が表示されます。必要に応じて、データソースの各列のデータ型を変更することができます。データ型は自動的に検出されるため、ほとんどの場合、型の変更は必要ではありません。



Setup Excel Data Source ダイアログ - オプション

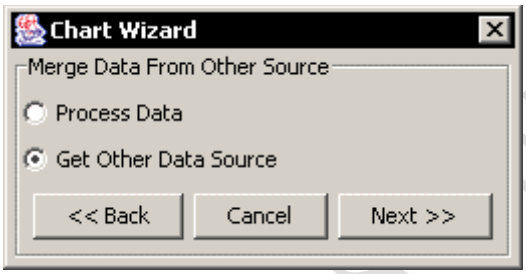
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.3.10 レポートの使用

3.3.10.1 複数のデータソースの使用

使用するデータソースを選択して **Next** ボタンをクリックすると、次の画面にデータソースの最初の 20 個のレコードが表示されます(ただし、データビューは例外で、フィールドを選択して条件を設定する必要があります)。**Show All Records** チェックボックスをオンにすると、すべてのレコードを表示できます。

EspressReport では、複数のデータソースからチャートを作成することができます。最初のソースを選択し、データテーブルウィンドウから **Next** をクリックすると、現在のデータを処理するか、別のデータソースを選択するかを尋ねるダイアログが表示されます。



Additional Data Source ダイアログ

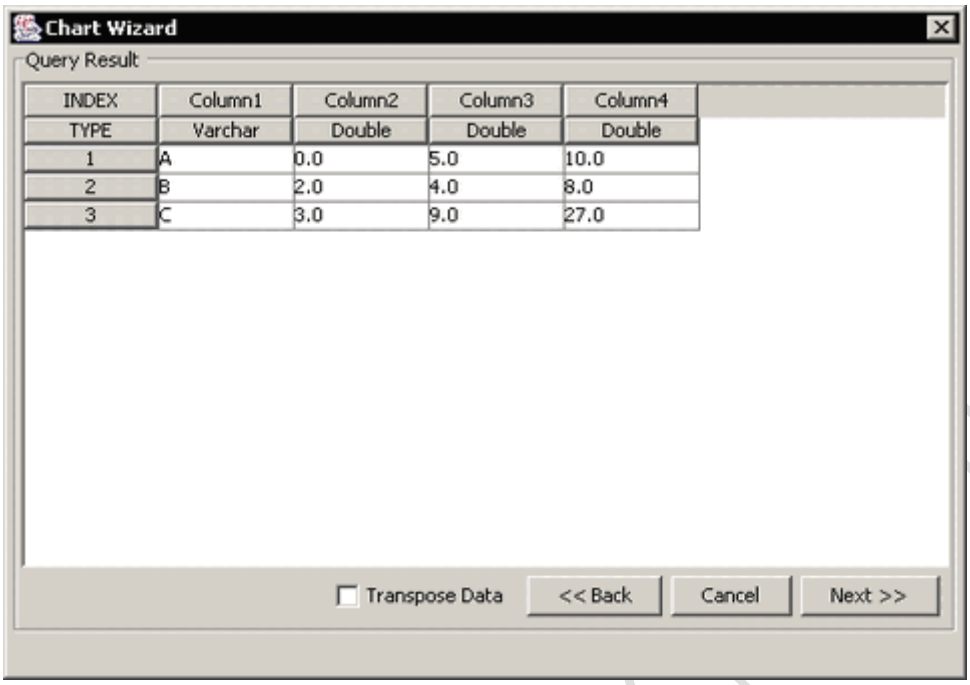
Process Data を選択して **Next** ボタンをクリックすると、グラフィックウィザードの次のステップに進みます。**Get Other Data Source** を選択すると、Data Source Manager に戻り、グラフの別のデータソースを選択します。このプロセスを繰り返して、必要な数のソースを選択することができます。

複数のデータソースがデータテーブル内で横方向に結合されています。これは、2 番目(または 3 番目、4 番目など)のデータソースの列が、最初のデータソースの列の右隣りの配置されることを意味します。1 つのデータソースの列に他の行よりも多くの行がある場合は、NULL 値が余分な行に配置されます。

たとえば、2 つのデータソースを使用してグラフを作成するとします。最初のソースによって生成されたデータテーブルは、以下のようになります。

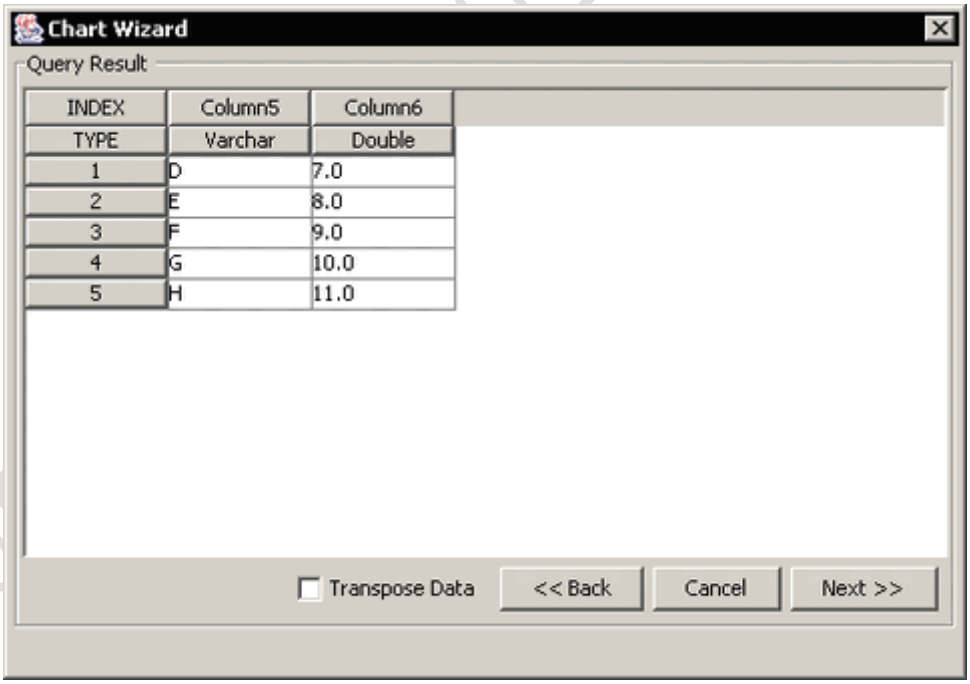
Copyright(C) 2018 Climb.Inc

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



最初のデータソースからのデータ

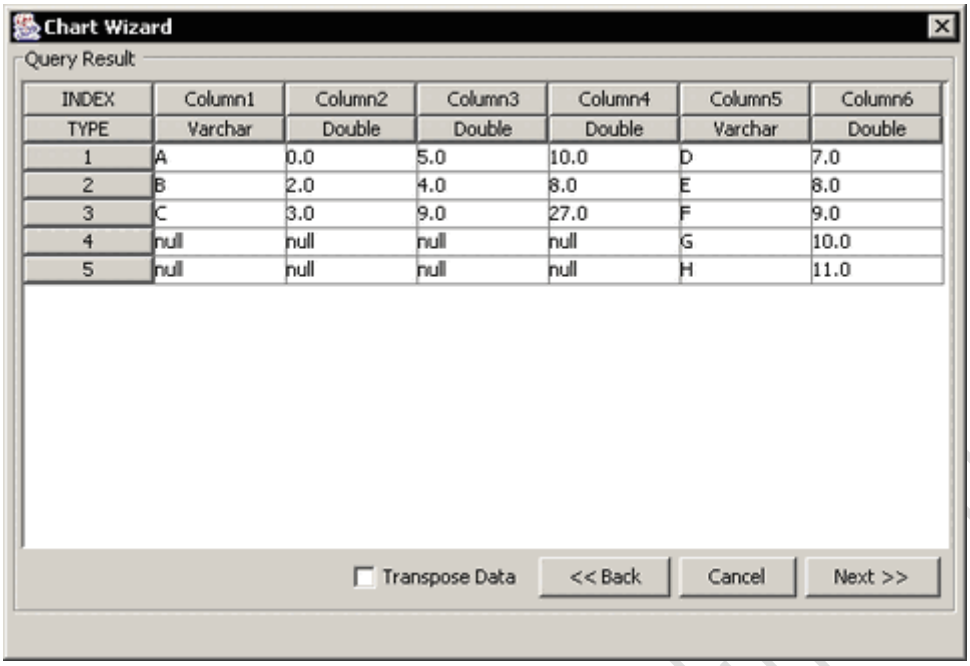
2 番目のソースからのデータは以下のようになります。



2 番目のデータソースからのデータ

2 つのデータソースを結合すると、以下のデータテーブルが得られます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



複合ソースからのデータ

ご覧のとおり、2 つのデータソースが並べられて配置されています。第 2 のソースは、第 1 のソースよりも多くの行のデータを有するので、null データの追加の行が追加されます。

パラメータ化されたデータソースを使用して複数のデータソースを作成することはできません。

3.3.10.2 データソースの変更

レポートデザインのどの時点でも、テンプレートのデータソースを変更できます。グラフテンプレートはデータソース情報とともに保存されるため、Report Designer 内のオプションを使用してテンプレートのデータソースを変更する必要があります。レジストリ内のデータソースを変更するだけで、データソース更新機能を使用しない限りテンプレートには影響しません（詳細は、データソースの更新を参照してください）。テンプレートのデータソースを変更するには、**Data > Change Data Source** を選択するか、ツールバーの **Change Data Source** ボタンをクリックします。これにより、Data Source Manager が表示され、新しいデータソースを選択したり、既存のデータソースを変更したりすることができます。

新しいデータソースが以前のデータソースと大幅に異なる場合は、データをレポートに再マップする必要があります。これは、レポートのデータ構造が大幅に変更されてもレイアウトや計算が正しく機能しないことがあるため必要です。レイアウトとデータマッピングの詳細については、レポートの種類とデータマッピングを参照してください。

3.3.11 データソースの更新

テンプレートのグループを移動したり、ある場所から別の場所に EspressReport を完全にインストールしたりするには、多くの状況があります。たとえば、アプリケーションを開発環境から実稼働環境に移行することができます。各環境では、データソースの場所と接続情報が異なる場合があります。このシナリオでは、前のセクションで説明したようにチャートテンプレートを 1 つずつ更新することは、多数

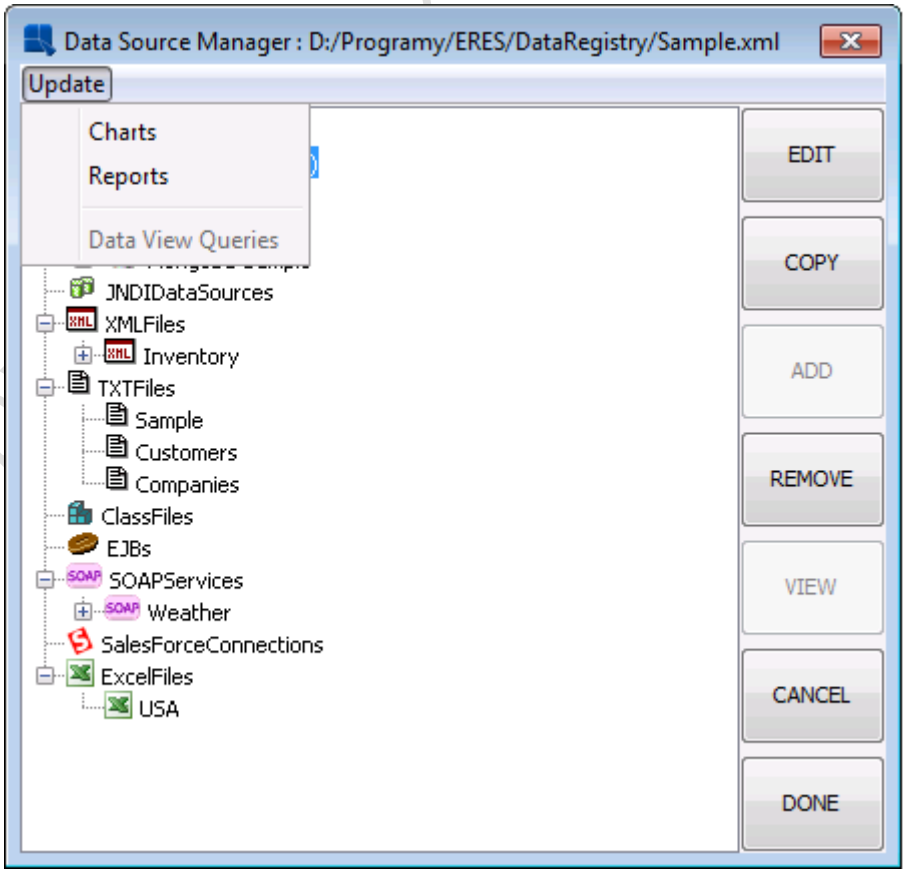
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

のテンプレートの接続情報を変更することは現実的ではありません。代わりに、EspressReport を使用すると、データレジストリの情報に基づいてテンプレートのグループをすばやく更新できます。

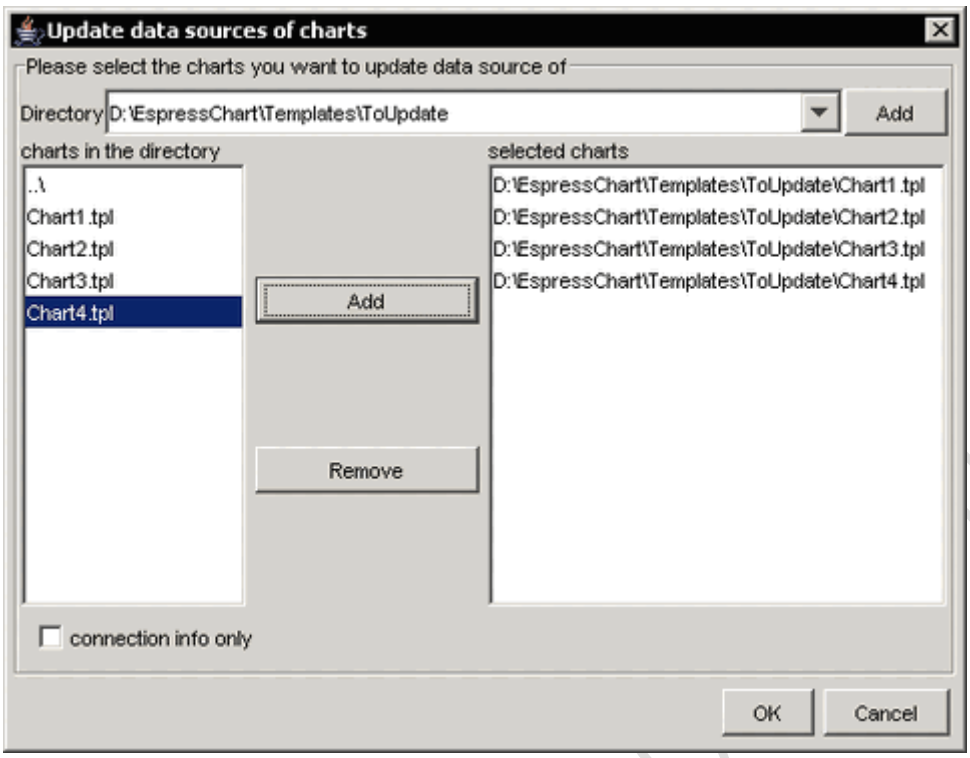
チャートテンプレートは、データソース情報の内部コピーを保持しており（独立して展開できるようにする）、作成されたデータレジストリに関する情報（場所とソース）も保持します。したがって、クエリの編集で詳述したように、チャートのクエリを変更すると、変更をデータレジストリに保存するオプションがあります。この機能を使用するには、データレジストリを最新の状態に保つ必要があります。これは、クエリの変更をレジストリに保存し、データビューの構造の変更をデータビューのクエリ更新で説明したようにデータビューのクエリに伝播する必要があることを意味します。

この機能を使用するには、まずチャートテンプレートに伝播するデータレジストリを変更します。これらの変更には、データベース接続情報、テキストのファイルの場所、XML データファイル、テンプレートに渡すデータビューやクエリへの変更などが含まれます。インストール間でチャートを移動する場合、データレジストリファイルを新しいインストールの同じ相対位置に移動する必要があることに注意してください。さらに、そのレジストリ（**.qry/.dvw/.ddt**）に関連するクエリファイルは、新しいインストールの/**queries**/ディレクトリに移動する必要があります（クエリファイル名はレジストリ名で始まります）。

Data Source Manager から、**Update** > **Charts** を選択します。これにより、更新するチャートを選択するためのダイアログが表示されます。



Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Select charts for Data Source Updating

更新するチャートを選択するにはまず、チャートを含むディレクトリを参照します。ディレクトリを選択すると、チャートテンプレートがダイアログの左側に表示されます。更新するテンプレートを選択し、**Add** ボタンをクリックします。テンプレートを選択するのと同じ数の異なるディレクトリに移動できます。**Connection Info Only** オプションを選択すると、接続情報(データベース URL、ドライバ、ユーザ名、パスワード、および XML ファイル、テキストファイルおよび Java クラスの場所のみが更新されます)。クエリとデータビューの情報は、テンプレート内で更新されません。

更新するテンプレートの選択が完了したら、**OK** をクリックして、更新プロセスを開始します。ダイアログに現在の進行状況とエラーが表示されます。

UpdateDaraSources という名前のログファイルも、インストールのルートディレクトリに進行状況画面の内容とともに書き込まれます。様々な理由で更新に失敗したチャートは、Chart Designer のオプションを使用して手動で更新できます。

現在のデータレジストリのチャートのみが変更されます。現在のレジストリ内のソースからデータを取得しないグラフを選択すると、無視されます。

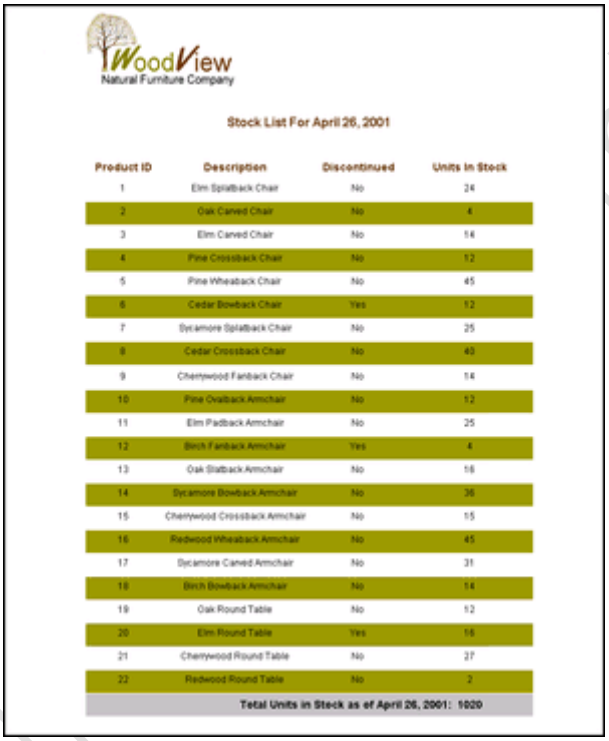
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.4 レポートの種類とデータマッピング

EspressReport は 5 つのレポートの種類（シンプルカラムナ、サマリブレーク、クロス集計、マスタ&詳細、メーリングラベル）をサポートしています。

3.4.1 シンプルカラムナレポート

シンプルカラムナレポートは ExpressReport でサポートされている最も基本的なレポートです。グルーピングや分割なしに、単一テーブルに列データを表示します。


WoodView
Natural Furniture Company

Stock List For April 26, 2001

Product ID	Description	Discontinued	Units In Stock
1	Elm Splatback Chair	No	24
2	Oak Carved Chair	No	4
3	Elm Carved Chair	No	16
4	Pine Crossback Chair	No	12
5	Pine Wheatback Chair	No	45
6	Cedar Bowback Chair	Yes	12
7	Sycamore Splatback Chair	No	25
8	Cedar Crossback Chair	No	40
9	Cherrywood Fanback Chair	No	14
10	Pine Ovalback Armchair	No	12
11	Elm Padback Armchair	No	25
12	Birch Fanback Armchair	Yes	4
13	Oak Slatback Armchair	No	16
14	Sycamore Bowback Armchair	No	36
15	Cherrywood Crossback Armchair	No	15
16	Redwood Wheatback Armchair	No	45
17	Sycamore Carved Armchair	No	31
18	Birch Bowback Armchair	No	14
19	Oak Round Table	No	12
20	Elm Round Table	Yes	16
21	Cherrywood Round Table	No	27
22	Redwood Round Table	No	2

Total Units in Stock as of April 26, 2001: 1020

シンプルカラムナレポート

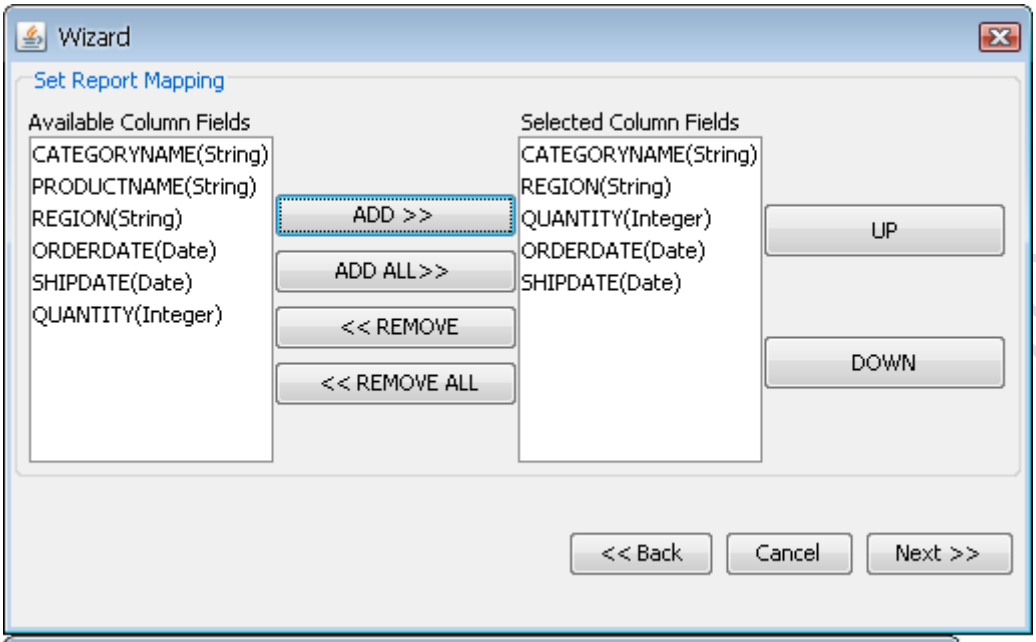
3.4.1.1 データマッピング

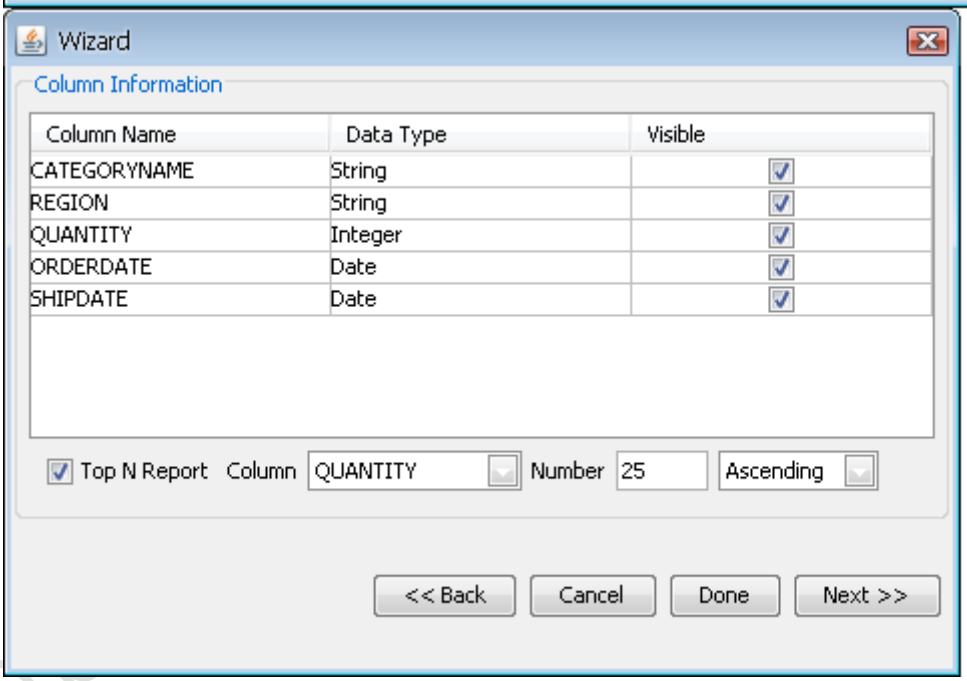
このタイプのデータマッピングの最初の手順は、Report ウィザードの **Set Report Mapping** ウィンドウからレポートに含める列を選択することです。ウィンドウの左側には、クエリまたはデータファイルから使用可能なすべての列がリストされ、右側にはレポートに追加される列がリストされます。カラム名をクリックして選択し（複数選択の場合は Shift＋クリック）、**ADD** または **REMOVE** ボタンを押して追加または削除します。このウィンドウで列を選択する順序は、レポートに表示される順序です。選択した列の順序を変更するには、移動する列を強調表示し、**UP** または **DOWN** のボタンをクリックします。選択したフィールドをクリックしてドラッグすることでフィールドを移動することもできます。

レポートマッピングの 2 番目の手順は、列オプションを設定することです。単純な列レポートの場合、使用可能なオプションは 1 つだけです。このウィンドウには、レポート用に選択したすべての列と利用可能なすべてのオプションが表示されたテーブルが表示されます。最初のフィールドは列名を表示し、2 番目のフィールドはデータ型を表示し、3 番目のフィールドは Visible というチェックボックスを表示

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

します。このボックスをオンまたはオフにすると、レポートの本文内で列が表示または非表示になります（デフォルトではすべての列が表示されます）。





シンプルカラムナレポートマッピング

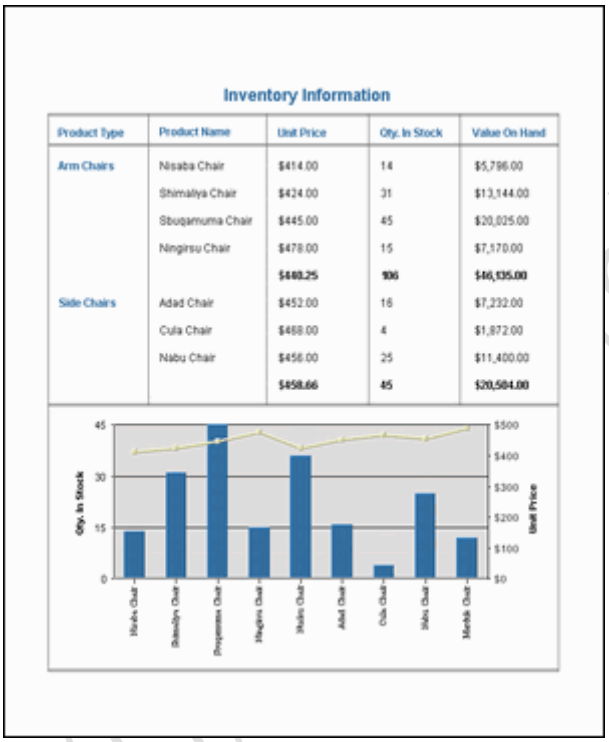
3.4.1.1.1 上位 N レポート

また、列形式を使用して上位 N レポートを作成することもできます。上位 N レポートでは、レポートの特定の列に基づいて最も高い値のセットが表示され、表示されます。たとえば、合計売上に基づいて上位 5 社の顧客を表示するとします。これを行うには、データマッピングウィンドウの下部にある **Top N Report** チェックボックスをオンにします。次に、メジャーとして使用する列（上記の例では、合計売上列）、検索する数値（つまり 10 または 20 など）、および列を表示するかどうかを指定できます。昇順または降順で並べ替えます。列のレポートは、指定された列の最高値でソートされたレコードの数を返します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.4.2 サマリブレークレポート

シンプルカラムナレポートと同様に、サマリブレークレポートタイプは列データを取り、テーブル形式で表示します。ただし、シンプルカラムナレポートとは異なり、データをセクションに分割してサマリフィールドを挿入することができます。



サマリブレークレポート

3.4.2.1 データマッピング

このタイプのデータマッピングの最初の手順は、Report ウィザードの **Set Mapping** ウィンドウからレポートに含める列を選択することです。ウィンドウの左側には、クエリまたはデータファイルから使用可能なすべての列がリストされ、右側にはレポートに表示される列がリストされます。カラム名をクリックして選択し（複数選択の場合は **SHIFT +クリック**）、**ADD** または **REMOVE** ボタンを押して追加または削除します。このウィンドウで列を選択する順序は、レポートに表示される順序です。選択した列の順序を変更するには、移動する列を強調表示し、**UP** または **DOWN** のボタンをクリックします。選択したフィールドをクリックしてドラッグすることでフィールドを移動することもできます。

データマッピングの 2 番目のステップは、列オプションを設定することです。サマリブレークレポートには 5 つの列オプションがあります。このウィンドウには、レポート用に選択したすべての列と使用可能なすべてのオプションを示す表が表示されます。最初のフィールドは列名を表示し、2 番目のフィールドはデータ型を表示し､3 番目のフィールドは Visible とマークされたチェックボックスを表示します。このボックスをオンまたはオフにすると、レポートの本文内で列が表示/非表示になります（デフォルトではすべての列が表示されます)。4 番目のフィールドは行区切りフィールドです。このボックスにチェックマークを付けると、選択した列フィールドが変更されるたびにレポートが途切れてサマリ列が挿

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

入されます。デフォルトでは、レポート用に選択された最初の列はブレークフィールドです。5 番目のフィールドで集計を選択できます。ウィンドウの下部にある列の集計を実行するチェックボックスをオンにすると、選択した集計が列全体で実行され、レポートに集計されたデータのみが含まれます。チェックボックスをオフにすると、レポートにはすべてのデータが表示され、各行区切り後の集計のみがサマリとして挿入されます。

ドロップダウンメニューでは、その列に対して実行される集計操作を選択できます。実行可能な集計操作は、**なし（none）、合計（sum）、最大（maximum）、最小（minimum）、カウント（count）、平均（average）、最初（first）、最後（last）、二乗和（sum of squares）、分散（variance）、標準偏差（standartd eviation）、個数別（count distinct）**です。列の集計を実行するように選択した場合、列ごとに集約オプションを選択する必要があります。行区切りフィールドとして選択された列は集約できません。

たとえば、次のデータテーブルがあるとします。

Order#	Product	Quantity
12	Chair	2
12	Table	3
14	Cabinet	2
14	Table	5

行ブレークとして**Order#**を設定し、**Perform Column Aggregation** をチェックせずに **Quantity** を **Sum** に集計すると、次のレポートが生成されます。

Order#	Product	Quantity
12	Chair	2
	Table	3
		5
14	Cabinet	2
	Table	5
		7

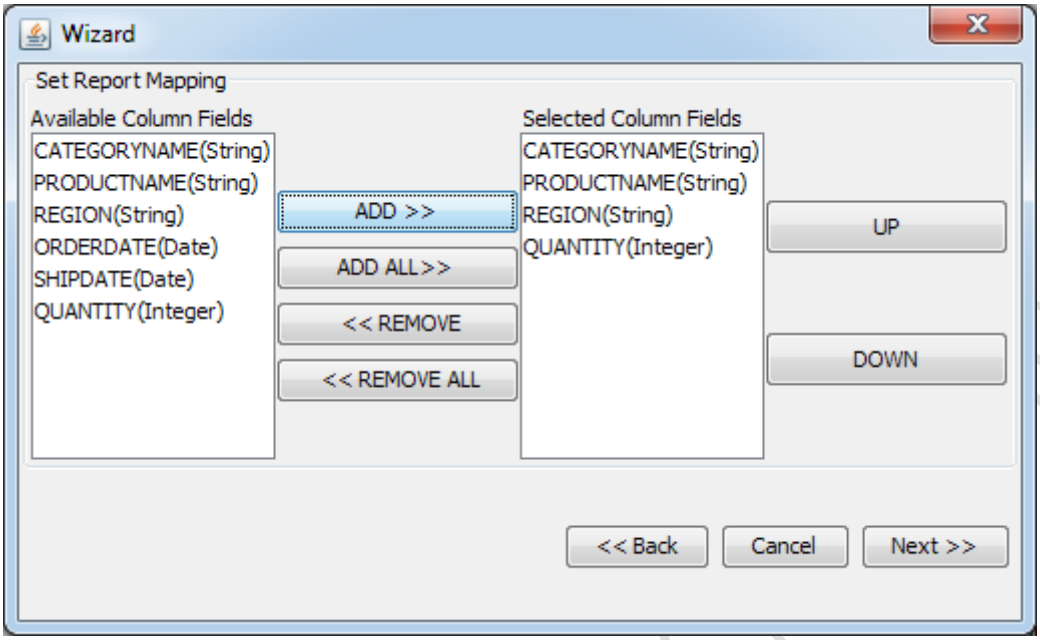
しかし、**Perform Column Aggregation** にチェックを入れ、**Product** を **Count** に変更すると次のレポートが生成されます。

Order#	Product	Quantity
12	2	5
14	2	7
	2	**5**

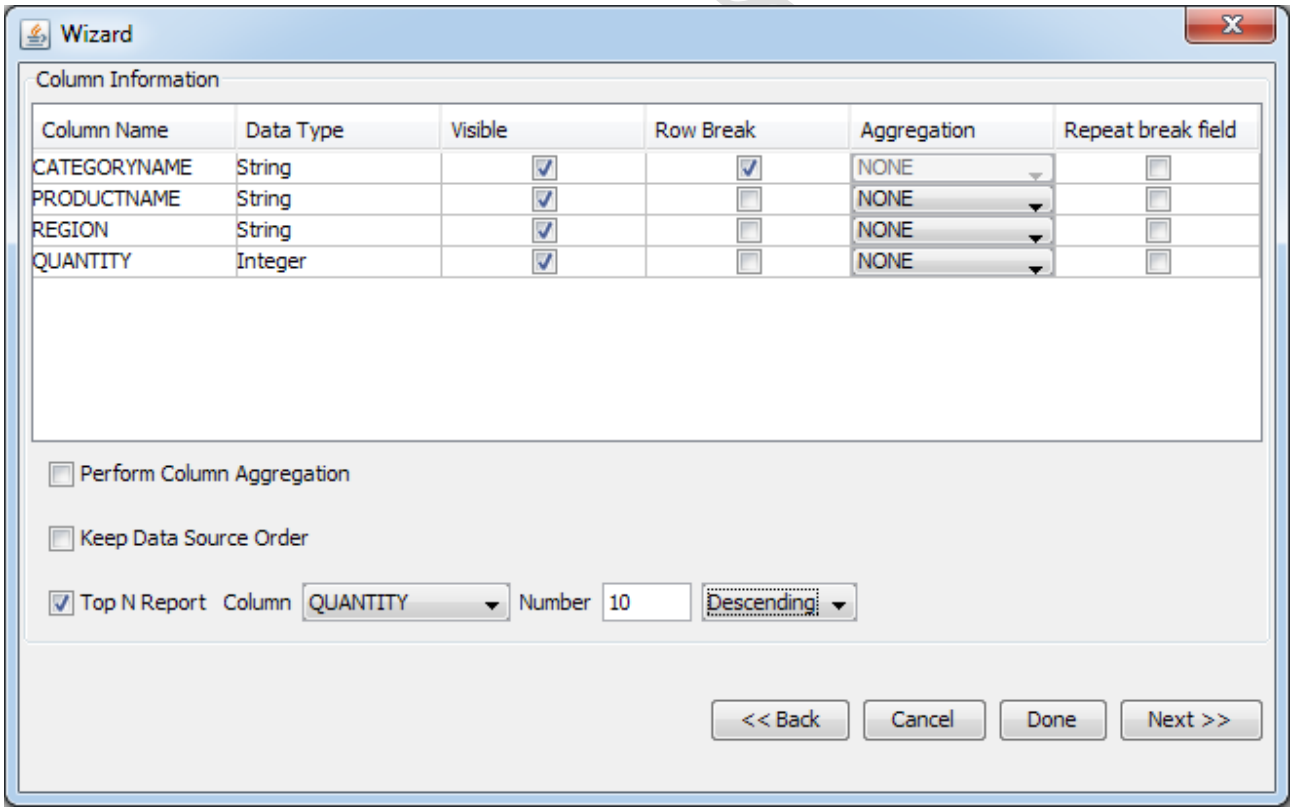
6 番目のフィールドでは、ブレークフィールドを繰り返すかどうかを選択できます。デフォルトでは、行区切りの列は、列の各個別の値（つまりグループごとに 1 回）のみを印刷します。このオプションを

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

選択すると、グループ内の各データ行に対してブレークフィールドが繰り返されます。



Summary Break Report マッピング



Summary Break Report マッピング

データソースで指定されたデータ順序を保持する場合は、**Keep Data Source Order** オプションを選択します。この機能の詳細については[データソースの順序の保持]を参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.4.2.1.1 上位 N レポート

サマリブレークフォーマットを使用して上位 N レポートを作成することもできます。上位 N レポートは、レポート内の特定の列に基づいて、最も高い値のセットを注文して表示します。サマリブレークレポートの場合は、各グループの最高値を表示するように指定することもできます。たとえば、売上高に基づいて各地域のトップ 5 の顧客を表示することができます。これを行うには、データマッピングウィンドウの下部にある **Top N Report** チェックボックスをオンにします。次に、メジャーとして使用する列（上記の例では、合計売上列）、取得する番号（つまり 10 または 20 など）、および列を昇順または降順で並べ替えます。サマリブレークレポートは、指定された列の最高値でソートされた指定されたレコード数（または行ブレークごとに指定されたレコード数）を戻します。

3.4.3 クロス集計レポート

クロス集計レポートは、列状のデータを行列形式で表示および要約するレポート形式です。クロス集計レポートは、多くの場合、スプレッドシートに似ています。行と列の両方が要約され、多次元データを 2 次元形式で表示することができます。

SALES

Category	Product	East	South	Midwest	West	Quantity
Side Chairs	Ishtar Chair	32	26	18	Null	76
	Shamash Chair	18	28	18	4	68
	Enlil Chair	30	32	40	8	110
	Ninurta Chair	18	34	21	Null	73
	An Chair	18	12	34	Null	64
	Enki Chair	26	39	90	16	171
	Ninhursag	39	Null	21	13	73
	Nergal Chair	29	30	34	Null	93
	Zabada Chair	12	49	34	7	102
		222	250	310	48	830
Arm Chairs	Marduk Chair	24	30	Null	14	68
	Cula Chair	58	16	47	8	129
	Nusku Chair	44	31	37	12	124
	Shimaliya	33	41	Null	7	81
	Nisaba Chair	82	39	3	49	173
	Adad Chair	38	Null	12	Null	50
	Ningirsu Chair	12	15	Null	Null	27
	Sbuqamuma	54	25	Null	Null	79
	Nabu Chair	Null	23	14	16	53
		345	220	113	106	784
Round Tables	Apep Table	25	23	31	6	85
	Anubis Table	13	21	36	4	74
	Bast Table	34	Null	18	Null	52
	Ningizida Table	Null	20	23	31	74
	Amon Table	Null	33	Null	Null	33

クロス集計レポート

実際のレポートは、実行中またはプレビュー中にのみ作成されます。クロス集計テーブルは基本的にレポートが実行されるたびにゼロから構築されるため、スムーズに折りたたんで拡大するクロス集計テー

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ブルを作成する方が簡単です。

3.4.3.1 データマッピング

クロス集計レポートのレポートマッピングは、他のレポートタイプよりも複雑であり、正しく実行できるように計画する必要があります。

このタイプのデータマッピングの最初の手順は、ウィザードの **Set Report Mapping** ウィンドウからレポートに含める列を選択することです。ウィンドウの左側には、クエリまたはデータファイルから使用可能なすべての列がリストされ、右側にはレポートに表示される列がリストされます。列名をクリックして選択（Ctrl＋クリックで複数選択）し、**ADD** または **REMOVE** ボタンを押して追加または削除します。このウィンドウで列を選択する順序は、レポートに表示される順序です。列を選択したら、**Next** ボタンをクリックします。

データマッピングの 2 番目のステップは、列オプションを設定することです。

列オプションウィンドウの上部には、レポート用に選択したすべての列と使用可能なすべてのオプションを示す表が表示されます。

- 最初のフィールドは列の名前です。
- 2 番目のフィールドはデータの種類です。
- 3 番目のフィールドは **Visible** チェックボックスがマークされた場合に表示されます。チェックのオンオフでレポートの本文内で列が表示/非表示になります。
- 4 番目のフィールドは行ブレークフィールドです。行ブレークフィールドとして列を選択すると、選択した列の一意のエントリごとに新しいクロス集計行がレポートに挿入されます。
- 5 番目のフィールドは列ブレークフィールドです。列ブレークとして列を選択すると、レポートにデータベース内の一意のエントリごとに新しい列を作成させます。
- 6 番目のフィールドは列ブレーク値フィールドです。列ブレーク値として選択した列は、レポートに要約されるフィールドになります。
- 7 番目のフィールドは集計フィールドです。ドロップダウンメニューでその列に対して実行する集計を選択できます。実行できる集計操作は**合計（sum）、最大（maximum）、最小（minimum）、個数（count）、平均（average）、最初（first）、最後（last）、二乗和（sum of squares）、分散（variance）、標準偏差（standard deviation）、個数別（count distinct）**です。行ブレークまたは列ブレークとして選択された列は集計できません。
- 8 番目のフィールドは列ブレークの列の順序を選択することができます。この順序はクロス集計列が左から右へどのように描画されるかを決定します。Ordering オプションはソートされず、昇順でも降順でもありません。

実行される集計は、**Column Break Value** 列の集計によって異なります。たとえば、**MAX** 集計を使用すると、行の最大値が **Grand Total** 列に表示されます。

Keep Data Source Order:

データソースで指定されたデータの順序を保持する場合は、このオプションを選択します。この機能の

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

詳細については、データソースの順序の維持を参照してください。

Show Column Aggregation Grand Total:
このオプションを有効にすると、列がレポートに追加されます。列の各行は、対応する行のすべての値を単一の値に集約します。

Column Aggregation Subtotal:
小計列は、行全体を集計するのではなく、**Column Break** グループの各集計を集計する以外は、総計集計列と非常によく似ています。このオプションを使用すると、**Row Aggregation** の列の無効や、列の位置（列ブレークグループの右または左）を設定できます。

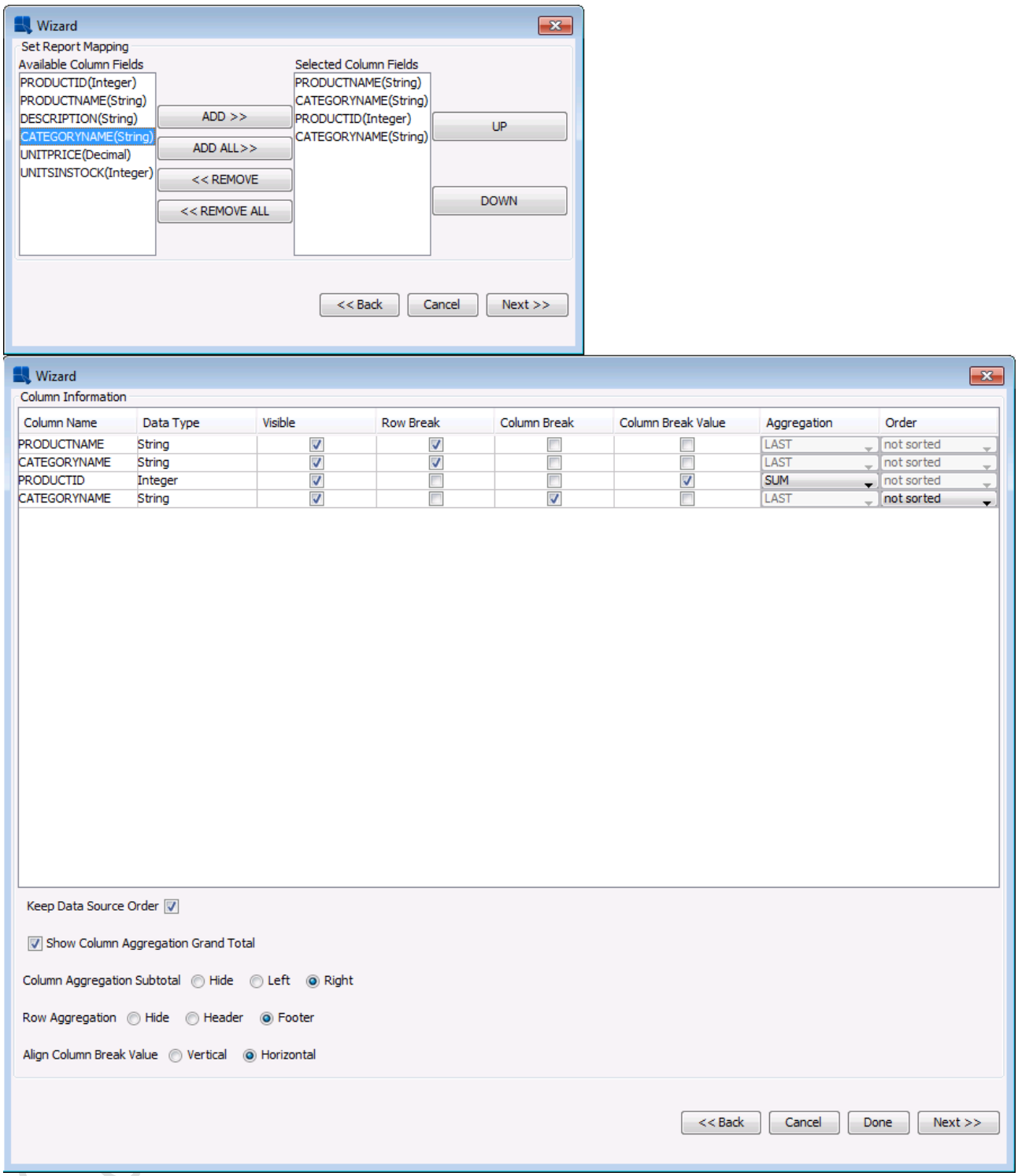
Row Aggregation:
これは、総合計列にも非常によく似ています。主な違いは、このオプションは行を集約するのではなく、列を集約することです。また、レポートに列を追加するのではなく、行を追加します。このオプションを使用して、**Row Aggregation** 行を無効にするか調整をします（テーブルの上またはテーブルの下）。

Align Column Break Value:
複数の **Column Break Value** 列を選択する場合は、このオプションを使用して値を行または列に並べるかどうかを選択します。

たとえば、次のクロス集計マッピングがあるとします。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



クロス集計フィールド固定フィールドマッピング

CATEGORYNAME と **PRODUCTNAME** を **RowBreak** として設定し、**PRODUCTID** を **SUM** の集計列を持つ **Column Break Value** フィールド SUM とし、および **CATEGORYNAME** を **Show Aggregation Grand Total** オプションを有効にした列ブレークフィールドとして、クロス集計レポートセクションの先頭からレポートを生成します。

Data メニューの **Change Data Mapping** オプションを使用すると、クロス集計の設定を簡単に変更できます。クロス集計レポートのマッピングダイアログが再度開きます。この機能の詳細は、データマッ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ピングの変更を参照してください。

次のテーブルに、クロス集計レポートのさまざまな書式を示します。

	列小計	行集計	列ブレーク値の整列	テンプレート
レポート A	右	フッター	水平	テンプレート A
レポート B	右	ヘッダー	水平	テンプレート B
レポート C	左	フッター	水平	テンプレート C
レポート D	左	ヘッダー	垂直	テンプレート D
レポート E	左	なし	垂直	テンプレート E

3.4.3.2 データ転置

クロス集計レポートには、ユーザが転置レポートを作成できる独自の機能もあります。データ転置によって、入力データが基本的に 90 度回転したレポートプレゼンテーションを作成できます。列は行になり、行は列になります。たとえば、次のデータセットがあります。

Region	Forcast	Sales
East	24100	25050
Midwest	23110	22400
South	22300	26500
West	18750	19220

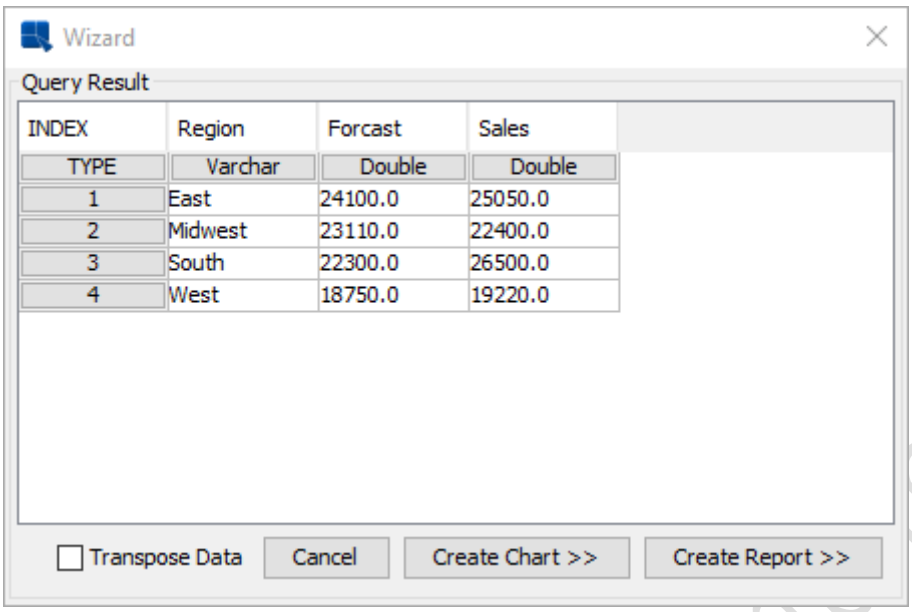
データ転置機能は上のテーブルから次のようなレポートを作成します。

	East	Midwest	South	West
Forcast	24100	23110	22300	18750
Sales	25050	22400	26500	19220

データを転置するには、レポートのデータソースを選択すると最初に表示される結果画面の下部にある **Transpose Data** オプションをクリックします。次のダイアログは、上の例のサンプルデータからの初期結果セットを示しています。

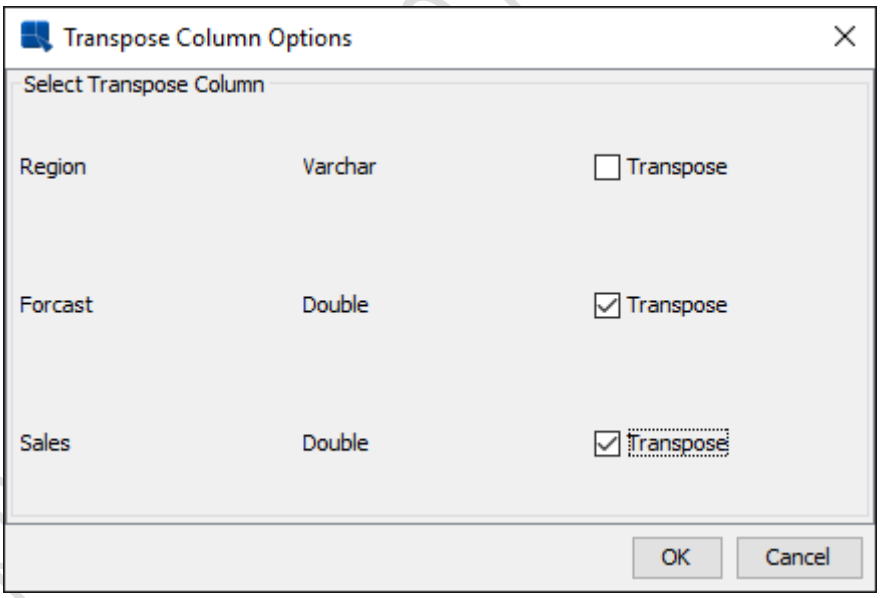
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Data Table ダイアログ

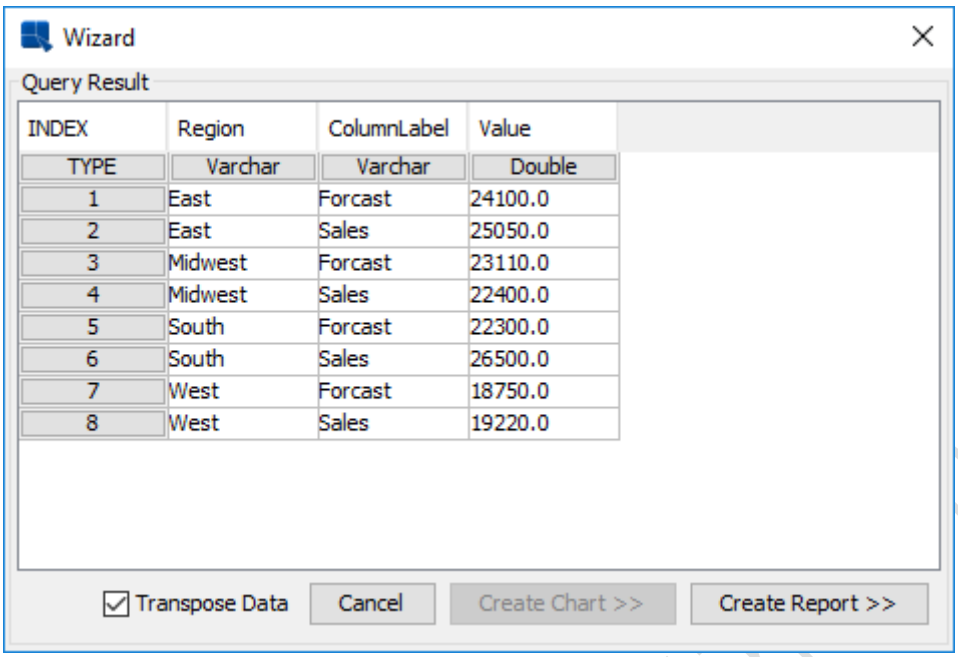
下のオプションをチェックすると、転置したい列を選択するための新しいダイアログが表示されます。この例では、**Forecast** カラムと **Sales** カラムを選択します。転置を実行するには、選択した列のデータ型が同じである必要があります。



Select Transpose ダイアログ

選択したら、**OK** ボタンをクリックして変更を適用します。Data Table ダイアログに転置が表示されます。Sales ヘッダーと Forecast ヘッダーは **ColumnLabel** という列に転記され、その値は **Value** という列にマージされます。この結果セットは、クロス集計レポートを使用してマッピングできます。ここで、**ColumnLabel** は行ブレーク、**Region** は列ブレーク、**Value** は列転置値で、最終転置レポートレイアウトを作成します。

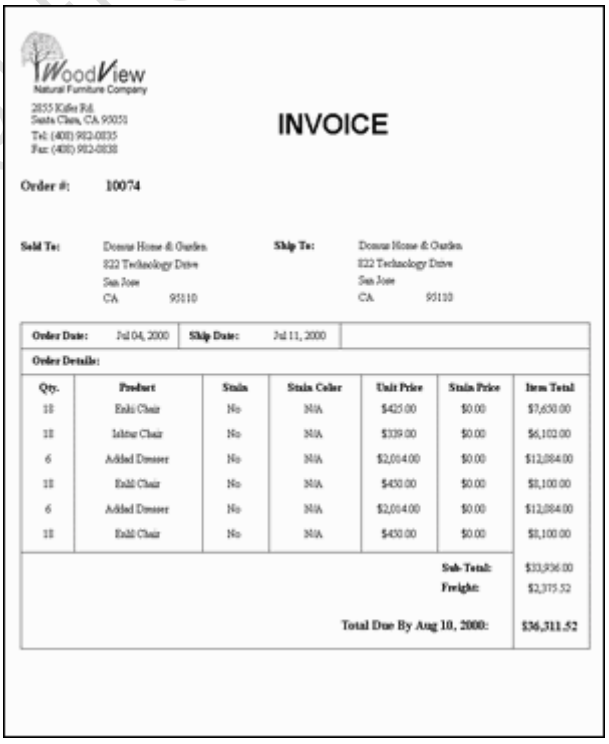
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



転置後のデータ

3.4.4 マスタ&詳細レポート

マスタ＆詳細レポートは、マスタフィールドに従ってグループ化された一連のテーブルデータです。このレポートタイプは、データテーブル内に 1 対 1 の関係を持つフィールドがある場合に最も一般的に使用されます。これの良い例は請求書です。データベース内の各注文番号には、顧客情報と価格情報付きのいくつかの項目があります。マスタ＆詳細レポートは、注文番号に従って情報をグループ化するために使用されます。



マスタ&詳細レポート

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.4.4.1 データマッピング

このタイプのデータマッピングの最初の手順は、Report ウィザードの **Set Mapping** ウィンドウからレポートに含める列を選択することです。ウィンドウの左側には、クエリまたはデータファイルから使用可能なすべての列がリストされ、右側にはレポートに表示される列がリストされます。列名をクリックして選択（**Ctrl +クリック**で複数選択）し、**ADD** または **REMOVE** ボタンを押して追加または削除します。このウィンドウで列を選択する順序は、レポートに表示される順序です。

データマッピングの 2 番目のステップは、列オプションを設定することです。マスタ＆詳細レポートには 3 つの列オプションがあります。列オプションウィンドウの下部にドロップダウンメニューがあります。**primary key** とラベルされています。ドロップダウンメニューには、レポート用に選択したすべての列が含まれます。列を **primary key** として選択すると、その列に従ってレポートがグループ化されます。選択した列の値が変更されるたびに、新しいグループが作成されます。

列オプションウィンドウの上部には、レポート用に選択したすべての列と使用可能なオプションを示す表が表示されます。最初のフィールドはカラム名を表示し、2 番目のフィールドはデータタイプを表示し、3 番目のフィールドは **Visible** とマークされたチェックボックスを表示します。このボックスをオンまたはオフにすると、レポートの本文内で列が表示/非表示になります。4 番目のフィールドは **Master field** フィールドです。列をマスタフィールドとして選択すると、レポートのデータセクションではなく列ヘッダーに列の値が配置されます。

データマッピングウィンドウの下部にあるチェックボックスを使用すると、レポートのレイアウトを変更できます。**Side-By-Side** チェックボックスをオンにすると、マスタセクションがその上に表示されるのではなく詳細セクションの横に表示されるようにレポートが配置されます。

たとえば、次のデータテーブルがあるとします。

Order#	Customer Name	Product	Unit Price	Quantity
12	Paul Campbell	Chair	$24.95	4
12	Paul Campbell	Table	$127.50	1
14	Sally Hayes	Cabinet	$227.25	2
14	Sally Hayes	Chair	$24.95	2
14	Sally Hayes	Table	$127.50	1

Order#を **primary key** として設定し、**Side-By-Side** を使用せずに **Customer Name** を **Master field** として入力すると次のレポートが表示されます。

Order#	12	
Customer Name	Paul Campbell	
Product	Unit Price	Quantity
Chair	$24.95	4

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

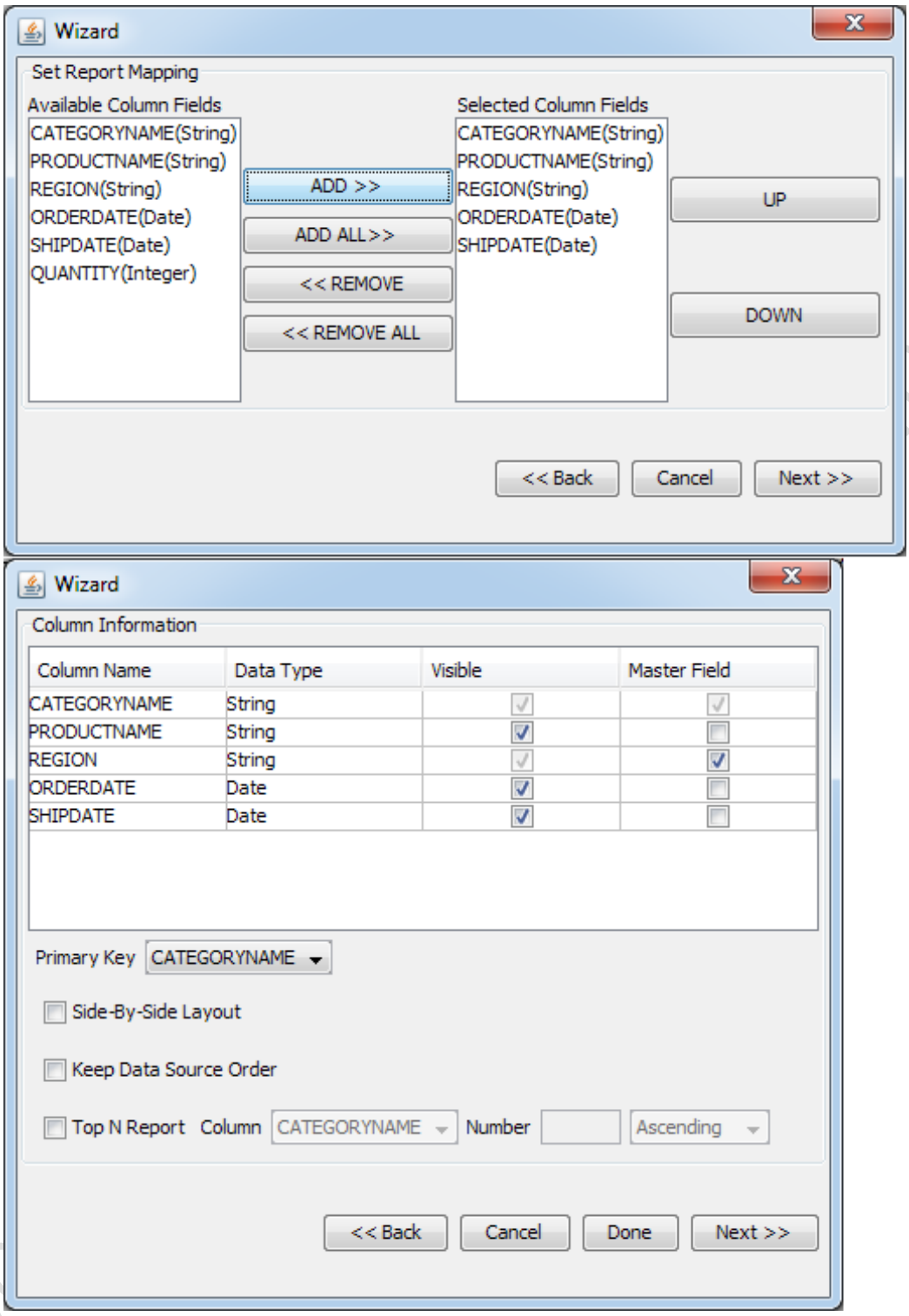
Table	$127.50	1
Order#	**14**	
Customer Name	**Sally Hayes**	
Product	**Unit Price**	**Quantity**
Cabinet	$227.25	2
Chair	$24.95	2
Table	$127.50	1

しかし、同じレポートでも **Side-By-Side** 機能を有効にすると次のようなレポートになります。

		Product	UnitPrice	Quantity
Order#	**12**	Chair	$24.95	4
Customer Name:	**Paul Campbell**	Table	$127.50	1
Order#	**14**	Cabinet	$227.25	2
Customer Name:	**Sally Hayes**	Chair	$24.95	2
		Table	$127.50	1

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



マスタ&詳細レポート　マッピング

データソース内のデータ順序の指定を維持したい場合、**Keep Data Source Order** オプションを選択してください。この機能の詳細については[データソースの順序の維持]を参照してください。

3.4.4.1.1 上位 N レポート

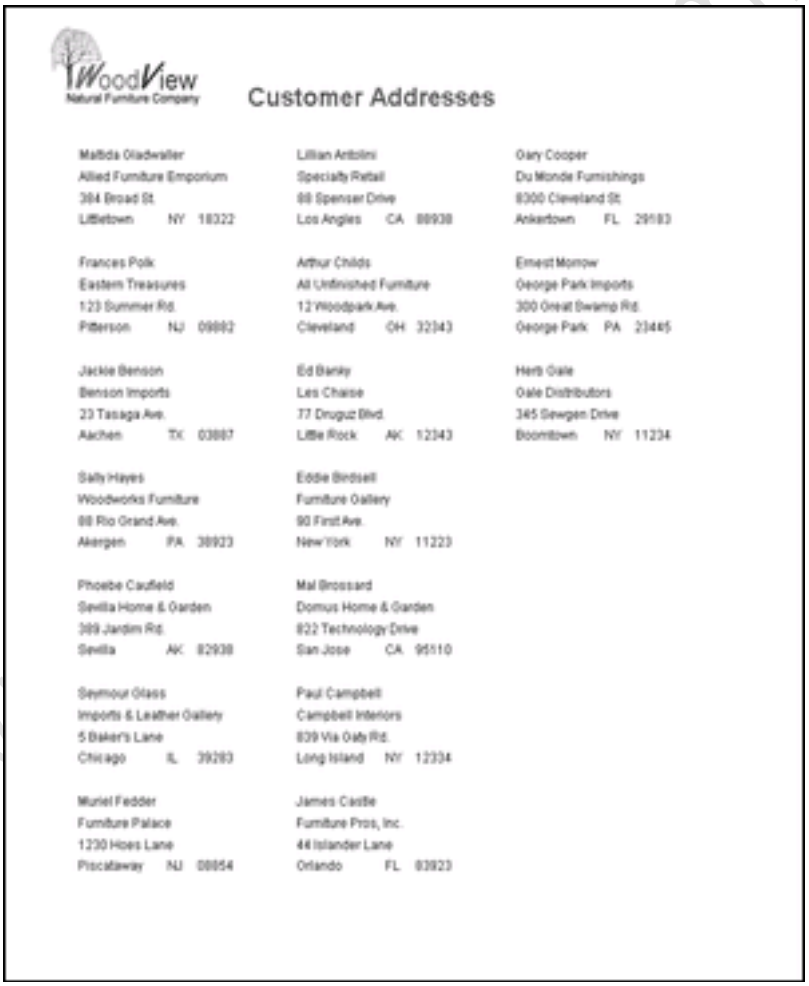
マスタ&詳細形式を使用して上位 N レポートを作成することもできます。上位 N レポートでは、レポートの特定の列に基づいて最も高い値のセットが表示され、表示されます。マスタ＆詳細レポートでは、各グループの最高値を表示するように指定することもできます。たとえば、売上高に基づいて各地域の

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

トップ 5 の顧客を表示することができます。これを行うには、データマッピングウィンドウの下部にある **Top N Report** チェックボックスをオンにします。次に、メジャーとして使用する列を指定できます（上記の例では、合計売上列）、取得する番号（つまり 10 または 20 など）、および列を昇順で並び替えて表示するか、降順で並び替えて表示するかを指定できます。マスタ＆詳細レポートは、指定された列の最高値でソートされた指定されたレコード数（または行ブレークごとに指定されたレコード数）を返します。

3.4.5 メーリングラベルレポート

メーリングラベルレポートはシンプルカラムナレポートに似ています。ただし、郵送用ラベルを作成できるようにデータが事前に整理されています。データはデータテーブル内で垂直方向に配置され、自動的に折り返され、郵送ラベルのアドレスリストを示すレポートを簡単に作成することができます。


WoodView
Natural Furniture Company

Customer Addresses

Matilda Oladweller
Allied Furniture Emporium
384 Broad St.
Littletown NY 18322

Lillian Antolini
Specialty Retail
88 Spenser Drive
Los Angles CA 88938

Gary Cooper
Du Monde Furnishings
8300 Cleveland St.
Ankertown FL 29183

Frances Polk
Eastern Treasures
123 Summer Rd.
Pitterson NJ 09882

Arthur Childs
All Unfinished Furniture
12 Woodpark Ave.
Cleveland OH 32343

Ernest Morrow
George Park Imports
300 Great Swamp Rd.
George Park PA 23445

Jackie Benson
Benson Imports
23 Tasaga Ave.
Aachen TX 03887

Ed Banky
Les Chaise
77 Druguz Blvd.
Little Rock AK 12343

Herb Gale
Gale Distributors
345 Sewgen Drive
Boomtown NY 11234

Sally Hayes
Woodworks Furniture
88 Rio Grand Ave.
Akergen PA 38923

Eddie Birdsell
Furniture Gallery
90 First Ave.
New York NY 11223

Phoebe Caufield
Sevilla Home & Garden
389 Jardim Rd.
Sevilla AK 82938

Mal Brossard
Domus Home & Garden
822 Technology Drive
San Jose CA 95110

Seymour Glass
Imports & Leather Gallery
5 Baker's Lane
Chicago IL 39283

Paul Campbell
Campbell Interiors
839 Via Oats Rd.
Long Island NY 12334

Muriel Fedder
Furniture Palace
1230 Hoes Lane
Piscataway NJ 08854

James Castle
Furniture Pros, Inc.
44 Islander Lane
Orlando FL 03823

メーリングラベルレポート

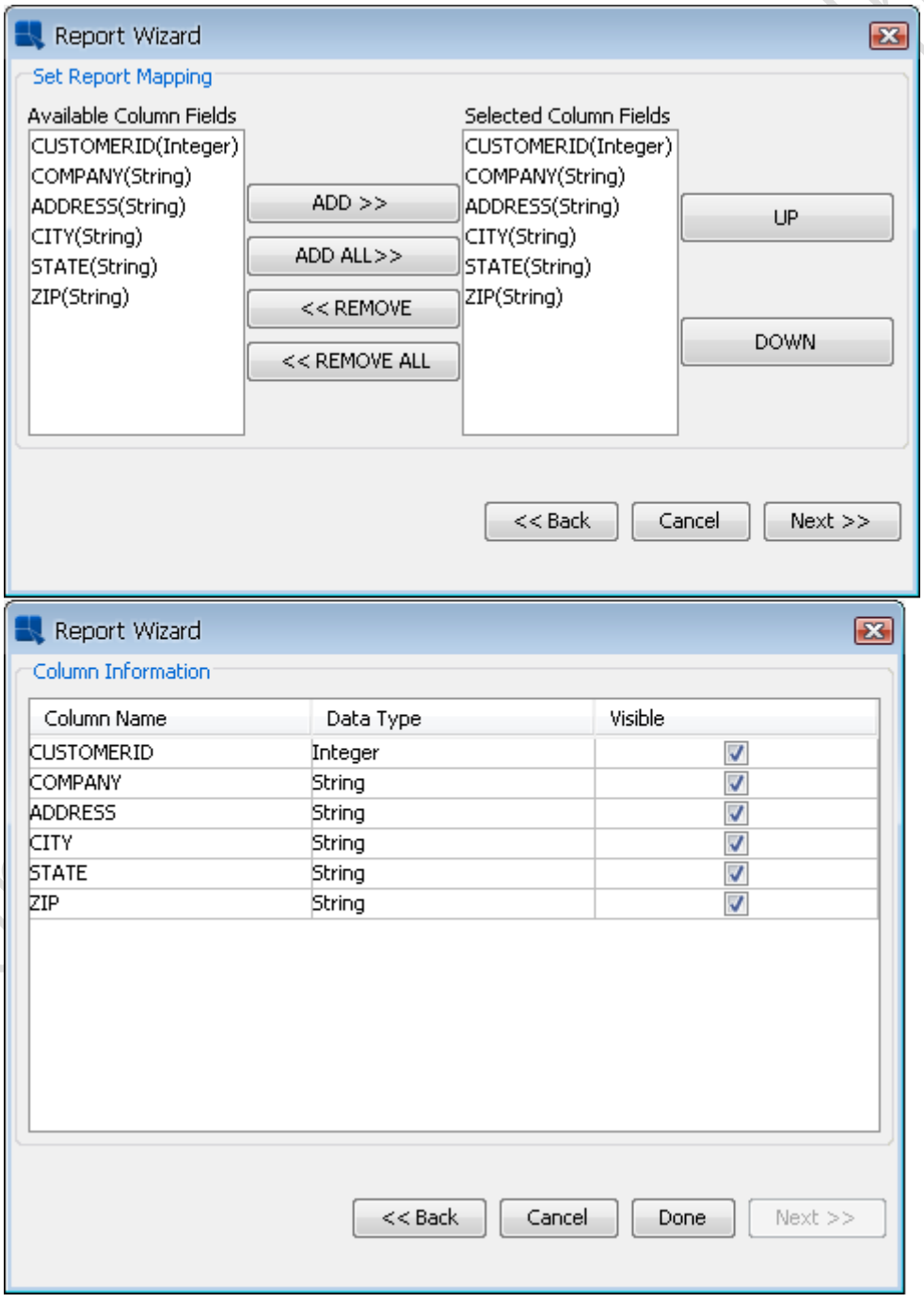
3.4.5.1 データマッピング

このタイプのデータマッピングは、シンプルカラムナレポートの場合とまったく同じです。まず、Report ウィザードの **Set Report Mapping** ウィンドウから、レポートに含める列を選択します。ウィンドウの左側には、クエリまたはデータファイルから使用可能なすべての列がリストされ、右側にはレポートに表示される列がリストされます。カラム名をクリックして選択し（複数選択の場合は **SHIFT +クリック**）、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ADD または **REMOVE** ボタンを押して追加または削除します。このウィンドウで列を選択する順序は、レポートに表示される順序です。

データマッピングの 2 番目のステップは、列オプションを設定することです。メーリングラベルレポートには利用可能なオプションが 1 つしかありません。このウィンドウには、レポート用に選択したすべての列と使用可能なすべてのオプションを示す表が表示されます。最初のフィールドは列名を表示し、2番目のフィールドはデータ型を表示し、3番目のフィールドは **Visible** チェックボックスを表示します。このボックスをオンまたはオフにすると、レポートの本文内で列が表示または非表示になります（デフォルトではすべての列が表示されます）。



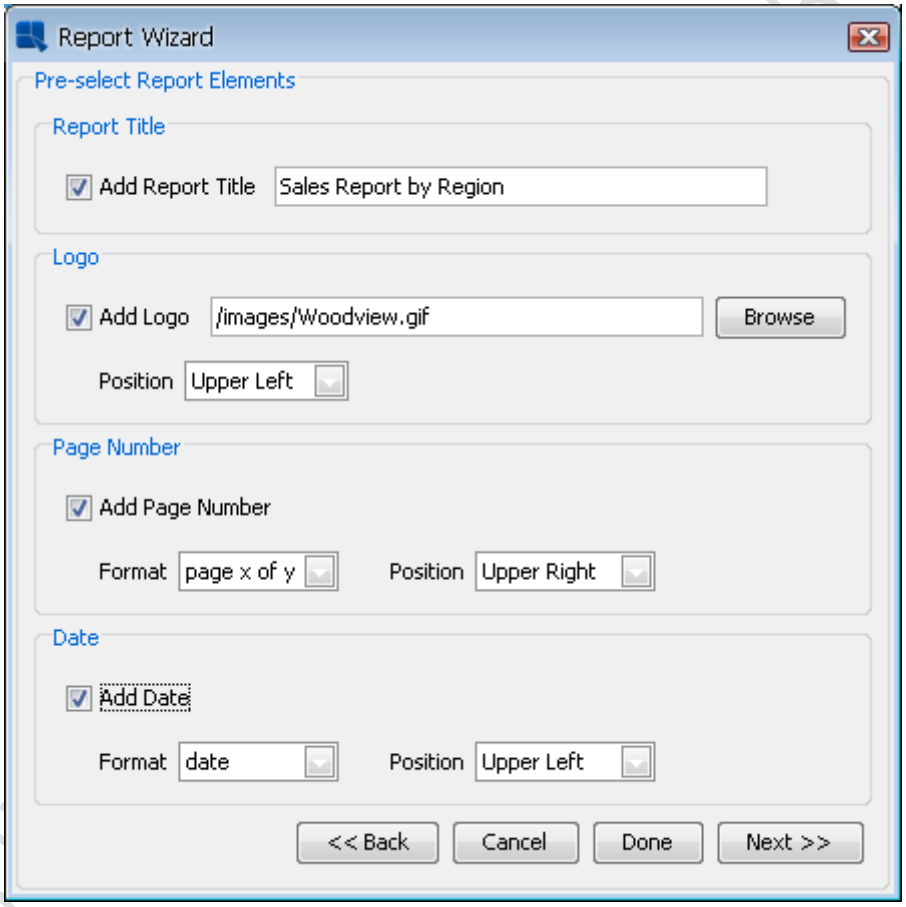
メーリングラベルレポートマッピング

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.4.6 フォーマットオプションの追加

マッピングオプションの指定が完了したら、ウィザードを終了してレポートの編集/変更を開始できます。ウィザードを終了するには、最後のデータマッピングウィンドウで **Done** ボタンをクリックします。これにより、メインデザイナーウィンドウに戻り、マッピング仕様に基づいて空の（書式なしの）レポートが生成されます。

EspressReport は、書式なしのレポートを生成するのではなく、ウィザードでいくつかの要素をレポートに自動的に配置、レポート要素の既定のスタイルを提供するためのオプションをいくつか追加しています。ウィザードで続行するには、最後のデータマッピングウィンドウで **Next** ボタンをクリックします。これにより、レポートにいくつかの要素を追加するように促すダイアログが表示されます。



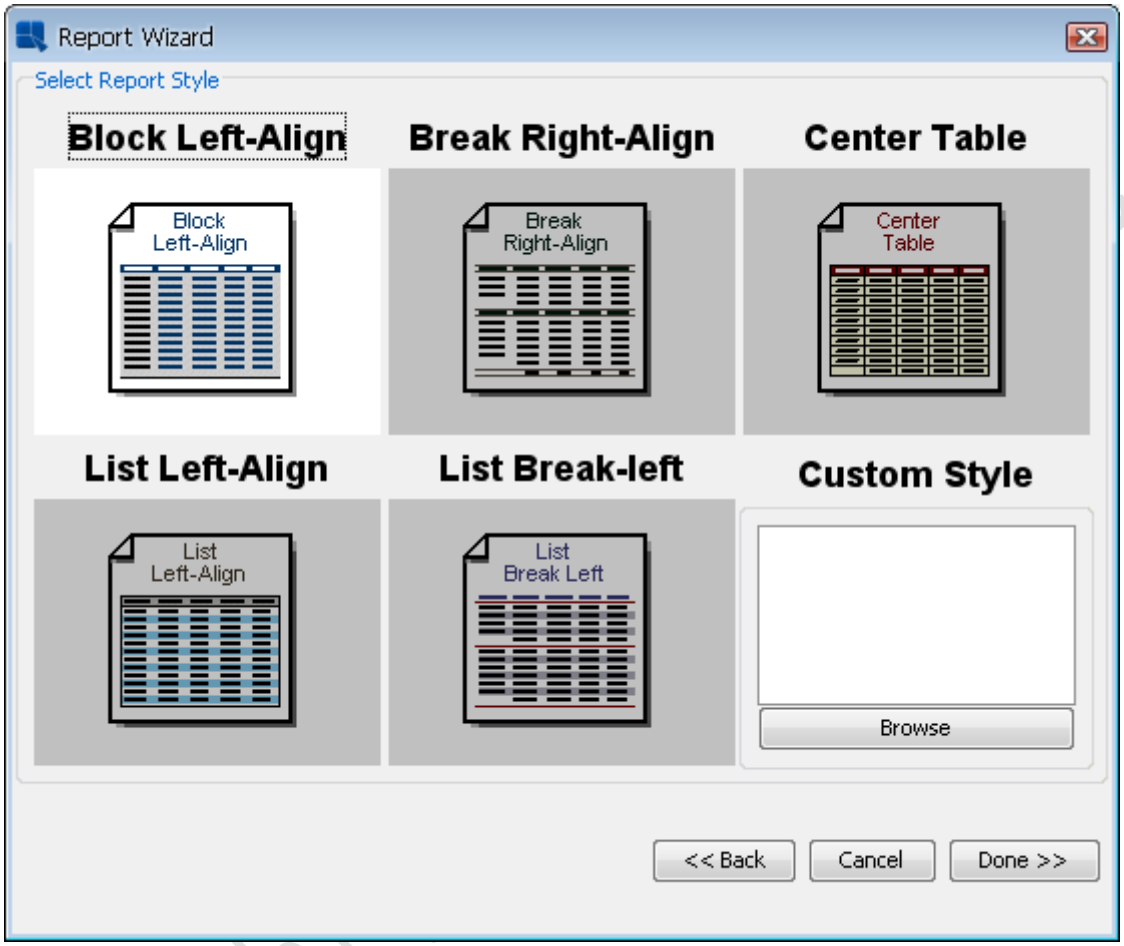
Add Elements ダイアログ

このダイアログでは、レポートヘッダーのレポートタイトルとロゴを指定できます。レポートタイトルを追加することを選択すると、タイトルテキストを入力することができます。ロゴを追加するように指定した場合は、ファイルまたは URL パスを使用してイメージファイルの場所を指定できます。また、ロゴをページの右上または左上隅に配置するかどうかを指定することもできます。

ページのヘッダーとフッターにページ番号と日付を追加するように指定することもできます。ページのヘッダーまたはフッターのいずれかで要素の書式と位置を選択し、それらをページの右、左または中央に揃えることができます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

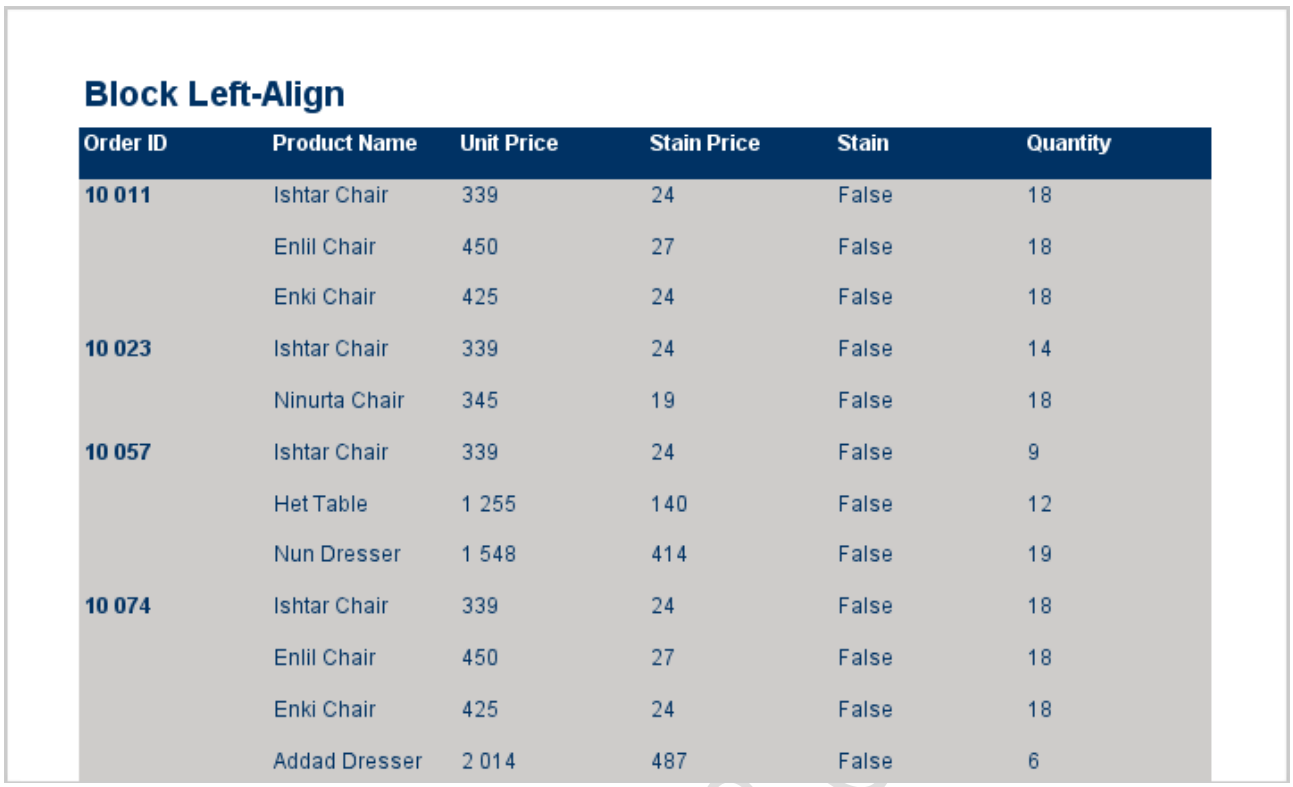
要素の指定が完了したら、**Done** ボタンを再度クリックしてウィザードを終了し、追加された要素でレポートの編集/修正を開始できます。**Next** をクリックすると、Report ウィザードの最後のダイアログに戻ります。このダイアログでは、レポートのスタイルを選択できます。



Report Style ダイアログ

ここから、事前定義されたスタイルまたはレポートに使用するカスタムスタイルを選択できます。スタイルを選択すると、レポート要素にフォーマットが適用され、新しいレポート要素のデフォルト属性が変更されます。以下は、同じサマリレポートに適用された場合の事前定義されたレポートスタイルの例です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

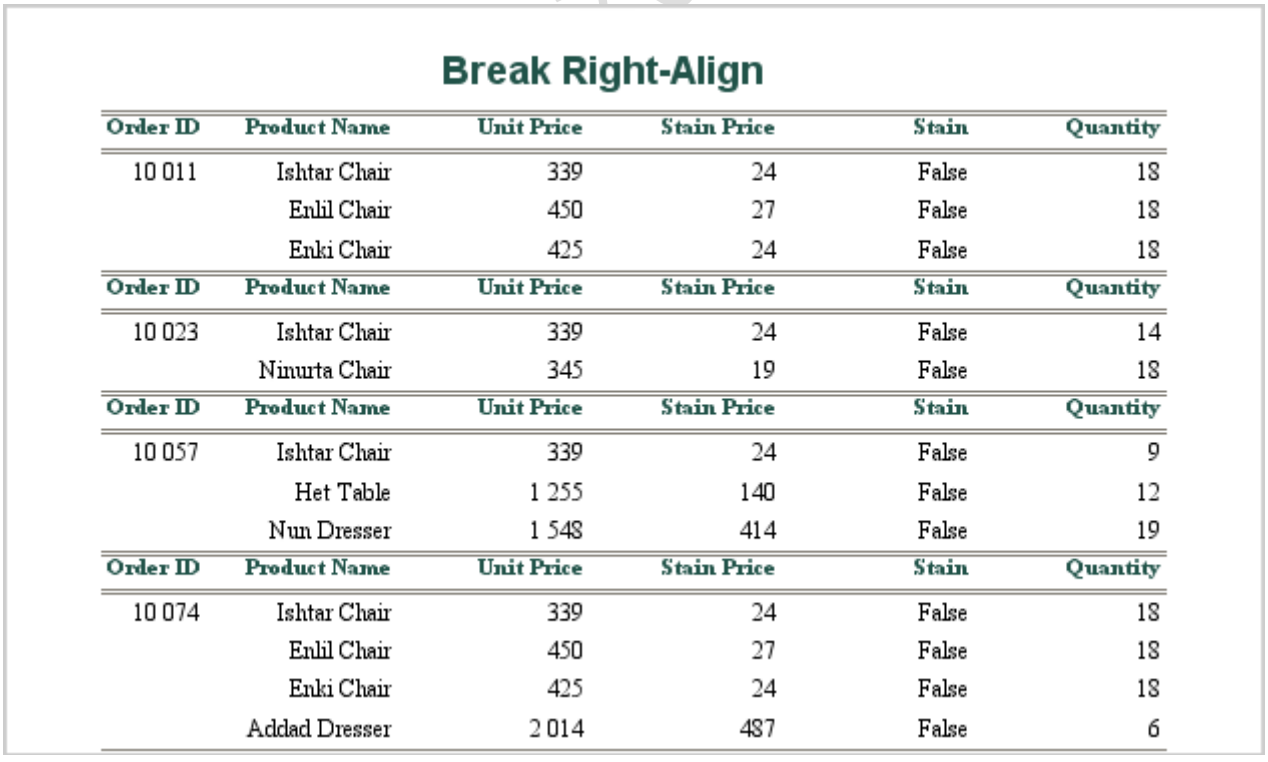


Block Left-Align

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 011	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
10 023	Ishtar Chair	339	24	False	14
	Ninurta Chair	345	19	False	18
10 057	Ishtar Chair	339	24	False	9
	Het Table	1 255	140	False	12
	Nun Dresser	1 548	414	False	19
10 074	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
	Addad Dresser	2 014	487	False	6

Block Left-Align

このスタイルでは、テキストの配置を左に設定し、青いヘッダーで灰色の背景にレポートを描画します。



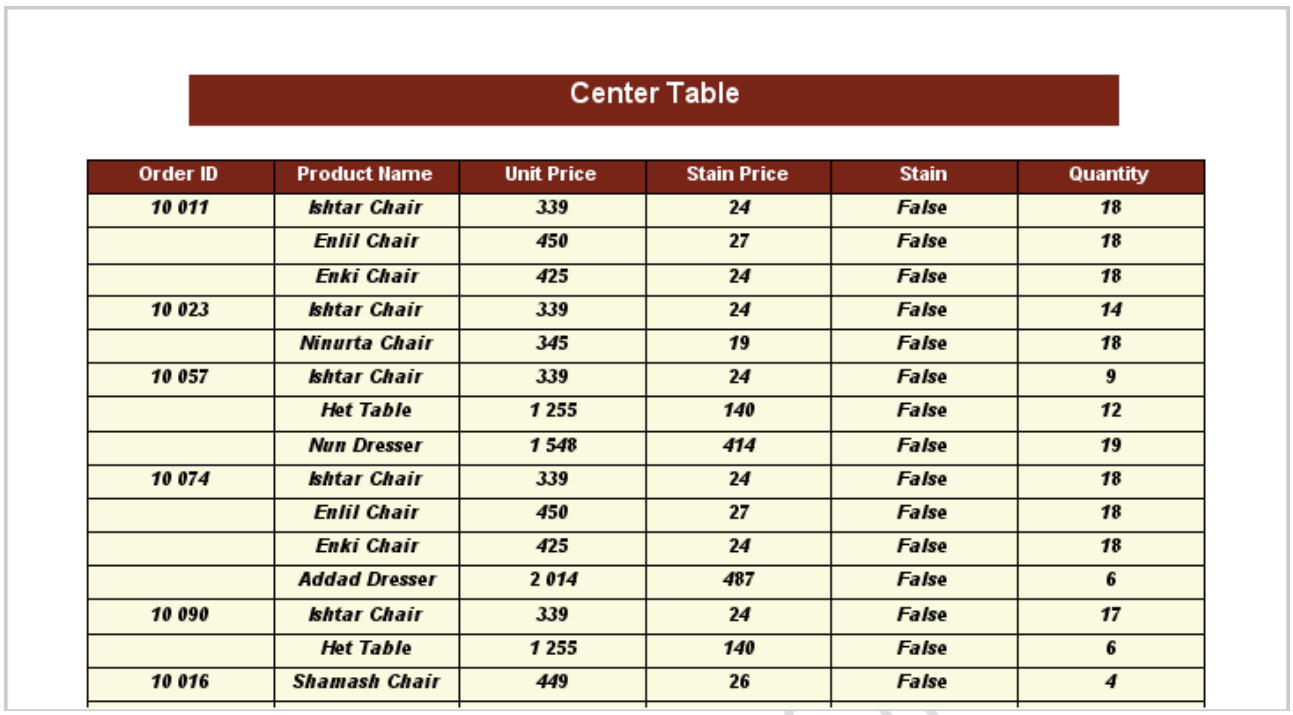
Break Right-Align

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 011	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
Order ID	**Product Name**	**Unit Price**	**Stain Price**	**Stain**	**Quantity**
10 023	Ishtar Chair	339	24	False	14
	Ninurta Chair	345	19	False	18
Order ID	**Product Name**	**Unit Price**	**Stain Price**	**Stain**	**Quantity**
10 057	Ishtar Chair	339	24	False	9
	Het Table	1 255	140	False	12
	Nun Dresser	1 548	414	False	19
Order ID	**Product Name**	**Unit Price**	**Stain Price**	**Stain**	**Quantity**
10 074	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
	Addad Dresser	2 014	487	False	6

Block Right-Align

このスタイルでは、テキストの配置を右に設定し、ヘッダーを区切る線を描画します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

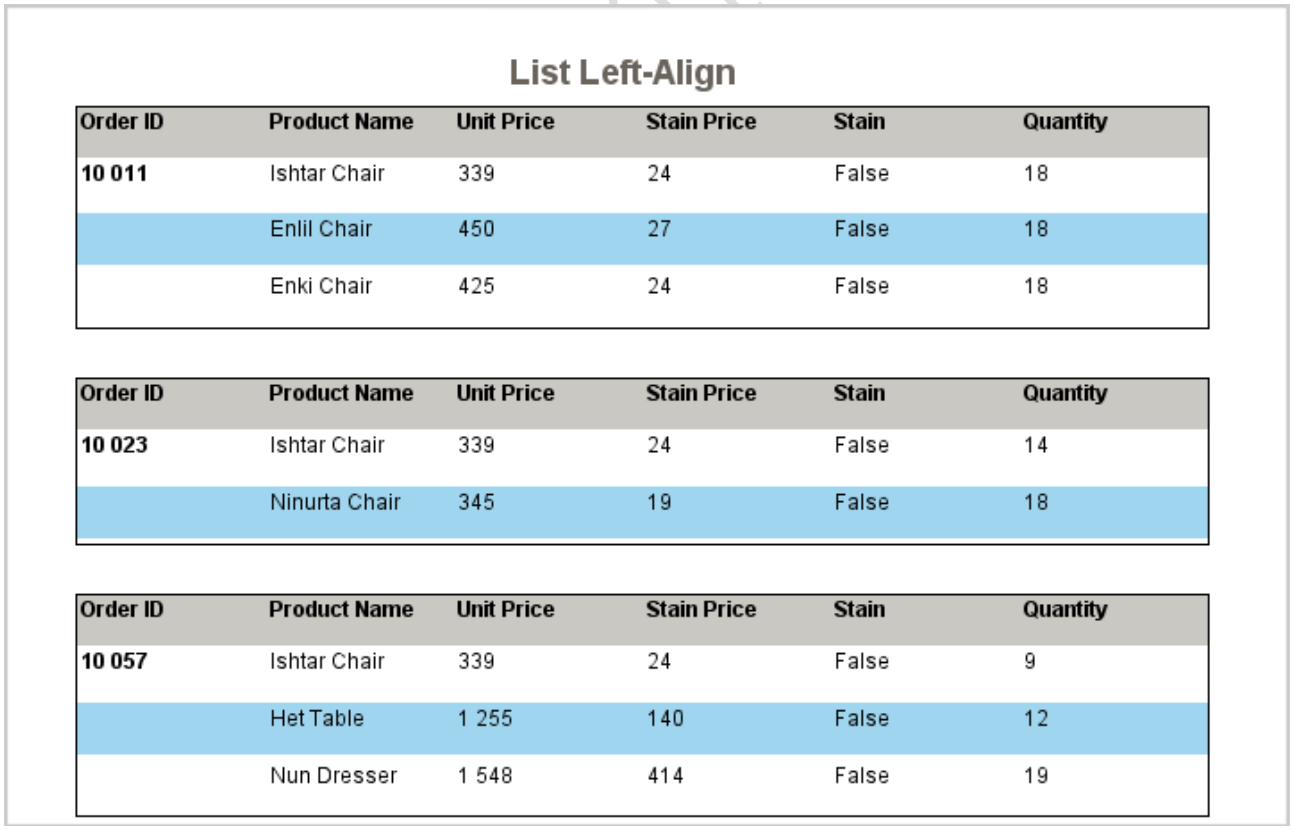


Center Table

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 011	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
10 023	Ishtar Chair	339	24	False	14
	Ninurta Chair	345	19	False	18
10 057	Ishtar Chair	339	24	False	9
	Het Table	1 255	140	False	12
	Nun Dresser	1 548	414	False	19
10 074	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
	Addad Dresser	2 014	487	False	6
10 090	Ishtar Chair	339	24	False	17
	Het Table	1 255	140	False	6
10 016	Shamash Chair	449	26	False	4

Center Table

このスタイルでは、テキストをセンタリングし、レポートセルの周囲にテーブルの境界線を描画します。



List Left-Align

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 011	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18

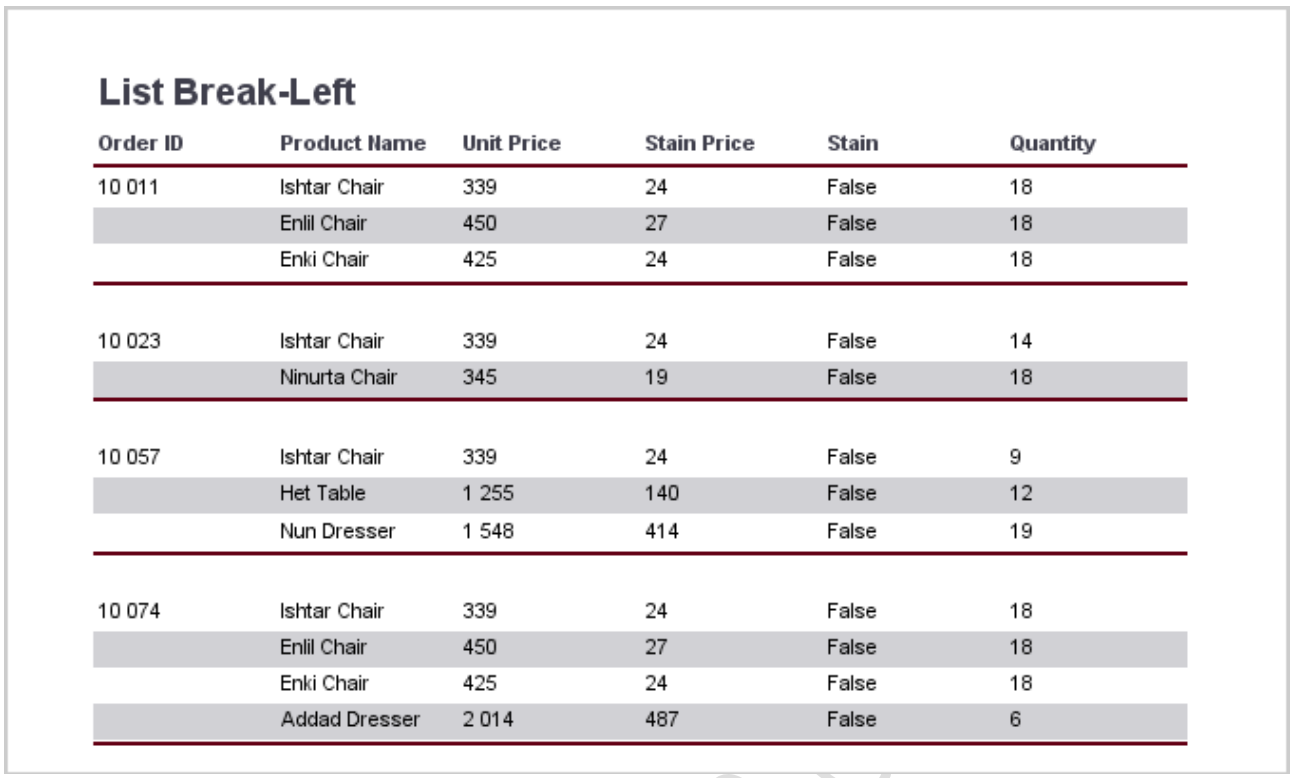
Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 023	Ishtar Chair	339	24	False	14
	Ninurta Chair	345	19	False	18

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 057	Ishtar Chair	339	24	False	9
	Het Table	1 255	140	False	12
	Nun Dresser	1 548	414	False	19

List Left-Align

このスタイルではテキストの配置を左に設定し、行ごとに色が交互に表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



List Break-Left

Order ID	Product Name	Unit Price	Stain Price	Stain	Quantity
10 011	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
10 023	Ishtar Chair	339	24	False	14
	Ninurta Chair	345	19	False	18
10 057	Ishtar Chair	339	24	False	9
	Het Table	1 255	140	False	12
	Nun Dresser	1 548	414	False	19
10 074	Ishtar Chair	339	24	False	18
	Enlil Chair	450	27	False	18
	Enki Chair	425	24	False	18
	Addad Dresser	2 014	487	False	6

List Break-Left

このスタイルではテキストの配置を左に設定し、列ヘッダーを区切るための行を追加し、行ごとに色を交互に描画します。

レポートスタイルはレポート要素の外観プロパティにのみ影響し、以前に選択されたレポートタイプやデータマッピングオプションは変更されません。スタイルを指定したら、**Done** ボタンをクリックしてウィザードを終了し、メインデザイナーウィンドウに進みます。

3.4.6.1 スタイルのカスタマイズ

EspressReport に含まれている 5 つの定義済みスタイルに加えて、ユーザは独自のスタイル定義を作成できます。スタイル定義は**.stl** 拡張子で保存された特別なバージョンのレポートテンプレートです。ユーザは、任意のレポートテンプレートを使用してスタイル定義を作成できます。

3.4.6.1.1 カスタムスタイルの作成

カスタムスタイルを作成する最初のステップは、グローバルフォーマットを設定することです。これにより、ユーザがレポートに挿入するすべての要素のデフォルトのスタイルを制御できます。グローバルフォーマットの詳細については、グローバル書式設定を参照してください。

カスタムスタイルを作成する 2 番目の手順は、レポートの各セクションの要素のスタイルを選択することです。これを行うには、スタイルが適用されたときにセクション内のすべての要素に適用する書式のレポート要素を選択し、ポップアップメニューから **Set Attributes as Section Style** を選択します。

属性の設定が完了したら、**save as** ダイアログの **Create Style** オプションをオンにして、レポートテンプレートをカスタムスタイルとして保存します。

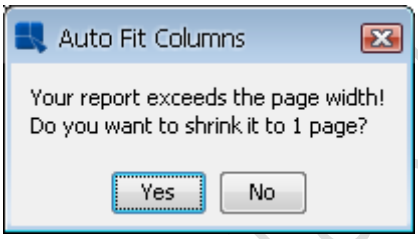
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.4.6.1.2 カスタムスタイルの適用

レポートを作成するときにカスタムスタイルを選択するには、**report style** ダイアログで **Use Custom Style** オプションをオンにし、使用するスタイル（**.stl**）ファイルを指定します。スタイルが適用されると、グローバルフォーマットが適用され、各セクションに定義された属性が新しいレポートの対応するセクションの要素に適用されます。さらに、そのスタイルで定義された線や画像が新しいレポートにも適用されます。

3.4.7 列をページ幅に合わせる

Report ウィザードを終了するとき、作成されたレポートはデフォルトのページ幅（8.5 "）よりも広くなる場合があります。この場合、警告が表示されます。



Auto Fit Columns ダイアログ

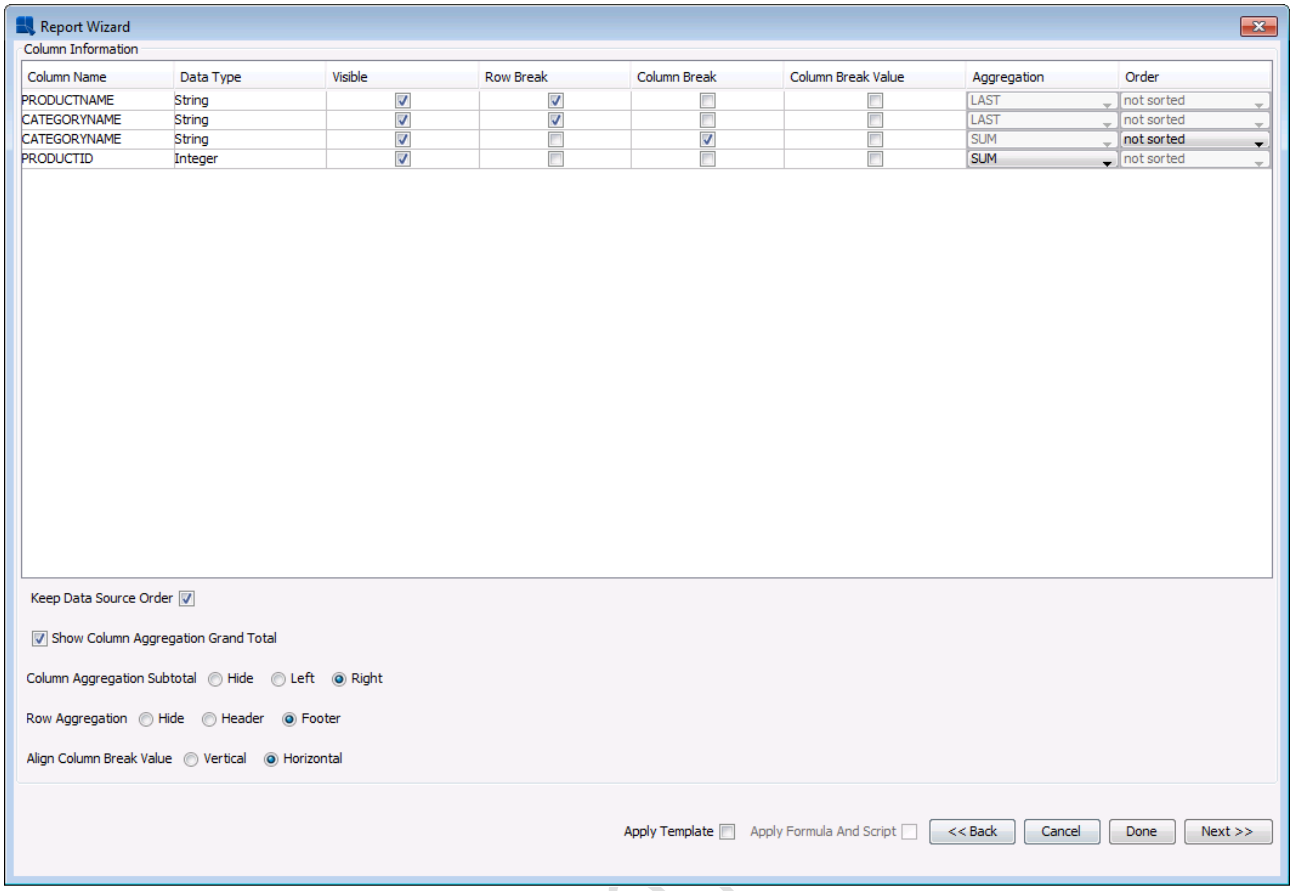
ダイアログには、ページ内に収まるようにレポートを縮小するオプションがあります。**Yes** を選択すると、すべての列がページの幅に合わせて自動的に縮小されます。ページ境界は、デザインウィンドウ内の垂直バーで示されます。列をページ境界と重複しないように選択すると、レポートをプレビューするときにオーバーフローが新しいページに印刷されます。**Format** メニューから **Auto Fit Columns** を選択して、レポートを編集するときにいつでも列を自動調整できます。ページ幅は、オプションメニューからページ設定を選択することによって調整することもできます。

3.4.8 データマッピングの変更

データマッピングが完了すると、レポートの大まかなバージョンが生成されます。正しくない場合は、**Data** メニューから **Change Data Mapping** を選択するか、ツールバーの **Change Data Mapping** ボタンをクリックして、データマッピングを変更できます。これにより、Report ウィザード（データマッピングから始まります）に戻り、レポートタイプとデータマッピングの変更に戻ることができます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



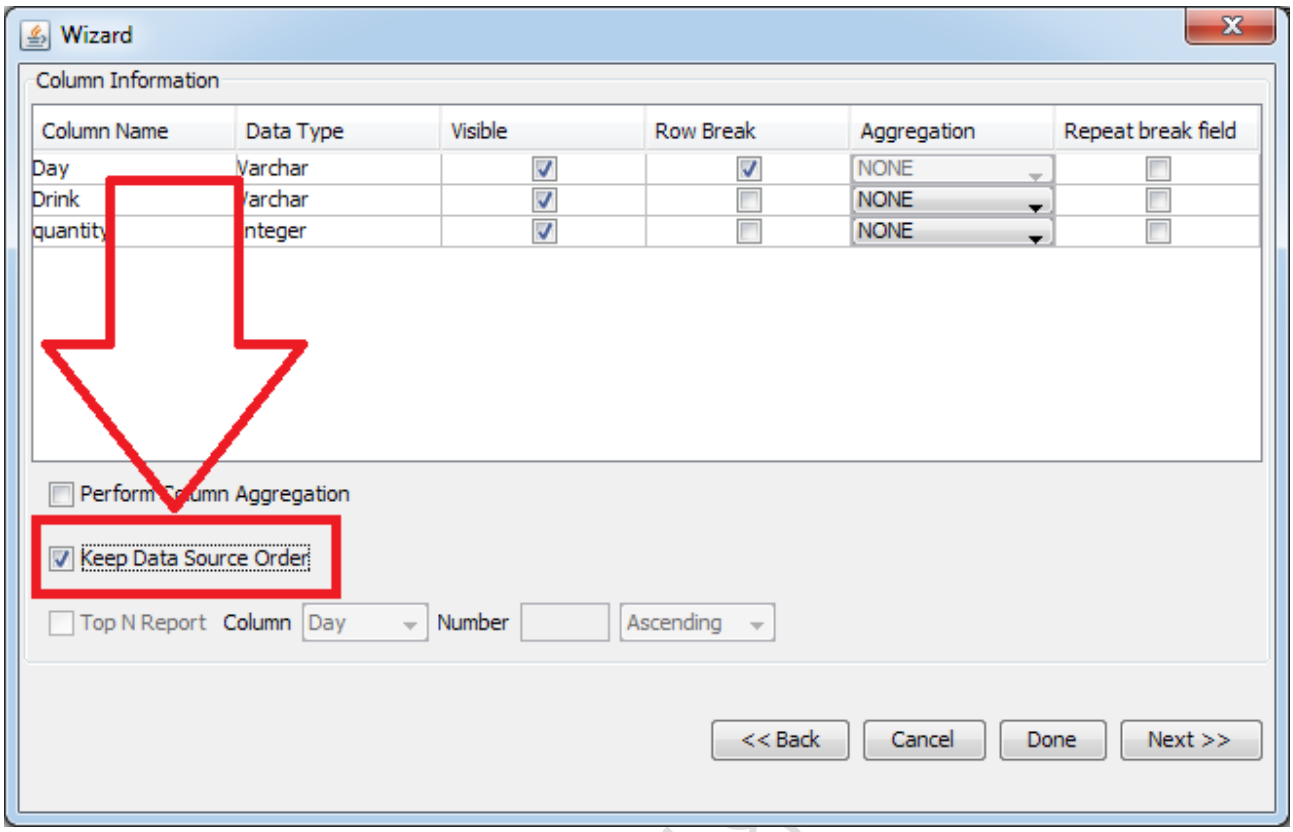
データマッピング変更ウィンドウ

データマッピングを変更すると、レポートウィザードの最後の画面の下部に 2 つのチェックボックスが表示されます。1 つは **Apply Template** ともう 1 つは **Apply Formula and Script** とマークされています。最初のボックスをチェックすると、レポートの書式が保持されます。ただし、ラベルと数式が正しくない可能性があります。このボックスのチェックを外すと、新しいデータマッピングで空のレポートが作成されます。多くのレポートオブジェクトをフォーマットしなかった場合、このボックスのチェックを外すことをお勧めします。これにより、新しいデータマッピングが完全に正確であることが保証されます。2 番目のボックスでは、**Apply Template** オプションを選択すると、適用するデータテーブルセクションに配置された関数とスクリプトを含めることができます（他のセクションの式とスクリプトが常に適用されます）。特定の列の列数やデータ型を変更した場合、関数やスクリプトが機能しなくなることがあります。

3.4.9 データソースの順序を維持

サマリブレーク、クロス集計、マスタ＆詳細レポートでは、メモリ最適化クエリ処理が使用されますが、残念ながらデータの順序が変更される可能性があります。言い換えると、レポート内のデータが、データソース内のデータとは異なる順序で表示されることがあります。これが発生しないようにするには、レポートマッピングダイアログで **Keep Data Source Order** オプションを選択します。これにより、メモリ最適化クエリ処理が無効になります。つまり、データが最適化されずにレポートに渡されます（つまり、元のデータ順序が保持されます）。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Keep Data Source order オプション

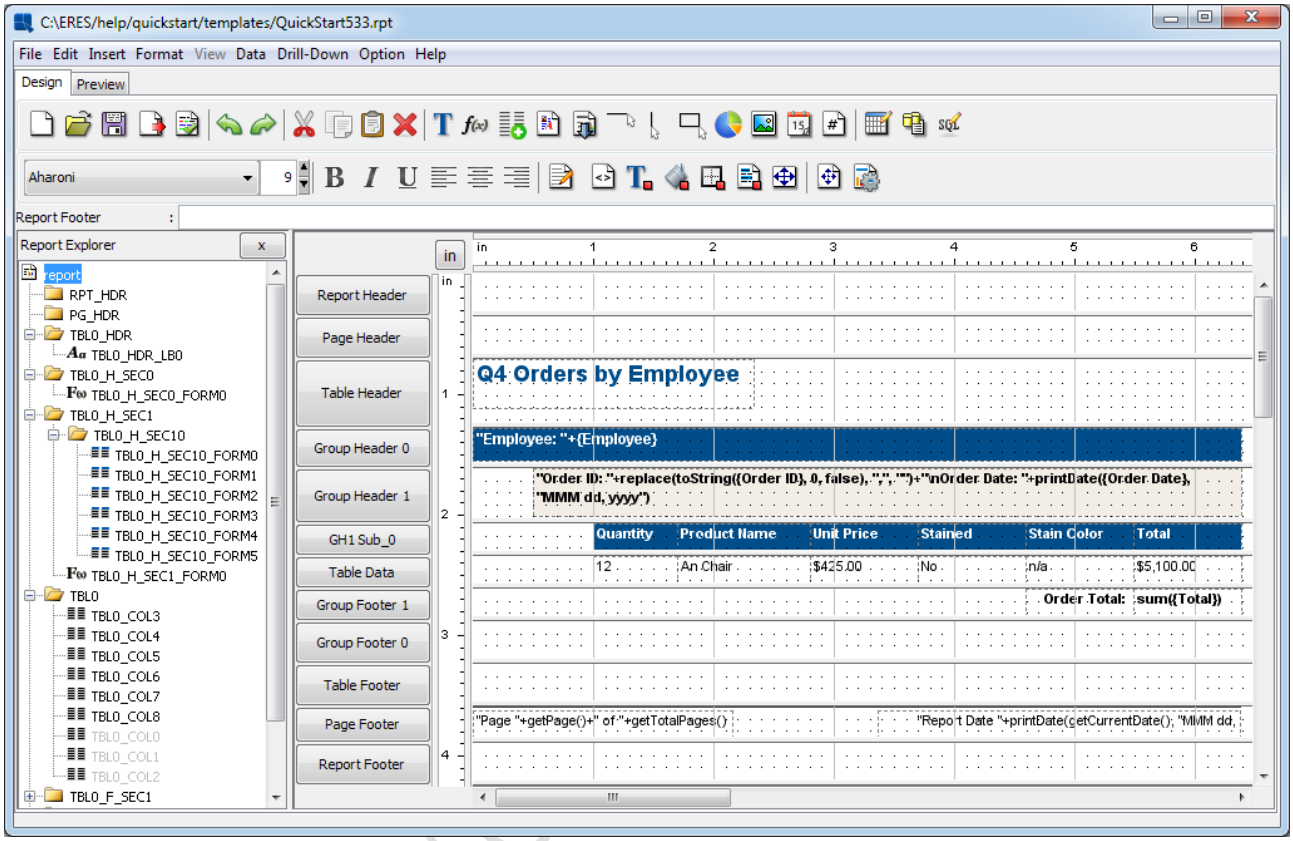
Keep Data Source Order オプションを選択した場合は、マスタ＆詳細およびサマリブレークレポートの **Tope N Report** オプションは無効になります。

Copyright(C) 2018

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.5 デザイナーインタフェース

他のレポート作成者に精通しているユーザは、Report Designer の帯状のインタフェースをつまずくことなく理解できます。レポート要素は、Designer ウィンドウ内で追加および変更することができ、結果レポートはプレビューウィンドウでプレビューできます。



Report Designer インタフェース

3.5.1 レポートセクション

デザイナーインタフェース内の各グループは、異なるレポートセクションを表します。 セクション名は、各レポートグループの左にあるボタンに表示されます。

Report Header:
このセクションはレポートのタイトルです。レポートの上部に一度だけ表示されます。

Page Header:
レポートのページごとに上部に表示される属性です。

Table Header:
このセクションは、レポートの詳細またはデータセクションのヘッダーとして機能します。デフォルトでは、レポートに 1 回だけ表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Group Header:
このセクションは、グループ化されたデータ（マスタ＆詳細レポート）または行区切りが挿入されたデータ（サマリレポート）などのレポートに表示されます。レポート内の各グループの先頭に繰り返し表示されます。

Table Data:
これは、ほとんどのデータを含むレポートのメインセクションです。レポート用に選択されたデータ列はこのセクションに配置され、列の各エントリに繰り返されます。

Group Footer:
このセクションは、グループ化されたデータ（マスタ&詳細レポート）または行区切りが挿入されたデータ（サマリレポート）などのレポートに表示されます。レポート内の各グループの最後に繰り返し表示されます。

Table Footer:
このセクションは、レポートのフッターまたはデータセクションとして機能します。デフォルトでは、レポートに 1 回だけ表示されます。

Page Footer:
このセクションに配置された要素は、レポートの各ページの下部に表示されます。

Report Footer:
これはレポートのフッターセクションの最後の要約です。レポートの最後に一度だけ表示されます。

3.5.1.1 ネストされるセクション

レポートヘッダー、テーブルヘッダー、グループヘッダー、グループフッター、テーブルフッター、レポートフッターセクションに加えて、ネストされるセクションもあります。ネストされるセクションは、可変長のサブレポートとリッチテキストフィールドを配置する場合に便利です。これにより、これらのタイプの要素は、下にある要素と重なることなくサイズを変更することができます。ネストされるセクションは、ページ区切りオプションと繰り返しオプションを含む、親セクションからセクションオプションのほとんどを継承します。

ネストされるセクションを追加するには、親セクションのセクションオプションポップアップメニューから **Insert Section** を選択します。セクションオプションの詳細については、セクションオプションを参照してください。ネストされるセクションからセクションを挿入するように選択すると、同じ親セクションを継承します（つまり、ネストされるセクションは親にすることはできません）。

3.5.2 セクションのリサイズ

セクションの高さを変更するには、デザイナーウィンドウ内のセクションディバイダにマウスを置きます。ポインタがサイズ変更ポインタに変わります。セクションの高さを拡大または縮小するには、マウ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

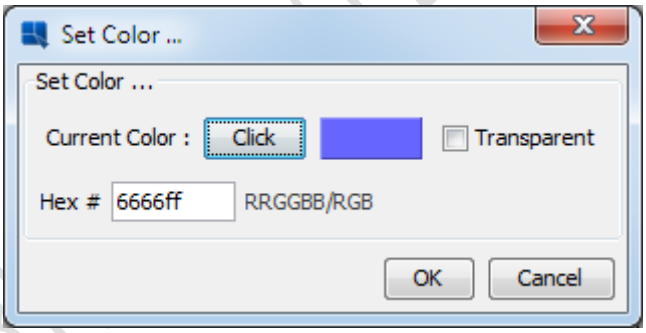
スをクリックしてドラッグします。

3.5.3 セクションオプション

各レポートセクションには、いくつかの書式設定と表示オプションがあります。セクションオプションメニューにアクセスするには、Designer の左側にあるセクション名のボタンをクリックするか、セクションフィールドの空白部分を右クリックします。

Background Color:
このオプションは、レポートのすべてのセクションで使用できます。セクション全体の背景色を設定することができます。このオプションを選択すると、背景を透明にするかどうかを指定するダイアログボックスが表示されます。背景が透明でないようにするには、チェックボックスをオフにしてボタンをクリックします。これにより、背景色を指定するダイアログが表示されます。カラーパレットからフォントの色を選択するか、HSB/RGB 値を入力します。Hex Color 値を **Hex#**フィールドに直接入力することもできます。

セクションに配置されている要素がない場合、そのセクションは表示されません。したがって、セクションの背景色を変更しても、セクションが空白の場合は完成したレポートには影響しません。



Background Color ダイアログ

Section Height:
このオプションは、レポートのすべてのセクションで使用できます。セクションの高さを設定することができます。セクションの下端をドラッグするのと同じ効果がありますが、ドラッグアンドドロップの方法とは異なり、セクションの高さをインチ単位で正確に設定できます。

Inseret Section:
このオプションは、レポートヘッダー、テーブルヘッダー、グループヘッダー、グループフッター、テーブルフッター、およびレポートフッターセクションで使用できます。このオプションを選択すると、現在選択されているレポートセクションにネストされたセクションが追加されます。

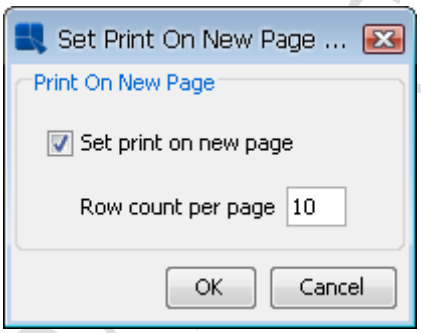
Remove Section:
このオプションは、レポートに追加したネストされたセクションでのみ使用できます（通常のセクションは不可視にしかレンダリングできません)。このオプションは、レポートからネストされたセクショ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ンを削除し、その内容を削除します。

Repeat On Every Page:
このオプションは、テーブルヘッダーおよびグループヘッダーのセクションで使用できます。このオプションを選択すると、ページ区切りごとにセクションが再び繰り返されます（グループヘッダーは、そのグループのデータが 1 ページを超える場合にのみ繰り返されます)。

Print On a New Page:
このオプションは、テーブルヘッダー、グループヘッダー、テーブルデータ、グループフッター、テーブルフッター、およびレポートフッターセクションで使用できます。**Table Data** セクションでは、各ページに表示する行数を選択できる新しいダイアログがポップアップ表示されます。この数値は、ページのテーブルデータ行の新しい最大値として設定されます。この設定値の行の後、改ページが挿入されます。この数にはフッターセクションの式は含まれません。たとえば、レポートに行区切りと集計がグループごとに計算されている場合、この要素は行数にカウントされません。



Print On New Page ダイアログ

他のセクションでこのオプションを選択すると、セクションが表示されるたびに改ページが挿入されます。

New Excel Sheet:
このオプションは、グループヘッダーセクションでのみ使用できます。このオプションを選択すると、レポートを Excel（XLS）または Excel 2007（XLSX）形式にエクスポートすると、各グループが別のワークシートに印刷されます。

Reset Page Number:
このオプションは、グループヘッダーセクションでのみ使用できます。このオプションを選択すると、セクションが実行されるたびにレポートのページ番号がリセットされます。したがって、グループヘッダーセクションが実行されるたびに、現在のページ番号は 1 になります。このオプションを設定すると、自動的に新しいページオプションで印刷が有効になります。

Fit Group On Page:
このオプションは、グループヘッダーセクションでのみ使用できます。これにより、グループ化されたデータのページネーションの一部を制御できます。このオプションを選択すると、EspressReport はグ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ループ全体をページに収めるようになります。グループ全体がページに印刷できない場合は、新しいページに印刷されます。グループが任意のページに収まらない場合（フィールドがページの高さよりも長い場合)、複数のページに分割されます。

Skip First Value:
このオプションは、テーブルヘッダー、グループヘッダー、グループフッター、およびテーブルフッターセクションで使用できます。このオプションは、新しいページオプションの印刷のサブセットです。このオプションを選択すると、セクションが最初に実行されたときの改ページは無視されます。たとえば、このオプションをグループヘッダーセクションに設定すると、セクションが最初に実行されたときにページ区切りを挿入しないようになります。ページ区切りは、レポートがグループヘッダーを 2 回実行したときにのみ発生します。

Skip First Group Value:
このオプションは、グループヘッダーセクションでのみ使用できます。このオプションは、新しいページオプションの印刷のサブセットです。このオプションを選択すると、データの各グループの最初の値の改ページは無視されます。このオプションは、複数のグループが新しいページに印刷するように設定されているレポートのネストされたグループ化に使用されます。グループ化された複数の列の値が変更されているレコードの空白ページを防止するには、**Skip First Group Value** を内部のグループヘッダーセクションに設定します。

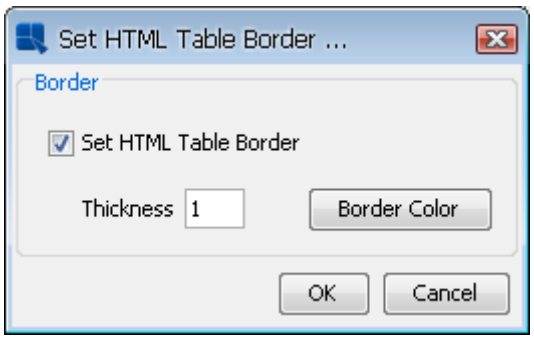
Resize Cells Proportionally:
このオプションは、セクション内のセルをコンテンツに合わせてサイズを変更した場合にのみ適用されます。このオプションを選択すると、セクション内のすべてのセルが可変長セルでサイズ変更されます。

Invisible:
このオプションは、レポートのすべてのセクションで使用できます。このオプションを選択すると、セクション全体が非表示になります。セクションをもう一度表示するには、Designer の上部にある **Format** メニューをクリックし、**Make Section Visible** オプションを選択します。

HTML Border:
このオプションはすべてのセクションで使用でき、レポートを HTML にエクスポートする場合にのみ有効です。テーブル形式の HTML（DHTML CSS の書式設定ではない）は表テーブルの選択的なセルの枠線を許可しないため、レポートが HTML 形式でレンダリングされるときに、セクション内のすべてのセルの枠線をオンまたはオフに選択できます。このオプションを選択すると、次のダイアログが表示され、HTML 枠のオプションを設定できます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



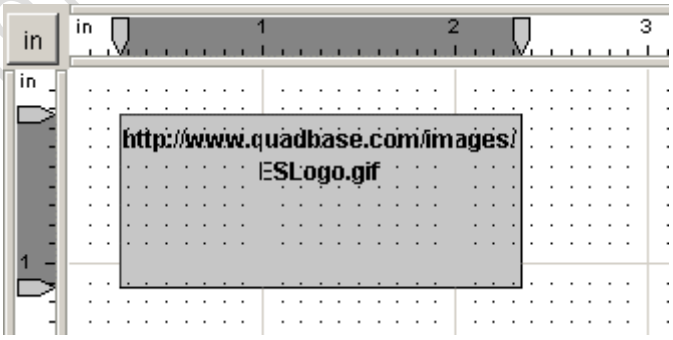
HTML Border オプション

このダイアログでは、**Border Color** ボタンをクリックして、太さと境界線の色を設定できます。これにより、カラーパレットダイアログが開き、HTML の枠線の色を選択することができます。

Scripting:
このオプションを使用すると、特定のセクション属性を動的に変更するためのスクリプトを作成、変更、または適用できるスクリプトインタフェースが表示されます。この機能の詳細については、セクションスクリプトを参照してください。

3.5.4 ルーラー

Report Designer には、デザインウィンドウの左上隅に 2 つのルーラーがあります。ルーラーは、レポートが物理ページの次元でどのように見えるかを測定します。寸法にページ余白が含まれていないことに注意してください。ルーラーは、印刷されたレポートがどのように表示されるかを知るのに役立ちます。また、ルーラーを使用して、レポート内の要素の境界を調整することもできます。選択された要素境界は、ルーラー内の陰影領域として表示され、マーカーをクリックしてドラッグすることで調整できます。



ルーラーおよびインチ/センチボタン

3.5.4.1 インチとセンチメートルの切り替え

デザイナーウィンドウの左上隅にあるボタンを使用すると、インチとセンチメートルの間で切り替えることができます。デフォルトでは、EspressReport は測定にインチを使用します。ボタンを押すと、ルーラーの尺度がセンチメートルに変わります。センチメートルを選択すると、要素の境界やページの余白のような手動で入力された値もすべて入力され、センチメートルで表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

デフォルト設定として、センチメートル測定値を使用するように Report Designer を設定できます。これを行うには、追加のコマンドライン引数 **metricSystem:ReportDesigner.bat/.sh** ファイルの実行時に **true/false** を指定します。たとえば、**ReportDesigner -metricSystem:true** を指定すると、メトリック測定を含む Report Designer が起動します。この引数をバッチファイルに追加することもできます。

Report Designer をアプレット経由で実行している場合、アプレット HTML ページに次のパラメータを追加する必要があります。

```
<PARAM NAME=metricSystem VALUE="true">。
```

3.5.5 デザイナーメニュー

Report Designer の機能のほとんどは、デザイナの上部にあるドロップダウンメニューを使用して制御できます。

3.5.5.1 ファイルメニュー

このメニューは、ほとんどのファイル操作を実行します。

New:
新しいレポートを作成します。

Open:
既存のレポート（**.pak/.rpt/.xml/stl**）を開きます。

Close:
現在開いているレポートを閉じます。

Apply Template:
テンプレートファイル（**.pak/.rpt/.xml**）を現在のレポートにインポートおよび適用します。すべてのレポートの属性がテンプレートの属性に変更されます。

Save:
現在のレポートを保存します。

Save As:
現在のレポートをテンプレート（**.pak**）ファイルに別名で保存できます。**Save As** ウィンドウ内のチェックボックスを使用すると、レポートビューアが埋め込まれたアプレットページの生成、XML テンプレートの生成を行うことができます。

Export:
レポートを選択したフォーマットでエクスポートします。エクスポート可能な形式は **DHTML、PDF、**

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

CSV（データファイル）、Excel（XLS）、Excel 2007（XLSX）、RTF、TXT、XML です。

Print:
現在のレポートを印刷します（このオプションはプレビュータブからのみ実行できます）。

Exit:
アプリケーションを終了します。

3.5.5.2 編集メニュー

このメニューでは、レポート要素を編集および切り取り/貼り付けることができます。

Undo:
デザイナで最後に実行された操作を取り消し、元の状態に戻ります。デザイナは過去 10 回の操作を記憶しています。

Redo:
undo コマンドの操作を元に戻します。たとえば、フォントの色を黒から赤に変更すると、このコマンドを取り消して黒に戻し、このコマンドをやり直して再度赤に戻すことができます。

Edit:
ラベル、数式、イメージ、およびグラフで使用できます。ラベルの場合、ラベルテキストの変更を促すメッセージが表示されます。数式の場合は、数式の変更を促すメッセージが表示されます。イメージの場合は、イメージファイルの URL を変更するように求められます。チャートの場合、チャートデザイナが表示され、チャートが編集されます。

Copy:
選択した要素のコピーを作成し、クリップボードに配置します。

Cut:
選択した要素を削除し、クリップボードに配置します。

Paste:
現在のクリップボード要素をレポートにペーストします。

Delete:
これにより、テーブルデータセクションの要素を除いて、選択した要素が削除されます。削除はされませんが、見ることができなくなります。

Remove Sub-Report:
現在のレポートからサブレポートを削除します。

3.5.5.3 挿入メニュー

このメニューでは、現在のレポートに要素を挿入できます。

Insert Label:
現在のレポートに label 要素を挿入します。

Insert Formula:

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

現在のレポートに式要素を挿入します。

Insert Column Field:

使用可能なフィールドからレポートに列フィールドを挿入します（使用可能なフィールドは、レポートで選択されたフィールドです)。データマッピング：レポートの種類とデータマッピングを参照してください。

Insert Database Field:

テンプレートのデータソースがデータベースの場合にのみ使用できます。元のクエリの一部ではないデータベースからフィールドを挿入することができます。

Insert Column Header:

いずれかのレポートフィールドの列見出しを挿入します。

Insert Parameter Value:

レポートパラメータに指定された値を返す数式を挿入します。

Insert Chart:

現在のレポートに新しいチャートを挿入します。

Insert Image:

ファイルからの画像をレポートに挿入します。画像は、JPEG、GIF、または PNG 形式にすることができます。

Insert Rich Text Field:

現在のレポートに新しいリッチテキストを挿入します。

Insert Table of Contents:

レポートに目次を挿入します。

Insert Line:

レポートに垂直線または水平線を挿入します。

Insert Rectangle:

グリッドの四角形を挿入します。

Insert Guideline:

レポートにガイドラインを挿入します。ガイドラインは、レポート内の要素の移動や配置に役立ちます。ガイドラインの詳細については、ガイドラインを参照してください。

Today's Date:

現在の日付をレポートに挿入するには、Date、Date＆Time、または Time の 3 つの方法があります。日付関数は実際には数式であり、挿入後に書式を調整することができます。式の書式設定の詳細については、数式と列フィールドのデータ書式設定を参照してください。

Page Number:

次のいずれかの方法でページ番号をレポートに挿入します:ページ番号または合計ページ数からのページ番号（例：Y のページ X)。ページ番号関数は実際には数式であり、挿入後に書式設定することができます。式の書式設定の詳細については、数式と列フィールドのデータ書式設定を参照してください。

Insert Sub-Report:

新しいレポートまたは既存のサブレポートを現在のレポートに挿入します。サブレポートの詳細については、サブレポートを参照してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.5.5.4 フォーマットメニュー

書式メニューには、レポート内で現在選択されている要素の書式オプションが含まれています。一部のメニューオプションは、現在選択されている要素のタイプによっては機能しません。

Data Format:

数式および列フィールドで使用できます。選択した要素のデータ型のデータ形式エディタ（文字列、数値、日付/時刻、またはブール値）が表示されます。

Chart Export Format:

チャートでのみ使用できます。レポートを DHTML 形式にエクスポートすると、チャートは静止画像に変換されます。このオプションを使用すると、エクスポート時にグラフのイメージプロパティを指定できます。

Edit Attributes:

すべてのレポート要素で使用できます。選択した要素（境界線、背景色、配置、フォントスタイルとサイズ、回転、境界線/円柱、データ形式など）に応じて属性を調整できるダイアログボックスが表示されます。

Font Style and Size:

ラベル、数式、列フィールドで使用できます。選択した要素のフォントの外観とサイズを変更するダイアログボックスが表示されます。

Background Color:

ラベル、数式、列フィールドで使用できます。選択した要素の背景色を調整することができます。

Dual Colors:

レポートの Table Data セクションにある要素に対してのみ使用できます。選択した列の背景色、フォントの色、フォントスタイル、および変更を行う行数や変更の基になる列フィールドを交互に指定できます。

Line Color & Thickness:

ラインでのみ使用できます。選択した線の色と太さを設定することができます。

Border:

すべてのレポート要素で使用できます。これは、選択した要素の周りに描かれたボーダーの角（シャープまたはラウンド）の太さ（ピクセル単位)、および色と形状を指定することを可能にします。ボーダーの厚さ/コーナーは、要素の各サイド/コーナーに対して個別に設定できます。

HyperLink:

列フィールド以外のすべての要素で使用できます。要素を Web ページや別のレポートにハイパーリンクすることができます。

Scripting:

すべてのラベル、数式、列フィールドで使用できます。スクリプトを作成し、選択した要素に条件付き書式を適用することができます。

Rotation:

すべてのラベル、数式、列フィールドで使用できます。要素を時計回りまたは反時計回りに 90 度回転させることができます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Z-Index:

選択したレポート要素の Z-index を設定できます。Z-index は、互いの上に置かれたときのレポート要素の振る舞い（最初に表示されるもの）を制御します。

Bounds:

すべてのレポート要素で使用できます。この機能を使用すると、要素の境界線を手動で入力することができます。ルーラーのデザイナーウィンドウの左上隅にあるトグルボタンで選択された単位に応じて、単位はインチまたはセンチメートルで表示されます。

Alignment:

ラベル、数式、列フィールドで使用できます。選択した要素を、左、中央、右、上、中、下に垂直に水平に整列させることができます。

Wordwrap:

ラベル、数式、列フィールドで使用できます。これにより、ユーザはレポート要素内の単語の折り返しを有効または無効にすることができます。

Resize To Fit Content:

ラベル、数式、列フィールドで使用できます。選択された要素の高さが内容に合わせて動的に調整されます。

Make Column Visible:

Table Data セクションで要素を削除した場合、またはレポートウィザードの列オプション画面から非表示のデータ列を選択した場合にのみ使用できます。これにより、不可視の列を表示することができます。

Make Section Visible:

レポートセクションを非表示にした場合にのみ使用できます（セクションオプションを参照）。セクションが再び表示されます。

Invisible:

テーブルデータセクションまたはレポートセクションの要素で使用できます。選択した要素が非表示になります。

Column Wrap:

列フィールドが折り返されるので、ページの最後の次のページに分割するのではなく、現在のページ内の元の列フィールドの隣に続行されます。

Auto Fit Columns:

レポート内のすべての要素を縮小し、ページ幅内のレポートに合わせます。

Swap Columns:

レポート内で 2 つの要素の位置を入れ替えることができます。Ctrl +クリックで要素を選択する必要があります。

3.5.5.5 表示メニュー

表示メニューには、プレビューウィンドウでレポートをナビゲートし、データを並べ替えるためのオプションがあります。レポートをプレビューするときにのみ有効です。

First Page:

プレビューウィンドウ内のレポートの最初のページに移動します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Previous Page:

プレビューウィンドウ内の 1 ページをスクロールバックします。

Next Page:

プレビューウィンドウ内で 1 ページ分スクロールします。

Last Page:

プレビューウィンドウ内のレポートの最後のページに移動します。

Go To Page...:

ページ番号を入力し、プレビューウィンドウ内のレポートのそのページに移動するように求められます。

Sort by (ascend):

レポート列のいずれかを選択してレポートを昇順（数値またはアルファベット順）で並べ替えることができる 2 番目のメニューが開きます。

Sort by (descend):

レポート列の 1 つを選択してレポートを降順（数値またはアルファベット順）で並べ替えることができる 2 番目のメニューが表示されます。

Sort by...

（複数の列で）レポートのネストされたソートを選択できるダイアログが開きます。並べ替えの方向は、列ごとに設定できます。

Launch Page Viewer:

レポート全体が Page Viewer ウィンドウにロードされます。この機能の詳細については、Page Viewer の使用を参照してください。

3.5.5.6 データメニュー

データメニューには、現在のレポートで使用されているデータとそのプロパティを表示および操作できるオプションが含まれています。

Refresh:

現在のレポートで使用されているデータソースを再クエリし、レポートを更新します。

Change Data Mapping:

レポートウィザードが再度開き、現在のレポートのデータマッピングを変更できます。

Change Data Source:

Data Source Manager が再起動され、レポートのデータソースを変更または変更できます。

Modify Database:

レポートでデータベースをデータソースとして使用する場合、このオプションを使用すると、レポートがデータベースへの接続に使用する接続情報を変更できます。この機能の詳細については、データベース接続の編集を参照してください。

Modify Query:

レポートでデータベースをデータソースとして使用する場合は、このオプションを使用して、レポートデータの取得に使用するクエリを変更できます。この機能の詳細については、クエリの編集を参照してください。

View Column Mapping:

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

現在のレポートの列マッピングを表示するウィンドウが表示されます。このウィンドウには、列インデックス、名前、データ型、オプション、および現在表示されているかどうか、または表示されているかどうかが表示されます。

View Table:

現在のレポートが生成されるデータテーブルを含むウィンドウを表示します。最初の 20 レコードのみがテーブルに表示されます。すべてのレコードを表示するチェックボックスをクリックすると、データテーブルのすべてのレコードが表示されます。

View Data Source Info:

テンプレートの作成に使用されたデータソースに関する情報を含むウィンドウを表示します。データソースの種類、場所、およびデータレジストリの場所が表示されます。

Sub-Report Parameter Sharing:

サブレポート内で定義されたパラメータを、メインレポートのパラメータまたは他のサブレポートで定義されたパラメータにリンクすることができます。この機能を有効にすると、各サブレポートに値を渡すことなく、ユーザ指定のパラメータ値を自動的に共有することができます。サブレポートの詳細については、サブレポートを参照してください。

Sub-Report Parameter Mapping:

パラメータ化されたサブレポートをプライマリレポートの列フィールドにリンクすることができます。サブレポートの詳細については、サブレポートを参照してください。

Chart Parameter Mapping:

パラメータ化されたチャート（独立したデータソースを持つ）をレポートの列フィールドにリンクすることができます。この機能の詳細については、チャートパラメータリンクを参照してください。

Select Multiple Drill-Down Values:

複数値のドリルダウンレポートをプレビューする場合にのみ使用できます。次のレベルに掘削するときに使用する値を選択できます。ドリルダウンレポートの詳細については、ドリルダウンを参照してください。

Security:

現在選択されているセルのセキュリティ設定ダイアログが表示されます。テンプレートセキュリティの詳細については、テンプレートセキュリティを参照してください。

Cell Properties:

レポートのすべての要素で使用できます。選択した要素の ID 値を含むプロパティと、列フィールドの列インデックス、名前、およびデータ型を表示するウィンドウが表示されます。

Set Preview Security Level:

テンプレートをプレビューするときに使用する、定義済みのセキュリティレベルを指定できます。テンプレートセキュリティの詳細については、テンプレートセキュリティを参照してください。

Set Preview Data Options:

プレビューウィンドウのライブデータのオン/オフ、レポートのプレビュー時に返されるレコードの最大数を設定できます。

Secured Query Parameters:

パラメータセキュリティダイアログが表示され、パラメータ値をセキュリティレベルに割り当

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

てることができます。テンプレートセキュリティの詳細については、テンプレートセキュリティを参照してください。

Preview Parameter Prompt:

レポートをプレビューするときにパラメータのプロンプトを有効/無効にすることができます。

3.5.5.7 ドリルダウンメニュー

ドリルダウンメニューには、レポート内のドリルダウンレイヤの追加、削除、および移動を可能にするオプションが含まれています。ドリルダウンの詳細については、ドリルダウンを参照してください。

Navigate:

ドリルダウンのレイヤーを編集、追加、および削除できるドリルダウンナビゲーションウィンドウが表示されます。

Drill-Down Link:

現在選択されている要素をドリルダウンレイヤにリンクできます。

3.5.5.8 オプションメニュー

このメニューには、表示オプションと印刷オプションがあります。

Report Explorer:

レポートエクスプローラの表示をオンまたはオフにします。エクスプローラインタフェースの詳細については、Report Explorer を参照してください。

Page Setup:

ページのサイズとマージンをカスタマイズできます。測定値は、ルーラーが会うデザイナーウィンドウの左上隅のトグルボタンで選択された単位に応じて、インチまたはセンチメートル単位で表示されます。

HTML Page Title:

レポートをDHTML形式にエクスポートするときに生成されるページタイトルを指定できます。

Background Color:

レポートのページの背景色を設定できます。ここに色を指定すると、レポートビューアで、DHTML または PDF にエクスポートするときにページ全体が色付けされます。透明でないセクション色またはレポート要素の背景色は、背景色に描画されます。

Background Image:

レポートの背景画像を指定できます。

Snap to grid:

グリッドスナップ機能の設定を制御できます。この機能をオン/オフにするだけでなく、ステップサイズを制御することもできます。

Font Mapping:

PDF エクスポート用のシステム (TrueType) フォントファイルをマップすることができます。この機能の詳細については、PDF フォントマッピングを参照してください。

Viewer Font Setup:

Report Designer とレポートビューアのテキストのフォントサイズを画面の解像度に比例させ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ることができます。デフォルトでオンになっているため、レポートをさまざまなエクスポート形式およびインストール間でより正確に変換できます。

Export Style Sheet:

スタイルシート（.css）ファイルをレポートの要素のスタイル定義とともにエクスポートできます。このスタイルシートファイルを使用して、DHTML エクスポート設定を他のレポートに適用できます。CSS の操作の詳細については、CSS オプションを参照してください。

Show Cell Outline:

レポート内のすべての要素のアウトラインを描画します。これにより、すべての要素の境界線を表示したり、テキストが表示されない要素を表示したりすることができます。

Shift On/Off:

セルシフトモードをオンまたはオフにすることができます。シフトモードがオンの場合、レポート要素のサイズ変更と移動は、サイズ変更された要素または再配置された要素に対応する周囲の要素をシフトさせます。

Show Formula Name:

デザインウィンドウの関数のテキストではなく、ファンクション名の表示を有効/無効にすることができます。数式の操作の詳細については、数式および式ビルダの使用を参照してください。

Enable Table of Contents:

目次を表示/非表示にすることができます（挿入されている場合）。

Global Format:

さまざまなタイプのレポート要素のそれぞれにフォーマットを設定できます。レポートに現在含まれている要素に変更が加えられ、既定のプロパティが変更されます。

Null Data Handler:

レポート内で null データの処理を指定できます。

3.5.5.9 ヘルプメニュー

このメニューでは、プログラムのバージョンを表示し、開発元ユーザーズガイドを開くことができます。

About:

プログラムのバージョンに関する情報を表示します。

Contents:

EspressReport ユーザーズガイドを開きます。

3.5.6 デザイナーツールバー

デザイナの上部にあるツールバーを使用すると、EspressReport の最も一般的に使用される機能に簡単にアクセスできます。ボタンの最初の行は、次の機能を実行します。

新しいレポートを作成

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

既存のレポートを開く

現在のレポートを保存

現在のレポートをエクスポート

現在のレポートにテンプレートを適用

最後の変更を元に戻す

最後に取り消された変更をやり直す

選択した要素を切り取り、クリップボードに配置

選択した要素をコピー

現在のクリップボードオブジェクトをレポートに貼り付ける

選択した要素を削除

ラベルを挿入

数式を挿入

列フィールドを挿入

リッチテキストフィールドを挿入

サブレポートを挿入

水平線を挿入

縦線を挿入

グリッドの長方形を挿入

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

グラフを挿入

画像を挿入

現在の日付を挿入

ページ番号を挿入

データマッピングの変更

データソースの変更

クエリの編集

ツールバーの 2 行目の最初の 2 つのボックスでは、現在選択されている要素のフォントとサイズを選択できます。2 行目のボタンは、次の機能を実行します。

フォントスタイルを太字に設定

フォントスタイルをイタリック体に設定

テキストに下線を引く

水平方向の配置を左に設定

水平方向の配置を中央に設定

水平方向の配置を右に設定

選択した要素を編集する

現在の要素のデータ形式を設定

現在の要素にスクリプトを適用

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

現在の要素のフォントスタイル、サイズ、色を設定

現在の要素の背景色を設定

現在の要素のボーダーの太さ、色、コーナーの丸めを設定

現在の要素に 2 色を設定

現在の要素の境界線を設定

ページ設定

プレビューの表示行を制限

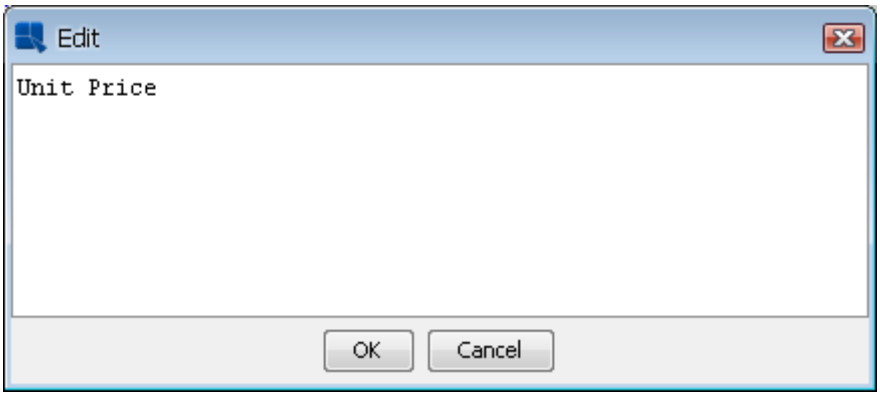
3.5.7 レポート要素の挿入と操作

レポートウィザードを終了すると、マッピングに基づいてレポートの大まかなバージョンが生成され、選択したオプションが表示されます。レポートを整理するには、新しいレポート要素を挿入するだけでなく、既存のものをフォーマットすることもできます。

3.5.7.1 要素を挿入する

Labels:

ラベルを挿入するには、**Insert** メニューから **Insert Label** を選択するか、ツールバーの **Insert Label** ボタン T をクリックします。**Insert Label** オプションを選択すると、デザインウィンドウの周りに小さなボックスがマウスポインタで表示されます。ラベルを挿入する場所にボックスを置き、をクリックします。ラベルテキストの入力を求めるダイアログボックスが表示されます。**OK** ボタンをクリックすると、レポートにラベルが表示されます。



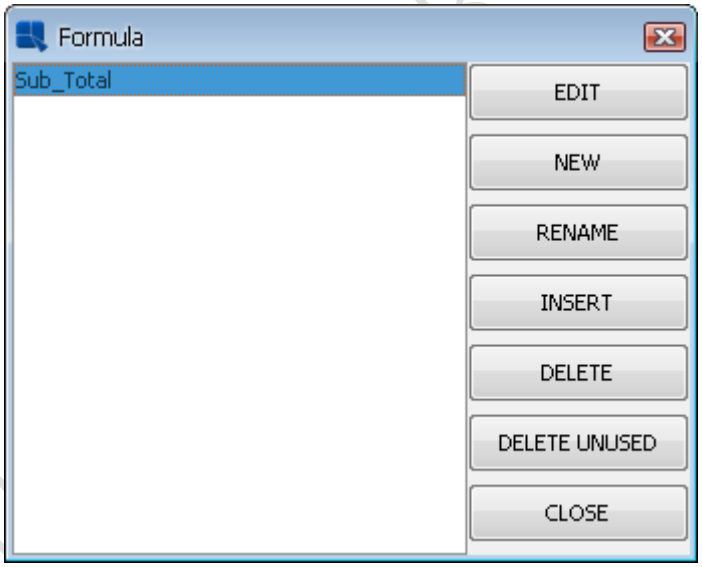
Insert Label ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Formulas:

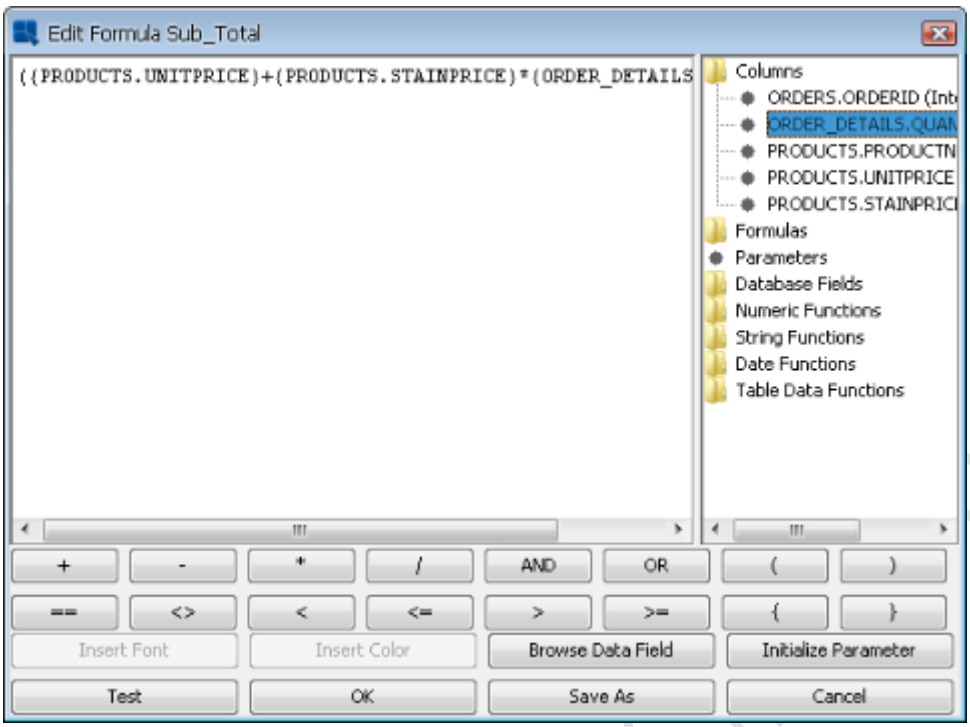
数式は、**Insert** メニューから **Insert Formula** を選択するか、ツールバーの **Insert Formula** ボタン $f(x)$ をクリックして挿入できます。**Insert Formula** オプションを選択すると、挿入する数式を選択できるダイアログボックスが開きます。数式を挿入するには、リストから数式を選択し、**Insert** ボタンをクリックします。デザインウィンドウの周りに小さなボックスがマウスポインタの後に表示されます。数式を挿入する場所にボックスを置き、それをクリックします。数式が追加されます。既存の数式を使用することも、新しい数式を作成することもできます。新しい数式を作成するには、**formula list** ダイアログの **New** ボタンをクリックします。新しい数式の名前を入力すると、式ビルダのウィンドウが開き、数式を作成できます（式ビルダと数式構文の使用の詳細については、数式および式ビルダの使用を参照してください)。任意の数式が挿入されたレポートセクションに自動的にアンカーされます。これは、セクションが繰り返されるたびに式がリセットされることを意味します。

式が Table Data セクションに挿入されていない場合、式のテキストのみがデザインウィンドウに表示されます。Option メニューから **Show Formula Name** を選択すると、テキストの代わりに数式名を表示することができます。



数式リスト

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

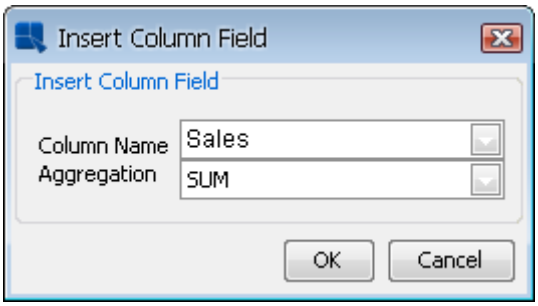


数式エディタ

Column Fields:

列フィールドは、**Insert** メニューから **Insert Column Field** を選択するか、ツールバーの **Insert Column Field** ボタンをクリックすることで挿入できます。**Insert Column Field** オプションを選択すると、2 つのドロップダウンメニューがあるダイアログボックスが表示されます。最初のレポートでは、レポートに挿入する列フィールドを選択できます。2 番目の列では、列フィールドの集約（存在する場合）を選択できます。目的の列フィールドを選択した後、**OK** ボタンをクリックすると、デザインウィンドウの周りのマウスポインタの後ろに小さいボックスが表示されます。列フィールドを配置する場所にボックスを置き、それをクリックします。列フィールドがレポートに表示されます。列フィールドは、挿入されたレポートセクションに自動的に固定されます。つまり、セクションが繰り返されるたびに、列フィールドにデータ列の次の行が表示されます。

データテーブルセクションに配置されていない場合、列値はデザインウィンドウに表示されません。表示されるのは列名**{name}**だけです。プレビューウィンドウで列のフィールド値を確認することができます。

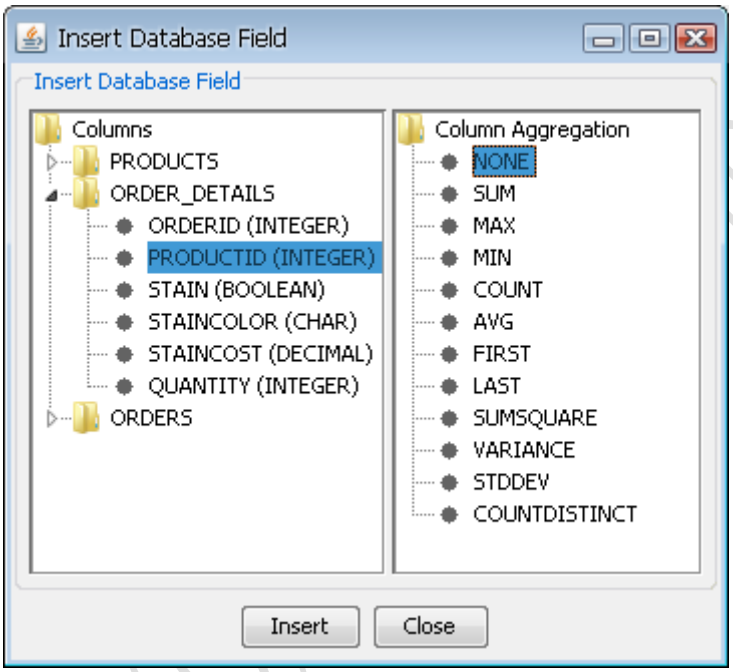


列フィールドの挿入

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Database Field:

このオプションは、現在のレポートでデータベースをデータソースとして使用する場合にのみ使用できます。これにより、クエリまたはデータマッピングの一部として最初に選択されなかったデータベースから報告書にフィールドを追加することができます。データベースフィールドを挿入するには、**Insert** メニューから **Insert Database Field** を選択します。これにより、クエリに含まれているすべてのテーブルとそれぞれのフィールドのダイアログが開き、追加するフィールドを選択することができます。別のオプションでは、フィールドで実行する集約（存在する場合）を選択できます。

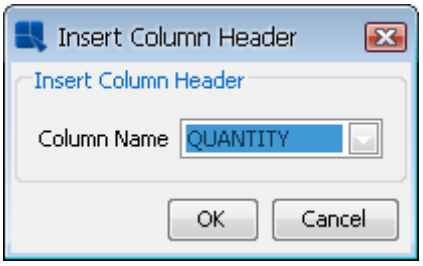


Insert Database Field ダイアログ

テンプレートのクエリに集計または **Group by** 句が含まれている場合、フィールドに **Database Aggregation** を選択するように指示する 3 番目のオプションがダイアログに表示されます。これは、グループ単位でも、データベースでサポートされている集計でもかまいません。

Column Header:

Column Header は、データソースの列ヘッダーを動的に表示する一意のレポート要素です。**Insert** メニューから **Insert Column Header** を選択して、列ヘッダーを追加できます。これにより、ヘッダーを取得する列を選択するダイアログが表示されます。



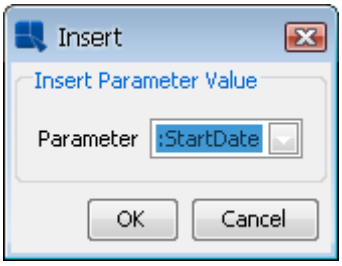
Insert Column Header ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

目的の列を選択し、**OK** ボタンをクリックします。ボックスを使用すると、レポートに必要な場所にヘッダーを配置することができます。クリックして列ヘッダーを追加します。

Parameter Values:

クエリまたは数式パラメータにユーザが入力した値を直接レポートに挿入できます。パラメータ値を挿入するには、**Insert** メニューから **Insert Parameter Value** を選択します。これにより、値を表示するパラメータを選択するダイアログが表示されます。



Insert Parameter Value ダイアログ

パラメータ値は、式を介してアクセスすることもできます。数式の詳細については、数式および式ビルダの使用を参照してください。

Images:

Insert メニューから **Insert Image** を選択するか、ツールバーの **Insert Image** ボタンをクリックして、イメージを挿入できます。このオプションを選択すると、マウスのポインタの後ろにデザインウィンドウの周りに小さなボックスが表示されます。画像を配置する場所にボックスを置き、をクリックします。挿入する画像の URL を入力するための新しいウィンドウが表示されます。

イメージを挿入するには、ローカルに画像を配置するか、URL からイメージを取得する 2 つの方法があります。

ローカル上の画像を見つけるには、**Browse** ボタンをクリックします。別の場所から画像を使用する場合は、その URL を挿入します。

URL からイメージを取得するには、**Image URL** テキストフィールドに URL を入力します。その後,Refresh Preview ボタンをクリックして URL を確認します。URL の画像が **Refresh Preview** セクションに表示される場合は、URL が正しいことを示します。

プロトコルの完全な URL（例：**http://**）を挿入する必要があります。

レポートを PAK として保存すると、イメージはレポートとともに PAK ファイルに保存されます。

RPT ファイル（古い EspressReport バージョンのもの）をバックグラウンドイメージで再オープンしたい場合は、イメージ自体がレポートに保存されていないことに注意してください。パスまたは URL

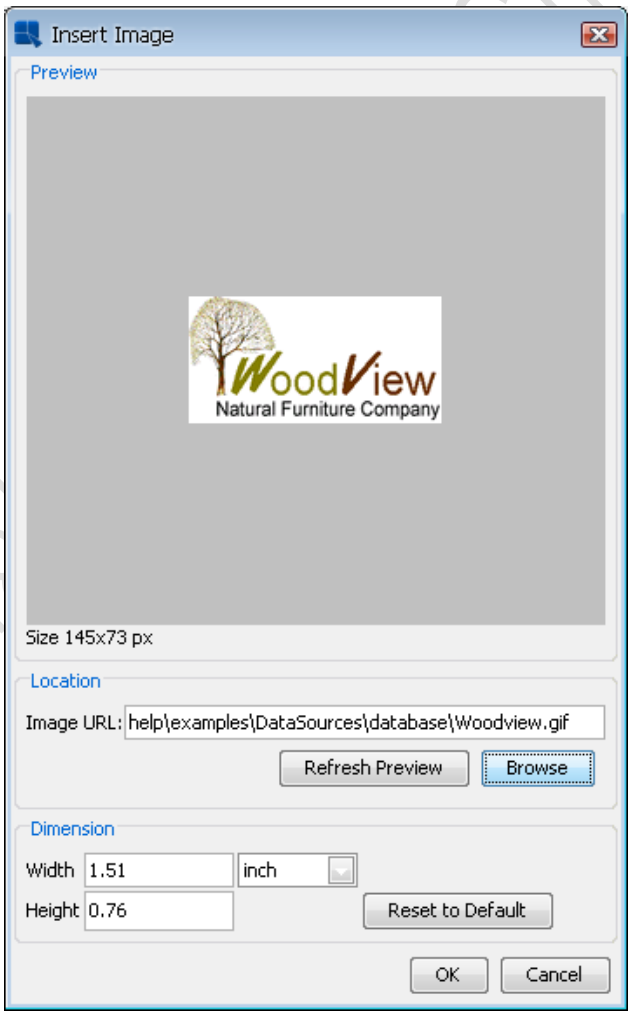
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

のみが保存されます。RPT レポートを移動する場合は、指定されたパスを使用してイメージにアクセスできることを確認する必要があります。

Dimension セクションでは、イメージの寸法をピクセル単位、インチ単位、またはミリメートル単位で設定できます。

画像のサイズがページのサイズを超えると、画像は表示されません（スペースはその画像に割り当てられます）。

EspressReport はデータベースイメージもサポートしています。BLOB フィールドをレポートのクエリの一部として選択することで、データベース（BLOB 形式）からイメージを追加できます。イメージはレポートの列として追加されます。BLOB フィールドを使用する代わりに、URL を使用してテーブルデータセクションのイメージを取得することもできます。詳細については、数式と列フィールドのデータ書式設定を参照してください。



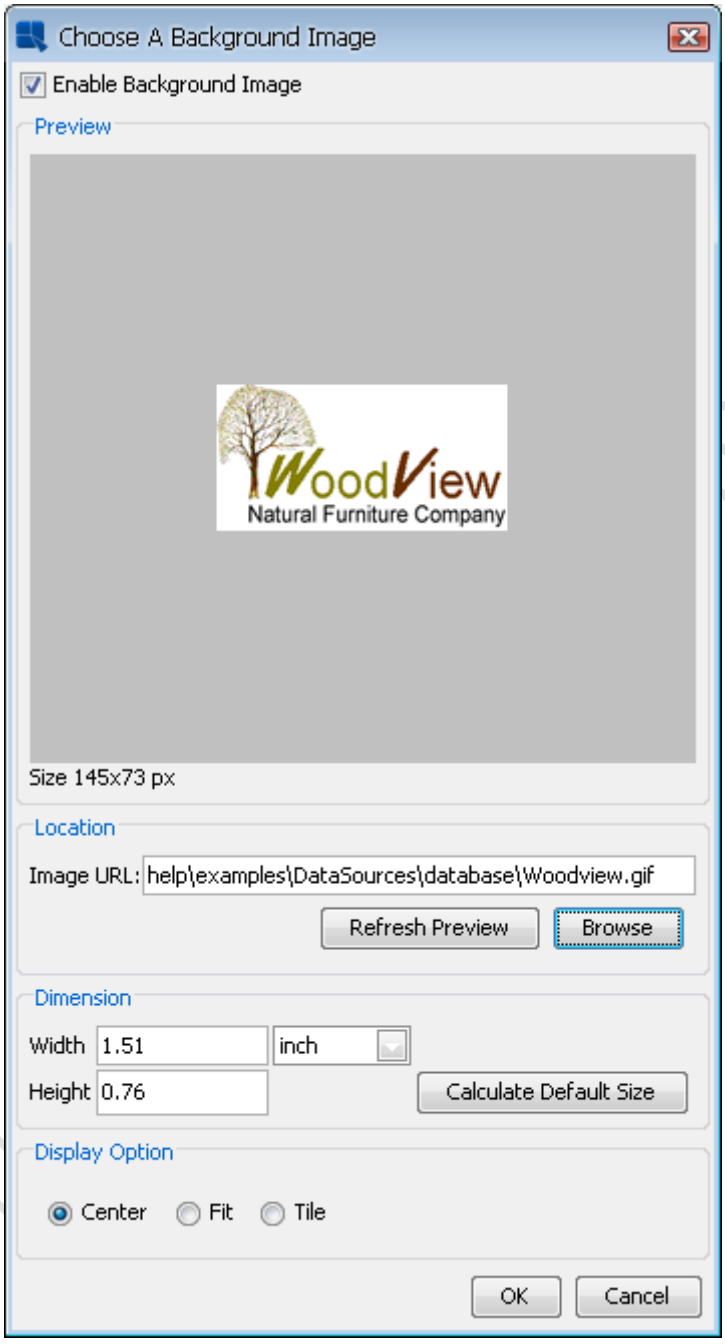
Insert Image ダイアログ

Background Images：

EspressReport では、レポートに背景画像を追加できます。背景画像は他のすべてのレポート要素の下に表示され、すべてのページで繰り返し表示されます。背景画像を追加するには、**Option** メニューか

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ら Background Image を選択します。これにより、イメージの場所といくつかの表示オプションを指定するダイアログが表示されます。



Background Image ダイアログ

新しい画像を挿入するには、**Enable Background Image** オプションを選択します。既存の背景画像を削除し、単純な背景色を使用する場合は、このオプションの選択を解除します。

このダイアログでは、**Display options**（**Center**、**Fit**、**Tile**）のいずれかを選択することもできます。

このダイアログの残りの部分は、前のセクションで説明した画像の挿入ダイアログと同じです。

Lines/Rectangles:

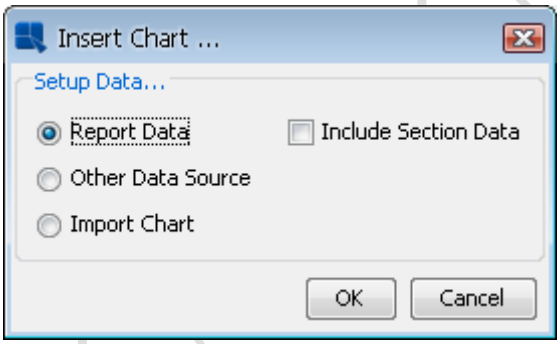
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

線またはグリッドの矩形を挿入するには、**Insert** メニューから **Insert Line** または **Insert Rectangle** を選択します。行を選択すると、縦線と横線のどちらを挿入するかをさらに選択できます。ツールバーの対応するアイコンを選択することもできます。線または四角形を選択すると、デザインウィンドウの周りにマウスポインタのあとに十字またはボックスが表示されます。線/矩形を挿入したい場所に十字ボタンを置き、クリックしてから、十字線をドラッグして線や四角形を描きます。レポートにラインまたはグリッドが挿入されます。

線またはグリッドは、実際にデザインウィンドウよりも太く表示されます。

3.5.7.1.1 チャートの挿入

チャートを二つの方法で挿入できます：**Insert** メニューから **Insert Chart** を選択するかツールバーの **Insert Chart** ボタンを選択します。このオプションを選択すると、デザインウィンドウの周りに小さなボックスがマウスポインタの後に表示されます。ボックスを配置して、グラフを配置したい場所をクリックします。次のダイアログボックスが表示されます。



Insert Chart ダイアログ

現在のレポートのデータを使用するか、外部データソースを使用するかを選択できます。レポートのデータを使用する場合は、データセットのプレビューが表示され、Chart Designer のチャート選択ウィンドウに直接移動します。それ以外の場合は、チャートに別のデータソースを選択するか、前に作成したチャートファイルをインポートするかの選択肢が与えられます。

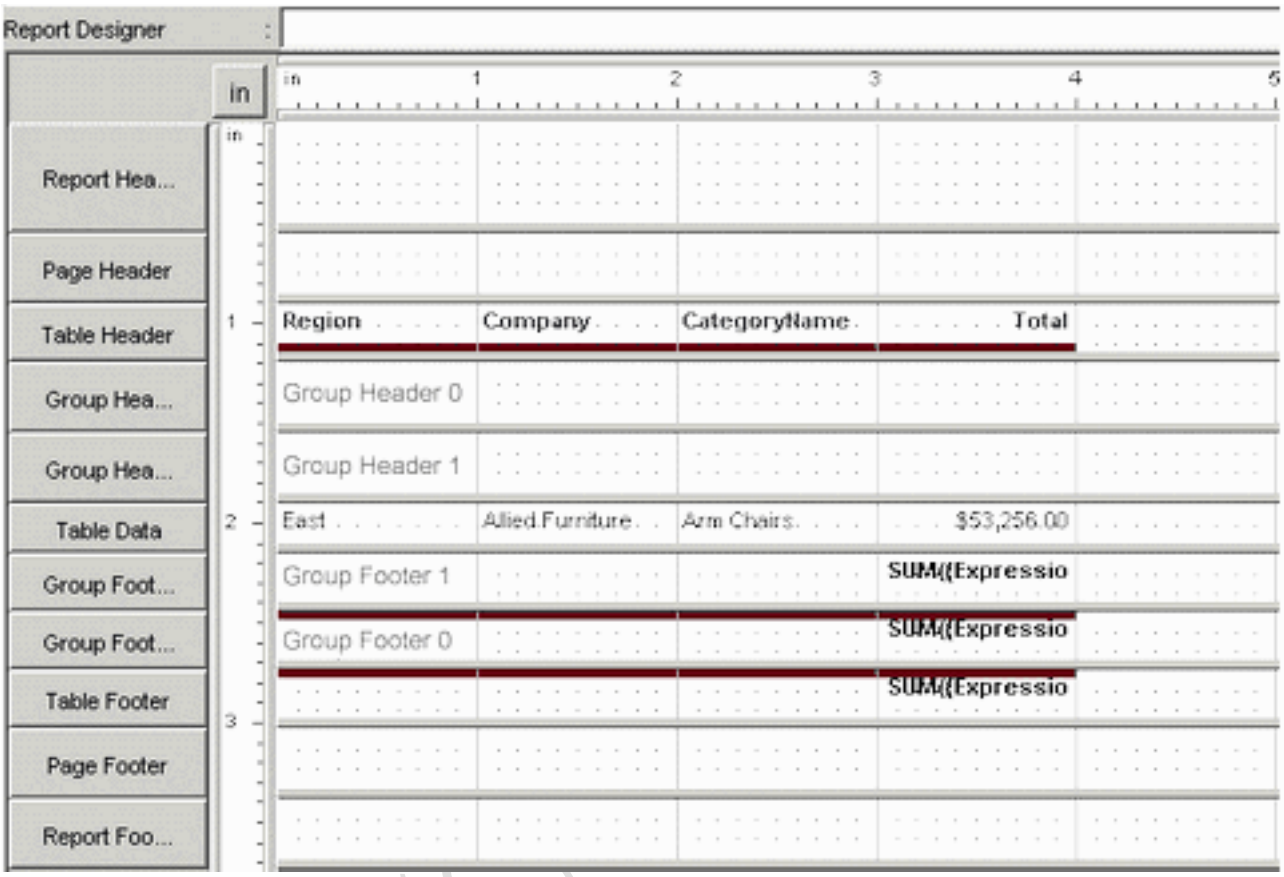
オプションを選択すると、Chart Designer が新しいウィンドウで開き、チャートをデザインしカスタマイズできます。Chart Designer の使い方の詳細についてはチャートガイドを参照してください。

実際のチャートは Design ウィンドウに表示されません。代わりに、グラフの URL に灰色の四角形が表示されます。プレビューウィンドウで実際のグラフを見ることができます。

レポートにグループ化（サマリブレーク、クロス集計、またはマスタ&詳細レポート）が含まれている場合は、**Include Section Data** オプションもあります。このオプションを選択すると、チャートに含まれる任意のセクション内にあるグループ集計を使用することができます。これは、チャートと同じセクションにある集計または内部の任意の集計のデータを使用してチャートを作成できることを意味します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

たとえば、2 つの行区切り列を持つサマリブレークレポートを作成する場合は、2 つのグループヘッダーセクションと 2 つのグループフッターセクションが作成されます。レポートの列のグループ集計があると仮定すると、レポートは次のようになります：

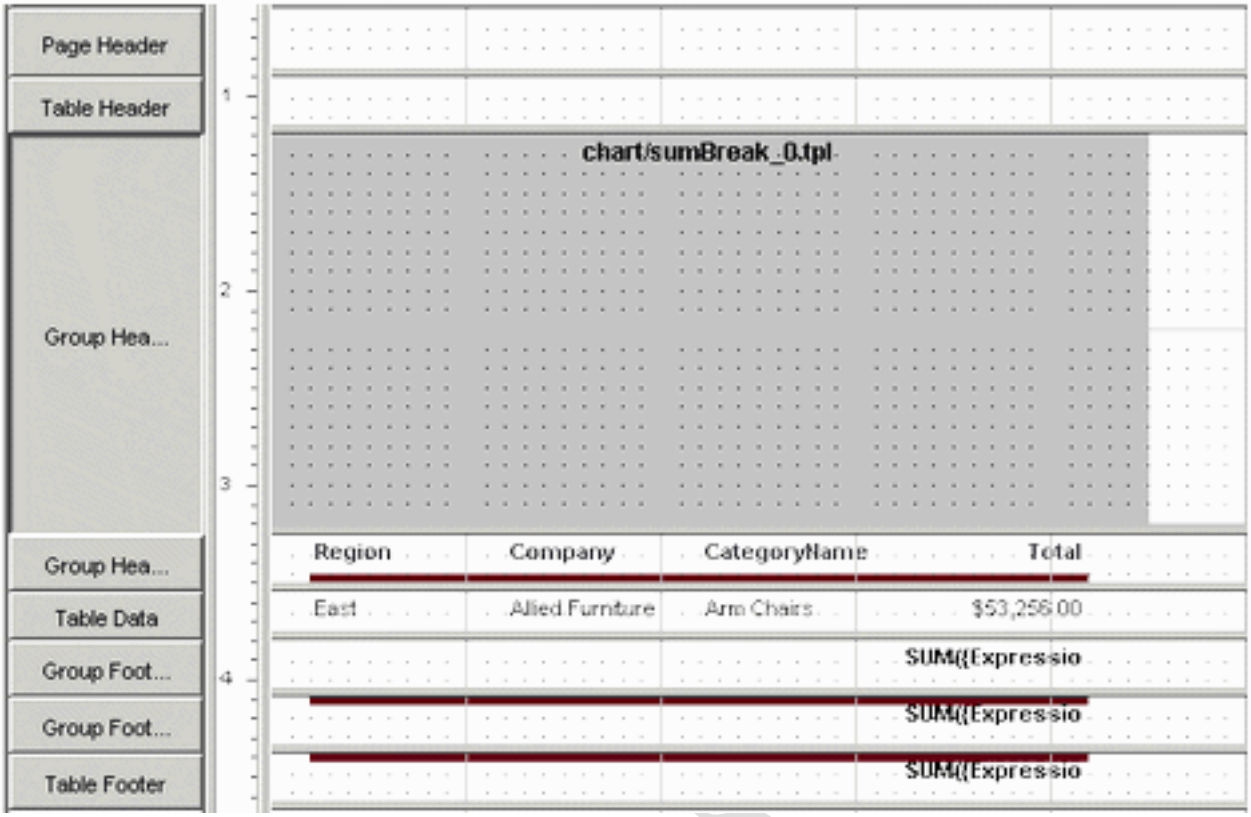


2 行の Break 列があるサマリブレークレポート

内側のグループヘッダー(Group Header 1)または内側のグループフッター(group footer 1)にチャートを配置すると、チャートの内側セクション（Group Footer 1 の集計）にのみグループ集約を使用できます。外側のグループヘッダー(Group Header 0)または外側のグループフッター(Group Footer 0)にチャートを配置すると、外側のセクションが内側のセクションを包含しているため、外側のセクションとチャートの内側のセクションの集約を使用することができます。チャートをテーブルヘッダーまたはテーブルフッターセクションに配置すると、レポート内のグループ集約を使用できます。

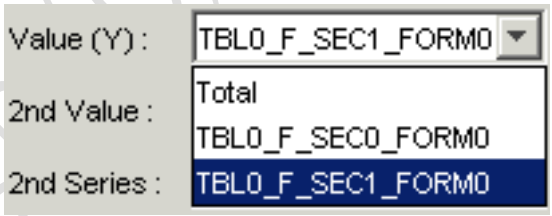
列見出しを Group Header 1 に移動し、下に示すように図を Group Header 0 に配置したとします。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



外部グループヘッダーにグラフを挿入した画面

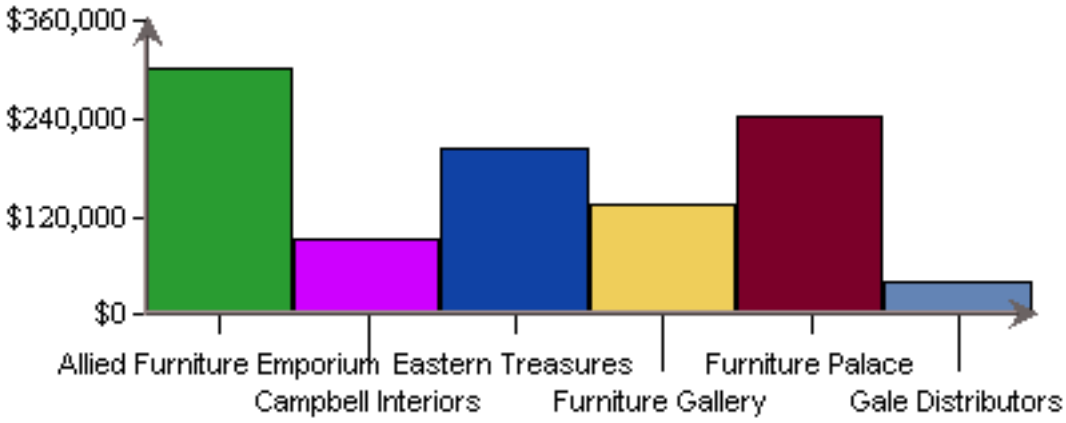
Chart Designer でデータマッピングウィンドウが表示されたら、グループフッター0（TBL0_F_SEC0_FORM0）とグループフッター1（TBL0_F_SEC1_FORM0）の両方から集計データを使用できます。



セクションデータによるチャートマッピング

集約を認識しやすくするために、カスタム ID を使用できます。カスタム ID の使用方法の詳細は、列フィールドのデータの使用を参照してください。
結果レポートは下の画像のようになります。チャートは、各会社(Company)の集計をプロットしています。グループフッター0 からの集計を使用した場合、グラフには代わりに各地域(Region)の集計が表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Region	Company	CategoryName	Total
East	Allied Furniture	Arm Chairs	$53,256.00
		Double Dressers	$58,144.00
		Oval Tables	$52,142.00
		Round Tables	$23,805.00
		Side Chairs	$23,540.00
		Single Dressers	$92,039.00
			$302,926.00

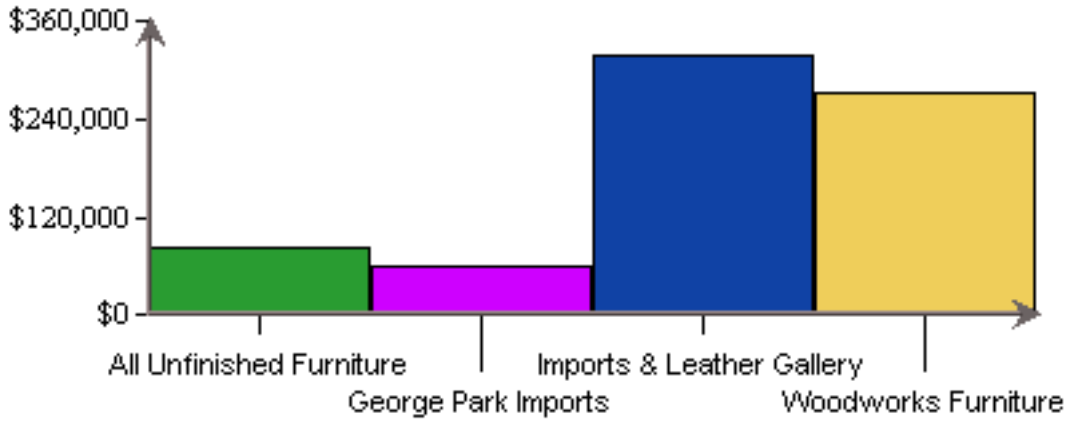
	Company	CategoryName	Total
	Campbell Interiors	Arm Chairs	$15,609.00
		Oval Tables	$11,394.00
		Rectangular Tables	$7,290.00
		Round Tables	$42,916.00
		Triple Dressers	$15,936.00
			$93,145.00

	Company	CategoryName	Total
	Eastern Treasures	Arm Chairs	$73,505.00
		Oval Tables	$3,750.00
		Rectangular Tables	$41,511.00

セクションデータを使用したチャートによるレポート(East)

チャートは Group Header 0 に配置されているため、すべての地域(region)で新しいチャートが表示されます。上の図は東部地域(East region)のチャートのみを示していますが、レポートには他の各地域に対応するチャートも含まれています。中西部地域(Midwest region)のチャートは次のとおりです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Region	Company	CategoryName	Total
Midwest	All Unfinished	Arm Chairs	$14,532.00
		Oval Tables	$27,884.00
		Rectangular Tables	$7,530.00
		Round Tables	$11,109.00
		Side Chairs	$9,903.00
		Single Dressers	$13,839.00
			$84,797.00

	Company	CategoryName	Total
	George Park	Arm Chairs	$5,424.00
		Oval Tables	$13,424.00
		Rectangular Tables	$5,416.00

セクションデータを使用したグラフによるレポート（Midwest）

Include Section Data オプションを選択すると、チャートで使用できるデータがセクションごとに異なるため、セクション間でチャートを移動することはお勧めしません。作成後にグラフを移動する必要がある場合は、グラフにマップされているデータがグラフを移動する場所からも使用できることを確認してください。

Include Section Data を使用した別の例を示します。次のクロス集計レポートがあり、チャートのデータポイントとして各列の集計を使用するとします。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Name	East	Midwest	South	West	Value
Arm Chairs	444	81	185	74	784
Double Dressers	68	12	27	0	107
Oval Tables	79	33	24	16	152
Rectangular	85	82	67	52	286
Round Tables	143	109	72	46	370
Side Chairs	150	216	377	100	843
Single Dressers	79	100	114	18	311
Triple Dressers	21	45	19	41	126
	1 069	**678**	**885**	**347**	**2 979**

グループ集計によるクロス集計レポート

後で使用するために集約を容易にするには、各グループの集計に、列見出しに一致するカスタム ID を付けます。たとえば、SUM（COL（1））に ID **SumEast** を指定し、SUM（COL（2））**SumMidwest** などを指定します。カスタム ID の詳細は、列フィールのドデータの使用を参照してください。次に、Report Data と Section Data を含む Table Footer セクションにグラフを挿入します。

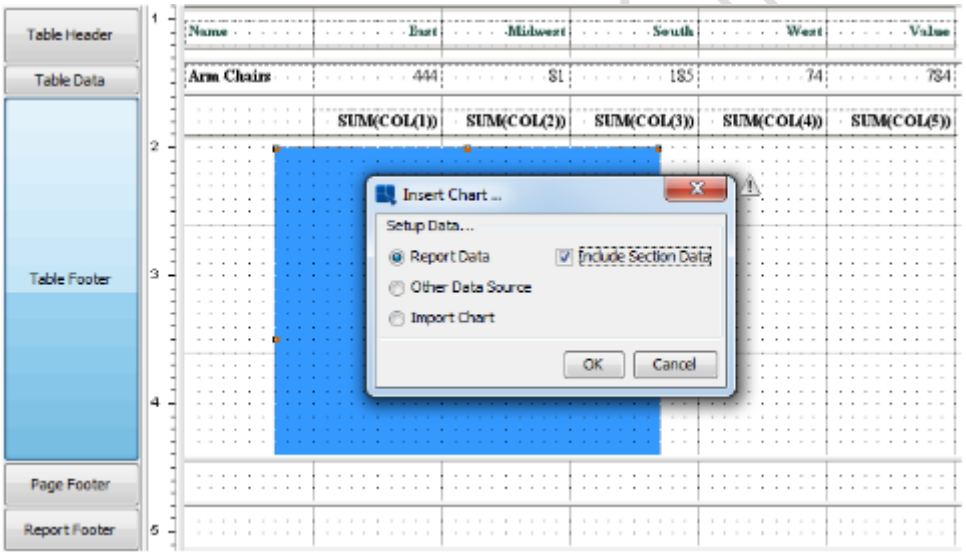
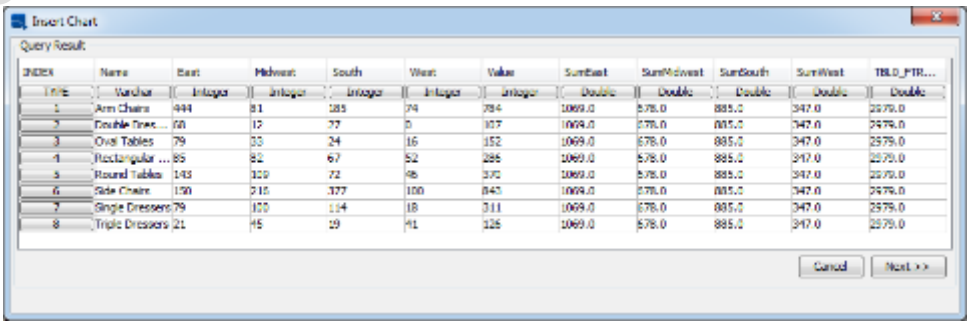


Table Footer へのグラフ挿入

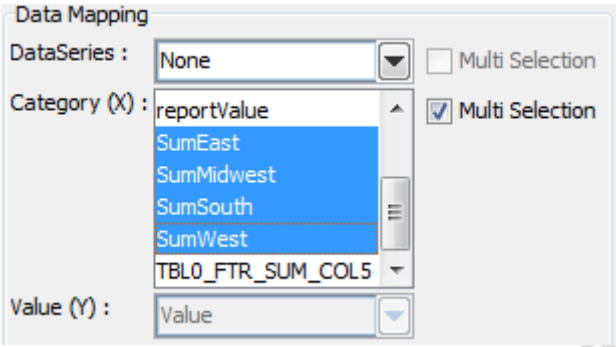
カスタム ID の数式は、データソースプレビューで表示されます（**include section data** オプションを選択した場合）。



セクションデータのデータソースプレビュー

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

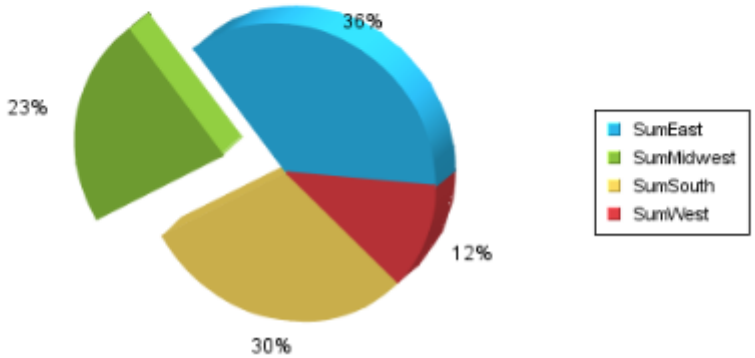
データポイントは異なる集計になっているので、転置機能を使用する必要があります。グラフのデータ転置の詳細については、データ変換の章を参照してください。
データマッピングウィンドウで、カテゴリ（X）の **Multi Selection** オプションを選択します。このオプションを使用すると、カテゴリの複数の列を選択できます。カスタム ID を持つすべての数式を選択します。値（Y）は自動的に値に設定する必要があります。



転置されたデータによるチャートマッピング

選択したグラフのタイプによって、完成したレポートは次のようになります：

Name	East	Midwest	South	West	Value
Arm Chairs	444	81	185	74	784
Double Dressers	68	12	27	0	107
Oval Tables	79	33	24	16	152
Rectangular	85	82	67	52	286
Round Tables	143	109	72	46	370
Side Chairs	150	216	377	100	843
Single Dressers	79	100	114	18	311
Triple Dressers	21	45	19	41	126
	1 069	**678**	**885**	**347**	**2 979**



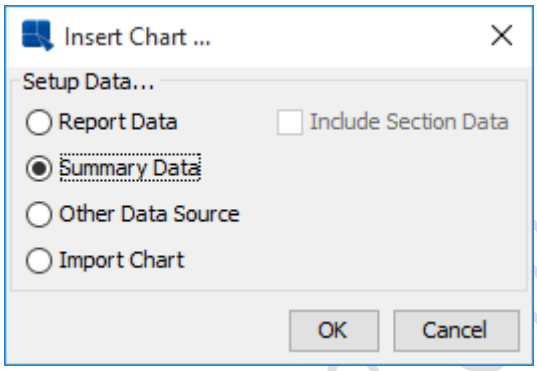
セクションデータを使用したチャートによるレポート

サマリチャートをクロス集計レポートに挿入する

サマリチャートをクロス集計レポートに挿入することもできます。サマリチャートは、固定フィールドクロス集計レポートの要約データから作成されるチャートの一種です。クロス集計レポートの詳細については、クロス集計レポートを参照してください。

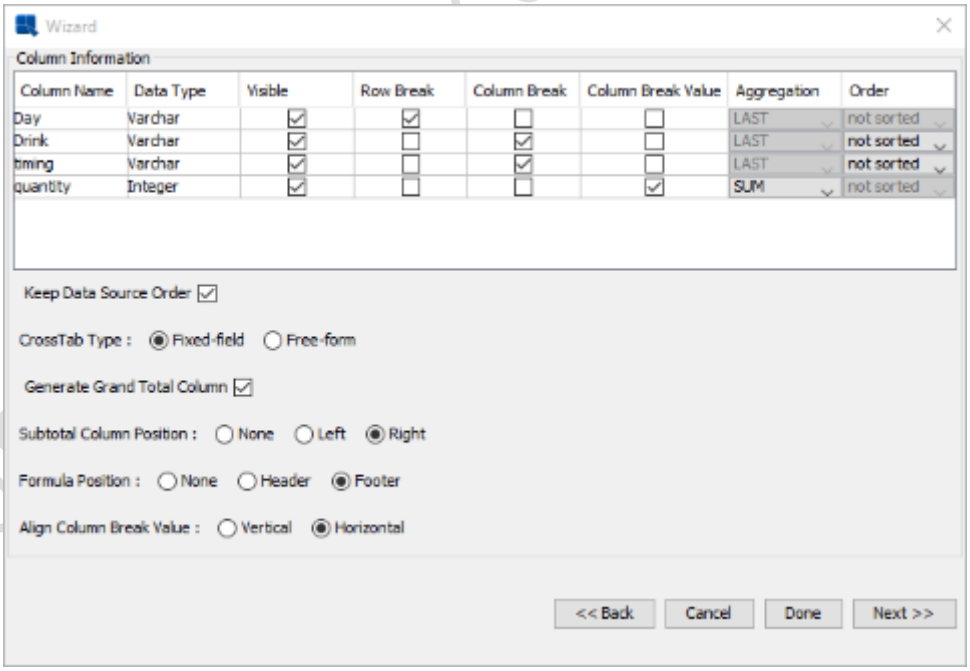
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

チャートをクロス集計レポートに挿入する場合、現在のレポートのデータを使用するか、外部データソースを使用するかを選択できます。それ以外の場合は、グラフのサマリデータソースを選択するか、前に作成したチャートファイルをインポートするか選択できます。サマリチャートは、Formula Position に応じて、Table Header/Footer セクションにのみ挿入できます。たとえば、ヘッダーセクションに Formula Position を描画するように設定された固定フィールドクロス集計レポートを作成した場合は、サマリチャートのみを Table Header セクションに挿入できます。他のレポートセクションにグラフを挿入しようとすると、**Insert Chart** ダイアログで **Summary Data** オプションを使用できなくなります。



Insert Chart ダイアログ

たとえば、クロス集計レポートに次のデータマッピングがあるとします：



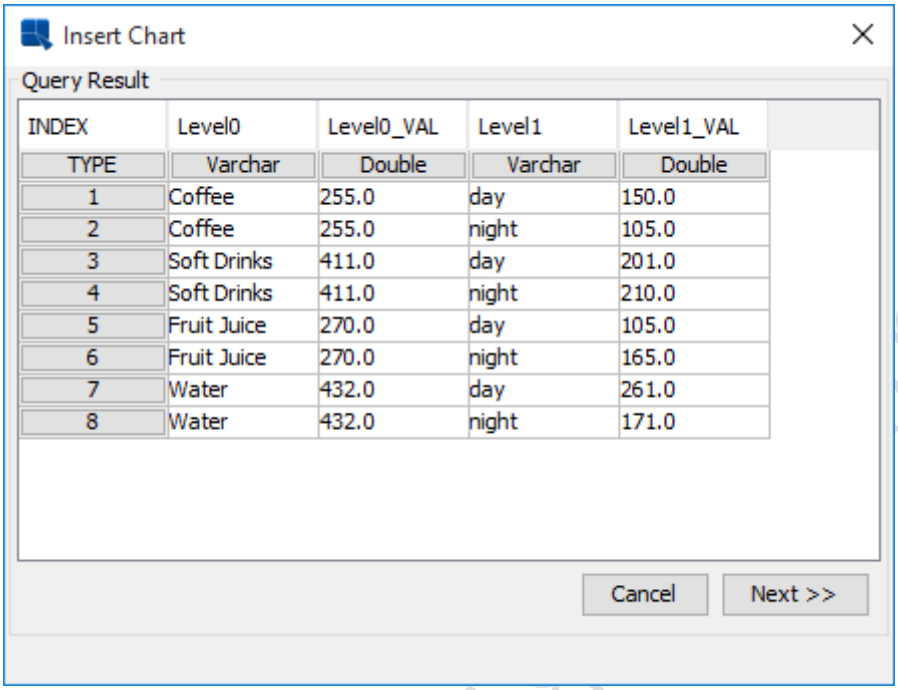
データマッピング

我々は、**Row Break** フィールドとして **Day** を選択し、**Column Break** フィールドとして **Drink** と **Timing** を、**Sum** を集計した **Column Break Value** フィールドとして **Quantity** を選択しました。

サマリチャートを表のフッターセクションに挿入し、**Insert Chart** ダイアログでデータソースとして

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Summary Data を選択すると、次の Query Result ダイアログが表示されます。このダイアログには、サマリチャートに使用されるデータが表示されます。



INDEX	Level0	Level0_VAL	Level1	Level1_VAL
TYPE	Varchar	Double	Varchar	Double
1	Coffee	255.0	day	150.0
2	Coffee	255.0	night	105.0
3	Soft Drinks	411.0	day	201.0
4	Soft Drinks	411.0	night	210.0
5	Fruit Juice	270.0	day	105.0
6	Fruit Juice	270.0	night	165.0
7	Water	432.0	day	261.0
8	Water	432.0	night	171.0

Query Result

選択したチャートのタイプと書式に応じて、完成したレポートは次のようになります：

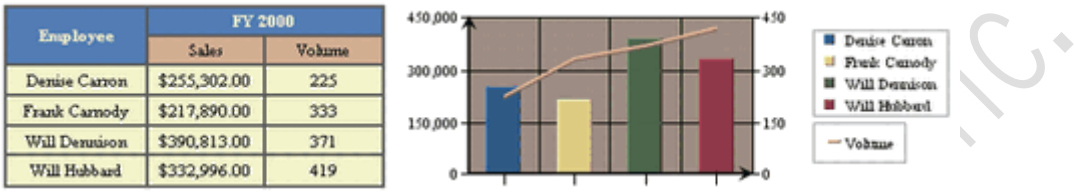
	Coffee			Soft Drinks		
Day	day	night	SUM (Coffee)	day	night	SUM (Soft Drinks)
Mon	3	30	33	24	24	48
Tue	60	12	72	27	27	54
Wed	6	24	30	30	15	45
Thu	18	3	21	24	36	60
Fri	54	33	87	60	36	96
Sat	6	0	6	27	21	48
Sun	3	3	6	9	51	60
	150	**105**	255	201	210	**411**

レポートプレビュー

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

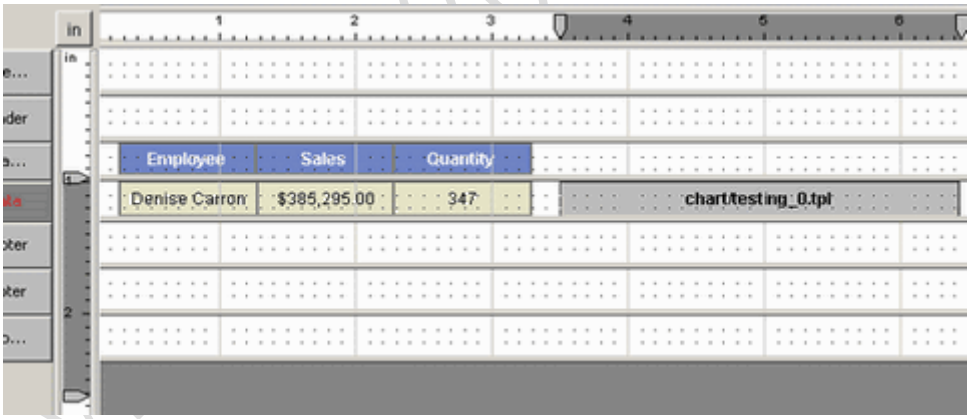
3.5.7.1.2 テーブルデータセクションへの要素の挿入

一般に、要素がレポートセクションに挿入されると、レポートセクションが繰り返されるたびに繰り返されます。ただし、Table Data セクションに特定の要素が挿入されている場合は適用されません。Table Data セクションに挿入すると、グラフ、イメージ、およびサブレポートは、データの各行（ラベル、列フィールド、および数式）に対して繰り返されません。これにより、次の図のように、データテーブルの横にグラフまたはサブレポートを配置するサイドバイサイド構成のレポートを設計できます。



サイドバイサイド構成のデータテーブルとチャート

通常、チャート、画像、サブレポートをレポートセクションに挿入すると、セクション全体の要素がセクションに収まるようにセクションのサイズを変更する必要があります（それ以外の場合は切り捨てられます）。これは、Table Data セクションに挿入されたチャート、画像、サブレポートの場合には当てはまりません。このような状況では、セクションに配置する要素のサイズに関係なく、セクションを列フィールドと同じ高さに保つ必要があります。



サイドバイサイド構成のデータテーブルとチャート(デザイナ)

ルーラーを使用して、またはセクションのサイズを一時的に変更して要素サイズを調整して、Table Data セクションで要素のサイズを変更できます。

3.5.7.2 要素の編集

要素を編集するには、次の 4 つの方法があります：**Edit** メニューから **Edit** を選択するか、ツールバーの **Edit** ボタンをクリック、もしくは要素を右クリックポップアップメニューから **Edit** を選択するか、要素をダブルクリックします。

Label の編集：

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Label の **Edit** オプションを選択すると、ラベルテキストを変更するように求めるダイアログボックスが表示されます。**OK** をクリックすると、ラベルが変更されます。

Formula の編集：

Formula の **Edit** オプションを選択すると、数式ビルダが表示され、数式を変更するよう指示されます。必要な変更を行い、**OK** ボタンをクリックすると、数式が変更されます。

同じ数式をレポートの複数の場所で使用する場合は、数式を変更するとどこでも変更されます。数式ビルダの **Save As** ボタンをクリックすると、これを回避できます。これにより、変更された数式の新しい名前を指定し、特定の要素に対してのみ変更することができます。

Chart の編集：

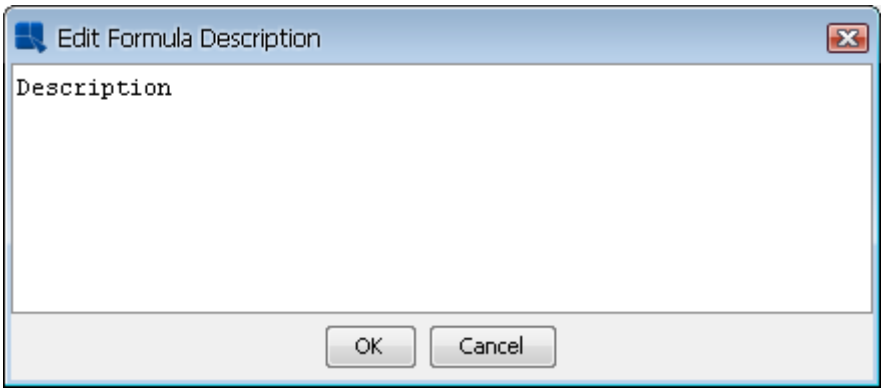
Chart の **Edit** オプションを選択すると、Chart Designer が開き、チャートを編集してフォーマットすることができます。

Image の編集：

イメージの **Edit** オプションを選択すると、新しいイメージまたはディレクトリの URL を選択するための新しいウィンドウが表示されます。新しい画像を選択した後、**OK** ボタンをクリックすると、新しい画像が古い画像に置き換わります。

Column Header の編集：

Column Header は最初にレポートを作成するときに生成される固有のタイプのレポート要素です。デフォルトでは、Column Header は動的であり、列が変更されても（データのマッピング、ソースの変更、またはテンプレートの適用によって）列の名前が表示されます。Column Header を編集するように選択すると、ヘッダーのテキストを変更するように求めるダイアログボックスが表示されます。



Edit Column Header ダイアログ

ヘッダーを編集すると、新しいテキストが表示されます。ただし、ヘッダーは動的ではなくなります。ヘッダーを戻して、（動的）列名を右クリックし、ポップアップメニューから元の列ヘッダーを選択して表示することができます。

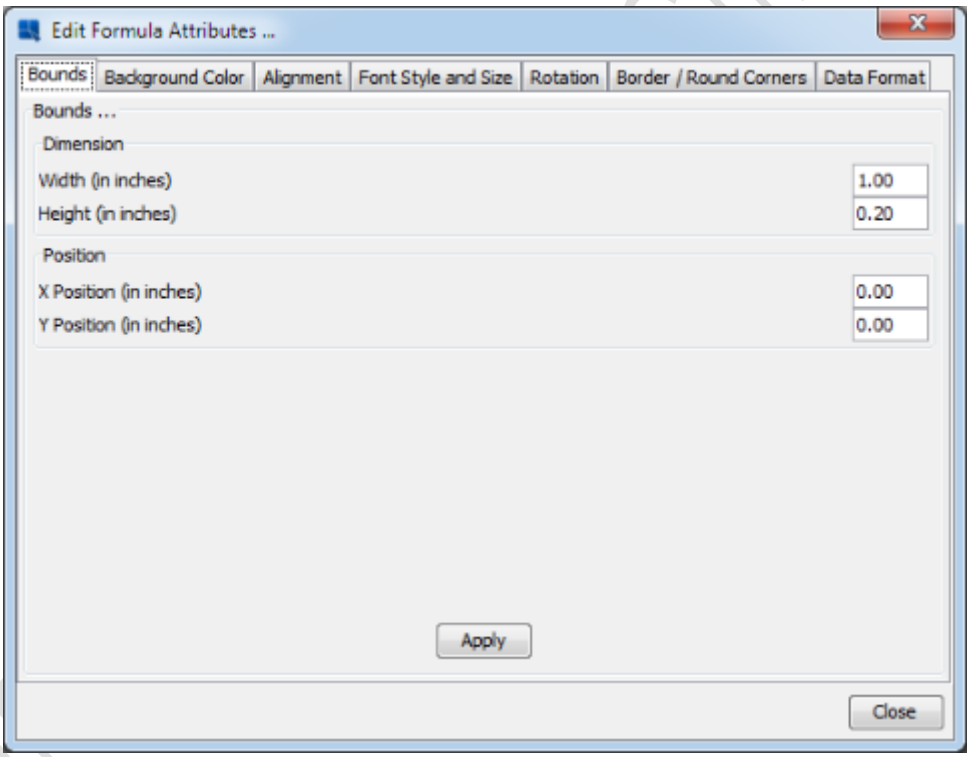
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

また、レポート要素間でセル属性をコピーして貼り付けることもできます。これにより、データの書式設定（データ型が同じであることを前提とする）、整列、フォント、色、境界線の属性、セルの境界などのすべての属性が適用されます。

要素の属性をコピーするには、その要素を右クリックし、ポップアップメニューから **Element Appearance** を選択します。選択すると 2 つの選択肢が出てきます。2 番目のメニューから **Copy** を選択します。コピーした属性を別のセルに適用するには、以前と同じ手順を繰り返し、セカンダリメニューから **Paste** を選択します。

3.5.7.2.1 要素属性の編集

要素を直接編集するだけでなく、**Format** メニューから **Edit Attributes** オプションを選択するか、要素を右クリックしてポップアップメニューから **Edit Attributes** を選択すると、要素のすべての属性に直接アクセスできます。これにより、タブのダイアログが表示され、要素のプロパティの数を一度に設定できます。



Multi-Attribute Editing ダイアログ

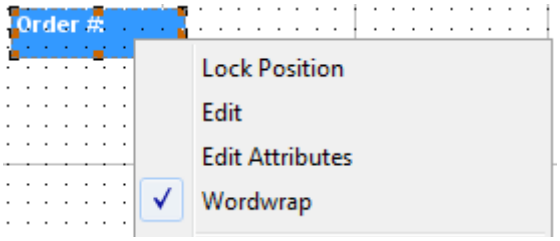
オプションは、選択する要素のタイプによって異なります。ダイアログの各タブについて、必要なオプションを設定してから、**Apply** ボタンをクリックして、その属性の変更を設定することができます。

3.5.7.2.2 ワードラッピング

デフォルトでは、ラベル、数式、および列を含むレポート要素内のテキストは、セル境界がテキストに収まらない場合は、次の行に折り返されます。ただし、この動作は無効にすることができます。テキストの折り返しを設定するには、要素を選択して **Format > Wordwrap** を選択するか、要素を右クリックしてポップアップメニューからオプションを選択します。これにより、テキストの折り返しが無効/有

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

効になります。

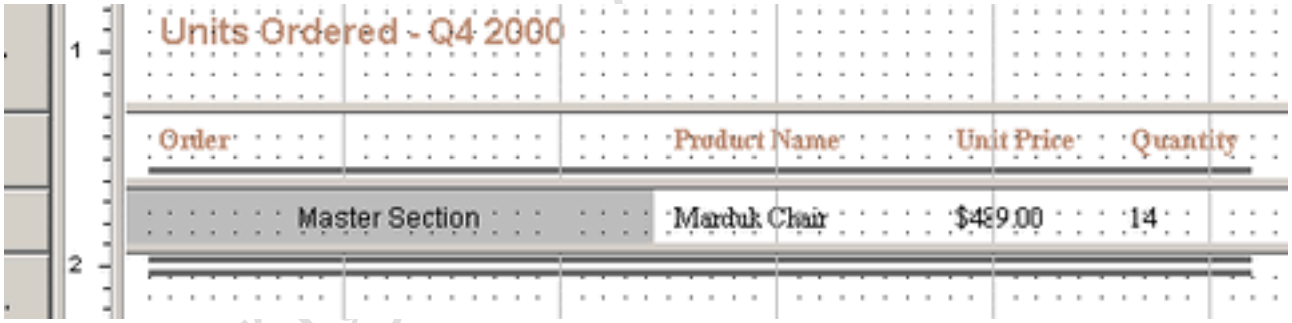


Word Wrapping ダイアログ

折り返しを無効にするには、オプションの選択を解除します。単語の折り返しが無効になっていると、セルの境界に関係なく、セルの内容全体が印刷されます。これは、他のレポート要素を覆い隠す可能性があります。また、リッチテキストフィールドと Excel ベースのエクスポートでは、ラップされたテキストが表示されるため、ワープロを無効にすることはできません。

3.5.7.2.3 サイドバイサイドマスタ&詳細レポートの編集

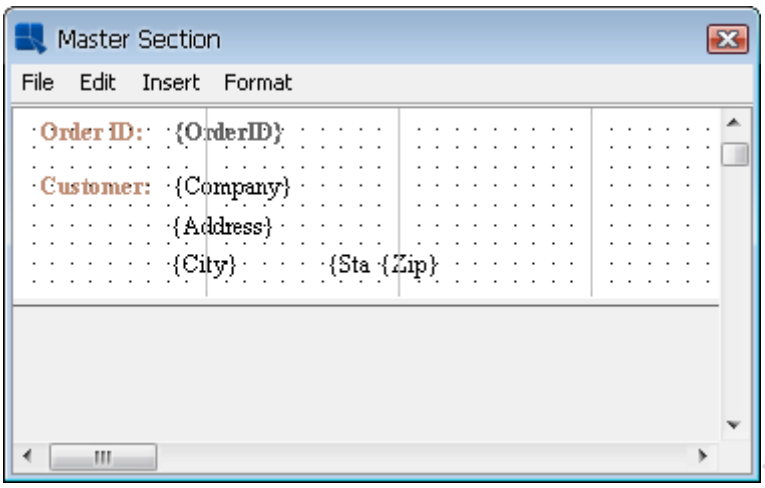
マスタ&詳細レポートで説明したように、マスタ&詳細レポートのための横並びのレイアウトを指定できます。これにより、詳細セクションの上のポジションではなく、詳細セクション（テーブルデータ）の隣にマスターフィールド（グループヘッダ）が印刷されます。このレポート形式を選択すると、マスタセクションはレポートのテーブルデータセクション内の列フィールドの横に灰色の四角形で表示されます。



サイドバイサイドマスタ&詳細

マスタセクションは、他のレポート要素と同様に自由形式で移動できます。マスタフィールドの内容を編集するには、ツールバーの **Edit** ボタンをクリックするか、右クリックしてポップアップメニューから **Edit** を選択するか、ダブルクリックします。マスタセクションが新しいウィンドウで開き、その内容を変更できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Edit Master Section ダイアログ

このセクションの要素は、レポート内の他の要素と同じ方法で移動、サイズ変更、および編集できます。マスタセクションは、グループヘッダーセクションと同じように機能します。これは、主キー列の各一意の値に対して繰り返されます。マスタセクションの内容の編集が完了したら、**File** メニューから **Close** を選択して閉じます。

マスタセクションの編集が終了すると、その内容に合わせて自動的にリサイズされます。

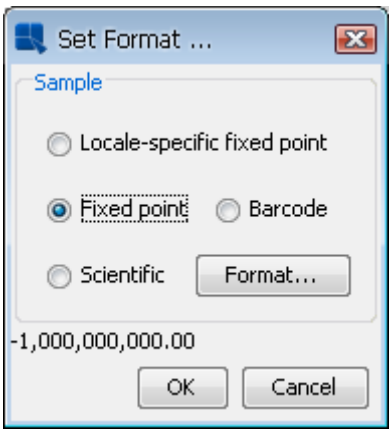
3.5.7.3 数式と列フィールドのデータ書式設定

数式と列フィールドの出力は、次の 3 つの方法のいずれかで変更できます：ツールバーの **Data Format** アイコンをクリックするか、**Format** メニューから **Data Format** を選択するか、数式または列フィールドを右クリックし、ポップアップメニューから **Edit Attributes** を選択します。フォーマットオプションを選択すると、ダイアログボックス（またはマルチアトリビュート編集ダイアログのフォーマットタブ）が表示されます。表示されるボックスは、選択した要素に存在するデータの種類（数値、文字列、日付/時刻、または論理/ブール値）によって異なります。

数値データの書式設定：

数値データのダイアログボックスには、**locale-specific fixed point**(ロケール固有の固定小数点)、**fixed point**(固定小数点)、**bar code**(バーコード)、および **scientific**(科学)の 4 つの主なデータオプションが含まれています。必要なオプションを選択してから、オプションを追加するには **Format** をクリックします。

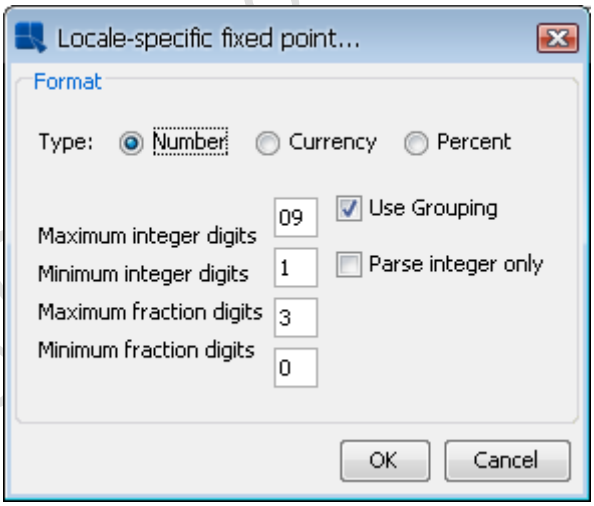
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Numeric Format ダイアログ

ロケール固有の固定小数点：

表示されているロケールに応じてデータ形式が変更されます。このオプションの追加の書式設定では、データを数値、通貨、またはパーセンテージとして表示するかどうかを指定できます。さらに、整数桁数と小数点桁数の最大値と最小値を設定できます。他の表示属性はロケールによって異なります。

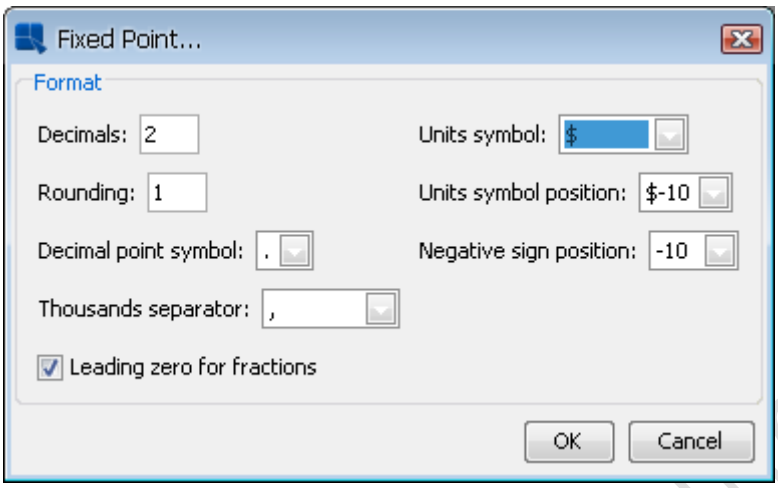


ロケール固有の書式設定

固定小数点：

ロケールに関係なくデータフォーマットが一貫して保持されます。このオプションの追加の書式設定では、小数点の桁数、桁数、単位記号、負の符号の位置、小数点と桁区切り記号を四捨五入し、小数部に先行ゼロを指定することができます。

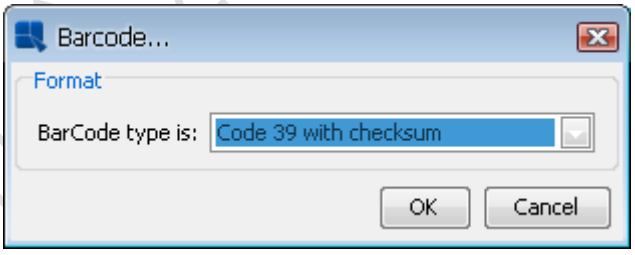
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



固定小数点の書式設定

バーコード：

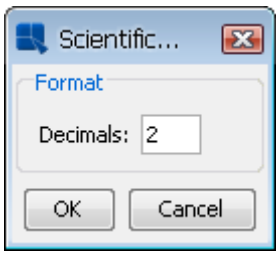
これにより、データがバーコードに変換されます。このオプションの追加の書式設定では、バーコードに使用するシンボルを選択できます。サポートされているシンボルは、チェックサム有/無の Code 39、Interleave 2 of 5、UPC A、EAN 13、EAN 128、チェックデジット有/無の Standard 2 of 5、Code 128、Code 128A、Code 128B、Code 128C、チェックサム有/無の USD 3、チェックデジット有/無の Standard 3 of 9、世界貿易品目番号、ランダム重量 UPCA、SCC 14 出荷コード、出荷識別番号、SSCC 18、および米国郵便サービスです。



バーコードの書式設定

科学：

これにより、科学的表記法でデータが表示されます。このオプションの追加の書式設定では、小数点以下の桁数を設定できます。



科学の書式設定

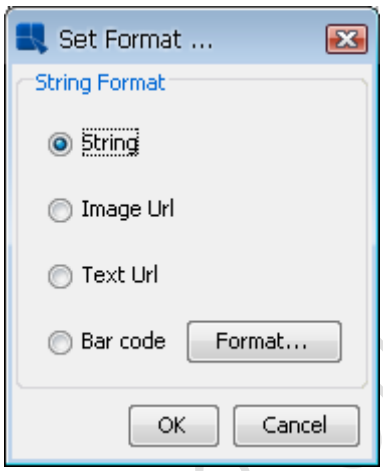
追加オプションの選択が終了したら、**OK** ボタンをクリックしてメインダイアログボックスに戻ります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

OK ボタンをクリックすると、データフォーマットが変更されます。

文字列データの書式設定：

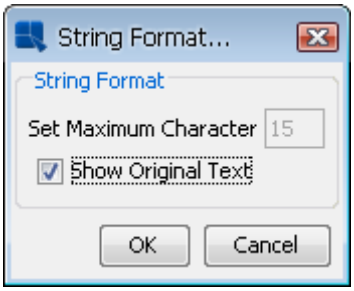
文字列データのダイアログボックスには、文字列、画像 URL、テキスト URL、バーコードの 4 つの主要なオプションがあります。必要なオプションを選択できます。文字列またはバーコードの場合は、**Format** をクリックして追加オプションを選択します。



文字列データの書式設定

String：

このダイアログでは、文字列の外観を書式設定できます。**Show Original Text** チェックボックスは、各データエントリの完全な文字列が表示されるかどうかを制御します。チェックを外すと、表示される最大文字数を指定できます。**OK** ボタンをクリックすると前のダイアログに戻ります。



文字列データの書式設定

Image URL：

文字列を画像に変換します。これは、データベース（BLOB）にオブジェクトとしてイメージを格納する代わりに、イメージファイルを指す URL をサーバ上に格納している状況で使用されます。このオプションを選択すると、URL が読み込まれ、レポートに配置されるイメージが取得されます。

Text URL：

文字列を大きなテキストオブジェクトに変換します。これは、データベース（CLOB）

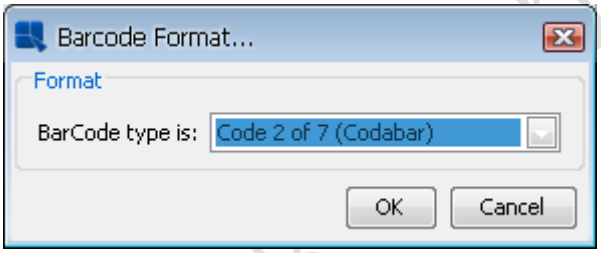
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

にオブジェクトとして大きなテキストファイルを格納する代わりに、サーバ上のファイルを指す URL を格納している状況で使用されます。このオプションを選択すると、URL が読み込まれ、レポートに配置されるテキストファイルが取得されます。

Bar Code：

これにより、データがバーコードに変換されます。このオプションの追加の書式設定では、バーコードに使用するシンボルを選択できます。文字列データのための利用可能な記号は、チェックサム有/無 Code 39、コード 2/7(Codabar)、Codabar、EAN 128、Code 128、Code 128A、Code 128B、Code 128C、世界貿易品目番号、Monarch、NW 7、SCC 14 出荷コード、出荷識別番号、SSCC 18、米国郵便サービス、および USD 4。

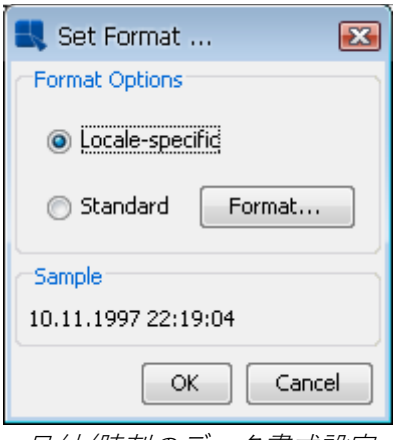


バーコード書式設定

Codabar はデータの開始/停止文字を受け入れません。入力データが間違っていると、バーコードが読み取れないことがあります。

日付/時刻データの書式設定：

日付/時刻データのダイアログボックスには、ロケール固有と標準の 2 つの主要なデータオプションが含まれています。必要なオプションを選択し、**Format** ボタンをクリックして追加のオプションを選択することができます。使用可能な追加オプションは、データの性質によって異なります。日付、時刻、およびタイムスタンプのデータはそれぞれ、日付、時刻、および日付と時刻のオプションを表示します。

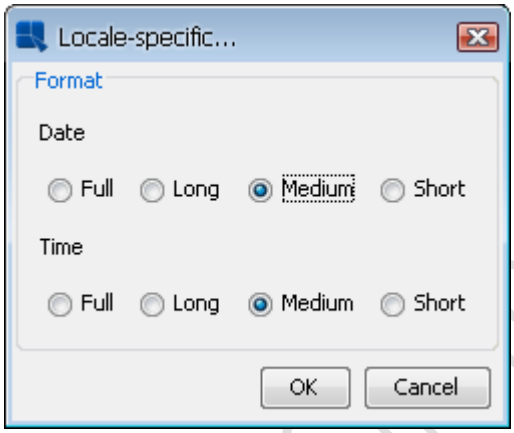


日付/時刻のデータ書式設定

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Locale Specific：

これにより、表示されているロケールに応じてデータのフォーマットが変更されます。このオプションの追加の書式設定では、日付と時刻情報の完全（Full)、長（Long)、中（Midium)、短い（Short）表記を選択できます。他の表示属性はロケールによって異なります。



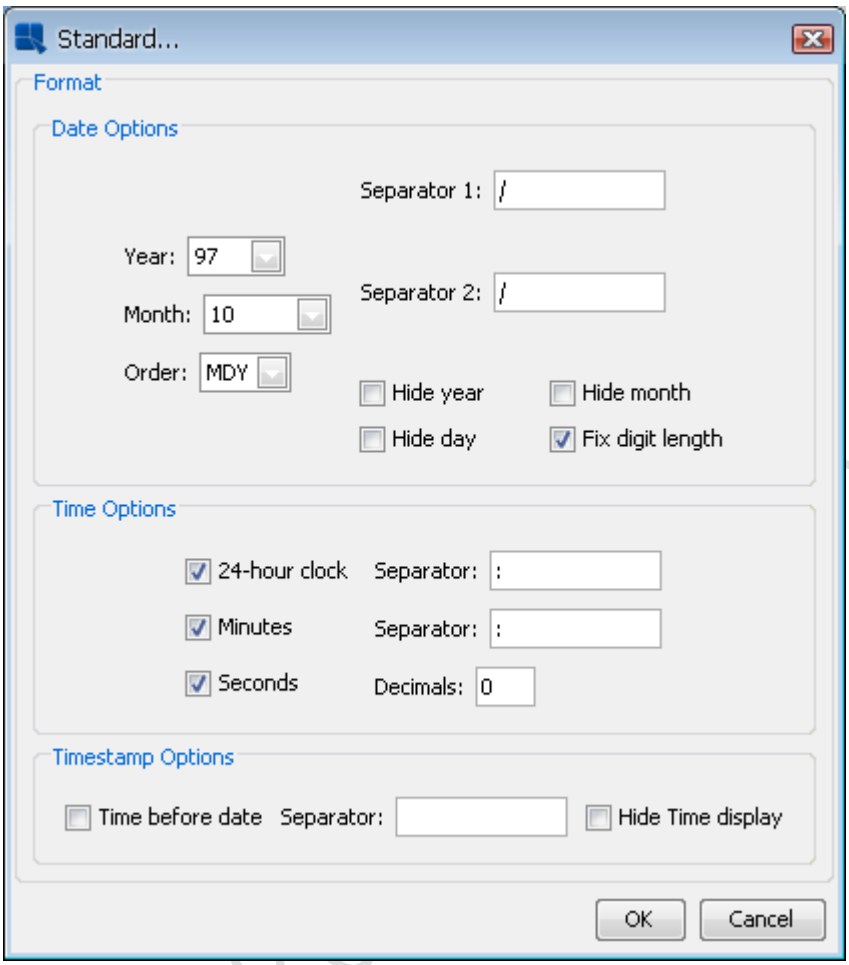
ロケール書式設定

Standard

これにより、ロケールに関係なくデータフォーマットが一貫して保持されます。このオプションの追加の書式設定では、年および月の表示、および月、日、年の情報の表示順序を選択できます。セパレータとして使用する文字を選択することもできます。時間オプションを使用すると、時間、分、秒を表示し、それらの間のセパレータを選択できます。タイムスタンプデータの場合は、日付の前後の時間とそれらの間に使用されるセパレータを表示するように選択できます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



標準の形式

追加の設定の選択が完了したら **OK** ボタンをクリックしメインダイアログボックスに戻ります。**OK** ボタンをクリックするとデータ書式設定が変更されます。

論理演算データの書式設定

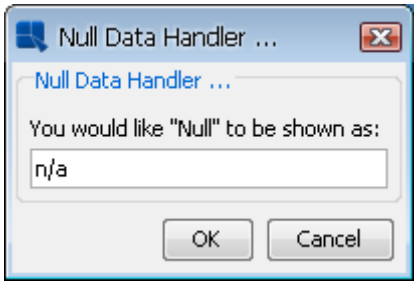
論理データまたはブールデータのダイアログボックスには、データを表示するための 5 つのオプションがあります。T/F、True/False、Yes/No、Y/N、1/0 です。使用する書式設定を選択し、**OK** ボタンをクリックしてデータ書式設定を変更します。

ブールデータの書式設定

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.5.7.3.1 Null データの書式

デフォルトでは Null データはレポート内で **Null** として表示されます。**Option** メニューから **Null Data Handler** を選択することで Null データの表示を変更できます。ダイアログが立ち上がると、Null をどう表示するか指定するよう求められます。



Null Data Handler

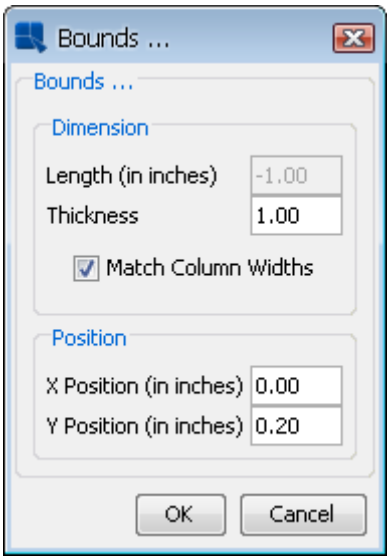
表示したい値を入力したら **OK** ボタンをクリックします。レポート内の全ての Null 値が指定した値の表示に切り替わります。

3.5.7.4 ライン/長方形フォーマット

ラインまたは四角形の要素を右クリックして、レポート内の線と四角形の外観を設定できます。ポップアップメニューには、使用可能なオプションのリストが表示されます。

ライン:

ラインには 3 つの書式設定オプションがあります。**Line Color & Thickness** では、ラインの色と太さを設定できます。**Line Style** は、実線、点線、または二重のいずれかよりラインのスタイルを指定できます。**Set Bounds** では、ラインの長さと太さ、および原点の X 座標と Y 座標を指定できます。さらに、水平線の幅をレポートの列の合計幅に合わせて設定することもできます。この機能は、列数が異なる可能性のあるクロス集計レポートやユーザ定義のレポートスタイルにも役立ちます。縦線の場合、セクションの高さに合わせて長さを設定できます。



Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Bounds ダイアログ – 水平線

長方形：

グリッド長方形には 4 つの書式設定オプションがあります。**Border** では、グリッドの矩形の罫線の太さと色を設定できます。**Background** では、長方形の塗りつぶし領域の背景色を設定できます。**Set Bounds** では、グリッド矩形の幅と高さ、その原点の X 座標と Y 座標、および角の丸めの度合いを指定できます。**Border Style** は、実線、点線、または二重のいずれかより、罫線の線のスタイルを選択することができます。

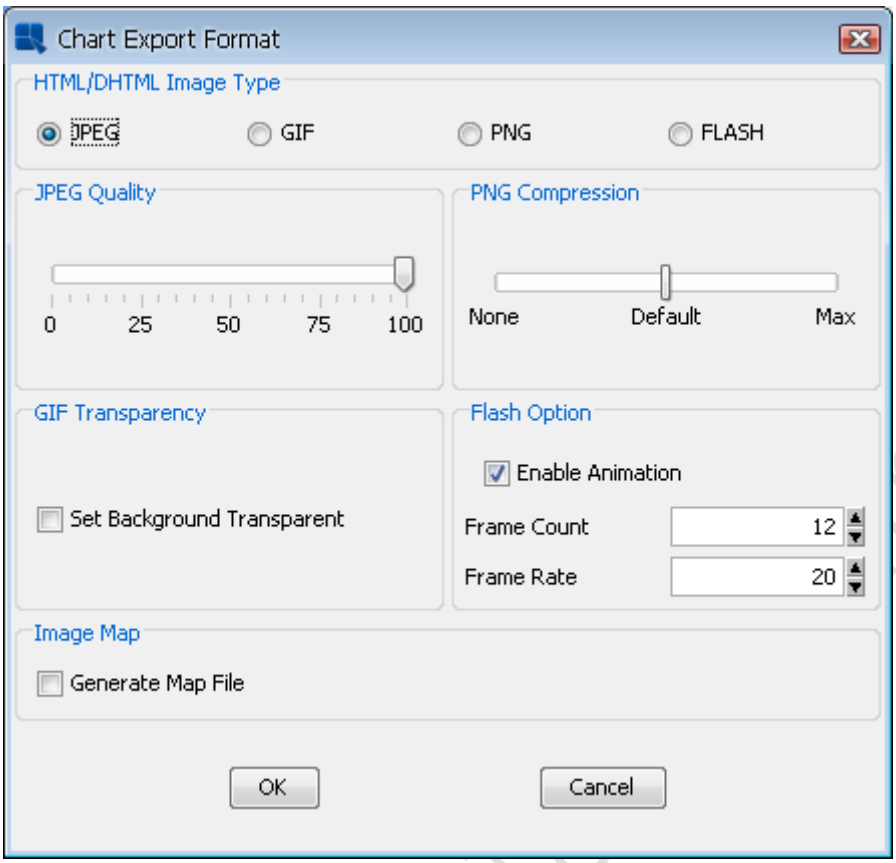
3.5.7.5 グラフのエクスポート形式

このオプションを使用すると、DHTML にエクスポートされたチャートの属性を指定できます。**Format** メニューから **Chart Export Format** を選択するか、グラフを右クリックしてポップアップメニューから **Export Format** を選択して、書式エクスポートの書式を設定できます。このオプションを選択すると、ダイアログボックスが表示されます。イメージタイプには GIF、PNG、JPEG、または FLASH を選択し、それぞれのタイプにいくつかのオプションを設定できます。GIF 画像の場合、背景を透明に設定できます。PNG イメージの場合、圧縮を設定できます。JPEG 画像の場合、画質を設定できます（画質はファイルサイズに比例します）。FLASH 画像の場合、アニメーションを有効にしてフレーム数とフレームレートを設定できます。

また、レポートのエクスポート時にイメージマップを作成するように指定することもできます。イメージマップを作成すると、レポートをエクスポートするときにチャートに定義されているハイパーリンクが有効になります。ハイパーリンクが定義されていない場合、マップにはデータポイント情報を含むチャートの表示ポップアップラベルが含まれます。FLASH イメージはイメージマップをサポートしていないことに注意してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



チャートエクスポートの形式

3.5.7.6 フォント、色、および罫線のオプション

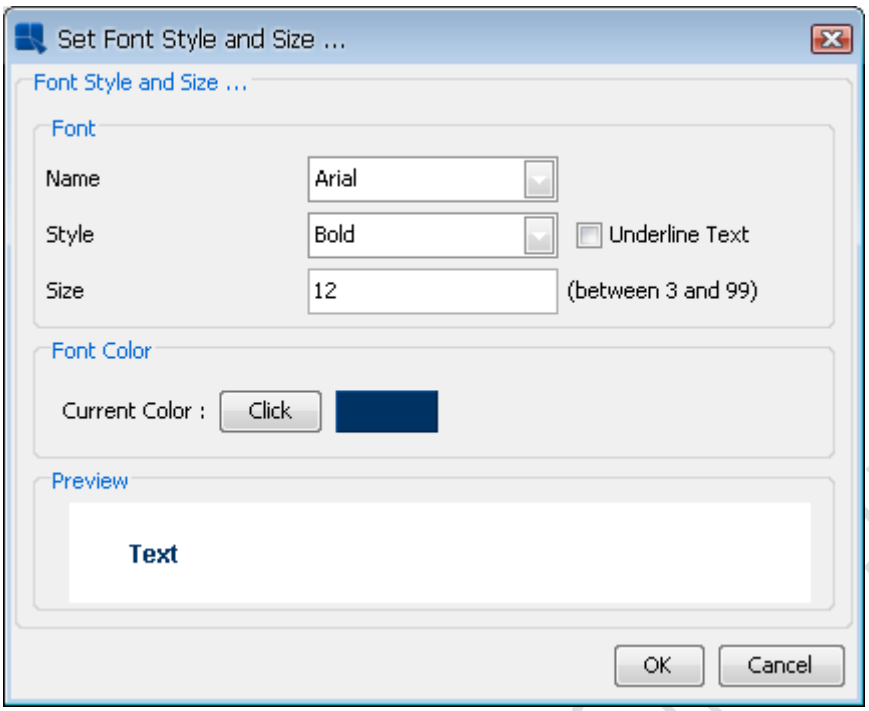
フォント、色、罫線のプロパティを変更して、レポート要素の外観を変更できます。

フォントスタイルとサイズ：

フォントスタイルとサイズは、ラベル、数式、および列フィールドの 3 つの方法のいずれかで調整できます。Report Designer ツールバーのオプションを使用して、フォントスタイルとサイズを直接変更できます。また、**Format** メニューから **Font Style and Size** を選択するか、レポート要素を右クリックして、ポップアップメニューから **Edit Attributes** を選択することでもできます。後者の 2 つのオプションは、フォントスタイルとフォントサイズを選択するか、属性編集ダイアログで別のダイアログを表示します。ここで、フォントオプションは'Font Style and Size'タブで設定できます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Font Style and Size ダイアログ

レポートを設計するときは、システムの任意のフォントを使用できますが、プラットフォーム間でレポートを移動すると、そのフォントが存在しなくなることがあります。また、レポートを PDF にアクセスするときは、フォントファイル（.ttf）をフォントに明示的にマッチングする必要があります。詳細は、PDF フォントマッピングを参照してください。

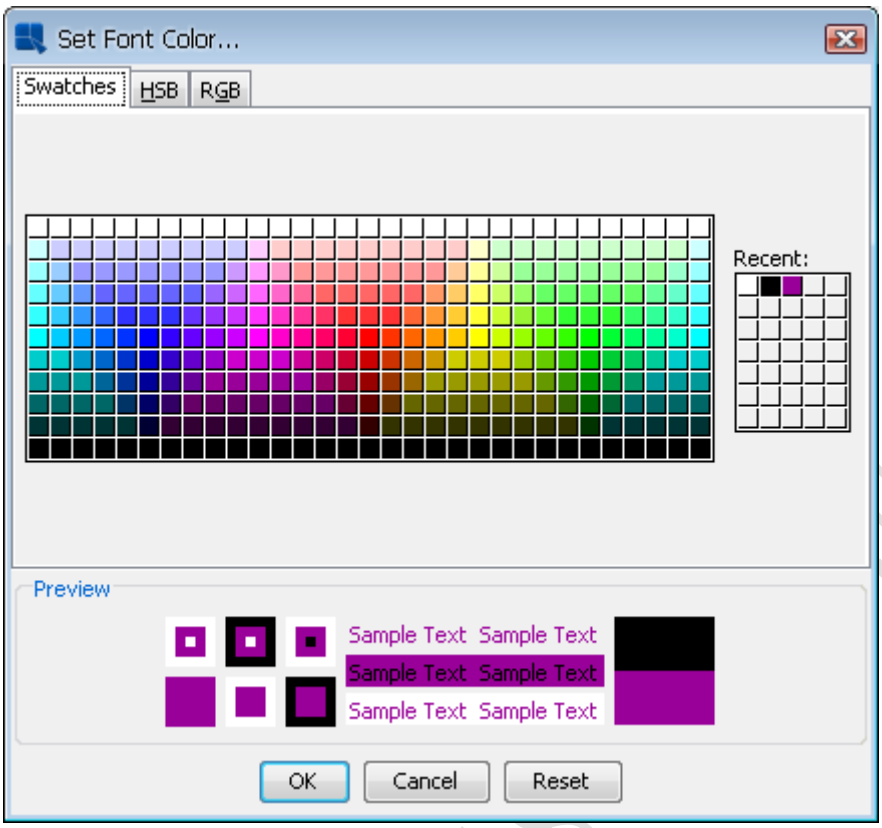
Alignment：

ラベル、式、および列フィールドの配置は、3 つの方法のいずれかで調整できます。ツールバーからオプションを選択すると、要素の水平方向の配置を直接調整できます。**Format** メニューから **Alignment** を選択するか、要素を右クリックしてポップアップメニューから **Edit Attributes** を選択することもできます。後者の 2 つのオプションでは、要素テキストの垂直方向と水平方向の配置を選択することもできます。

Font Color：

フォントの色は、ラベル、式、および列フィールドの 3 つの方法のいずれかで調整できます。**Format** メニューから **Font Style And Size** を選択するか、ツールバーの **Font Style And Size** ボタン をクリックするか、要素を右クリックして **Edit Attributes** を選択します。このオプションを選択すると、ダイアログボックス（または **Font Style And Size** タブでオプションを設定できるマルチ属性編集ダイアログ）が表示されます。フォントの色を変更するには、**Current Color:**フィールドの横にある **Click** ボタンをクリックします。フォントカラーを設定するダイアログが表示され、スウォッチからフォントカラーを選択できます。HSB、HEX、RGB の値を入力することもできます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



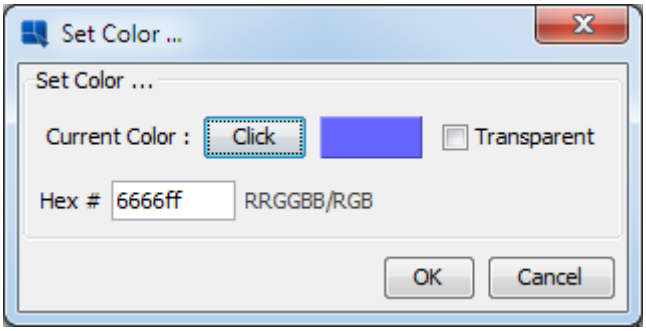
Font Color ダイアログ

Background Color：

背景色は、次の 3 つの方法のいずれかで、ラベル、数式、および列フィールドに対して調整できます。

Format メニューから **Background Color** を選択、ツールバーの **Background Color** ボタン を クリックして、背景色を選択、または要素を右クリックし、ポップアップメニューから **Edit Attributes** を選択します。

このオプションを選択すると、背景を透明に設定するかどうかを指定するプロンプトが表示されます。ダイアログボックス（または **Background Color** タブでオプションを設定できる複数属性編集ダイアログ）が表示されます。背景が透明でないようにするには、チェックボックスをオフにしてボタンをクリックします。これにより、背景色を指定するダイアログが表示されます。スウォッチからフォントカラーを選択するか、HSB または RGB 値を入力します。Hex カラー値を **Hex** # フィールドに直接入力することもできます。



Background Color ダイアログ

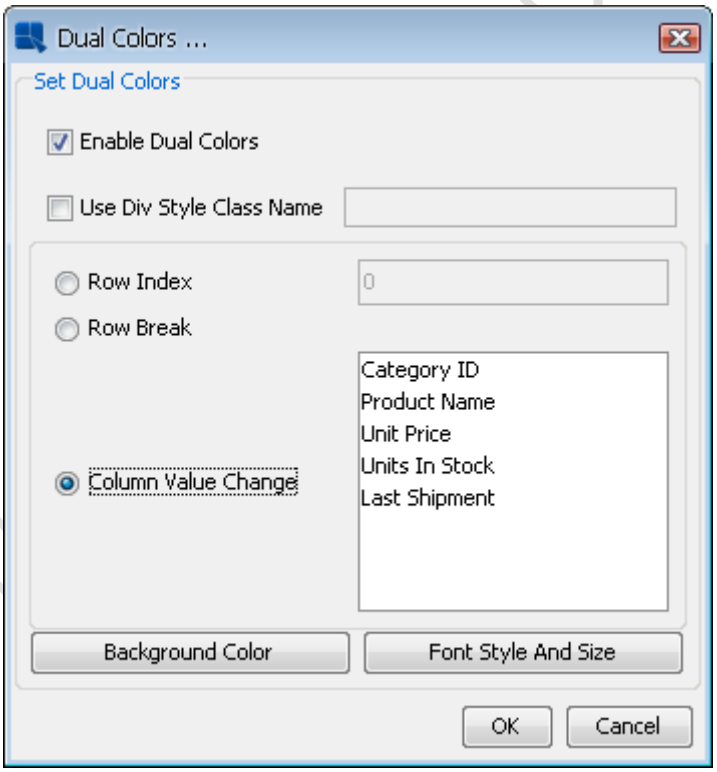
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Dual Colors：

次の 3 つの方法のいずれかを使用して、レポートのデータテーブルセクションにあるラベル、数式、または列フィールドの色を交互に指定できます。**Format** メニューから **Dual Colors** を選択し、ツールバーの **Dual Colors** ボタン をクリックするか、要素を右クリックし、ポップアップメニューから **Edit Attributes** を選択します。

このオプションを選択すると、ダイアログボックス（または、**Dual Colors** タブでオプションを設定できるマルチアトリビュート編集ダイアログ）が表示され、交互の色の間の行数を指定することができます。代替行番号を指定する代わりに、行ブレーク（サマリブレークまたはクロス集計レポートの場合）または特定の列値の変更に応じて変更する 2 色を設定することもできます。この機能を使用すると、サマリブレークレポートで新しいグループに到達したときに変更するセル属性を設定できます。

代替行の場合、背景色、フォントの色、フォントのスタイルとサイズを設定できます。その他のレポート要素の場合と同じように、主属性を設定します。



Dual Colors ダイアログ

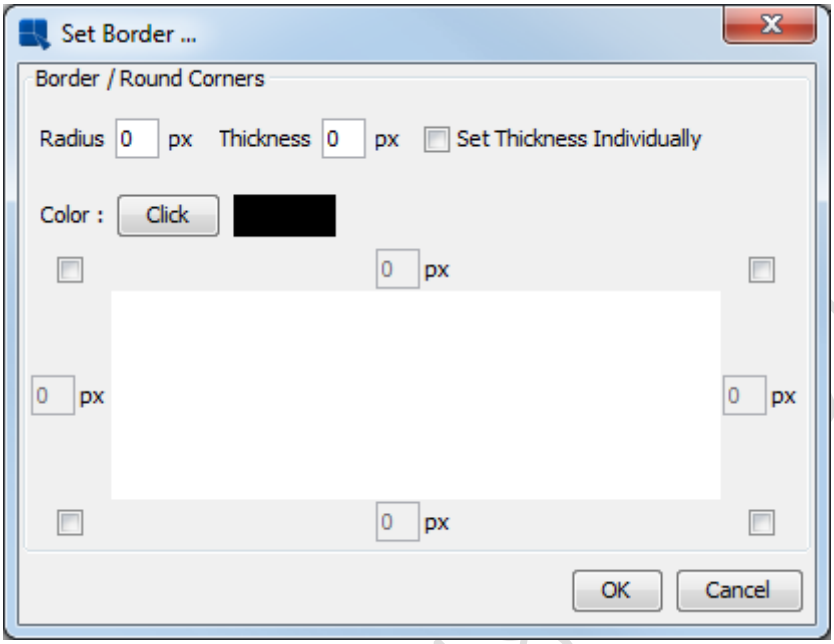
Border：

次の 3 つの方法のいずれかを使用して、レポート要素の周りに描画される罫線の太さと色を指定できます。

Format メニューから **Borders** を選択するか、ツールバーの **Border** ボタン を選択するか、要素を右クリックし、ポップアップメニューから **Edit Attributes** を選択します。このオプションを選択すると、ボーダー/ラウンドコーナータブでオプションを設定できるダイアログボック

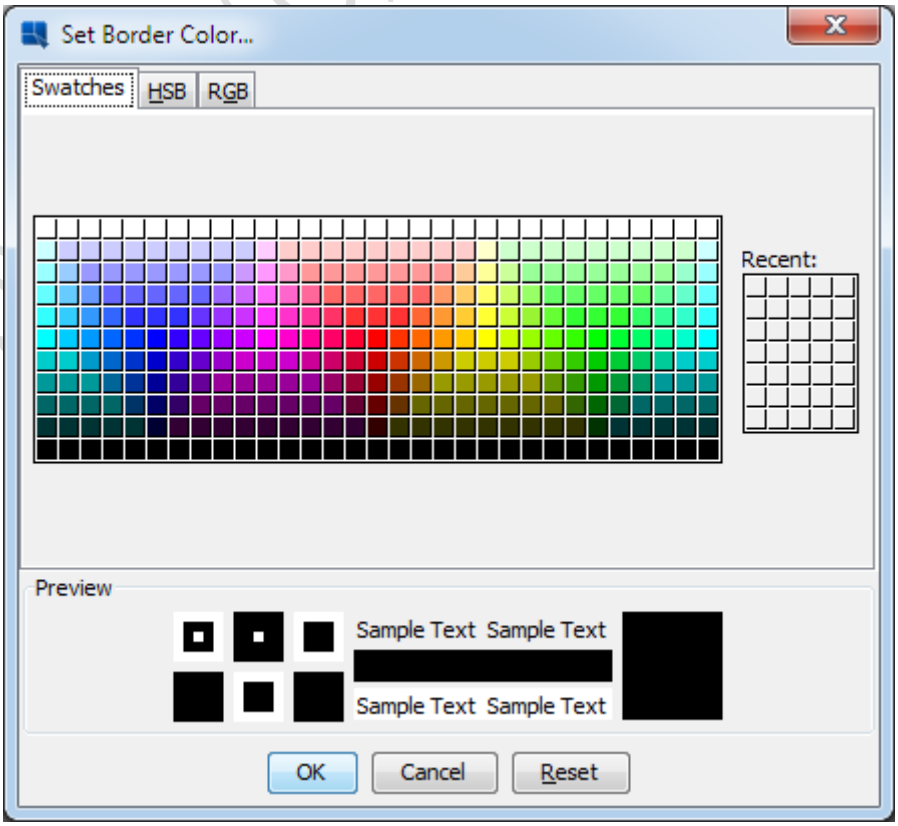
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ス（または、**Border/Round Corners** タブでオプションを設定できるマルチアトリビュート編集ダイアログ）が表示され、ボーダーの太さをピクセル単位で入力し、色を選択するように求められます。ボーダーの太さの値として 0 を入力すると、ボーダーが削除されます。



Border ダイアログ

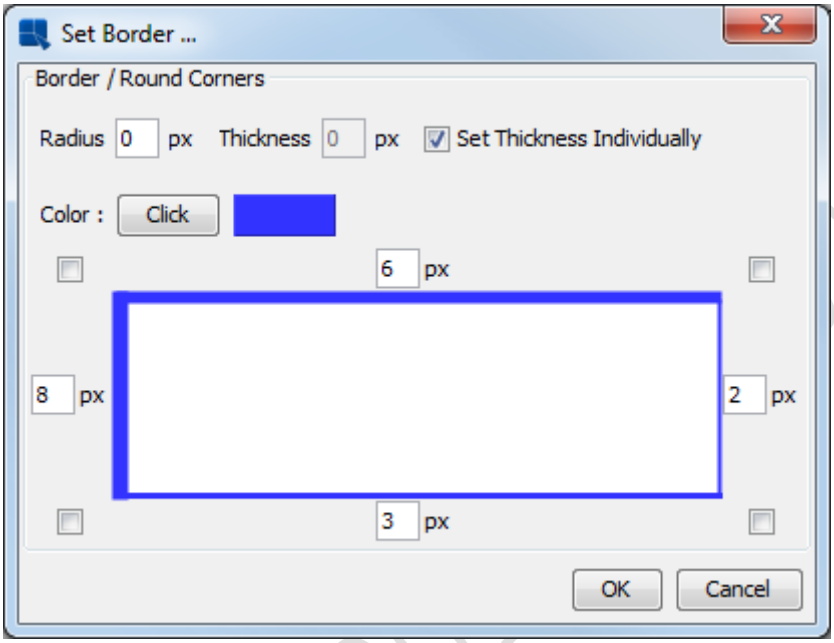
罫線の色を変更するには、**Color:**フィールドの横にある **Click** ボタンをクリックします。**Set Border Color** ダイアログが表示され、見本から罫線の色を選択できます。HSB、HEX、RGB の値を入力することもできます。



Copyright(C) 2018 Climb.Inc. All Rights Reserved.

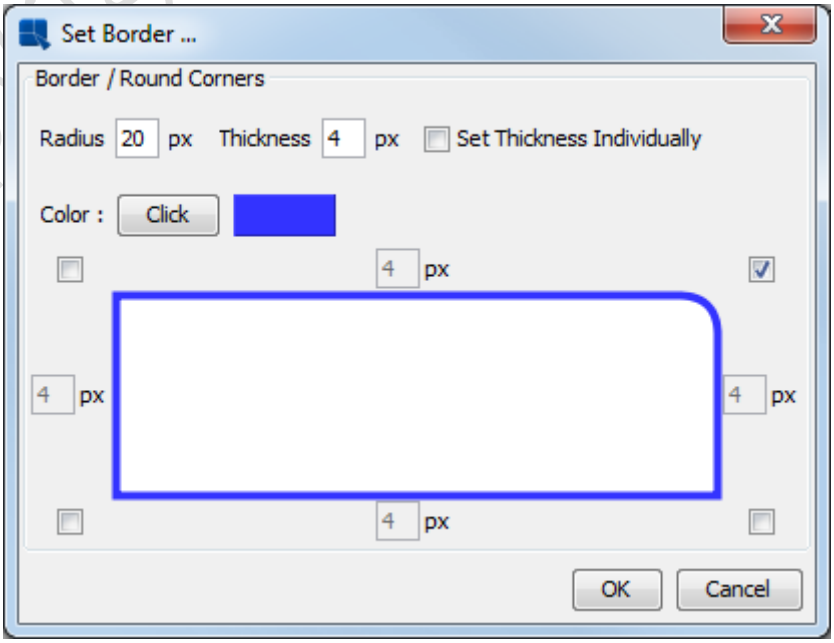
Border Color ダイアログ

レポート要素の各辺の罫線の太さを設定するには、**Set Thickness Individually** オプションをオンにします。次に、選択した要素を表す四角形の辺の近くのテキストフィールドに太さ（ピクセル単位）を入力します。



罫線/個別の線の太さ

丸い角を要素に設定するには、要素が透明でない背景色または目に見える境界線のいずれかを持つ必要があります。変更しないと、要素は変更されません。コーナーの近くに 4 つのチェックボックスがあり、各チェックボックスは要素コーナーを表します。角を丸めるには、チェックボックスをオンにして（一度に複数の角を選択することもできます)、半径テキストフィールドに半径（ピクセル単位）を入力します。



Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Border/Round Corners ダイアログ

丸めた角を使用してレポートをエクスポートする場合は、DHTML および PDF エクスポート形式でのみ使用できることに注意してください。

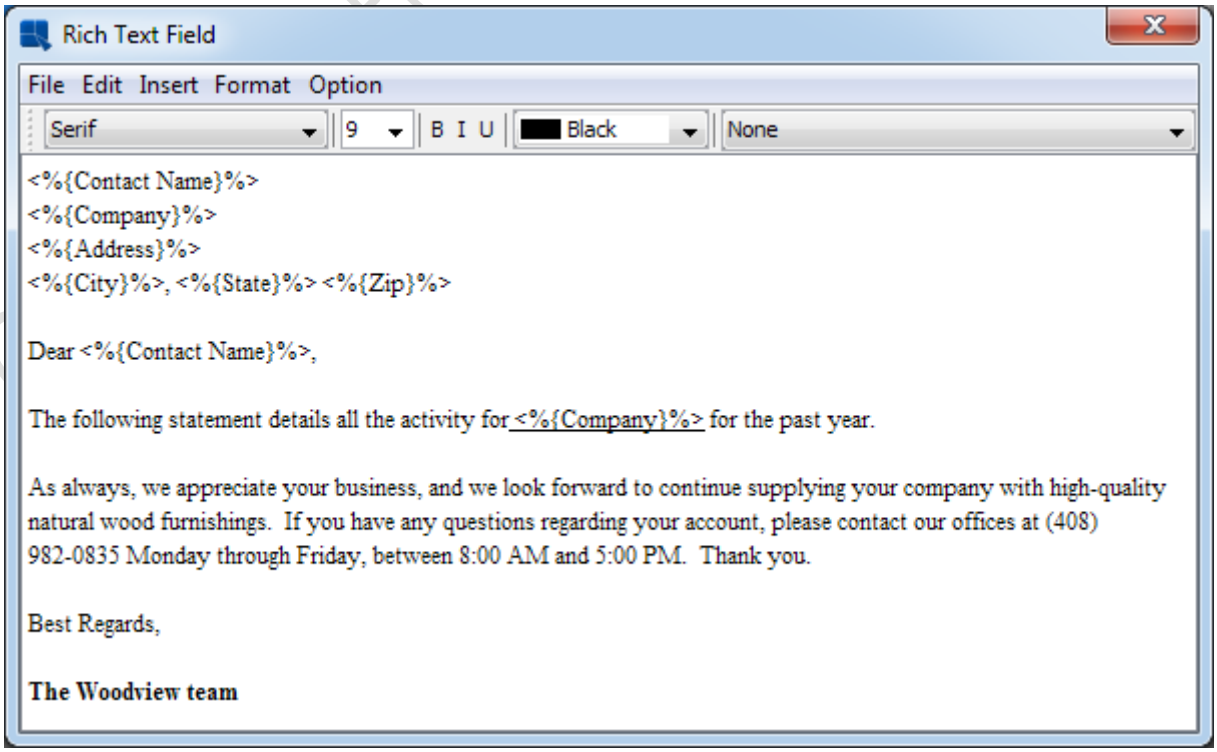
3.5.7.6.1 セルのアウトラインを表示

実際に枠線をオンにすることなく、デザインウィンドウでレポート要素の境界や境界線を表示したい場合があります。最初のレコードが空のプレースホルダーセルまたは列がある場合、これは特に重要です。セルの境界線をオンにするには、Option メニューから **Show Cell Outline** を選択します。これにより、デザインウィンドウの各レポート要素の周囲に灰色の点線の境界線が描画されます。

3.5.7.7 リッチテキストフィールド

一般的なレポートの必要性の 1 つに、書式文字やその他の書式付きテキストブロックをレポートに作成し、データフィールドと関数をマージする機能があります。この機能は、リッチテキストフィールドを使用して使用できます。リッチテキストフィールドは、ラベルや文字列関数フィールドとは異なり、セル内の複雑な段落、フォント、および色の書式設定を可能にします。また、列フィールドと関数をテキストフローに直接埋め込むこともできます。

リッチテキストフィールドをレポートに追加するには、Insert メニューの **Insert Rich Text Field** を選択するか、ツールバーの **Rich Text Field** ボタン をクリックします。このオプションを選択すると、デザインウィンドウの周りに小さなボックスがマウスポインタの後に表示されます。フィールドを挿入する場所にボックスを置き、それをクリックします。新しいウィンドウが開き、リッチテキストフィールドを入力することができます。

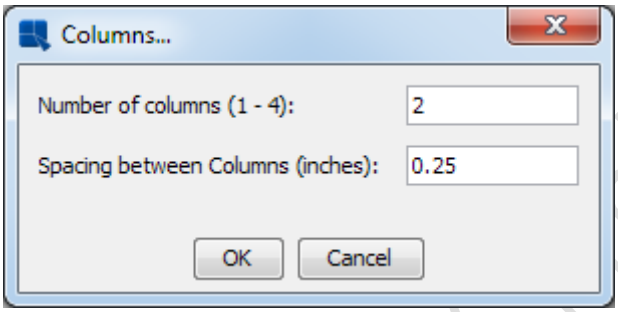


リッチテキストエディタ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

リッチテキストエディタは、小さなワープロのように動作します。最初のドロップダウンボックスでは、フォントとフォントサイズを選択できます。3 つのボタンで太字、斜体、または下線のテキストを指定できます。次のドロップダウンボックスでは、フォントの色を指定できます。最後の 1 つでは、関数または列フィールドの書式を選択できます。

Option メニューから **Columns** を選択すると、テキストの列レイアウトを指定できます。これにより、リッチテキストフィールドの列数と列間の間隔を指定できるダイアログが表示されます。



Rich Text Column Options ダイアログ

リッチテキストフィールドのテキストフローにインラインイメージを追加することもできます。イメージを追加するには、**File** メニューから **Insert Image** を選択します。インポートする画像ファイルを指定するダイアログが表示されます。これを指定すると、新しいダイアログが開き、インラインイメージのサイズをピクセル単位で指定できます。サイズを設定したら、Ok ボタンをクリックすると、画像がリッチテキストフィールドに追加されます。

任意のリッチテキストファイル（.rtf 形式）をリッチテキストフィールドにインポートできます。これを行うには、エディタの **File** メニューから **Import** を選択します。これにより、インポートするファイルを指定するダイアログが表示されます。

リッチテキストファイルをインポートすると、特定の段落設定が失われます。

3.5.7.7.1 数式の追加

任意の数式をリッチテキストフィールドのテキストフローに直接追加できます。列フィールドとパラメータ値を追加することもできます。数式、列フィールド、またはパラメータ値を挿入するには、**Insert** メニューをクリックし、カテゴリ（列、数式またはパラメータ）を選択し、使用可能なすべてのフィールド/パラメータ/式のリストから項目を選択します。次の構文がテキストに追加されます。

```
<%formula name/parameter name/column field name%>
```

Insert > **Formula** リストには、レポートのすべての数式が含まれます。また、カスタム式をリッチテキストで直接作成することもできます。これを行うには、**<%Formula Syntax%>**構文を入力します（実際の数式構文で式構文を置き換えます）。式の構文の詳細については、数式と式ビルダの使用を参照し

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

てください。このアプローチは、単純な数式の場合にのみ推奨されます。より複雑な数式を挿入する場合は、式ビルダを使用してレポート内に作成し、**Insert** > **Formula** メニューから選択してリッチテキストフィールドに追加できます。式ビルダで数式を作成する方法については、数式および式ビルダの使用を参照してください。

式と列フィールドの書式設定をするには、まず書式をあらかじめ定義しておく必要があります。これを行うには、エディタの **Format** メニューから **New** を選択し、書式を作成するデータ型を選択します。これにより、そのデータ型の書式設定ダイアログが表示され、書式を定義することができます。書式設定ダイアログを閉じると、書式設定の名前を指定するよう求められます。たとえば、ロケール固有の通貨書式を数値に使用する **Currency** という書式を定義できます。

数式の事前定義された書式を適用するには、数式を選択し、リッチテキストエディタの右側にあるドロップダウンボックスから適用する書式を選択します。

3.5.7.8 ハイパーリンク

列フィールド以外の任意のレポート要素にハイパーリンクを適用できます。要素にハイパーリンクを適用するには、その要素を選択し、**Format** メニューから **HyperLink** を選択するか、要素を右クリックし、ポップアップメニューから **HyperLink** を選択します。次に、リンク、ヒント、およびターゲットの入力を求めるダイアログボックスがポップアップ表示されます。任意の URL をリンクまたは別の.pak ファイルとして入力できます。ヒントは、レポートが表示されたときにヒントボックスにマウスを重ねるとテキストを表示します。

EspressReport をスタンドアロンモードで実行している場合、URL へのハイパーリンクはプレビューウィンドウでは機能しません。DHTML または PDF にエクスポートすると.pak ファイルへのリンクが機能しなくなります。

Hyperlink ダイアログ

セルスクリプトを使用して、列フィールドを含むレポート要素にダイナミックハイパーリンクを適用することもできます。セルスクリプトの詳細については、スクリプトを参照してください。

3.5.7.9 列にラップ

この機能は、1 ページよりも長いデータ列が少ない、または狭いレポートの場合に便利です。データがページ幅の一部だけを占有し、複数のページにまたがるのではなく、レポートの最初のページの元の列の右に続くように列をラップすることができます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

たとえば、次のレポートの商品名とオーダー数のレポートがあるとします。

Product Name	Units Ordered
Adad Chair	4
Addad Dresser	5
Amon Table	2
An Chair	5
Anahita Dresser	3
Anubis Table	7
Apep Table	10
Apsu Dresser	3
Asherat Dresser	2
Atun Table	4
Bast Table	4
Bes Table	3
Cula Chair	10
Enki Chair	12
Enlil Chair	7
…	…
…	…

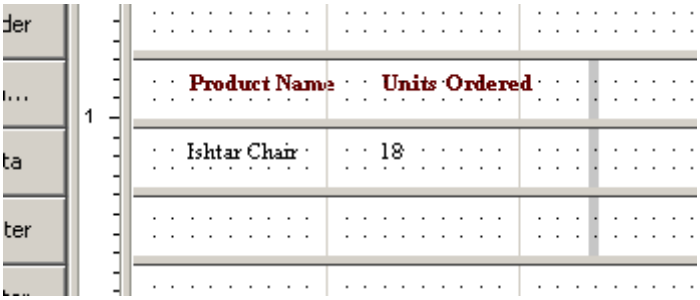
このレポートが複数のページを包含するのに十分長いと仮定すると、より少ないページを使用してページ幅の空白を埋めるため、列の折り返しを使用することが理想的です。 レポートは列の折り返しで次のようになります。

Product Name	Units Ordered	Product Name	Units Ordered
Adad Chair	4	Bast Table	4
Addad Dresser	5	Bes Table	3
Amon Table	2	Cula Chair	10
An Chair	5	Enki Chair	12
Anahita Dresser	3	Enlil Chair	7
Anubis Table	7	…	…
Apep Table	10	…	…
Apsu Dresser	3		
Asherat Dresser	2		
Atun Table	4		

列の折り返しは、簡単な柱状および要約のブレーク・レポート・タイプで使用できます。列の折り返し

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

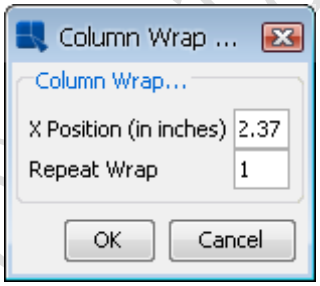
を実装するには、**Format** メニューから **Column Wrap** を選択します。列の折り返し配置を示すデザインウィンドウの中央に灰色の縦線が描画されます。



デザインウィンドウでの列の折り返し

列の折り返しはすぐにバーの右側に始まります。したがって、十分なスペースを確保してください。また、列の折り返しバーの右側にあるものは切り捨てられます。

列ラッププロパティを編集するには、最初に列ラップバーを選択し、**Format** メニューから **Column Wrap** を選択します。列ラップバーを右クリックし、ポップアップメニューから列ラップを選択することもできます。ダイアログボックスが表示され、列の折り返しのプロパティを編集できます。



Column Wrapping オプション

このダイアログボックスでは、列の折り返しの X 位置と、列をページに折り返す回数を指定できます。X の位置は、列の折り返しを開始するページの右側からの距離（インチまたはセンチメートル）です。デフォルトでは、列の折り返しが 1 回発生します。Repeat Wrap を-1 に設定すると、折り返しがページ幅内に収まる回数だけ発生します。

列の折り返しを指定して、ページ幅内に収まるよりも何度も出現させることはできません。 たとえば、3 回指定して、結果のレポートがページよりも広い場合は、折り返しが 2 回だけ繰り返されます。

3.5.7.10 目次

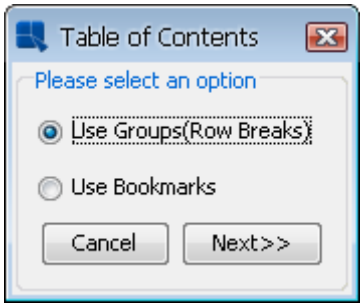
EspressReport には、レポートの目次を生成する機能があります。目次には、定義されたグループをレポートに表示するか、ユーザ定義のブックマークのリストを表示できます。目次は、レポートビューア、Page Viewer、DHTML、および PDF エクスポートに表示されます。

3.5.7.10.1 目次の追加

レポートに目次を追加するには、**Insert** メニューから **Insert Table of Contents** を選択します。カーソ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ルが十字に変わります。レポートヘッダーセクションまたはテーブルヘッダーセクションの目次を配置する場合にクリックします。他のセクションに追加することはできません。目次にグループやブックマークを使用するかどうかを尋ねるダイアログが表示されます。



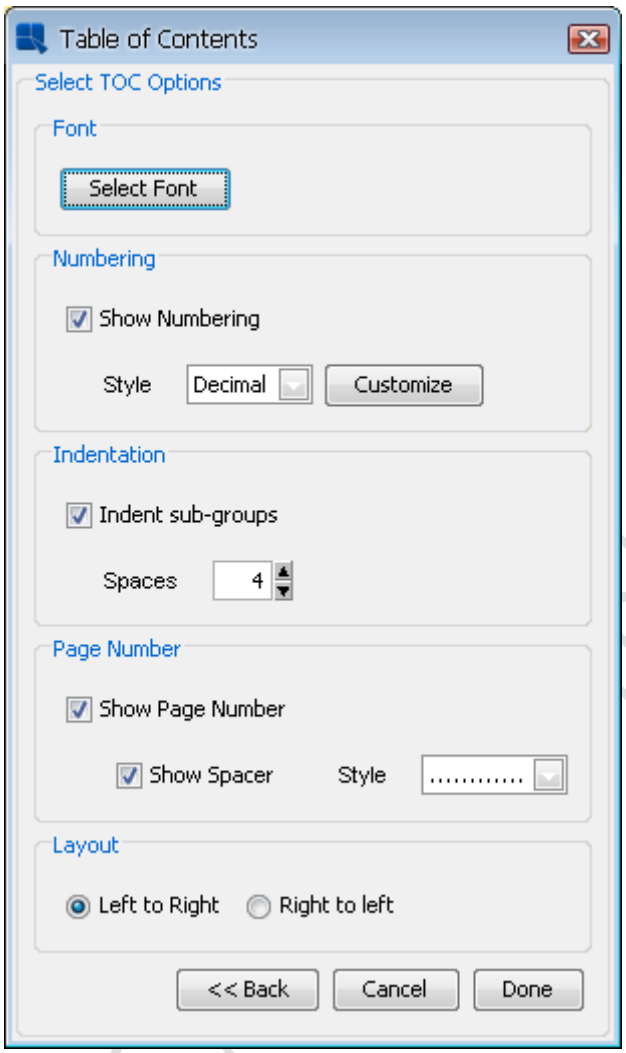
TOC Type ダイアログ

グループを使用することを選択すると、目次はレポート内の各グループ（およびサブグループ）のエントリを生成します。このオプションを使用するには、レポートにグループ化されたデータ（サマリブレーク、クロスタブ、マスタ詳細レポート）と詳細レコードが含まれている必要があります。1 つの行区切りを持つクロス集計レポートまたは列集計を使用した集計レポートをグループ化できますが、詳細レコードがないため、このプレゼンテーションの目次を作成することはできません。ブックマークの使用を選択すると、レポートに定義されているブックマークごとに 1 レベルの目次が生成されます。ブックマークはセルスクリプトを使用して定義されます。ブックマークの設定の詳細については、アクションの書式設定を参照してください。

次に表示されるダイアログでは、目次のオプションを指定できます。

Copyright(C) 2018 Climb.Inc.

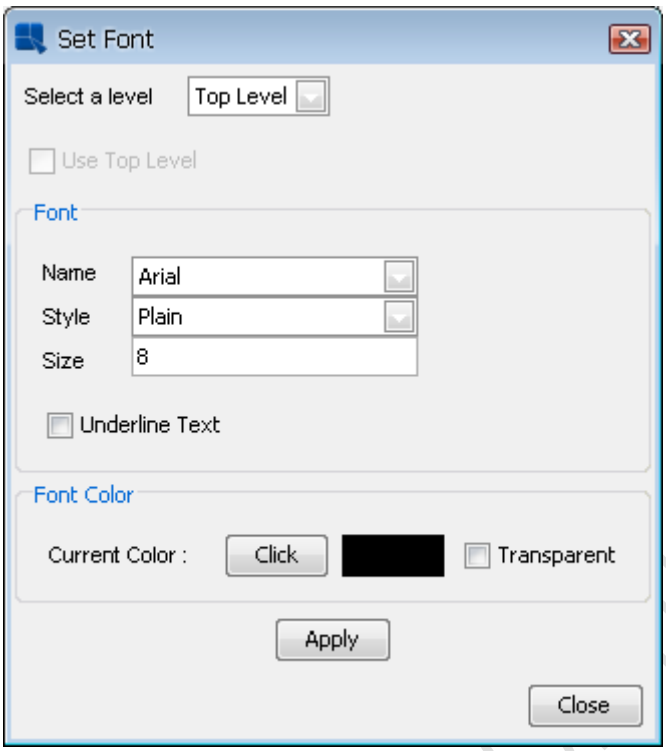
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



TOC Options ダイアログ

このダイアログの最初のオプションでは、目次の各レベルのフォントを設定できます。**Select Font** ボタンをクリックすると、フォントオプションを指定できる新しいダイアログが開きます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



TOC Font ダイアログ

各レベルについて、フォントの面、スタイル、サイズ、および色を設定できます。レベルは、レポートのネストされたグループ化のレベルに対応することに注意してください。ブックマークを含むレポートを作成するように選択した場合は、最上位レベルのみが使用可能になります。サブレベルは、親レベルからフォントを継承できます。特定のレベルのフォントを変更したら、**Apply** ボタンをクリックして変更を保存します。

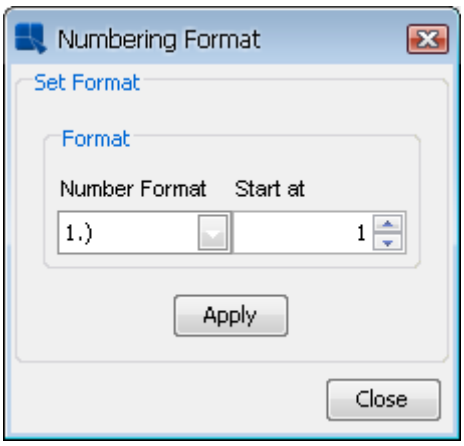
次のオプションでは、目次の番号と使用するスタイルを表示するかどうかを選択できます。**Customize** ボタンをクリックすると、各スタイルをカスタマイズできます。以下のスタイルがサポートされています。

Decimal:

これは、一番外側のグループの整数と各サブグループの小数の目次エントリの番号付き構成を使用します。このオプションの **Customize** ボタンをクリックすると、次のダイアログが表示されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

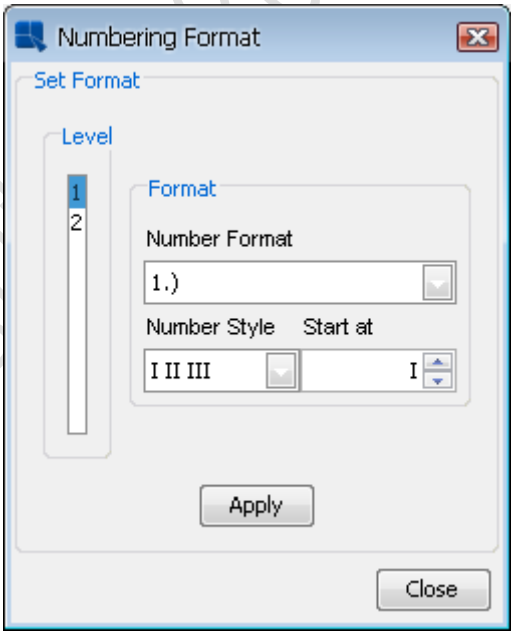


TOC Decimal Format のカスタマイズオプション

最初のドロップダウンリストでは、番号のフォーマット方法を選択できます。2 番目のオプションでは、開始番号を選択できます。選択した後、**Apply** ボタンをクリックして変更を適用します。

Outline:

このオプションでは、目次の各レベルに数字、文字、またはローマ数字を選択できるアウトライン形式が使用されます。このオプションの **Customize** ボタンをクリックすると、次のダイアログが表示されます。



TOC Outline Format のカスタマイズオプション

左側のリストでは、書式設定を適用するレベルを選択できます。最初のドロップダウンでは、数字の書式を選択することができます（レベルに数字またはローマ数字を選択した場合に適用されます）。2 番目のドロップダウンでは、数字、大文字、小文字、大文字のローマ数字、または小文字のローマ数字のいずれかの形式を選択できます。最後のオプションでは、選択したスタイルの開始点を選択できます。選択した後、**Apply** ボタンをクリックして変更を適用します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Bulleted：

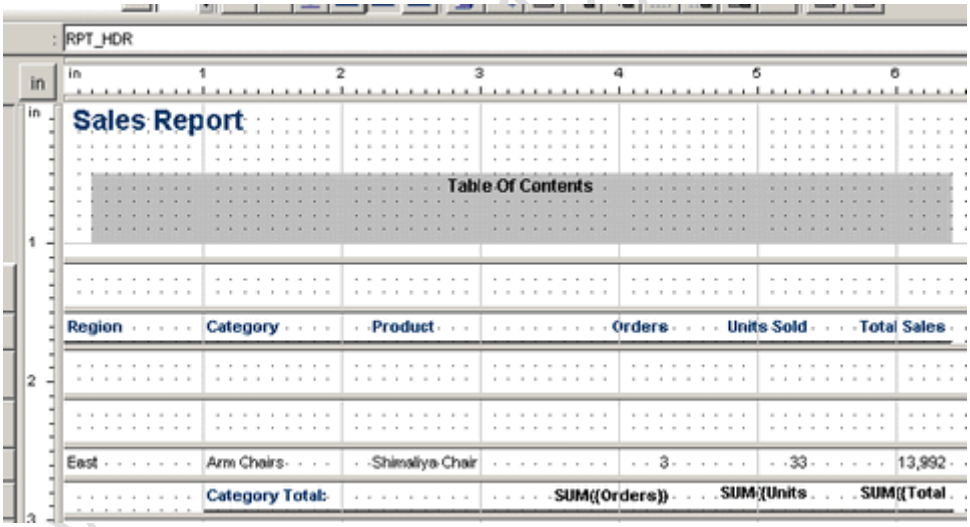
このオプションは、各エントリに箇条書きを使用します。これで、この形式のカスタマイズオプションが追加されました。

次のオプションでは、サブグループのインデントを指定できます。インデントをオン/オフにし、使用するスペースの数を指定することができます。

次のオプションでは、目次にページ番号を表示するかどうか、およびページ番号にスペーサーを描画するかどうかを指定できます。スペーサーの点線または実線を描くことができます。

ダイアログの最後のオプションでは、目次のレイアウトを右から左、または左から右に設定できます。目次のオプションの設定が完了したら、**Done** ボタンをクリックすると、レポートに追加されます。

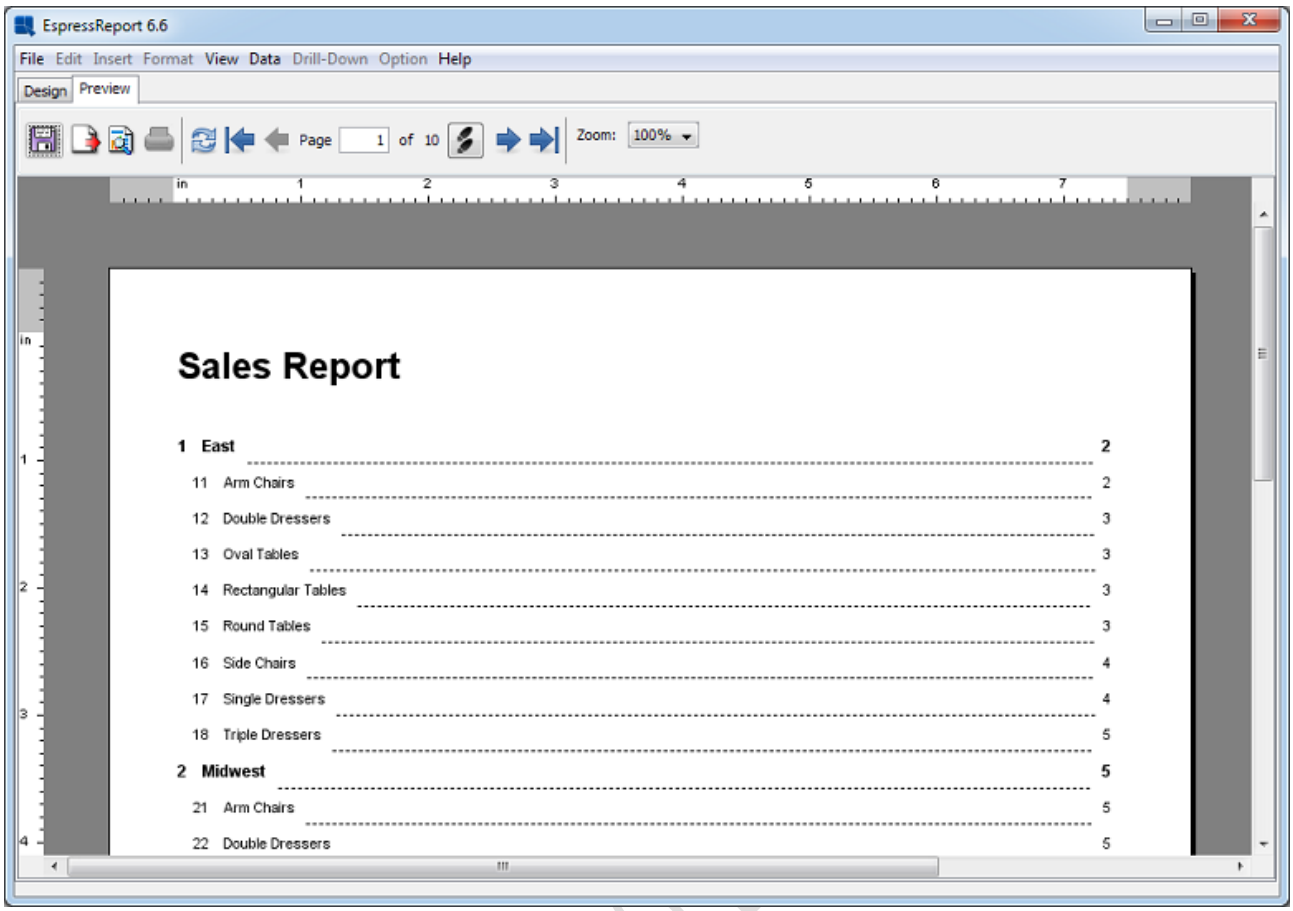
レポートでは、目次は小さな灰色の四角形で表示されます。デフォルトでは、コンテンツに合わせてサイズを変更するオプションがオンになっていますが、目次の幅はセクション内で割り当てられたサイズに基づいています。一般的には、これを少なくともレポートの幅と同じにしたいと考えています。



デザインウィンドウの目次

レポートをプレビューすると、完全な目次が表示されます。目次は、DHTML および PDF エクスポートでのみサポートされています。レポートを他の形式にエクスポートすると表示されません。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



プレビューウィンドウの目次

3.5.7.11 レポート要素の移動およびリサイズ

レポート要素を移動してサイズを変更する方法はいくつかあります。ルーラーやマウスを使用することも、要素境界を手動で入力することもできます。

Rulers:

デザインウィンドウの左上隅にある定規を使用して要素のサイズを変更したり移動したりすることができます。レポート要素をマウスでクリックして選択すると、各ルーラーに影の付いた領域が表示され、要素の水平および垂直境界がマークされます。

デザインウィンドウの上部にあるルーラーで、左のマーカーをクリックしてドラッグすると、要素が水平方向に移動します。右のマーカーをクリックしてドラッグすると、要素が水平に伸びます。

デザインウィンドウの左側のルーラーで、上のマーカーをクリックしてドラッグすると、要素が垂直方向に移動します。下のマーカーをクリックしてドラッグすると、要素が垂直に伸びます。

Mouse:

マウスを使用して要素のサイズを変更したり移動したりすることができます。要素を移動するには、単にクリックしてドラッグします。要素のサイズを変更するには、オブジェクトが選択

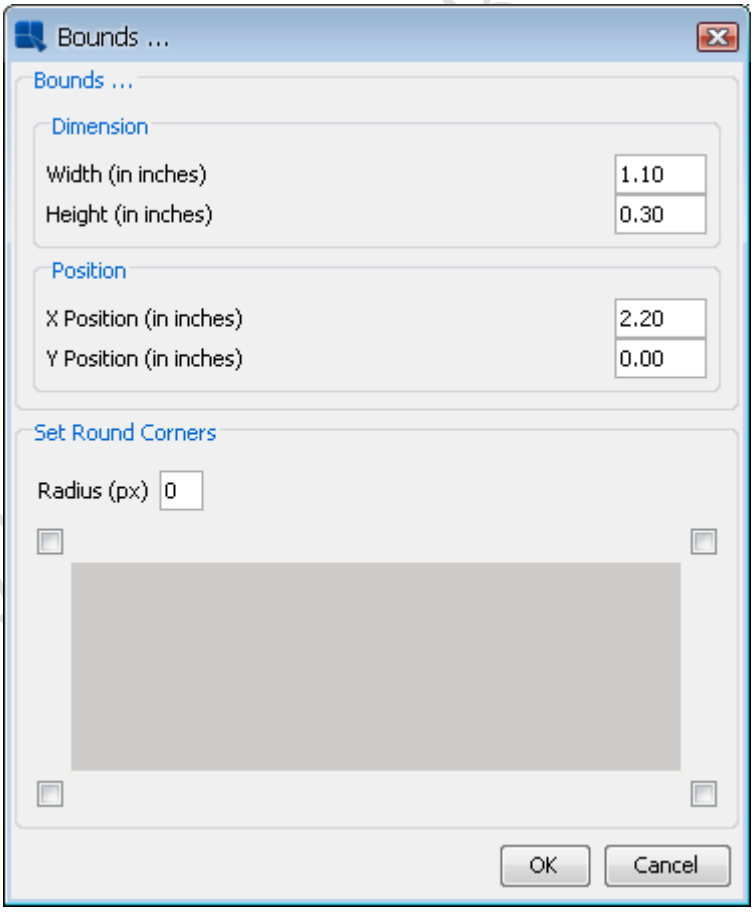
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

されているときにオブジェクトの端に表示されているサイズ調整ハンドルをクリックしてドラッグします。セル内を右クリックしてドラッグすると、セルのサイズを変更することもできます。

Manually Set Bounds:

ツールバーの **Bounds** アイコンをクリックするか、**Format** メニューから **Bounds** を選択して、任意のレポート要素の境界を手動で設定できます（**Format** メニューから開くことができる **Edit Attributes** ダイアログ、オブジェクトをクリックし、ポップアップメニューからオブジェクトを選択します)。新しい要素の境界を入力するダイアログボックスが表示されます。測定値は、ルーラーが会うデザイナーウィンドウの左上隅にあるトグルボタンで選択された単位に応じて、インチまたはセンチメートルで表示されます。X と Y の座標は、特定のレポートセクションごとです。要素の X および Y 位置をマークするポイントは、要素の左上隅にあります。

同じダイアログから、選択した要素の角を設定することもできます。ラウンドコーナーの詳細については、フォント、色、および罫線のオプションを参照してください。



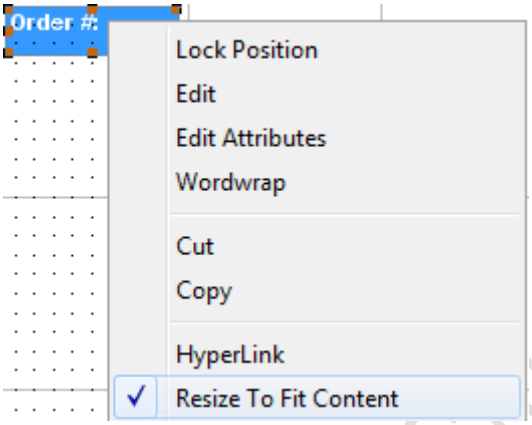
Set Bounds ダイアログ

Resizing To Fit Contents:

多くの場合、さまざまな長さのデータを含む列フィールドがあります。この場合、データが切

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

り捨てられるか、フィールドが不足しているときにセル内に大量の空白が存在する可能性があります。この問題を軽減するために、要素の内容を含むように高さを動的に変更するように要素を設定できます。これを行うには、まずサイズを変更するセルを選択し、**Format** > **Resize** を選択します。要素はコンテンツに合わせて動的にサイズが変更されます。セルの高さのみがサイズ変更されることに注意してください。幅は同じままです。



Dynamic Resize ダイアログ

Resize Cells Proportionally というセクションオプションを呼び出して、セクション内の他のすべてのセルにサイズ変更されたセルの高さを調整させることもできます。セクションオプションの詳細については、セクションオプションを参照してください。

Group Move/Resize:

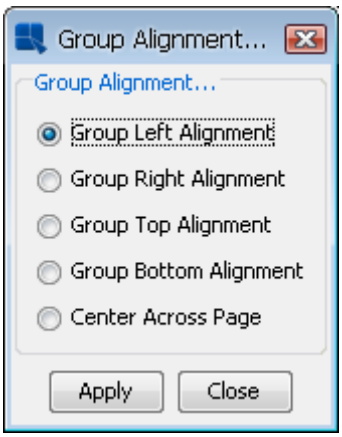
要素を最初に選択することによって、要素のグループを移動またはサイズ変更することができます。レポート内の複数の要素は、選択ボックスを描画するか、または **CTRL+クリック**を使用して選択することによって選択できます。レポートセクションの空白部分をクリックしてドラッグすると、選択ボックスを描画できます。要素が選択されたら、グループ内の任意の要素をクリック＆ドラッグしてそのグループを移動します。サイズ変更ハンドルの 1 つをクリック＆ドラッグして、グループ内の要素のサイズを調整します。

Group Alignment:

グループの左側、右側、上または下に要素のグループを揃えることができます。要素のグループを選択し、右クリックします。**Group Edit** ダイアログボックスが表示され、グループを調整するオプションが表示されます。**Group Left Alignment**、**Group Right Alignment**、**Group Top Alignment** または **Group Bottom Alignment** を選択し、**Apply** ボタンをクリックします。グループ内のすべての要素は、グループの左端、上端または下端に移動します。また、**Center Across Page** オプションを選択することもできます。これは選択されたセルのグループを取り、それらをページの中央に配置します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Group Options/Alignment Pop-up ダイアログ

Moving Elements Between Sections:

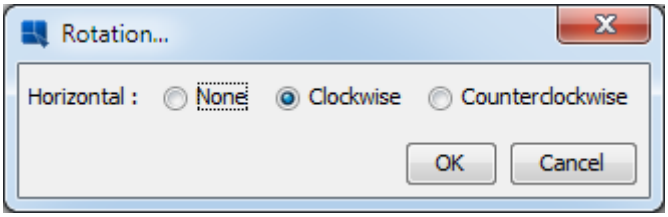
単一の要素または要素のグループは、セクション境界の上にドラッグすることで、レポートセクション間で移動できます。要素を **Table Data** セクションにドラッグすることはできません。

Column Swap:

レポートセクション内の 2 つの要素の位置を入れ替えることができます。これは主に、レポート内の列フィールドの位置を再配置するために使用されます。2 つの要素を交換するには、**Ctrl+クリック**または選択ボックスを使用して要素を選択し、**Format** メニューから **Swap Columns** を選択します。2 つの要素の位置が切り替えられ、スワップに対応するために周囲の要素が移動されます。

Rotate:

ラベル、数式、列ヘッダー、列フィールドなどのテキスト要素は、時計回りまたは反時計回りに 90 度回転できます。要素を回転させるには 2 つの方法があります。要素を回転させる最初の方法は、要素を右クリックしてマウスポインタを回転オプションに移動することです。サブメニューには、**None**、**Clockwise**、**CounterClockwise** の 3 つのオプションが含まれています。要素を回転させるもう 1 つの方法は、要素を選択し、**CounterClockwise** メニューから **Rotation** オプションを選択することです。



Rotation ダイアログ

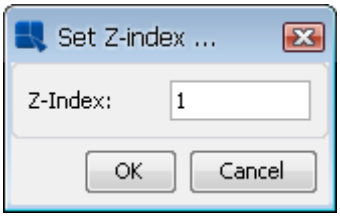
テキスト回転は、DHTML、PDF、XSLX、および XLS エクスポート形式でのみ使用できます。

3.5.7.11.1 要素のオーバーラップのコントロール

EspressReport では、レポートの各要素には、関連付けられた Z インデックスがあります。この Z イン

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

デックス番号は、同じスペース（互いの上にある）に配置されたときの要素の動作を決定します。要素が同じスペースに配置されると、Z インデックスが最も高い要素が上に表示され、Z インデックスの低いすべての要素が連続して下に描画されます。レポート要素の Z インデックスを設定するには、レポート要素を右クリックし、ポップアップメニューから **Set Z-Index** を選択します。これにより、その要素のインデックスの番号を指定できるダイアログが表示されます。



Z Index ダイアログ

HTML や Excel などの表形式のレイアウトを必要とする特定のエクスポート形式では、レポートに重複する要素があると正しく表示されないことがあります。

3.5.7.11.2 要素の位置の固定

Report Designer では、要素を右クリックしてポップアップメニューから **Lock Position** を選択すると、セルの位置をロックできます。この機能を有効にすると、レポート要素をマウス、ルーラー、またはグループの書式設定や整列機能で移動することはできません。ロック位置オプションがオフの場合にのみ移動できます。

3.5.7.12 グリッドにスナップ

既定では、Report Designer は Snap to Grid モードで動作します。これにより、レポート要素は設定された増分だけで移動します。グリッドレイアウトは、デザインウィンドウに描画された小さな点で表されます。デフォルトでは、グリッドのステップサイズは 0.1 インチまたは 0.25cm です（どの測定値が選択されているかによる）。

グリッドスナップオプションを変更するには、**Option** メニューから **Snap To Grid** を選択します。グリッドプロパティを設定できるダイアログが開きます。

Snap To Grid ダイアログ

このダイアログでは、グリッドステップサイズを変更したり、グリッドスナップを無効にしたりすることができます。これを無効にすると、レポート要素はデザインウィンドウの周りをフリーフォームで移

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

動できます。

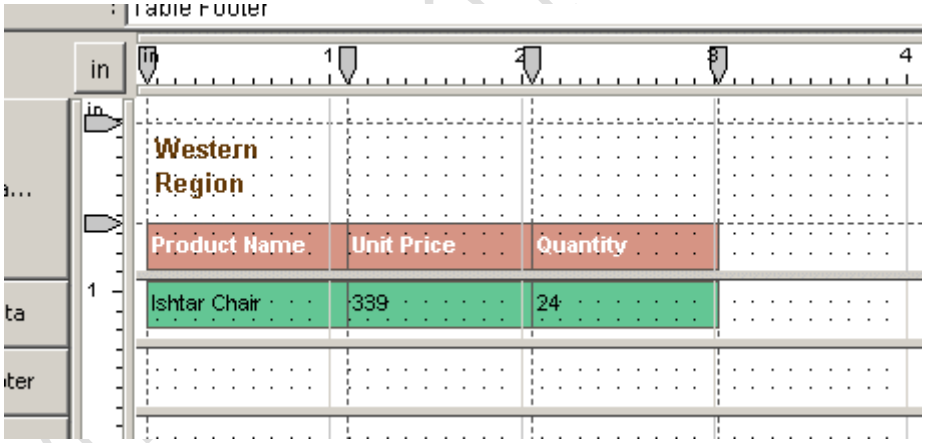
3.5.7.13 シフトモード

Option メニューから **Shift On/Off** を選択すると、Shift モードがオン/オフされます。シフトモードを有効にすると、他のレポート要素の変更に合わせてレポート要素が移動します。したがって、1 つのレポート要素の幅を 1"から 2"に変更すると、現在の要素の次の要素が右に移動します。シフトモードが選択されていない場合、リサイズされた要素が別のレポート要素と重なっても、他のレポート要素は移動しません。

シフトモードは、要素のサイズを変更したり、要素を少量だけ移動したりする場合に最適です。 Shift モードを有効にした状態で明瞭な距離の要素を移動すると、他のレポート要素も同様の距離だけ移動し、レポートに奇妙な外観が表示されます。

3.5.7.14 ガイドライン

要素のグループを配置および移動する別の方法は、ガイドラインを使用することです。ガイドラインを使用すると、デザインウィンドウ内に任意の位置線を配置し、レポート要素をそれらの線にスナップすることができます。ガイドラインを使用すると、正確な動作の実行を心配することなく、レポート要素を正確に整列して配置することができます。



ガイドラインに基づいて配置された要素

3.5.7.14.1 ガイドラインの挿入

Insert メニューから **Insert Guideline** を選択すると、水平および垂直ガイドラインを挿入できます。このオプションを選択すると、サブメニューが表示され、縦または横のガイドラインを選択するかどうかを選択するメッセージが表示されます。必要なオプションを選択したら、デザインパネルでガイドラインを描く場所をクリックします。ガイドラインは、上ルーラーまたは左ルーラーにマーカーが付いた点線で表示されます。ルーラーのマーカーをクリックしてドラッグすることで、ガイドラインを移動できます。

ガイドライン（またはルーラー内のガイドラインマーカー）を右クリックし、ポップアップメニューから **Delete** を選択すると、レポートから任意のガイドラインを削除できます。

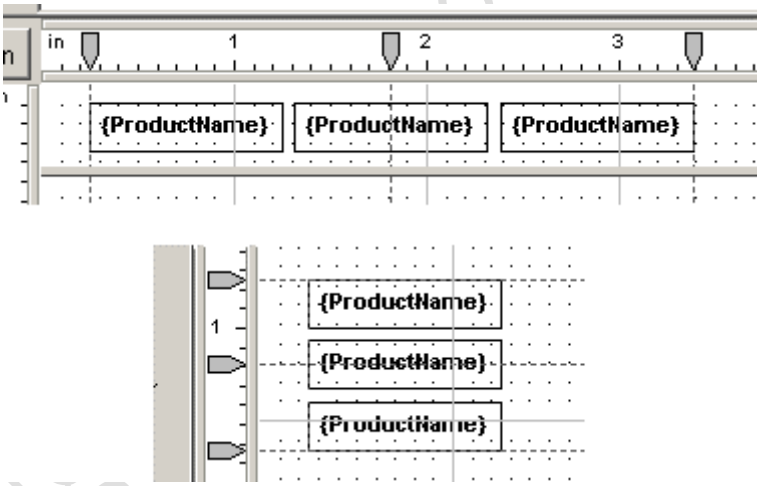
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

レポートのセクション間で水平のガイドラインを移動することはできません。ガイドラインを新しいレポートセクションに移動する場合は、新しいガイドラインを挿入する必要があります。

3.5.7.14.2 ガイドラインによる要素の配置

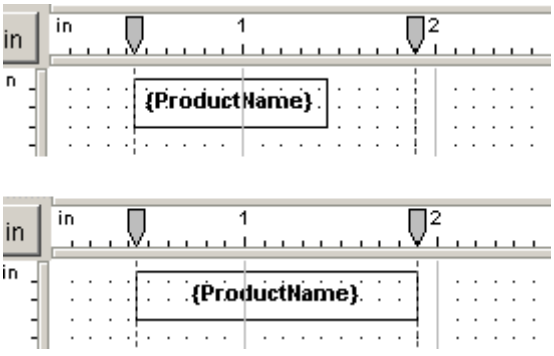
オブジェクトをガイドラインにスナップするには、2 つの方法があります。最初のオプションは、ガイドラインの近くにあるすべての要素をスナップすることです。これを行うには、ガイドライン（またはルーラーのガイドラインマーカー）を右クリックし、Snap Elements in Range 選択します。次に、セルの左端、右端、または中央をスナップするように選択できます。これにより、ガイドラインから最大 0.1 インチ離れたすべてのセルがスナップされます。

もう 1 つの選択肢は、要素をガイドラインに選択的にスナップすることです。これを行うには、まず、**CTRL+クリック**（要素を 1 つしか選択できませんが、**CTRL+クリック**を使用して選択する必要があります）を使用してガイドラインにスナップする要素を選択し、スナップしたいガイドラインを右クリックします。（またはルーラーのガイドラインマーカー）。セルの左端、右端、または中央を、水平および垂直のガイドラインのガイドラインにスナップするように指定できます。ポップアップメニューから必要なオプションを選択すると、セルがガイドラインにスナップされます。



ガイドラインの配置オプション

要素がガイドラインに添付されると、ガイドラインを移動すると要素も同様に移動します。要素を複数のガイドラインにスナップすることもできます。要素を 2 つのガイドラインにスナップすると、要素が伸びてフィットします。



Copyright(C) 2018 Climb.Inc. All Rights Reserved.

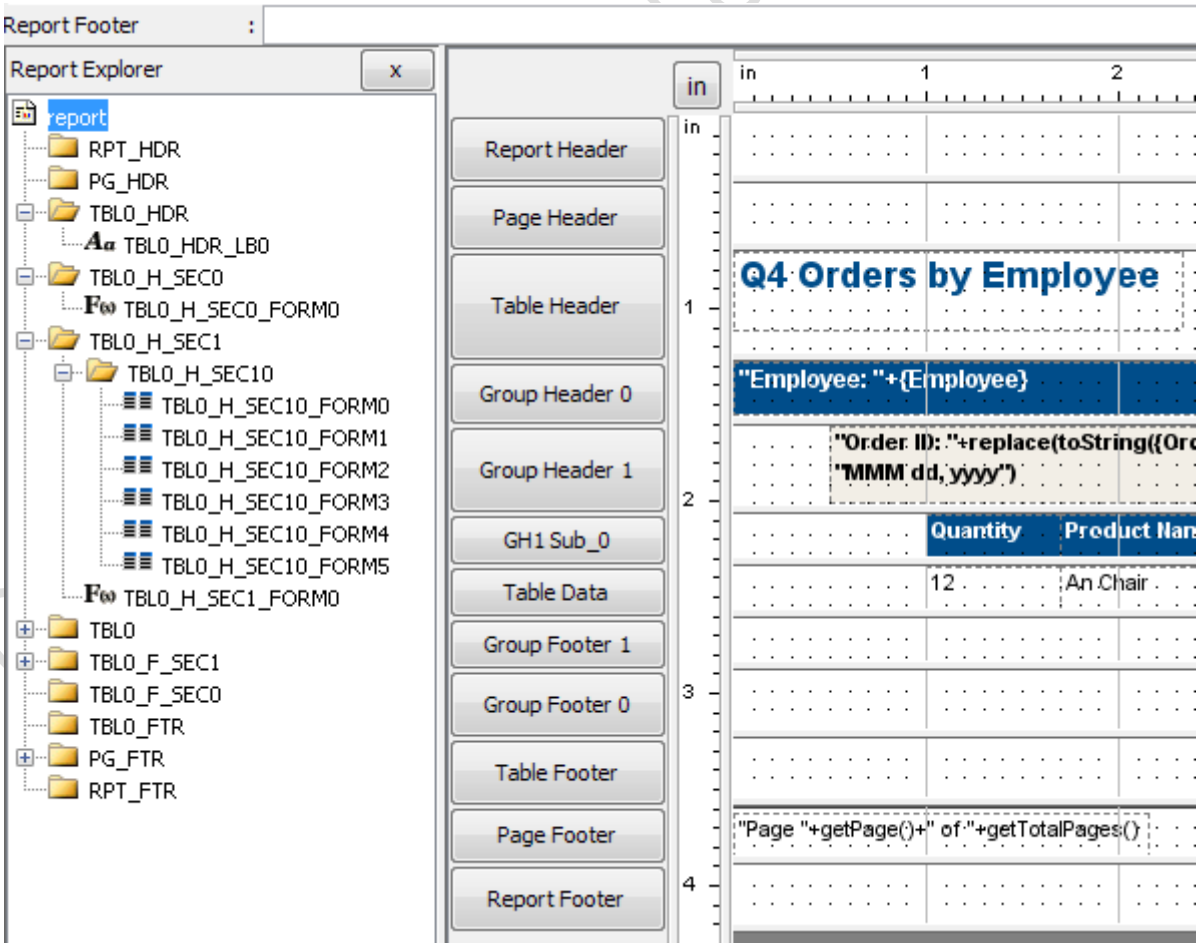
要素が 2 つのガイドラインにスナップ

2 つのガイドラインに要素が添付されている場合は、いずれかのガイドラインを移動すると要素のサイズが変更されます（ストレッチまたは縮小のいずれか）。

要素をガイドラインから解放するには、その要素を右クリックし、ポップアップメニューから *Separate From Guidelines* を選択します。これにより、要素が添付されているガイドラインから要素が解放されます。また、ガイドライン（またはルーラーのガイドラインマーカー）を右クリックし、ポップアップメニューから **Separate All** を選択することによって、ガイドラインに添付されているすべての要素を解放することもできます。

3.5.8 Report Explorer

メインのデザインウィンドウに加えて、EspressReport は、レポート内の要素（レポートエクスプローラ）を選択して編集するための別のインタフェースを提供します。レポートエクスプローラでは、レポートのすべての要素がデザイナの左側にツリー構造で表示されます。オプションメニューからレポートエクスプローラを選択してオン/オフを切り替えることができます。



Report Explorer ウィンドウ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ツリー内の各親ノードは、レポート内のセクションを表します。各セクションの中で、アイコンは要素タイプと要素 ID を示します。要素のカスタム ID を指定した場合は、代わりにその ID が表示されます。カスタム ID の詳細は、列フィールドのデータの使用を参照してください。

エクスプローラ内からレポート要素を編集することもできます。ツリー内の要素を選択すると、その要素も **Design** ウィンドウで選択され、**Design** ウィンドウがスクロールしてその要素を画面に表示します。ツールバー、メニュー、またはポップアップメニューを使用して、要素プロパティのいずれかを設定できます。ポップアップメニューは、ツリー内の任意の要素を右クリックすることによっても有効にすることができます。

3.5.9 グローバル書式設定、グループ書式設定、テンプレート

この時点で、個々のレポート要素のプロパティを編集して書式を設定する方法しか見ていません。ただし、複数の要素を一度に整形したいことがあります。これを行うには、グローバル書式設定、グループ書式設定、およびテンプレートを使用できます。

3.5.9.1 グローバル書式設定

グローバル書式設定機能を使用すると、レポート内のすべての要素に一貫した外観を付けることができます。**Global Format** を設定するには、**Option** メニューからグローバルフォーマットを選択します。これにより、チャート、列、数式、イメージ、ラベル、線、矩形、列ヘッダー、およびタイトルの各レポート要素タイプがリストされた 2 番目のメニューが表示されます。これらのいずれかを選択すると、タブ付きのダイアログボックスが表示され、これらの要素のフォーマットを設定するよう求められます。各タブには、要素属性の書式設定オプションが含まれています。書式設定のオプションは次のとおりです。

Chart:

境界線の色、境界線の太さ、境界線の設定

Column:

データフォーマット、フォントのスタイルとサイズ、フォントの色、背景の色、境界の色、境界の太さ、二重の色、境界の設定、配置、回転、スクリプト。

Formula:

データフォーマット、フォントの色、フォントのスタイルとサイズ、背景色、境界線の色、境界線の太さ、境界の設定、配置、回転、スクリプト。

Image:

境界線の太さ、境界線の色、境界線の設定、スクリプト。

Label:

フォントの色、フォントスタイルとサイズ、背景色、境界線の色、境界の太さ、境界線の設定、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

配置、回転、スクリプト。

Line:

行の色、境界、スクリプトを設定します。

Rectangle:

背景色、境界線の色、境界線の太さ、境界線の設定、スクリプト。

Column Header:

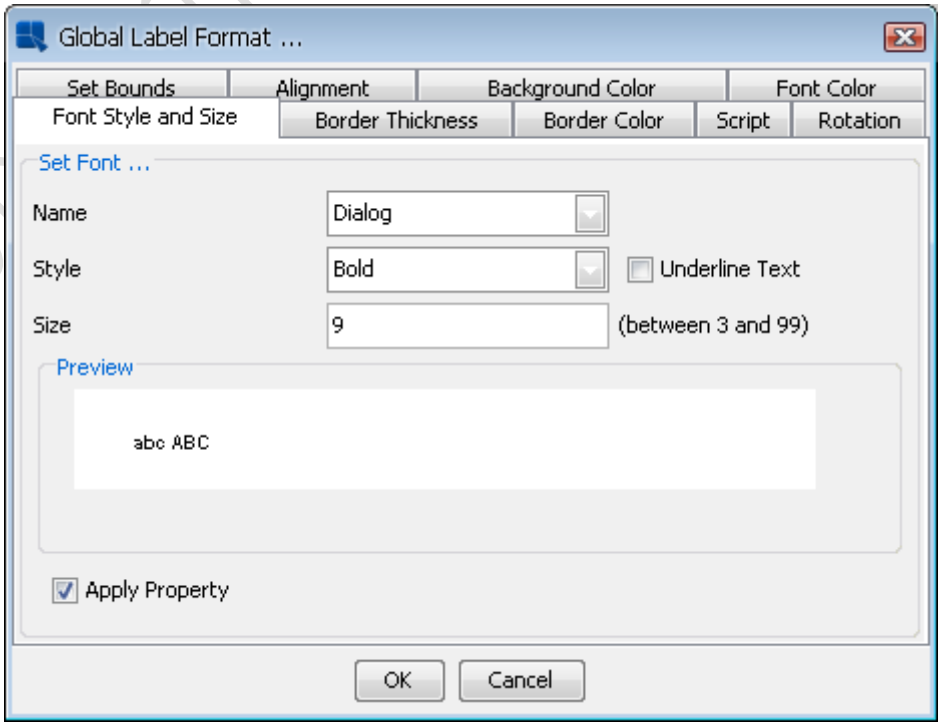
フォントの色、フォントスタイルとサイズ、背景色、境界線の色、境界の太さ、境界線の設定、配置、回転、スクリプト。

Title:

フォントの色、フォントのスタイルとサイズ、背景色、境界線の色、境界線の太さ、境界線の設定、配置。

各タブの下部には、**Apply Property** とマークされたチェックボックスがあります。このチェックボックスをオンにすると、**OK** をクリックするとグローバルフォーマットにプロパティが適用されます。プロパティを変更すると、ボックスが自動的にチェックされます。すべてのプロパティを変更した後、**OK** ボタンをクリックすると、レポート全体でそのタイプのすべての要素のプロパティが変更されます。 また、レポートに追加された要素のデフォルト属性になります。

画像を元のサイズ(デフォルトの属性)で挿入するオプションがあります。これは、画像の Global Format ダイアログの **Set Bounds** タブの **Default Size** というチェックボックスを使用して行うことができます。



Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Global Format ダイアログ

3.5.9.1.1 グローバル書式設定のインポート/エクスポート

インポート/エクスポート機能を使用して、レポート間でグローバルフォーマットを渡すことができます。**Global Format** サブメニューから **Export** を選択すると、グローバルフォーマットをエクスポートできます。これによりファイル名を指定するダイアログボックスが表示されます。グローバルフォーマットは XML ファイルとして保存されます。**Global Format** サブメニューから **Import** オプションを選択すると、グローバルフォーマットの XML ファイルをロードできます。これにより、インポートする XML ファイルを指定するダイアログボックスが表示されます。**OK** ボタンをクリックすると、XML ファイルに保存されている書式が現在のレポートに適用されます。

3.5.9.1.2 グローバル書式設定とクロス集計レポート

クロス集計レポートを処理する場合は、グローバルフォーマットが非常に重要です。レポートの性質上、データ、データソース、またはフィルタリング基準の変更に応じて、列の数が増減できます。クロス集計レポートで新しい列が生成されると、既定の書式が継承されます。クロス集計レポートで新しい列の外観を制御するには、グローバルフォーマットを使用してデフォルトの書式を設定する必要があります。

また、レポートに新しい列が表示される間隔を制御するため、グローバル書式も重要です。新しい列は、グローバルフォーマットの列幅で定義された間隔で表示されます。したがって、デフォルトの幅をインチに設定すると、クロス集計レポートの新しい列が 1 インチ間隔で表示されます。このため、クロス集計レポートのデータ列の幅を一定に保つことをお勧めします。幅を変更すると、新しい列がレポートに追加されたときに予期しない動作が発生する可能性があります。

3.5.9.2 グループ書式設定

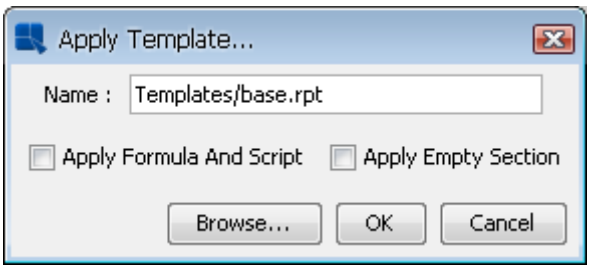
要素のグループに書式を適用するには、まず選択ボックスを描画するか、**Ctrl+クリック**を使用して要素のグループを選択します。また、セルを右クリックし、ポップアップメニューから **Select Column** または **Select Row** オプションを選択して、セルの行または列を選択することもできます。

グループを選択したら、単一の要素と同じようにプロパティの書式を設定します。書式設定は、適用可能な要素にのみ有効です。たとえば、4 つの要素、2 つのラベル、数式、およびグラフを選択し、フォントサイズを変更すると、グラフには何の影響も与えません。

3.5.9.3 テンプレートの適用

レポート（.rpt または.xml）ファイルの機能と要素は、別のレポートに適用できます。これは、テンプレート適用機能を使用して実行されます。現在のレポートにテンプレートを適用するには、**File** メニューの **Apply Template** を選択するか、ツールバーの **Apply Template** ボタンをクリックします。ダイアログボックスが表示され、適用するファイルを選択するよう求められます。テンプレートを適用すると、すべての要素とそれぞれの書式が現在のレポートに適用されます。データソース情報のみが持ち越されません（これには、レポートのテーブルデータセクションの関数とスクリプトが含まれます）。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Apply Template ダイアログ

ダイアログには 2 つの追加オプションがあります。

Apply Formula and Scripts:

このオプションは、式の列とセルスクリプトをテンプレートからレポートに適用します。デフォルトでは適用されません。新しいレポートの列のデータ型がテンプレートのデータ型と異なる場合、テンプレートの数式とスクリプトが正しく機能しないことがあります。

Apply Empty Section:

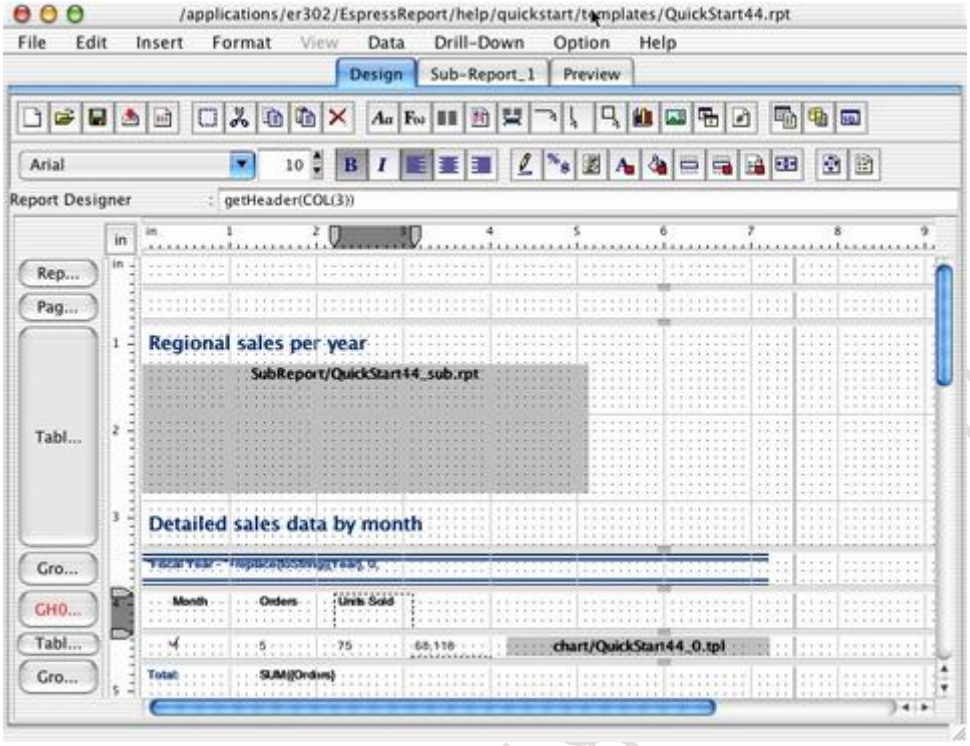
このオプションは、テンプレートからレポートに空白のセクションを適用するかどうかを示します。空白のセクションを適用すると、レポート内のセクションが上書きされ、セクション内の要素が実質的に削除されます。このオプションは、新しいレポートで適用するテンプレートを完全に複製する場合に、一般的に使用されます。ブランクセクションを適用しないと、レポートのセクションは上書きされません。要素を含むテンプレートのセクションのみが新しいレポートに適用されます。この機能は、レポートのグループに渡す特定のデフォルトレイアウト（ヘッダー/フッター）がある場合に使用できます。空のセクションを適用しないように選択することによって、ページのヘッダー/フッターだけに要素が定義され、これらのヘッダー/フッターを他のレポートに適用するテンプレートを定義できます。

ラベルや数式などを入れ替えずに、同じ外観を持つレポートを作成したい場合は、テンプレート全体を適用するのではなく、グローバル形式のインポート/エクスポート機能を使用してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.5.10 Mac OS X での Report Designer



OS X の Report Designer

OS X で Report Designer を実行する場合、ほとんどのコントロールは Windows または Unix/Linux と同じですが、2 つの例外があります。

Multiple Select:

複数のレポート要素またはリストボックスオプションを選択するには、**CTRL+クリック**の代わりに **Command+Click** を使用します。

Right click:

ポップアップメニューを呼び出してレポート要素のサイズを変更するには、右クリックの代わりに **CTRL+クリック**を使用します

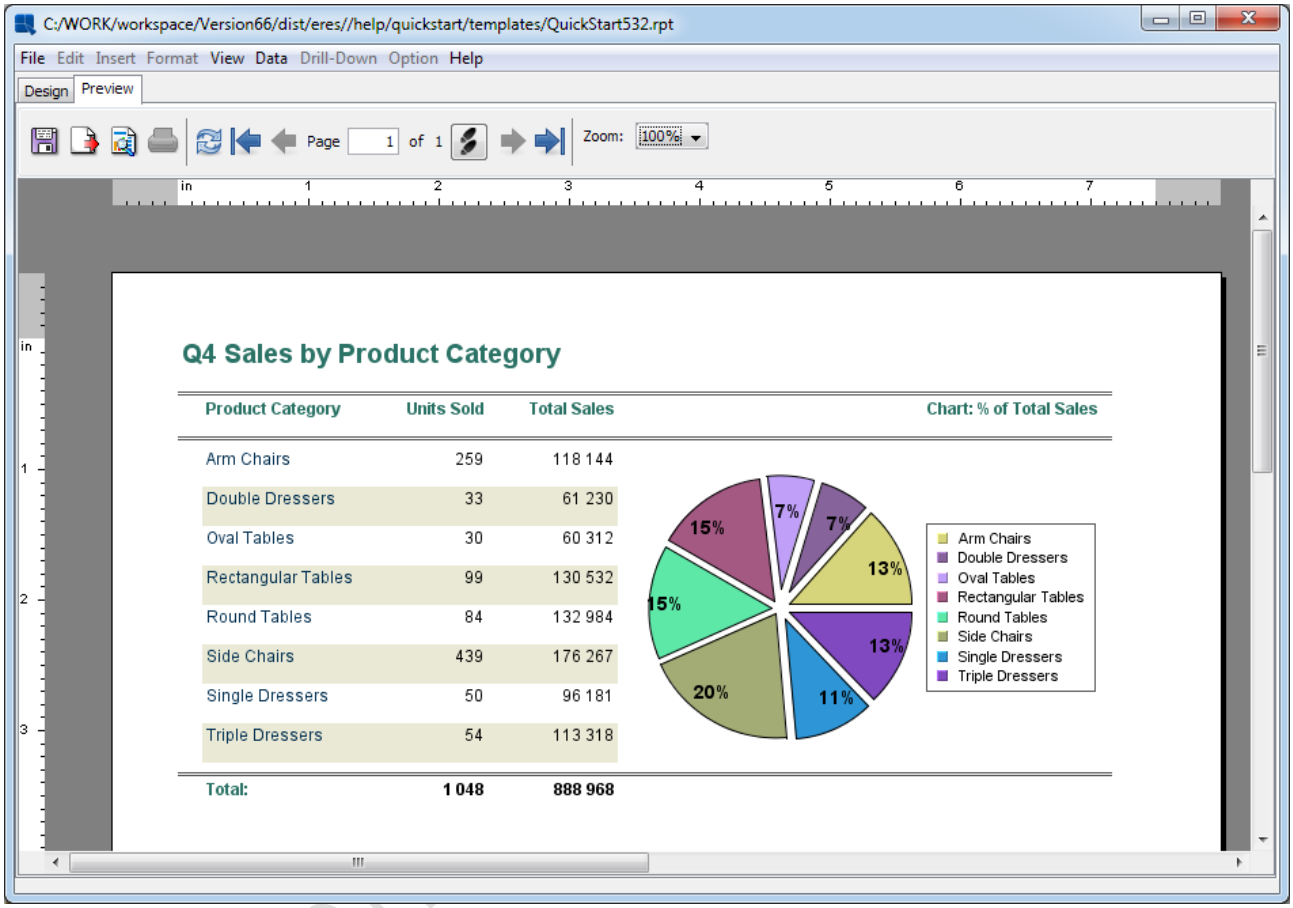
EspressReport は Apple の WebObjects®でもうまく動作します。WebObjects アプリケーションで EspressReport を使用して開発および展開する方法の詳細については、WebObjects を使った作業を参照してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.6 プレビューウィンドウ

レポートオブジェクトを挿入または操作した後、プレビューウィンドウで結果を表示できます。Report Designer の左上隅にあるタブを使用して、デザインウィンドウとプレビューウィンドウを切り替えることができます。プレビューウィンドウでは、印刷またはエクスポートされたレポートの外観を正確に把握できます。



プレビューウィンドウ

プレビューには、ページの大きさがインチ単位またはセンチメートル単位で表示されます（ルーラーが会うデザイナーウィンドウの左上隅にあるトグルボタンで選択される単位に応じて）。ツールバーの右側にあるドロップダウンメニューでは、プレビューのズームを設定できます。

3.6.1 レポートのナビゲート

ツールバーまたは **View** メニューを使用して、レポートをナビゲートすることができます。ツールバーボタンは、以下の機能を実行します。

レポートの最初のページに移動します。

レポートの前のページに移動します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

特定のページ（**Page** テキストボックスに入力したもの）に移動します。

レポートの次のページに移動します。

レポートの最後のページに移動します。

これらのナビゲーション機能は、**View** メニューから **First Page**、**Previous Page**、**Next Page**、**Last Page**、または **Go To Page** を選択することによっても実行できます。

3.6.2 その他のプレビューウィンドウオプション

プレビューウィンドウツールバーのその他のオプションは、次の機能を実行します。

ファイルを保存します。

ファイルをエクスポートします。

Page Viewer ウィンドウにレポート全体を読み込みます。詳細は、Page Viewer の使用を参照してください。

レポートを印刷します。（このオプションは、Report Designer がアプレットとして実行されている場合は使用できません）

データを更新します。

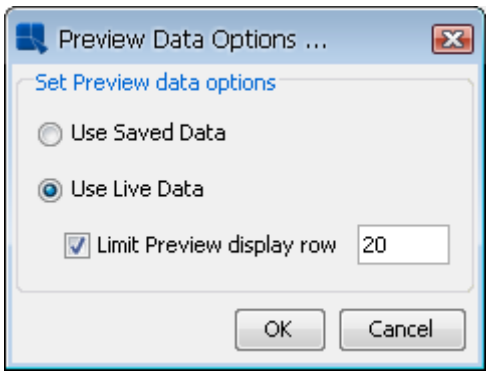
また、**File** メニューと **Data** メニューはアクティブのままで、ファイルとデータを操作できます。

3.6.2.1 データオプションのプレビュー

最初にレポートをプレビューすると、データオプションダイアログが表示されます。このダイアログでは、プレビューウィンドウにレポートデータを表示する方法に関するいくつかのオプションを設定できます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Preview Data Options ダイアログ

最初のオプションでは、レポートのプレビュー時にライブデータを使用するかどうかを選択できます。ライブデータを選択すると、レポートはデータソースに接続され、レポートがプレビューされるたびにデータが取得されます。データが取得できない場合（データベース接続が切断された場合など）、プレビューにエラーが表示されます。その後、レポートをリロードするか（接続が再確立されたとき）、**Preview Data Options** に戻り、**Use Saved Data** を選択する必要があります。保存されたデータを使用するように選択すると、レポートをプレビューするときのデータは何でも表示されます。これは、テンプレートに格納されているバックアップデータの 2 つのレコード、レポートの最初の作成時に取得される最初の 20 個のレコード、または前のプレビューからの完全なデータセットのいずれかです（オプションがライブデータを使用するように設定されている場合）。

ライブデータを使用してレポートをプレビューするように選択すると、データソースから取得する必要があるレコードの数を制限できる 2 番目のオプションを設定できます。レポートで大規模なデータセットが使用されている場合、このオプションを使用すると、処理時間を制限してクライアントメモリを保持しながら、データとともにレポートを表示できます。

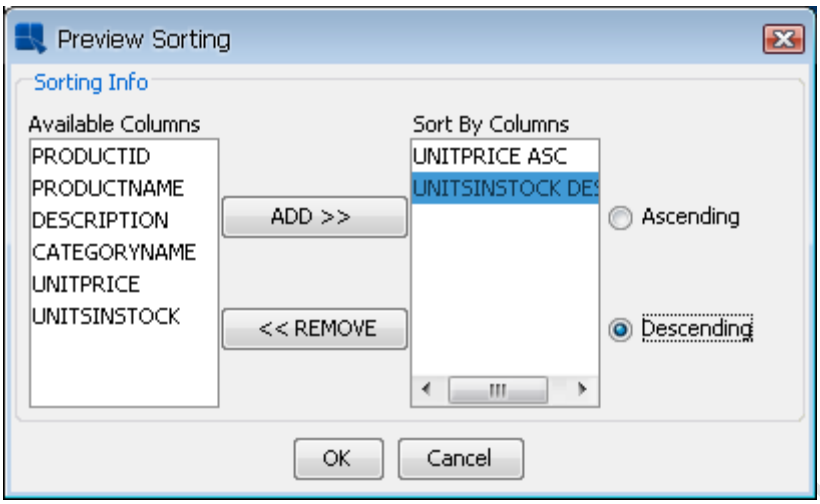
レポートをプレビューするときに初めてこのオプションを設定すると、レポートをプレビューするときに再度確認する必要はありません。これらの設定を変更する場合は、**Data** メニューから **Set Preview Data Options** を選択します（このオプションは **Design** タブでのみ有効です）。ダイアログが再度表示され、プレビューの設定を変更できます。

3.6.2.2 データの並び替え

プレビューウィンドウ内のデータは、レポートの任意の列に基づいて昇順または降順にソートできます。これを行うには、**View** メニューから **Sort by (ascend)** または **Sort by (descend)** を選択します。各オプションは、すべてのレポート列フィールドを含むセカンダリメニューを表示します。列フィールドを選択すると、レポートはその列ごとに昇順または降順のいずれかでソートされます。この機能は、Report Viewer アプレットでも使用できます。

複数の列で並べ替えるには、**View** メニューから **Sort By...** を選択します。これにより、並べ替える列と方向を選択できるダイアログが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Preview Sorting ダイアログ

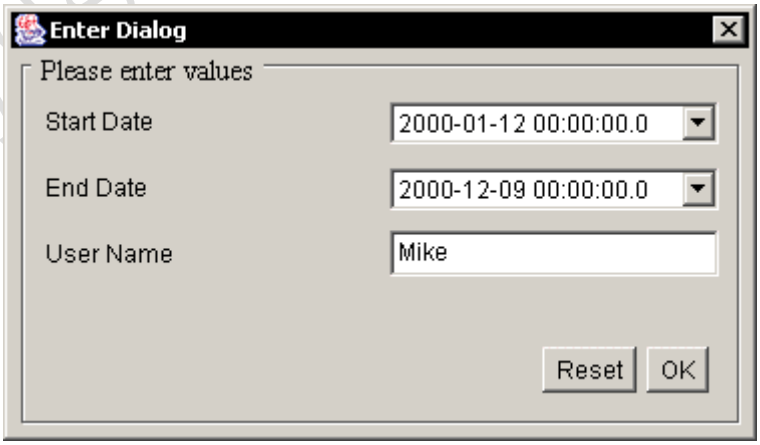
列で並べ替えるには、左側のダイアログで列を選択し、**Add** ボタンをクリックします。右側の列を選択し、**Ascending** または **Descending** のいずれかのボタンをクリックすると、ソートの方向を設定できます。

実行された並べ替えは、テンプレートと共に保存されません。テンプレートを再度開く（実行する）と、データは元の順序を使用します。ただし、並べ替え後にレポートをエクスポートすると、書き出したファイルに並べ替えられたデータが反映されます。

大量のデータを含むレポートでこの機能を使用することは推奨されません。大量のデータが使用され、パフォーマンスが低下する可能性があります。

3.6.2.3 パラメータプロンプト

レポートにパラメータ化されたクエリが使用されている場合、またはレポートに式パラメータが定義されている場合は、レポートをプレビューするときにパラメータ値を選択するよう求められます。

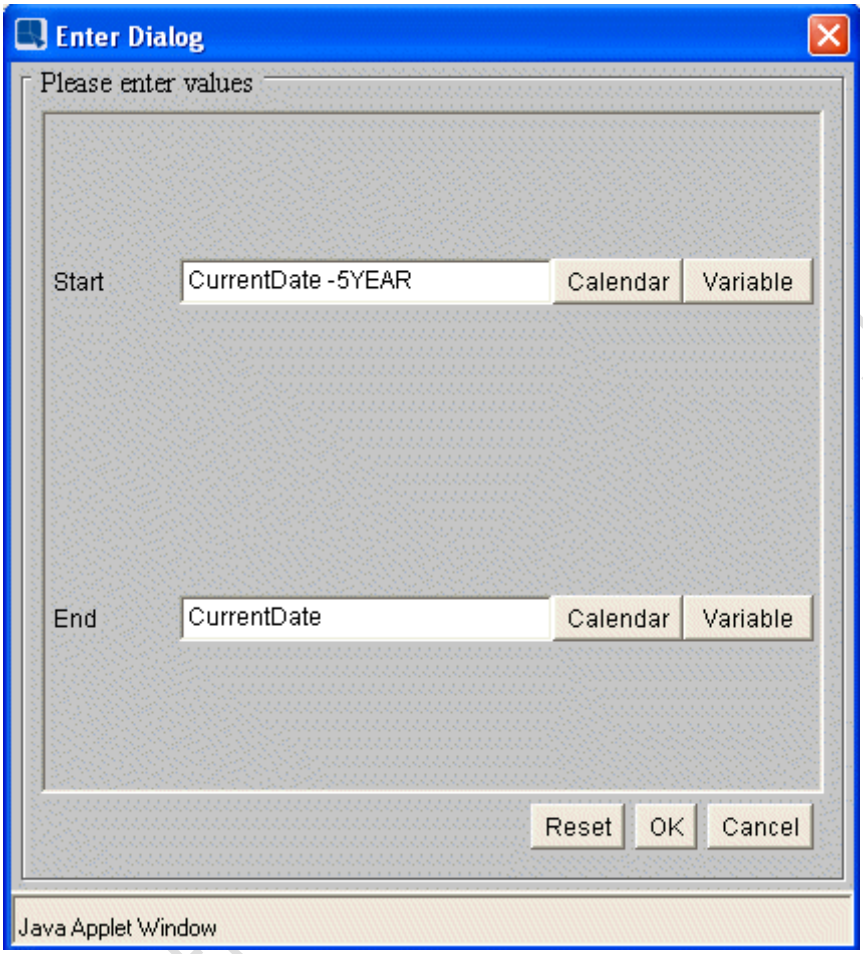


Parameter Prompt ダイアログ

入力するか、パラメータのマッピング方法に応じてパラメータ値を選択するように求められます。**OK** ボタンをクリックすると、指定した値を使用してレポートが生成されます。**Data** メニューの **Preview**

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Parameter Prompt オプションを切り替えることで、パラメータの入力を無効にすることができます。日付変数がデータベース列にマップされていない場合は、次のようにパラメータプロンプトが表示され、カレンダーの日付または日付変数を入力するオプションが表示されます。



Data Parameter Prompt ダイアログ

クエリパラメータの詳細は、パラメータ化されたクエリを参照してください。数式パラメータの詳細については、式パラメータを参照してください。

3.6.2.4 Page Viewer の使用

一般的に、レポートをプレビューすると、データを含むレポート全体がメモリに保持されます。これにより、生成されたレポートをすばやくプレビューしてナビゲートできます。ただし、大規模なレポートの場合、レポート全体をプレビューすると、レポート全体がメモリにロードされるため、システムのパフォーマンスに大きな影響を与える可能性があります。この問題を回避するには、データオプションのプレビューで説明されているように、プレビュー中にフェッチされるレコードの数を制限することができます。

一方、アドホックレポートを作成するときに、レポート全体をメモリに読み込まずにプレビューすることができます。これを行うために、EspressReport では、Page Viewer でレポートを読み込むことができます。これを行うには、**View** メニューから **Launch Page Viewer** を選択するか、ツールバーの

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Launch Page Viewer アイコン をクリックします。これにより、レポートを含む Page Viewer ウィンドウが開きます。

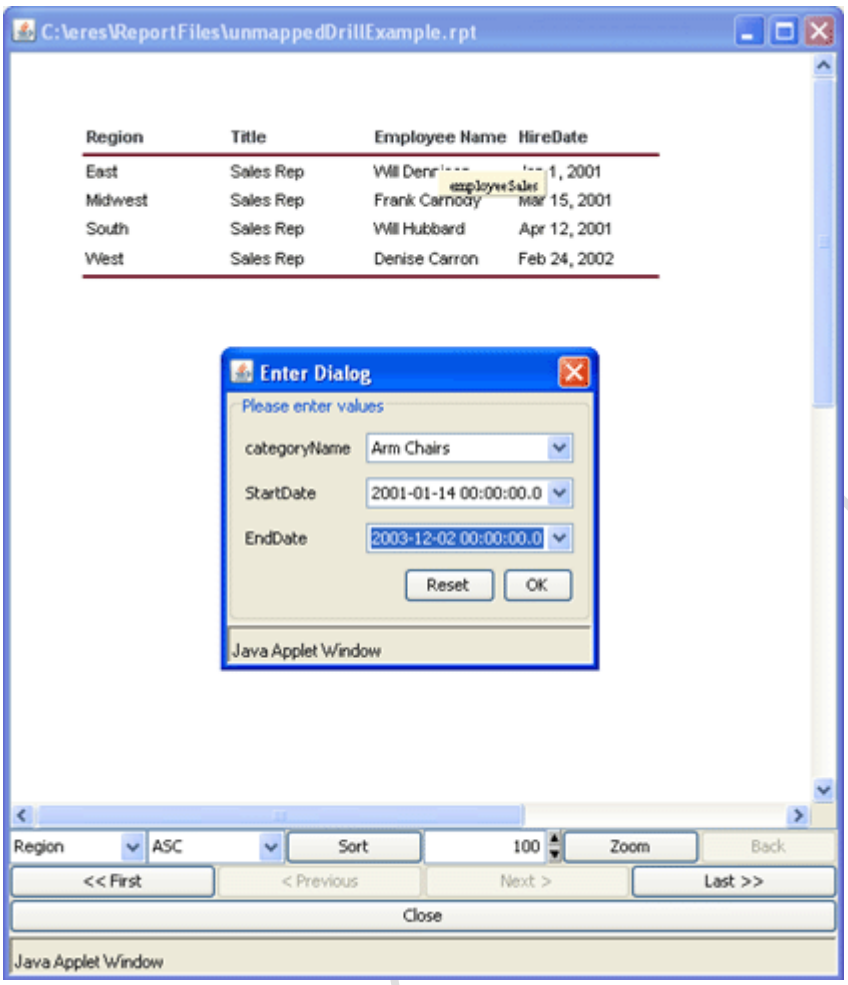
Page Viewer ウィンドウに表示されるレポート

Page Viewer ウィンドウにレポート全体が表示されます。ウィンドウの下部にあるツールバーのボタンをクリックするか、右クリックしてポップアップメニューからページを変更することで、さまざまなページをナビゲートできます。Page Viewer は静的な形式なので、ウィンドウはレポートに加えられたその後の変更を自動的に反映しません。ウィンドウを閉じて、Page Viewer を再起動する必要があります。の詳細については、Page Viewer の概要を参照してください。

3.6.2.4.1

マップされていないドリルダウンパラメーターを含むレポートを表示し、メインレポートのリンクをクリックすると、Parameter Selection ダイアログがポップアップし、マップされていないパラメータを入力できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

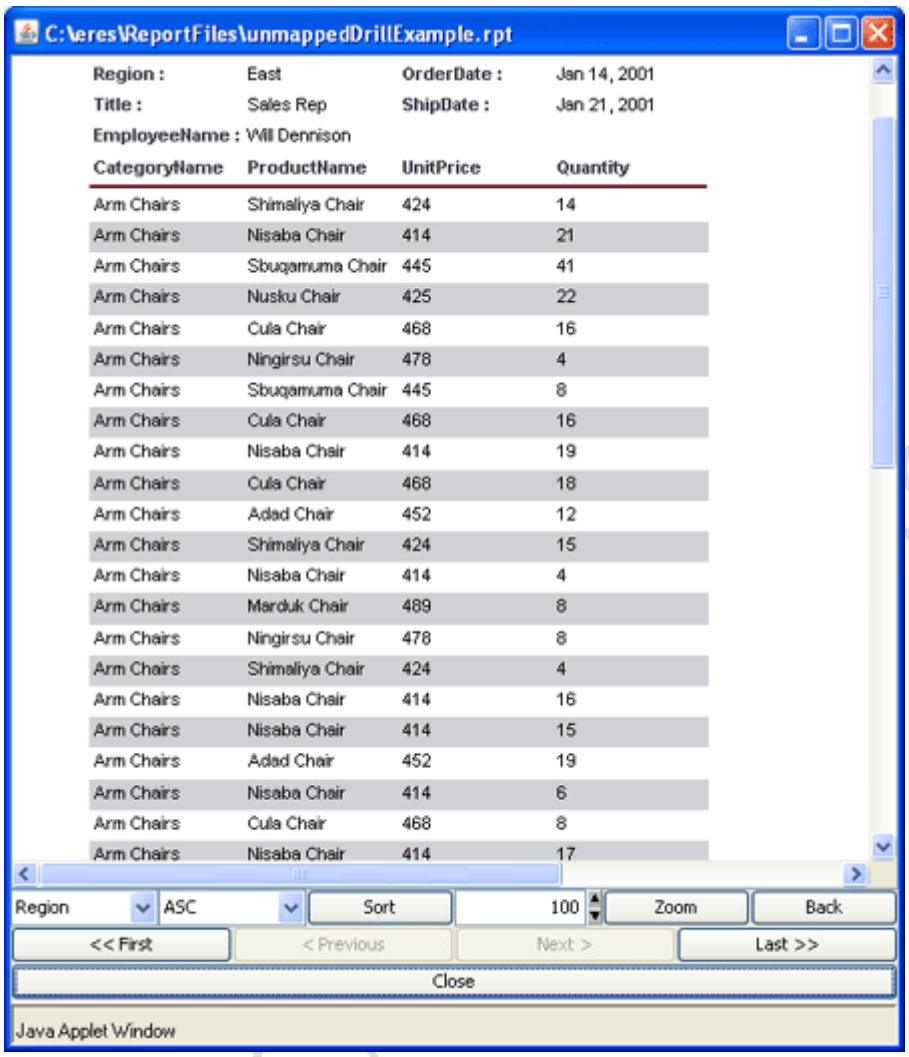


Parameter Selection ダイアログ

値を入力したら、**OK** ボタンをクリックすると、ドリルレポートが表示されます。前のレベルに戻るには、**Page Viewer** ツールバーの **Back** ボタンをクリックします。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

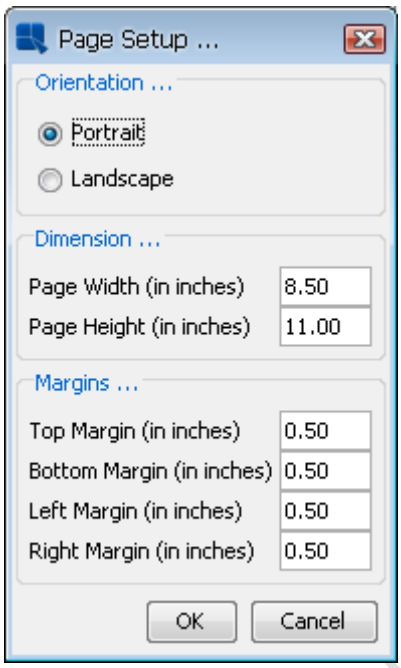


ドリルダウンレベル

3.6.3 ページプロパティの設定

プレビューウィンドウに表示される出力は、デザインウィンドウで設定されたページの寸法と余白に基づいています。ページの大きさは、ルーラーによってマークされ、影の付いた領域はページの余白を示します。これらのプロパティを調整するには、**Option** メニューから **Page Setup** を選択します。これにより、ページの向き（横または縦）、高さ、幅、および余白を設定するように求めるダイアログボックスが表示されます。現在選択されているメトリックシステムに応じて、測定単位はインチまたはセンチメートルになります。方向を変更すると、デフォルトの印刷プロパティも変更されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Page Properties ダイアログ

Copyright(C) 2018 Climb.Inc.

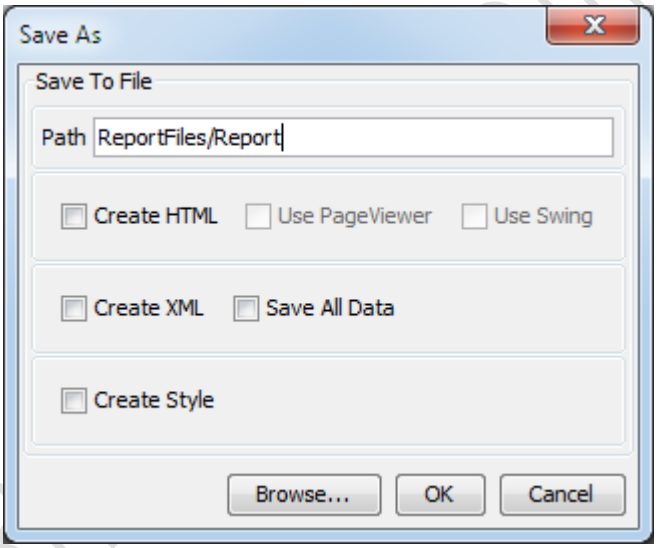
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.7 レポートの保存とエクスポート

レポートの編集が完了したら、それをテンプレートとして保存するか、DHTML、PDF、CSV、Excel（XLS）、Excel 2007（XLSX）、テキスト、XML、またはリッチテキスト形式にエクスポートできます。

3.7.1 レポートの保存

File メニューの **Save** または **Save As** を選択するか、ツールバーの **Save** ボタン をクリックすると、現在のレポートを保存できます。すべてのレポートは、テンプレートとして.pak（バイナリ）または XML 形式で保存されます。保存オプションを選択すると、まだ保存されていない新しいレポートを作成していない限り、既存のファイルが上書きされます。**Save As** オプションを選択すると、ファイル名の作成を促すダイアログボックスが表示されます。**Save As** ダイアログボックスにもいくつかのチェックボックスがあります。



Save As ダイアログ

Create HTML:

このオプションは、レポートビューアまたは Page Viewer アプレットが埋め込まれた HTML ページを作成します。ファイルは.pak ファイルと同じ名前になり、HTML ディレクトリに配置されます。デフォルトでは、Report Viewer アプレットが使用されます。レポートビューアの詳細については、Report Viewer の概要を参照してください。

Use Page Viewer:

このオプションを選択すると、レポートビューアの代わりに Page Viewer アプレットを使用してアプレットページを生成することを示します。Page Viewer の詳細については、Page Viewer の概要を参照してください。

Use Swing:

このチェックボックスは、アプレットページで選択されているビューアに応じて、レポートビ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ューアまたは Page Viewer のスイングバージョンを使用するかどうかを指定します。

Create XML:

このチェックボックスは、レポートを XML 形式で保存するかどうかを指定します。デフォルトでは、レポートはバイナリ（.pak）形式で保存されます。このチェックボックスをオンにすると、EspressReport はテンプレートを XML 形式で生成します。XML ファイルはデザイナによって保存され、再度開くことができます。他のレポートのテンプレートとしても適用できます。

Create Style:

このチェックボックスは、レポートをカスタムスタイルとして保存するかどうかを指定します。この形式で保存されたレポートは、新しいレポートにデフォルトの外観として適用できます。ただし、レポートとして使用することはできません。この章の詳細については、スタイルのカスタマイズを参照してください。

Save All Data:

このオプションは、テンプレート内のすべてのレポートデータを保存します。デフォルトでは、2 つのレコードのみが保存されます。この機能は、データソースを変更するとき、またはデータソースなしでレポートを編集するときにクロスタブの構造を保持する場合に便利です。また、ユーザは、データソースが利用できないときに完全なレポートを共有することもできます。レポートは、常に Report Designer のバックアップデータで開かれます。ユーザは、レポートをプレビューする際に **Live Data** オプションを選択して、最新のデータを取得することができます。

3.7.1.1 保存場所の変更

デフォルトでは、すべてのレポートは/**templates**/ディレクトリから保存/開かれます。デフォルトのディレクトリを変更するには、**reportdesigner.bat(.sh)**ファイルまたは Report Designer の起動に使用されるアプレットページの HTML ソースを変更します。

デフォルトの保存場所を変更するには、**reportdesigner.bat(.sh)**ファイルに次の引数を追加します。**-templatesDirectory:<Path>**ここで、**<Path>**はインストールのルートディレクトリを基準とするディレクトリです。

例

-templatesDirectory:templates/mytemplates

デフォルトディレクトリを**<EspressReport InstallDir>/templates/mytemplates** に設定します。

アプレットを使用して Report Designer を実行している場合、アプレットの HTML ページに次のパラメータを追加する必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

<PARAM NAME = templates_directory VALUE = "templates/mytemplates">

3.7.1.2 XML エンコーディング

EspressReport のデータレジストリファイルとレポートテンプレートの両方を XML 形式で保存できます。デフォルトでは、両方のタイプのファイルが西ヨーロッパのエンコーディングを使用します。データレジストリ（クエリを含む）とレポートテンプレートを XML で正しく保存するには、XML エンコーディングを設定して別の文字セットを使用する必要があります。この機能の使用の詳細については、XML エンコーディングを参照してください。

3.7.2 レポートのエクスポート

現在のレポートは、Report Designer から直接複数の形式にエクスポートできます。ファイルをエクスポートするには、**File** メニューの **Export** を選択するか、ツールバーの **Export** ボタン をクリックします。これにより、DHTML、PDF、CSV、Excel（XLS）、Excel 2007（XLSX）、テキスト、XML、またはリッチテキストのいずれかにエクスポートするファイルの種類を選択できる 2 番目のメニューが表示されます。また、ツールバーの **Export File** ボタンをクリックしてエクスポートすることもできます。

DHTML:

目的のエクスポート形式として DHTML を選択すると、新しい DHTML ファイルのファイル名と場所を入力するよう求められます。DHTML はテキストのみの形式なので、チャートは DHTML ファイルと同じディレクトリに別々のイメージファイルとして保存されます。レポートにグラフが含まれている場合は、DHTML にエクスポートする前にチャートのエクスポート形式を設定して、必要な画像形式と属性を指定してください。レポートにイメージがある場合、イメージファイルは DHTML ドキュメントで参照されます。DHTML は HTML よりはるかに正確な出力を生成します。ただし、旧式のブラウザでは、すべてのレポート機能が正確に翻訳されるわけではありません。Internet Explorer 11、Chrome、Firefox、または Edge をお勧めします。

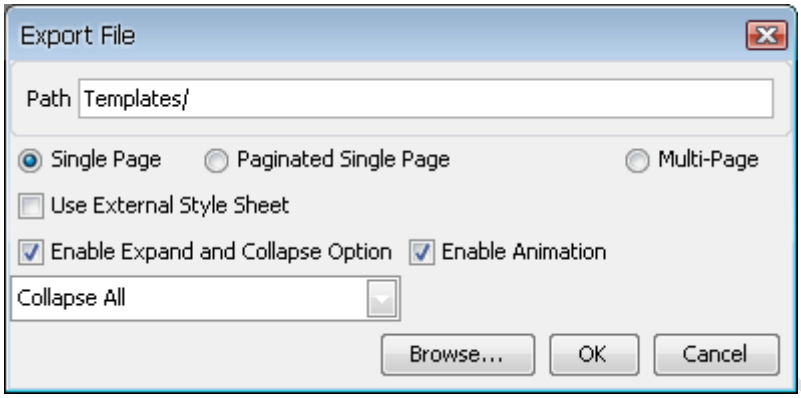
レポートを 1 ページ、複数ページ、またはページ単位で 1 ページにエクスポートするように指定できます。1 ページを指定すると、レポート全体が 1 つの DHTML ファイルにエクスポートされます。複数のページを指定すると、レポートの各ページが新しい DHTML ファイルで生成されます。出力ページは、各ページの上部にあるナビゲーションバーによってリンクされています。ページされた単一ページとしてエクスポートするように指定した場合は、1 つの DHTML ファイルのみが生成されます。ただし、ページサイズ、間隔、および余白は保持され、レポートがブラウザから印刷され、PDF または Report Designer から印刷された場合と同じように表示されます。

内部または外部のスタイルシート（CSS）定義を使用して、各レポート要素のスタイル定義を生成する代わりに、エクスポートされた DHTML に書式を適用するように指定することもできます。このオプションの詳細は、PDF フォントマッピングインポート/エクスポートを参照し

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

てください。



Export to DHTML ダイアログ

グループ化されたデータを含むレポート（サイドバイサイドマスタ&詳細レポートを除く）では、グループ化の展開コントロールと折りたたみコントロールを追加することができます。これにより、レポートをよりコンパクトに表示できます。各グループは、視聴要件に応じて拡大または縮小することができます。このオプションを選択すると、最初のプレゼンテーションを選択することもできます。つまり、レポートを完全に折りたたんだり、完全に展開したり、特定のグループレベルに展開したりすることができます。展開または折りたたまれているグループがフェードインまたはビュー外になっているアニメーションを有効にすることもできます。展開と折りたたみコントロールは 1 ページのエクスポートに対してのみ選択でき、セクションヘッダーまたはフッターが空の場合はコントロールが表示されません。

エクスポートされた DHTML ファイルのタイトルを指定することができます。タイトルは、エクスポートの<title>メタ要素として表示されます。タイトルを追加するには、**Option** メニューから **Html Page Title** を選択します。これにより、タイトルを入力できるダイアログが表示されます。

PDF:

目的のエクスポート形式として PDF を選択すると、新しい PDF ファイルのファイル名と場所を入力するよう求められます。PDF 形式はイメージをサポートしているため、エクスポートによって作成されたファイルはありません。

PDF ファイルの暗号化を設定することもできます。エクスポートダイアログで **Encrypt PDF Export** ボックスをオンにすると、生成されたドキュメントのユーザーパスワードと所有者パスワードを指定できる 2 つの新しいオプションが表示されます。API を使用してファイルをエクスポートする場合、さらに多くの暗号化オプションを使用できます。

PDF へのエクスポートの詳細については、PDF エクスポートオプションを参照してください。

CSV:

CSV エクスポートは、現在のレポートで使用されているデータのテキストファイルをカンマ区

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

切り文字で作成します。目的のエクスポート形式として CSV を選択すると、新しい CSV ファイルのファイル名と場所、ファイルの **Delimiter** を入力するよう求められます。改行文字区切り文字。

CSV エクスポートは、現在のレポートで使用されているデータのテキストファイルをカンマ区切り文字で作成します。 目的のエクスポート形式として CSV を選択すると、新しい CSV ファイルのファイル名と場所、ファイルの **Delimiter**（Tab、Space、Double Space、Comma、Semi-Colon のいずれか）、**Newline Delimiter**（Windows (¥r¥n)、Mac (¥r)、その他 (¥n)、システム）を入力するよう求められます。

CSV エクスポートでは、レポートに関連付けられたデータのみがエクスポートされ、レポート自体のオブジェクトや属性はエクスポートされません。

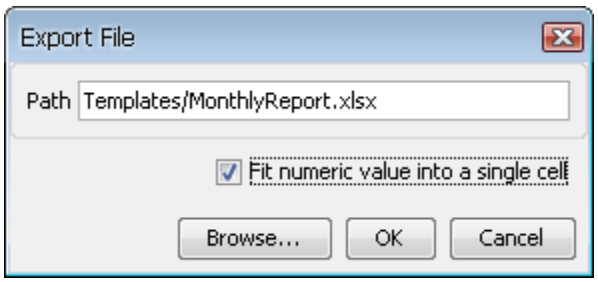
Excel (XLS):

Excel スプレッドシートを目的のエクスポート形式として選択すると、ファイル名と新しいスプレッドシートの場所を入力するよう求められます。EspressReport は、間隔、フォント、色、線、およびグリッドが適用される、書式設定された Excel 出力を生成します。Excel のエクスポートには、チャートや画像も含まれます。チャートはイメージとしてエクスポートされるため、Excel では編集できません。すべての gif、jpeg、png 画像が Excel エクスポートに表示されます。CSV エクスポートを使用して、Excel にプレーンデータダンプを取得できます。

Excel 2007 (XLSX):

Excel 2007（XLSX）エクスポートは、Microsoft Excel 2007 以降で使用される XLSX ファイルを作成します。Excel 2007 を目的のエクスポート形式として選択すると、ファイル名と新しいスプレッドシートの場所を入力するよう求められます。EspressReport は、間隔、フォント、色、線、およびグリッドが適用される書式付き Excel 2007 出力を生成します。Excel 2007 のエクスポートには、チャートやイメージも含まれます。再度、チャートはイメージとしてエクスポートされるため、Excel では編集できません。

レポートで Excel 関数を使用する場合は、**Fit numeric value into a single cell** オプションを選択します。このオプションでは、各数値データを独自のセルにエクスポートして、これらの値を Excel 関数で使用できるようにします。このオプションは、エクスポートされたレポートのセルの配置に影響することがあります。



Export to Excel ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

グループ化されたデータを含むレポートでは、エクスポートされたファイル内の別のシートに各グループを印刷するように選択できます。このオプションは、**Group Header** セクションの **Section** メニューで設定します。詳細は、セクションオプションを参照してください。

Excel スプレッドシートは固定フィールド形式なので、すべてのレポート要素が正しく翻訳されるわけではありません。

TXT:

カンマ区切りのデータファイルを作成する CSV とは異なり、テキストエクスポート機能はレポート全体をテキストファイルとして生成します。レポートは純粋なテキストとしてエクスポートされるため、書式情報は保持されません。目的のエクスポート形式としてテキストを選択すると、ファイル名と新しいテキストファイルの場所、ファイルの **Delimiter**（Tab、Space、Double Space、Comma、Semi-Colon のいずれか）、および **Newline Delimiter** を入力するよう求められます（Windows（¥ r¥n）、Mac（¥ r）、その他（¥ n）、システム）。

XML:

2 つの XML エクスポートバージョンがあります。1 つはデータファイルで、他のエクスポートはフォーマット情報でデータをレポートします。

Pure Data:

この XML エクスポートオプションは、レポートで使用されるデータを XML 形式でエクスポートします。XML ファイルのフォーマットは、XML 入力データのフォーマット要件と同じです。

Data + Format:

この XML エクスポートによって、レポート全体の XML バージョンが作成されます。すべてのレポートデータと書式設定情報が含まれています。このファイルは DHTML エクスポートの XML バージョンと似ており、HTML コンテンツを表示するためにスタイルシートとともに使用することができます。このエクスポートでは、レポート全体を 1 つの XML ファイルとして、またはページごとに別々のファイルとしてエクスポートできます。

Rich Text:

リッチテキストを希望の形式として選択すると、ファイル名と新しいリッチテキストファイルの場所を入力するよう求められます。EspressReport は、RTF 1.5 仕様の全機能を使用して、実際のレポートと非常によく似たリッチテキスト出力を生成します。リッチテキストエクスポートは、すべてのビューアで機能しない場合があります。デフォルトでは、EspressReport は RTF エクスポートのすべての色を直接出力にエクスポートします。ただし、古いバージョンの Microsoft Word は 16 色しかサポートしていません。これらのビューアをサポートするために、16 色を使用して RTF を書き出すことができます。このオプションは、レポートで使用される

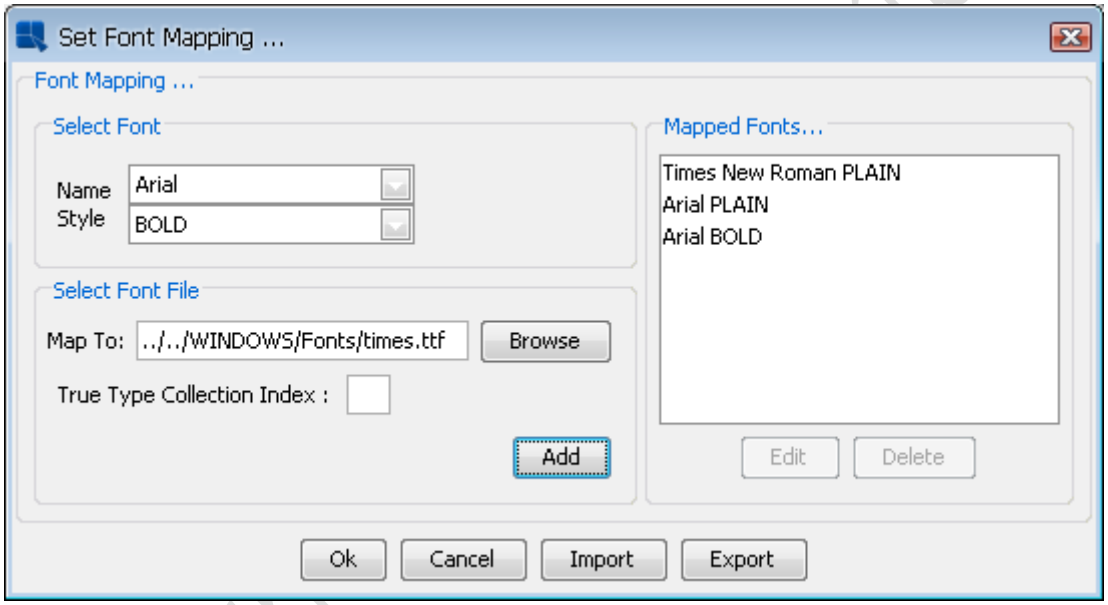
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

色を Microsoft Word 97 の基本カラーパレットに変換します。

3.7.2.1 PDF フォントマッピング

EspressReport では、システム上の任意のフォントをレポート用に使用できます。 ほとんどの書式設定されたエクスポート（DHTML / リッチテキスト/ Excel/Excel 2007）のシステムフォントは、生成された出力で自動的に翻訳されます（リッチテキストと Excel/Excel 2007 の場合、フォントが見つからない場合はデフォルトで Arial になります）。ただし、PDF の場合は、レポートで使用する任意のシステムフォントに対して.ttf（TrueType フォント）.ttc（TrueType コレクション）、.pfb、または.afm ファイルを手動で指定する必要があります。

PDF エクスポートのフォントマッピングを設定するには、**Option** メニューから **Font Mapping** を選択します。これにより、フォントファイルを指定できるダイアログが表示されます。



Font Mapping ダイアログ

フォントとスタイルの組み合わせごとに、特定の.ttf、.ttc、.pfb、または.afm ファイルを選択できます。フルパスを入力するか、フォントファイルを参照することができます。.ttc ファイルを使用している場合は、指定したボックスにフォントインデックスを指定する必要があります（.ttc ファイルには複数のフォントが含まれています）。正しいファイルを指定したら、**Add** ボタンをクリックしてマッピングをリストに保存します。既存のマッピングを編集または削除するには、リスト内でそれらを選択し、該当するボタンをクリックします。

3.7.2.1.1 PDF フォントマッピングインポート/エクスポート

インポート/エクスポート機能を使用して、あるレポートから別のレポートにフォントマッピングを渡すことができます。フォントマッピングをエクスポートするには、フォントマッピングダイアログの **Export** ボタンをクリックします。これにより、ファイル名を指定するダイアログボックスが表示されます。フォントマッピングは XML ファイルとして保存されます。ダイアログから **Import** オプションを選択すると、フォントマッピング XML ファイルを読み込むことができます。これにより、インポート

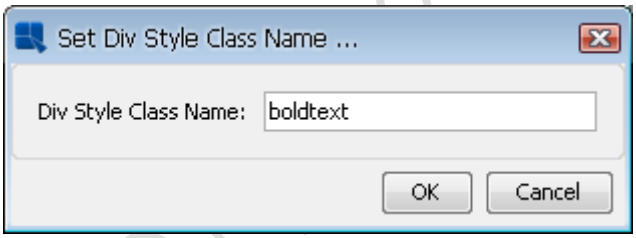
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

する XML ファイルを指定するダイアログボックスが表示されます。**OK** ボタンをクリックすると、XML ファイルに保存されているマッピングが現在のレポートに適用されます。

3.7.2.2 CSS オプション

DHTML にレポートをエクスポートする場合、スタイル定義はデフォルトでレポートの各要素に対して作成されます。これにより、エクスポートが高速になります。ただし、大きなレポートではファイルサイズが非常に大きくなる可能性があります。生成された DHTML ファイルのサイズを制限するために、ユーザはグローバルスタイル定義の内部リストを生成するように選択できます。このオプションはやや遅くなりますが、ファイルがかなり小さくなります。さらに、EspressReport には、レポートをエクスポートするときに.css ファイルを選択することで、ユーザが独自のスタイル定義をレポート要素に適用できるオプションが用意されています。

外部スタイルシート定義を使用する場合は、まず、レポート内の要素に関連付ける外部スタイル定義からクラス名を選択する必要があります。レポート要素のクラスを設定するには、セルを右クリックし、ポップアップメニューから **Set Style Class Name** を選択します。これにより、そのレポート要素のスタイル名を指定できるダイアログが表示されます。



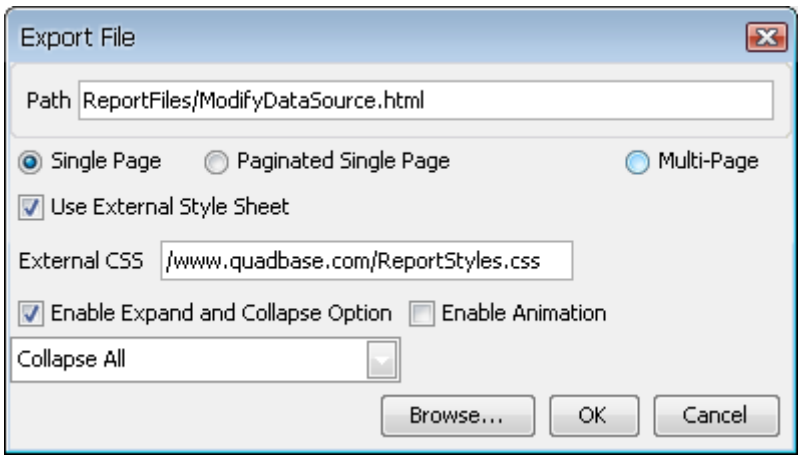
Set Style Class ダイアログ

スタイルクラスを選択しない要素は、レポートを DHTML にエクスポートするときに、レポートの設定に基づいて独自のスタイルを生成します。

また、レポートに定義されている書式の.css ファイルをエクスポートすることもできます。 これらのスタイル定義は、内部スタイルシートオプションによって生成されるものと同じです。 この機能を使用して、Report Designer で作成した書式を他のレポートや他の Web ページに適用することができます。.css ファイルをレポートからエクスポートするには、**Option** メニューから **Export Style Sheet** を選択します。ダイアログが開き、生成されたファイルの名前と場所を尋ねられます。

CSS オプションは、DHTML エクスポートのエクスポートダイアログで設定できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



DHTML Export ダイアログ

Use External Style Sheet オプションでは、レポート要素に割り当てたスタイルクラスの定義を含む.css ファイルの場所（相対パスまたは完全な HTTP パス）を指定できます。

外部スタイルシートファイルからの書式設定は、DHTML エクスポートにのみ適用されます。他のエクスポート形式では、レポートテンプレートで定義された書式が使用されます。

Copyright(C) 2018 Climb Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

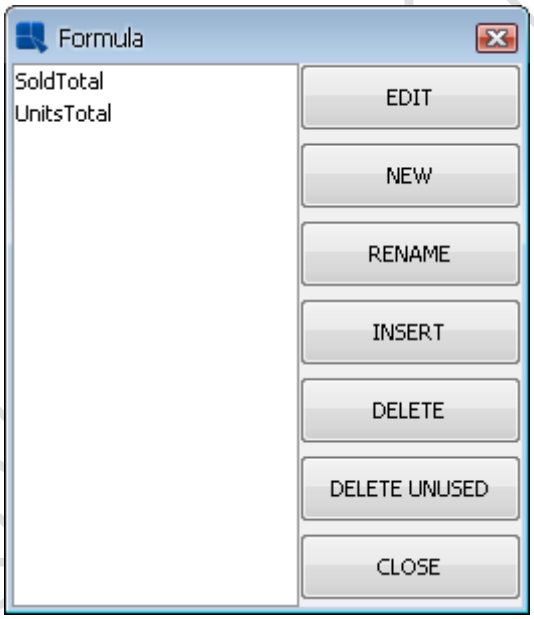
3.8 数式および式ビルダの使用

EspressReport は、さまざまな数式をサポートしており、レポートデータを操作および表示するための重要なメカニズムを提供します。数式を使用すると、70 を超える組み込み関数の 1 つを使用して、要約と集計を追加したり、基本的な計算を実行したり、複雑な式を作成することができます。

EspressReport の式の構文の一部は、v2.51 と v3.0 の間で大幅に変更されています。ほとんどの場合、非推奨の構文は引き続き正しく動作します。ただし、古いテンプレートをチェックして v2.0 からバージョン 3.0 以降にアップグレードした場合でも関数が正しく返されるようにすることをお勧めします。

3.8.1 数式の作成

Insert メニューから **Insert Formula** を選択するか、ツールバーの **Insert Formula** ボタンをクリックして新しい式をレポートに挿入すると、レポートに定義されているすべての数式の一覧が表示されます。



Formula List ダイアログ

レポートに式を挿入するには、リストから式を選択し、**Insert** ボタンをクリックします。**Formula List** ダイアログが閉じ、数式を配置することができます。

レポートでセクションが繰り返されるたびに式がリセットされるため、式を配置するレポートセクションは非常に重要です。たとえば、サマリブレークレポートのグループフッターに数式を配置すると、レポート内の各グループの数式が再計算されます。数式が列の値を参照する場合、各ブレークグループ内のデータのみが使用されます。レポートフッターセクションに同じ数式を配置すると、1 回だけ計算され、ブレークとグループ化に関係なく、列フィールドのすべてのデータが使用されます。

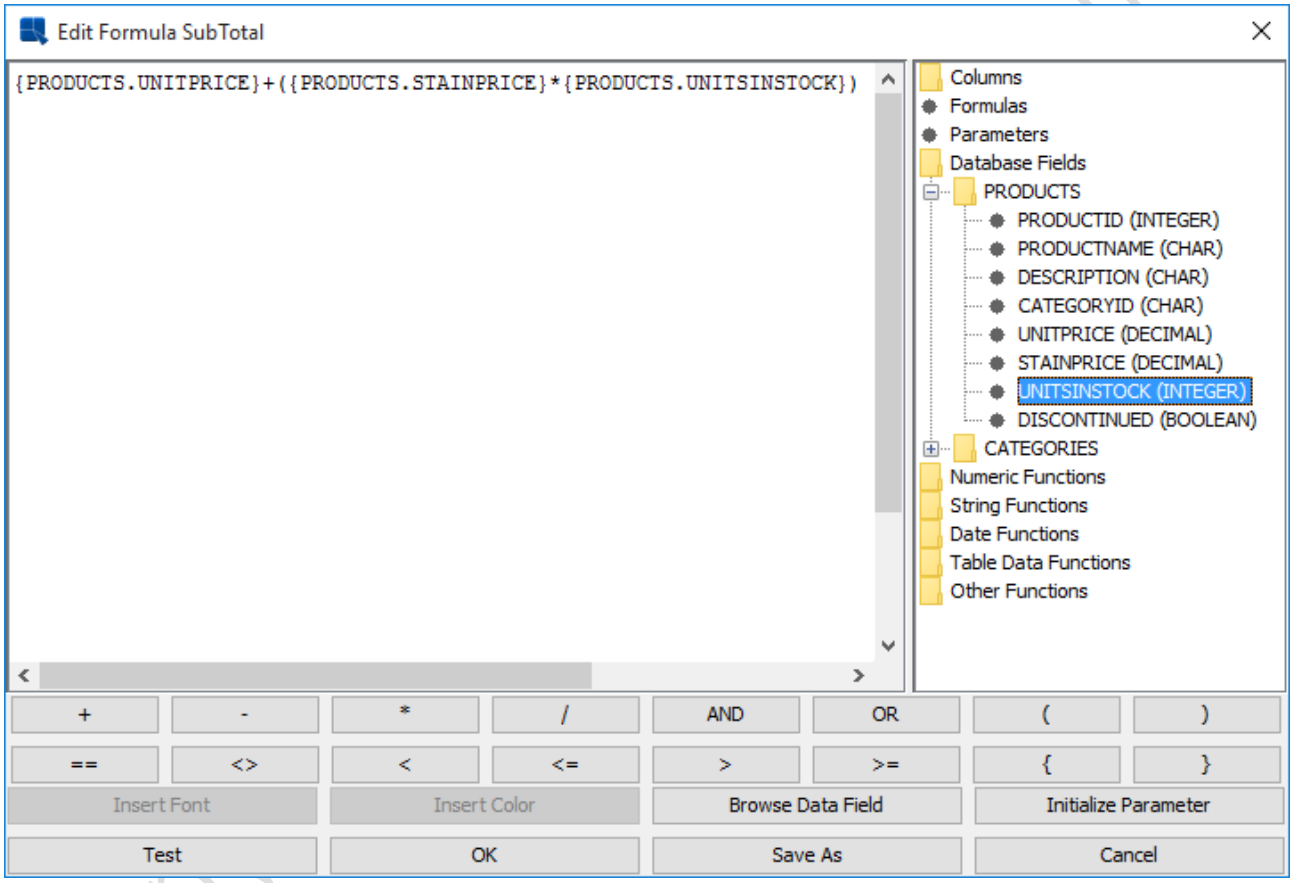
レポートのテーブルデータセクションに配置された数式とラベルは、計算列になり、列フィールドと式として扱うことができます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ラベルは自動的に文字列として扱われ、それらをダブルクリックして編集することができます。つまり、式で整数を使用し、数式ビルダを開き、数値から二重引用符を削除します。

3.8.1.1 式ビルダ

数式は、数式ビルダインタフェースを使用して作成されます。数式ビルダを起動するには、Formula List ダイアログから **New** を選択して、新しい数式を作成することを選択できます。新しい式の名前を指定するよう求められます。名前を指定すると、数式ビルダが開き、数式を作成できます。数式ビルダを使用して既存の数式を編集することもできます。数式を編集するには、リストから選択し、**Edit** ボタンをクリックするか、デザインウィンドウで数式セルをダブルクリックします。



数式ビルダウィンドウ

数式ビルダのメインウィンドウには、数式のテキストが含まれています。右側のフォルダには、列フィールド、その他の数式、パラメータ値、データベースフィールド、および組み込み関数を含む、数式に追加できるさまざまな要素が含まれています。最初の 2 行のボタンには、最も一般的な算術演算子とブール演算子が含まれています（演算子と関数については、次のセクションで説明します）。最後の 2 行のボタンは、コマンドボタンとして機能し、次の機能を実行します。

Insert Font

このオプションは数式には使用できません。スクリプトにのみ使用可能です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Insert Color

このオプションは数式には使用できません。スクリプトにのみ使用可能です。

Browse Data Field

このオプションはデータベースフィールドにのみ使用できます。データベースにクエリし、列の最初の 20 個の値を返します。

Initialize Parameter

このオプションは、数式で定義されたパラメータのパラメータ初期化ダイアログを表示します。パラメータについては、式パラメータで説明しています。

Test

このオプションは、現在の数式をチェックし、正しいかどうかを確認します。

OK

このオプションを選択すると、数式ビルダが閉じられ、Formula List ダイアログに戻り、数式をレポートに挿入できます。

Save As

このオプションを使用すると、現在の数式を別の名前で保存できます。レポートの別の場所で式を再利用できるため、**Save As** オプションを使用すると、式が参照されているすべての場所で式を変更することなく、1 つの場所で式を変更できます。

Cancel

このオプションは、式/変更を保存せずに式ビルダを終了します。

3.8.2 数式構文

以下のセクションでは、EspressReport で利用可能なさまざまな演算子と関数、およびその使用法について詳しく説明します。

3.8.2.1 列フィールドのデータの使用

数式の列フィールドのデータを使用することがよくあります。これは、次の構文を使用して式で簡単に実行できます。{<**Field Name**>}。中括弧はフィールドを区切り、フィールド名は参照している列フィールドの名前です。よって、基本的な例では: {**UnitPrice**} * {**Quantity**}は 2 つの列フィールドを一緒に掛けます。

ほとんどの場合、列名はデータソースで指定された名前（つまり、データベースの列名またはエイリアス）です。正しい名前がわからない場合は、数式ビルダの右パネルの列ノードから、使用する列フィールドを選択するだけです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

列データを取得するために使用できる関数には、次の 2 つがあります。

id()

このオプションを使用すると、列フィールドを含むレポート内のセルの値を取得できます。構文は **id(<Object ID>)**です。セルを右クリックし、ポップアップメニューから **Property** を選択すると、オブジェクトの ID を取得できます。また、**Data** メニューから **View Column Mapping** を選択することで、列フィールドの ID を取得することもできます。

デフォルト ID を使用する代わりに、レポートの要素にカスタム ID を割り当てることもできます。要素にカスタム ID を割り当てるには、Report Designer でカスタム ID を右クリックし、ポップアップメニューから **Custom ID** を選択します。これにより、その要素の ID を入力できるダイアログが表示されます。**id()**関数の引数は、要素の元の ID でも、定義済みのカスタム ID でもかまいません。

要素 ID を使用して、Report API の要素へのハンドルを取得することもできます。元の ID またはカスタム ID のいずれかを **getData()**メソッドに渡して、要素へのハンドルを取得することができます。

id()関数はまた別の重要な目的を果たします。計算された列の計算を遅らせるのに使用できます。通常、レポートのテーブルデータセクションの式は、グループまたはテーブルフッターの集計よりも前に計算されます。ただし、列フィールドの前に集計を計算することがあります。たとえば、合計の各行の割合を計算する列を作成するとします。正しい結果を得るには、列の前に合計を計算する必要があります。これを行うには、単に **id()**関数を使用してテーブル/グループフッターの集計を参照します。テーブルデータセクションの完成した関数は、**{Quantity}/id（TBL0_FTR_FORM0）**のようになります。

col()

このオプションは、以前のバージョンの EspressReport で列フィールドからデータを取得するために使用される非推奨の関数です。この機能はまだ有効ですが、新しいフィールド記法を使用することをお勧めします。この関数の構文は **col(Column Index)**です。列インデックスを表示するには、**Data** メニューから **View Column Mapping** を選択します。各列フィールドに割り当てられたインデックス番号が表示されます。インデックス値は、レポートの列が選択された順序に基づいて割り当てられます。最初の列のインデックスは 0 です。

3.8.2.1.1 列集約

EspressReport で提供される集計関数のいずれかを使用して、任意の列フィールド（計算列を含む）を集計できます。次の集約オプションを使用できます。

average()

このオプションは、指定された列の平均値を返します。構文は **average(<Column Field>)**です。**UnitPrice** という列フィールドの平均値を返すには、**average({UnitPrice})**の式を使用します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

median()

このオプションは、指定された列の中央値を返します。構文は **median(<Column Field>)**です。列の中央値が 2 つの値（偶数行）の場合、集計は 2 つの値の平均を返します。

count()

このオプションは、指定された列の値の数を返します。構文は **count(<Column Field>)**です。

countdistinct()

このオプションは、指定された列内のすべての個別値の数を返します。構文は **countdistinct(<Column Filed>)**です。

max()

このオプションは指定された列内の最大値を返します。構文は **max(<Column Field>)**です。

min()

このオプションは指定された列内の最小値を返します。構文は **min(<Column Field>)**です。

stddev()

このオプションは指定された列内の標準偏差を返します。構文は **stddev(<Column Field>)**です。

sum()

このオプションは指定された列内の合計値を返します。構文は **sum(<Column Field>)**です。

sumsquare()

このオプションは指定された列内の平方根を返します。構文は **sumsquare(<Column Field>)**です。

variance()

このオプションは、指定された列の分散を返します。構文は **variance(<Column Field>)**です。

集計関数は、列フィールドを引数として取ることができるだけであることに注意することが重要です。式や別の関数を引数として使用することはできません。したがって、**sum({UnitPrice} * {Quantity}**は無効です。この計算を行うには、まず、レポートのテーブルデータセクションに**{UnitPrice} * {Quantity}**を数式として追加する必要があります。次に、最初の数式で作成された計算列を **sum(@formulaname)**で合計することができます。

3.8.2.2 式の使用

EspressReport では、レポート式を新しい式に直接差し込んで簡単に再利用できます。次の構文を使用して式を参照できます。**@<FormulaName>**。数式名は、作成された最初の数式に割り当てられた名前です。たとえば、**Total** という名前の数式**{UnitPrice} * {Quantity}**を作成した場合は、@Total を入力してこの数式の結果を取得できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

正しい名前が不明な場合は、数式ビルダの右パネルの **Formulas** ノードから使用する式を選択するだけです。

数式がレポートのデータテーブルセクションに配置されている場合は、この表記法を使用して計算列を集計することもできます。

3.8.2.3 パラメータ値の使用

列フィールドと数式値に加えて、数式の一部としてパラメータ値にアクセスすることもできます。パラメータ値は、クエリまたは式パラメータのいずれかにユーザが入力した値です。クエリパラメータの詳細は、パラメータ化されたクエリを参照してください。数式パラメータの詳細については、式パラメータを参照してください。

パラメータ値を参照するには、クエリパラメータの場合は<**ParameterName**>、式パラメータの場合は<**ParameterName**>の構文を使用します。たとえば、レポートに **StartDate** という名前のクエリパラメータがあり、ユーザに日付を入力するように要求する場合、**StartDate** は、レポートの実行時にユーザが指定した日付を返します。

正しい名前が分からない場合は、数式ビルダの右パネルにある **Parameter** ノードから、使用するパラメータを選択するだけです。また、数式パラメータ名を間違って入力すると、数式作成者は新しいパラメータを定義しようとしていると見なします。

3.8.2.4 データベースフィールドの使用

レポートでデータベースをデータソースとして使用する場合は、式で選択されていないデータベースフィールドを使用することもできます。データベースフィールドは、列フィールドと同様の方法で参照され、構文{<**Table Name**> <**Field Name**>}を使用します。データベースに 3 つの部分名が必要な場合は、フィールドを適宜参照する必要があります。

数式ビルダの右側のパネルの **Database Field** ノードから追加するデータベースフィールドを選択できます。また、数式ビルダを使用すると、データベースフィールドをプレビューできます。これを行うには、まず、右側のパネルでフィールドを選択し、**Browse Data Field** ボタンをクリックします。これにより、選択した列の最初の数行を示す新しいダイアログが表示されます。

3.8.2.5 コンスタント

数式で定数を使用する場合、各データ型に必要な形式があります。

Numeric Data

数値定数を表すには、単に数値を入力するだけです。整数と小数の両方を受け入れることができます。ただし、千単位の区切り文字は入力できません。したがって、数式で定数 12.28 を指定するには、**12.28** と入力します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

String Data

文字列定数を表すには、二重引用符で囲んだ文字列を入力します。たとえば、文字列 Hello を指定するには、**"Hello"**と入力します。

Boolean Data

ブール定数を表すには、デリミタなしで真または偽のいずれかを入力します。偽のブール値を指定するには、**false** と入力します。大文字と小文字は区別されません。

Date Data

日付定数を表すには、デリミタなしの次の形式で日付を入力します。**MM/dd/yyyy**。2002 年 8 月 5 日の日付を指定するには、**08/05/2002** と入力します。

Time Data

時定数を表すには、区切り文字なしの時刻を hh:mm:ss の形式で入力します。時刻を午後 2:30 に指定するには、**14:30:00** と入力します。時間は 24 時間形式でなければならないことに注意してください。

Timestamp Data

タイムスタンプ定数を表すには、区切り文字なしの次の形式で日付と時刻を入力します。MM/dd/yyyy hh:mm:ss。2002 年 8 月 5 日午後 2 時 30 分にタイムスタンプを指定するには、**08/05/2002 14:30:00** と入力します。

3.8.2.6 式パラメータ

数式に定数を追加するのではなく、パラメータを指定することができます。パラメータを使用すると、実行時にユーザ（または他の場所）から一定の値を収集し、それらの値に基づいて数式を動的に計算することができます。数式にパラメータを指定するには、疑問符**"？"**の後にパラメータ名を続けて使用します。

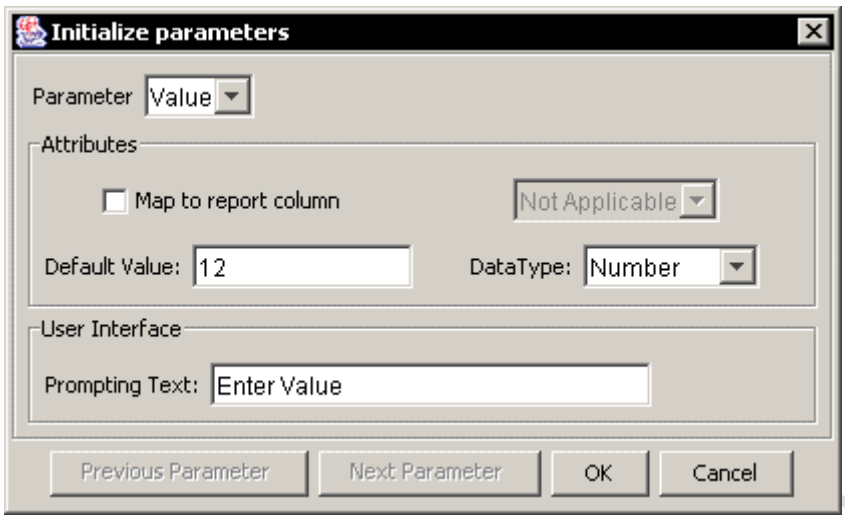
たとえば、次の数式**{Quantity}+？Value** を使用すると、レポートの **Quantity** フィールドにユーザが入力した値が追加されます。

数式内で新しいパラメータを定義する場合は、一意の名前を使用する必要があります。レポート内の別の数式で既に定義されているパラメータのパラメータ名を指定すると、新しい数式で前のパラメータの値が返されます。

3.8.2.6.1 式パラメータの初期化

クエリパラメータ（クエリパラメータの初期化を参照）と同様に、式をレポートで使用する前に、式パラメータを初期化する必要があります。現在の数式で定義されたパラメータを初期化するには、数式ビルダの **Initialize Parameter** ボタンをクリックします。これにより、パラメータのオプションを指定するダイアログが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Initialize Formula Parameter ダイアログ

このダイアログでは、パラメータのデフォルト値を指定し、パラメータをレポート列にマップします（これにより、パラメータに入力するのではなく、個別の値のドロップダウンリストが表示されます）。指定されたパラメータに対してデータ型を指定するとともに、エンドユーザのプロンプトを指定することができます。**Previous Parameter** ボタンと **Next Parameter** ボタンをクリックすると、数式で定義されている各パラメータを初期化できます。すべてのパラメータのオプションの設定が完了したら、**OK** をクリックします。

数式パラメータの場合は、各パラメータのデフォルト値を指定する必要があります。また、パラメータがレポート列にマップされている場合、初めてレポートをプレビューするときには、その列の最初の 20 個の値のみがダイアログに表示されます。これは、レポートが最初に作成されたときに完全なデータセットが取得されないためです。実行時に、ダイアログには、使用可能なすべての値がレポート列に表示されます。

3.8.2.7 演算子

EspressReport には、数式で式を作成できるようにする算術演算子とブール演算子がいくつか用意されています。演算子は、演算子を式の引数、つまり<**argument1**> <**operator**> <**argument2**>の間に配置する in-fix 表記法を使用します。

数式ビルダの演算子ボタンの 1 つをクリックすると、自動的に演算子が数式に挿入されます。

3.8.2.7.1 算術演算子

+、**-**、*****、/の 4 つの基本算術演算子がサポートされています。これらは、2 つのオブジェクトをそれぞれ加算、減算、乗算、および除算します。たとえば、1+2 は 3 を返します。

一般に、算術演算子は引数として数値をとり、2 つの例外を持つ数値を返します。+は文字列を連結するために使用できます。たとえば、**Hello+"World"**は **Hello World** を返します。また、**-**を使用すると、2 つの日付オブジェクトを減算し、日数の差を返すことができます。例えば、**8/2/2002-4/7/2002** は **117** を返します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.8.2.7.2 ブール演算子

AND、OR、==、<、<=、>、>=、<>の 8 つのブール演算子がサポートされています。**AND** と **OR** はブール引数を取り、ブール値を返します。たとえば、**true AND true** は **true** を返し、**1 <2 OR 4 <3** は **true** を返します。

==、<、<=、>、> =、<>は、それぞれ等しい、より小さい、より小さい、より大きい、等しい、より大きい、の意味を持ちます。比較演算子は、同じデータ型の 2 つのオブジェクトを比較し、ブール値を返します。たとえば、3<4 の場合は true、**Yes == "No"**は false を返します。

ブール演算子は、数式ではなくセルスクリプトでよく使用されます。セルスクリプトについては、スクリプトで説明しています。

3.8.2.8 関数

基本的な演算子に加えて、EspressReport にはいくつかの組み込み関数が用意されています。関数構文は一般に、カンマで区切られた引数 **function(<argument>,<argument>)**を囲む括弧のセットの外にある関数名の形式をとります。

数式ビルダの数値関数、文字列関数、日付関数ノードで必要な関数を選択することにより、数式に関数を挿入できます。関数には、引数の型と数を示すヒントが挿入されます。関数が適切に戻るためには、これらのヒントを削除し、有効な引数に置き換える必要があります。

3.8.2.8.1 数値関数

数値関数を次に示します。

abs()

このオプションは、数値の絶対値を返します。構文は **abs(<Number>)**です。たとえば、**abs (-12)** は **12** を返します。

acos()

このオプションは、0 から pi の範囲のアークコサインを返します。構文は **acos(<radians>)**です。たとえば、**acos(-sqrt(2)/2)**は **3*pi/4** を返します。

asin()

このオプションは、-pi/2 から pi/2 の範囲の角度のアークサインを返します。構文は **asin(<radians>)**です。たとえば、**asin(sqrt(2)/2)**は **pi/4** を返します。

atan()

このオプションは、-pi/2 から pi/2 の範囲のアークタンジェントを返します。構文は **atan(<radians>)**です。たとえば、**atan(1)**は **pi/4** を返します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

atan2()

このオプションは、直交座標(**b, a**)を極座標(**r, θ**)に変換します。この関数は、**b/a** の逆正接を-pi から pi の範囲でとって **θ** を返します。構文は **atan2(<y_coordinate>,<x_coordinate>)**です。たとえば **atan2(1,0)**は **pi/2** を返します。

ceil()

このオプションは、指定された数値より大きい最小の整数値を返します。構文は **ceil(<Number>)** です。たとえば、**ceil(15.3)**は **16** を返します。

cos()

このオプションは、角度の余弦をラジアンで返します。構文は **cos(<radians>)**です。たとえば、**cos(3* pi()/4)**は**-sqrt(2)/2** を返します。

e()

このオプションは自然対数の底 **e** の値を返します。構文は引数なしの **e()**です。

exp()

このオプションは、e の n 乗を返します。構文は **exp(<Number>)**です。たとえば、**exp(3)**は **e^3** または **20.086** を返します。

factorial()

このオプションは、指定されたオブジェクトの階乗を返します。構文は **factorial(<Number>)**です。たとえば **fatorial(3)**は **6** を返します。

floor()

このオプションは、指定された引数よりも小さい最大の整数値を返します。構文は **floor(<Number>)**です。たとえば **floor(5.52)**は **5** を返します。

getColumnCount()

このオプションは、レポートの列の総数を示す整数を返します。構文は引数なしの **getColumnCount()**です。この関数は、表示されているかどうかにかかわらず、列の合計数を返します。

getRowIndex()

このオプションは、レポートの現在の行（テーブルデータの）のインデックス値を示す整数を返します。構文は引数なしで **getRowIndex()**です。

getTotalRowIndex()

このオプションは、レポートのすべての行（テーブルデータ内）の数を返します。構文は引数なしの **getTotalRowIndex()**です。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

getRowIndexOfCurrentTable()

このオプションは、レポート内の現在のグループ（テーブル）の現在の行を返します。この機能は、通常、サマリブレークやマスタ＆詳細レイアウトなどのデータをグループ化したレポートでのみ有効です。たとえば、現在の行がデータのグループの 3 番目の行である場合、**getRowIndexOfCurrentTable()**は、（**getRowIndex()**関数を使用して取得できる）合計レポートの行インデックスに関係なく、常に 2 を返します。構文は引数なしの **getRowIndexOfCurrentTable()** です。

getTotalRowIndexOfCurrentTable()

このオプションは、レポート内の現在のグループ（テーブル）内のすべての行の数を返します。この機能は、通常、サマリブレークやマスタ＆詳細レイアウトなどのデータをグループ化したレポートでのみ有効です。構文は引数なしの **getTotalRowIndexOfCurrentTable()**です。

getSiblingCount()

このオプションは、ネストレベル内の兄弟グループの数のカウントを返します。この機能は、通常、サマリブレークやマスタ＆詳細レイアウトなどのデータをグループ化したレポートでのみ有効です。たとえば、グループ化のレベルが 1 つのサマリレポートがあります。行ブレーク列には 3 つの異なる値があり、レベルに 3 つのグループが作成されます。グループ内（つまり、グループヘッダー、テーブルデータ、またはグループフッターセクション）では、**getSiblingCount()**は **3** を返します。構文は引数なしで **getSiblingCount()**です。

getChildCount()

このオプションは、ネストレベル内の子グループの数のカウントを返します。この機能は、通常、サマリブレークやマスタ＆詳細レイアウトなどのデータをグループ化したレポートでのみ有効です。たとえば、前の関数で説明したのと同じ例があります。グループ内では、グループ化のレベルが 1 つのみであるため、**getChildCount()**関数は 0 を返します。ただし、グループ外（テーブルヘッダーまたはテーブルフッターセクション）では、**getChildCount()**は **3** を返します。構文は引数なしで **getChildCount()**です。

getGroupIndex()

このオプションは、現在のグループのインデックス値を示す整数を返します。この機能は、サマリブレークやマスタ＆詳細レイアウトなどのデータをグループ化したレポートにのみ関係します。この関数は、ネストされたグループを持つレポートの最も内側のグループのインデックスのみを返します。

log()

このオプションは、指定された数値の自然対数を返します。構文は **log(<Number>)**です。たとえば、**log(10)**は **2.303** を返します。

mod()

このオプションは、2 つの数値で除算した後の余りを返します。最初の引数は分子で、2 番目の引

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

数は分母です。構文は **mod(<Number>,<Number>)**です。例えば **mod (12,7)** は **5** を返します。

pi()

このオプションは円周率を返します。構文は **pi()**で、引数を持ちません。

pow()

このオプションは最初の引数の二つ目の引数での累乗値を返します。構文は **pow(<Number>,<Number>)**です。例えば、**pow(2,3)**は、**8** を返します。

random()

このオプションは 0 以上 1 以下のランダムな数値を返します。構文は **random()**で引数を持ちません。

rint()

このオプションは、指定された引数を最も近い整数に丸めます。構文は **rint(<Number>)**です。たとえば、**rint(5.6)**は **6** を返します。

sin()

このオプションは、角度の正弦をラジアンで返します。構文は **sin(<radians>)**です。たとえば **sin(pi()/2)**は **1** を返します。

sqrt()

このオプションは、指定された数値の平方根を返します。構文は **sqrt(<Number>)**です。たとえば、**sqrt(64)**は **8** を返します。

tan()

このオプションは、角度の正接をラジアンで返します。構文は **tan(<radians>)**です。たとえば、**tan(pi()/4)**は **1** を返します。

toDegrees()

このオプションは、弧度法を度数法に変換します。構文は **toDegrees(<Number>)**です。たとえば、**toDegrees(pi())**は **180** を返します。

toRadians()

このオプションは度数法を弧度法に変換します。構文は **toRadians(<Number>)**です。たとえば、**toRadians(180)**は **3.142** を返します。

toString()

このオプションは数値を文字列に変換します。構文は **toString(<Number>,<Number of Decimals>,<Round Up(True/False)>)**です。例えば、**toString(12.216,2,True)**は文字列として **"12.22"**を返します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.8.2.8.2 文字列関数

次の関数は文字列引数をとります。

getAllRowData()

このオプションは、**(COLUMN1_NAME) VALUE1(COLUMN2_NAME)VALUE2** のような形式でデータの行全体を返します。引数は必要ありません。

getHeader()

このオプションは、列フィールドの列名を返します。構文は **getHeader(<Column Field>)**です。たとえば、**UnitPrice** という名前の列がある場合、**getHeader({UnitPrice})**は **UnitPrice** を文字列として返します。

getPage()

このオプションは現在のページ番号を返します。構文は引数なしで **getPage()**です。

getTotalPages()

このオプションは、現在のレポートの総ページ数を返します。構文は引数なしの **getTotalPages()**です。

getTotalSections()

このオプションは、現在のレポートのセクションの総数（つまり、水平に必要なページ数）を返します。構文は、**getTotalSections()**です。オブジェクトはありません。

indexOf()

このオプションは、文字列内の指定されたパターンが出現する最初のインデックス値を返します。この関数には 2 つの構文があります。

1. **indexOf(<String>,<Pattern>)**。たとえば、**indexOf("Banana","an")**は **1** を返します。
2. **indexOf(<String>,<Pattern>,<Starting Index>)**。たとえば、**indexOf("Banana","an",2)** は 3 を返します。

insert()

このオプションを使用すると、文字列に新しい文字を挿入できます。構文は、**insert(<String>,<Character Number>,<New Characters>)**です。例えば、**insert("Wood Natural Furniture",4,"View")**は **WoodView Natural Furniture** を返します。

lastIndexOf()

このオプションは、文字列内の指定されたパターンが出現する最後のインデックス値を返します。この関数には 2 つの構文があります。

1. **lastIndexOf(<String>,<Pattern>)**。たとえば、**lastIndexOf("Banana","an")**は **3** を返します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2. **lastIndexOf(<String>,<Pattern>,<Starting Index>)**。たとえば、**lastIndexOf("Banana","an",2)** は **1** を返します。

replace()

このオプションを使用すると、文字列内の文字セットを別の文字セットに置き換えることができます。この関数には 2 つの構文があります。

1. **replace(String,Character Start Number, Character End Number, New Character)**。たとえば、replace(“Today is a rainy day”,11,16,”sunny”)は、**Today is a sunny day** を返します。
2. **replace(<String>,<Old Characters>,<New Characters>)**。たとえば、**replace(“Today is a rainy day”,”rainy”,”sanny”)**は **Today is a sunny day** を返します。

setMaxLength()

このオプションを使用すると、文字列の最大長を設定できます。構文は **setMaxLength(<String>,<Maximum Number of Characters>)**です。たとえば、**setMaxLength("You're a firefighter",13)**は、**You’re a fire** を返します。

strcmp()

このオプションは、2 つの文字列を比較し、それらの辞書的な違いを返します。最初の引数が 2 番目の引数より大きい場合、結果は正の値をとります。最初の引数が 2 番目の引数よりも小さい場合、結果は負の値をとります。文字列が同じ場合、結果は 0 です。構文は **strcmp(<String>,<String>)** です。たとえば、**strcmp("a","c")**は**-2** を返します。

strcmpIgnoreCase()

このオプションは、2 つの文字列を比較し、大文字と小文字を区別せずにそれらの辞書的な違いを返します。構文は **strcmpIgnoreCase(<String>,<String>)**です。たとえば、**strcmpIgnoreCase("a","A")**は **0** を返します。

strcat()

このオプションは、複数の文字列を連結します。構文は **strcat(<String>,<String>,<String> ...)** です。たとえば、**strcat("Wood","View")**は **WoodView** を返します。また、"+"演算子を使用して文字列を連結することもできます。

strlen()

このオプションは、指定された文字列の文字数を示す整数を返します。構文は **strlen(<String>)** です。たとえば、**strlen("ReportDesigner")**は **14** を返します。

substring()

このオプションは、引数で指定された文字列の一部を返します。この関数には 2 つの構文があります。

1. **substring(<String>,<Character Start Number>)**。たとえば、**substring("unhappy",2)**は **happy** を返します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2. **substring(<String>,<Character Start Number>,<Character End Number>)**。たとえば、**substring("smiles"、1,5)**は **mile** を返します。

trim()

このオプションは、指定された文字列の先頭または末尾のスペースを削除します。構文は **trim(<String>)**です。たとえば、**trim(" Hello ")**は **Hello** を返します。

toLowerCase()

このオプションは、指定した文字列内の大文字を小文字に変換します。構文は **toLowerCase(<String>)**です。たとえば、**toLowerCase("ABCdef")**は **abcdef** を返します。

toNumeric()

このオプションは文字列を **double** に変換します。構文は **toNumeric(<String>)**です。たとえば、**toNumeric("425.52")**は **425.52** を返します。この数式が正しく機能するには、文字列引数に数字を含める必要があります。

toUpperCase()

このオプションは、指定された文字列内のすべての小文字を大文字に変換します。構文は **toUpperCase(<String>)**です。たとえば、**toUpperCase("ABCdef")**は **ABCDEF** を返します。

3.8.2.8.3 日付/時間関数

次の関数は、日付/時刻引数を使用します。日付/時刻関数の中には、カレンダーフィールドを示す特殊な引数を使用するものがあります。カレンダーフィールドの引数を指定するには、区切り文字なしで次のいずれかを入力します。

ERA	DAY_OF_MONTH	HOUR
YEAR	DAY_OF_YEAR	HOUR_OF_DAY
MONTH	DAY_OF_WEEK	MINUTE
WEEK_OF_YEAR	DAY_OF_WEEK_IN_MONTH	SECOND
WEEK_OF_MONTH	AM_PM	MILLISECOND

addTime()

このオプションは、指定された日付/時刻に指定された時間を追加します。構文は **addTime(<Date/Time>,<Calendar Field>,<Number>)**です。数値は、カレンダーフィールドを日付/時刻オブジェクトに追加する必要がある量を指定します。たとえば、
addTime(12/5/1998,MONTH,5)は **May 5, 1999** を返し、
addTime(12/5/1998,DAY_OF_MONTH,-25)は **Nov 10, 1998** を返します。

getAmPm()

このオプションは、指定された時間の文字列として AM または PM を返します。構文は

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

getAmPm(<Time>)です。たとえば、**getAmPm(13:24:00)**は **PM** を返します。

getCurrentDate()

このオプションはシステムの現在の日付を返します。構文は **getCurrentDate()**で、引数はありません。

getCurrentDateTime()

このオプションはシステムの現在の日付と時間を返します。構文は **getCurrentDateTime()**で、引数はありません。

getCurrentTime()

このオプションはシステムの現在の時間を返します。構文は **getCurrentTime()**で、引数はありません。

getDateTime()

このオプションは、指定されたカレンダーフィールドの値を指定された日時に返します。構文は **getDateTime(<Date/Time>,<Calendar Field>)**です。たとえば、**getDateTime(12:24:00,MINUTE)**は **24** を返し、**getDateTime(10/10/2001,DAY_OF_WEEK)**は **4**（水曜日を意味します）を返します。

3.8.2.8.4 テーブルデータ関数

次の関数は数値を引数にとります:

.row()

このオプションは、行インデックスに従って列フィールドの値を返します。これを行うには、次の構文を持つ列フィールドを追加します:**.row(Row Index)**。行インデックスは、レポートの各行に関連付けられたインデックス値です。たとえば、**{ProductName}.row（3）**は **ProductName** フィールドの 4 番目の値を返します。

.getScriptValue()

このオプションは、スクリプトを適用した後の行インデックスに従って列フィールドの値を返します。列にスクリプトが適用されていない場合は**.row()**と同じ値を返します。詳細は、他のスクリプトで計算された値へのアクセスを参照してください。

3.8.2.8.5 その他の関数

次の関数は文字列を引数にとります:

first()

このオプションは、列フィールドの最初の値を返します。構文は **first(<Column Field>)**です。たとえば、**first({UnitPrice})**は UnitPrice フィールドの最初の値を返します。

Last()

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

このオプションは、列フィールドの最後の値を返します。構文は **last(<Column Field>)**です。たとえば **last（{UnitPrice}）** は UnitPrice フィールドの最後の値を返します。

3.8.2.8.6 カスタム関数

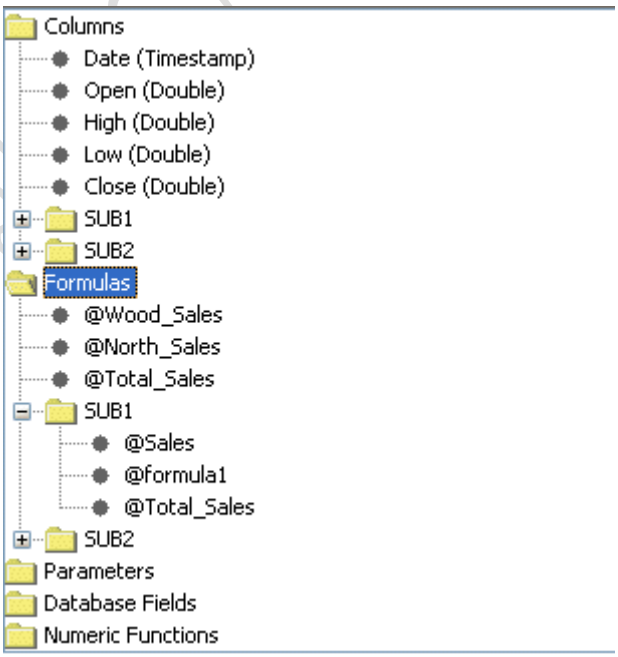
EspressReport で提供される関数に加えて、API を介して Report Designer を起動するときに、独自のカスタム関数を数式ビルダに含めることができます。すべての Java 関数がサポートされています。この機能の詳細については、カスタム関数を含む Report Designer を開くを参照してください。

3.8.3 サブレポートの数式アクセス

メインレポートでは、サブレポートの列と数式にアクセスし、接頭辞 **SUB#** を使用して独自の列として使用できます。この"#"は 1 から始まるサブレポートの番号です。
たとえば、サブレポート 1 から **UnitPrice** 列にアクセスする構文は、**SUB1.{UnitPrice}**または **SUB1.COL(i)**です。ここで、"i"は **UnitPrice** の列番号です。サブレポート 1 から数式 **Sales** にアクセスするための構文は **SUB1.@Sales.**です。

サブレポート列（即ち、**SUB1.{UnitPrice}**）にアクセスすると、列の最初の行の値である単一の値が返されます。 末尾に**.row(index)**を追加することで、特定の行の値を取得できます。"**Index**"は 0 から始まる行番号ですが、常に 1 つの値しか返しません。

数式ビルダのヘルプパネルの列と数式の両方に、各サブレポートのフォルダがあります。



サブレポートの列と数式を使用すると、より複雑な式やスクリプト（つまり、**SUB1.@Total+SUB2.@Total**）を作成できます。

サブレポートの列と数式は複雑な式で使用できますが、サブレポートの列と数式は集計関数（SUM、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

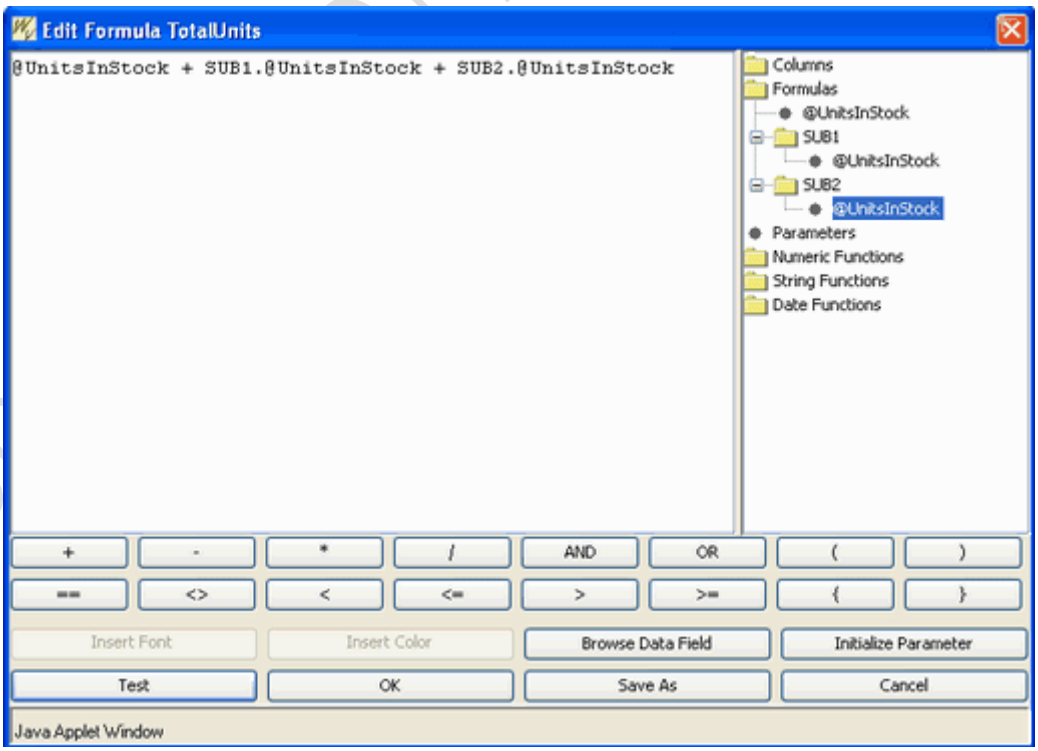
COUNT、MAX、AVERAGE など）では使用できません。たとえば、**SUM（{UnitPrice}）**は有効な式ですが、**SUM（SUB1.{UnitPrice}）**はそうではありません。これをするには、式 **SUB1.{UnitPrice}**を作成し、メインレポートに列として追加します。次に、最初の数式（つまり **SUM（@sub_total）**）に集計を適用する 2 番目の数式を作成します。

メインレポートはサブレポートの列やフォルダにアクセスできますが、サブレポートはメインレポートやその他のサブレポートの列や数式にアクセスできません。

メインレポートで使用されるサブレポートの式または列が削除された場合、レポートを実行するとメインレポートの式に NULL 値が表示され、スクリプトは適用されません。

たとえば、製品情報が複数のデータベースに分散されている場合、3 つのデータベース全体で在庫をすべて集計したい場合は、1 つのメインレポートを作成し、いくつかのサブレポートを追加し、レポートフッタにすべての在庫を追加するための数式を作成することができます。この機能を実証するには、カテゴリ名、製品 ID、製品名、単価、および在庫単位のフィールドを持つインベントリ XML ファイルから作成したレポートを開きます。Woodview Access データベースからのサブレポートと同じフィールドを含む Woodview HSQL データベースからのサブレポートを追加します。

各サブレポート内およびメインレポート自体内で、**UnitsInStock、SUM（{UnitsInStock}）**という式を作成します。次に、これら 3 つの関数を使用してメインレポート内に最終的な数式を作成することができます。



サブレポート式を使用した数式の作成

この数式は、3 つのレポートのそれぞれから数式値を取り、それらを合計して最終値を計算します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

この機能は、リンクされたサブレポートにも適用されますが、サブレポートの適用は異なります。たとえば、グループフッターにサブレポートを含む顧客アドレスを含むサマリブレークレポートを作成して、その特定の顧客が行ったすべての購入を詳述すると、メインレポート内のすべての売上情報を合計することができます。

最初に、サブレポートのすべての売上高を合計する **Sum_Sales** というサブレポート内に式を作成します。メインレポートに戻り、数式 **sub_total** を作成します。これは **SUB1.@Sum_Sales** です。それをテーブルデータセクションに挿入します。必要に応じて非表示にすることもできますが、テーブルデータセクションに配置する必要があります。その後、メインレポートの合計 **SUM（@sub_total）**に 2 番目の数式を作成します。この数式はレポートのフッターに追加され、すべてのサブレポートのすべての売上列が合計されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.9 スクリプト

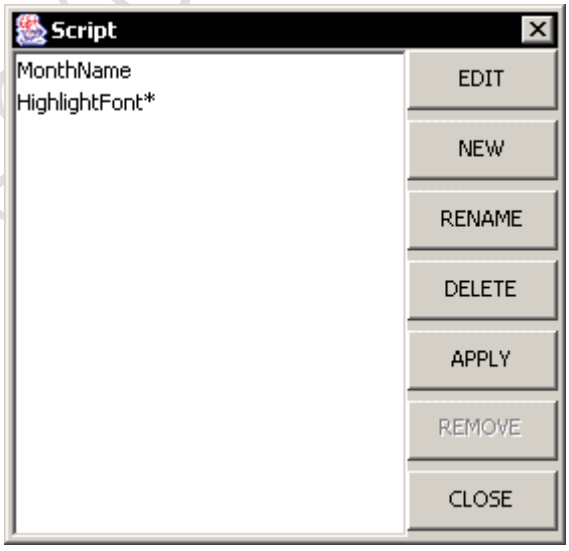
セルスクリプティングまたは条件付き書式設定では、特定の条件が満たされたときに特定のオブジェクトプロパティを動的に変更できます。スクリプトを使用すると、特定のデータ値を強調表示し、読みやすいレポートを表示できます。これの簡単な例は、財務実績やキャッシュフローを示すレポートです。列フィールドにスクリプトを適用すると、負の数が赤で表示され、収入や現金不足の領域が動的に強調表示されます。

EspressReport では、スクリプトは要素固有であり、レポートセルのプロパティを動的に変更するには、その特定のセルにスクリプトを適用する必要があります。スクリプトは、ラベル、列フィールド、数式、またはレポートセクションに適用できます。

EspressReport セルスクリプトの構文の一部は、v2.51 と v3.0 の間で大幅に変更されました。ほとんどの場合、非推奨の構文は引き続き正しく動作します。ただし、古いテンプレートをチェックして v2.0 以降に v3.0 以上にアップグレードした場合でもスクリプトが正しく機能することを確認することをお勧めします。

3.9.1 スクリプトの作成

スクリプトを作成するには、まずスクリプトを適用するセルを選択し、**Format** メニューから **Scripting** を選択するか、ツールバーの **Scripting** ボタン をクリックします。これにより、レポート内で定義されたすべてのスクリプトのリストが表示されます。



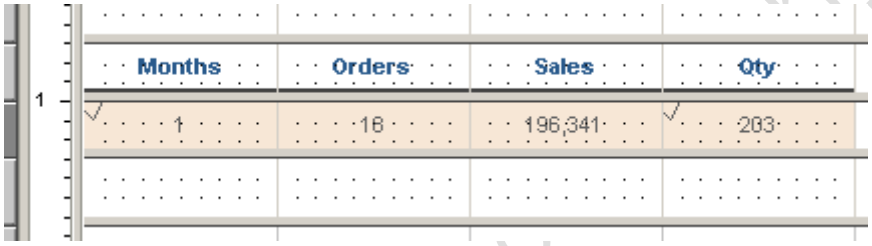
Script List ダイアログ

数式と同様に、スクリプトはレポート全体で維持され、レポートの複数の場所で再利用できます。ただし、式とは異なり、スクリプトはレポートに挿入できません。既存のレポート要素にのみ適用できます。リスト内のスクリプト名の横にある星印は、スクリプトが現在のセルに適用されていることを示します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

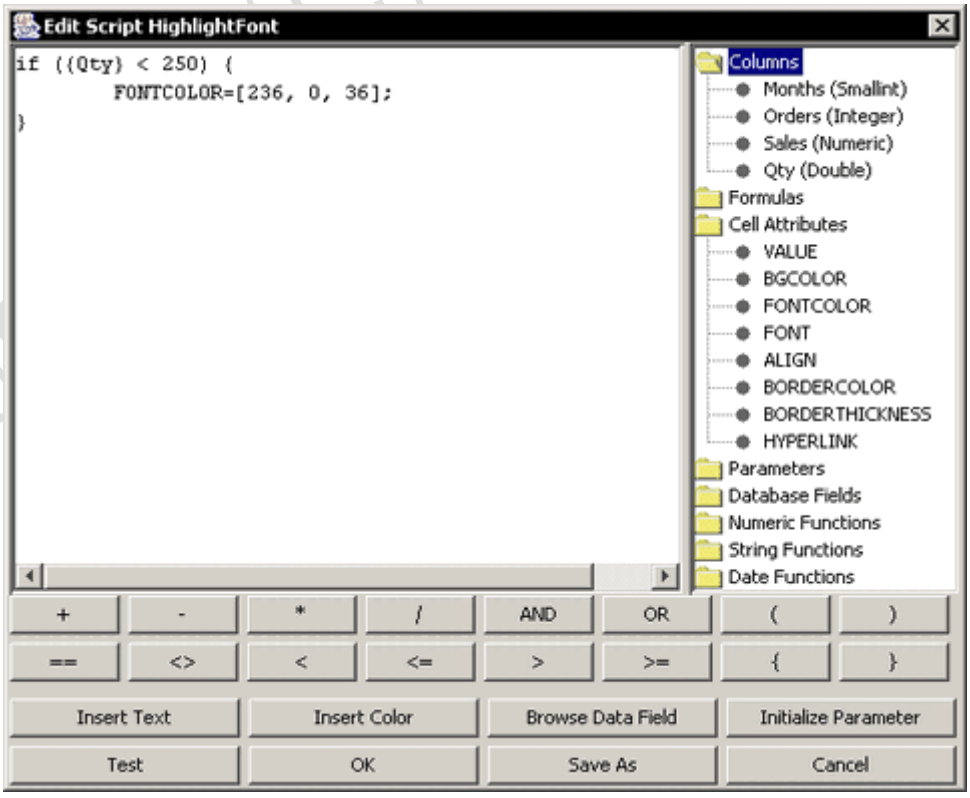
このダイアログでは、既存のスクリプトを編集、新しいスクリプトを作成、スクリプトの名前の変更、スクリプトの削除、現在の要素にスクリプトを適用、現在の要素からスクリプトを削除することができます。スクリプトが使用されている場合は、それを削除することができます（処理を続行するかどうかを尋ねるメッセージが表示されます)。スクリプトは削除される前にどのセルからも最初に削除されます。スクリプトがセキュリティレベルで使用されている場合は、スクリプトを手動で削除してから削除する必要があります。現在選択されているレポート要素にスクリプトを適用するには、リストからスクリプトを選択し、Apply ボタンをクリックします。スクリプトリストが閉じ、スクリプトが適用されます。

スクリプトが適用されているレポート要素のセルの左上隅に小さなチェックマークが付きます。スクリプトをプレビューまたはエクスポートするまで、スクリプトは実行されません。



デザインビューでのスクリプトインジケータ

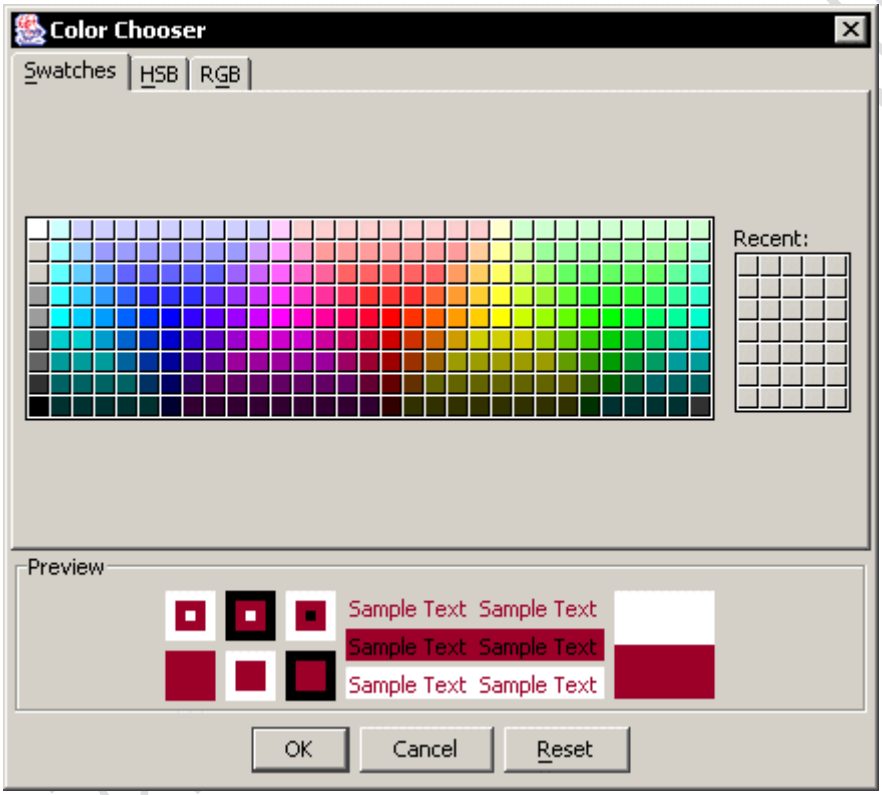
スクリプトリストダイアログから New ボタンをクリックして新しいスクリプトを作成することを選択すると、新しいスクリプトの名前を指定するよう求められます。名前を指定すると、式ビルダが開き、スクリプトを作成できます。



式ビルダ（スクリプト作成）

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

スクリプトは、数式と同じ数式ビルダインターフェースで開発されます。スクリプトは異なる構文を使用しますが、数式フィールド、演算子、および関数はすべてスクリプトでアクセス可能です。数式と数式ビルダの詳細については、数式および式ビルダの使用を参照してください。数式に使用できるオプションに加えて、スクリプトを作成する際には 2 つの追加オプションがあります。右側のパネルの **Cell Attributes** フォルダには、スクリプトで変更可能な(値に加えて)セル属性のリストが含まれています。属性とその使用については、アクションの書式設定で説明しています。**Insert Color** ボタンをクリックすると、セル属性の色を視覚的に選択するための色選択ダイアログが表示されます。色を選択すると、自動的に RGB 値の配列がスクリプトに挿入されます。



Choose Script Color ダイアログ

3.9.2 スクリプト構文

ほとんどのスクリプトの構文はかなりシンプルで、条件とアクションの 2 つの基本的なコンポーネントで構成されています。

if（Condition）{Action(s);}

Condition は一般に、ブール演算子で説明したブール演算子の 1 つを使用して導出されたブール式です。たとえば、({**Quantity**} < **5**) は、Quantity フィールドの値が 5 より小さい場合に **true** を返します。AND および OR 演算子を使用して、複数の条件をチェックできます。例えば、(({**Quantity**}<**5**) **AND** ({**InStock**}=="**Yes**")) は、Quantity フィールドの値が 5 より低く、InStock フィールドが**"Yes"**と等しい場合に **true** を返します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

したがって、負の値がレポートフィールドに表示されている場合、フォントの色を赤に変更する単純なスクリプトは、次のようになります。

```
if（{SubTotal}<0）{
    FONTCOLOR = [255,0,0];
}
```

if else 構文を使用して、複数のアクションに対して複数の条件を指定すると、より複雑なスクリプトを作成できます。

```
if（Condition 1）{Action 1;}
else if（Action 2）{<i>Action 2</i>;}
else{<i>Action 3</i>;}
```

指定可能なネストレベルの数には制限はありません。1 つの条件に対する複数のアクションも指定できます。

```
if（Condition 1）{
     Action 1;
     Action 2;
     Action 3;
}
```

これは、条件が満たされたときに複数のアクションを実行します。スクリプトの各行はセミコロンで終わらなければならないことに注意してください。

3.9.2.1 アクションの書式設定

以下は、スクリプトを使用して変更できる要素属性のリストです。

VALUE:

スクリプトが適用される要素の値を変更できるアクション構文は、**VALUE=New Value;**です。新しい値は、必ずしもレポート要素と同じデータ型である必要はありません（たとえば、number を string に置き換えることができます）。ただし、文字列を数値または日付に変更すると、書式の不規則性が発生する可能性があります。

BGCOLOR:

要素の背景色を変更できるアクション構文は、**BGCOLOR=[R、G、B];**です。ここで、R は新しい色の赤色値の番号、G は新しい色の緑色値の番号、B は新しい色の青色値の番号です。数式ビルダの **Insert Color** ボタンを使用して、RGB 配列を自動的に生成できます。

FONTCOLOR:

要素のフォント色を変更できるアクション構文は、**FONTCOLOR = [R、G、B];**です。ここで、R

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

は新しい色の赤色値の番号、G は新しい色の緑色値の番号、B は新しい色の青色値の番号です。数式ビルダの **Insert Color** ボタンを使用して、RGB 配列を自動的に生成できます。

FONT:

要素のフォント、フォントスタイル、およびフォントサイズを変更できるアクション構文は、**FONT=[Font Name,Font Style,Font Size];**です。フォント名は、使用するフォントの名前です（“Dialog”,“Serif”など）。フォントスタイルのオプションは、**BOLD**、**ITALIC**、または **BOLD+ITALIC** です。フォントサイズは、フォントサイズの数値表示です。たとえば、**FONT=[Dialog,BOLD,12];** フォントを 12 ポイントの太字のダイアログに変更します。

ALIGN:

この操作では、セル内のデータの水平方向の配置を変更できます。構文は **ALIGN=Position;**です。位置オプションは、**left, right, and center** です。

BORDERCOLOR:

要素の境界線の色を変更できるアクション構文は **BORDERCOLOR=[R、G、B];**です。ここで、R は新しい色の赤色値の番号、G は新しい色の緑色値の番号、B は新しい色の青色値の番号です。数式ビルダの **Insert Color** ボタンを使用して、RGB 配列を自動的に生成できます。

BORDERTHICKNESS:

要素の境界線の太さを変更できるアクション構文は、**BORDERTHICKNESS=Thickness;**です。厚さは画素数を示す整数です。たとえば、BORDERTHICKNESS=2;は境界線の太さを 2 ピクセルに設定します。

HYPERLINK:

要素のハイパーリンクプロパティを設定できるアクション構文は、**HYPERLINK=[URL,"Hint","Target"];**です。URL は、リンクが指し示す場所です。Hint はマウスポインタの上で表示されるヒントで、Target はリンクの指定されたターゲットです。ヒントまたはターゲットを指定しない場合は、空の文字列""を指定します。bat ファイルから Report Designer を起動した場合、プレビューウィンドウのハイパーリンクをクリックすることはできませんが、レポートを DHTML または PDF 形式にエクスポートするとリンクは引き続き機能します。

BOOKMARK:

この操作では、レポート内で名前付きの場所を指定できます。DHTML にエクスポートすると、アンカータグになります。PDF にエクスポートすると、生成された PDF ファイル内のブックマークになります。構文は **BOOKMARK=Name** です。BOOKMARK には任意の文字列値を使用できます。

XPOSITION:

セクション内のセルの X 位置を変更できるアクション構文は、**XPOSITION=Position;**です。(測定値に応じて、単位はインチまたはセンチメートルに変わります)。たとえば、**XPOSITION=1.2;**という構文はセクションの左端から 1.2 インチ/センチメートル離れたレポート要素の左端を設定し

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ます。

YPOSITION:

セクション内のセルの Y 位置を変更できるアクション構文は、**YPOSITION=Position;**です。(測定値に応じて、単位はインチまたはセンチメートルに変わります)。たとえば、**YPOSITION=0.3;**という構文はセクションの上部から 0.3 インチ/センチメートル離れたレポート要素の上端を設定します。

WIDTH:

レポート要素の幅を設定できるアクション構文は、**WIDTH=Size;**です。(測定値に応じて、単位はインチまたはセンチメートルに変わります)。たとえば、**WIDTH=1.5;**という構文は要素の幅を 1.5 インチ/センチメートルに設定します。

HEIGHT:

レポート要素の高さを設定できるアクション構文は、**HEIGHT=Size;**です。(測定値に応じて、単位はインチまたはセンチメートルに変わります)。例えば、**HEIGHT=0.25;**という構文は要素の高さを 0.25 インチ/センチメートルに設定します。

ROTATION:

レポート要素の回転を設定できるアクション構文は **ROTATION=Angle** です。(度で、負の値は反時計回りを示し、適用可能な値は-90,0 および 90 です)。たとえば、**ROTATION=-90;**という構文は要素の角度を反時計回りに 90 度に設定します。

VISIBLE:

この操作では、セルが表示されるかどうかを制御できます。構文は **VISIBLE=True/False;**です。visible を **false** に設定すると、レポート要素が非表示になります。

現在のセルのこれらのプロパティを変更することに加えて、これらのプロパティを他のレポート要素から取得することもできます。式の**{Column Field},@Formula,:Parameter** などと同じ方法で、または他のセルやラベルに id()関数を使用して、レポート要素から値を取得できます。詳細については、数式の使用を参照してください。

次の構文 **ATTRIBUTE（Report Element）**を使用して、他のすべての属性を取得できます。たとえば、**FONTCOLOR（id（RPT_HDR_LB0））**は、**Report Header** セクションのラベルのフォント色を表す整数配列**[R、G、B]**を返します。これを使用して、属性に基づいて比較を実行、また現在のセルの属性と別のレポート要素の属性を一致させることができます。たとえば、**BGCOLOR=BGCOLOR（{ProductName}）;**は現在のセルの背景色を **ProductName** 列の背景色に合わせて変更します。

スクリプトが適用されている要素のタイプに応じて、特定の書式設定アクションは影響を与えません。たとえば、線や矩形にスクリプトを設定できますが、境界や色を変更することしかできません。

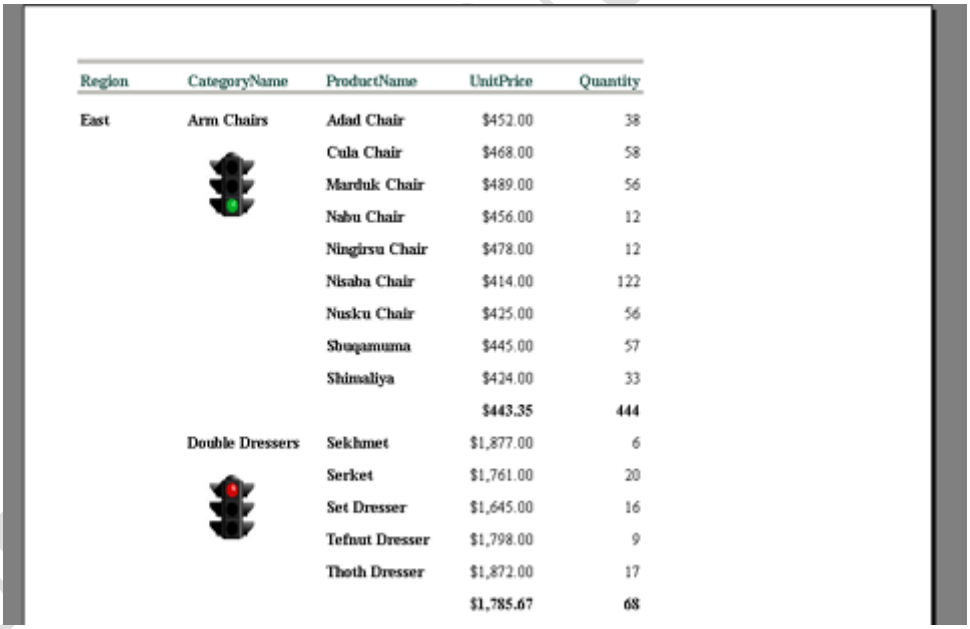
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.9.2.1.1 画像 URL のスクリプト

if else 条件と Image URL フォーマットオプションを使用すると、レポートのデータに基づいてイメージを動的に変更するイメージ URL スクリプトを作成できます。 読者が特定のカテゴリの売り上げが所望の目標を達成したかどうかを判断するのを助けるために二つの画像を含めたいとします。また、各カテゴリの販売目標が 100 単位であるとします。この情報を使用すると、目標が達成されたときに成功イメージを表示し、目標が達成されなかった場合に失敗したイメージを表示する次のスクリプトを書くことができます。

```
if(@SUM_Quantity > 100){
      value="http://www.quadbase.com/images/green.jpg";
} else {
      value="http://www.quadbase.com/images/red.jpg";
}
```

文字列データ型を持つ新しい数式セルを作成します（コンテンツには""を使用できます)。このセルにスクリプトを適用します。次に、このセルのデータ書式を「Image Url」に設定します。データ書式の詳細については、数式と列フィールドのデータ書式設定を参照してください。このスクリプトを使用したレポートの表示方法は次のとおりです。



Region	CategoryName	ProductName	UnitPrice	Quantity
East	Arm Chairs	Adad Chair	$452.00	38
		Cula Chair	$468.00	58
		Marduk Chair	$489.00	56
		Nabu Chair	$456.00	12
		Ningirsu Chair	$478.00	12
		Nisaba Chair	$414.00	122
		Nusku Chair	$425.00	56
		Shugamuma	$445.00	57
		Shimaliya	$424.00	33
			$443.35	444
	Double Dressers	Sekhmet	$1,877.00	6
		Serket	$1,761.00	20
		Set Dresser	$1,645.00	16
		Tefnut Dresser	$1,798.00	9
		Thoth Dresser	$1,872.00	17
			$1,785.67	68

スクリプト化された画像の URL

3.9.2.2 変数と配列

if else ステートメントに加えて、スクリプト内で変数と配列を定義することもできます。変数と配列は、スクリプトの先頭（任意の文の前）に定義されます。

3.9.2.2.1 変数

スクリプト内の複数の場所で使用する定数または式がある場合は、定数または式を使用する場所ごとに再入力するのではなく、変数を定義する方が簡単な場合があります。変数は、スクリプトの先頭で構文

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

DataType VariableName;を使用して宣言できます。データ型は変数値の型で、Number、Boolean、String、または Date のいずれかです。

DateVar は、Date 変数のデータ型宣言として使用されます。これにより、date()関数との混同を避けることができます。変数名は、演算子または関数名以外の任意の名前にすることができます。

次の例では、式を含む変数を使用しています。

```
String Line1;
String Line2;
Line1 = {Address1} + "¥n";
Line2 = {City} + "," + {State} + " " + {Zip};

if ({Address2} == NULL) {

      VALUE = Line1 + Line2;
} else VALUE=Line1 + {Address2} + "¥n" + Line2;
```

このスクリプトは、複数のレポートフィールドをまとめてアドレスエントリを作成します。スクリプトの最初の 2 行は変数を宣言します。スクリプトは、Address2 エントリがあるかどうかによって、フィールドと文字列を連結します。変数を使用すると、式を再入力する必要がなくなります。"¥n"は、テキストを改行するために使用されます。

スクリプト内のアクションの一部として変数に値を割り当てることもできます。これを使用して、いくつかの条件に基づいて変数値を変更し、変数値をセルに戻すことができます。これは、1 つの value ステートメントだけを書きたい場合に便利です。次の例では、変数に値を割り当てます。

```
if ({Flags} == "M"){
      LinkAppend = "Main";
}else if({Flags} == "R"){
      LinkAppend = "Remote";
} HYPERLINK = ["StatPage.html#" + LinkAppend,"",""];
```

この例では、変数は初期値なしで定義されています。変数の値は、レポートの文字列フラグに基づいて割り当てられます。スクリプトの最後に、変数値を使用してページの場所を指定する要素のハイパーリンクが定義されます。

3.9.2.2.2 配列

変数の他に、スクリプトの先頭に配列を宣言することもできます。後で配列の特定の値を取得できます。これは、月や曜日など、スクリプト内のさまざまな場所で使用する固定値の数が多い場合に便利です。配列宣言は変数宣言に似ており、次の構文を使用します:**DataType[] ArrayName = [Value1, Value2, Value3, ...];**。配列のデータ型は変数と同じで、Number、Boolean、String、または Date のいずれかに

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

なります。変数名は、演算子または関数名以外の任意の名前にすることができます。初期値の割り当てはオプションです。

次の構文を使用して、配列から値を取得できます。ArrayName [Index]。次の例では、配列を使用して月の列（1～12）の数値を置き換え、適切な月の名前に置き換えます。

```
String[] MonthNames = ["January","February","March","April",
"May","June","July","August","September","October",
"November","December"];

VALUE = MonthNames[({Month} - 1)];
```

指定されたインデックスは**{Month}-1** です。これは、最初の月の値が 1 で、配列の最初のインデックス値が 0 であるためです。

変数のように、配列の値をアクションの一部として割り当てたり上書きしたりすることもできます。この構文は **ArrayName [Index]=Value;**です。

3.9.2.3 数式の使用

スクリプトの条件とアクションの両方のコンポーネントと、変数値の両方でメインレポートまたはサブレポートの EspressReport 式を使用することができます（サブレポートの式へのアクセスの詳細については、サブレポートの数式アクセスを参照してください）。

最も一般的な式は、現在の要素または他のレポート要素のいずれかから値を取得するために使用されます。**@ColumnName** を使用して既存のレポート式の値である**{Column Field}**と、それぞれ**ParamName** と **? ParamName** を使用してクエリと数式パラメータにユーザが入力した値を使用して、列フィールドから値を取得できます。

これらのメソッドに加えて、スクリプトで使用できる追加の関数が 1 つあります。

This() -

この関数は、現在の要素（つまり、スクリプトが適用されているセル）の値を返します。構文は引数なしの **this()**です。この関数は、スクリプトを再利用する場合や、スクリプトが適用されているセルを参照していることを確認したい場合に便利です。たとえば、**If(this()<0){FONTCOLOR = [255,0,0];}**がフォントの色を赤にする場合、現在のセルの値は 0 より小さいということになります。

数式を使用して、比較や要素の値を変更する式を作成することもできます。EspressReport の式とその構文の詳細については、数式および式ビルダの使用を参照してください。

3.9.2.4 Null チェック

セルスクリプトを使用すると、レポート列内の NULL 値をチェックできます。 定数として null を指定

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

するには、単にデリミタなしで null という単語を使用します。 したがって、Null をチェックする簡単なスクリプトは次のようになります。

```
if (this() == null) {
    VALUE="yes";
}
```

Report Designer で null データハンドラを設定した場合でも、この方法でスクリプトを使用できます。これは、null ハンドラがセルの内容を変換する機会を得る前にスクリプトが実行されるためです。

3.9.2.5 行固有のオプション

スクリプトを使用すると、列フィールドの特定の行の値を取得および変更できます。通常、列フィールドの値を取得または置換する場合、スクリプトが Table Data セクションのフィールドに適用されている場合は、列または現行の行の最初の値が取得または置換されます。ただし、取得/置換する行を指定するオプションもあります。これを行うには、構文**.row(Row Index)**を持つ列フィールドを追加できます。行インデックスは、レポートの各行に関連付けられたインデックス値です。たとえば、**{ProductName}.row(3)**は ProductName フィールドの 4 番目の値を返します。

getRowIndex()関数と getTotalRowIndex()関数をそれぞれ使用して、現在の行インデックスとレポート内の行の総数を取得できます。次の例では、この機能を使用して、列フィールドの繰り返し値を非表示にします。

```
if (({Category}.row(getRowIndex()) == {Category}.row(getRowIndex()-1)) AND (getRowIndex()
<> 0)) {
    FONTCOLOR=[255,255,255];
}
```

この例では、レポートの背景色が白で、それに応じてフォントの色を調整してフィールドを非表示にすることを想定しています。

3.9.2.6 ループ

条件付きの else 文に加えて、EspressReport ではスクリプトにループを追加することもできます。変数と行固有のオプション/書式設定を組み合わせて使用する際、ループは、レポートの現在の行またはセクションとは独立して、実行中の合計/計算を実行する必要がある場合に便利です。ループの基本的な構文は次のとおりです。

while (Condition) {Action(s);}

指定されたアクションは、条件（ブール式）が真である限り実行されます。

たとえば、次のレポートがあるとします。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Product	Quantity	Price
Chair	6	$327.00
Chair	4	$218.00
Chair	12	$418.00
Chair	5	$221.00
Chair	18	$248.00
Table	4	$875.00
Table	2	$1.024.00
Table	8	$967.00
Table	7	$1.106.00

テーブルフッター(**sum({Quantity})**)に合計数量を計算する数式を追加すると、66 が返されます。ただし、テーブルフッターに椅子の合計数量を表示したかったとします。これを行うには、ループを作成します。次のスクリプトを集約に適用します。

```
Number i;
Number s;
i = 0;
s = 0;

while (i < getTotalRowIndex()) {

    if ({Product}.row(i) == "Chair") {

        s = s + {Quantity}.row(i);

    }

    i = i + 1;

}

VALUE = s;
```

Chair の数量欄の合計 45 を返します。スクリプトはレポートの各行をループし、Product 列の値が"Chair"であるかどうかを確認します。そうである場合、その行の数量値が変数 s に追加されます。

3.9.2.7 セクションスクリプト

個々のセルに加えて、スクリプトをレポートセクションに適用することもできます。レポートデータ内の条件に基づいて、個々のセクションオプションをオン/オフにすることができ、セクションの背景色

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

を設定することができます。レポートセクションにスクリプトを追加するには、デザイナの左側にあるセクション名のボタンをクリックします。また、セクションフィールドの空白部分を右クリックし、ポップアップメニューから **Scripting** を選択することもできます。

セクションスクリプトの記述は、同じ構文を使用するレポート要素用のスクリプトの記述とまったく同じです。唯一の違いは、数式ビルダの **Cell Attributes** フォルダの代わりに、セクションスクリプトで実行できるさまざまな書式設定アクションを含む **Section Attributes** フォルダが表示されることです。

3.9.2.7.1 セクションスクリプトのパラメータの形式

以下は、スクリプトを使用して変更できるセクション属性のリストです（すべてのオプションがすべてのレポートセクションで使用できるわけではありません）。

BGCOLOR:

要素の背景色を変更できるアクション構文は、**BGCOLOR = [R、G、B];**です。ここで、R は新しい色の赤色値の番号、G は新しい色の緑色値の番号、B は新しい色の青色値の番号です。数式ビルダの **Insert Color** ボタンを使用して、RGB 配列を自動的に生成できます。

PRINT_ON_NEW_PAGE:

この操作により、新しいページセクションの印刷オプションをオン/オフすることができます。構文は **PRINT_ON_NEW_PAGE = True/False;**です。新しいページの印刷機能の詳細については、セクションオプションを参照してください。

RESET_PAGE_NUMBER:

この操作により、リセットページ番号セクションオプションをオン/オフすることができます。構文は **RESET_PAGE_NUMBER = True/False;**です。リセットページ番号機能の詳細については、セクションオプションを参照してください。

VISIBLE:

このアクションでは、セクションが表示されるかどうかを制御できます。構文は **VISIBLE=True/False;**です。visible を false に設定すると、セクションが非表示になります。ネストされたセクション機能（ネストされるセクションを参照）と組み合わせたこの属性を使用すると、データに応じて表示する異なるヘッダー/フッターを作成できます。

HEIGHT:

このアクションでは、セクションの高さを設定できます。構文は **HEIGHT = Size;**です。（測定値に応じて、単位はインチまたはセンチメートルに変わります）。例えば、**HEIGHT = 0.25;**という構文はセクションの高さを 0.25 インチ/センチメートルに設定します。

セクション属性を設定または参照するスクリプトは、セクションにのみ適用できます。他のレポート要素には適用できません。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.9.2.8 スクリプトのコメント

スクリプトにコメントを追加して、その設計と操作を説明することができます。
コメントの前に 2 つのスラッシュがあります。これと同じ行に続くものはコメントとして扱われます。
コメントは複数の行に分割できますが、各行は//で始まる必要があります。

3.9.2.9 他のスクリプトで計算された値へのアクセス

レポート内のセルにスクリプトを適用している場合は.getScriptValue()構文を使用してこれらの値にアクセスできます。たとえば、次のデータがあります。

```
String, int
Month, Revenue
January, 3567
"February", 6854
"March", 5421
"April", 8425
"May", 5611
"June", 2356
"July", 6543
"August", 2563
"September", 4432
"October", 8841
"November", 1023
"December", 2265
```

このデータを使用して年間収益の列を作成する場合は、次のスクリプトを使用します。このスクリプトは、前の行のスクリプト値を現在の行の収益に加算して表示します。このスクリプトは、テーブルデータセクションのカラムとして数字 0 で構成される式に置かれるように書かれています。このスクリプト内では**{formula1}**と呼ばれます。

```
if(getRowIndex()==0)

    VALUE={revenue};
else
    VALUE={revenue} + {formula1}.getScriptValue(getRowIndex()-1);
```

これにより、次のようなレポートが生成されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Month	Revenue	Year to Date
January	3,567	3,567
February	6,854	10,421
March	5,421	15,842
April	8,425	24,267
May	5,611	29,878
June	2,356	32,234
July	6,543	38,777
August	2,563	41,340
September	4,432	45,772
October	8,841	54,613
November	1,023	55,636
December	2,265	57,901

3.9.3 スクリプトの管理

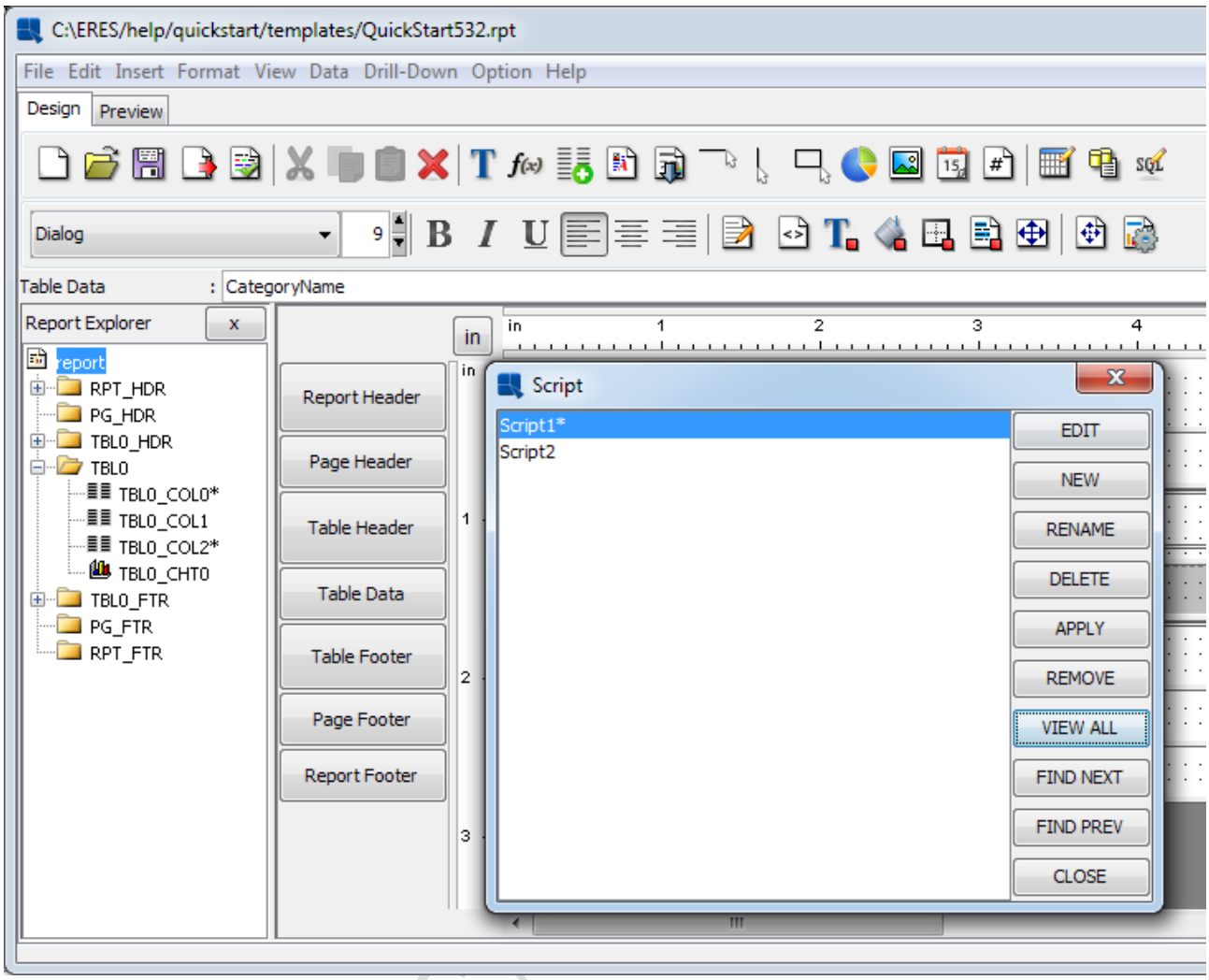
いくつかの要素に適用されるいくつかのスクリプトまたはスクリプトを含むレポートの場合、スクリプトが適用される要素の管理および検索に役立つオプションがあります。スクリプトダイアログでは、削除、削除、名前の変更などの標準オプションに加えて、すべてを表示、次を検索、前の要素を検索するボタンもあります。これらのボタンを使用する前にスクリプトを選択してください。

VIEW ALL:

このボタンは Report Explorer が開いていない場合開き、選択したスクリプトにアスタリスクが適用されているすべての要素を強調表示します。ツリーは、セルが強調表示されているセクションを自動的に展開します。これらの要素の１つをクリックすると、デザイナビューがその要素に集中し、適用されたスクリプトを変更または削除できます。別のスクリプトを選択してもう一度 VIEW をクリックすると、ハイライトが変更されます。スクリプトダイアログボックスを閉じると、すべてのハイライトが削除され、エクスプローラツリーが折りたたまれます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



View All の結果

FIND NEXT and FIND PREVIOUS

これらのボタンは、選択したスクリプトが適用された要素を順番に表示するために使用されます。これらのボタンをクリックしたときにデザイナですでに要素が選択されている場合は、現在の要素から検索が開始されます。要素が選択されていない場合は、レポートの先頭で検索が開始されます。別のスクリプトを選択して FIND NEXT をクリックすると、アスタリスクが新しく選択されたスクリプトに移動し、スクリプトが適用された最初の要素がデザイナーで強調表示されます。**FIND NEXT** と **FIND PREVIOUS** は、表示されている要素を一巡するだけです。スクリプトで目に見えない要素を表示するには、**VIEW ALL** を使用します。これはすべての非表示要素と表示要素を表示します。

レポートエクスプローラの詳細については、レポートエクスプローラを参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.10 ドリルダウン

多くの場合、レポートは大量のデータを表示するために使用されます。しかし、大量のデータセット（特にデータがかなり詳細な場合）では、すべての情報を 1 つのレポートに表示することは、情報を表示する最善の方法ではない可能性があります。この問題の 1 つの解決策は、集計されたデータのみを表示し、ユーザがクリックして基礎となるデータを表示できるようにするトップレベルのレポートを作成することです。これはドリルダウンの概念です。

EspressReport では、ユーザはこの方法で簡単にデータを表示できます。プライマリ（最上位）レポートの列フィールドにセカンダリ（サブレベル）レポートへのリンクを設定できます。サブレベルのレポートでは、プライマリレポートの値を入力として受け入れるパラメータ化されたクエリを使用します。したがって、サブレベルのレポートには、ユーザがクリックした特定の値に関連するデータのみが表示されます。

たとえば、1 年分の売上データを表示するレポートを設計するとします。1 年分のデータを表示するレポートを作成するのではなく、集計されたデータを月ごとに表示する最上位レポートを設計できます。

Month	Units Sold	Total Sales
January	5	$10,224.12
February	4	$12,286.14
March	6	$14,415.16
April	5	$12,316.11
May	2	$9,482.64
June	3	$10,116.12
July	7	$17,624.18
August	5	$10,361.54
September	4	$10,116.82
October	3	$8,612.41
November	5	$10,552.24
December	11	$22,614.89

レポートを表示しているユーザは、ドリルダウンして毎月のデータをさらに見ることができます。この場合、ユーザが March をクリックしたとします。その月の詳細な販売情報を提供する基礎となるレポートに移動します。

Sales Data For March 2001				
Order #	Customer	Product	Quantity	Total
10042	Allied Furniture Emporium	Chair	2	$6,425.16

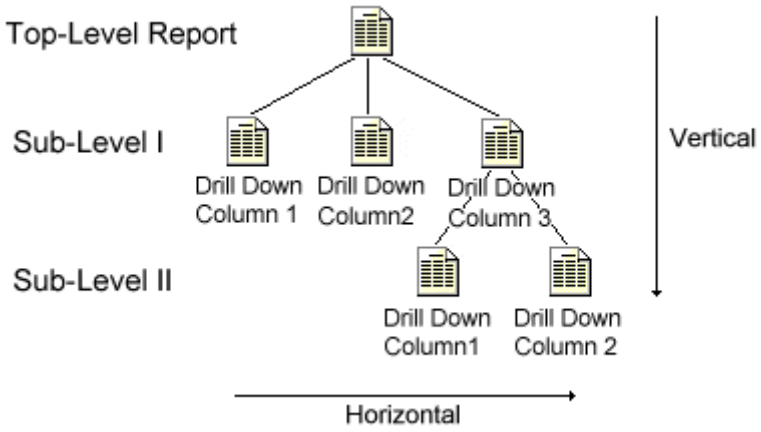
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

10051	Furniture World	Table	1	$3,114.12
		Dresser	1	$2,116.83
10068	Domus Home & Garden	Chair	1	$800.65
		Dresser	1	$1,958.40
Total:			6	$14,415.16

このレポートから、ドリルダウンの別のレイヤーを追加することができ、ユーザはより詳細な顧客情報を参照することができます。この場合、ユーザが Allied Furniture Emporium をクリックしたとします。彼らは、顧客に関するより多くの情報を示す第 3 のレポートに進む。

Customer Information		
		Address:
Company:	Allied Furniture Emporium	384 Broad St. Littletown, NY 18322
Contact Name:	Matilda Gladwaller	
Order History:		
Year	Orders	Total Sales
1999	14	$52,614.18
2000	18	$68,115.89
2001	12	$38,812.12

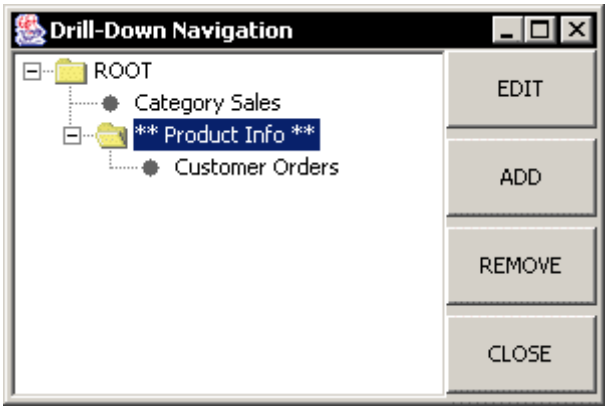
このようにドリルダウンを使用すると、大量の情報を理解してナビゲートする方法で提示することができます。EspressReport では、ドリルダウンの水平と垂直のレイヤーが可能です。これは、同じレポート内の異なる列がドリルダウン（水平）することができること、またはサブレベルのレポートが後続のドリルダウンレポート（垂直）を持つことができることを意味します。サブレポートにドリルダウンレイヤを追加することもできます。



Report Designer では、ドリルダウンはドリルダウンメニューから制御されます。このメニューから **Navigate** オプションを選択すると、ナビゲーションウィンドウが表示されます。このウィンドウから、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ドリルダウンレポートを追加および編集できます。



Drill-Down Navigation ダイアログ

ナビゲーションウィンドウの左側には、ドリルダウンレポートの階層が表示されます（上の図を参照）。ROOT はトップレベルのレポートです。現在編集中のレベルには「**」と表示されます。別のレポートを編集するには、そのレポートを選択し、右側の **EDIT** ボタンをクリックします。そのレポートは Designer で開き、カスタマイズすることができます。レポートを選択し、**REMOVE** ボタンをクリックすると、レポートを削除することもできます。

3.10.1 ドリルダウンレポートの作成

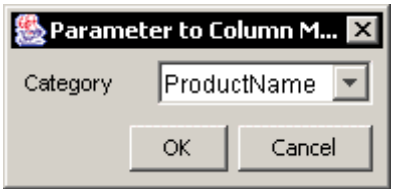
ドリルダウンレポートを作成する最初の手順は、トップレベルレポートを作成することです。ドリルダウンはデータセット間の関係に依存するため、ドリルダウンレイヤにはパラメータ化されたクエリ、XML クエリ、またはクラスファイルを使用するレポートのみを使用できます。最上位レベルのレポートを設計するときは、少なくとも 1 つの列がサブレベルのレポートにパラメータとして渡されることに注意してください。最上位レベルのレポートの設計が完了したら、**Drill-Down** メニューから **Navigate** を選択します。ナビゲーションウィンドウが開きます。ドリルダウンレイヤを追加するレポートを選択します（レポートにドリルダウンがない場合、ROOT ノードのみが表示されます）。**ADD** ボタンをクリックします。サブレポートにドリルダウンを追加する場合は、まずサブレポートタブに行き、ナビゲーションダイアログをロードします。

新しいレポートを作成するか、ドリルダウンレイヤに既存のレポートを使用するかどうかを確認するメッセージが表示されます。既存のレポートを使用できます。ドリルダウン層のレポートには、データソースとしてパラメータ化されたクエリまたはクラスファイルが必要です。既存のレポートに独自のドリルダウンレベルが含まれている場合は、すべてのドリルダウンレベルもインポートされます。新しいレポートの作成を選択すると、レポートウィザードが開き、データソース、レポートの種類、および列のマッピングを選択できます。

次のステップでは、トップレベルレポートの列をサブレベルレポートのパラメータにマップします。ドリルダウン層の既存のレポートを選択したとき、またはドリルダウン層の新しいレポートのデータソースを選択したときにこれを行うように求められます。いずれの場合も、ダイアログボックスが表示され、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

列をマップするよう求められます。



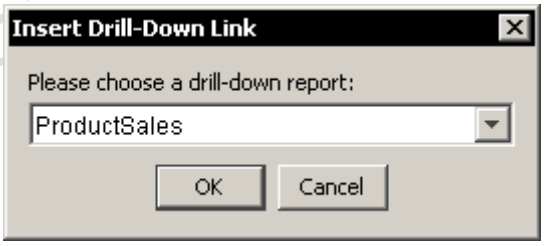
Parameter-Column マッピングウィンドウ

ドロップダウンメニューで使用できるオプションは、データタイプに基づいています。たとえば、パラメータ化されたクエリで文字列をパラメータとして使用する場合、文字列データを含む列のみをマップできます。サブレベルのクエリに複数のパラメータがある場合、None オプションもあります。これにより、マップされていないドリルダウンレポートを作成することができます。詳細は、マップされていないドリルダウンを参照してください。ドリルダウンマッピングを指定すると、サブレベルレポートの名前を指定するよう求められます。これで、サブレベルレポートがデザインウィンドウに表示され、書式を設定してカスタマイズすることができます。

3.10.1.1 ドリルダウンリンク

デフォルトでは、サブレベルレポートにマップされているトップレベルレポートの列にリンクが配置されます。デフォルトのリンクを使用する代わりに、ドリルダウンメニューのドリルダウンリンクオプションを使用して、リンクを別の列または複数の列に配置することができます。

レポート列からドリルダウンリンクを追加または削除するには、まず、現在正しいテンプレート（上位または上位レベル）を編集していることを確認します。次に、列フィールドを選択し、ドリルダウンメニューからドリルダウンリンクを選択します。これにより、リンク先のサブレベルレポートを選択できるドロップダウンリストが表示されます。ドリルダウンリンクを削除するには、ドロップダウンリストの上部にある空白のオプションを選択します。



Drill-Down Link ダイアログ

3.10.1.1.1 チャートからリンク

列にリンクを配置するほかに、チャートにドリルダウンリンクを設定することもできます。この機能を使用すると、ユーザはチャート内のデータポイントをクリックし、次のレベルのドリルダウンに移動できます。チャートにドリルダウンリンクを設定するには、Report Designer でチャートオブジェクトを選択し、ドリルダウンメニューからドリルダウンリンクを選択します。次に、列の場合と同じようにサブレベルレポートを選択します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

チャートの要素とサブレベルレポートのパラメータの数に応じてチャートにドリルダウンリンクを設定すると、選択されたデータポイントのカテゴリ、データ系列、and/or は合計値はパラメータ値としてサブレベルレポートに渡されます。この機能を正しく機能させるには、チャートでレポートと同じデータソースを使用する必要があります。また、チャートには、サブレベルレポートのパラメータと一致するように選択された列にカテゴリがマップされている必要があります。グラフの列マッピングの詳細については、基本的データマッピングおよびチャートの種類ごとのデータマッピングを参照してください。

任意のチャートタイプからリンクを設定できます。ただし、PDF にエクスポートすると、すべてのチャートタイプが機能するわけではありません。2 次元バー、カラム、スタックバー、スタックカラム、100%カラムチャートのみが PDF でリンクを生成します。PDF エクスポートは矩形のイメージマップのみをサポートしています。

3.10.2 複数値ドリルダウン

EspressReport は、一度に 1 つのフィールドでドリルするだけでなく、複数値のドリルダウンという概念もサポートしています。この機能を使用すると、一度にトップレベルのレポートからいくつかの異なる値にドリルダウンできます。

たとえば、100 人の異なる顧客の販売情報を詳述するレポートがあるとします。マルチバリュードリルダウンを使用することで、ユーザはそれぞれの顧客を個別に見るのではなく、一番興味のある顧客を選択して一度にドリルダウンすることができます。

3.10.2.1 複数値ドリルダウンレポートの作成

レポートに複数値ドリルダウンのレイヤーを追加することは、通常のドリルダウンレイヤを追加することとほぼ同じです。唯一の違いは、下位レベルのレポートのクエリです。複数値のドリルダウンレイヤを作成するには、下位レベルのレポートに、上位レベルのレポートの列にマップされた複数値のパラメータがクエリに含まれている必要があります。

前の例を使用すると、下位レベルのクエリには、顧客 ID のための複数値のパラメータがあります(Where Customers.CustomerID IN（：CustID）など)。CustID パラメータが最上位レポートの Customer ID 列にマップされている場合、これにより自動的に複数値のドリルダウンレイヤが作成されます。複数値パラメータの詳細は、複数値パラメータを参照してください。複数の値のドリルダウンレイヤを作成しない場合は、下位レベルのレポートで単一の値パラメータを使用します。

複数値のドリルダウンを使用する場合は、レベルごとに 1 つのレポートしか作成できません。同じレポートの異なる列を異なるレポートにリンクすることはサポートされていません。

3.10.3 クロス集計ドリルダウン

EspressReport は、トップレベルのレポートがクロス集計レポートの場合に、ドリルダウン機能の特別

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

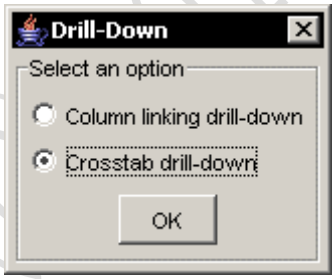
な実装を提供します。クロス集計ドリルダウン機能を使用すると、クロス集計マトリックス内のセルをクリックして、特定の行または列の情報をドリルダウンできます。たとえば、製品と地域別に販売数量を示す次のレポートがあるとします。

	East	South	Midwest	West
Chair	144	208	131	108
Cabinet	208	114	158	206
Dresser	100	101	112	109

標準のドリルダウン構成を使用して、ユーザがハイパーリンクされたセルの 1 つをクリックすると、その行の値のみが下位レベルのレポートに渡されます。ただし、クロス集計ドリルダウンでは、そのセルの列ブレーク値と行ブレーク値が下位レベルのレポートに渡されます。したがって、ユーザがセル値 144 をクリックすると、値「Chair」および「East」が下位レベルレポートのパラメータに渡されます。

3.10.3.1 クロス集計ドリルダウンレポートの作成

ドリルダウンレイヤをクロス集計レポートに追加することを選択すると、標準（列リンク）ドリルダウンプレゼンテーションを作成するのか、クロス集計ドリルダウンを使用するのかを確認するメッセージが表示されます。



Crosstab Drill-Down オプション

クロスタブドリルダウンを使用するように選択すると、他のドリルダウンプレゼンテーションの場合と同様にセットアップが続行されます。唯一の違いはパラメータマッピングにあります。下位レベルのレポートのパラメータをメインレポートの列にマッピングするのではなく、最上位レベルのレポートの行ブレークと列ブレークの値にマップします。

Crosstab Drill-Down の Parameter-Column マッピング

ドリルダウンレイヤを追加すると、トップレベルレポートのクロス集計マトリックスのセル（列ブレーク値）が、追加したレイヤーに自動的にリンクされます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.10.4 ドリルダウンレポートの表示

レポートビューアアプレット（およびプレビューウィンドウ）内のドリルダウンレポートを直接ナビゲートして表示することができます。ドリルダウンを含むレポートを開くと、ドリルダウンリンクを持つ列の任意のエントリをクリックできます。サブレベルレポートが表示され、その行の値がパラメータとして使用されます。

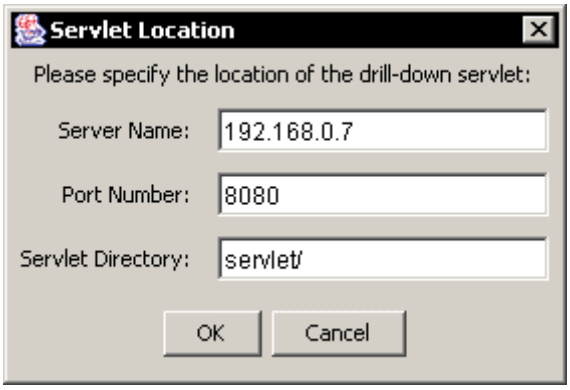
複数値のドリルダウンレポートの場合、**Preview** ウィンドウの **Data** メニューまたは **Viewer** ポップアップメニューから **Select Multiple Drill-Down Values** を選択できます。これにより、マップされた列のすべての値を含むダイアログが表示されます。次に、必要な値を選択し、**OK** ボタンをクリックして次のレベルに進みます。

Multi-Value Drill-Down Selection ダイアログ

サブレベルレポートのテンプレートは、DrillDown ディレクトリに保存されます。別のマシン上でレポートを表示している場合、レポートビューアはそのディレクトリ内のテンプレートを検索するため、移動する必要があります。

ドリルダウンレポートは任意の形式にエクスポートできます。ただし、DHTML および PDF 形式のみがドリルダウン機能を保持できます（複数値のドリルダウンレポートでは DHTML のみがサポートされます）。その他の形式では、トップレベルレポートのみがエクスポートされます。DHTML と PDF のエクスポートは、インストールの **DrillDownLinkGenerator** ディレクトリにあるドリルダウンレポートサーブレットと連携して機能します。レポートをエクスポートする前に、サーブレットコードをコンパイルし、アプリケーションサーバ/サーブレットランナー内にクラスファイルをデプロイする必要があります。Designer から DHTML/PDF にエクスポートすると、ドリルダウンサーブレットの場所を指定するダイアログが表示されます。

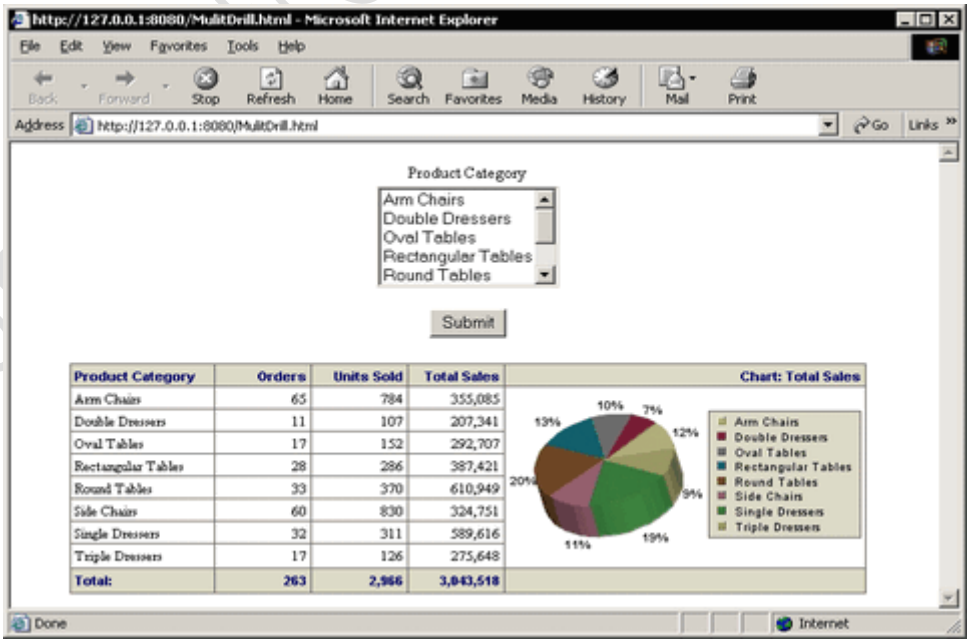
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Drill-Down Servlet ダイアログ

最初のオプションは、サーバの名前の入力を求めます。マシン名（**machine.domain.com** など）またはアプリケーションサーバ/サーブレットランナーの IP アドレスを入力することができます。2 番目のオプションは、アプリケーションサーバ/サーブレットランナーが使用しているポート番号を指定するように求めます。3 番目のオプションは、アプリケーションサーバ/サーブレットランナーのサーブレットディレクトリを指定するように要求します。エクスポートすると、この情報はドリルダウン列のリンクを生成するために使用されます。

複数値のドリルダウンレポートの場合、DHTML エクスポートは、ユーザがドリルインする値を選択できるようにするフォームを上部に作成します。複数値のドリルダウンレポートをエクスポートする場合は、同じサーブレット情報を指定する必要があります。ただし、ドリルダウンレポートサーブレットを使用するのではなく、インストールの **ParamReportGenerator** ディレクトリにあるパラメータレポートサーブレットを使用します。このサーブレットは、これらのレポートが正しく機能するように、アプリケーションサーバ/サーブレットランナー内でコンパイルおよびデプロイする必要があります。



DHTML の Multi-Value Drill-Down

ドリルダウンレポートの実行の詳細は、サブレポート、チャート、ドリルダウンレポートを参照してく

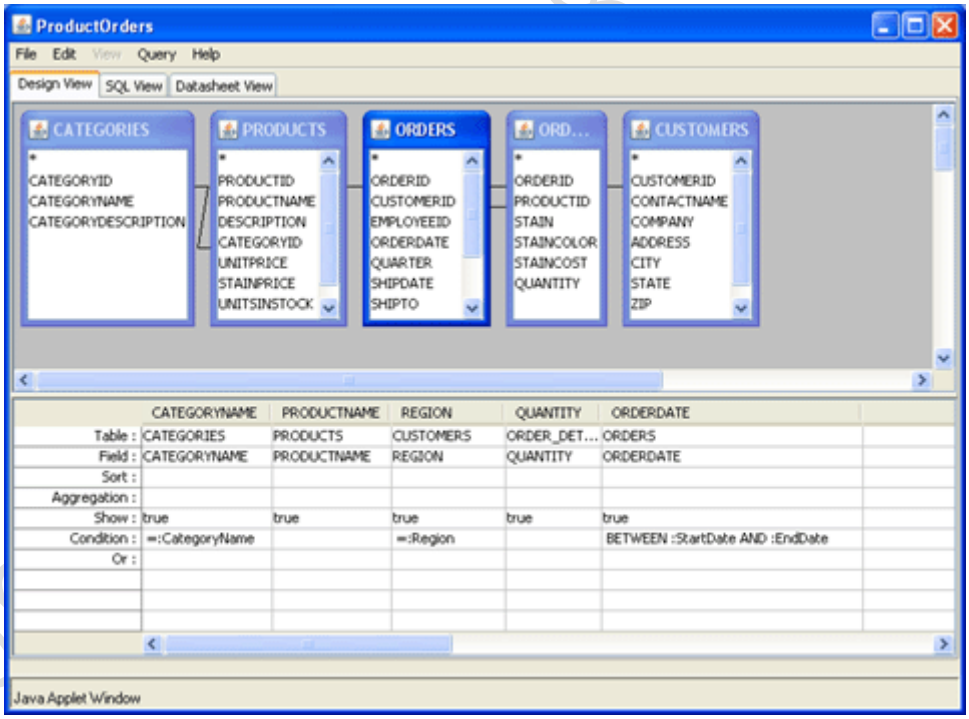
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ださい。

3.10.5 ドリルダウンのマッピング解除

複数のパラメータを使用してサブレベルのクエリを作成する場合、これらのパラメータの一部をマップしないままにしておくと、ドリルダウンリンクをクリックした後にプロンプトを表示することができます。パラメータのマッピングを選択するときにこの機能を使用するには、マップされないままにするパラメータに **none** を選択します。サブレベルレポートには、ルートレベルからドリルダウンするときにパラメータプロンプトが表示されるかどうかを決定するデータメニューオプションプレビューパラメータプロンプトも含まれています。このレベルが以前にプレビューされたことがない場合、このレベルはオフにされ、デフォルトのパラメータが使用されます。それ以外の場合は、後で使用するために最後に使用したパラメータ値を保存します。

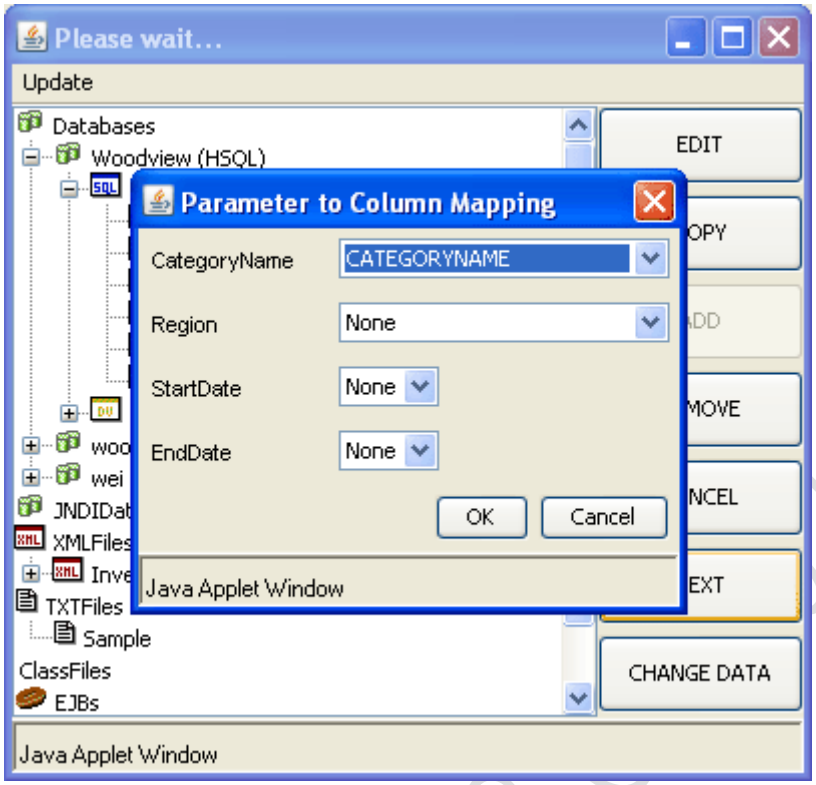
各サブレベルドリルダウンレポートには、少なくとも 1 つのマップされたパラメータが必要です。したがって、マッピングされていないパラメータを使用してレポートを作成するには、サブレベルクエリに少なくとも 2 つのパラメータが必要です。例：



Drill-Down レベルクエリ

少なくとも 1 つのパラメータをデータベース列にマップし、他のパラメータに対しては **none** を選択します。

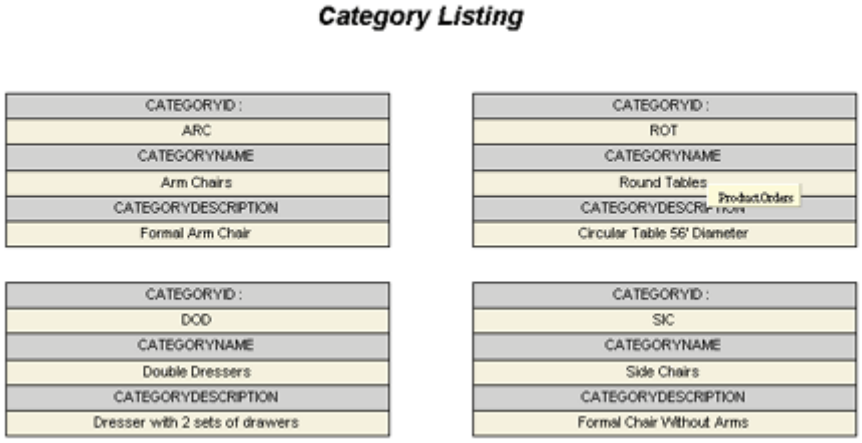
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Creating Unmapped Parameters

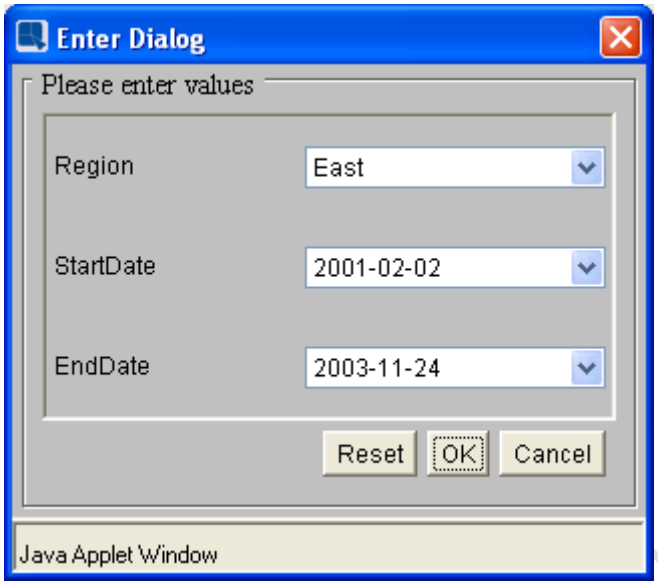
ドリルダウンレポートの作成の残りの部分は他のものと同じです。

レポートをプレビューすると、ルートレポートが通常のように表示されます（ルートレポートにパラメータが含まれている場合はパラメータプロンプトが表示されます)。



リンクをクリックし、ドリルダウンレポートのプレビューパラメータプロンプトメニューオプションが選択されている場合は、次のプロンプトが表示されます。CategoryName パラメータは表示されません。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



マップされていないドリルダウンのパラメータプロンプト

結果のレポートは次のようになります。

Product Orders

CATEGORYNAME	PRODUCTNAME	REGION	QUANTITY	ORDERDATE
Round Tables	Atun Table	East	21	Jul 11, 2003
	Apep Table	East	6	Nov 24, 2003
	Anubis Table	East	16	Feb 14, 2003
	Atun Table	East	6	Feb 21, 2003
	Ningizida Table	East	4	Jan 16, 2003
	Amon Table	East	12	Feb 14, 2003
	Ningizida Table	East	18	Jun 22, 2002
	Anubis Table	East	4	Jun 22, 2002
	Bast Table	East	16	Dec 4, 2001
	Bast Table	East	12	Mar 12, 2002
	Anubis Table	East	13	Feb 2, 2001
	Apep Table	East	15	Feb 24, 2001

エクスポートされたファイルから、デフォルト値が使用されますが、フィルタオプションから変更することができます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.11 サブレポート

サブレポートは、レポート内のレポートです。サブレポートは、プライマリレポートのセクション内に完全に入れ子になっています。各サブレポートには、独自のデータソースとデザインがあります。サブレポートを使用すると、関連しないデータを使用するレポートを作成できます。複数のデータテーブルを持つレポートを作成することもできます。

EspressReport では、サブレポートは一般的にプライマリレポートとは無関係です（ただし、パラメータ化されたサブレポートはプライマリレポートの列フィールドにリンクできます。リンクされたサブレポートを参照してください）。各サブレポートは独自のデータソースを使用し、プライマリレポートとは独立して設計できます。また、プライマリレポートに配置されるセクション内で完全に実行されます。既存のレポート（.pak、.rpt、.xml）テンプレートを使用するか、サブレポートの新しいテンプレートを作成することができます。

3.11.1 サブレポートの追加

既存のレポートにサブレポートを追加するには、ツールバーの **Insert Sub-Report** ボタン をクリックするか、**Insert** メニューから **Insert Sub-Report** を選択します。2 番目のメニューが開き、新しいサブレポートを追加するか、既存のレポートを再利用するかを選択できます。新しいサブレポートを挿入するには、**New...**を選択します。

カーソルが十字に変わります。サブレポートを挿入する場所に配置し、をクリックします。既存のレポートを使用するか、新しいレポートを作成するかを尋ねるダイアログボックスが表示されます。

.pak、.rpt、.xml 形式の既存のレポートをサブレポートとして使用できます。既存のレポートを使用するように選択すると、レポートファイルの場所を指定するよう求められます。ファイルを指定すると、そのファイルがサブレポートとして挿入されます。

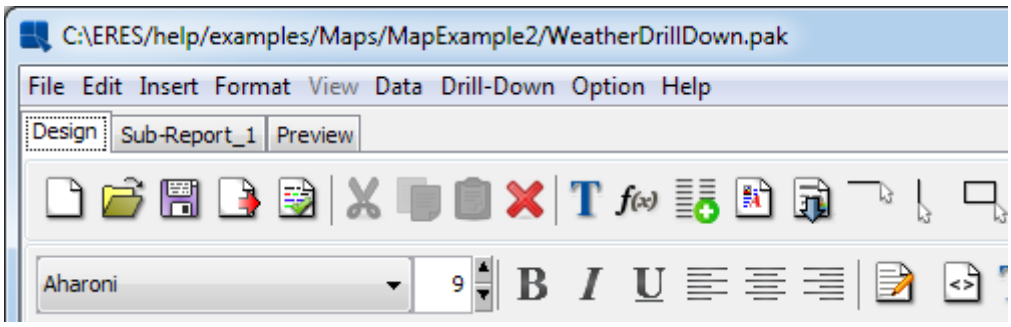
サブレポートの新しいレポートを作成するように選択すると、レポートウィザードが起動し、データソース選択からレポートマッピングにレポートを設計するプロセスを実行できます。レポートマッピングが終了すると、新しいレポートがサブレポートとして挿入されます。

3.11.2 サブレポートの編集

サブレポートを追加すると、新しいタブが Report Designer ウィンドウの上部に表示されます。

Copyright(C) 2018 Climb.Inc.

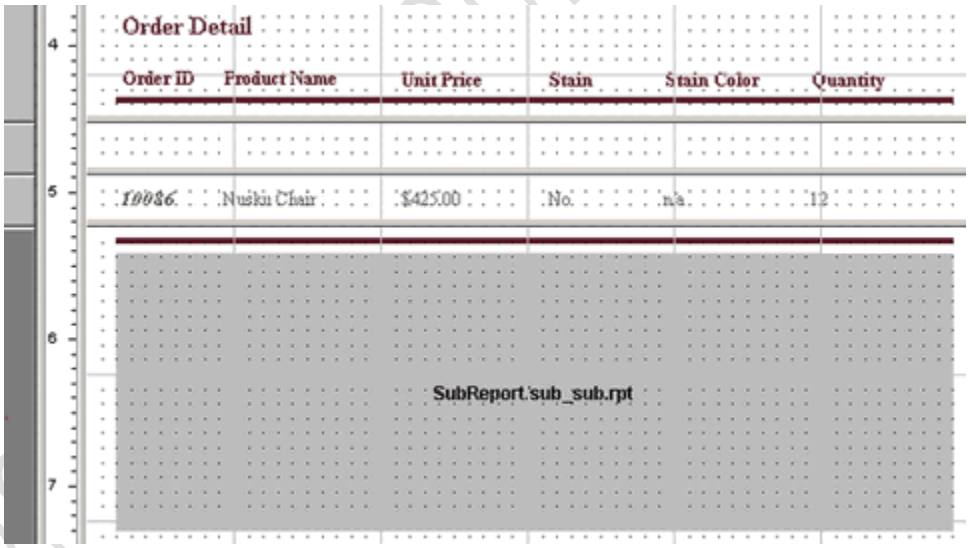
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



サブレポートタブ

Sub-Report タブを選択すると、サブレポートを変更できる新しいデザインペインが開きます。サブレポートは、レポートを変更するのと同じ方法で変更できます。追加するサブレポートごとに、新しいタブが表示されます。サブレポートがある場合、**Preview** タブは状況依存型になります。**Sub-Report** タブの **Preview** タブをクリックすると、プレビューウィンドウにはサブレポートのみが表示されます。**Design** タブの **Preview** タブをクリックすると、サブレポートが埋め込まれたメインレポートがプレビューされます。

デザインウィンドウ内で、サブレポートは灰色の四角形で表示されます。任意のオブジェクトのように移動およびサイズ変更できます。また、デザインウィンドウで対応する四角形をダブルクリックしてサブレポートを編集することもできます。



デザインウィンドウでのサブレポート

サブレポートは、レポートセクションが配置されているセクションですべて実行され、レポートセクションが繰り返されるたびに繰り返されます。つまり、サブレポートがテーブルヘッダーセクションに配置されている場合は、（**Print on Every Page** オプションが有効になっていない限り）1 回だけ実行されます。ただし、サブレポートが **Group Header** セクションに配置されている場合、サブレポートはレポートの各グループごとに繰り返されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

これに対する例外はテーブルデータセクションです。テーブルデータセクションに挿入されたサブレポートは、各行で繰り返されるのではなく、プライマリレポートのデータと並行して実行されます。これにより、2 つのデータテーブルが互いに隣り合って実行されるレポートを作成することができます。

3.11.2.1 サブレポートサイジング

メインレポートのデザインウィンドウの灰色の矩形のサイズは、サブレポートのサイズを表します。サブレポートがデザインウィンドウで許可するスペースより大きい場合、サブレポートの一部が切り捨てられます。

メインレポート内でサブレポートのサイズを固定することは、サブレポート内のレコード数が異なる場合には最適な解決策ではありません。これを解決するには、サブレポートに可変の高さを設定します。サブレポートの動的サイジングは、サイズ変更に合わせてオプションを使用して他のレポートセルと同様に設定されます。この機能の詳細については、レポート要素の移動およびリサイズを参照してください。

また、サブレポートの幅を動的に設定することもできます。この機能は、データまたはフィルタの変更に応じてレポートの列数が異なるクロス集計タイプのレポートに最も役立ちます。これにより、サブレポートには、サブレポートのすべての表示可能な列に収まるように常にサブレポートが描画されます。サブレポートの動的幅を設定するには、デザインウィンドウでサブレポートを右クリックし、ポップアップメニューから **Auto Resize Width** を選択します（この機能を無効にするには、ポップアップメニューを開いてこのオプションをもう一度クリックします）。

サブレポートに可変の高さを設定すると、プライマリレポートの同じセクションのサブレポートのすぐ下に配置されたオブジェクトは、サイズ変更されたサブレポートに対応するように移動しません。したがって、これらのオブジェクトはサブレポートによってオーバーラップされることがあります。

オブジェクトが重複しないようにするには、独自のレポートセクションに動的な高さのサブレポートを配置することをお勧めします。必要に応じて、ネストされたセクションをレポートに追加できます。詳細は、ネストされたセクションを参照してください。

3.11.3 サブレポートの削除

プライマリレポートからサブレポートを削除するには、2 つの方法があります。 プライマリレポートでサブレポートオブジェクトを選択して削除するだけです。これにより、サブレポートがプライマリレポートから削除されますが、サブレポートはそのまま使用できます。これは、サブレポートタブがまだアクティブであり、サブレポートをプライマリレポートに再挿入できることを意味します。

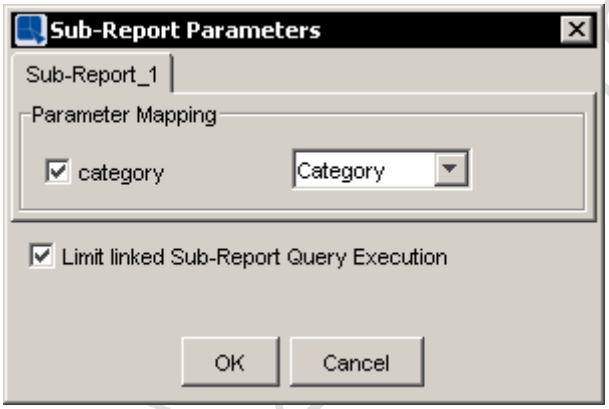
2 番目の方法は、**Edit** メニューから **Remove Sub-Report** を選択することです。2 番目のメニューが開き、現在のレポートに埋め込まれているすべてのサブレポートのリストが表示されます。削除するサブレポートを選択します。サブレポートタブは削除され、サブレポートは使用できなくなります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.11.4 リンクされたサブレポート

場合によっては、サブレポートをプライマリレポートのデータと調整することもできます。たとえば、グループ化されたデータがある場合、サブレポートをフィルタリングして、各グループに関連するデータを表示することができます。リンクされたサブレポートを使用してこれを行うことができます。

サブレポートは、クエリパラメータを使用してプライマリレポートにリンクされます。したがって、リンクされたサブレポートを作成するには、サブレポートにパラメータ化されたクエリが必要です。サブレポートを作成したら、**Data** メニューの **Sub-Report Parameter Mapping** を選択してサブレポートをリンクできます。これにより、プライマリレポートの列フィールドをサブレポートのクエリパラメータにマップできるダイアログボックスが表示されます。この操作は、ドリルダウンで使用される操作に似ています。



Sub-Report Parameter Mapping ダイアログ

ダイアログには、パラメータ化された各サブレポートのタブが含まれています。サブレポートの各パラメータは、プライマリレポートの別の列フィールドにマップできます。

通常、リンクされたサブレポートが実行されると、サブレポートはレポート列の現在の値を使用して実行されるたびにクエリを発行します。ただし、サブレポートが頻繁に実行されるようにレポートを構成すると、これらの繰り返しクエリがパフォーマンスに影響を与えることがあります。このダイアログの最後のオプションをチェックしてサブレポートのクエリ実行を制限すると、EspressReport はサブレポートクエリをマージし、サブレポートが実行されるときにデータベースへの呼び出し回数を制限しようとします。このオプションは、サブレポートがデータベースクエリを使用してデータを取得する場合にのみ有効です。これにより、パラメータ化されたクラスまたは xml ファイルをデータソースとして使用するサブレポートには影響しません。

リンクされたサブレポートを使用する場合は、カスケードパラメータの機能を使用しないことをお勧めします。これは、カスケードパラメータとその表示順序によっては間違ったデータが表示される可能性があるためです。メインレポートの列によって渡された値の代わりに、パラメータのデフォルト値が常に選択されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.11.4.1 リンクされたサブレポートの使用

リンクされたサブレポートはどの設定でもレポートセクションでも実行されますが、リンクされたプレゼンテーションを作成するときはサブレポートの動作を認識することが重要です。サブレポートの編集で説明したように、サブレポートはそれらが配置されているセクションの回数だけ実行され、テーブルデータセクションに配置されたときに 1 回だけ実行されます。

したがって、リンクされたサブレポートを 1 回だけ実行するセクション（テーブルヘッダー、レポートヘッダー、テーブルデータなど）に配置すると、サブレポートは 1 回だけ実行され、そのサブレポートは 1 回だけ実行され、レポートをフィルタリングするためにリンクされています。サブレポートがリンクされている列の各値に対して実行されるリンクプレゼンテーションを生成するには、サマリブレークレイアウトを使用する必要があります。列を行区切りとして設定し、列とサブレポートの両方を Group Header セクションに配置すると、目的の結果が得られます。

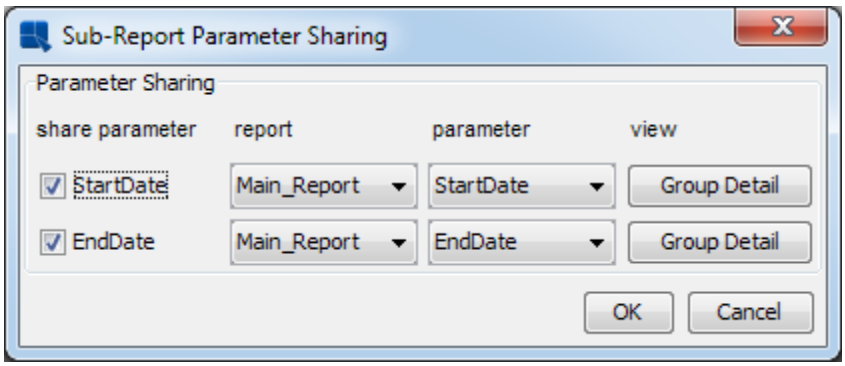
列とサブレポートの関係がデータベースに既に存在する場合は、リンクされたサブレポートではなく、サイドバイサイドのマスタ&詳細レイアウトを生成することができます。マスタ&詳細レイアウトを使用すると、パフォーマンスが向上します。詳細については、データマッピングを参照してください。

3.11.5 パラメータのリンク

サブレポートは、メインレポートの列から値を取得するだけでなく、メインレポートまたは他のサブレポートとパラメータ値を共有することもできます。この構成では、ユーザは、メインレポートのデータをフィルタリングし、サブレポートデータに同じフィルタを適用するように値を入力することができます。

パラメータのリンクは、パラメータが同じデータ型を共有する場合にのみ使用できます。デフォルトでは、メインレポートと同じ名前とデータタイプのパラメータを持つサブレポートを作成すると、リンクされます。パラメータ値がメインレポートに入力されると、自動的にサブレポートにも渡されます。

パラメータのリンク動作は、変更するサブレポートに移動し、**Sub-Report** タブの **Data** メニューから **Sub-Report Parameter Sharing** を選択することで変更できます。これにより、サブレポートパラメータのリンク動作を変更できるダイアログが表示されます。



Parameter Sharing ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

サブレポートに定義されている各パラメータについて、リンクを有効または無効にすることができます。また、リンクするメインレポートまたは別のサブレポートのパラメータを設定することもできます。**Group Detail** ボタンを押すと、そのパラメータの現在の関係を示すダイアログが表示されます。

すべてのパラメータに必要な設定を指定したら、**OK** ボタンをクリックして変更を適用します。

クエリパラメータと初期化の詳細については、パラメータ化されたクエリを参照してください。

リンクされたサブレポートで説明されているように、メインレポートからサブレポートパラメータに列フィールドをマップするように選択すると、パラメータのリンクがオーバーライドされます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.12 テンプレートセキュリティ

レポートツールで最も一般的なニーズの 1 つは、異なるレポートテンプレートの数を最小限に保つことです。テンプレートの再利用を容易にする 1 つの方法は、異なるクラスのユーザのテンプレートの特定の側面を簡単に制御できるようにすることです。そのために、EspressReport では、テンプレートを表示するユーザに応じて、列レベル、行レベル、およびセルレベルの外観の両方を制御するために、任意の数のさまざまなセキュリティ構成を設定できます。

たとえば、販売地域ごとにグループ化された総売上を示す販売レポートがあるとします。レポートを表示するエグゼクティブは、各地域の合計額と寄付額を表示する必要がありますが、個々のアカウントマネージャーは地域のデータを見るだけで済みます。レポートの基本テンプレートは次のようになります。

Sales Report				
Total Units Sold: 29 Total Sales: $24,449.69				
Eastern Region				
Order #	Quantity	Product	Unit Price	Total Sales
1001	2	Chair	$325.16	$650.32
	1	Table	$1,114.18	$1,114.18
			Order Total:	$1,764.50
1004	3	Table	$1,114.18	$3,342.54
	2	Dresser	$1,518.60	$3,037.20
			Order Total:	$6,379.74
Total for Eastern Region:				$8,144.24
Midwestern Region				
Order #	Quantity	Product	Unit Price	Total Sales
1003	4	Chair	$325.16	$1,300.25
	2	Table	$1,114.18	$2,228.36
	1	Dresser	$1,518.60	$1,518.60
			Order Total:	$5,047.21
Total for Midwestern Region:				$5,047.21
Southern Region				
Order #	Quantity	Product	Unit Price	Total Sales
1005	2	Table	$1,114.18	$2,228.35
	1	Dresser	$1,518.60	$1,518.60
			Order Total:	$3,746.95
1007	3	Chair	$325.16	$975.48
	1	Table	$1,114.18	$1,114.18

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

			Order Total:	$2,089.66
Total for Southern Region:				$5,836.61
Western Region				
Order #	Quantity	Product	Unit Price	Total Sales
1006	2	Dresser	$1,518.60	$3,037.20
	4	Chair	$325.16	$1,300.25
	1	Table	$1,114.18	$1,114.18
			Order Total:	$5,451.63
Total for Western Region:				$5,451.63

セキュリティ機能を使用して、この 1 つのテンプレートを使用して各グループの要件を満たすことができます。この例では、レポートに領域に基づいてフィルタリングするクエリパラメータがあるとします。

エグゼクティブにとっては、上記のビューはレポートを見るときに見たいものなので、すべての地域のデータを取得する管理というセキュリティレベルを作成できます。これは、リージョンパラメータに使用可能なすべての値を管理レベルにマップすることによって実現されます。この例では、東部、中部、南部、西部地域の値がセキュリティレベルに割り当てられます。EspressReport では、クエリパラメータに複数の値を指定するには、複数値のパラメータを定義する必要があります。これについての詳細は、複数値のパラメータを参照してください。

アカウントマネージャーは、各地域ごとに異なるセキュリティレベルを作成できます。上記の例には 4 つの地域があるため、セキュリティレベル East、Midwest、South、および West が追加されます。アカウントマネージャーは集計データを表示するべきではないので、セルレベルのセキュリティで説明したセルレベルのセキュリティ機能を使用して、ヘッダーに要約を含むセルを領域レベルで非表示にします。各レベルが適切な地域のデータを返すように、東部地域のパラメータ値は「東部」セキュリティレベルにマップされ、中部地域のパラメータ値は中西部のセキュリティレベルにマップされ、パラメータ値は南部地域は南の治安レベルにマッピングされ、西部地域のパラメータ値は西の治安レベルにマッピングされます。パラメータ値を適用するセキュリティレベルの設定の詳細については、セキュリティパラメータを参照してください。

レポートが実行されると、セキュリティレベルが提供されます（通常、ユーザのアプリケーションは、ログインまたはその他の認証方法に基づいて使用するセキュリティレベルを決定します）。
EspressReport には独自のユーザ認証メカニズムがありません。そのセキュリティレベルのセル設定が適用され、そのレベルに関連付けられたパラメータ値が、ユーザに確認を求めずに自動的にクエリに提供されます。たとえば、東のセキュリティ設定で実行されたレポートは次のとおりです。

Sales Report
Eastern Region

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Order #	Quantity	Product	Unit Price	Total Sales
1001	2	Chair	$325.16	$650.32
	1	Table	$1,114.18	$1,114.18
			Order Total:	$1,764.50
1004	3	Table	$1,114.18	$3,342.54
	2	Dresser	$1,518.60	$3,037.20
			Order Total:	$6,379.74
Total for Eastern Region:				$8,114.24

3.12.1 セキュリティレベル

他のアプリケーション環境に組み込まれるように設計されたツールキットとして、EspressReport はユーザまたはアクセス権のリストを維持しません。むしろ開発者は、レポートテンプレート内で任意の数の一意のセキュリティレベルを作成できます。各レベルで、特定のセルの振る舞いやデータのフィルタリングを割り当てることができます。実行時に、セキュリティレベルを API を介してレポートに提供することができます。このソリューションは、ほとんどのアプリケーションセキュリティモデルに適応するのに十分な柔軟性を備えています。

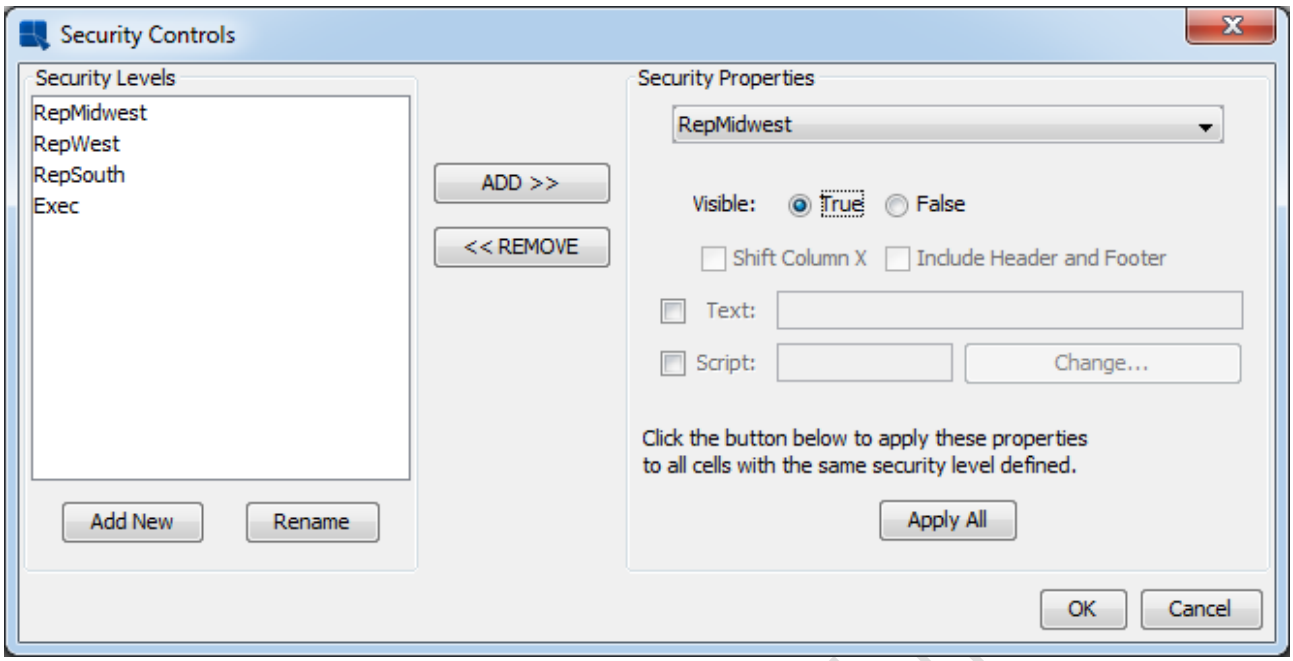
テンプレート内で定義できるさまざまなセキュリティレベルの数に制限はありません。唯一の要件は、各レベルに固有の名前が必要であることです。セキュリティレベルはレポート内でグローバルに使用され、セルレベルの設定と、それらに割り当てられたパラメータ値を持つことができます。セキュリティレベルは、セキュリティ設定が適用されるのと同時に定義できます。

3.12.2 セルレベルのセキュリティ

レポートのセル/可視列では、セキュリティレベルごとに異なる動作を設定できます。セルにセキュリティ設定を適用するには、セルを選択して **Data** メニューから **Security** を選択するか、セルを右クリックしてポップアップメニューから **Security** を選択します。これにより、そのセルのセキュリティ設定を指定できるダイアログが表示されます。

Copyright(C) 2018 Climb.Inc

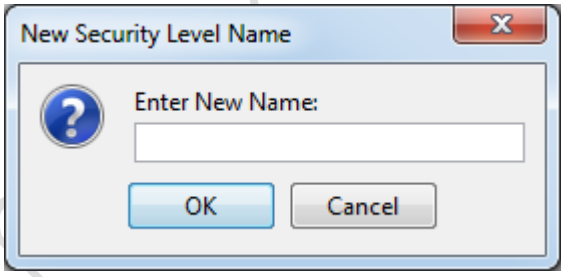
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Cell Security Setting ダイアログ

ダイアログには、テンプレートのすべての定義済みセキュリティレベルのリストが含まれています。

Add New ボタンをクリックすると、新しいセキュリティレベルを追加できます。これにより、新しいレベルの名前を指定できる追加のダイアログが表示されます。



Add Security Level ダイアログ

セキュリティレベルに必要な任意の名前を入力できます。**Rename** ボタンを選択して、レポート内の定義済みのレベルの名前を変更することもできます。レポートセルまたはクエリパラメータに関連付けられていないセキュリティレベルは、自動的に削除されます。

レポートセルのセキュリティ設定を追加するには、変更を有効にするレベルを選択し、**Add** ボタンをクリックします（新しいセキュリティレベルを作成すると、セルに自動的に追加されます）。セキュリティレベルは、セルセキュリティダイアログの左側にあるドロップダウンリストに表示されます。これで、そのセキュリティレベルのセルを変更できます。セルには 3 つの基本的なコントロールがあります。セキュリティレベルでセルを非表示にすることができます。セルが表示されている場合は、セルにカスタムテキストを入力するか、セルスクリプトを適用できます。セルスクリプトの詳細については、スクリプトを参照してください。列（表データセクションのセル）を非表示にする場合は、データセルとともに列ヘッダーを選択して、両方が同じセキュリティ設定を含むようにしてから、**Shift Column X** オプ

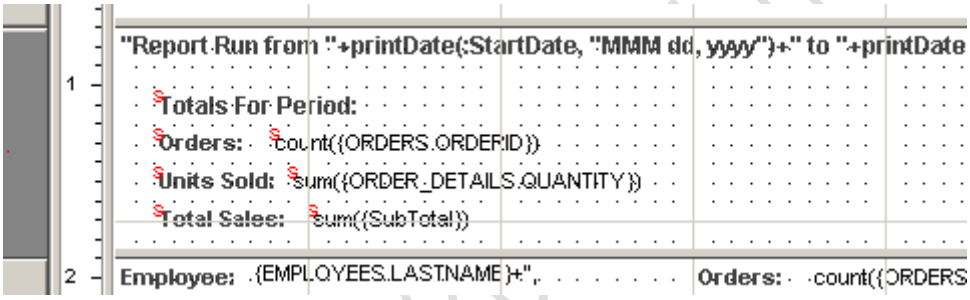
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ションをチェックしてもう一方の列を作成しますセキュリティレベルが適用されると、列が左に移動します。

セルのテキストを変更するように指定すると、デフォルトの列ヘッダーでは機能しません。これは、列ヘッダーがデータソースから値を取得する一意の要素であるためです。列ヘッダーのテキストを変更する場合は、最初に静的ラベルに変換する必要があります。詳細については、要素の編集を参照してください。

この動作は、右側のボタンをクリックしてこのセキュリティレベルのコントロールを持つセルに伝播できます。必要に応じて、さまざまなレベルのセルに対してコントロールを指定できます。

セルのセキュリティ設定を指定したら、**OK** ボタンをクリックします。デザインウィンドウに戻り、セキュリティ設定がセルに適用されたことを示す左上隅に赤が表示されます。



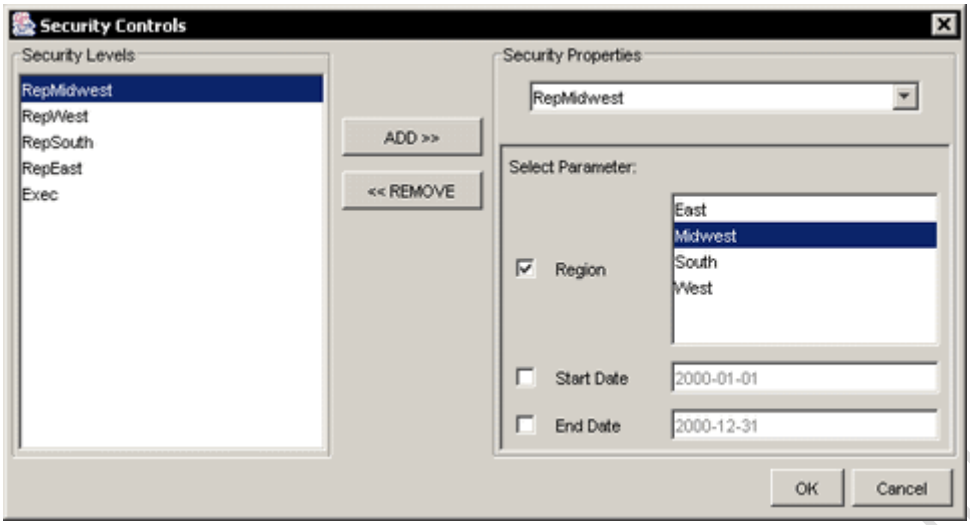
セキュリティ設定でセルを報告する

3.12.2.1 セキュリティパラメータ

レポート内のセルの特定の動作を制御するだけでなく、セキュリティレベルにパラメータ値を割り当てることもできます。たとえば、販売レポートが地域別にグループ化されている前述のシナリオを使用すると、販売地域別にフィルタリングするクエリパラメータを追加し、その地域のセキュリティレベルに適切な地域値を割り当てることができます。従業員がログインすると、適切なセキュリティレベルでレポートが実行され、適切なパラメータ値がセキュリティレベルでレポートに提供されるため、ユーザは販売地域のデータのみが表示されます。セキュリティパラメータを使用するには、レポートでパラメータ化されたデータベースクエリまたはパラメータ化されたクラスのいずれかをデータソースとして使用する必要があります。

パラメータのセキュリティ設定を追加するには、**Data** メニューから **Secured Query Parameters** を選択します。これにより、クエリパラメータ値を選択できるダイアログが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Security Parameter ダイアログ

セルセキュリティダイアログと同様に、パラメータセキュリティダイアログには、左側のテンプレートに定義されているすべてのセキュリティレベルのリストが含まれています。セルレベルのセキュリティダイアログと同じ方法でレベルを作成および編集できます。

レポートパラメータにセキュリティ設定を追加するには、パラメータ値を割り当てるレベルを選択し、**Add** ボタンをクリックします。ダイアログボックスの右側にあるドロップダウンリストにセキュリティレベルが表示されます。選択したセキュリティレベルごとに、パラメータの横にあるチェックボックスを使用して、保護するパラメータを選択できます。セキュリティで保護されたパラメータごとに、このセキュリティレベルに関連付ける値を選択できます。パラメータ値の割り当てが終了したら、**OK** ボタンをクリックしてデザインウィンドウに戻ります。

3.12.3 サブレポート＆ドリルダウンレイヤのセキュリティ

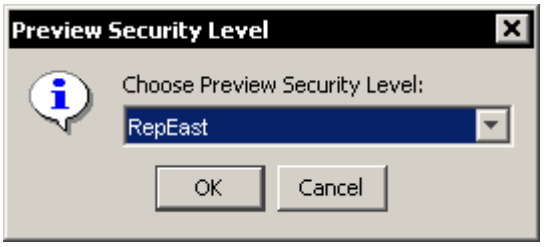
テンプレートセキュリティ設定は、サブレポートおよびドリルダウンレイヤに自動的に適用できます。どちらの場合も、セキュリティを追加するには、サブレポートまたはドリルダウンレイヤを編集する必要があります（サブレポートタブをクリックするか、ドリルダウンレイヤに移動します）。サブレポートまたはドリルダウンレイヤでは、メインレポートと同じようにセキュリティ設定を追加できます。メインレポートとサブレポート/ドリルダウンレイヤの両方にセキュリティを適用する場合は、セキュリティレベルにメインレポートと同じ名前を割り当てる必要があります。レポートがセキュリティレベルで実行されると、セキュリティレベルはサブレポートおよび/またはドリルダウンレイヤに自動的に適用されます。

ドリルダウンおよびサブレポートの詳細については、ドリルダウンおよびサブレポートを参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.12.4 保護されたテンプレートの表示

Report Designer の **Preview** ウィンドウで、セキュリティ設定の効果をプレビューできます。 これを行うには、**Data** メニューから **Preview Security Level** を選択します。これにより、レポートの表示に使用するセキュリティレベルを選択するダイアログが表示されます。



Preview Security Level ダイアログ

必要なレベルを選択し、レポートをプレビューします。選択したレベルにセキュリティ設定が適用されたレポートが表示されます。このレベルは、設定を変更するまで選択されたままです。このダイアログのリストの上部にある空白のエントリを選択すると、セキュリティ設定を適用せずにレポートが実行されます。

API を使用してセキュリティ保護されたテンプレートを実行する場合は、使用するセキュリティ設定を適用できます。セキュリティレベルは、**QbReport** コンストラクタの引数として渡すことも、**setSecurityLevel()**メソッドを使って設定することもできます。API でセキュアなテンプレートを実行する方法の詳細については、安全なレポートを参照してください。

また、EspressReport インストールに含まれるセキュリティ設定でレポートを実行する方法を示すサーブレットの例もあります。ソースコードとセットアップの説明は、**EspressReportInstallDir/help/examples/servlet/ReportSecurity** ディレクトリにあります。

3.12.4.1 保護された Report Designer

API を使用して Report Designer を実行する場合、セキュリティを適用することもできます。Designer が起動すると、**setSecurityLevel()**メソッドを使用してセキュリティレベルを渡すことができます。Report Designer がセキュアモードで実行されている場合、ユーザは指定されたレベルでのみレポートをプレビューできます。 さらに、定義済みのセキュリティ設定またはセキュリティ設定として適用されたセルスクリプトを使用してレポートセルを変更することはできません。 API から Report Designer を実行する方法の詳細については、レポート API からの Report Designer を呼び出すを参照してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3.13 スケジューリング

EspressReport にはシンプルなインタフェースが含まれているため、サーバ側の他のイベントと同様に、レポートのスケジュールを設定できます。スケジューラは EspressManager と連携して実行され、スケジュールされたタスクを管理するだけでなく、インタフェースを通じてスケジュールを設定することができます。

スケジューラは、サーバ上のアプリケーションとして、または Web ブラウザを介してリモートで実行できます。スケジューラを実行するには、次の手順を実行します。

アプリケーション:

スケジューラをローカルで実行している場合は、まず **espressmanager.bat** ファイルを実行して EspressManager を起動してください。その後、スケジューラバッチファイルを実行してスケジューラを開始します。これでアプリケーションが起動します。

ブラウザ:

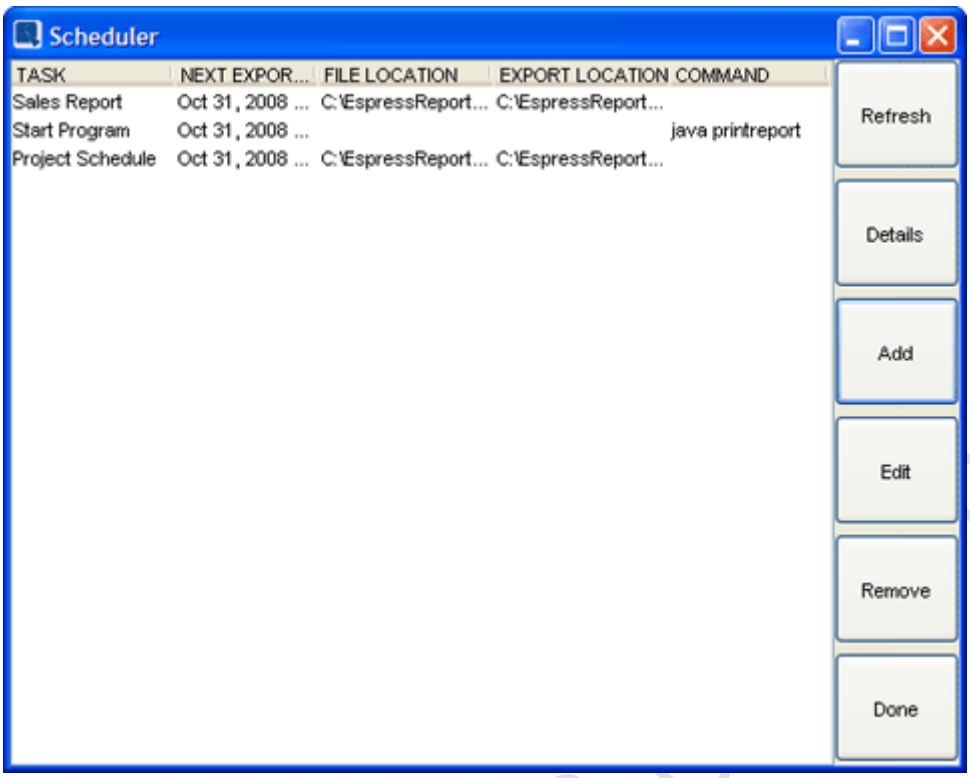
スケジューラをリモートで実行している場合は、サーバマシン上で EspressManager が実行されていることを確認してから、ブラウザに Scheduler の URL（**http://machinename/espressreport/scheduler.html**）を入力します。これにより、スケジューラがブラウザのアプレットウィンドウとして表示されます。

3.13.1 スケジューラインタフェース

メインスケジューラインタフェースは、スケジュールされたジョブの一覧を含むウィンドウです。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



メインスケジューラインタフェース

ジョブごとに、ジョブ名、次のスケジュールされたエクスポート時刻、レポートファイルの場所（該当する場合)、エクスポートされたファイルの場所（該当する場合)、スケジュールされたコマンドライン（該当する場合）が表示されます。ウィンドウの右側にあるボタンは、次の機能を実行します。

Refresh:

このオプションは、スケジュールされたタスクのリストを更新します。これは、スケジュールを追加する他のユーザーがいる場合に便利です。

Details:

このオプションは、現在選択されているスケジュールされたタスクに関する詳細情報を表示する新しいウィンドウを開きます。

Add:

このオプションを使用すると、新しいスケジュールを設定できます。

Edit:

このオプションを使用すると、現在選択されているスケジュールされたタスクを編集できます。

Remove:

このオプションは、現在選択されているスケジュールされたタスクを削除します。

Done:

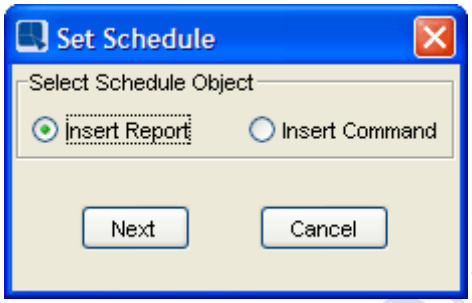
このオプションはスケジューラを終了します（このボタンは、スケジューラをアプリケーショ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ンとして実行している場合のみ表示されます)。

3.13.2 スケジュールの設定

新しいスケジュールを作成するには、スケジューラのメインウィンドウで **Add** ボタンをクリックします。これにより、レポートまたはコマンドをスケジュールするかどうかを尋ねるダイアログが表示されます。



First Schedule ダイアログ

Report：

このオプションを使用すると、スケジュールするレポートテンプレート（.pak、.rpt、または.xml形式）を選択できます。スケジュールの周期性と配信オプションを指定することができます。この機能の詳細については、レポートのスケジュール設定を参照してください。

Command：

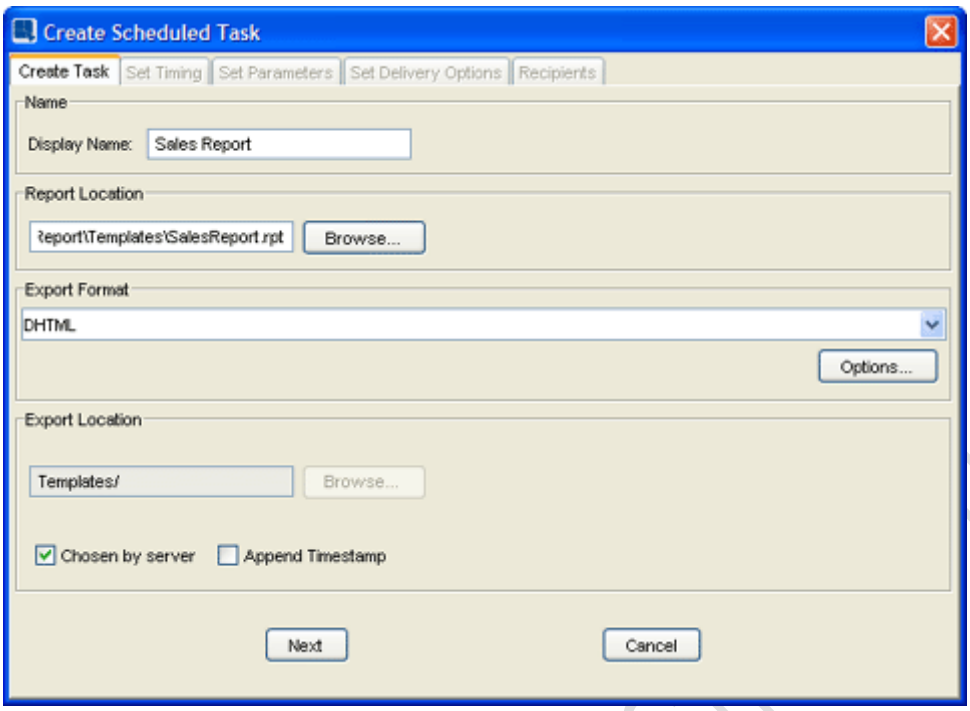
コマンドオプションを使用すると、レポートファイル以外のスケジュールを設定できます。この機能を使用すると、EspressManager と同じディレクトリで実行されるコマンドを記述することができます。したがって、この機能を使用して、Report API を含む Java プロセスを実行できます。この機能の詳細については、コマンドのスケジューリングを参照してください。

3.13.2.1 レポートのスケジュール設定

レポートをスケジュールするには、スケジューラで新しいスケジュールを追加し、**Insert Report** オプションを選択します。スケジューリングプロセスの最初のダイアログが開きます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Schedule ダイアログ

この画面では、最初にスケジュールされたイベントの表示名を指定するよう求められます。2 番目のセクションでは、スケジュールに使用するテンプレートを指定できます.3 番目のセクションでは、レポートのエクスポート形式を指定できます。使用可能なフォーマットには、DHTML、PDF、CSV、Excel（XLS)、Excel 2007（XLSX)、テキスト、リッチテキスト、または XML があります。**Options...**ボタンを使用すると、エクスポートされたファイルのフォーマット固有のオプションを設定できます。DHTML では、単一または複数ページの書き出しを設定できます。また、使用する CSS オプションを指定することもできます。PDF の場合は暗号化を設定でき、テキストの場合は区切り文字を指定できます。レポートのエクスポートオプションの詳細については、レポートのエクスポートを参照してください。

最後のセクションでは、レポートのエクスポート場所を指定できます。エクスポートされたファイル（拡張子なし）のパスとファイル名を指定するか、サーバが場所とファイル名を割り当てることができます。場所とファイル名を入力すると、スケジュールが実行されるたびにサーバはそのファイルを最新バージョンで上書きします。サーバが場所とファイル名を割り当てることを許可すると、ファイルはサーバルートの下にエクスポートされたレポートの ScheduledReports ディレクトリに置かれます。

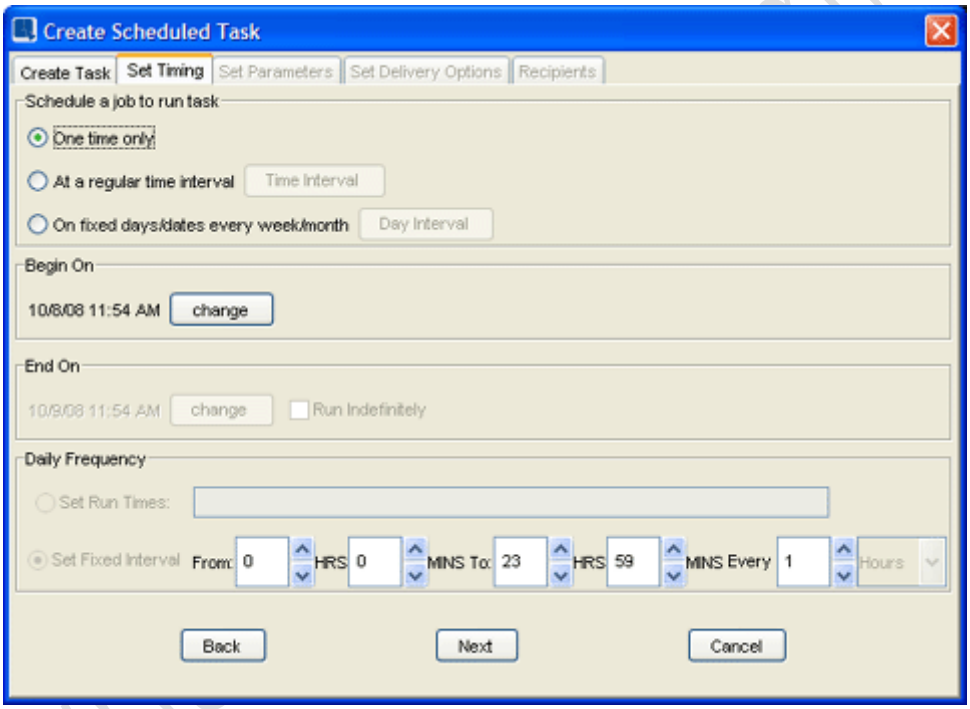
Append Timestamp オプションを使用すると、ファイルの保存方法を制御できます。通常、スケジュールジョブが実行されると、生成された最新のバージョンが上書きされます。ただし、スケジュールが実行されるたびにファイル名へのエクスポート時刻をタイムスタンプに付加するように指定することができます。これにより、最後のファイルを上書きするのではなく、スケジュールされた実行ごとに一意のファイルが生成されます。

イメージと名前のオプションの指定が終了したら、**Next** をクリックします。レポートにセキュリティレベルが含まれている場合、このレポートに使用するセキュリティレベルを尋ねるダイアログボックスが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

セキュリティレベルを設定する

このダイアログでは、テンプレートに定義されているセキュリティレベルのいずれかを選択するか、テンプレートにセキュリティ設定を適用しないデフォルト設定を使用できます。スケジュールは、定義したセキュリティレベルを使用して実行されます。使用するセキュリティレベルを選択したら、**OK** をクリックします。スケジュールの周期性を指定するよう求められます。



Set Start & End Date ダイアログ

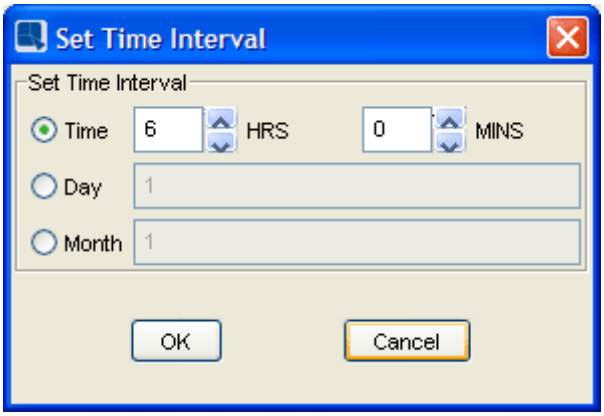
次の画面では、スケジュールされたイベントの開始日と終了日を選択できます。Begin On および End On セクションでは、スケジュールの開始日と終了日を選択できます。また、終了日を指定する代わりに、**Run Indefinitely（無期限に実行）**をチェックすることもできます。

ウィンドウの上部では、チャートまたはレポートを実行する 3 つのオプションの 1 つを選択できます：**One time only**、**At a regular time interval**、**and On fixed dates/days every week/month**。

最初のオプションは、指定された開始日にイベントを 1 回だけ実行します。このオプションを選択すると、終了日を指定する必要はありません。2 番目のオプションを選択した場合は、開始日と終了日を指定し、**Time Interval** ボタンをクリックしてスケジュールの間隔を指定してください。時間間隔を時間/分、日、または月単位で選択できるダイアログが表示されます。エクスポートは開始時刻に開始され、

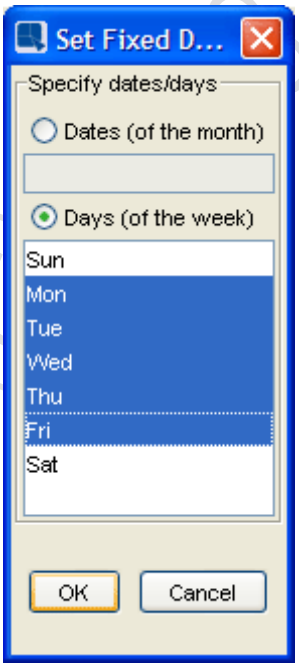
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

指定された間隔で実行されます。



Select Time Interval ダイアログ

3 番目のオプションを選択した場合は、開始日と終了日を指定し、**Day Interval** ボタンをクリックしてスケジュールの間隔を指定してください。ダイアログが表示され、イベントが実行される月の日付（カンマ区切り、つまり 5,15）または曜日が入力できます。



Select Fixed Days/Dates ダイアログ

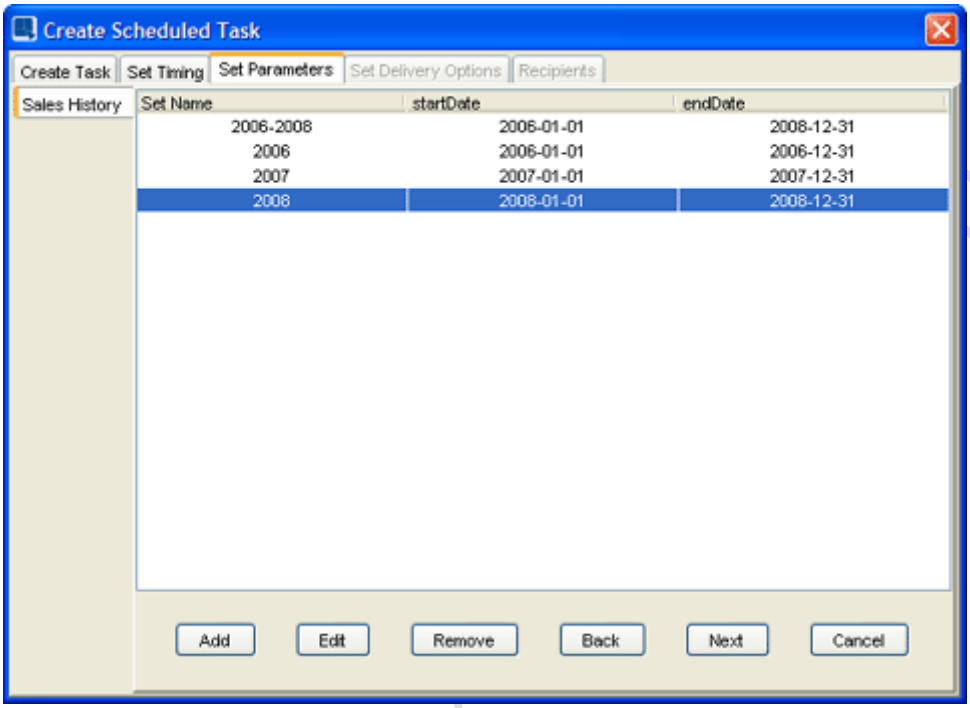
3 番目のオプションを選択すると、1 日の頻度を指定する必要があります。Daily Frequency セクションでは、指定した日にスケジュールを実行するタイミングを微調整できます。**Set Run Time** オプションを使用すると、1 日にエクスポートを実行する正確な時刻を指定できます。たとえば、**Set Run Times** オプションを使用すると、午前 9 時、午後 5 時 30 分、午前 12 時などの不定期に実行するスケジュールを指定できます。**Set Fixed Interval** オプションを使用すると、定期的な間隔に基づいて実行時間を指定できます。たとえば、このオプションを使用して、午前 9 時から午後 5 時までの 30 分ごとに実行されるスケジュールを設定できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

周期性の指定が終わったら、**Next** をクリックしてスケジュールウィザードを続行します。

3.13.2.2 パラメータ化されたレポートのスケジュール設定

パラメータを含むレポートのスケジュールを選択した場合は、次のタブが表示され、使用するパラメータ値を設定できます。



Set Schedule Parameters ダイアログ

パラメータのセットを追加するには、**Add** ボタンを選択します。これにより、パラメータプロンプトダイアログが表示され、使用するパラメータ値のセットを選択できます。パラメータセットを選択すると、その選択肢がダイアログに表示されます。必要に応じて、さまざまな組み合わせのパラメータセットを追加できます。指定したパラメータのセットごとに個別のファイルが生成されます。

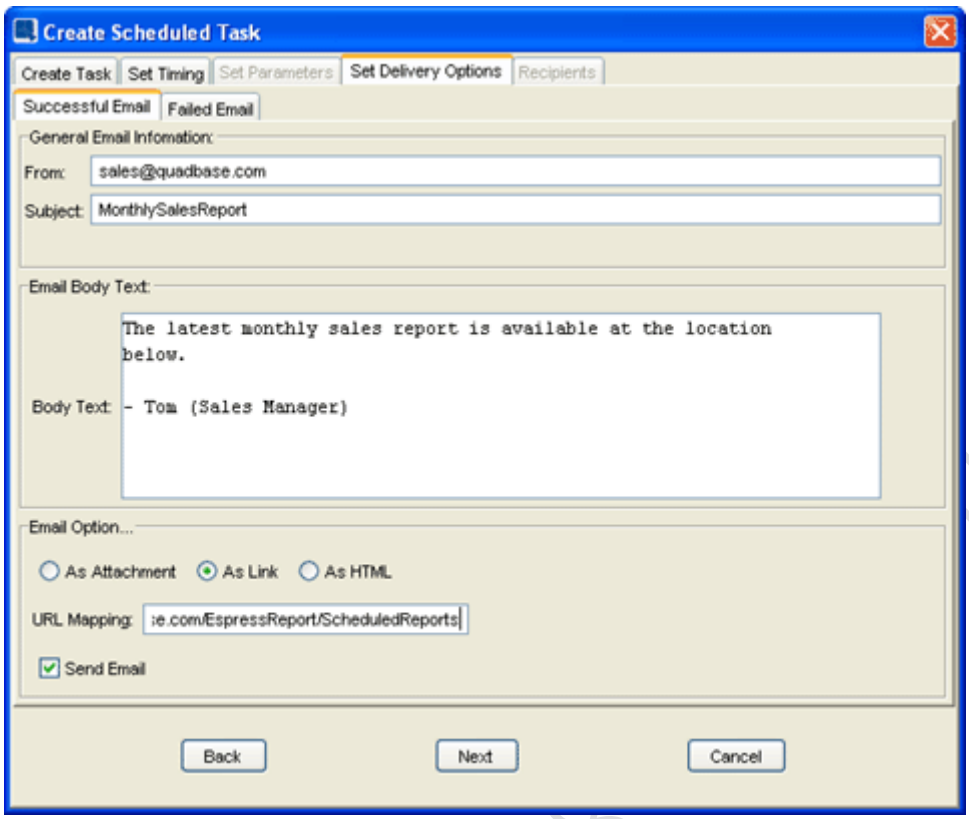
また、最初の列をダブルクリックして、各パラメータセットの名前を指定することもできます。ここで指定された名前は、後のダイアログで受信者の整理に役立ちます。

必要なパラメータをすべて指定したら、**Next** ボタンをクリックして続行します。

3.13.2.3 メール配信オプション

スケジュールの設定が完了すると、レポートを電子メールでユーザに配信するための 2 つのタブ付きダイアログが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Email Delivery Options ダイアログ

レポートを電子メールで配信することを選択した場合は、下部にある **Send Email box** をオンにします。最初のタブには、レポートが正常にエクスポートされたときに送信された電子メールの情報が含まれています。このタブの **General Email Information** の部分では、電子メールの送信元アドレスと件名を指定できます。本文テキスト部分では、電子メールの本文にメッセージを指定できます。

スケジュールされたレポートを電子メールで送信するには、3 つの方法があります。生成されたレポートを電子メールの添付ファイルとして送信したり、生成されたレポートへのリンクを送信したり、レポートを電子メールで送信することができます（このオプションは DHTML エクスポートでのみ使用できます）。生成されたレポートへのリンクを送信する場合は、レポートが生成されるディレクトリへの http パスを指定する必要があります。そうしないと、リンクが正しく形成されません。たとえば、EspressReport が Tomcat の Web ルートにインストールされていて、サーバが生成されたファイルの場所/名前を選択できるようにした場合、実際のディレクトリの場所は次のようになります。

```
C:¥Tomcat5.5¥webapps¥ROOT¥EspressReport¥ScheduledReports
```

対応する URL マッピングは次のようになります。

```
http://hostname:8080/EspressReport/ScheduledReports
```

特定のジョブに電子メールの配信/通知を含めたくない場合は、**Send Email** と表示されているオプションの選択を解除するだけです。電子メールオプションの指定が完了したら、**OK** ボタンをクリックする

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

と、スケジュールが完了します。

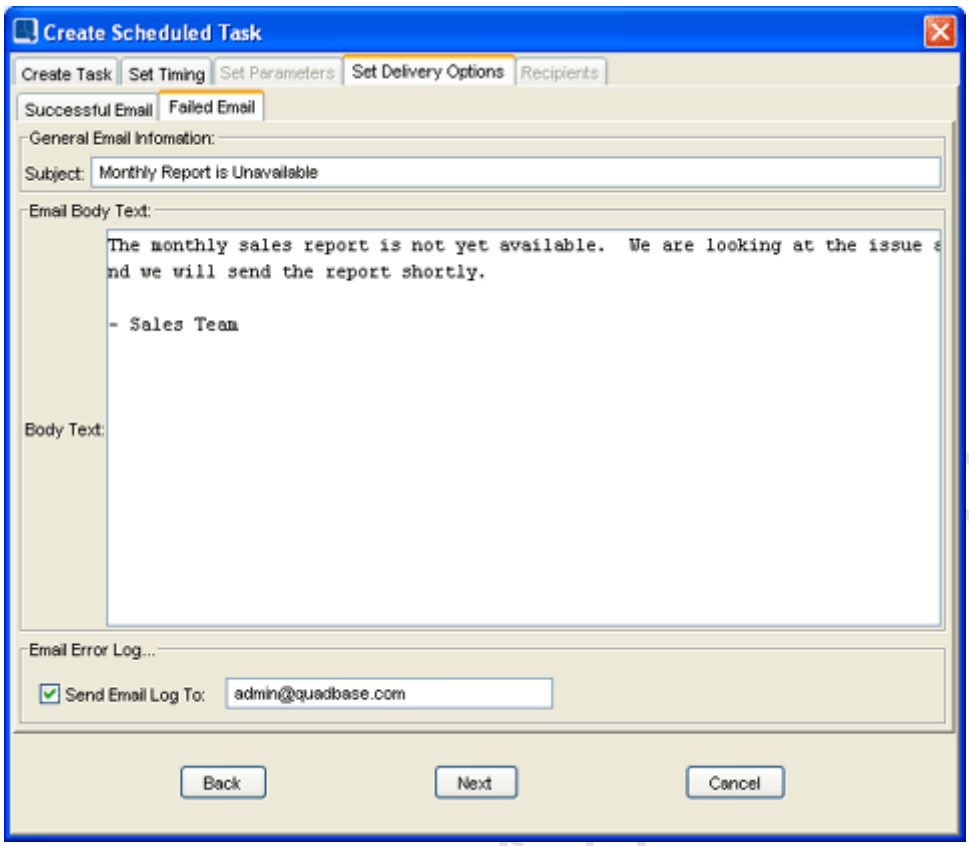
電子メール配信が正常に動作するためには、EspressManager 設定ファイルで smtp 情報を正しく設定する必要があります。詳細は、設定を参照してください。

電子メールを正しく送信するには、smtp ホストを userdb ディレクトリの設定ファイルに設定する必要があります。smtp ホストを設定するには、config.txt ファイルを開きます。[smtp host]という見出しの下に、スケジュールされたレポートを電子メールで送信するときに使用する SMTP サーバの名前を追加します。セキュア（TLS）smtp サーバを使用している場合は、[smtp secured]、[smtp port]、[smtp username]、[smtp password]の各値も入力してください。例えば：

```
[smtp host]
quadbase.com

[smtp secured]
true

[smtp port]
587

[smtp username]
guest

[smtp password]
password
```

Failed Email タブでは、レポートの実行中にエラーが発生した場合に、受信者に別の電子メール（件名と本文）を送信できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



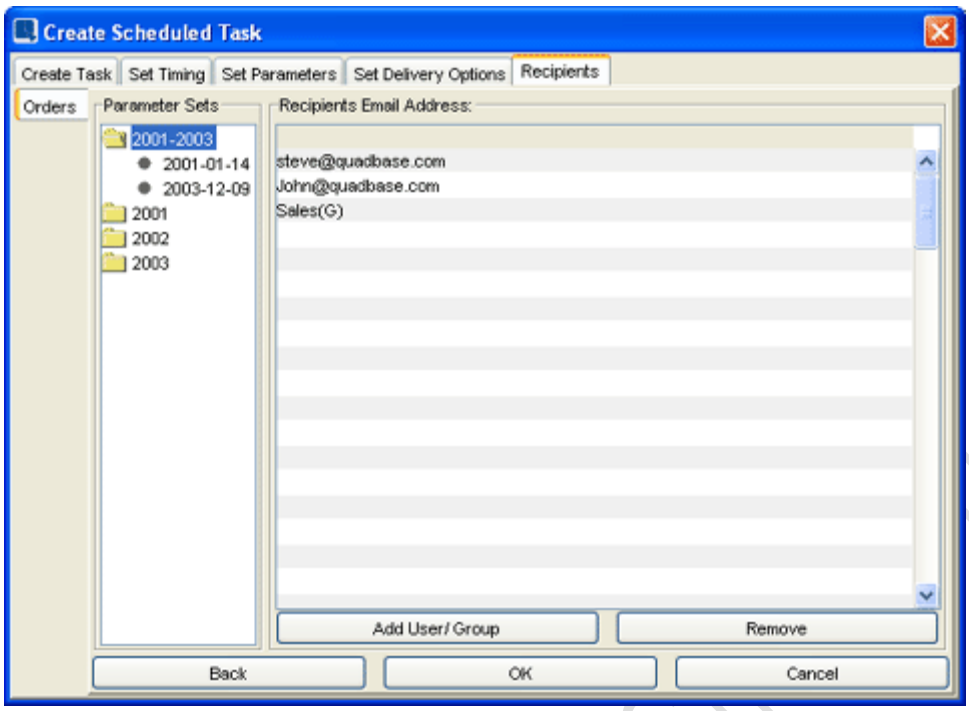
パラメータ化されたレポートの受信者一覧

このタブでは、エラーログを特定のユーザに送信することもできます。通常、これは管理者または技術ユーザで、問題の診断とトラブルシューティングが可能です。

3.13.2.4 電子メール受信者の指定

スケジュールされたファイルを電子メールで送信することを選択した場合は、次のタブで受信者リストを指定できます。受信者リストは、スケジュールしているレポートまたはチャートの種類によって異なる設定になっています。レポートまたはチャートにパラメータが含まれている場合は、次のダイアログが表示されます。これにより、異なるユーザまたはグループに異なるパラメータセットを送信することができます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



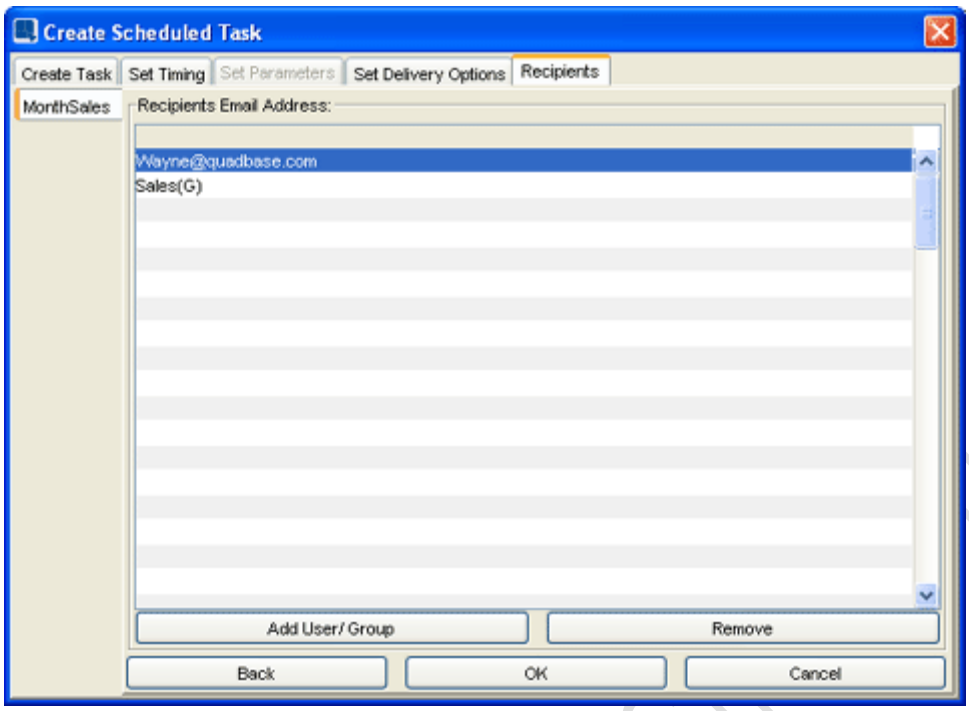
パラメータ化されたレポートの受信者一覧

受信者をリストに入力するのは簡単です。受信者リスト内の任意の行をダブルクリックして入力すると、通常の電子メールアドレスを追加できます。同じユーザに異なるパラメータセットを送信するには、受信者リストを右クリックするとコピー＆ペーストオプションが使用できます。**Add User/Group** をクリックして、以前にリストした受信者を表示することもできます。

グループ化されたレポート（サマリブレーク・レポートなど）をスケジュールし、そのレポートがデータベースをデータソースとして使用している場合は、レポートバーストのオプションも表示されます。レポートバーストのスクリーンショットと詳細については、レポートバーストを参照してください。

それ以外の場合は、受信者リストのみを含む次のダイアログが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



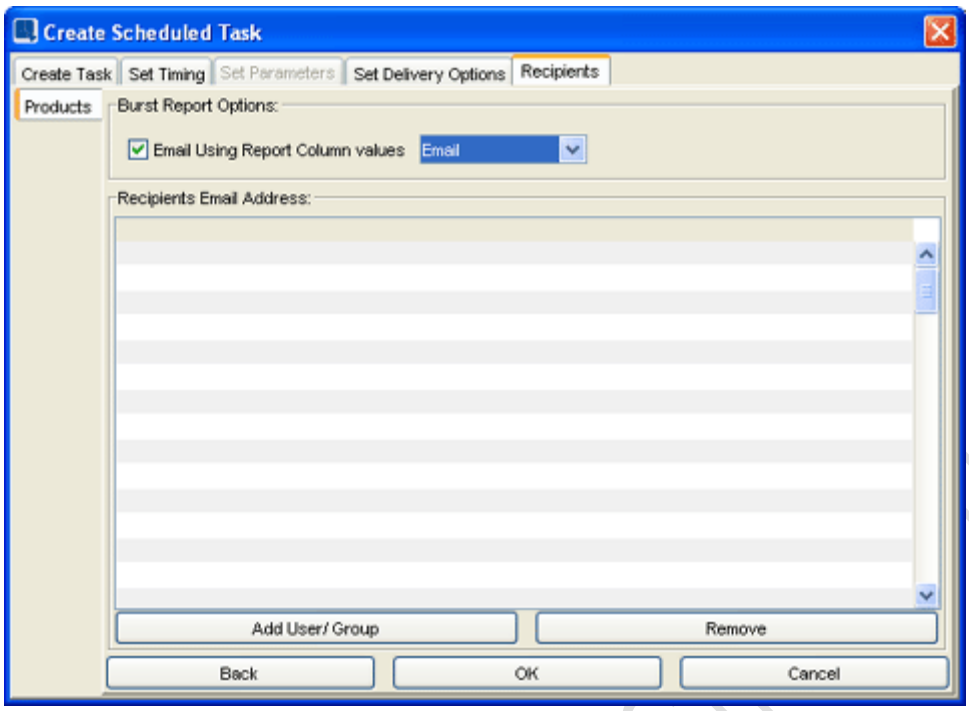
受信者リスト

3.13.2.5 レポートバースト

スケジュールされているレポートにグループ化されたデータ（サマリブレークレポートなど）が含まれている場合は、スケジュール配信を破棄するように選択できます。レポートがバーストすると、スケジュールジョブはデータのグループごとに個別のレポートを生成します。その後、各グループのレポートを特定の電子メールアドレスに配信し、エンドユーザが関連するデータを含むスケジュールされたレポートを受信できるようにします。大規模なレポートを一度実行し、エクスポートを小規模のレポートに分割することで、複数の小規模なスケジュールジョブを実行する場合に比べてパフォーマンスとセキュリティが大幅に向上します。レポートバーストオプションは、レポートのデータソースがデータベースの場合にのみ使用できます。

Recipients タブに、スケジュールの電子メール配信オプションを選択できるダイアログが表示されます。レポートにグループ化されたデータが含まれている場合は、上部にバーストオプションが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



レポートバーストの受信者リスト

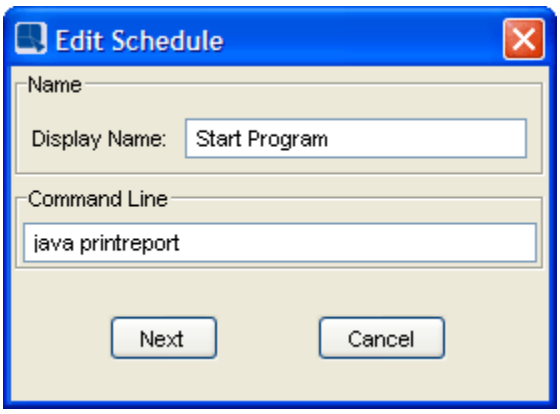
バーストレポート機能を使用するには、グループ行（行区切り）列に正しく対応する電子メールアドレスを含む列がレポートに存在している必要があります。たとえば、レポートが顧客 ID 列でグループ化された口座明細である場合、レポートには各顧客の電子メールアドレスを含む列も含まれている必要があります。バースト機能を使用するには、電子メール列を表示する必要はありません。

レポートが前述の要件を満たしている場合は、ダイアログの上部にある **Email Using Report Column Values** オプションを選択して、レポートバーストを有効にすることができます。これにより、レポートに対して 1 つのクエリが実行され、グループごとにレポートが分割され、各グループがエクスポートされ、グループごとに一覧表示された電子メールアドレスにエクスポートが配信されます。

3.13.2.6 コマンドのスケジューリング

追加のスケジューリングレポートでは、コマンドの実行をスケジュールすることもできます。schedule コマンドオプションを使用すると、EspressManager と同じディレクトリで実行されるコマンドを記述できます。これは非常に柔軟なオプションであり、どのコマンドでも使用できます。たとえば、カスタムコードを含む Java プログラムを実行して、レポート（Java Runreport など）を変更およびエクスポートすることができます。スケジュールされたコマンドを作成するには、Scheduler インタフェースから新しいスケジュールを作成し、**Insert Command** オプションを選択します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

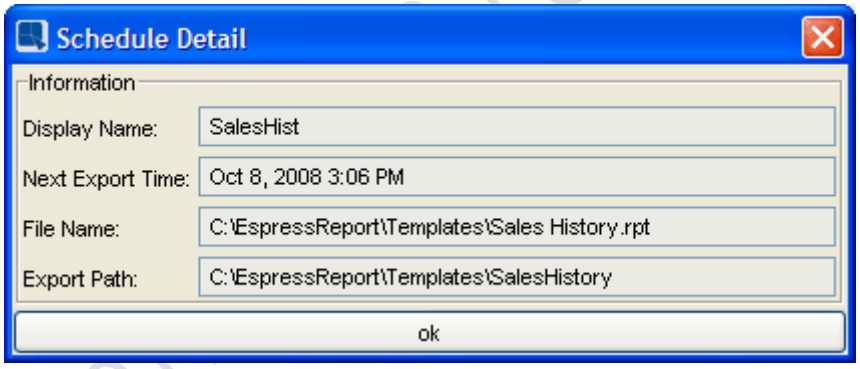


Command Options ダイアログ

この画面では、最初にスケジュールされたイベントの表示名を入力するよう求められます。次のオプションでは、スケジューラで実行するコマンドを入力できます。

3.13.2.7 スケジュールの表示と編集

スケジューラのメインウィンドウには、スケジュールされたすべてのタスクのリストが表示されます。特定のタスクの詳細については、そのタスクを選択して **Detail** ボタンをクリックします。これにより、スケジュールの詳細を表示する新しいウィンドウが表示されます。



Schedule Details ダイアログ

スケジュールを編集するには、メインウィンドウでスケジュールされたタスクを選択し、**Edit** ボタンをクリックします。これにより、スケジュールのダイアログが再び開き、異なる詳細を指定することができます。

スケジュールを実行するには、EspressManager が実行されている必要があります。スケジュールは EspressManager なしでは実行できません。EspressManager を終了すると、再起動するとスケジュールが再開されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4 プログラミングガイド

4.1 Report Viewer の概要

4.1.1 前書き

Report Designer で作成されたすべてのレポートは、後で表示できるさまざまな静的ファイル形式で保存できます。これらのフォーマットには、DHTML と PDF が含まれます。さらに、Report Designer には、レポートを.rpt ファイルとして保存するオプションが用意されています。

Report Viewer は、Web ブラウザを使用してレポートを動的に表示および操作できるアプレットです。Report Viewer は、Report Designer またはレポート API によって出力されたファイル（.rpt 形式）を読み取り、レポートを表示します。ファイルのサイズが小さいため、Web 上でレポートを配布するのに適しています。データは EspressManager から Report Viewer に転送されながら暗号化されます。したがって、機密情報にある程度のセキュリティがあります。

.rpt 形式のファイルは、Report Viewer を使用して表示されます。ブラウザを使用して.rpt ファイルを含む Web ページを表示すると、（EspressManager を介して）Report Viewer によって最新のデータが自動的に取得されます。したがって.rpt ファイルを使用して、最新のレポートをリアルタイムでユーザーに提供することができます。

Report Viewer を使用して、レポートのさまざまなセクションやページに移動できます。ユーザーがデータ要素をクリックして関連 URL にジャンプできるようにする組み込みのコールバックメカニズムがあります。Report Viewer は純粋な Java で書かれているため、Java をサポートするすべてのプラットフォーム上で動作します。Report Viewer は、スケジュールされたリフレッシュ（デザイナによって指定された定期的な間隔でレポートのデータが更新される）と、ロード時にレポートのパラメータが提供されるパラメータサービングもサポートします。

4.1.2 Report Viewer の起動

Report Viewer を起動するには、Report Viewer をアプレットに埋め込むか、Report Designer の Preview タブをクリックします。

Report Viewer をアプレットに埋め込むには、次のコードを使用します：

```
<applet codebase = ".." code = "quadbase.reportdesigner.ReportViewer.Viewer.class" width = 100%
height = 100% archive="ReportAPIWithChart.jar">
        <PARAM name = "filename" value = "yourReport.rpt">
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
</applet>
```

filename のパラメータ値には、レポートを含むファイルの名前を指定します。ファイル名の値は、URL で指定することもできます。たとえば、リモートデータファイルにアクセスするために http://という接頭辞を付けることができます。Report Viewer で表示すると.rpt ファイルは指定されたデータソースに接続し、データを動的に取得してレポートを生成します。

サーバ側では、ReportAPIWithChart.jar ファイルを EspressReport Installation の lib/ディレクトリから、Web クライアントがアクセスできる Web サーバ上のディレクトリにコピーする必要があります。

4.1.3 Report Viewer のパラメータ

Report Designer を使用せずに、パラメータを使用して Report Viewer プレットにデータを渡すこともできます。つまり、Report Viewer を使用してデータファイルを直接表示したり、HTML コード内のデータ行の形式で直接データを渡したりすることができます。以下にパラメータの一覧を示します：

comm_protocol:

ファイアウォールがある場合、またはサーブレットとして動作している EspressManager に接続する場合に使用するプロトコル。

comm_URL:

ファイアウォールの場合や EspressManager がサーブレットとして実行されている場合に、EspressManager に接続するための URL。

servlet_context:

サーブレットとして動作する EspressManager に接続するためのコンテキスト。

server_address:

EspressManager 接続の IP アドレス。

server_port_number:

EspressManager 接続のポート番号。

Parameter Server:

データをクライアントにプッシュするパラメータサーバのポート番号。

RefreshInterval:

データを更新する（つまり、データソースからデータを取得する）間隔を秒単位で指定します。

FileName:

.rpt ファイルの名前を指定します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ReportData:

レポートを文字列として指定します。

SourceDB:

データベース情報を指定します。

SourceFile:

データファイル情報を指定します。

SourceData:

データ情報を指定します。

ReportType:

レポートの種類を指定します。

Aggregation:

集計方法を指定します。

ColInfo:

列マッピングを設定します。

RowBreak:

行区切りとして使用する列を指定します。

ColumnBreak:

列区切りとして使用する列を指定します。

ColumnBreakValue:

列ブレーク列の値として使用する列を指定します。

PrimaryKey:

主キーとして使用する列を指定します。

MasterKey:

マスターキーとして使用する列を指定します。

EspressManagerUsed:

false の場合、EspressManager は使用されません。デフォルトは true です。

RefreshData:

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

false の場合、レポートはリフレッシュできません。デフォルトは true です。

ExportEnabled:

false の場合、レポートをエクスポートできません。デフォルトは true です。

例：パラメータ RefreshInterval を使用すると、次の行を挿入できます：

```
<PARAM name="RefreshInterval" value="60">
```

上記の例では、Report Viewer アプレットは、指定されたデータソースから（EspressManager の助けを借りて）データをフェッチし、60 秒ごとにレポートをすべて透過的に再描画します。 頻繁に更新されるデータベースにアクセスする場合に便利です。

複数の値を持つことができる（または複数の値を持つ必要がある）パラメータを指定する場合、セパレータは空白（二重引用符なし）です。例えば：

```
<PARAM name="RowBreak" value="0 1 2">
```

上記の例では、行区切りは、列マッピングの列 0、列 1 および列 2（すなわち、最初の 3 つの列）に設定されています。 パラメータの値には、それらを区切るスペース（二重引用符なしの ""）があります。

4.1.4 Report Viewer のデータソースの指定

4.1.4.1 データベースから読み取られたデータ

Report Viewer を使用してデータベースから抽出したデータを含むレポートを表示するサンプル HTML コードを次に示します：

```
<applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class" width=640 height=480>
<PARAM name="sourceDB" value="jdbc:odbc:DataSource ,
        sun.jdbc.odbc.JdbcOdbcDriver,
        username ,
        password ,
        select * from products">

<PARAM name="ColInfo" value="0 1 3">
<PARAM name="ReportType" value="SummaryBreak">
<PARAM name="RowBreak" value="0">
<PARAM name="Aggregation" value="SUM">
</applet>
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ColInfo の引数は、レポートのマッピングを指定します。詳細については、EspressReport API 入門の列マッピングに関する章を参照してください。

引数 ReportType は、表示されるレポートのタイプを指定し、次のいずれかになります：

1. SimpleColumnar
2. CrossTab
3. MasterDetails

引数 RowBreak は、行区切りが適用される列を指定します。

引数 Aggregation は、適用される集計方法を指定します。次のいずれかを指定できます：

1. NONE
2. SUM
3. MAX
4. MIN
5. COUNT
6. AVG
7. FIRST
8. LAST
9. SUMSQUARE
10. VARIANCE
11. STDDEV
12. COUNTDISTINCT

4.1.4.2 データファイルから読み取られたデータ

この HTML コードは、データファイルのデータを使用してレポートを生成します。

```
<applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class" width=640 height=480>
<PARAM name="sourceFile" value="http://.../test.dat">
<PARAM name="ColInfo" value="1 0 2 3">
<PARAM name="ReportType" value="SimpleColumnar">
</applet>
```

4.1.4.3 引数から読み込まれるデータ

Report Viewer をデータファイルまたはデータベースからではなく HTML ファイルから直接読み込むことができます。

```
<applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class" width=640 height=480>
        <PARAM name="SourceData" value="int, string, int |
                value, name, vol |
                10, 'John', 20 |
                3, 'Mary', 30 |
                8, 'Kevin', 3 |
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            9, 'James', 22">

        <PARAM name="ColInfo" value="1 0 2">
        <PARAM name="ReportType" value="SimpleColumnar">
        </applet>
```

パラメータ SourceData の書式はデータファイルの書式と同じですが、各行は垂直バー"|"で終了します。

4.1.5 Report Viewer の使用

Report Viewer を使用すると、ポップアップメニューを使用してレポートを参照できます。レポートを右クリックすると、ポップアップメニューが表示され、レポートをトラバースすることができます。ポップアップメニューには、さまざまなオプションが表示されます：

Back:

前のレポートに戻ります。 このオプションは、ハイパーリンクをクリックすると表示されます。

Section:

さまざまなセクションをナビゲートするためのいくつかのオプションが含まれています。

First Section:

最初のセクションに移動します。

Previous Section:

前のセクションに移動します。

Next Section:

次のセクションに進みます。

Last Section:

最後のセクションに移動します。

Page:

さまざまなページをナビゲートするためのいくつかのオプションが含まれています。

First Page:

最初のページに移動します。

Previous Page:

前のページに移動します。

Next Page:

次のページに進みます。

Last Page:

最後のページに移動します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Output:

レポートを出力するさまざまな形式が含まれています。

Generate HTML (Single Page):

レポートを単一のファイルとして HTML にエクスポートします。

Generate HTML (Multiple Pages):

レポートを HTML に複数のファイル（ページあたり 1 ファイル）としてエクスポートします。

Generate DHTML (Single Page):

レポートを単一のファイルとして DHTML にエクスポートします。

Generate DHTML (Paginated Single Page):

レポートを DHTML に 1 つのファイルとしてエクスポートし、ブレークを使用してページをシミュレートします。

Generate DHTML (Multiple Pages):

レポートを DHTML に複数のファイル、ページあたり 1 ファイルとしてエクスポートします。

Generate PDF:

レポートを PDF にエクスポートします。

Generate CSV:

レポートを CSV にエクスポートします。

Generate EXCEL (XLS):

レポートを Excel（xls ファイル）にエクスポートします。

Generate EXCEL 2007 (XLSX):

レポートを Excel 2007（xlsx ファイル）にエクスポートします。

Generate TXT:

レポートを TXT にエクスポートします。

Generate RTF:

レポートを RTF にエクスポートします。

Generate XML (Data + Format):

完全なレポートを XML にエクスポートします。

Generate XML (Pure Data):

レポートデータを XML にのみエクスポートします。

Print:

レポートを印刷します。

Refresh:

データソースからデータを更新します。

Go To:

指定したページとセクションに移動します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Zoom:

レポートに適用するズーム率を指定します。

Sort by (ascend):

選択した列の昇順の値に基づいてレポートをソートします。

Sort by (descend):

選択した列の降順値に基づいてレポートをソートします。

Show/Hide Report Toolbar:

このオプションは、ビューアウィンドウの下部にナビゲーションツールバーを表示するか、非表示にします。

ポップアップメニューに加えて、以下も可能です：

1. セルに関連付けられているハイパーリンクを左クリックします（関連付けられた URL が.rpt ファイルでない場合、新しいブラウザウィンドウが作成されます)。ブラウザで戻るボタンを使用して前のレポートに戻ることはできません。
2. リンクに関連付けられたヒントテキストを表示するには、ハイパーリンクを含むセル上でマウスを動かします。

4.1.6 パラメータサーバ

パラメータサーバを使用すると、EspressReport Viewer は TCP/IP を使用して指定されたポートからの要求を聞き、データを動的に更新できます。構文は次のとおりです。

```
<applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class" width=640 height=480>
<PARAM name="ParameterServer" value="machine:portno">
</applet>
```

セキュリティ上の理由から、マシンは通常、Web サーバマシンと同じです。したがって、マシンを空のままにすることができます（つまり、value = "：portno")。パラメータサーバの詳細については、付録 A. パラメータサーバを参照してください。

4.1.7 EspressManager に接続する

4.1.7.1 アプリケーションとして実行する EspressManager

EspressManager は主にアプリケーションとして動作します。 Report Viewer を使用してアプリケーションとして動作する EspressManager に接続する場合は、HTML コードに次のタグを追加する必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
<PARAM name="server_address" value="Machine">
<PARAM name="server_port_number" value="PortNumber">
```

上記のパラメータを HTML コードに追加すると、Report Viewer が Machine 上で実行され、PortNumber で待機している EspressManager に接続されます。

4.1.7.2 サーブレットとして実行する EspressManager

EspressManager はサーブレットとしても実行できます。Report Viewer を使用してサーブレットとして動作する EspressManager に接続する場合は、HTML コードに次のタグを追加する必要があります。

```
<PARAM name="comm_protocol" value="servlet">
<PARAM name="comm_URL" value="http://Machine:PortNumber">
<PARAM name="servlet_context" value="EspressReport/servlet">
```

上記のパラメータを HTML コードに追加すると、http://Machine:PortNumber/EspressReport/servlet の EspressManager に Report Viewer が接続されます。

4.1.8 Swing バージョン

Report Viewer の JFC/Swing バージョンは、**EspressReport/lib** ディレクトリの **ReportViewer.jar** の代わりに **SwingReportViewer.jar** を参照するためにも使用できます。

4.1.9 ビューアからのエクスポート

このセクションは、アプレット内のビューアからのエクスポートにのみ関係します。Report Viewer API を使用してレポートを表示するコンポーネントを作成する場合、ユーザーはコンポーネントを右クリックしてさまざまなオプションのポップアップメニューを起動できます。オプションの 1 つ（**Output > Client > Generate...**）は、レポートをさまざまなファイル形式にエクスポートするためのものです。これにより、サーバ側でエクスポートされたファイルが作成されます。ただし、サーバ側の Java サーブレットを使用してエクスポートされたコンテンツをクライアントのブラウザにストリーミングする API 機能があります。この機能の使用方法については、次の情報を参照してください。

まず、エクスポートされたファイルをクライアント側にストリームするサーブレットをデプロイする必要があります。このサーブレットは、ViewerExportServlet と呼ばれ、**<EspressReport InstallDir>/ViewerExport/ViewerExportServlet.java** にあります。単にサーブレットをコンパイルし、サーブレットコンテナにデプロイするだけです。

次に、レポートコンポーネントを取得するコードでは、2 行のコードを追加する必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
Viewer viewer = new quadbase.reportdesigner.ReportViewer.Viewer();
Component comp = viewer.getComponent(report);
viewer.setExportServlet("http://host:port/servlet/ViewerExportServlet");
viewer.setDynamicExport(true, "host", port, "servlet/");
```

setExportServlet メソッドの引数は、デプロイされた ViewerExportServlet の URL の場所です。これら 2 つのメソッドの引数の詳細については、javadoc の仕様を参照してください。

ユーザーがレポートコンポーネントを表示すると、**Output** > **Client** > **Generate....**の下に新しいオプションが表示されます。これらのオプションはすべて、レポートのストリーミングされたコンテンツを含むポップアップブラウザーウィンドウになります。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.2 Page Viewer の概要

4.2.1 前書き

Report Viewer コンポーネントは、Web ブラウザを使用してレポートを表示および操作する機能をユーザーに提供します。ビューアはクライアント上でレポートテンプレートをロードし、最新のデータでユーザーに表示します。ただし、レポートに大量のデータが含まれている場合、Report Viewer は理想的なデプロイメントソリューションではありません。このような状況では、クライアントでレポート全体をロードすると、クライアントはすぐにメモリ不足になります。

このような状況の場合、EspressReport は Page Viewer という別のビューアを提供します。クライアントでレポート全体を読み込む代わりに、Page Viewer は、クライアントが一度に 1 ページのレポートのみを読み込むことを可能にするページ提供技術を採用しています。これにより、レポートの読み込み速度が向上し、クライアントのメモリが節約されます。

Page Viewer は、レポート内の一連のページの描画情報を含む一連のページ（.page）ファイルにレポートをエクスポートすることによって機能します。.page ファイルは、クライアントのビューアにロードされます。EspressManager を使用しているユーザーは、Page Viewer コードの一部としてレポートテンプレートを直接渡すことができます。レポートは最新のデータでサーバ上にエクスポートされ、クライアントは.page ファイルをロードします。EspressManager を使用せずに実行しているユーザーは、API または Report Designer のいずれかのビュー形式でレポートをエクスポートできます。これらのエクスポートされたファイルは、EspressManager なしで Page Viewer に直接ロードすることができます。

4.2.2 Page Viewer の起動

Page Viewer を起動するには、.rpt ファイル(EspressManager)またはエクスポートされたビュー(.view)ファイル（EspressManager の有無にかかわらず）を使用してアプレットに埋め込むことができます。Page Viewer を使用して、Report Designer のプレビューウィンドウから直接レポートを読み込むこともできます。

Page Viewer をアプレットに埋め込むには、次のコードを使用します。

```
<applet codebase = ".." code = "quadbase.reportdesigner.PageViewer.Viewer.class" width = 100% height = 100% archive="PageViewer.jar">

        <PARAM name = "filename" value = "yourReport.rpt">

</applet>
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ファイル名のパラメータ値には、レポートを含むファイルの名前を指定します。バックエンドで EspressManager を使用して Page Viewer を実行している場合は、レポートテンプレートの名前を指定できます。このパラメータにエクスポートされたビューファイルを使用することもできます。Page Viewer でビューファイルを使用する場合、EspressManager を実行する必要はありません。ファイル名の値は、URL で指定することもできます。たとえば、リモートデータファイルにアクセスするために http://という接頭辞を付けることができます。

また、アプリケーションで Page Viewer を使用することもできます。これを行うには、次のコードを使用して java.awt.Component オブジェクトを取得します。

```
quadbase.reportdesigner.PageViewer.Viewer.getComponent(java.awt.Frame frame, java.lang.String fileName, long bufferTimeInSec)
```

テンプレートファイル（またはビューファイル）を渡すだけでなく、QbReport オブジェクトを渡すこともできます。Viewer クラスを使用する代わりに、ViewerAPI クラスを使用することに注意してください。

```
quadbase.reportdesigner.PageViewer.ViewerAPI.getComponent(java.awt.Frame frame, QbReport report, long bufferTimeInSec, java.lang.String securityLevel);
```

上記の方法は、Page Viewer をコンポーネントとして取得するための最も簡単な方法です。このコンストラクタのパラメータまたは Page Viewer に関連するその他のメソッドの詳細については、EspressReport API 仕様を参照してください。

サーバ側では、EspressReport Installation の lib/ディレクトリから PageViewer.jar ファイルを Web クライアントからアクセス可能な Web サーバのディレクトリにコピーする必要があります。

4.2.2.1 Report Designer からの起動

Page Viewer は、Report Designer から呼び出すこともできます。大量のデータを含むレポートをプレビューすると、レポートを Page Viewer ウィンドウにロードできます。これにより、ユーザーは、すべてのデータをメモリにロードせずに、レポート全体をプレビューして表示することができます。この機能の詳細については、Page Viewer の使用を参照してください。

4.2.3 Page Viewer のパラメータ

Report Designer を使用せずに、パラメータを介して Page Viewer アプレットにデータを渡すこともできます。つまり、Page Viewer を使用してデータファイルを直接表示したり、HTML コード内のデータ行の形でデータを直接渡したりすることができます。以下にパラメータの一覧を示します。

Page Parameters

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

comm_protocol:

ファイアウォールがある場合、またはサーブレットとして動作している EspressManager に接続する場合に使用するプロトコル。

comm_URL:

ファイアウォールの場合や EspressManager がサーブレットとして実行されている場合に、EspressManager に接続するための URL。

servlet_context:

サーブレットとして動作する EspressManager に接続するためのコンテキスト。

server_address:

EspressManager 接続の IP アドレス。

server_port_number:

EspressManager 接続のポート番号。

server_hosts:

EspressManager ホストが持つことができる他の名前は、カンマをセパレータとして使用して指定できます。

FileName:

.rpt ファイルの名前を指定します。

EspressManagerUsed:

false の場合、EspressManager は使用されません。デフォルトは true です。Page Viewer を使用してレポートテンプレートを実行する場合、EspressManager を使用する必要があります。ビューファイルが使用されている場合、EspressManager は実行されている必要はありません。

BufferTime:

EspressManager から新しいページを取得する頻度を制御する時間（秒単位）。.page ファイルのローカルバージョンを調べます（利用可能な場合）。最後に更新されてからの時間が bufferTimeInSec パラメータを超える場合、新しいバージョンのページが EspressManager から再度要求されます。それ以外の場合は、.page ファイルのローカルバージョンを使用します。.page ファイルのローカルバージョンが存在しない場合、EspressManager からページが取得されます。

DataHintOffsetX:

ユーザーがグラフ上のデータポイント（棒グラフ上の棒グラフまたは折れ線グラフ上のポイント）上にマウスを移動すると、データポイントの詳細を示すヒントボックスが表示されます。このパラメータは、データポイントに対するヒントボックスの水平オフセット位置を設定しま

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

す。

DataHintOffsetY:

ユーザーがグラフ上のデータポイント（棒グラフ上の棒グラフまたは折れ線グラフ上のポイント）上にマウスを移動すると、データポイントの詳細を示すヒントボックスが表示されます。このパラメータは、データポイントに対するヒントボックスの垂直オフセット位置を設定します。

DataHintBgColor:

ユーザーがグラフ上のデータポイント（棒グラフ上の棒グラフまたは折れ線グラフ上のポイント）上にマウスを移動すると、データポイントの詳細を示すヒントボックスが表示されます。このパラメータは、ヒントボックスの背景色を設定します。

DataHintFontColor:

ユーザーがグラフ上のデータポイント（棒グラフ上の棒グラフまたは折れ線グラフ上のポイント）上にマウスを移動すると、データポイントの詳細を示すヒントボックスが表示されます。このパラメータは、ヒントボックスのフォントの色を設定します。

DataHintFont:

ユーザーがグラフ上のデータポイント（棒グラフ上の棒グラフまたは折れ線グラフ上のポイント）上にマウスを移動すると、データポイントの詳細を示すヒントボックスが表示されます。このパラメータは、ヒントボックスのフォントタイプを設定します。

Printing:

有効な値："true"|"false"

印刷オプションをポップアップメニューに表示するかどうかを設定します。

RefreshData:

有効な値："true"|"false"

false の場合、レポートはリフレッシュできません。デフォルトは true です。

PopupMenu:

有効な値："true"|"false"

ポップアップメニューを表示するかどうかを設定します。

EnableExport:

有効な値："true"|"false"

false の場合、レポートはエクスポートできません。デフォルトは true です。

ShowDataHint:

有効な値："true"|"false"

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

グラフのデータポイント上にマウスを置いたときにデータヒントボックスを表示するかどうかを設定します。

ShowLinkHint:

有効な値："true"|"false"
ハイパーリンク上にマウスを置いたときにデータヒントボックスを表示するかどうかを設定します。

例：パラメータ BufferTime を使用すると、次の行を挿入できます。

```
<PARAM name="DataHintBgColor" value="ff00aa">
```

4.2.4 Page Viewer の使用

Page Viewer を使用すると、ポップアップメニューを使用してレポートのページを参照できます。レポートを右クリックすると、レポートをトラバースするために使用できるポップアップメニューが表示されます。ポップアップメニューには、さまざまなオプションが表示されます。

Back:

前のレポートに戻ります。 このオプションは、ハイパーリンクをクリックすると表示されます。

Section:

さまざまなセクションをナビゲートするためのさまざまなオプションが含まれています。

First Section:

最初のセクションに移動します。

Previous Section:

前のセクションに移動します。

Next Section:

次のセクションに進みます。

Last Section:

最後のセクションに移動します。

Page:

さまざまなページをナビゲートするためのさまざまなオプションが含まれています。

First Page:

最初のページに移動します。

Previous Page:

前のページに移動します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Next Page:

次のページに進みます。

Last Page:

最後のページに移動します。

Output:

レポートを出力するさまざまな形式が含まれています。

Generate HTML (Single Page):

レポートを単一のファイルとして HTML にエクスポートします。

Generate HTML (Multiple Pages):

レポートを HTML に複数のファイル（ページあたり 1 ファイル）としてエクスポートします。

Generate DHTML (Single Page):

レポートを単一のファイルとして DHTML にエクスポートします。

Generate DHTML (Paginated Single Page):

レポートを DHTML に 1 つのファイルとしてエクスポートし、ブレークを使用してページをシミュレートします。

Generate DHTML (Multiple Pages):

レポートを DHTML に複数のファイル、ページあたり 1 ファイルとしてエクスポートします。

Generate PDF:

レポートを PDF にエクスポートします。

Generate CSV:

レポートを CSV にエクスポートします。

Generate EXCEL (XLS):

レポートを Excel（xls ファイル）にエクスポートします。

Generate EXCEL 2007 (XLSX):

レポートを Excel 2007（xlsx ファイル）にエクスポートします。

Generate TXT:

レポートを TXT にエクスポートします。

Generate RTF:

レポートを RTF にエクスポートします。

Generate XML (Data + Format):

完全なレポートを XML にエクスポートします。

Generate XML (Pure Data):

レポートデータのみを XML にエクスポートします。

Print:

レポートを印刷します。

Refresh:

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

データソースからデータを更新します。

Go To:

指定したページとセクションに移動します。

Zoom:

レポートに適用するズーム率を指定します。

Sort by (ascend):

選択した列の昇順の値に基づいてレポートをソートします。

Sort by (descend):

選択した列の降順値に基づいてレポートをソートします。

Show/Hide Report Toolbar:

これは、ビューアウィンドウの下部にナビゲーションツールバーを表示するか、非表示にします。

ポップアップメニューに加えて、以下も可能です：

- 左クリックしてセルに関連付けられたハイパーリンクにジャンプします（関連付けられた URL が.rpt ファイルでない場合、新しいブラウザウィンドウが作成されます。ブラウザで Back ボタンを使用して前のレポートに戻ることはできません)。
- リンクに関連付けられたヒントテキストを表示するには、ハイパーリンクを含むセル上でマウスを動かします。

4.2.5 バッファ時間

Page Viewer に関連付けられたビュー（.view）ファイルとページ（.page）ファイルで動作する問題は、ファイルシステム上のレポートページをページ単位でキャッシュしてメモリを節約しパフォーマンスを向上させると、Page Viewer のファイルデータが時代遅れになる可能性があります。

Page Viewer は、新しいビューファイルをサーバから取得する必要があるまでの経過時間を調整するメカニズムを提供します。バッファ時間とも呼ばれるこの時間間隔は、Page Viewer アプレットで BufferTime パラメータを設定することで調整できます。BufferTime パラメータに値として秒単位でバッファ時間を渡します。VIEW ファイルの現在のバージョンが指定されたバッファ時間よりも古い場合、Page Viewer は新しいページを取得します。

また、Page Viewer には、3 日以上更新されていない古いビューファイルとページファイルを自動的に削除する組み込みのメカニズムがあります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.2.6 EspressManager に接続する

4.2.6.1 アプリケーションとして実行する EspressManager

EspressManager は主にアプリケーションとして動作します。 Page Viewer を使用してアプリケーションを実行する EspressManager に接続する場合は、HTML コードに次のタグを追加する必要があります：

```
<PARAM name="server_address" value="Machine">
<PARAM name="server_port_number" value="PortNumber">
```

上記のパラメータを HTML コードに追加すると、Page Viewer が Machine 上で実行されている PortNumber に接続する EspressManager に接続されます。

4.2.6.2 サーブレットとして実行する EspressManager

EspressManager はサーブレットとしても実行できます。Page Viewer を使用してサーブレットとして動作する EspressManager に接続する場合は、HTML コードに次のタグを追加する必要があります：

```
<PARAM name="comm_protocol" value="servlet">
<PARAM name="comm_URL" value="http://Machine:PortNumber">
<PARAM name="servlet_context" value="EspressReport/servlet">
```

上記のパラメータを HTML コードに追加すると、
http://Machine:PortNumber/EspressReport/servlet の EspressManager に Page Viewer が接続されます。

4.2.7 Swing バージョン

EspressReport/lib ディレクトリの **PageViewer.jar** の代わりに **SwingPageViewer.jar** を参照するために、Page Viewer の JFC/Swing バージョンを使用することもできます。

4.2.8 ビューアからのエクスポート

このセクションは、アプレット内の Page Viewer からのエクスポートにのみ関係します。Page Viewer API を使用してレポートを表示するコンポーネントを作成する場合、ユーザーはコンポーネントを右クリックして、さまざまなオプションのポップアップメニューを起動できます。オプション（**Output** ＞ **Server** ＞ **Generate**）の 1 つは、レポートをさまざまなファイル形式にエクスポートするためのオプションです。これにより、サーバ側でエクスポートされたファイルが作成されます。ただし、サーバ側 Java サーブレットを使用してエクスポートされたコンテンツをクライアントのブラウザにストリーミングする API 機能があります。この機能の使用方法は次のとおりです：

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

まず、エクスポートされたファイルをクライアント側にストリームするサーブレットをデプロイする必要があります。このサーブレット（**ViewerExportServlet**）は、**<EspressReport InstallDir>/ViewerExport/ViewerExportServlet.java** にあります。サーブレットをコンパイルし、サーブレットコンテナにデプロイするだけです。

次に、レポートコンポーネントを取得するコードでは、4 行のコードを追加する必要があります：

```
Viewer viewer = new quadbase.reportdesigner.PageViewer.Viewer();
Component comp = viewer.getComponent((Applet)null, reportTemplate, 0);
viewer.setExportServlet("http://host:port/servlet/ViewerExportServlet");
viewer.setDynamicExport(true, "host", port, "servlet/");
```

reportTemplate は、レポートテンプレートファイルの名前とパスを含む文字列です。 また、次のコードを使用して QbReport オブジェクトを渡すこともできます：

```
ViewerAPI viewer = new quadbase.reportdesigner.PageViewer.ViewerAPI();
Component comp = viewer.getComponent((Applet)null, report, 0, null);
viewer.setExportServlet("http://host:port/servlet/ViewerExportServlet");
viewer.setDynamicExport(true, "host", port, "servlet/");
```

レポートは **QbReport** オブジェクトです。

setExportServlet メソッドの引数は、デプロイされた **ViewerExportServlet** の URL の場所です。これら 2 つのメソッドの引数の詳細については、javadoc の仕様を参照してください。

次に、ユーザーがレポートコンポーネントを表示すると、**Output** > **Client** > **Generate** の下に新しいオプションが表示されます。これらのオプションはすべて、レポートのストリーミングされたコンテンツを含むポップアップブラウザウィンドウになります。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.3 EspressReport レポート API

4.3.1 導入とセットアップ

EspressReport は使いやすいアプリケーションプログラミングインターフェース（API）を提供し、ユーザーはサーバ側またはクライアント側のアプリケーション（およびアプレット）内のレポートを作成およびカスタマイズできます。これは 100%純粋な Java で書かれているため、ほとんどまたは全く変更を加えずにどのプラットフォームでも実行できます。レポートのすべての部分は、API を使用してカスタマイズできます。1 行のコードを使ってレポートを生成できます。

主なクラス **QbReport** は **java.awt.Component** を拡張します。このコンポーネントには、**quadbase.reportdesigner.ReportAPI**、**quadbase.reportdesigner.ReportElements**、**quadbase.reportdesigner.ReportViewer**（およびその Swing カウンターパート **quadbase.reportdesigner.ReportViewer.swing**）、**quadbase.reportdesigner .lang** と **quadbase.reportdesigner.util**。このドキュメントの残りの部分では、API の構成要素とその使用方法について説明します。完全な API ドキュメントは **help/apidocs/index.html** にあります。

API を使用するには、**ReportAPIWithChart.jar** および **qblicense.jar**（**EspressReport/lib** ディレクトリにあります）を CLASSPATH に追加します。SOAP データソースまたは XML ファイルを使用してデータを読み書きする場合は、**axercesImpl.jar** と **xml-apis.jar**（同じディレクトリにもあります）を CLASSPATH の順番で追加する必要があります。チャートまたは画像を PNG 形式にエクスポートする場合は、CLASSPATH に **ExportLib.jar** も含める必要があります。レポートを Excel ファイル（つまり.xls ファイル）にエクスポートするには、CLASSPATH に **poi.jar** を含める必要があります。パラメータ化されたデータベースクエリをデータソースとして使用する場合は、**jsqlparser.jar** を CLASSPATH に追加します。MS Excel 2007 形式（OOXML- 拡張子.xlsx）にエクスポートする場合は、**poi-ooxml-schemas.jar**、**poi-ooxml.jar**、**geronimo-stax-api.jar**、**xmlbeans.jar**、**dom4j.jar** のファイルも CLASSPATH に含める必要があります。アプリケーションが Windows または Solaris マシン上にある場合、（プラットフォームに応じて）次の環境変数を追加する必要があります：

```
For Windows) set PATH=%PATH%;&lt;path to EspressReport root directory&gt;¥lib

For Solaris) export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:&lt;path to EspressReport root directory&gt;/lib
```

アプリケーションによっては、サードパーティの jar ファイルも必要になることに注意してください。たとえば、データベースへの JDBC 接続を作成する場合は、JDBC jar ファイルも CLASSPATH に含める必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.3.2 レポート API を使用するための推奨アプローチ

EspressReport は、レポートをゼロからプログラムで生成するための API を提供します。しかし、これには大量のコードが含まれています。Report Designer を使用し、最初にレポートファイル（.rpt ファイル）を作成することをお勧めします。これはテンプレートとして機能します。このテンプレートは QbReport コンストラクタで渡すことができ、テンプレートの外観を新しく作成したレポートオブジェクトに適用することができます。したがって、レポート生成時には、外観の大部分が既に設定されていました。また、API を使用してレポートファイル（.RPT または.PAK ファイル）を開くこともできます。いずれの方法を使用しても、プロパティを変更、追加、または削除するコードを記述できます。このアプローチにより、コーディング時間が節約され、パフォーマンスも向上します。

EspressReport Report API を使用する場合、EspressManager に接続するオプションがあります。Report Designer を使用する場合は接続が必要ですが、EspressReport Report API を使用するアプリケーションを実行する場合は、EspressManager に接続する必要はありません。EspressManager に接続せず、アーキテクチャ内の別のレイヤーを避けることをお勧めします。詳細については、次のセクションを参照してください。

マニュアルに記載されているすべての例とコードは、テンプレートベースのアプローチと EspressManager への接続の 2 つの推奨事項に従います。特に指定のない限り、すべてのコード例でテンプレートを使用し（テンプレートは対応する章からダウンロードできます）、EspressManager には接続しません。

また、EspressReport Report API を使用するアプレットを使用する場合は、ブラウザに 1.5 以上の JVM プラグインが必要です。

4.3.3 EspressManager とのインタラクション

レポートを作成して使用する方法の詳細を説明する前に、アプリケーションで EspressManager を使用するオプションについて調べてみましょう。EspressReport は、一般的に EspressManager と組み合わせて使用されます。レポートコンポーネントは EspressManager に接続して、データベースにアクセスするためのファイルを読み書きし、特定の高度な機能（集計など）に必要なデータ前処理を実行します。ただし、レポートコンポーネントは、EspressManager を使用せずにファイル I/O とデータベースアクセスを直接実行するスタンドアロンモードでも使用できます。

たとえば、アプレットまたはアプリケーションが Client マシン上で実行されている可能性があり、DbMachine マシン上のデータベースからのデータが必要な場合があります。ただし、DbMachine マシンがファイアウォールの背後にあるか、セキュリティ制限のため DbMachine マシンを Client マシンから直接接続できないことがあります。EspressManager は Server マシン上で実行でき、アプレット/アプリケーションは Server マシン上の EspressManager に接続できます。JDBC を使用して Server マシンからマシン DbMachine に接続し、データを取得できます。次に、データが Client マシンに渡され、レポートが生成されます。これは、Client マシンからデータを保護し、アクセスできないようにし、す

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

べての接続を Server マシン（EspressManager を実行しているマシン）を経由させたい場合に便利です。このオプションを利用して、すべてのクライアントのログに EspressReport を介してデータを保存することもできます（EspressManager の起動時にログファイルを作成することができます。ログファイルは espressmanager.log です）。この機能にはコストがかかります。コードが EspressManager を介してデータに接続している（つまり、別のレイヤーが追加されている）ため、パフォーマンスはわずかに低下します。

デフォルトでは、レポートコンポーネントには EspressManager が必要です。モードを変更するには、（QbReport オブジェクトが作成される前に）アプレット/アプリケーションの先頭で **QbReport** クラス静的メソッドを使用します：

```
static public void setEspressManagerUsed(boolean b)
```

アプリケーションとアプレットの両方を、EspressManager にアクセスするかどうかにかかわらず実行できます。通信は http プロトコルを使用して行われます。サーバの場所は、API コードで渡される IP アドレスとポート番号によって決まります。EspressManager への接続方法は次のとおりです：

4.3.4 EspressManager に接続する

4.3.4.1 アプリケーションとして実行する EspressManager

EspressManager は主にアプリケーションとして動作します。ReportAPI を使用してアプリケーションとして動作する EspressManager に接続する場合は、API メソッドを使用して、EspressManager が配置されている IP アドレス/マシン名と EspressManager が待機しているポート番号を指定できます。

接続情報を設定するには、次の 2 つの API メソッドを使用できます：

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

たとえば、次のコードの行があります。

```
QbReport.setServerAddress("Machine");
QbReport.setServerPortNumber(PortNumber);
```

Machine 上で動作し、ポート番号で待機している Express Manager に接続します。

EspressManager の接続情報が指定されていない場合、コードはローカルマシン上の EspressManager に接続し、デフォルトポート番号（22071）を待機します。

これらのメソッドは、**QbReport**、**QbChart**、**QbReportDesigner**、および **QbScheduler** クラスに存

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

在します。

4.3.4.2 サーブレットとして実行する EspressManager

EspressManager はサーブレットとしても実行できます。Report API を使用してサーブレットとして動作する EspressManager に接続するには、次のメソッドを使用する必要があります：

```
public static void useServlet(boolean b);
public static void setServletRunner(String comm_URL);
public static void setServletContext(String context);
```

たとえば、次のコードの行があります。

```
QbReport.useServlet(true);
QbReport.setServletRunner("http://Machine:PortNumber");
QbReport.setServletContext("EspressReport/servlet");
```

http://Machine:PortNumber/EspressReport/servlet で動作する EspressManager に接続します。

これらのメソッドは、**QbReport**、**QbChart**、**QbReportDesigner**、**QbScheduler** クラスに存在します。

4.3.5 API の使用

次のセクションでは、API の使用方法について詳しく説明します。ここでも、API の例とコードは、上記の推奨事項（テンプレートベースおよび EspressManager なし）を使用して設計されています。

特に明記しない限り、すべての例では、**<ERInstallDir>/help/Samples/DataSources/database** ディレクトリにある Woodview HSQL データベースを使用しています。例を実行するには、データベース HSQL JDBC ドライバ（**hsqldb.jar**）をクラスパスに追加する必要があります。ドライバは **<ERInstall>/lib** ディレクトリにあります。

また、すべての API の例では、マニュアルの中核コードが表示されます。例をコンパイルするには、CLASSPATH に **ReportAPIWithChart.jar** と **qblicense.jar** が含まれていることを確認してください。

API メソッドの詳細については、API のドキュメントを参照してください。

4.3.5.1 レポートの読み込み

レポートは、RPT 形式（プロプライエタリ形式）または XML 形式を使用してファイルに保存できます。RPT/XML ファイルには、実際のデータを除くすべてのレポート情報が格納されます（ただし、RPT ファイル内のデータ全体を保存するオプションがあります）。この形式を使用して、レポートオブジェクトを再構築することができます。RPT/XML ファイルを開くたびに、データは元のデータソースから自

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

動的にリロードされます。

RPT/XML ファイルにはデフォルトでデータが**含まれていない**ことに注意することが重要です。レポートテンプレート情報（つまり、レポートの外観）と共に指定されたデータソースが含まれます。したがって、レポートのデータは、RPT/XML ファイルをロードするときにデータソースを照会することによって取得されます。この形式は、Report Designer または **QbReport** クラスで提供されるエクスポート方法を使用して取得できます。RPT ファイルに完全なデータを含めるように選択することもできます（データソース情報とともに）。

アプレットまたはアプリケーションとして実行できる次の例は、RPT ファイルを読み取り、レポートを再構成します：

```
Component doOpeningTemplate(Object parent) {

        //Do not use EspressManager
        QbReport.setEspressManagerUsed(false);

        //Open the template
        QbReport report = new QbReport(parent, //container
                        "OpeningTemplate.rpt"); //template

        //Show report in Viewer
        return (new Viewer().getComponent(report));
}
```

フルソースコード
エクスポートした結果

この例のコンストラクタは次のとおりです：

```
public QbReport(Object parent, String file);
```

parent はコンテナで、ファイルは RPT/XML ファイル名です。

ファイル名は、URL 文字列または相対/絶対パスを使用して指定できます。

パス名は、EspressManager が現在使用されていない場合、現在のアプリケーションディレクトリまたは EspressManager の現在のディレクトリからの相対パスとして解釈されます。

4.3.5.1.1 サブレポート、チャート、ドリルダウンレポート

レポートテンプレートには、チャート、サブレポート、および/またはドリルダウンレポートが含まれ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ている場合があります。これらの付属テンプレートは、メインレポートテンプレートとは別に保存されます。 チャートテンプレートは、チャートディレクトリ、サブレポートディレクトリのサブレポートテンプレート、ドリルダウンディレクトリのドリルダウンテンプレートに保存されます。相対 URL（EspressReport インストールディレクトリからの相対パス）とテンプレート名が指定され、メインレポートテンプレートに格納されます。

アプレットまたはアプリケーションとして実行できる次の例は、RPT ファイルを読み取り、Chart Path を設定します：

```
Component doSetChartPath(Object parent)  {

        //Do not use EspressManager
        QbReport.setEspressManagerUsed(false);

        //Open the template
        QbReport report = new QbReport (parent,                    //container
                                        "setChartPath.rpt");       //template

        //Set Chart Path
        report.setChartPath(".");

        //Show report in Viewer
        return (new Viewer().getComponent(report));
}
```

フルソースコード
エクスポートされた結果

上記のコードでは、チャートテンプレートファイルの場所が設定されているため、クラスファイルの作業ディレクトリの下にチャートサブディレクトリを置く必要はありません。

次の方法も同様です：

```
QbReport.setDrillDownPath(String directory);
QbReport.setSubReportPath(String directory);
QbReport.setImagePath(String directory);
```

4.3.5.1.2 PAK ファイル

PAK 形式でレポートをパックし、PAK ファイルとして展開することもできます。この代替方法は、メインレポートに関連付けられているすべてのチャート、サブレポート、ドリルダウン、および画像ファイルを 1 つの.pak ファイルに含めることで、展開手順を簡略化します。あなたのコードは.pak ファイ

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ル名（.rpt/.xml ではなく）を使用してレポートを作成できます。.pak ファイルのアプローチでは、チャート、サブレポート、および/またはドリルダウンテンプレートが配置されているディレクトリを指定する必要はありません。

4.3.5.1.3 バックアップデータ

デフォルトでは、レポートテンプレートは常に 2 行のバックアップデータとともに保存されます。これは、データソースが存在しない場合でも、**QbReport** オブジェクトの作成にテンプレートを使用できるようにするためです。既存のテンプレートに基づいて **QbReport** オブジェクトを作成すると、テンプレートで指定されたデータソースが検索されます。データソースが見つからない場合、バックアップデータが使用されます。ただし、次のコンストラクタを使用してデータソースを検索する代わりに、API にバックアップデータを使用させることができます。

```
public QbReport(Object parent, String file, boolean isEnterpriseServer,
                boolean optimizeMemory, boolean multiPageExp,
                boolean useBackupData);
```

API にバックアップデータの使用を強制するには、上記コンストラクタの最後の引数だけを true に設定する必要があります。 他のブール引数の値は関係ありません。

4.3.5.1.4 パラメータ化されたレポート

レポートには、データを何らかの形で制限するか、追加情報を提供するためのパラメータも含めることができます。通常、データソースはクエリ（または IN）パラメータを使用してデータを制限し、パラメータ化された式はレポート内にさらにデータを含めるために使用されます。

クエリパラメータは、単一値と複数値の両方のパラメータタイプにできますが、数式パラメータは単一値のみです。

パラメータ化されたテンプレートが次のコンストラクタを使用して開かれたとき：

```
QbReport(Object parent, String templateName);
```

ダイアログボックスが表示され、パラメータの値を要求します。このダイアログボックスは、レポートのプレビュー時に Designer に表示されるダイアログボックスと同じです。

オブジェクト配列

パラメータ化されたレポートは、ダイアログボックスに任意の値のプロンプトを表示せずに開くこともできます。これは、クエリパラメータ用と式パラメータ用の 2 つのオブジェクト配列を渡すことによって行うことができます。配列の各要素は、その特定のパラメータの値を表します。

配列の順序は、Designer でパラメータが作成された順序と一致する必要があります。正しい結果を得るには、値のデータ型もパラメータのデータ型と一致する必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

クエリパラメータは、複数値のパラメータタイプでもあります。多値パラメータの場合、ベクトルがオブジェクト配列に渡されます。ベクトルには、多値パラメータのすべての値が含まれます。

以下の例は、アプレットまたはアプリケーションとして実行でき、テンプレートを開くときにパラメータ値を渡します：

```
Component doObjectArray(Object parent) {

        //Do not use ERES Server
        QbReport.setEspressManagerUsed(false);

        //Object array for Query Parameters
        Vector vec = new Vector();
        vec.add("CA");
        vec.add("NY");

        GregorianCalendar beginDate = new GregorianCalendar(2001, 0, 4);
        GregorianCalendar endDate = new GregorianCalendar(2003, 1, 12);

        long beginLong = beginDate.getTimeInMillis();
        long endLong = endDate.getTimeInMillis();

        Date beginDateTime = new Date(beginLong);
        Date endDateTime = new Date(endLong);

        Object queryParams[] = new Object[3];
        queryParams[0] = vec;
        queryParams[1] = beginDateTime;
        queryParams[2] = endDateTime;

        //Object array for Formula Parameter
        Object formulaParams[] = new Object[1];
        formulaParams[0] = "Sarat";

        //Open the template
        QbReport report = new QbReport(parent, //container
                        "ObjectArray.rpt", //template
                        queryParams, //Query Parameters
                        formulaParams); //Formula Parameters
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        //Show report in Viewer
        return (new Viewer().getComponent(report));
}
```

フルソースコード
エクスポートした結果

getAllParameters メソッド

上記に加えて、**getAllParameters** メソッドを使用してパラメータを渡すこともできます。**getAllParameters** メソッドは、レポート内のすべてのパラメータのリストを返します（これには、共有されていないサブレポートのパラメータが含まれます）。各パラメータが取得され、値が設定されます。

アプレットまたはアプリケーションとして実行できる次の例は、上記と同じレポートを取得し、**getAllParameters** メソッドを使用してパラメータを渡します：

```
Component doGetAllParameters(Object parent) {

        //Do not use EspressManager
        QbReport.setEspressManagerUsed(false);

        //Object array for Query Parameters
        Vector vec = new Vector();
        vec.addElement("CA");
        vec.addElement("NY");

        GregorianCalendar beginDate = new GregorianCalendar(2001, 0, 4);
        GregorianCalendar endDate = new GregorianCalendar(2003, 1, 12);

        long beginLong = beginDate.getTimeInMillis();
        long endLong = endDate.getTimeInMillis();

        Date beginDateTime = new Date(beginLong);
        Date endDateTime = new Date(endLong);

        Object queryParams[] = new Object[3];
        queryParams[0] = vec;
        queryParams[1] = beginDateTime;
        queryParams[2] = endDateTime;

        //Object array for Formula Parameter
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        Object formulaParams[] = new Object[1];
        formulaParams[0] = "Sarat";

        //Open the template with backup data
        QbReport report = new QbReport(parent, //container
                        "ObjectArray.rpt", //template
                        false, false, false, true);

        //Pass in parameters using getAllParameters
        report.getAllParameters().get(0).setValues((Vector) queryParams[0]);
        report.getAllParameters().get(1).setValue(queryParams[1]);
        report.getAllParameters().get(2).setValue(queryParams[2]);
        report.getAllParameters().get(3).setValue(formulaParams[0]);

        try {
                //Get the data for the report
                report.refreshWithOriginalData();
        } catch (Exception ex)
        {
                ex.printStackTrace();
        }

        //Show report in Viewer
        return (new Viewer().getComponent(report));
}
```

フルソースコード
エクスポートした結果

Report Designer、Page Viewer、およびメニューページのパラメータプロンプトダイアログには、パラメータが順番に表示されます。順序付けられたパラメータのリストを取得するには、パラメータ値を true に設定してオーバーロードされたメソッドを使用します。

```
report.getAllParameters(boolean ordered)
```

この方法の結果は 2 つの性質を維持する。まず、数式パラメータが常に最初に返されます。第 2 に、カスケードパラメータの順序付けが維持される。カスケードパラメータの詳細は、カスケーディングパラメータを参照してください。

4.3.5.1.5 安全なレポート

レポートの情報は、レポートの作成時にセキュリティパラメータを渡すことで変更できます。セキュリ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ティレベルは Designer で作成され、適切なセキュリティレベルは API を使用して渡されます。

アプレットまたはアプリケーションとして実行できる次の例は、West 以外のすべての地域の販売情報を示す安全なレポートを開きます：

```
Component doSecurity(Object parent) {

	//Do not use EspressManager
	QbReport.setEspressManagerUsed(false);

	//Open the template
	QbReport report = new QbReport(parent, //container
			"Security.rpt"); //template

	try {
		//Set Security Level
		report.setSecurityLevel("NoWest");
	} catch (Exception ex)
	{
		ex.printStackTrace();
	}

	//Show report in Viewer
	return (new Viewer().getComponent(report));
}
```

フルソースコード
エクスポートした結果

また、**Properties** オブジェクトを使用してセキュリティレベルを渡すこともできます。これは、クエリ/式パラメータが安全である場合に特に便利です。

アプレットまたはアプリケーションとして実行できる次の例は、West 以外のすべての地域の販売情報を示す安全なレポートを開きます：

```
Component doSecurityProperties(Object parent) {

	//Do not use EspressManager
	QbReport.setEspressManagerUsed(false);

	//Set up Properties
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        Properties props = new Properties();
        props.put("security level", "NoWest");

        //Open the template
        QbReport report = new QbReport(parent, //container
                        "Security.rpt", //template
                        props); //properties

        //Show report in Viewer
        return (new Viewer().getComponent(report));
}
```

フルソースコード
エクスポートした結果

4.3.5.2 レポートテンプレートの適用

QbReport オブジェクトを最初から作成すると（レポートの作成を参照)、デフォルトの属性を使用してレポートが作成されます。ただし、レポートテンプレート（.rpt または.xml）を使用して、レポートの作成時にユーザー定義の属性を指定できます。ほとんどの属性（データソースとレポートタイプを除く）はテンプレートから抽出され、**QbReport** オブジェクトに適用されます。テンプレート名は通常、**QbReport** コンストラクタの最後の引数として表示されます。

また、**QbReport** クラスの **applyTemplate**（**String fileName**）メソッドを使用してテンプレート名を指定することもできます。

以下の例は、アプレットまたはアプリケーションとして実行でき、**QbReport** オブジェクトにテンプレートを適用します。

```
Component doApplyingTemplate(Object parent) {

    //Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    String templateLocation = "..";

    //Apply the template
    QbReport report = new QbReport (parent, //container
                    QbReport.SUMMARY, //report type
                    data, //data
                    columnMapping, //column mapping
                    templateLocation); //template
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
    //Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

フルソースコード
エクスポートした結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

また、テンプレートから列マッピングを取得し、作成する **QbReport** オブジェクトに適用することもできます。これは、**ColInfo** []オブジェクトの代わりに "**null**"（二重引用符を使用しない）を渡すことによって行われます。

列マッピングに加えて、テンプレートからデータベース接続情報も取得されます（テンプレートでデータソースとしてデータベースが使用されていることが前提です）。これは、**DBInfo** または **SimpleQueryFileInfo** のいずれかのパラメータに"**null**"（二重引用符は不要）を渡すことによって行われます。

既定では、テンプレートを適用すると、テーブルデータセクションの数式は適用されません。テンプレートまたは.rpt/.xml ファイルから **QbReport** オブジェクトに式またはスクリプトを適用するには、次の方法を使用する必要があります。

```
QbReport.applyTemplate(String templateName, boolean applyFormula);
```

4.3.5.3 データソースの変更

Designer でレポートテンプレートを作成し、API を使用してそれらのテンプレートを開くことができます。作成された **QbReport** オブジェクトは、テンプレートと同じデータソースを使用し、データをフェッチしようとします。ただし、テンプレートの外観がすべて必要な場合でも、データソースが正しくない可能性があります。次のセクションでは、バックアップデータでテンプレートを開き、レポート全体を再作成せずにデータソースを切り替える方法を示します。

最良の結果を得るには、列数と各列のデータ型が、2 つのデータソース（Designer でテンプレートを作成するために使用されたものと新しいデータソース）の間で一致している必要があります。

データソースを切り替えた後、**QbReport** オブジェクトを強制的に新しいデータをフェッチする必要があります。これは、**QbReport** クラスの refresh メソッドを呼び出すことで実行できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.3.5.3.1 データベースからのデータ

データソースを、データベースを参照するように切り替えるのは簡単です。必要なのは、データベース接続情報と使用するクエリを提供し、それを QbReport オブジェクトに渡すだけです。DBInfo オブジェクトにデータベース接続情報（クエリと同様に）を提供することができます（DBInfo オブジェクトの作成の詳細については、データベースのデータを参照してください）。

以下の例は、アプレットまたはアプリケーションとして実行でき、**QbReport** オブジェクトのデータソースをデータベースに切り替えます：

```
Component doSwitchToDatabase(Object parent) {

    //Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    //Open the template with backup data
    QbReport report = new QbReport (parent, //container
                    "SwitchToDatabase.rpt", //template
                    false, false, false, true);

    //New database connection information
    DBInfo newDatabaseInfo = new DBInfo(.....);

    try {
        //Switch data source
        report.getInputData().setDatabaseInfo(newDatabaseInfo);

        //Refresh the report
        report.refresh();

    } catch (Exception ex)
    {
        ex.printStackTrace();

    }

    //Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

フルソースコード

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

エクスポートした結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

上記の方法では、現在のレポートレベルのデータソースのみが変更されます（メインレポートまたはサブレポートのいずれかになります）。

レポートテンプレートにサブレポート、ドリルダウンレポート、独立したグラフ（つまり、レポートデータをデータソースとして使用しないグラフ）があり、それらすべてのデータソースを変更したい場合は、**IInputData** インタフェースで **setAllDatabaseInfo** メソッドを使用します。この方法では、新しいクエリを指定することはできず、元のクエリが使用されることに注意してください。データベース接続情報のみが変更されます。

アプレットまたはアプリケーションとして実行できる次の例では、**setAllDatabaseInfo** メソッドを使用してデータソースを切り替えます。

```
Component doSwitchToDatabaseSetAll(Object parent) {

        //Do not use EspressManager
        QbReport.setEspressManagerUsed(false);

        //Open the template with backup data
        QbReport report = new QbReport(parent, //container
                "SwitchToDatabase.rpt", //template
                false, false, false, true);

        try {
            //Switch data source
            report.getInputData().setAllDatabaseInfo("jdbc:hsqldb:woodview",
                    "org.hsqldb.jdbcDriver", "sa", "");

            //Refresh report
            report.refresh();
        } catch (Exception ex)
        {
            ex.printStackTrace();

        }

        //Show report in Viewer
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            return (new Viewer().getComponent(report));
        }
```

フルソースコード
エクスポートした結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

パラメータ化されたクエリ

通常のクエリと同様に、データソースをパラメータ化されたクエリに切り替えることができます。 パラメータ化されたクエリでは、パラメータとデータベース接続情報およびクエリを指定する必要があります。

以下の例は、アプレットまたはアプリケーションとして実行でき、**QbReport** オブジェクトのデータソースをパラメータ化されたクエリに切り替えます。

```
Component doSwitchToDatabaseParam(Object parent) {

    //Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    //Open the template with backup data
    QbReport report = new QbReport (parent, //container
                    "SwitchToDatabaseParam.rpt", //template
                    false, false, false, true);

    //New database connection and parameter information
    SimpleQueryFileInfo newDatabaseInfo = new SimpleQueryFileInfo(.....);

    try {
        //Switch data source
        report.getInputData().setDatabaseInfo(newDatabaseInfo);

        //Refresh the report
        report.refresh();

    } catch (Exception ex)
    {
        ex.printStackTrace();
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
    }

    //Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

フルソースコード
エクスポートした結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

ここでも、通常のクエリと同様に、**setAllDatabaseInfo** メソッドを使用して新しいデータソースに切り替えることができます。このアプローチでは、データベース接続情報が変更されている間は、元のクエリおよびパラメータ情報が使用されます。データベース情報を切り替えた後、レポートをリフレッシュする前にパラメータ値を指定する必要があります。

アプレットまたはアプリケーションとして実行できる次の例では、**setAllDatabaseInfo** メソッドを使用してデータソースを切り替えます：

```
Component doSwitchToDatabaseParamSetAll(Object parent) {

    //Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    //Open the template with backup data
    QbReport report = new QbReport (parent, //container
                    "SwitchToDatabaseParam.rpt", //template
                    false, false, false, true);

    try {
        //Switch data source
        report.getInputData().setAllDatabaseInfo(...);

        //Pass in parameter value
        report.getAllParameters().get(0).setValue("TRD");

        //Refresh the report
        report.refresh();
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
    } catch (Exception ex)
    {
        ex.printStackTrace();

    }

    //Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

フルソースコード
エクスポートした結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

JNDI

また、データソースを JNDI データソースに変更することもできます。これは、**DBInfo** オブジェクト内の JNDI 接続情報を指定し、それを **QbReport** オブジェクトに渡すことによって行われます。

以下の例は、アプレットまたはアプリケーションとして実行でき、**QbReport** オブジェクトのデータソースを JNDI データベースに切り替えます：

```
Component doSwitchToDatabaseJNDI(Object parent) {

        //Do not use EspressManager
        QbReport.setEspressManagerUsed(false);

        //Data Source.  Replace comp with computer and env with environment.
        //The environment hashtable is empty for tomcat.
        //If other application server is used, need to set INITIAL_CONTEXT_FACTORY and PROVIDER_URL.
        DBInfo newDatabaseInfo = new DBInfo(
                        "java:comp/env/jdbc/Woodview",
                        "SELECT Categories.CategoryName, Products.ProductName, Products.UnitPrice, Products.StainPrice, Products.UnitsInStock FROM Products, Categories WHERE (Products.CategoryID = Categories.CategoryID)",
                        new Hashtable());

        //Open the template with backup data
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        QbReport report = new QbReport(parent, //container
                        "SwitchToDatabaseJNDI.rpt", //template
                        false, false, false, true);

        try {
                //Switch data source
                report.getInputData().setDatabaseInfo(newDatabaseInfo);

                //Refresh report
                report.refresh();
        } catch (Exception ex)
        {
                ex.printStackTrace();
        }

        //Show report in Viewer
        return (new Viewer().getComponent(report));
}
```

フルソースコード
エクスポートした結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。アプリケーションコードを実行するには、Woodview データベースを Tomcat 環境の JNDI データソースとして設定し、アプリケーションコードを接続情報に合わせて変更する必要があります。

4.3.5.3.2 データファイルからのデータ（TXT/DAT/XML）

テキストファイルが Quadbase のガイドラインに従っている限り、データソースをテキストファイルに切り替えることができます（詳細はデータファイルからのデータ（TXT/DAT/XML）を参照してください）。

以下の例は、アプレットまたはアプリケーションとして実行でき、QbReport オブジェクトのデータソースをテキストファイルに切り替えます：

```
Component doSwitchToDataFile(Object parent) {

        //Do not use EspressManager
        QbReport.setEspressManagerUsed(false);

        //Open the template with backup data
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        QbReport report = new QbReport(parent, //container
                        "SwitchToDataFile.rpt", //template
                        false, false, false, true);

        try {
                //Switch data source
                report.getInputData().setDataFile("sample.dat");

                //Refresh report
                report.refresh();
        } catch (Exception ex)
        {
                ex.printStackTrace();
        }

        //Show report in Viewer
        return (new Viewer().getComponent(report));
}
```

フルソースコード
エクスポートした結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

また、データをソートする（これによりパフォーマンスが向上する）か、テキストファイルのエンコーディングを指定することもできます。これは、**IInputData** で次のメソッドを使用して実行できます：

```
setDataFile(String dataFile, boolean sortedData, String encoding);
```

4.3.5.3.3 XML データソースからのデータ

データに付随する.dtd または.xml スキーマが存在する限り、データソースをカスタム XML データに切り替えることができます。XML データ情報が指定されています（詳細は XML データソースからのデータを参照してください）。その後、**QbReport** オブジェクトに渡されます。

以下の例は、アプレットまたはアプリケーションとして実行でき、**QbReport** オブジェクトのデータソースを XML データに切り替えます：

```
Component doSwitchToXMLData(Object parent) {

    //Do not use EspressManager
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
    QbReport.setEspressManagerUsed(false);

    //Open the template with backup data
    QbReport report = new QbReport (parent, //container
                        "SwitchToXMLData.rpt", //template
                        false, false, false, true);

    //XML data source information

    XMLFileQueryInfo newData = new XMLFileQueryInfo(...);

    try {
        //Switch data source
        report.getInputData().setXMLFileQueryInfo(newData);

        //Refresh the report
        report.refresh();

    } catch (Exception ex)
    {
        ex.printStackTrace();

    }

    //Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

フルソースコード
エクスポートした結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

IInputData で次のメソッドを使用して、データを並べ替えるかどうかを指定することもできます（これによりパフォーマンスが向上します）：

```
setXMLFileQueryInfo(XMLFileQueryInfo xmlInfo, boolean sortedData);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.3.5.3.4 カスタム実装からのデータ

通常のデータソースに加えて、独自のカスタムデータを渡すこともできます。カスタムデータは、**IDataSource** または **IParameterizedDataSource** インタフェースを使用して **QbReport** オブジェクトに渡されます。詳細については、カスタム実装で渡されるデータを参照してください。

以下の例は、アプレットまたはアプリケーションとして実行でき、データソースをカスタム実装に切り替えます。

```
Component doSwitchToCustomData(Object parent) {

	//Do not use EspressManager
	QbReport.setEspressManagerUsed(false);

	//Open the template with backup data
	QbReport report = new QbReport(parent, //container
			"SwitchToCustomData.rpt", //template
			false, false, false, true);

	try {
		//Switch data source
		report.getInputData().setClassFile("Furniture_Report");

		//Refresh report
		report.refresh();
	} catch (Exception ex)
	{
		ex.printStackTrace();
	}

	//Show report in Viewer
	return (new Viewer().getComponent(report));
}
```

フルソースコード
エクスポートした結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

また、**IInputData** で次のメソッドを使用してデータを並べ替える（これによりパフォーマンスが向上する）か、パラメータダイアログを表示するかを指定することもできます：

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
setClassFile(String classname, boolean sortedData, boolean showPromptDialog);
```

パラメータ化されたカスタム実装

パラメータ値を必要とするカスタム実装を新しいデータソースとして使用することができます。このシナリオでは、カスタム実装では **IParameterizedDataSource** インタフェースを使用する必要があります。 詳細については、カスタム実装で渡されるデータを参照してください。データソース情報を切り替えた後、レポートをリフレッシュする前にパラメータ値を指定する必要があります。

以下の例は、アプレットまたはアプリケーションとして実行でき、データソースをパラメータ化されたカスタム実装に切り替えます：

```
Component doSwitchToCustomDataParam(Object parent) {

	//Do not use EspressManager
	QbReport.setEspressManagerUsed(false);

	//Open the template with backup data
	QbReport report = new QbReport(parent, //container
			"SwitchToCustomDataParam.rpt", //template
			false, false, false, true);

	try {
		//Switch data source
		report.getInputData().setClassFile("ProductParamInfo", false, false);

		report.getAllParameters().get(0).setValue(new Integer(25));

		//Refresh report
		report.refresh();
	} catch (Exception ex)
	{
		ex.printStackTrace();
	}

	//Show report in Viewer
	return (new Viewer().getComponent(report));
}
```

フルソースコード
エクスポートした結果

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

4.3.5.3.5 メモリ内の配列に渡されたデータ

配列を使用してデータを渡すこともできます。配列データは通常メモリに格納され、**QbReport** オブジェクトに渡されます（詳細についてはメモリ内の配列に渡されたデータを参照してください）。

アプレットまたはアプリケーションとして実行できる次の例は、データソースをメモリ内の配列に切り替えます：

```
Component doSwitchToArrayData(Object parent) {

    //Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    //Open the template with backup data
    QbReport report = new QbReport (parent, //container
                    "SwitchToArrayData.rpt", //template
                    false, false, false, true);

    //Create array data
    DbData newData = new DbData(...);

    try {
        //Switch data source
        report.getInputData().setData(newData);

        //Refresh the report
        report.refresh();

    } catch (Exception ex)
    {
        ex.printStackTrace();

    }

    //Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フルソースコード
エクスポートした結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

IInputData で次のメソッドを使用して、データがソートされ、パフォーマンスが向上するかどうかを指定することもできます。

```
setData(IResultSet rs, boolean sortedData);
```

4.3.5.3.6 DrillDownReportServlet を使用したドリルダウン

ドリルダウンレポートのデータソースを変更する場合は、**IInputData** で **setAllDatabaseInfo** メソッドを使用します。ただし、レポートをエクスポートし、**DrillDownReportServlet** を使用している場合（詳細は DrillDownReportServlet を参照）、セッションへのハンドルを取得し、**QbReport** の **setDrillDownDatabaseInfo** メソッドを使用して、新しいデータソースが何であるかを知ることができます。

アプレットまたはアプリケーションとして実行できる次の例は、ドリルダウンレポートのデータソースを切り替えます：

```
public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
        IOException
{
        HttpSession session = req.getSession(true);

        //Set the "content type" header of the response
        res.setContentType("text/html");

        //Get the response's OutputStream to return content to the client.
        OutputStream toClient = res.getOutputStream();

        try
        {
                //Do not use EspressManager
                QbReport.setEspressManagerUsed(false);

                //Open report with backup data (data source will be switched later)
                QbReport report = new QbReport(null, "SwitchDrillDownServlet.pak",
                                false, false, false, true);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            //New database connection information
            String newDatabaseURL = "jdbc:hsqldb:woodview";
            String newDatabaseDriver = "org.hsqldb.jdbcDriver";
            String newDatabaseUID = "sa";
            String newDatabasePassword = "";

            //Switch data source
            report.getInputData().setAllDatabaseInfo(newDatabaseURL, newDatabaseDriver,
                        newDatabaseUID, newDatabasePassword);

            //Put new data source information in session for DrillDownReportServlet
            report.setDrillDownDatabaseInfo(session, newDatabaseURL, newDatabaseDriver,
                        newDatabaseUID, newDatabasePassword);

            report.setDynamicExport(true, "localhost", 8080);

            ByteArrayOutputStream tempStream = new ByteArrayOutputStream();

            report.refresh();

            //Export the report to DHTML
            report.export(QbReport.DHTML, tempStream);

            tempStream.writeTo(toClient);
        } catch (Exception e) {
            e.printStackTrace();
        }

        //Flush the outputStream
        toClient.flush();

        //Close the writer; the response is done.
        toClient.close();
}
```

フルソースコード
エクスポートした結果のルート
エクスポートした結果のドリルダウンレベル

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

能な完全なアプリケーションコードが含まれています。コードを正常に実行するには、テンプレートの場所を変更してサーブレットを配備する必要があります。

QbReport で次のメソッドを使用して、データベースの接続オブジェクトを指定することもできます。

```
setDrillDownConnection(Object session, Connection conn);
```

4.3.5.4 列マッピングの変更

レポートのデータソースを変更できるように、API を使用して列マッピングを変更することもできます。ただし、レポートにデータに依存する式やスクリプトが含まれている可能性があるため、これはお勧めしません。列の数および列のデータが元のマッピングと一致しない場合、特定の数式およびスクリプトが機能しないことがあります。このセクションでは、マッピングを切り替える方法を示していますが、このようなシナリオでは新しい **QbReport** オブジェクトを作成して（レポートの作成を参照)、テンプレートを適用することをお勧めします。

以下の例は、アプレットまたはアプリケーションとして実行でき、レポートの列マッピングを変更します：

```
Component doModifyColumnMapping(Object parent) {

    //Do not use EspressManager
    QbReport.setEspressManagerUsed(false);

    //Open the template with backup data
    QbReport report = new QbReport (parent, //container
                    "ModifyColumnMapping.rpt", //template
                    false, false, false, true);

    //Create new column mapping
    ColInfo[] newColumnMapping = new ColInfo[..];

    try {
        //Switch mapping
        report.getInputData().setMapping(newColumnMapping);

        //Refresh the report
        report.refreshWithOriginalData();

        //Reapply template
        report.applyTemplate("ModifyColumnMapping.rpt");
```

Copyright(C) 2018 Climb.Inc

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
    } catch (Exception ex)
    {
        ex.printStackTrace();

    }

    //Show report in Viewer
    return (new Viewer().getComponent(report));

}
```

フルソースコード
エクスポートした結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

4.3.5.5 レポートコンポーネント

EspressReport レポートはさまざまなセグメントで構成されています。各セグメントは、他のセグメントとは独立して設定および変更できます。以下は、レポートのさまざまな部分です。

4.3.5.5.1 ReportElement

ReportElement オブジェクトは、Report API でオブジェクトのコアを形成します。ReportElement オブジェクトは、個々の要素の外観だけでなく内容を操作する方法を提供します。他のすべてのレポートオブジェクト（ReportCell、ReportSection、ReportColumn など）は、ReportElement クラスを拡張します。レポートのすべての部分は、ReportElement オブジェクトまたはその拡張部分で構成されます。各 ReportElement オブジェクト内で、フォント、色、データ形式などのプロパティを操作できます。ReportElement オブジェクトをコピーすることも、1 つの ReportElement オブジェクトのテンプレートを別の ReportElement オブジェクトに適用することもできます。テンプレートが適用されると、オブジェクトのデータは変更されません。ReportElement オブジェクトの外観のみが変更されます。

それを変更するときは、ReportElement のハンドルを取得するのではなく、サブクラスのメソッド（ReportCell、ReportSection、ReportColumn など）を使用します。

4.3.5.5.2 ReportCell

ReportCell オブジェクトは、ラベル、式、テキスト、チャート、または画像をレポートの異なるセクション（つまり、レポートの ReportSection）に挿入するために使用されます。ReportCell オブジェクトを使用して、レポートのさまざまな部分のセルの書式設定を支援することもできます。

数値インデックス（セクション内のセルの位置）、ID（文字列）、またはカスタム ID（セルに 1 つがあることが前提）を使用して、セルへのハンドルを取得できます。**ReportSection** で以下のメソッドを使用

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

して、目的のセルを取得できます：

```
getData(int i);
getData(String ID);
```

ReportSection で次のメソッドを使用して、特定のセクション内のすべてのレポートセルを取得することもできます：

```
getData();
```

4.3.5.5.3 ReportSection

レポートは多くのセクションに分かれています。以下は、レポートのさまざまなセクションを示しています：

Page Header :-

このセクションはページのヘッダーとして機能します。これは、レポートの永久保存の最上部に表示されます。

Table Header :-

このセクションは、レポートの詳細またはデータセクションのヘッダーとして機能します。 デフォルトでは、レポートには 1 回だけ表示されます。

Group Header:-

このセクションは、グループ化されたデータ（マスタ&詳細レポート）を含むレポート、または改行が挿入されたデータ（要約レポート）で表示されます。レポート内の各グループの先頭に繰り返し表示されます。

Table Data:-

これは、ほとんどのデータを含むレポートのメインセクションです。レポート用に選択されたデータ列はこのセクションに配置され、列の各エントリごとに繰り返されます。

Group Footer:-

このセクションは、グループ化されたデータ（マスタ&詳細レポート）を含むレポート、または行区切りが挿入されたデータ（要約レポート）で表示されます。レポート内の各グループの最後に繰り返し表示されます。

Table Footer:-

このセクションは、レポートの詳細またはデータセクションのフッターとして機能します。デフォルトでは、レポートに 1 回だけ表示されます。

Page Footer:-

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

このセクションは、ページのフッターとして機能します。レポートの各ページの下部に表示されます。

Report Footer:-

これはレポートの最後の要約またはフッターセクションです。レポートの最後に一度だけ表示されます。

QbReport の次のメソッドを使用して、Page Footer、Page Header、Report Footer、Report Header、および Report Table セクションへのハンドルを取得できます：

```
getPageFooter();
getPageHeader();
getReportFooter();
getReportHeader();
getTable();
```

Report Table の次のメソッドを使用して、Group Footer、Group Header、Table Header、および Table Footer セクションのハンドルを取得できます。

```
getRowBreakFooter(int breakLevel);
getRowBreakHeader(int breakLevel);
getFooter();
getHeader();
```

4.3.5.5.4 ReportColumn

ReportColumn オブジェクトは、選択した列に表示される ReportCell オブジェクトの配列です。ReportColumn オブジェクトを使用すると、一度に 1 つのセルの書式設定を適用する代わりに、列全体を通して書式設定を 1 回行うことができます。

ReportTable で次のメソッドを使用して、ReportColumn へのハンドルを取得できます。

```
getColumn(int index);
```

4.3.5.5.5 ReportTable

これがレポートの主要部分です。これは、レポート用に選択されたデータ列を含むセクションです。このセクションには、グループフッターとさまざまなサマリーに使用されるデータが含まれています。

QbReport で次のメソッドを使用して、**ReportTable** セクションのハンドルを取得できます：

```
getTable();
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.3.5.5.6 ReportImage

ReportImage オブジェクトは、レポートにイメージを追加するために使用されます。EspressReport でサポートされている形式は、GIF、JPEG、および PNG です。**ReportImage** オブジェクトは、イメージを含むセルと考えることができます。画像がはっきりと見え、切り捨てられないように、セルの寸法を定義するように注意しなければなりません。

QbReport で次のメソッドを使用すると、すべての **ReportImage** オブジェクトのハンドルを取得できます。

```
getReportImages();
```

4.3.5.5.7 ReportChartObject

ReportChartObjects は、TPL または CHT（Quadbase 独自の形式）のグラフをレポートに追加するために使用されます。**ReportChartObject** での追加は、**ReportImage** オブジェクトでの追加とは異なります。ここでは、TPL またはテンプレートファイルの場所が示されています。その後、EspressReport はテンプレートを取得し、関連するセクションのデータを使用してチャートを作成します。したがって、同じテンプレートを使用すると、レポートのさまざまな場所に異なるチャートを持つことができます。これらのチャートは、データが異なっていても同じ外観を共有します。

チャートテンプレートファイルは、まず Report Designer で作成されます。したがって、チャートのマッピングは、レポート内のデータのマッピングに基づいています。API を使用して生成されるレポートと同様のデータマッピング（つまり、同じ種類のデータとデータ型）を含む同じタイプのレポートからチャートテンプレートを使用することをおすすめします。

ReportChartObjects を使用してスタンドアロンチャート（つまり、レポートからのデータではないチャート）を追加することもできます。スタンドアロンチャートは API から作成され、チャートのデータソースはレポートとは独立しています。

ここでも、**ReportChartObject** オブジェクトはチャートを含むセルと見なすことができます。チャートがはっきり見えるように、また切り捨てられないように、セルの寸法を定義するために同様の注意を払う必要があります。

QbReport で次のメソッドを使用すると、すべての **ReportChart** オブジェクトのハンドルを取得できます。

```
getReportChartObjects();
```

4.3.5.5.8 ChartObject

ChartObjects は、API を通じて完全に作成されたチャートをレポートに追加する前に作成された中間オブジェクトです。チャートはレポートからデータを取得し、チャートのマッピングはレポートの列に基づいています。これらのチャートは、スタンドアロンチャートとは異なります。これらのチャートの

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

データはレポートそのものであり、スタンドアロンチャートのデータはレポートから独立しているためです。2D/3D、チャートタイプ、マッピングなどの他のチャートプロパティも指定され、**ChartObject** が作成されます。

ChartObject は **ReportChartObject** に追加されます。この **ReportChartObject** がレポートに追加されます。

ChartObject の使用方法の詳細についてはチャートガイドを参照してください。

4.3.5.5.9 ReportDocument

ReportDocument は、レポートに **clob** を追加するときに **clob**（大文字オブジェクト）を表すために使用されます。**clob** の長さに応じて、**ReportDocument** に文字列またはファイルに格納され、ファイル名は **ReportDocument** に格納されます。

ここでも、**ReportDocument** オブジェクトは **clob** を含むセルと見なすことができます。同様の注意は、コンテンツがはっきりと見え、切り捨てられないように、セルの寸法を定義するために行われるべきである。

ReportDocument オブジェクトは API を使用して変更されないため、すべての変更はテンプレートで行うことをお勧めします。

4.3.5.5.10 ReportRTFObject

ReportRTFObject は、RTF ファイルのコンテンツをレポートに追加するために使用されます。ファイルコンテンツは **ReportRTFObject** にストリームされ、このオブジェクトが後でレポートに追加されます。

ここでも、**ReportRTFObject** オブジェクトは、RTF コンテンツを含むセルと見なすことができます。同様の注意は、コンテンツがはっきりと見え、切り捨てられないように、セルの寸法を定義するために行われるべきである。

ReportRTFObject オブジェクトは、API を使用して変更されないため、すべての変更をテンプレートで行うことをお勧めします。

4.3.5.5.11 SubReportObject

SubReportObject は、サブレポートをレポートに追加するために使用されます。 サブレポートのコンテンツは、ファイルから作成することも、API から完全に作成することもできます。

ここでも、**SubReportObject** オブジェクトはサブレポートを含むセルと見なすことができます。同様の注意は、コンテンツがはっきりと見え、切り捨てられないように、セルの寸法を定義するために行われるべきである。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

QbReport で次のメソッドを使用して、すべての **SubReportObject** オブジェクトのハンドルを取得できます。

```
getSubReports();
```

SubReportObject 内の次のメソッドを使用することで、**SubReportObject** からサブレポートのハンドルを取得できます。

```
get SubReport(IReport qbReport);
```

取得した **SubReport** オブジェクトを **QbReport** にキャストできます。

4.3.5.5.12 ReportGrid

ReportGrid オブジェクトは、テーブルの異なるセクション内にグリッドを描画するために使用されます。グリッドには、各行と各列（該当する場合は、セクション内の各セル）が含まれます。現在、使用できる線のスタイルはダッシュ、二重、実線のスタイルです。

ReportGrid オブジェクトは API を使用して変更せず、すべての変更をテンプレートで行うことをお勧めします。

4.3.5.5.13 ReportLine

ReportLine オブジェクトは、テーブルの異なるセクションに線を描画するために使用されます。レポートの任意の部分に行を挿入することができます。**ReportGrid** オブジェクトの場合と同様に、**ReportLine** オブジェクトには 3 つのスタイル（ダッシュ、二重線、実線）があります。

ReportLine オブジェクトは API を使用して変更せず、すべての変更をテンプレートで行うことをお勧めします。

4.3.5.6 レポート属性の変更

EspressReport には何百ものプロパティがあり、レポートのさまざまな要素を細かく制御できます。開発者は、実行時に動的にレポートの外観をカスタマイズできます。使い易さを容易にするために、ほとんどのプロパティはグループに分類され、インタフェースの形式で公開されています。アプリケーションは、最初に **getXXX** メソッドを使用してグループインタフェースへのハンドルを取得し、そのインタフェース上のメソッドを呼び出すことによって直接レポートのプロパティを操作します。ほとんどのインタフェースは、**quadbase.reportdesigner.util** パッケージに含まれています。

4.3.5.6.1 新しいセルを追加する

EspressReport では、新しいセルを作成して、レポート内のどこに配置するかを指定できます。

レポートにセルを追加するには、**ReportCell** オブジェクトを作成する必要があります（ただし、**ReportCell** オブジェクトである必要はありません。追加する情報の種類に応じて、**ReportImage** オブ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ジェクト、**ReportChartObject**、**ReportLine** オブジェクト、または **ReportGrid** オブジェクトを指定することもできます)。**ReportCell** プロパティを作成および指定したら、レポートの任意の部分に追加できます。

次の例では、セルがページフッターとして追加されます。

```
ReportCell cell = new ReportCell("Inventory Report");
cell.setWidth(7);
cell.setAlign(IAlignConstants.ALIGN_RIGHT);
report.getPageFooter().addData(cell);
```

ReportCell で次のメソッドを使用して、新しく作成した **ReportCell** オブジェクトに別のセルの外観を適用することもできます。

```
applyTemplate(ReportCell templateCell, boolean applyScript);
```

4.3.5.6.2 画像/チャートの追加

画像とチャートの両方をレポートの実質的にどの部分にも含めることができます。GIF、JPEG、PNG、TPL（Quadbase 独自のフォーマット）画像が EspressReport でサポートされています。チャート/画像をレポートに追加すると、一般的にプレゼンテーションがより良くなり、より多くの情報が得られ、レポートを読みやすく理解しやすくなります。

イメージを追加するには、**ReportImage** オブジェクトを定義する必要があります。**ReportImage** オブジェクト内では、イメージの URL を **http:**//または **file:**//として渡す必要があります。DHTML レポートがイメージを一貫してピックアップするため、レポートを DHTML 形式にエクスポートする場合は、**http:**//の URL を使用することをお勧めします。また、**ReportImage** オブジェクトの画像の種類、寸法、位置合わせを指定してから、レポートに必要な場所に追加することもできます。

次の例では、gif タイプの **logo3** という画像がページヘッダーに追加されています。

```
ReportImage reportImage = new ReportImage();
try {

    reportImage.setImageType(IExportConstants.GIF);
    java.net.URL imageLocation =
            new java.net.URL ("http://someMachineName/Gifs/logo3.gif");
    reportImage.setImageURL(imageLocation);
    reportImage.setWidth(7);
    reportImage.setHeight(1);
    reportImage.setAlign(IAlignConstants.ALIGN_LEFT);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
} catch (Exception ex)  {

      ex.printStackTrace();

}

report.getPageHeader().addData(reportImage);
```

また、コードで指定されたフル URL（または Report Designer）ではなく、使用する相対リンクを指定することもできます。これは、**ReportImage** の次のメソッドで相対パスを渡すことで実行できます。

```
public void setImagePath(String path);
```

たとえば、上記の方法で**../../logo3.gif** を渡すと、絶対 URL ではなく**<img src>**タグに相対リンクがある DHTML エクスポートが発生します。

チャートを追加するには、レポートへのチャートの追加を参照してください。

4.3.5.6.3 ハイパーリンクの追加

また、レポート内の 1 つまたは複数のセルにハイパーリンクを追加し、異なるセルに異なるハイパーリンクを設定することもできます。リンクとターゲットのヒントをセルに設定すると、レポートをエクスポートするときにそのリンクが保持されます。RPT ファイルへのリンクは、レポートに挿入することもできます。ただし、そのようなファイルへのリンクはアプレット環境外では機能しません。DHTML 形式の RPT ファイルを表示するには、別々の DHTML ファイルを作成し、それらのファイルにリンクすることをお勧めします。

次の例では、ページへのリンクを含むセルが作成され、ページヘッダーに追加されます。

```
ReportCell cell = new ReportCell("ABC Incorporated);
cell.setLink(new String("http://www.quadbase.com"));
cell.setHint(new String("Click here to go to Quadbase's Homepage"));
cell.setTarget(new String("ABC Home Page"));
report.getPageHeader().addData(cell);
```

4.3.5.6.4 グリッド線と線の追加

API を使用して **ReportGrid** と **ReportLine** をレポートに追加できますが、Designer を使用して、すべてのグリッドとラインをレポートに挿入することをお勧めします。グリッド線をレポートに追加するには、**ReportGrid** オブジェクトを定義する必要があります。ReportGrid オブジェクト内では、グリッドに囲まれるセクションを指定する前に、色、線種、太さ、幅などのさまざまなプロパティを定義できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次の例では、レポートテーブルの周囲に青色のグリッド線が追加されています。

```
ReportGrid reportGrid = new ReportGrid();
reportGrid.setBorderColor(Color.blue);
reportGrid.setGridStyle(ReportGrid.DOUBLE);
reportGrid.setBorder(1);
reportGrid.setWidth(7);
report.getTable().addImage(reportGrid);
```

レポートに行を追加するには、**ReportLine** オブジェクトを定義する必要があります。ここでは、追加するセクションを指定する前に、色、線種、太さ、幅などのさまざまなプロパティを定義することもできます。

次の例では、ページヘッダーに行が追加されます。

```
ReportLine reportLine = new ReportLine(false);
reportLine.setBorderColor(Color.red);
reportLine.setLineStyle(ReportLine.DOUBLE);
reportLine.setBorder(1);
reportLine.setWidth(7);
report.getPageHeader().addData(reportLine);
```

4.3.5.6.5 リッチテキストフィールドオブジェクトの追加

リッチテキストフィールドをレポートに追加するには、**ReportRTFObject** オブジェクトを定義する必要があります。**ReportRTFObject** オブジェクト内では**.rtf** ファイルの場所を指定してから、オブジェクトを追加するセクションを指定する前にセルのその他のプロパティを指定できます。**ReportRTFObject** をレポートに追加するには**.rtf** ファイルが存在する必要があります。

ただし、この方法を使用して RTF コンテンツを含めるのではなく、Designer のテンプレートに追加することをお勧めします。

次の例では、**RText1.rtf** ファイルの内容がテーブルデータセクションに追加されます。

```
ByteArrayOutputStream fileOne = new ByteArrayOutputStream();
FileInputStream fileIn = new FileInputStream("RText1.rtf");

int b = fileIn.read();
while (b != -1) {

    fileOne.write(b);
    b = fileIn.read();
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
}

ReportRTFObject rtfObjectOne = new ReportRTFObject(fileOne.toByteArray());
rtfObjectOne.setWidth(1);
rtfObjectOne.setHeight(.3);
rtfObjectOne.setResizeToFitContent(true);
rtfObjectOne.setX(report.getTable().getColumn(0).getWidth() +
report.getTable().getColumn(1).getWidth() +
        report.getTable().getColumn(2).getWidth());      //Set Object at the end of table data

report.getTable().addRTFObject(rtfObjectOne);
//report is an object of type QbReport
```

RTF ファイルの内容は複数の列形式で表示することもできます。したがって、デフォルトの列数 1 の代わりに、2 つ以上の列に内容を表示することができます。列数と列間の間隔は、次の方法を使用して設定できます。

```
ReportRTFObject.setColumnCount(int columnC);
ReportRTFObject.setColumnSpacing(double space);
```

たとえば、上記の例では、次のコード行が追加されたとします。

```
rtfObjectOne.setColumnCount(3);
rtfObjectOne.setColumnSpacing(.3);
```

レポートに **ReportRTFObject** を追加する前に、レポートには、隣接する 2 つの列を区切る 0.3 インチの 3 列形式の RTF 部分が表示されます。

リッチテキストフィールドコンテンツの詳細については、リッチテキストフィールドを参照してください。

4.3.5.6.6 ネストされたセクションの追加

レポートのネストされたセクションを追加するには、親セクションのハンドルを取得し、子セクションを作成する必要があります。ネストされたセクションは、以下のオブジェクトが重複することなくサイズ変更が必要なセルやオブジェクトと組み合わせて使用する場合に非常に便利です。ネストされたセクションは、ページ区切りオプション以外の親セクションからセクションオプションのほとんどを継承することに注意してください。

次の例では、2 つのネストされたセクションがテーブルヘッダーセクション内に作成され、**ReportCell** オブジェクト（**reportCell**）がセクションの 1 つに追加されています。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
ReportSection tableHeader = report.getTable().getHeader();
tableHeader.addSection();
tableHeader.insertSection(0);
tableHeader.getSection(0).setHeight(.5);
tableHeader.getSection(1).setHeight(.5);
tableHeader.getSection(0).addData(new ReportCell("Test Cell"));
```

ネストされたセクションに使用されるインデックスは、ベクトルインデックスに従います。セクション内にセルがない場合、高さが設定されていてもセクションは表示されません。

4.3.5.6.7 背景色、フォントの変更.....

レポートの任意のセクションのプロパティは、適切なハンドルを取得してメソッドを呼び出すことによって変更できます。レポートのすべての部分が **ReportElement** を継承しているため、ほとんどすべてのプロパティ（単一の ReportElement オブジェクトに適用可能）がここにあり、一度に 1 つ、またはグループに適用できます。

たとえば、次のコードは、**ReportCell** オブジェクトの背景色を赤に設定します。

```
ReportCell cell;
cell.setBgColor(Color.red);
```

以下のコードは、レポートの最初の列の背景色を青色に設定します。

```
ReportTable table = report.getTable();
table.getColumn(0).setBgColor(Color.blue);
```

テーブルの全ての列の背景色を青に設定したい場合は、for ループ文を用いて設定します。

```
for(int i = 0; i < table.getColumnCount(); i++)
    table.getColumn(i).setBgColor(Color.blue);
```

次のコードを書くことに注意してください。

```
table.setBgColor(Color.blue);
```

テーブルの列を青色に設定しないでください。テーブルセクションの色（すなわち、列のセルの後ろの領域）を青色に設定するだけで何も変化がありません。

以下のコードも同様です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
ReportCell cell;
cell.setFont(new Font("Arial", Font.BOLD, 14);
table.getColumn(0).setFont(new Font(table.getFont().getName(), Font.BOLD, 16));
```

セルのフォントとレポートの最初の列のフォントを設定します。**ReportCell** または **ReportColumn** オブジェクト（**ReportTable** または **ReportSection** など）以外のレポートの任意の部分のフォントを設定すると、セクション内の個々のセルのフォントを設定することに注意してください。独自のメソッドを作成して、レポートの任意のセクションのフォントを設定することができます。例えば、以下のコード：

```
void setFont(Font font, ReportElement elts[]) {

    if (elts == null) return;

    for (int i = 0; i < elts.length; i++) {
        elts[i].setFont(font);

}

setFont(new Font(table.getFont().getName(), Font.BOLD, 16), rowBreakHeaderZero.getData());
setFont(new Font(table.getFont().getName(), Font.BOLD, 18), tableHeader.getData());
```

独自のメソッドを作成し、それを使用して行ブレークゼロヘッダーとテーブルヘッダーのフォントを設定します。

レポートを PDF 形式でエクスポートする予定の場合は、次の方法で True Type フォントをマップして使用することもできます。

```
report.setFontMapping(String fontName, int style, String ttf);
```

ここで、**fontName** はフォント名、**style** は **QbReport FONT** 定数、**ttf** はインストールされた**.ttf** フォントのパスとファイル名です。例えば：

```
report.setFontMapping("Dialog", QbReport.BOLDITALIC,
"C:/EspressReport/help/examples/fonts/bookosbi.ttf");
```

PDF フォントマッピングの詳細については PDF フォントマッピングを参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.3.5.6.8 フォーマットの変更

必要に応じて任意のデータの形式（数値、ブール値、または文字列）を変更することができます。書式は、個々の **ReportCell** オブジェクトまたは **ReportColumn** オブジェクトに対して設定できます。

たとえば、次のコードです。

```
NumericFormat contentFormat = new NumericFormat();
contentFormat.decimal = 2;
contentFormat.currencySymbol = '$';

for (int i = table.getStartOfColumnBreakColumn(); i < table.getColumnCount(); i++ )
    table.getColumn(i).setDataFormat(contentFormat);
```

クロス集計レポートの列ブレーク列の数値データの書式を設定します。"**$**"記号を追加し、小数点以下 2 桁を表示するようにセルを設定します。

同様に、**string** または **boolean** データのプロパティも、要件に合わせて変更できます。

4.3.5.6.9 バーコードを表示するための列の変更

列内のデータ（文字列と数値）をバーコードとして表すこともできます。これは、ほとんどのデータソースにバーコードを格納する容量がないため便利です。バーコードに刻印された情報のみが文字列または数値データとして保存されます。このデータをレポートに列として追加し、データ形式を変更してバーコードを表示することができます。書式は、個々の **ReportCell** オブジェクトまたは **ReportColumn** オブジェクトに対して設定できます。サポートされているバーコード記号は、**Code 39、UPC A、EAN 13、Interleaved 2 of 5、Codabar** です。

次のコードは一例です。

```
BarcodeFormat barCode = new BarcodeFormat(BarcodeFormat.UPCA);
table.getColumn(0).setDataFormat(barCode);
```

列 0 のデータを UPC A シンボルを使用してバーコードとしてエンコードします。

4.3.5.6.10 レポートを上位 N レポートに変更する

グループ内の最高値または最低値のセットを表示するようにレポートを変更できます。これは、最も収益の高い、またはエラーの発生率が最も低いレポートを表示する場合に役立ちます。

レコードの数と順序は、次の方法で指定します。

```
QbReport.createTopNReport(int colIndex, int topN, boolean ascending);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

colIndex 引数はレポートの列を参照し、**topN** 引数はレコード数を指定し、昇順引数は最高レコードまたは最低レコードのどちらを表示するかを指定します。

次のコードは一例です。

```
report.createTopNReport(3, 20, true);
```

（レポートのグループ化に応じて）列 3 内の最高 20 レコードを取得し、順番に表示します。

シンプルカラムナ、サマリブレーク、マスタ&詳細レポートのみを上位 N レポートに変更できます。

4.3.5.6.11 セルスクリプト

カスタムセルスクリプトを開発することもできます。これらのスクリプトは、セルまたは列に割り当てることができ、特定の条件または要件が満たされたときに実行できます。これらのスクリプトは、周囲のセルと異なって見えるようにセルの書式を変更することができます。API を使用したセルスクリプトと Report Designer のセルスクリプトは異なる点に注意してください。

次の例は、アプリケーションまたはアプレットとして実行することができ、0 より小さいすべてのデータのフォントを変更するスクリプトを設定します。

```
public class myScript implements ICellScript  {

    //Format the cells according to the specified parameters
    public ReportCell formatCell(int rowIndex, ReportCell cell, Object originalData, IFormat dataFormat) throws Exception {
        if (originalData instanceof Double)
            if (((Double)originalData).intValue() < 0)
                cell.setFontColor(java.awt.Color.red);

        return cell;

    }

}
```

ReportColumn で次のメソッドを使用して、特定の列にスクリプトを適用できます。

```
setCellScript(ICellScript script);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フルソースコード
エクスポートされた結果

別の列に異なるスクリプトを作成することはできますが、列にはスクリプトが 1 つしかないことに注意してください。

4.3.5.7 レポートのエクスポート

EspressReport API には、さまざまな形式のレポートをエクスポートする機能があります。これらには、PDF、DHTML、TXT、CSV 形式などがあります。また、レポートに含まれるチャートは、JPEG、GIF、および PNG 形式にエクスポートすることもできます。既定では、レポートのグラフは JPEG としてエクスポートされます。チャートのフォーマットは、**ReportChartObject** オブジェクトの作成時に、イメージタイプを指定することで変更できます（**setImageType**（**int**）メソッドを使用）。

レポートは独自の PAK 形式にエクスポートすることもできます。PAK ファイルには、実際のデータを除くすべての情報が格納されます。PAK ファイルを使用して、レポートオブジェクトを構築することができます。PAK ファイルの場合、レポートオブジェクトの作成時に元のデータソースからデータが自動的にロードされます。（PAK ファイルは、レポートビューアアプレットを使用して直接表示できます）。

レポートをエクスポートするには、次の方法を使用します：

```
public export(int format, String filename)
```

上記の方法では、format は下記のフォーマット定数の 1 つであり、filename は出力ファイル名（拡張子の有無にかかわらず）です。

次のリストは、レポートコンポーネントのエクスポートに使用できる書式定数を示しています：

QbReport.CSV
　　カンマ区切りテキストファイル

QbReport.TXT
　　区切られたテキストファイル

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

QbReport.DHTML

ダイナミックハイパーテキストマークアップ言語（DHTML）

QbReport.HTML

ハイパーテキストマークアップ言語（HTML）

QbReport.PDF

パスワード保護オプション付きポータブルドキュメントフォーマット（PDF）

QbReport.RPT

レポートファイルフォーマット（RPT）

QbReport.RPT_DATA

データによるレポートファイルフォーマット（RPT）

QbReport.PAK

パックフォーマット（PAK）

QbReport.XML_DATA_AND_FORMAT

拡張マークアップ言語、データおよびレポートテンプレート（XML）

QbReport.XML_PURE_DATA

拡張マークアップ言語、データのみ（XML）

QbReport.XML_TEMPLATE

拡張マークアップ言語、レポートテンプレートのみ（XML）

QbReport.EXCEL

Excel フォーマット（XLS）

QbReport.EXCEL_OOXML

xcel 2007 フォーマット（XLSX）

QbReport.RTF

リッチテキストフォーマット（RTF）

QbReport.VIEW

ファイルを表示（VIEW）

さらに、レポートに含まれるチャートは、次のいずれかのイメージフォーマット（対応する定数付きの形式）にエクスポートできます：

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

QbReport.GIF

GIF

QbReport.JPEG

JPEG

QbReport.PNG

PNG

アプレットまたはアプリケーションとして実行できる次のコードは、レポートを作成およびエクスポートする方法を示しています：

```
Component doExportReport(Object parent) {
	QbReport.setEspressManagerUsed(false);
	ColInfo colInfo[] = new ColInfo[4];
	for (int i = 0; i < colInfo.length; i++) {
		//                          Map data column to the Report column
		colInfo[i] = new ColInfo(i);
	}

	QbReport report = null;
	try {
		report = new QbReport
				(parent, //applet
                                        "ExportReport.rpt"); //template
                ReportChartObject chartObject = new ReportChartObject();
		String chartLocation0 = new String("ExportReport0.tpl");
		chartObject.setText(chartLocation0);
		chartObject.setWidth(7);
		chartObject.setHeight(3);
		chartObject.setImageType(QbReport.JPEG);

		report.getReportFooter().addData(chartObject);

		report.export(QbReport.DHTML, "ExportReport");

	} catch (Exception ex) {
		System.out.println("Cannot create the report");
		ex.printStackTrace();
	}
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        return (new Viewer().getComponent(report));
}
```

フルソースコード
エクスポートされた結果

レポートをテキストファイルにエクスポートすると、デフォルトの区切り文字はタブスペースになります。ただし、次のコードを使用して別の区切り文字を設定することもできます（区切り文字は ","、";"、""）：

```
report.setExportDelimiter(IDelimiterConstants.COMMA);     //where report is an Object of type QbReport
```

レポートを Excel または Excel 2007 ファイルにエクスポートするときに、各数値が 1 つのセルだけを占めるようにメソッドを設定することもできます。これは、エンドユーザーがエクスポートされたレポートで Excel 機能を使用する場合に役立ちます。この機能を使用するには、レポートをエクスポートする前に、次のメソッドを true に設定します。

```
QbReport.setExcelExportFitCell(boolean b)
```

メソッドを使用して **QbReport** オブジェクトをバイト配列にエクスポートすることもできます。

```
public byte[] exportReportToByteArray();
```

これは、.rpt をバイト配列の形で与えます。これは、**QbReport** オブジェクトをシリアライズする必要がある場合に便利です。バイト配列から **QbReport** オブジェクトを再作成するときに、サブレポート、ドリルダウン、および/またはチャート（EspressManager を使用していない場合）のディレクトリを指定する必要があります。

4.3.5.7.1 複数のページのエクスポート

単一の「ページ」ではなく、レポートを複数の「ページ」にエクスポートすることもできます。つまり、レポートをさまざまなファイルにエクスポートして、断片的に表示することができます。このオプションは、HTML、DHTML、および XML（Data + Format）エクスポートでのみ使用できます。

次のコードを使用して、レポートを複数のファイルにエクスポートします：

```
report.setExportToMultiPages(true);
```

このオプションを使用すると、いくつかのファイルが生成されます（生成されるファイルの数はレポート内のページ数に等しい）。最初のファイルの名前は、エクスポートメソッドで指定されたファイル名

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

と同じです。後続のファイルには、ファイル名の末尾に対応するページ番号が付いています。最後のページを含むファイルは、ページ番号ではなくファイル名に「LAST」を含みます。たとえば、3 ページのレポートを **SalesSummary** にエクスポートするとします。生成されるファイルは、**SalesSummary.html**（最初のページを含む）、**SalesSummary_2.html**（2 番目のページを含む）、**SalesSummary_LAST.html**（最後のページを含む）です。

特定のページを一度にエクスポートして（レポート全体ではなく）、ページを特定のファイルにエクスポートすることもできます。これは以下のメソッドを使用して行われます：

```
export(int format, OutputStream out, int pageNumber);
```

たとえば、レポートのページ 3 を HTML ファイル ThirdPage にエクスポートするには、次のコードを使用します：

```
FileOutpuStream dout3 = new FileOutputStream("ThirdPage.html");
report.export(QbReport.DHTML, dout3, 3);
dout3.close();
```

最後のページをエクスポートするには、**pageNumber** の引数として "-1"（二重引用符なし）を渡すことができます。また、特定のページをエクスポートするために、**setExportToMultiPages** を true に設定する必要はありません。

4.3.5.7.2 PDF エクスポートオプション

QbReport クラスの特定のコンストラクタでは、**userPass** パラメータ、**ownerPass**、および **permissions** が使用可能です。これらのパラメータは、それぞれ PDF 文書のユーザーのユーザーパスワード、所有者パスワード、および権限を設定するために使用されます。すべてのアクセス許可は、permissions 引数の指定された権限（ユーザーパスワードを使用して PDF ドキュメントを開くユーザーのみ）が使用できる間に、PDF ドキュメントの所有者（PDF ドキュメントを開くために所有者パスワードを使用するユーザー）に与えられます。オプション付き PDF へのエクスポートの詳細については、EspressReport Java API Documentation を参照してください。

PDF オプションを設定するエクスポートメソッドは、読者の便宜のためにここにリストされています。

```
public void export(int format,
                   java.io.OutputStream out,
                   int exportPage,
                   java.lang.String userPass,
                   java.lang.String ownerPass,
                   int permissions);
```

また、PDF エクスポートに埋め込み Java スクリプトを渡すことで、PDF コンテンツがクライアントブ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ラウザにストリーミングされたときに Java スクリプトが実行されるようにすることもできます。次の方法では、Java スクリプトを PDF コンテンツに埋め込むことができます

```
public void export(int format,
                   java.io.OutputStream out, //Or java.lang.String specifying the file Name
                   java.lang.String userPass,
                   java.lang.String ownerPass,
                   int permissions,
                   java.lang.String javaScript);
```

たとえば、次のメソッドは、クライアントブラウザで PDF を表示したときに自動的に印刷するように PDF を設定します。

```
QbReport report = new QbReport(.....);
.....
.....
String javaScript = "this.print(true);¥r";
report.export(QbReport.PDF, someOutputStream, null, null, QbReport.AllowAll, javaScript);
```

上記の呼び出しでは、レポートを PDF コンテンツ（すべての権限を持つ）としてクライアントのブラウザにデータを渡すストリーム（**someOutputStream**）にエクスポートします。 PDF がクライアントブラウザにロードされると、javascript が自動的に実行されます（この場合、ブラウザにコンテンツの印刷が指示されます）。

4.3.5.7.3 スタイルシートを使用した DHTML コンテンツのエクスポート

レポートを DHTML 形式または HTML 形式にエクスポートする際に使用するスタイルシートを指定できます。レポートをエクスポートする前に、内部スタイルシートまたは外部スタイルシートを指定できます。

内部スタイルシートを使用してレポートをエクスポートするには、**QbReport** で次のメソッドを使用します。

```
public void setUseStyleSheet(boolean state);
```

外部スタイルシートを使用してレポートをエクスポートするには、まず、**ReportElement** で次のメソッドを使用して、指定した **ReportElement** オブジェクトに使用するスタイルを指定する必要があります。

```
public void setStyleName(String newStyleName);
```

QbReport で次のメソッドを使用して、外部スタイルシートファイルの名前を指定します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
public void setExternalStyleSheetName(String css);
```

たとえば、次のコード：

```
report.getTable().getColumn(0).setStyleName("style_1"); //where report is an object of type QbReport
report.setExternalStleSheetName("http://someMachine/someDirectory/styles.css");
```

指定されたスタイルシートファイル（この場合は **http://someMachine/someDirectory/styles.css**）から style_1 を使用するようにレポートの最初の列を設定します。

4.3.5.7.4 HTML ブロックのエクスポート

レポートは、完全な HTML ファイルではなく、HTML コードのブロックとしてエクスポートできます。これにより、生成された HTML レポートにカスタムコンテンツを追加することができます。

以下のコード：

```
report.setHeadTagIncluded(false);          //where report is an object of type QbReport
```

レポートのエクスポート時に DHTML テーブルとしてレポートを生成します。このブロックは、レポート以外の内容を含む別のページに含めることができます。

4.3.5.7.5 複数ページの DHTML/HTML エクスポートのカスタムリンク

また、ページの上部に独自のリンクを作成したり、ページの上部に生成されたリンクを削除して、単一ページおよび複数ページのエクスポートを行うこともできます。これにより、カスタムリンクを HTML レポートと DHTML レポートの両方に追加することができます。

カスタムリンクは、IHTMLLinksProvider インタフェースを実装することで生成できます。たとえば、アプリケーションまたはアプレットとして実行できる以下のコードは、カスタムリンクを作成する方法を示しています：

```
Component doCustomLinks(Object parent) {
        QbReport.setEspressManagerUsed(false);
        QbReport report = new QbReport(parent, "CustomLinks.rpt");
        report.setHTMLLinksProvider(this);
        QbReport.setExportToMultiPages(true);

        try
        {
                report.export(IExportConstants.DHTML, "CustomLinks.html");
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
		} catch (Exception ex) {
			ex.printStackTrace();
		}

		return (new Viewer().getComponent(report));

}

public HTMLBlock getLinksForDHTML(int cPage, int tPage, String prefix, boolean top)
{

		String linksText = "<CENTER><font face=¥¥¥"verdana¥¥¥" size=¥¥¥"2¥¥¥">" +

				"<a href=¥¥¥"" + HTMLBlock.getFirstFileName(prefix) +
"¥¥¥">FIRST</a> | ";
		if (cPage <= 2) {
			linksText += "<a href=¥¥¥"" + HTMLBlock.getFirstFileName(prefix) +
"¥¥¥">PREV</a> | "
				+ "<a href=¥¥¥"" + HTMLBlock.getNextFileName(cPage, tPage,
prefix)
				+ "¥¥¥">NEXT</a> | " + "<a href=¥¥¥"" +
HTMLBlock.getLastFileName(prefix)
				+ "¥¥¥">LAST</a> [" + "<a href=¥¥¥""
				+ HTMLBlock.getCurrentFileName(cPage, tPage, prefix)
				+ "?x=x¥¥¥" target=¥¥¥"print¥¥¥">Print Version</a>]" +
"</font></CENTER>";
		} else {
			linksText += "<a href=¥¥¥"" + HTMLBlock.getPreviousFileName(cPage, tPage,
prefix)
				+ "¥¥¥">PREV</a> | " + "<a href=¥¥¥""
				+ HTMLBlock.getNextFileName(cPage, tPage, prefix) +
"¥¥¥">NEXT</a> | "
				+ "<a href=¥¥¥"" + HTMLBlock.getLastFileName(prefix) +
"¥¥¥">LAST</a> ["
				+ "<a href=¥¥¥"" + HTMLBlock.getCurrentFileName(cPage,
tPage, prefix)
				+ "?x=x¥¥¥" target=¥¥¥"print¥¥¥">Print Version</a>]" +
"</font></CENTER>";
		}

		String text = "<SCRIPT>¥n" + "document.write(¥"" + linksText + "¥");¥n" + "</SCRIPT>";
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        return new HTMLBlock(text, 100);

}

public HTMLBlock getLinksForHTML(int cPage, int tPage, String prefix, boolean top)
{
        return getLinksForDHTML(cPage, tPage, prefix, top);
}

public HTMLBlock getTOCLinkForDHTML(int cPage, int tPage, String body) {
        return null;
}
```

フルソースコード
エクスポートされた結果

同様に、他のリンクを生成してレポートに設定することもできます。

4.3.5.7.6 DHTML/HTML エクスポートのピクセル/インチ比率の設定

EspressReport では、HTML または DHTML コンテンツをエクスポートするときに、ピクセル/インチ比率を指定できます。これは、プラットフォーム間でレポートを移動および展開する場合や、グラフィックスなしのシステムでレポートをエクスポートする場合に特に便利です。**QbReport** で次の方法を使用して比率を指定できます。

```
public void setPixelPerInchForExport(int pixelPerInchRatio);
```

4.3.5.7.7 プレロードチャート

EspressReport は、レポートをエクスポートする前に独立したデータソースを持つチャートをプリロードすることができます。プレロードは同時に行われるため、エクスポートのパフォーマンスが向上します（特に、レポートに複数のグラフがある場合）。レポートをエクスポートする前に、次のメソッドを呼び出してグラフをあらかじめロードするように指定できます。

```
public void preloadChartObjects();
```

4.3.5.7.8 .view ファイル用の IExportThreadListener

レポートは.view ファイルにエクスポートできます。これらのファイルは、Page Viewer で（Report Viewer でレポート全体を読み込むのではなく）一度に数ページずつレポートを表示するために使用されます。.view ファイルへのエクスポートは、1 つのエクスポート方法で複数のページが作成されるという点で、複数のページにエクスポートするのと同じです（レポートのページ数が多いほど、生成されるファイルの数が増えます）。ただし.view ファイルが生成されない限り（レポートのページは.page ファイルとして生成されます）、Page Viewer を起動することはできません。EspressReport は、最初の

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ページがエクスポートされたときとエクスポートが完了したときにアクションを実行するために実装できる IExportThreadListener インタフェースを提供します。

IExportThreadListener を使用した例を以下に示します：

```
public class ExportView extends Applet implements IExportThreadListener {
	static final long serialVersionUID = 1;

	public static void main(java.lang.String[] args) {

		try {
			ExportView report = new ExportView();
			report.exportView(null);
		} catch (Exception ex) {
			ex.printStackTrace();
		}

	}

	//	creates report and return it
	void exportView(Object parent) throws Exception {

		//		Turn off ReportServer as it is not needed.
		QbReport.setEspressManagerUsed(false);

		//		Create the colinfo array to be used in the QbReport constructor
		QbReport report = new QbReport(parent, "ExportView.rpt");

		report.setMultiPageExp(true);
		report.export(QbReport.VIEW, "ExportView", new Properties(), this);
		System.out.println("DONE!");

	}

	public void endAction() {
		System.out.println("END ACTION");
	}

	public void firstPageFinishedAction() {
		System.out.println("FIRST PAGE FINISHED ACTION");
		quadbase.reportdesigner.PageViewer.Viewer.main(new String[]
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
{ "ExportView.view" });
        }

}
```

フルソースコード

4.3.5.7.9 仮想メモリ/ページング

メモリに最適化されたエクスポートに加えて、EspressReport は大量のデータを処理するページング機能も備えています。ページングでは、メモリ量と一時ディレクトリを指定します。使用されるメモリの量が指定された数を超えると、データは圧縮され、指定された temp ディレクトリのディスクに格納されます。

この機能は EspressManager の有無にかかわらず動作します。ただし、EspressManager を使用する場合は、クライアントコードと EspressManager の両方で同じオプション値を使用することをお勧めします。

メモリページングエクスポートを使用するには、**QbReport** コンストラクタを呼び出す前に、次のコードを追加する必要があります：

```
QbReport.setTempDirectory
        (<specify the temp directory to store data.  Default is ./temp>);
QbReport.setMaxFieldSize(int fieldSize);
QbReport.setPagingThreshold(int pagingThreshold);
QbReport.setPageBufferSize(int bufferSize);
QbReport.setTotalPageBufferSize(int totalBufferSize);
```

Max Field Size:

この引数は、varchars などの大きなフィールドタイプの予想される最大フィールドサイズを指定します。この値より大きいフィールドは必ずしも切り捨てられる必要はありません。動作は、許容される Total Paging Buffer Size に関連した現在のメモリ使用量に依存します。利用可能なメモリの量が減少すると、これらの大きなフィールドに格納されるデータの量は、この値に達するまで減少します。たとえば、指定された最大フィールドサイズの値より大きいフィールドサイズのレポートをユーザーが実行しているとします。サーバがビジーでなく、使用可能なメモリが十分にある場合、レポートは切り捨てなしで完全なフィールドを生成します。ただし、サーバの負荷が高く、使用可能なメモリがゼロに近い場合、レポートのフィールドは最大フィールドサイズに切り捨てられます。数値型、ブール型、日付型、時間型、および文字型のデータの場合、データは切り捨てられません。最大フィールドサイズより小さいサイズ制限（たとえば、最大フィールドサイズ= 500、フィールドは varchar（20）として定義されている）で定義されているデータベースフィールドの場合、最大フィールドサイズの代わりにデータベースの制限が使用されます。この値はバイト単位で設定されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Paging Threshold:

このプロパティは、ページング機能がいつアクティブになるかを指定します。このしきい値に達すると、サーバはデータをシステム上の一時ファイルにページングを開始します。この引数を-1 に設定すると、データは決してページングされません。この値は、Page Buffer Size オプションとは関係ありません。この値が Page Buffer Size より大きい場合、しきい値に達するまでシステムはページングを開始しません。したがって、各レポートまたはグラフは、Page Buffer Size で指定された量より多くのメモリを使用することがあります。 この値はメガバイト単位で設定され、デフォルトは-1 です。

Page Buffer Size:

この引数を使用すると、ページング機能が呼び出されたときに各レポートまたはチャートが使用するメモリ量を設定できます。バッファのサイズはパフォーマンスに影響します。バッファサイズが大きいほど、レポートやグラフが速く生成されますが、使用されるメモリが増えます。ページングシステムによって使用される総メモリ量が Total Paging Buffer Size に近づくと、指定された合計量を超えないように、新しいレポートに提供されるメモリ量が減少し始めます。この値はメガバイト単位で設定されます。

Total Paging Buffer Size:

この引数を使用すると、すべてのレポートとチャートでページング機能によって使用されるメモリの合計量を設定できます。メモリ使用量がこの値に近づくにつれて、各レポートに割り当てられるメモリは減少します。この値はメガバイト単位で設定されます。

4.3.5.7.10 ストリーミングレポート

ローカルドライブへのエクスポートに加えて、レポートをバイトストリームとしてエクスポートし、Web ブラウザに直接ストリーミングすることもできます。ただし、チャート、ドリルダウンおよびパラメータが正しく機能するためには、このエクスポート方法ではいくつかのサポートサーブレットが使用可能である必要があります。これを行う 3 つのサーブレットは、**RPTImageGenerator**、**DrillDownReportServlet** 、および **ParamReportGeneratorServlet** です。それらは **<EspressReport>/WEB-INF/classes/**ディレクトリにあります。3 つのファイルは、サーブレットランナーのサーブレットディレクトリにコピーする必要があります。サーブレットに接続するには、次のコードを使用します：

```
report.setDynamicExport(true, "Machine Name or IP Address", Port);
report.setServletDirectory("Servlet Directory");
```

これらのメソッドは、これらのサーブレットを呼び出すと、正しいマシンとポートを指していることがわかります。サーブレットは URL の **http://<MachineName:>:<Port:>/<ServletDirectory:>/<Servlet:>**を使用して呼び出されます。レポートをストリーミングするときに動的エクスポートを設定する必要がありますが、サーブレットディレクトリの設定はオプションです。使用するサーブレットディレクトリを指定しないと、自動的に

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

servlet/に設定されます。

以下は、レポートを DHTML にエクスポートしてブラウザにストリームする例です。この例を実行するには、ソースコードを設定してコンパイルし、サーブレットランナーのサーブレットディレクトリにクラスファイルをデプロイする必要があります。**reportTemplate** 変数を、アプリケーションサーバの作業ディレクトリに対する絶対パスまたはパスのいずれかに置き換えます。作業ディレクトリは通常サーブレットディレクトリと同じではないことに注意してください。また、**ReportAPIWithChart.jar** と **qblicense.jar** をアプリケーションサーバのクラスパスに追加してください。どちらの jar ファイルも <**EspressReport**>/**lib/**にあります。

```
public class StreamingReport extends HttpServlet {
	static final long serialVersionUID = 1;

	@Override
	public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException,
			IOException
	{
		//Do not use EspressManager
		QbReport.setEspressManagerUsed(false);

		//Location of report template
		String reportTemplate = "StreamingReport.rpt";

		//Create QbReport object
		QbReport report = new QbReport(null, reportTemplate);

		//Set up connection information for RPTImageGenerator and DrillDownReportServlet
		report.setDynamicExport(true, "localhost", 8080);
		report.setServletDirectory("servlet/");

		//Export the report to DHTML and stream the bytes
		ByteArrayOutputStream reportBytes = new ByteArrayOutputStream();

		try {
			report.export(QbReport.DHTML, reportBytes); //where report is an object of type QbReport
			//			report.export(QbReport.PDF, reportBytes);
	//where report is an object of type QbReport
		} catch (Exception ex)
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                {
                        ex.printStackTrace();
                }

                res.setContentType("text/html"); //where res is the HttpServletResponse
                //                  res.setContentType("application/pdf");     //where res is the
HttpServletResponse
                res.setContentLength(reportBytes.size());

                OutputStream toClient = res.getOutputStream();
                reportBytes.writeTo(toClient);
                toClient.flush();
                toClient.close();
        }
}
```

フルソースコード

この例を実行するには、コードによってアクセスされる場所にレポートテンプレートをコピーしてください。相対パス（例のように）を使用している場合は、現在のディレクトリがアプリケーションサーバの作業ディレクトリであることに注意してください。たとえば、Tomcat の作業ディレクトリは **<Tomcat>/bin** なので、**<Tomcat>/templates/**ディレクトリを作成し、そこにレポートテンプレートをコピーしてコードを実行する必要があります。結果レポートは次のとおりです：

Woodview Sales By Region

CategoryName	ProductName	East	Midwest	South	West	Total Sales
Arm Chairs	Adad Chair	$963,414.00	$292,896.00	0	0	$1,256,310.00
	Cula Chair	$1,526,364.00	$644,274.00	$960,336.00	$216,432.00	$3,347,406.00
	Marduk Chair	$1,478,736.00	$336,960.00	0	0	$1,815,696.00
	Nabu Chair	$295,488.00	0	$270,864.00	$762,156.00	$1,328,508.00
	Ningirsu Chair	$317,304.00	$154,872.00	$232,308.00	0	$704,484.00
	Nisaba Chair	$2,827,656.00	$389,448.00	$558,900.00	$201,204.00	$3,977,208.00
	Nusku Chair	$1,311,984.00	$206,550.00	$895,050.00	$459,000.00	$2,872,584.00
	Shugamuna Chair	$1,374,894.00	0	$563,220.00	0	$1,938,114.00
	Shimaliya Chair	$782,784.00	0	$977,616.00	$173,880.00	$1,934,280.00
	Category Total:	**$10,878,624.00**	**$2,025,000.00**	**$4,458,294.00**	**$1,812,672.00**	**$19,174,590.00**
Double Dressers	Sekhmet Dresser	$608,148.00	0	0	0	$608,148.00
	Serket Dresser	$2,214,864.00	0	$1,655,640.00	0	$3,870,504.00
	Set Dresser	$1,421,280.00	0	$671,328.00	0	$2,092,608.00
	Shu Dresser	0	$1,148,256.00	0	0	$1,148,256.00
	Tefnut Dresser	$873,828.00	0	0	0	$873,828.00
	Thoth Dresser	$2,097,630.00	0	$505,440.00	0	$2,603,070.00
	Category Total:	**$7,215,750.00**	**$1,148,256.00**	**$2,832,408.00**	**$0.00**	**$11,196,414.00**

ストリーミングレポート

RPTImageGenerator

RPTImageGenerator は、チャートまたは BLOB イメージを含むレポートを HTML または DHTML 形式にエクスポートするときに使用されます。HTML と DHTML ファイルは純粋にテキストです。したが

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

って、チャートとイメージは別々に保存する必要があります。レポートをローカルにエクスポートすると、チャートはイメージファイルとして保存され、ファイルパスは HTML ページの IMG タグに追加されます。

```
<IMG SRC="stream_0.jpg" ALIGN="CENTER" VALIGN="TOP"BORDER="0" HEIGHT="288" WIDTH="576"></IMG>
```

同様に、DHTML ファイルをストリーミングすると、テキストコンポーネントのみがブラウザに送信されます。グラフは **RPTImageGenerator** サーブレットで処理されます。IMG タグをイメージファイルに指定する代わりに、タグは **RPTImageGenerator** サーブレットに設定され、チャートイメージの ID をパラメータとして追加します。

```
<IMG SRC="http://localhost:8080/servlet/RPTImageGenerator?ID=1166125348416_0" ALIGN="CENTER" VALIGN="TOP"BORDER="0" HEIGHT="288" WIDTH="576"></IMG>
```

ローカルに保存されたイメージを含むレポートには、サーブレットは必要ありません。IMG ソースはイメージファイルを直接指します。ただし、データソース（BLOB）から取得されたイメージはチャートと同様に扱われ、**RPTImageGenerator** サーブレットの助けが必要です。

DrillDownReportServlet

レポートをバイトストリームまたはファイルにエクスポートするかどうかにかかわらず、ドリルダウンを使用したレポートでは、**DrillDownReportServlet** を適切に機能させるにはアクセス可能である必要があります。ドリルダウンレポートを DHTML または PDF フォーマットにエクスポートする場合は、サーブレットランナーのサーブレットディレクトリに **DrillDownReportServlet** がなければなりません。このサーブレットは、次のレベルへのリンクを提供します。

```
<a href="http://localhost:8080/servlet/DrillDownReportServlet?FILENAME=DrillDown%2Fstream_lvl11.rpt&FORMAT=4&PARAM0=112+Ishtar+Chair" title="tt" alt="tt">
Ishtar Chair<BR></a>
```

ドリルダウンレベルにはパラメータ化されたクエリが含まれている必要があるため、**ParamReportGeneratorServlet** もサーブレットディレクトリにあることを確認する必要があります。**ParamReportGeneratorServlet** の詳細は、次のいくつかのセクションで使用できます。

レポートを HTML または DHTML にエクスポートすると、ドリルダウンレポートにリンクされた列がハイパーテキストリンクとして表示されます。PDF フォーマットでは、リンクされた列は他の列と同じように見えません。ただし、その列の上にマウスを置くと、カーソルが手に変わります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

CategoryName	ProductName	UnitPrice	StainPrice	UnitsInStock
Arm Chairs	Adad Chair	452	35	16
	Cula Chair	468	33	4
	Marduk Chair	489	31	12
	Nabu Chair	456	31	25
	Ningirsu Chair	478	35	15
	Nisaba Chair	414	29	14
	Nusku Chair	425	31	36
	Sbuqamuma Chair	445	32	45
	Shimaliya Chair	424	36	31
Double Dressers	Sekhmet Dresser	1,877	412	14
	Serket Dresser	1,761	414	4
	Set Dresser	1,645	427	17

ルートレベル

クリックされたリンクに応じて、サーブレットは適切な次のレベルのレポートを配信します。

CategoryName :	Arm Chairs	
ProductName	OrderDate	Quantity
Marduk Chair	Mar 21, 2002	8
Marduk Chair	May 27, 2003	16
Marduk Chair	Jun 9, 2003	12
Marduk Chair	Oct 12, 2003	14
Marduk Chair	Dec 2, 2003	18
Nabu Chair	Feb 12, 2002	11
Nabu Chair	Sep 17, 2002	16
Nabu Chair	Nov 12, 2002	14
Nabu Chair	Dec 2, 2003	12
Cula Chair	Apr 14, 2001	16
Cula Chair	Sep 17, 2001	16
Cula Chair	Nov 16, 2001	18
Cula Chair	Apr 22, 2002	8
Cula Chair	Oct 24, 2002	16
Cula Chair	Dec 4, 2002	8
Cula Chair	Apr 22, 2003	18
Cula Chair	Jul 21, 2003	3

ドリルダウンレベル

DHTML または PDF 以外のフォーマットにエクスポートされたドリルダウンレポートには、現在のレベルのみが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フルソースコード

HTML パラメータページの生成

レポートをストリーミングする場合、API（パラメータ化されたレポートで説明）を使用してパラメータ値を設定する強力な代替方法は、テキストまたは選択入力を含む HTML フォームを自動的に生成することです。ユーザーがフォームを送信すると、パラメータ値が処理される **ParamReportGeneratorServlet** に渡され、これらのパラメータ値で初期化されたレポートがストリームバックされます。

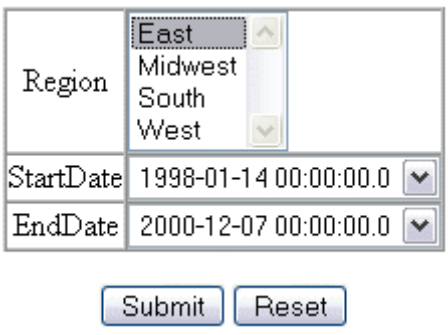
この目的のための主なクラスは、**ParameterPage**、**ParameterPageWriter**、**HtmlParameterPageWriter**、および **CssHtmlParameterPageWriter** です。HTML パラメータページを生成する一般的な方法には、次のコードがあります：

```
OutputStream out = ...; //response.getOutputStream();
ParameterPage paramPage = QbReport.getParameterPage();
Writer writer = new PrintWriter(out);
HtmlParameterPageWriter paramPageWriter = new HtmlParameterPageWriter(paramPage, writer);
paramPageWriter.writePage();
writer.flush();
writer.close();
```

フルソースコード

上記の機能を使用する場合は、サーブレットランナーのサーブレットディレクトリに **ParamReportGeneratorServlet** が必要です。このサーブレットは、HTML ページで指定されたパラメータを使用して、指定されたフォーマットでレポートを生成します。

この例を実行すると、次のページが自動的に生成されます：



生成されたパラメータページ

選択した値に応じて、送信ボタンをクリックすると、次のレポートが表示されることがあります：

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Region	Company	ContactName	State	Orders
East	Furniture Gallery	Eddie Birdsell	NY	5
	Eastern Treasures	Frances Polk	NJ	10
	Gale Distributors	Herb Gale	NY	2
	Allied Furniture Emporium	Matilda Gladwaller	NY	9
	Furniture Palace	Muriel Fedder	NJ	8
	Campbell Interiors	Paul Campbell	NY	3

Region	Company	ContactName	State	Orders
Midwest	All Unfinished Furniture	Arthur Childs	OH	4
	George Park Imports	Ernest Morrow	PA	2
	Woodworks Furniture	Sally Hayes	PA	7
	Imports & Leather Gallery	Seymour Glass	IL	8

パラメータ化されたレポート

パラメータページ全体を自動的に生成すると便利ですが、ページの外観をより詳細に制御する必要がある場合があります。これを実現する 1 つの方法は、メインフォームのみを生成することで、メインフォームの周りにコンテンツをデザインして独自の外観を提供することができます。別の方法は、CSS を使用して目的のフォーマットを設定することです。3 番目の方法は、フォーム内の各コンポーネントを個別に生成して、ページ上のすべての要素を個別に制御することです。これらのさまざまな方法については、次の 2 つのセクションで詳しく説明します。

- カスケーディングスタイルシート（CSS）の活用
  HTML パラメータページを変更する 2 つの強力な方法は、フォームの周りにコンテンツを追加し、カスケードスタイルシートを利用することです。これらのメソッドを使用すると、過剰な量のコードを書く必要なく、Web アプリケーションに一貫した外観を維持できます。

  以下の例は、これらのメソッドを使用して、定義済みの色、フォント、スタイル、および書式を使用してパラメータページを表示する方法を示しています。この例を実行するには、テンプレートパスを変更する必要があります。相対パスを使用すると、アプリケーションサーバの作業ディレクトリからの相対パスになります。コード内の CSS リンクからアクセスできるように、CSS ファイルを Web アプリケーションのルートディレクトリにコピーする必要があります（CSS ファイルを別の場所に配置してリンクを変更することもできます）。例では、ドリルダウンレポートが含まれているため、サーブレットディレクトリで ParamReportGeneratorServlet と DrillDownReportServlet の両方を使用できるようにする必要があります。

```
CssHtmlParameterPageWriter cssParamPageWriter =
                    new CssHtmlParameterPageWriter(paramPage, tempStringWriter);

//Specify the css file to be used
cssParamPageWriter.setCssFile("http://localhost:8080/CssExample.css");
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
//Create head of html page, adding title and css specification
cssParamPageWriter.write("<HEAD>¥n" +
                    "<link REL=¥"stylesheet¥" TYPE=¥"text/css¥" " +
                    "href=¥"" + cssParamPageWriter.getCssFile() + "¥">" +
                    "¥n<TITLE>Welcome to Woodview</TITLE>¥n" +
                    "</HEAD>");

//Body section - Most of the formatting is done in the css file,
//very little is needed here
cssParamPageWriter.write("¥n<body><p>
");

//Add title and fieldset
cssParamPageWriter.write("<TABLE class=¥"outer¥"><TR><TD>" +
                    "<p class=¥"heading2¥">Welcome to the Order History Database." +
                    "</TD></TR><TR><TD> </TD></TR>" +
                    "<TR><TD><FIELDSET><LEGEND>Select</LEGEND>");
//Add the parameter forms
cssParamPageWriter.writeBodyBody();

//Finish it off
cssParamPageWriter.write("</FIELDSET></TD></TR>" +
                    "</TABLE></body>¥n</html>");
```

フルソースコード

上記のコードセグメントでは、タイトルを挿入できるように<HEAD>が手動で書き込まれます。本文セクションでは、フォームのみが生成されます。このメソッドを使用すると、ユーザーはタイトル、イメージ、およびその他のコンポーネントを追加して、独自の外観を維持できます。提供された CSS ファイルに応じて、結果は次のようになります：

CSS を使用したパラメータページ

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

選択したパラメータオプションに応じて、結果は次のようになります：

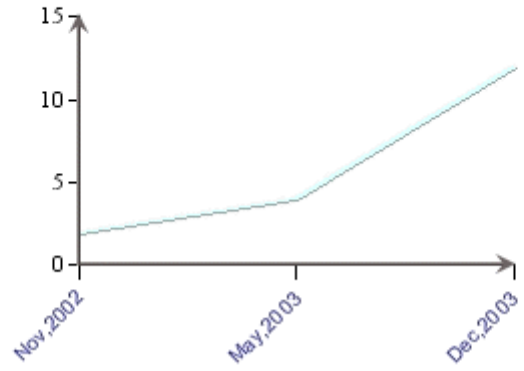
Woodview Furniture

Product Order History

ProductID : 30

ProductName : Horus Table **Description :** Redwood

OrderID	OrderDate	UnitPrice	Quantity
10,049	Nov 12, 2002	1,354	2
10,068	May 22, 2003	1,354	4
10,096	Dec 9, 2003	1,354	12
		Total Sold:	18



結果の例

- パラメータページのカスタマイズ
  パラメータページをカスタマイズするもう 1 つの方法は、HtmlParameterPageWriter または CssHtmlParameterPageWriter を拡張する Java クラスを記述することです。この方法では、これらのクラスで使用可能な保護されたメソッドにアクセスできます。これらのクラスで利用可能な一連のメソッドの詳細については、APIDocs を参照してください。

  以下は、**HtmlParameterPageWriter** クラスを拡張した例です。この例を実行するには、2 つの html ページをアプリケーションサーバのルートディレクトリにコピーする必要があります。テンプレートファイルと Java クラスファイルは、アプリケーションサーバの作業ディレクトリとサーブレットディレクトリにそれぞれコピーする必要があります。この例では、**ParamReportGeneratorServlet** と **DrillDownReportServlet** をサーブレットディレクトリに配置

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

する必要があります。

```
private class MyHtmlParameterPageWriter extends HtmlParameterPageWriter{

    public MyHtmlParameterPageWriter(ParameterPage pp, StringWriter sw) {
        super(pp,sw); }

    public void printCustom() {
        try {
            write("<BODY bgcolor=#FDFAED>");

            write("<FORM action = ¥"" + servletName +"¥" target=¥"main¥" method = GET>¥n");
            write("<TABLE                                    ALIGN=¥"CENTER¥"><TR><TD><IMG
src=http://www.quadbase.com/FurnitureImages/Woodview.gif></TD>");
            write("<TD WIDTH=50></TD><TD>");
            writeParamTable();

            write("</TD><TD WIDTH=50></TD><TD>");
            writeSubmitButton();
            write("</TD>");
            write("</TR></TABLE>");

            writeHiddenParamValue("ReportFilePath", reportLocation);
            write("¥n");

            writeHiddenParamValue("ReportExportFormat", ""+format);
            write("¥n");

            writeHiddenParamValue("ServerName", servletAddress);
            write("¥n");
            writeHiddenParamValue("ServletRunnerPort", ""+servletPort);

            write("¥n");
            writeHiddenParamValue("ServletDirectory", servletDirectory);

            write("¥n</FORM>");
            html.bodyEnd();
            html.htmlEnd(); } catch (Exception e) {
            e.printStackTrace(); } }

}
```

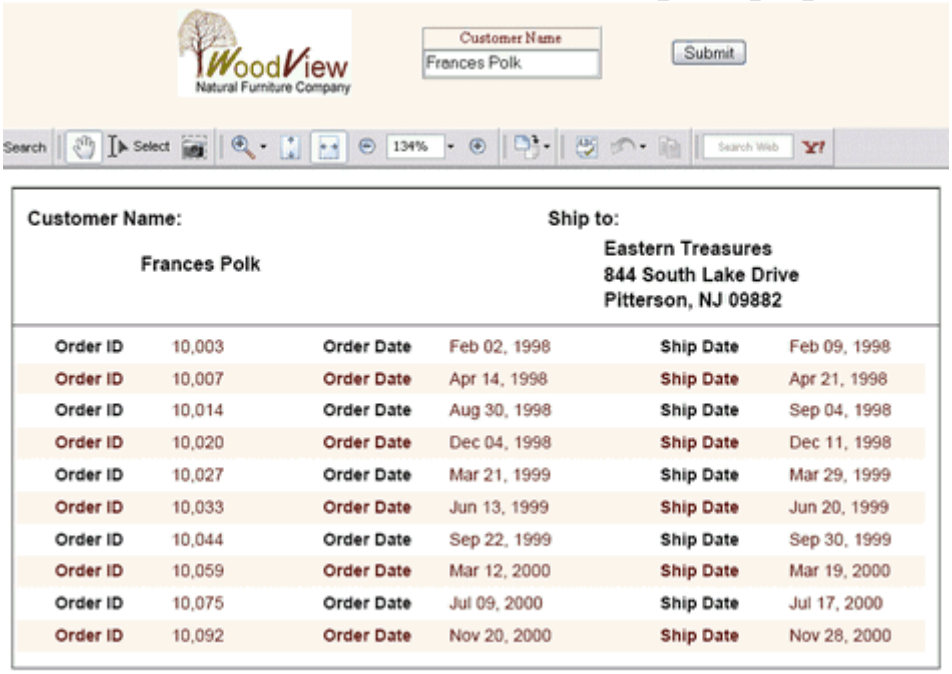
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フルソースコード

保護されたメソッドを使用することにより、FORM タグや隠しパラメータなど、コード内にいくつかの詳細を追加する必要があることに注意してください。ただし、これにより、会社のロゴなどの他の要素をページのフォームセクションに追加し、要素を好みに合わせることもできます。この方法では、リセットボタンなどの特定の要素をページから削除することもできます。

ファイルが正しい場所にコピーされたら、ブラウザに **http://<IP アドレス/localhost>:<Port>/CustomParamPage.html** という URL を入力して、この例を実行できます。2 つのフレームがあるページが表示されます。上部のフレームはパラメータページで、下部のフレームは最初は簡単なメッセージを表示します。レポートを表示するには、顧客のフルネーム（例：Francis Polk）を入力します。



注文履歴レポート

この例では、DHTML の代わりに PDF にエクスポートします。PDF フォーマットでリンクは表示されませんが、Order ID 列の上にマウスを置くと、マウスポインタが手元になり、この列からドリルダウンできることが警告されます。選択した注文 ID に応じて、フォローレポートが表示されることがあります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Order Summary for ID:10,007

Side Chairs

Product Name	Description	Unit Price	Stain Price	Stain	Quantity	SubTotal
Ninhursag Chair	Cedar Bowback Chair	$369.00	$25.00	No	14	$5,166.00

Arm Chairs

Product Name	Description	Unit Price	Stain Price	Stain	Quantity	SubTotal
Cula Chair	Birch Fanback Armchair	$468.00	$33.00	No	16	$7,488.00

Oval Tables

Product Name	Description	Unit Price	Stain Price	Stain	Quantity	SubTotal
Ma'at Table	Cedar Oval Table	$1,875.00	$378.00	Yes	2	$4,506.00
					Total:	$17,160.00
					Tax (7.25%):	$1,244.10
					Grand Total:	$18,404.10

オーダー履歴の詳細（ドリルダウン）

4.3.5.7.11 レポート API から Report Designer を呼び出す

Report Designer は、アプリケーションまたはアプレットのレポート API から呼び出すことができます。コードに応じて、パラメータを渡して、指定した.rpt ファイルまたは指定したデータソースまたはその他のパラメータを使用して Report Designer を開くことができます。

レポート API から Report Designer を呼び出すには、次の操作を行う必要があります：

EspressManager が起動して動作していることを確認してください。

ReportAPIWithChart.jar、**EspressManager.jar** および **qblicense.jar** を **CLASSPATH** に追加します。

関連する API 呼び出しを使用して EspressManager に接続する情報が指定されていることを確認してください（EspressManager が別のマシンにある場合、または 22071 以外のポート番号で起動した場合は特に重要です）。

images、reportimages、および backgroundimages ディレクトリを作業ディレクトリにコピーするか、コード内**の QbReportDesigner.setUseSysResourceImages**（**true**）を使用して jar ファイルのイメージを使用します。

EspressManager の**-RequireLogin** および**-QbDesignerPassword** フラグに応じて、EspressManager に接続するためにユーザーID およびパスワードを渡す必要があります。EspressManager に **-RequireLogin** フラグが設定されている場合は、**setVisible()**または **getDesigner** メソッドを呼び出す前に、次のコード行を追加する必要があります：

```
public login(String userName, String password); //Method found in QbReportDesigner class
```

EspressManager に**-QbDesignerPassword** を指定した場合、**setVisible()**または **getDesigner** メソッ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ドを呼び出す前に、次のコード行を追加する必要があります：

```
public login(String password); //Method found in QbReportDesigner class
```

また、Report Designer に外観を指定することもできます（Report Designer はデフォルトでシステムの外観を使用します）。これは、**QbReportDesigner** で次のメソッドを使用して行います：

```
public static void setLookAndFeel(javax.swing.LookAndFeel newLookAndFeel);
```

Report Designer を使用して Report Designer を呼び出すには、以下のような方法があります。サンプルコードをアプレットとして実行している場合は、EspressManager マシンを **127.0.0.1**（または **localhost**）から EspressManager マシンの名前または IP アドレスに変更する必要があります。

レポートテンプレートを指定する

Report Designer は、指定したレポートテンプレートファイルを使用して開くことができます。これにより、エンドユーザーは Report Designer GUI で独自のカスタムレポートを作成し、完成したレポートを表示できます。

次のコンストラクタが使用されます：

```
QbReportDesigner(Object parent, String templateFileName);
```

指定された.rpt ファイルを使用して Report Designer を呼び出す例を以下に示します：

```
public void doReportDesignerApplet(Applet applet) throws Exception {

    //Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    //Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    //Create a new QbReportDesigner instance
    designer = new QbReportDesigner(applet,
            "RDWithTemplateFile.rpt");

    //Overwrites default saveFile method
    designer.setReportIO(this);

    //Start Designer
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        designer.setVisible(true);

}

//Save the file to a temp directory
public void saveFile(byte[] data, String fileLocation) {

        System.out.println("OLD LOCATION: " + fileLocation);
        String newLoc = fileLocation.replace('/', '¥¥');
        int idx = newLoc.lastIndexOf("¥¥");
        newLoc = "temp/" + newLoc.substring(idx+1);
        try {
                designer.writeFile(newLoc, data);
                System.out.println("NEW LOCATION: " + newLoc);

        } catch (Exception ex) {
                ex.printStackTrace();

        }

        designer.getDesigner().setTitle(newLoc);
        designer.getDesigner().repaint();

}
```

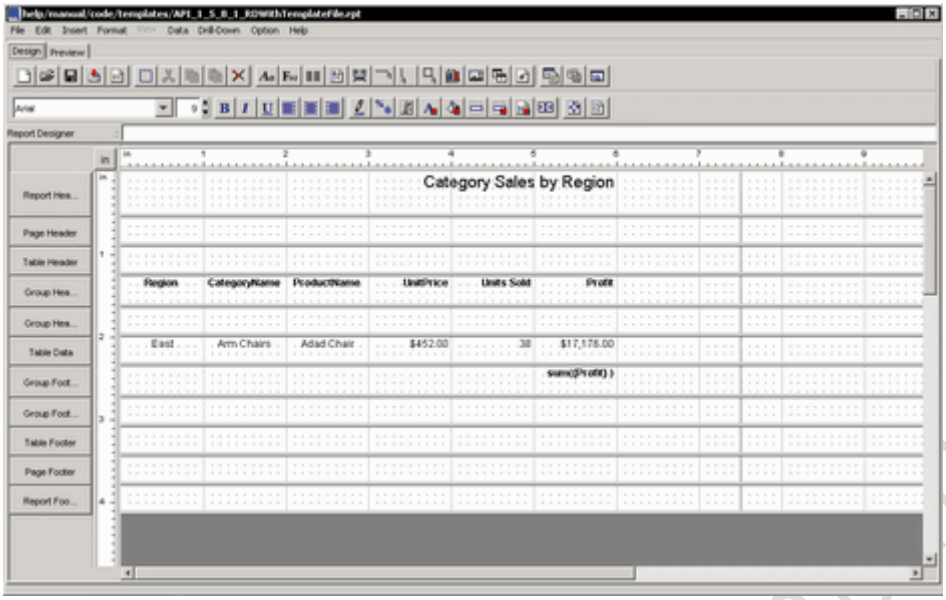
フルソースコード
エクスポートされた結果

上記のコードは、アプリケーションとアプレットの両方として使用できます。上記のコードでは.rpt ファイルが常に temp ディレクトリに保存されるように、Report Designer の保存機能も変更されています。

ユーザーがこのコードを実行すると、指定したレポートテンプレートで Report Designer が起動します。

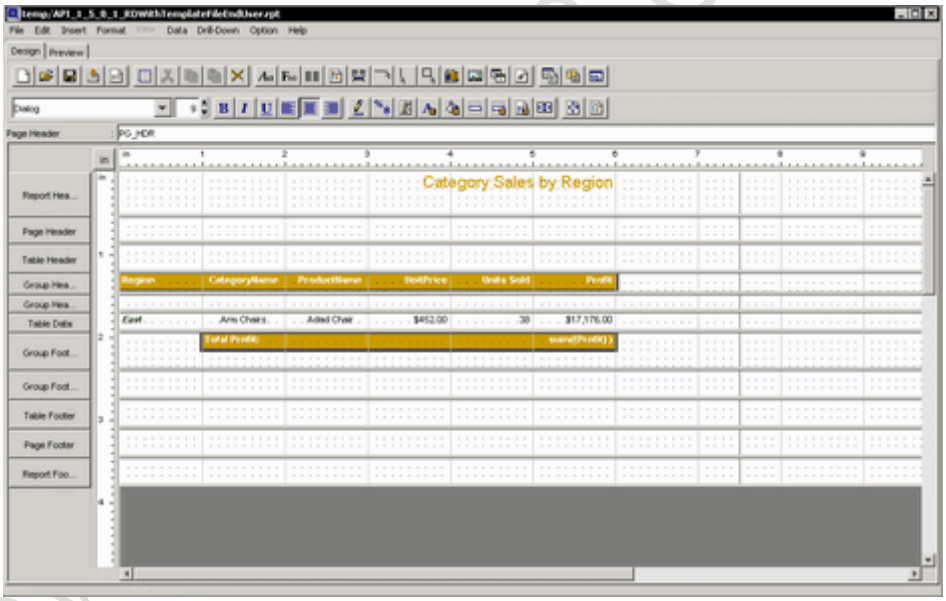
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



テンプレートを使用した Report Designer

ユーザーはレポートをカスタマイズして結果を保存できます。



エンドユーザーのカスタマイズ

また、上記のコードを実行する際に使用するレジストリを指定することもできます。これは、デザイナを表示する前に次のコード行を追加することによって実現されます。

```
designer.setDataRegistry("DataRegistry/Data_Registry.xml");
```

上記の行では、新しいレポートのデータソースを選択するときに Data Registry として **Data_Registry.xml** を使用するように Report Designer を指定しています。

データレジストリを指定する

Report Designer を開き、Data Source Manager（データレジストリに指定された.xml ファイルを使用）

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

から開始することができます。これにより、エンドユーザーは Report Designer の GUI でデータソースを選択し、独自のカスタムレポートを作成し、完成したレポートを表示できます。

次のコンストラクタが使用されます：

```
QbReportDesigner(Object parent, String nameOfXMLFile, boolean
doNotStartWithOpenNewReportDialogWizard);
```

指定された.xml ファイルを使用して Report Designer を呼び出す例を以下に示します：

```
public void doReportDesignerWithDataRegApplet(Applet applet) throws Exception {

    //Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    //Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    //QbReportDesigner (component, name of XML file, boolean);
    //true = start from Data Source Manager false = start from Create a new report
    //or open existing report choice before going to Data Source Manager
    designer = new QbReportDesigner(applet, "DataRegistry/Sample.xml", true);

    designer.setVisible(true);

}
```

フルソースコード

上記のコードは、アプリケーションとアプレットの両方として使用できます。

使用可能なオプションを有効または無効にすることによって、ユーザーが Data Registry で行う操作を制御することもできます。たとえば、Data Registry のデータソースの完全または一部を削除し、データソースの追加、コピー、編集、および削除のオプションを無効にすることができます。ノードは非表示ではなく、削除されています。したがって、Data Registry が（制限なしで）開かれると、削除されたノードの内容は失われます。Data Registry .xml ファイルのバックアップを作成し、そのバックアップを使用してオプションを有効または無効にすることができます。

以下は、指定されたデータレジストリを持つ Report Designer を呼び出して、データベースデータソースのみを使用できるように設定し、クエリまたはデータビューを追加できないように設定する例です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
public void doReportDesignerWithDataRegOptionApplet(Applet applet) throws Exception {

    //Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    //Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    //QbReportDesigner (component, name of XML file, true = start from
    //Data Source Manager false = start from Create a new report or open
    //existing report choice before going to Data Source Manager
    designer = new QbReportDesigner(applet,
            "DataRegistry/Sample.xml", true);
    designer.addDataSourceManagerListener(this);
    designer.setVisible(true);

}

public JTree modifyDataSourceTree(JTree tree) {

    //The following code will remove all nodes from the tree except for
    //the database nodes.
    DefaultDataSourceNode root = (DefaultDataSourceNode)tree.getModel().getRoot();
    for (int i=root.getChildCount()-1; i>=1; i--)
    {
        //System.out.println("removing");
        root.remove(i);

    }

    //The following code will prevent the user from adding any query or dataview.
    DefaultDataSourceNode databaseHeading = (DefaultDataSourceNode)root.getChildAt(0);
    for (int i=0; i<databaseHeading.getChildCount(); i++)
    {
        ((DefaultDataSourceNode)databaseHeading.getChildAt(i)
                .getChildAt(0)).setAddEnabled(false);
        ((DefaultDataSourceNode)databaseHeading.getChildAt(i)
                .getChildAt(1)).setAddEnabled(false);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
    }

    return tree;

}
```

フルソースコード

上記のコードは、アプリケーションとアプレットの両方として使用できます。

DBInfo オブジェクトの指定（データベース情報）

Report Designer を開き、**Select Report Type Wizard** ウィンドウ（指定した DBInfo オブジェクトを使用）から開始することができます。これにより、エンドユーザーは Report Designer GUI で独自のカスタムレポートを作成し、完成したレポートを表示できます。

次のコンストラクタが使用されます：

```
QbReportDesigner(Object parent, DBInfo databaseInformation, boolean
doNotStartWithOpenNewReportDialogWizard);
```

以下は、指定された DBInfo オブジェクトを使用して Report Designer を呼び出す例です：

```
public QbReportDesigner designer;
String URL = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
String driver = "org.hsqldb.jdbcDriver";
String username = "sa";
String password = "";
String query = "SELECT * FROM Order_Details";

public void doReportDesignerWithDBInfoApplet(Applet applet) throws Exception {

    //Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    //Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    //QbReportDesigner (component, Database Information, true = start from Data Source
Manager
    //false = start from Create a new report or open existing report choice before
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
    //going to Data Source Manager
    DBInfo dbInfo = new DBInfo(URL, driver, username, password, query);
    designer = new QbReportDesigner(applet, dbInfo, true);

    designer.setVisible(true);

}
```

フルソースコード

上記のコードは、アプリケーションとアプレットの両方として使用できます。

指定した DBInfo オブジェクトを使用して Query Builder を開く（データベース情報）

Query Builder は、指定された DBInfo オブジェクトで開くことができます。これにより、エンドユーザーは独自のカスタム SQL クエリを作成して Query Builder GUI に保存できます。

以下は、指定された DBInfo オブジェクトを使用して Query Builder を呼び出す例です：

```
public void doQueryBuilderApplet(Applet applet, String sqlName) {

    //Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    //Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    queryMain = new QbQueryBuilder(applet, this);
    if (sqlName != null) {
        //sqlName .qry file name (without extension)
        //System.out.println("OPEN QUERY - " + sqlName);
        queryMain.openQuery(sqlName, false, null, null, driver, URL, username, password);

    } else {
        //System.out.println("NEW QUERY ");
        queryMain.newQuery("Setup Query", false, null, null, driver, URL, username, password);

    }
    frame = queryMain.getBuilder();
    frame.setVisible(true);
    try { queryMain.showTablesWindow();
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
    } catch (Exception ex) {};

}

public void back() {};

public void cancel() {};

public void next() {

    queryMain.getBuilder().setVisible(false);
    DBInfo dbInfo = new DBInfo(URL, driver, username, password, queryMain.getSQL());
    designer = new QbReportDesigner(applet, dbInfo, queryMain.getInSet(), true);
    designer.setVisible(true);

}
```

フルソースコード

上記のコードは、アプリケーションとアプレットの両方として使用できます。

また、DBInfo オブジェクトではなく、Java オブジェクトとしてデータベース接続（スキーマ）を渡すことで、Query Builder を開くこともできます。 以下は、スキーマ情報を渡して Query Builder を呼び出す例です：

```
public QBWithDBInfo2() {

    //Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    //Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    try {
        Class.forName(driver);
        conn = DriverManager.getConnection(URL, username, password);
        metaData = conn.getMetaData();

    } catch (Exception ex) { ex.printStackTrace(); }
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        queryMain = new QbQueryBuilder(null, this);
        queryMain.newQuery("Setup Query", this);
        queryMain.setVisible(true);

        try { queryMain.showTablesWindow();
        } catch (Exception ex) {};

}

public void back() {};
public void cancel() {};

public void next() {

        //System.out.println("EXIT");

}

public String getDatabaseProductName() throws Exception {

        return metaData.getDatabaseProductName();

}

public ResultSet getTables(String catalog, String schemPattern, String tableNamePattern,
String[] types) throws Exception {

        return metaData.getTables(catalog, schemPattern, tableNamePattern, types);

}

public String getNumericFunctions() throws Exception {

        return metaData.getNumericFunctions();

}

public String getStringFunctions() throws Exception {

        return metaData.getStringFunctions();
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
}

public String getTimeDateFunctions() throws Exception {

    return metaData.getTimeDateFunctions();

}

public String getSystemFunctions() throws Exception {

    return metaData.getSystemFunctions();

}

public ResultSet executeQuery(String sql) throws Exception {

    Statement stmt = conn.createStatement();
    return stmt.executeQuery(sql);

}
```

フルソースコード

上記のコードは、アプリケーションとアプレットの両方として使用できます。

特定のクラスファイルデータソースを持つ Report Designer を開く

指定したクラスファイルのデータソースを使用して Report Designer を開くことができます。これにより、エンドユーザーは Report Designer の GUI で独自のカスタムレポートを作成し、完成したレポートを表示できます。クラスファイルのデータソースは、パラメータ化されていなくてもパラメータ化されていてもかまいません。

次のコンストラクタが使用されます：

```
QbReportDesigner(Object parent, String classFile,
quadbase.common.paramquery.QueryInParamSet inset, boolean newReport, int displayRow,
String[] imagesPath);
```

指定されたクラスファイルのデータソースを使用して Report Designer を呼び出す例を以下に示します：

```
public void doReportDesignerWithQueryApplet(Applet applet) {
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
    //Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    //Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    //QbReportDesigner (component, class file location, parameter information,
    //true = start from Data Source Manager false = start from Create a new
    //report or open existing report choice before going to Data Source
    //Manager, number of rows to display (-1 means show all rows), path to
    //Report Designer images
    designer = new QbReportDesigner(applet, "ParamClassFile",
          null, true, -1, null); //Parameterized class file

    designer.setVisible(true);

    JFrame frame = designer.getDesigner();

}
```

フルソースコード

カスタムクラスファイルのデータソースの作成の詳細については、カスタム実装で渡されるデータを参照してください。

カスタム関数を含む Report Designer を開く

ユーザーが API から Report Designer を呼び出すときに使用できる Formula and Scripts ダイアログボックスには、独自の関数を含めることもできます。実際に複雑な計算を処理する関数を作成し、どのユーザーも同じ機能を Report Designer で使用できるようにすることができます。

以下は、カスタム関数を Formulas and Scripts ダイアログボックスに割り当てることによって、Report Designer をカスタマイズする方法を示すサンプルコードです。コードは数字を取り込み、**192.168.0** の最後に数字を付加して IP アドレスに変換します。

```
public void doRDWithCustomFunctions(Applet applet) throws Exception {
        QbReportDesigner.setServerAddress("127.0.0.1");
        QbReportDesigner.setServerPortNumber(22071);

        QbReportDesigner.setUseSysResourceImages(true);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        designer = new QbReportDesigner(applet, "RDWithCustomFunctions.rpt");
        designer.setCustomDefinedFunctions(this);
        designer.setVisible(true);
}

public String[] getAllFunctionNames() {
        return new String[] { "inet_ntoa" };
}

public int getReturnType(String functionName) {
        if (functionName.equals("inet_ntoa")) {
                return STRING;
        }
        return -1;
}

public int[] getParamTypes(String functionName) {
        if (functionName.equals("inet_ntoa")) {
                return new int[] { NUMERIC };
        }
        return null;
}

public Object getValue(String functionName, Object[] args) throws Exception {
        if (functionName.equals("inet_ntoa")) {
                return "192.168.0." + ((Double) args[0]).intValue();
        }
        return null;
}
```

フルソースコード

上記のコードは、アプリケーションとアプレットの両方として使用できます。

関数がコアセットの一部でないため、関数を含むセルは最初は Null として表示されます。ただし、データをプレビューするときに関数が正しく計算されます。その後、値は正しく表示されます。

事前設定されたディレクトリを持つ Report Designer を開く

また、レポートをロードするディレクトリと保存するディレクトリを設定することもできます。この機能により、ユーザーは指定されたディレクトリの上の任意のレベルにナビゲートすることができなくな

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

り、ユーザーが見ることができるファイルを制御することができます。

以下は、ロードフォルダと保存フォルダに異なるディレクトリを指定して、Report Designer をカスタマイズする方法を示す例です：

```
public void doCustomizeDesignerMenuApplet(Applet applet) throws Exception {
        QbReportDesigner.setServerAddress("127.0.0.1");
        QbReportDesigner.setServerPortNumber(22071);

        QbReportDesigner.setUseSysResourceImages(true);

        designer = new QbReportDesigner(applet, "RDWithModifiedBars.rpt");

        //      Add a new menu item
        JMenuBar menuBar = designer.getReportMenuBar();
        JMenu fileMenu = menuBar.getMenu(0);
        newItem = new JMenuItem("Hello World");
        newItem.addActionListener(this);
        fileMenu.insert(newItem, 7);

        //      Remove toolbar buttons
        JToolBar designBar = designer.getDesignerToolBar();
        designBar.remove(5); //Remove Separator
        designBar.remove(4); //Remove Apply Template Button
        designer.setSaveOnExitEnabled(false); //Do not prompt to save the report if unsaved on exiting Designer

        designer.setVisible(true);

}

public void actionPerformed(ActionEvent e) {

        /*** save report **********/
        designer.save("RDWithModifiedBars_Temp.rpt");
        /**** create new testing frame *********/
        JPanel contentPane = new JPanel();
        contentPane.setLayout(new BorderLayout());
        contentPane.add(new JLabel("Hello World!"), "Center");
        JFrame frame = new JFrame();
        frame.setContentPane(contentPane);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        frame.pack();
        frame.setVisible(true);

}
```

フルソースコード

上記のコードは、アプリケーションとアプレットの両方として使用できます。また、上記の例では、Save As を選択し、.rpt ファイルを保存するディレクトリを指定できます。ただし、.rpt ファイルは常に temp ディレクトリに保存されます。

上記の方法に加えて、次の方法を使用して、Report Designer で使用される既定のディレクトリを取得することもできます：

```
public BrowseDirectories getBrowseDirectories();
```

次に、BrowseDirectories のメソッドを使用して、異なるブラウズダイアログのディレクトリのデフォルトの場所を取得できます。

修正されたメニューバーとツールバーを含む開いた Report Designer

メニューに項目を追加して、ツールバーから項目を削除することもできます。 以下にその例を示します：

```
public void doCustomizeDesignerMenuApplet(Applet applet) {

    //Connect to EspressManager
    QbReportDesigner.setServerAddress("127.0.0.1");
    QbReportDesigner.setServerPortNumber(22071);

    //Use images from jar file
    QbReportDesigner.setUseSysResourceImages(true);

    designer = new QbReportDesigner(applet,
"help/manual/code/templates/RDWithModifiedBars.rpt");

    //Add a new menu item
    JMenuBar menuBar = designer.getReportMenuBar();
    JMenu fileMenu = menuBar.getMenu(0);
    newItem = new JMenuItem("Hello World");
    newItem.addActionListener(this);
    fileMenu.insert(newItem, 7);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        //Remove toolbar buttons
        JToolBar designBar = designer.getDesignerToolBar();
        designBar.remove(5); //Remove Separator
        designBar.remove(4); //Remove Apply Template Button
        //Do not prompt to save the report if unsaved on exiting Designer
        designer.setSaveOnExitEnabled(false);

        designer.setVisible(true);

}

public void actionPerformed(ActionEvent e) {

        /*** save report **********/
        designer.save("RDWithModifiedBars_Temp.rpt");

        /**** create new testing frame *********/
        JPanel contentPane = new JPanel();
        contentPane.setLayout(new BorderLayout());
        contentPane.add(new JLabel("Hello World!"), "Center");
        JFrame frame = new JFrame();
        frame.setContentPane(contentPane);
        frame.pack();
        frame.setVisible(true);
}
```

フルソースコード

上記のコードは、アプリケーションとアプレットの両方として使用できます。

ウィザードをスキップして Report Designer を開く

いくつかのウィザードのステップをスキップして、Report Designer を開くことができます。 次の関数を呼び出して、**query result**、**multiple data source**、および **pre-defined templates** ステップをスキップするだけです：

```
qbReportDesigner.setSkipQueryResultStep(true);
qbReportDesigner.setSkipMultiDataSourceStep(true);
qbReportDesigner.setSkipPredefinedTemplatesStep(true);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

カスタマイズされたメッセージとチャートデザイナを含む Report Designer を開く

サブレポートとドリルダウンレポートの間を移動するときに、カスタマイズされた警告で Report Designer を開くことができます。また、Report Designer の既定の名前を付けて保存の動作を変更することもできます。次の例は、これらのさまざまなコンポーネントを変更する方法を示しています。この例は、読みやすくするために断片に分割されています。完全なソースコードはこのセクションの最後にあります。

```
public class RDWithCustomMessage {

    public static void main(String[] args) {
        try {
            startDesigner(); } catch (Exception ex) {
            ex.printStackTrace(); } }

    static void startDesigner() {
        //Connect to EspressManager
        QbReportDesigner.setServerAddress("127.0.0.1");
        QbReportDesigner.setServerPortNumber(22071);

        //Use images from jar file
        QbReportDesigner.setUseSysResourceImages(true);

        QbReportDesigner designer = new QbReportDesigner((Frame)null);
        //modify warning message before adding drill-down layer
        designer.modifyWarningMessage(QbReportDesigner.SAVE_RPT_BEFORE_DRILLDOWN,
              "The report designer will save the current report for you, continue?");

        //modify warning message before inserting Sub-Report
        designer.modifyWarningMessage(QbReportDesigner.SAVE_RPT_BEFORE_SUBREPORT,
             "The report designer will save the current report for you, continue?");

        //by pass save as dialog when submitting save location
        designer.setReportIO(new ReportIO(designer));

        //by pass save as dialog when user try to navigate to next level
        designer.setByPassSaveAsIO(new ByPassSaveAsForReport());

        //modify chart designer from report
        designer.setChartDesignerHandle(new ChartDesignerHandle());

        //REMOVE "SAVE AS" option for Report Designer
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        JMenu fileMenu = designer.getReportMenuBar().getMenu(0);
        fileMenu.remove(6);
        designer.setVisible(true); }

}
```

上記のコードの部分は、サブレポートまたはドリルダウンレベルを追加するときに表示される警告メッセージを変更します。また、**SaveResortIO**、**setByPassSaveAsIO**、**setChartDesignerHandle** を呼び出すには、Save As ダイアログを渡します（詳細は後述）。また、コードは、Report Designer のメニューバーから Save As オプションを削除します。

```
public static class ReportIO implements IReportIO {

    String fileName = null;
    QbReportDesigner designer = null;

    public ReportIO(QbReportDesigner designer) { this.designer = designer; }

    public void saveFile(byte[] data, String fileName) {
        System.out.println("SAVE REPORT FILE - " + fileName);
        try {
            FileOutputStream fout = new FileOutputStream(fileName);
            fout.write(data, 0, data.length); } catch (Exception ex) {
            ex.printStackTrace(); }
        this.fileName = fileName; }

}
```

上記のフラグメントは、**IReportIO** インタフェースを実装しています。前述のコードフラグメントで **ReportIO** を設定することにより、Report Designer は、ユーザーがレポートを保存しようとすると、ここで **saveFile** メソッドを呼び出します。ファイルはこのメソッドにバイト配列として渡され、ファイル名は文字列として渡されます。このメソッドは、ファイル名を使用して出力ストリームを作成し、レポートバイト配列をそのストリームに書き込みます。

```
public static class ByPassSaveAsForReport implements IByPassSaveAsForReport {

    public ByPassSaveAsForReport() {};

    public String getFileName(String originalFileName) {
        System.out.println("BY PASS REPORT SAVE AS OPTION...");
        if (originalFileName == null) return "TEMP_REPORT_FILE.rpt";
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        else return originalFileName; }

    public Properties getSaveAsProperties(String originalFileName) {
        return new Properties(); }

}
```

Save As オプションを渡すと、前のコードフラグメントの **saveFile** メソッドは、上のクラスの getFileName メソッドからファイル名を取得します。ここでは、ファイルの名前をハードコーディングしてプロセスを単純化しています。通常、ファイル名を操作して、ユーザーがお互いのファイルを上書きしないようにする必要があります。**getSaveAsProperties** メソッドは、保存オプションを含む Properties インスタンスを返します。可能なオプションには、**CREATE_STL**、**SAVE_ALL_DATA**、**CREATE_HTML**、**USE_SWINGVIEWER**、**CREATE_XML**、**CREATE_PACK**、および **USE_PAGEVIEWER** があります。ここでは、何も使用しないことを選択します。

```
public static class ChartDesignerHandle implements IChartHandle {

    public ChartDesignerHandle() {};

    public void processDesigner(QbChartDesigner designer) {
        System.out.println("PROCESS CHART....");
        //REMOVE "SAVE AS" option for Chart Designer
        JMenu fileMenu = designer.getChartMenuBar().getMenu(0);
        fileMenu.remove(6);
        designer.setChartIO(new ChartIO());
        designer.setByPassSaveAsIO(new ByPassSaveAsForChart()); }

}

public static class ChartIO implements IChartIO {

    public ChartIO() {};

    public String saveChartFile(byte[] data, String fileName) {
        System.out.println("SAVE CHART FILE - " + fileName);
        try {
            FileOutputStream fout = new FileOutputStream(fileName);
            fout.write(data, 0, data.length);
            fout.close(); } catch (Exception ex) {
            ex.printStackTrace(); }
        return fileName; }
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
}

public static class ByPassSaveAsForChart implements IByPassSaveAsForChart {

    public ByPassSaveAsForChart() {};

    public String getFileName(String originalFileName) {
        System.out.println("BY PASS CHART SAVE AS OPTION...");
        if (originalFileName == null) return "TEMP_CHART_FILE.cht";
        else return originalFileName; }

    public Properties getSaveAsProperties(String originalFileName) {
        return new Properties(); }

}
```

フルソースコード

上記の 3 つのクラスは、Chart Designer 側と同じ変更を行います。ChartDesignerHandler クラスは、メニューバーから名前を付けて保存オプションを削除します。**ChartIO** クラスはファイル名とバイト配列を指定してファイルを保存します。また、**ByPassSaveAsForChart** クラスは、グラフの固定ファイル名を設定します。

4.3.5.7.12 Report/Page Viewer のオプションの変更

Report Viewer または Page Viewer を使用してレポートを表示しているときに、ユーザーが実行できる操作と実行できない操作を構成することがあります。

ユーザーが Viewer を使用してレポートを表示しているときに、レポートを右クリックするとメニューがポップアップ表示されます。このメニューを使用すると、レポートのさまざまなページをナビゲートすることができます。また、DHTML または PDF ファイルとしてエクスポートするなど、レポート上の他のタスクを実行することもできます。これは、レポートビューアと Page Viewer API を使用して簡単に行うことができます。ただし、レポートビューアのこれらの強力な機能は、平均的なユーザーにとっては複雑すぎる可能性があります。したがって、いくつかの API メソッドが、Report/Page Viewer のポップアップメニューで使用できるオプションを制御するために導入されました。

これらの API コールは次のとおりです：

```
viewer.setMenuVisible(boolean b);
viewer.setPageMenuVisible(boolean b);
viewer.setPageMenuItemVisible(String[] items, boolean b);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
viewer.setOutputMenuVisible(boolean b);
viewer.setOutputMenuItemVisible(String[] items, boolean b);
viewer.setRefreshMenuItemVisible(boolean b);
viewer.setGoToMenuItemVisible(boolean b);
viewer.setSortMenuVisible(boolean b);
```

これらの API メソッドの詳細については、APIDocs を参照してください

4.3.5.8 Javadoc

API 全体の Javadoc は EspressReport と共に提供されています。API は、Report API と Charting API の両方をカバーします。Quadbase の Web サイトにあります。

4.3.5.9 スイングバージョン

1.1 Report API の JFC/Swing バージョンも利用できます。詳細については、quadbase.reportdesigner.ReportViewer.swing を参照してください。

4.3.5.10 概要

EspressReport API は、ビジネスアプリケーション用の使いやすいと同時に強力なアプリケーションプログラミングインタフェースを提供します。Report Designer と組み合わせることで、アプレットに 1 行のコードを追加するだけで簡単にプログラミングできます。レポートのすべての属性は、Report Designer で作成できるテンプレートファイルで設定できます。EspressReport API は、Windows 95、Windows NT/2000/XP、Solaris、Linux、および Linux 上の Netscape の Communicator（4.06 以上）、Microsoft の Internet Explorer（4.x 以上）、Sun の Appletviewer（1.2 以上） AIX、および HP プラットフォームをサポートします。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.4 スケジューラ

4.4.1 前書き

EspressReport には、レポートを特定の時間または特定の間隔でエクスポートできる Scheduler インタフェースがあります。これにより、EspressReport は、特定の時間にユーザーがログインして明示的に操作を実行するよりも、レポートのエクスポートなどの日常的なタスクを処理することができます。スケジューラの使用方法の詳細については、スケジューリングを参照してください。

スケジューラ API コードを書くとき、2 つの主要なクラスが使用されます。
スケジュールジョブの詳細を示すために使用される **ScheduleObject** と、
Scheduler を変更するために使用される **ScheduleModifier**。

スケジューラを使用するコードの場合は、**ReportAPIWithChart.jar** と **Scheduler.jar**（EspressReport/lib ディレクトリ内）の両方を **CLASSPATH** に追加する必要があります。

マニュアルに記載されているすべての例とコードは、EspressManager がローカル（つまりローカルマシン）で実行されており、デフォルトのポート番号（22071）を使用していることを前提としています。これを変更するには、ソースコード（対応する章の「フルソースコード」リンクをクリックしてソースコードをダウンロードできます）を開き、コードを編集して新しい接続情報を入力し、コードを再コンパイルします。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.4.2 EspressManager に接続する

最初に EspressManager に接続せずに Scheduler（または Scheduler のコード）を実行することはできません。EspressManager がアプリケーションとして実行されているかサーブレットとして実行されているかに応じて、API メソッドを使用して接続情報を指定できます。

4.4.2.1 アプリケーションとして実行する EspressManager

EspressManager がアプリケーションとして実行されている場合は、API メソッドを使用して、EspressManager が配置されている IP アドレス/マシン名と EspressManager がリッスンしているポート番号を指定できます。

接続情報を設定するには、次の 2 つの API メソッドを使用します。

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

たとえば、次のコード行があります。

```
QbScheduler.setServerAddress("someMachine");
QbScheduler.setServerPortNumber(somePortNumber);
```

someMachine で動作し、**somePortNumber** でリッスンしている EspressManager に接続します。

EspressManager の接続情報が指定されていない場合、コードはローカルマシンの EspressManager に接続し、デフォルトのポート番号（22071）を聞き取ります。

これらのメソッドは、**QbScheduler** クラスと **ScheduleModifier** クラスに存在します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.4.2.2 Servlet として動作する EspressManager

EspressManager がサーブレットとして実行されている場合は、次の方法を使用できます。

```
public static void useServlet(boolean b);
public static void setServletRunner(String comm_URL);
public static void setServletContext(String context);
```

たとえば、次のコード行があります。

```
QbScheduler.useServlet(true);
QbScheduler.setServletRunner("http://someMachine:somePortNumber");
QbScheduler.setServletContext("EspressReport/servlet");
```

http://someMachine:somePortNumber/EspressReport/servlet で動作する EspressManager に接続します。

これらのメソッドは、QbScheduler クラスと ScheduleModifier クラスに存在します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.4.3 スケジューラグラフィカルインタフェースの呼び出し

スケジューラは、アプリケーションまたはアプレットの Report API から呼び出すことができます。 アプレットまたはアプリケーションとして実行できる次の例は、**QbScheduler** オブジェクトを作成して Scheduler を呼び出します。

```
QbScheduler qbScheduler = new QbScheduler((Applet)null, frame);
((JPanel)frame.getContentPane()).add("Center", qbScheduler.getScheduler());
frame.pack();
frame.setVisible(true);
```

フルソースコード

プログラマが特定の機能をオンまたはオフにしてスケジューラインタフェースを開くように設定する場合は、いくつかの **Set()**メソッドも使用できます。
ブラウズのためにルートディレクトリを設定し、挿入コマンド機能 **insert command** をオフに設定するために使用できる 2 つの方法を次に示します。

```
qbScheduler.setRootDirectoryForBrowse(String root);
qbScheduler.setInsertCommandEnabled(boolean state);
```

Copyright(C) 2018 Climb.Inc

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.4.4 エクスポートのスケジューリング

スケジューラを使用してエクスポートするレポートをスケジュールできます。
以下のコードは、これを行う方法を示しています：

```
ScheduleObject sObj = new ScheduleObject("Report_OneTime", ScheduleObject.REPORTOBJ);
        sObj.setFileLocation("help/manual/code/templates/Account.rpt");
        sObj.setReportType(IExportConstants.DHTML);
        sObj.setTaskOption(ScheduleObject.ONE_TIME);
        String exportLocation = sObj.pickDefaultExportLocation();
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.MINUTE, 1);
        sObj.setStartDate(calendar.getTimeInMillis());
        sObj.setSendEmail(false);

        ScheduleModifier.addScheduleTask(sObj);
```

フルソースコード

相対 URL 参照（**help/manual/code/templates/Account.rpt** など）を使用するデータソースやレポートファイルへの参照は、EspressManager が実行されているディレクトリからの相対パスであることに注意してください。

上記のコードは、デフォルトのエクスポートディレクトリの DHTML にエクスポートされるレポート（Account.rpt）をスケジュールし、1 分と 1 分の間にエクスポートを設定します。

以下の例は、定期的なスケジュールを設定する方法を示しています。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
sObj.setFileLocation("Account.rpt");
        sObj.setReportType(IExportConstants.DHTML);
        String exportLoc = sObj.pickDefaultExportLocation();

        sObj.setTaskOption(ScheduleObject.TIME_INTERVAL);
        /*** every 5 mins ****/
        sObj.setIntervalType(ScheduleObject.TIME);
        sObj.setTimeInterval(5); //export every 5 mins

        /*** every day ****/
        //                  sObj.setIntervalType(ScheduleObject.DAYS);
        //                  sObj.setDayInterval(1); //export everyday

        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.MINUTE, 5);
        sObj.setStartDate(calendar.getTimeInMillis());
        Calendar calendar2 = Calendar.getInstance();
        calendar2.add(Calendar.MINUTE, 26);
        sObj.setEndDate(calendar2.getTimeInMillis());
        sObj.setSendEmail(false);

        ScheduleModifier.addScheduleTask(sObj);
```

フルソースコード

上記のコードスニペットは、ジョブを5分ごとに実行するようスケジュールする方法を示しています(またはコメントするコードのセクションによっては毎日)。

次のコードスニペットは、特定の期間にジョブを実行するようにスケジュール設定する方法を示しています。

Copyright(C)

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
ScheduleObject sObj = new ScheduleObject("Report_Time", ScheduleObject.REPORTOBJ);
        sObj.setFileLocation("Account.rpt");
        String exportLoc = sObj.pickDefaultExportLocation();
        sObj.setReportType(IExportConstants.PDF);
        sObj.setTaskOption(ScheduleObject.FIXED_DAYS);
        /*** export every 2 hrs. monday through friday ****/
        sObj.setSpecifyDays(new int[] { 1, 2, 3, 4, 5 });
        sObj.setStartTime(9 * 60); //9:00AM in minutes
        sObj.setEndTime(22 * 60); //10:00PM in minutes
        sObj.setTimeInterval(2 * 60); //export every 2 hours
        /*** export at specific times on specific dates ***/
        //sObj.setSpecifyDates(new int[]{26,27,28,29}); //export on specific dates
        //sObj.setSpecifyTime(new int[]{16 * 60, 17 * 60, 18 * 60}); //4PM, 5PM and 6PM in
minute
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.MINUTE, 5);
        sObj.setStartDate(calendar.getTimeInMillis());
        sObj.setEndDate(-1); //run indefinitely
        sObj.setSendEmail(false);

        ScheduleModifier.addScheduleTask(sObj);
```

フルソースコード

パラメータ化されたチャートまたはレポートをエクスポートするようにスケジューリングするときに、パラメータを指定することもできます。次のコードは、その方法を示しています。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
ScheduleObject sObj = new ScheduleObject("ParamReport_OneTime",
ScheduleObject.REPORTOBJ);
        //                  Single Param Report
        sObj.setFileLocation("EmployeeDetails.pak");
        String exportLoc = sObj.pickDefaultExportLocation();
        sObj.setReportType(IExportConstants.DHTML);
        sObj.setTaskOption(ScheduleObject.ONE_TIME);
        //                  1st param set {"Denise Carron"}
        Object tmp1[] = { "Denise Carron"};
        //                  2st param set {"Frank Carnody"}
        Object tmp2[] = { "Frank Carnody"};
        Vector paramList = new Vector();
        paramList.addElement(tmp1);
        paramList.addElement(tmp2);
        sObj.setParamList(paramList);
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.MINUTE, 5);
        sObj.setStartDate(calendar.getTimeInMillis());
        sObj.setSendEmail(false);

        ScheduleModifier.addScheduleTask(sObj);
```

フルソースコード

レポートのセキュリティレベルも指定できます。次のコードは、セキュリティレベル（レポートセルまたはクエリパラメータ）を指定する方法を示しています。

```
ScheduleObject sObj = new ScheduleObject("Report_Security", ScheduleObject.REPORTOBJ);
        sObj.setFileLocation("OrderFormByRegion.rpt");
        sObj.setReportType(IExportConstants.DHTML);
        sObj.setTaskOption(ScheduleObject.ONE_TIME);
        sObj.setSecurityLevel("ParamSet");
        String exportLocation = sObj.pickDefaultExportLocation();
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.MINUTE, 10);
        sObj.setStartDate(calendar.getTimeInMillis());
        sObj.setSendEmail(false);

        ScheduleModifier.addScheduleTask(sObj);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フルソースコード

リンクまたは添付ファイルとして、1 つまたは複数の電子メールアドレスに送信するスケジュールを指定することもできます。次のコードは、パラメータ化されていないレポートを含むスケジュールされたジョブを複数の受信者に送信する方法を示しています。

```
ScheduleObject sObj = new ScheduleObject("Report_Email", ScheduleObject.REPORTOBJ);
        sObj.setFileLocation("Account.rpt");
        sObj.setReportType(IExportConstants.DHTML);
        sObj.setTaskOption(ScheduleObject.ONE_TIME);
        String exportLocation = sObj.pickDefaultExportLocation();
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.MINUTE, 10);
        sObj.setStartDate(calendar.getTimeInMillis());
        sObj.setSendEmail(true);
        sObj.setFromAddress("user1@quadbase.com");
        sObj.setSubject("Report Email Account Report");
        sObj.setBodyText("Scheduled Export of Account from user1");
        sObj.setEmailType(ScheduleObject.ASATTACHMENT);
        sObj.setToAddresses(new String[] { "user2@quadbase.com", "user3@quadbase.com"});

        ScheduleModifier.addScheduleTask(sObj);
```

フルソースコード

同様に、エクスポートをスケジュールするときに、パラメータ化されたレポートまたはチャート用に選択された各パラメータセットに対して電子メールを送信できます。少なくとも 1 つのメールアドレスを各パラメータセットに指定する必要があり、同じ受信者を複数のパラメータセットに使用できることに注意してください。

Copyright(C)

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
ScheduleObject sObj = new ScheduleObject("ParamReport_Email",
ScheduleObject.REPORTOBJ);
        //                  Single Param Report
        sObj.setFileLocation("EmployeeDetails.pak");
        String exportLoc = sObj.pickDefaultExportLocation();
        sObj.setReportType(IExportConstants.DHTML);
        sObj.setTaskOption(ScheduleObject.ONE_TIME);
        //                  1st param set {"Denise Carron"}
        Object tmp1[] = { "Denise Carron"};
        //                  2nd param set {"Frank Carnody"}
        Object tmp2[] = { "Frank Carnody"};
        Vector paramList = new Vector();
        paramList.addElement(tmp1);
        paramList.addElement(tmp2);
        sObj.setParamList(paramList);
        Vector paramNameList = new Vector();
        paramNameList.addElement("Denise");
        paramNameList.addElement("Frank");
        sObj.setParamList(paramNameList, paramList);
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.MINUTE, 5);
        sObj.setStartDate(calendar.getTimeInMillis());
        sObj.setSendEmail(true);
        sObj.setFromAddress("user1@quadbase.com");
        sObj.setSubject("Param Report Email Account Report");
        sObj.setBodyText("Scheduled Export of Employee Details from user1");
        sObj.setEmailType(ScheduleObject.ASATTACHMENT);
        Hashtable toAddr = new Hashtable();
        toAddr.put("Denise", new String[] { "user2@quadbase.com"});
        toAddr.put("Frank", new String[] { "user3@quadbase.com"});
        sObj.setParamAddresses(toAddr);

        ScheduleModifier.addScheduleTask(sObj);
```

フルソースコード

スケジュールされたレポートは、Web アクセス可能な場所に保存することも、URL をさまざまな受信者に電子メールで送信することもできます。次のコードは、方法を示しています

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
ScheduleObject sObj = new ScheduleObject("Report_Email", ScheduleObject.REPORTOBJ);
        sObj.setFileLocation("Account.rpt");
        sObj.setReportType(IExportConstants.DHTML);
        sObj.setTaskOption(ScheduleObject.ONE_TIME);
        String exportLocation = sObj.pickDefaultExportLocation();
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.MINUTE, 10);
        sObj.setStartDate(calendar.getTimeInMillis());
        sObj.setSendEmail(true);
        sObj.setFromAddress("user1@quadbase.com");
        sObj.setSubject("Report Email Account Report");
        sObj.setBodyText("Scheduled Export of Account from user1");
        sObj.setEmailType(ScheduleObject.ASLINK);
        sObj.setURLMapping("http://someMachine:somePort/ScheduledReports/");
        sObj.setToAddresses(new String[] { "user2@quadbase.com",
"user3@quadbase.com" });

        ScheduleModifier.addScheduleTask(sObj);
```

フルソースコード

Copyright(C) 2018

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.4.5 コマンドのスケジューリング

エクスポートのスケジューリングに加えて、特定の時間（または時間間隔）で実行されるコマンドをスケジュールすることもできます。コードは、レポートの場所、エクスポートの形式、およびエクスポートの場所ではなく、実行するコマンドが指定されていることを除いて、エクスポートのスケジュール設定のコードと同じままです。

```
ScheduleObject sObj = new ScheduleObject("Command_OneTime", ScheduleObject.COMMANDOBJ);
        //Note that someClass does not exist
        sObj.setCommand("java someClass");
        sObj.setTaskOption(ScheduleObject.ONE_TIME);
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.MINUTE, 5);
        sObj.setStartDate(calendar.getTimeInMillis());
        sObj.setSendEmail(false);

        ScheduleModifier.addScheduleTask(sObj);
```

フルソースコード

4.4.6 スケジュールの削除

スケジュールの追加に加えて、API を使用してスケジュールを削除することもできます。これは、スケジュールされたすべてのジョブのリストを取得し、特定のジョブを削除することによって実行されます。

次のコードは、特定のジョブを削除する方法を示しています。

```
Vector schedList = ScheduleModifier.getScheduler().getScheduleList();

 ScheduleObject obj = (ScheduleObject)schedList.elementAt(0);
 ScheduleModifier.removeScheduleTask(obj);
```

フルソースコード

上記のコードはスケジュールリストの最初のジョブを削除することに注意してください。すでに削除マークが付けられていると、ジョブが表示されないことがあります。推奨されるアプローチは、ベクトルを通り、各 **ScheduleObject** を名前で検索し、正しいものを削除することです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.4.7 失敗したスケジュールの詳細の取得

場合によっては、スケジュールされたジョブがなんらかの理由で失敗することがあります。EspressManager は、すべての失敗したジョブを追跡し、後でその詳細を取得できるようにします。次のコードは、詳細を取得する方法を示しています。

```
Vector vec = ScheduleLog.getFailedScheduledJob();
  for (int i = 0; i < vec.size(); i++) {

      FailedScheduledJob obj = (FailedScheduledJob) vec.elementAt(i);
      System.out.println("FAIL # "+ i + ": Name = "+ obj.getScheduledJobName());
      System.out.println("File Location = "+ obj.getFileLocation());
      System.out.println("Time = "+ obj.getExportTime());
      System.out.println("StackTrace = "+ obj.getStackTrace());

  }
```

フルソースコード

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

また、スケジュールに失敗した場合に備えて電子メールを送信するように設定することもできます。
次のコードは、設定の方法を示しています。

```
ScheduleObject sObj = new ScheduleObject("Report_FailedEmail", ScheduleObject.REPORTOBJ);
        sObj.setFileLocation("Account.rpt");
        sObj.setReportType(IExportConstants.DHTML);
        sObj.setTaskOption(ScheduleObject.ONE_TIME);
        String exportLocation = sObj.pickDefaultExportLocation();
        Calendar calendar = Calendar.getInstance();
        calendar.add(Calendar.MINUTE, 10);
        sObj.setStartDate(calendar.getTimeInMillis());
        sObj.setSendEmail(true);
        sObj.setFromAddress("user1@quadbase.com");
        sObj.setSubject("Report Email Account Report");
        sObj.setBodyText("Scheduled Export of Account from user1");
        sObj.setEmailType(ScheduleObject.ASATTACHMENT);
        sObj.setToAddresses(new String[] { "user2@quadbase.com", "user3@quadbase.com"});
        sObj.setFailSubject("Report_FailedEmail schedule failed");
        sObj.setFailBodyText("Cannot send Account report from Report_FailedEmail");
        sObj.setFailToAddress("user1@quadbase.com");

        ScheduleModifier.addScheduleTask(sObj);
```

フルソースコード

Copyright(C)

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.4.8 ICallBackScheduler インタフェース

スケジューラがジョブを実行することに基づいて、特定のアクションを実行する追加コードを作成することもできます（成功したかどうかにかかわらず）。このコードは **ICallBackScheduler** インタフェースを実装し、ジョブが正常に完了したかどうかを指定します。

コードを作成した後、**EspressManager.bat(.sh)**（<EspressReport InstallDir>内）を変更して、以下を追加する必要があります。

```
-SchedulerCallBackClass:<name of class file implementing ICallBackScheduler>
```

次のコードは、**ICallBackScheduler** インタフェースの使用方法を示しています。

```
import quadbase.scheduler.ICallBackScheduler;
    import quadbase.scheduler.ScheduleObject;

    public class CallBackScheduler implements ICallBackScheduler {

        public CallBackScheduler() {};

        public void exportSucceeded(ScheduleObject obj, String path) {

            System.out.println(obj.getName() + "- EXPORT SUCCEEDED");
            System.out.println("PATH = "+ path);
        }

        public void exportFailed(ScheduleObject obj, String path, Throwable e) {

            System.out.println(obj.getName() + "- EXPORT FAILED");
            System.out.println("PATH = "+ path);
            System.out.println("ERROR = "+ e.toString());
        }

    }
```

上記のコードを使用するには、**EspressManager.bat/.sh** に以下を追加します。

```
-SchedulerCallbackClass:CallBackScheduler
```

CallBackScheduler が **CLASSPATH** にあることを確認してください。
コードは、各ジョブの最後に、ジョブが正常に実行されたかどうかを示すメッセージを出力します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.4.9 スケジューラリスナー

EspressReport を使用すると、サーバエクステンションを使用してカスタムコードでスケジュールレポートを変更できます。これらのカスタムクラスは、レポートをエクスポートする前にレポートをインターセプトし、ユーザーがスケジューリングプロセスに追加のビジネスロジックを実装できるようにします。

スケジューラーリスナーを使用するには、**SchedulerListener** を実装する独自のカスタムクラスを作成する必要があります。以下はその例です。

```
import quadbase.reportdesigner.ReportAPI.QbReport;
import quadbase.scheduler.ScheduleObject;
import quadbase.ext.SchedulerListener;

public class MyEspressReportSchedulerListener implements SchedulerListener {
   public QbReport modifyBeforeExport(QbReport report, ScheduleObject so, String exportPath) {
      System.out.println("modifyBeforeExport(" + report + "," + so + "," + exportPath + ")");
      return report;
   }
}
```

上の例では、レポートをエクスポートする前に単純な **System.out.println** 文を出力します。

SchedulerListener を実装するカスタムクラスを使用するには、DefaultListenerManager を実装する別のカスタムクラスを実装する必要があります。
以下はその例です。:

Copyright(C) Climb Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
import quadbase.ext.DefaultListenerManager;
  import quadbase.ext.SchedulerListener;

  public class MyEspressReportListenerManager extends DefaultListenerManager {

      public MyEspressReportListenerManager() {}

      public EspressReportSchedulerListener getSchedulerListener() {

          return new MyEspressReportSchedulerListener();

      }

  }
```

上記のコードを使用するには、**EspressManager.bat/.sh** に以下を追加します。

```
-ListenerManagerClass:MyEspressReportListenerManager
```

MyEspressReportListenerManager と **MyEspressReportSchedulerListener** の両方が **CLASSPATH** にあることを確認してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.4.10 レポートのバースト

また、スケジューラのレポートバースト機能は、API を使用して使用することもできます。
レポートではデータベースをデータソースとして使用する必要があり、レポートをバーストするにはグループ化する必要があることに注意してください。

以下に、レポートバーストの使用方法の例を示します。

```
ScheduleObject sObj = new ScheduleObject("ReportBursting_All",
ScheduleObject.REPORTOBJ);
sObj.setFileLocation("InventoryInformation.rpt");
String exportLoc = sObj.pickDefaultExportLocation();
sObj.setReportType(IExportConstants.DHTML);
sObj.setTaskOption(ScheduleObject.ONE_TIME);
Calendar calendar = Calendar.getInstance();
calendar.add(Calendar.MINUTE, 5);
sObj.setStartDate(calendar.getTimeInMillis());
sObj.setSendEmail(true);
sObj.setFromAddress("user1@quadbase.com");
sObj.setSubject("Test report bursting");
sObj.setBodyText("You should find a report of one group");
sObj.setEmailType(ScheduleObject.ASATTACHMENT);
//export every group as a single report and email using report column values
sObj.setBurstReport(ScheduleObject.ALLBURSTING);
//get email address from 16th column in template
sObj.setEmailColumnIndex(16);

ScheduleModifier.addScheduleTask(sObj);
```

フルソースコード

上記のコードはすべてのグループを破棄し、レポートの列をアドレスのソースとして使用して電子メールを送信します。

上記のコードでは、列 16 はテンプレートに存在しません。
この例では、破棄のために電子メール列を渡す方法を示します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.4.11 Javadoc

API 全体の Javadoc は EspressReport と共に提供されています。
詳細は Quadbase の Web サイトをご覧ください。

4.4.12 概要

EspressReport API は、スケジュールを追加したり削除したりするだけでなく、スケジューラーインターフェースを照会する使いやすく強力な API を提供します。スケジューラでジョブが正常に完了した（または終了していない）場合、独自のアクションを実行するためのコードを記述することもできます。

API には、JDK 1.5 以上が必要です。EspressReport API は、Windows、Solaris、Linux、AIX、および HP プラットフォームでテストされています。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.5 サーブレットと Java Server Pages

4.5.1 サーブレット

4.5.1.1 前書き

SunMicrosystems®Web サイト（**http://java.sun.com**）で説明されているように、Java サーブレットは Web 開発者に、Web サーバの機能を拡張し、既存のビジネスシステムにアクセスするためのシンプルで一貫したメカニズムを提供します。サーブレットは、サーバ側で実行されるアプレットとして、顔なしでほとんど考えることができます。Java サーブレットは、多くの Web アプリケーションを可能にしました。

今日、サーブレットはインタラクティブな Web アプリケーションを構築するための一般的な選択肢です。サーブレットコンテナは、Apache Web Server、iPlanet Web Server（以前の Netscape Enterprise Server）、Microsoft IIS などで使用できます。サーブレットコンテナは、BEA WebLogic Application Server、IBM WebSphere、Netscape Application Server などの Web 対応アプリケーションサーバと統合することもできます。

以下のセクションでは、EspressReport（**EspressReport/help/examples/servlet/StreamingReport** ディレクトリにある）で提供されるサーブレットの例を設定する方法について説明します。各セクションでは、サーブレットの例を具体的に扱っています。ただし、他のサーブレットの例に加えて、独自のサーブレットを設定して実行するためのガイドとして使用することもできます。

このサーブレットに加えて、他のサーブレットの例も提供されています。レポートの簡単な表示（HTML または DHTML 形式のいずれか）から、グラフを含むレポートをブラウザに直接ストリーミングすることにまで及ぶ。これらの例のセットアップと実行は、以下のセクションを参考にして行うことができます。

4.5.1.2 セットアップ

- 次のファイルがあることを確認します。
  - Summary_1.rpt
  - Summary_2.rpt
  - CrossTab.rp

  EspressReport/help/examples/templates ディレクトリにあります。
- EspressManager が実行されていることを確認します。
- 次に、使用しているサーブレットサーバ（ランナー）に応じて、以下のガイドラインに従います。ブラウザから **index.html**（**EspressReport/help/examples/servlet/StreamingReport** ディレクトリにあります）を実行します。

命令は Windows プラットフォームを参照していますが、Unix/Linux プラットフォームでも使用できま

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

す。Unix/Linux 標準に準拠するようにファイル名とパスを変更することは、唯一必要なステップです。

すべての例では、簡単にするためにシングルスレッドモデルを使用しています。ただし、EspressReport API はマルチスレッド環境でも使用できます。

以下は、**EspressReport/help/examples/servlet** にある StreamingReport サーブレットの例を使用しています。

4.5.1.3 ランニングアンダー

4.5.1.3.1 Apache Tomcat 5.5.20 または 6.0.10

1. バージョン 5.5.20 または 6.0.10 の Tomcat サーバがインストールされていることを確認してください。ただし、このガイドはおそらく以前のバージョンの Tomcat サーバにも当てはまります。

   ApacheTomcat サーバの Windows オペレーティングシステム用（各バージョン用）のインストールファイルバージョンは 2 つあります：

   zip file
   -**startup.bat** を使用する
   exe file
   -Windows サービスをインストールし、"Apache Tomcat Monitor"アプリケーションを使用する

2. クラスパス-**ReportDesigner.jar** ファイルと **servlet-api.jar** ファイルをクラスパスに含める必要があります。

   ReportDesigner.jar
   ファイルは **EspressReport/lib** ディレクトリにあります
   servlet-api.jar
   Tomcat のインストールディレクトリ（zip インストールファイルを使用する場合は、**apache-tomcat-5.5.20¥common¥lib¥**ディレクトリにあります。exe インストールファイルを使用する場合、**C:¥Program Files¥Apache Software Foundation¥Tomcat 5.5¥common¥lib¥**ディレクトリをデフォルトの Tomcat インストールディレクトリにします）

   次に、コマンドラインコンソールを使用したクラスパス設定の例を示します：

```
set classpath=%classpath%;C:¥Program Files¥Apache Software Foundation¥Tomcat 5.5¥common¥lib¥servlet-api.jar;C:¥EspressReport¥lib¥ReportDesigner.jar
```

（Tomcat 5.5 zip インストールファイルを使用している場合）

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
set classpath=%classpath%;C:¥apache-tomcat-5.5.20¥common¥lib¥servlet-api.jar;C:¥EspressReport¥lib¥ReportDesigner.jar
```

（Tomcat 5.5 インストーラ-exe インストールファイルを使用している場合）

Tomcat 6.0 バージョンの場合は、以下を使用してください：

```
set classpath=%classpath%;C:¥Program Files¥Apache Software Foundation¥Tomcat 6.0¥lib¥servlet-api.jar;C:¥EspressReport¥lib¥ReportDesigner.jar
```

（Tomcat 6.0 zip インストールファイルを使用している場合）

```
set classpath=%classpath%;C:¥apache-tomcat-6.0.10¥lib¥servlet-api.jar;C:¥EspressReport¥lib¥ReportDesigner.jar
```

（Tomcat 6.0 インストーラ-exe インストールファイルを使用している場合）

クラスパスは必ずしも設定する必要はありませんが、**.java** ファイルのコンパイルは簡単です（クラスパスを追加する必要はありません）。

MyComputer > Administrate > Details タブを使用してクラスパス変数を設定することもできます。

3. サンプルファイルのコンパイル（**streamingReportServlet.java**）

コンパイルコマンド：

```
javac -classpath "C:¥Program Files¥Apache Software Foundation¥Tomcat 5.5¥common¥lib¥servlet-api.jar;C:¥EspressReport¥lib¥ReportDesigner.jar" streamingReportServlet.java
```

（Tomcat 5.5 用）

```
javac -classpath "C:¥Program Files¥Apache Software Foundation¥Tomcat 6.0¥lib¥servlet-api.jar;C:¥EspressReport¥lib¥ReportDesigner.jar" streamingReportServlet.java
```

（Tomcat 6.0 用）

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

（CLASSPATH 環境変数がこのガイドの手順 2 で設定されていない場合）

または単に：

```
javac streamingReportServlet.java
```

（classpath が適切に設定されている場合）

4. Tomcat の **webapps¥ROOT¥WEB-INF¥classes** ディレクトリにコピーする。

   ReportServlet.class ファイルを Tomcat の **webapps¥ROOT¥WEB-INF¥classes** ディレクトリにコピーします。

   > **webapps¥ROOT¥WEB-INF¥classes** ディレクトリはデフォルトでは存在する必要はなく、作成する必要があります。

5. lib ディレクトリにサーブレットを実行するために必要なライブラリを追加する。

   この例を実行するには、Apache Tomcat の **webapps¥ROOT¥WEB-INF¥lib** ディレクトリにライブラリを追加する必要もあります。**EspressReport¥lib** ディレクトリにある **ReportDesigner.jar** ファイルを **webapps¥ROOT¥WEB-INF¥lib** ディレクトリにコピーしてください。

   > **webapps¥ROOT¥WEB-INF¥lib** ディレクトリはデフォルトでは存在する必要はなく、作成する必要があります。

6. Tomcat サーバの **web.xml** ファイルにサーブレットアプリケーションを登録する。

   サーブレットアプリケーションを実行する前に、Tomcat サーバの **webapps¥ROOT¥WEB-INF¥**ディレクトリにある **web.xml** ファイルを変更する必要があります。

   streamingReportServlet を登録するには、次のコードを追加してください：

```
<servlet>
  <servlet-name>streamingReportServlet</servlet-name>
  <servlet-class>streamingReportServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>streamingReportServlet</servlet-name>
  <URL-pattern>/servlet/streamingReportServlet</URL-pattern>
</servlet-mapping>
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

<**web-app**>タグと</**web-app**>xml タグの間に挿入します。

> バージョン 5.5.20 で Tomcat を使用する場合、webapp ディレクトリに配置されたすべてのサーブレットの**自動起動**を有効にすることもできます。この方法で、サーブレットアプリケーションを登録する必要はありません。
>
> これは、テストのみを目的としています。プロダクション環境でインボーカサーブレットを使用することは推奨されず、サポートされていません。

7. Tomcat サーバを起動（再起動）する

zip インストールファイルを使用した場合は、Tomcat インストールディレクトリの bin ディレクトリから **startup.bat**（Linux の場合は **startup.sh**）ファイルを実行してください。

exe インストールファイルの場合は、Tomcat サービスを起動する必要があります。この目的のために、Tomcat の"Monitor Tomcat"アプリケーションを使用することができます。

> Tomcat サーバが既に稼働している場合は、例を実行可能にするために、再起動（シャットダウンと再起動）が必要な場合があります。

8. サンプルの実行

最後のステップは、この例を実行することです。この例を実行するには、ブラウザの **EspressReport¥help¥examples¥servlet¥StreamingReport** ディレクトリにある **index.html** ファイルを実行してください。

GetReport ボタンをクリックすると、**streamingReportServlet** が呼び出され、適切なレポートが生成されます。

> この例を実行する前に EspressManager が実行されていることを確認してください。 実行されていない場合、空のレポートが生成されます。

> サーブレットランナーとクライアントブラウザの両方が同じマシン上にある場合にのみ、設計されているサーブレットが機能することに注意してください。クライアントを別のマシンに置くには、**Servlet.html** ファイル（**EspressReport¥help¥examples¥servlet¥StreamingReport** ディレクトリに置かれています）の行を変更します。
>
> from:

```
<form action="http://127.0.0.1:8080/servlet/streamingReportServlet" method="post">
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

to:

```
<form action="http://<machine_name>:8080/servlet/streamingReportServlet" method="post">
```

ウェブブラウザで自動設定ディスカバリを使用すると、ネットワーク設定に応じて **127.0.0.1** マシンに到達する問題が発生する可能性があります。

代わりに **localhost** を使用してください。

```
<form action="http://localhost:8080/servlet/streamingReportServlet" method="post">
```

4.5.1.3.2 JRun 3.1

- **Servlet.html** ファイル内のマシン名とポート番号を次の場所に置き換えます。

```
<FORM ACTION=http://<machine name>:8100/servlet/streamingReportServlet method=POST>
```

- JRun Default Server Administrator にログインします。
- **Java Settings** をクリックし、**Classpath** をクリックします。
- **ReportAPIWithChart.jar** へのパスを追加し、**Update** をクリックします。
- JRun Default Server を再起動します。
- サーブレットをコンパイルし、**<jrun_InstallDir>/servers/default/default-app/WEB-INF/classes** にコピーします。

4.5.1.3.3 ColdFusion Server 4.5

- すべての html ファイルを拡張子.cfm に置き換えます（たとえば、**Servlet.html** は **Servlet.cfm** になります）。cfm ファイルのマシン名とポート番号を次のように置き換えます。

```
<FORM ACTION=http://<machine name>:8100/servlet/streamingReportServlet method=POST>
```

- JRun Default Server Administrator にログインします。
- **Java Settings** をクリックし、**Classpath** をクリックします。
- **ReportAPIWithChart.jar** へのパスを追加し、**Update** をクリックします。
- JRun Default Server を再起動します。
- サーブレットをコンパイルし、**<jrun InstallDir>/servers/default/default-app/WEB-INF/classes** にコピーします。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.5.1.3.4 WebLogic 6.0

サーブレットの例を設定して実行するには、まず **wlserver6.0¥config¥examples** ディレクトリに移動します。次の手順に従って、**setExamplesEnv.cmd** ファイルと **startExamplesServer.cmd** ファイルの両方を変更します：

- **@rem Set user-defined variables** セクションに、**ES_REPORT=<EspressReport InstallDir>**を追加します。**ES_REPORT** 変数には、マシン内の EspressReport ホームディレクトリへのパスが含まれています。マシンのパスに対応するパス値を変更します。また、**WL_HOME** 変数と **JAVA_HOME** 変数がコンピュータ上の正しいパスに対応していることを確認してください。
- **%ES_REPORT%¥lib¥ReportAPIWithChart.jar;**を追加します。同じファイルの **CLASSPATH** フィールドに設定します。
- **wlserver6.0¥config¥examples¥applications¥examplesWebApp** ディレクトリに html ファイルを置きます。
- コード内のマシン名を **127.0.0.1:8080** から **Servlet.html** ファイルの**<machineName>：7001** に変更します。
- **streamingReportServlet.java** を **wlserver6.0¥samples¥examples¥servlets** ディレクトリに配置します。
- **wlserver6.0¥config¥examples¥applications¥examplesWebApp¥WEB-INF¥**ディレクトリにある **web.xml** ファイルに次のコードフラグメントを挿入します。**<servlet>**コードは、ファイル内の**<servlet>**コードセクションと**<servlet-mapping>**セクションにあります。

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>
    <init-param>

        <param-name>reportType</param-name>
        <param-value>Summary</param-value>
        <param-name>rowBreaks</param-name>
        <param-value>ProductType</param-value>
        <param-name>browser</param-name>
        <param-value>Netscape</param-value>

    </init-param>

</servlet>
<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/streamingReportServlet/*</URL-pattern>
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
</servlet-mapping>
```

- コマンドプロンプトウィンドウで、**wlserver6.0¥config¥examples** ディレクトリに移動し、**setExamplesEnv** を実行します。
- **wlserver6.0¥samples¥examples¥servlets** ディレクトリに移動し、下記のコマンドラインを使用して **streamingReportServlet.java** を個別にコンパイルします。

```
javac –d %EX_WEBAPP_CLASSES% streamingReportServlet.java
```

- 同じコマンドプロンプトウィンドウで **wlserver6.0¥config¥examples** ディレクトリに戻り、**startExamplesServer** と入力して Enter キーを押して WebLogic Server を起動します。
- Web ブラウザを開き、**http://yourMachineName:7001/examplesWebApp/index.html** に移動して、サーブレットの例を表示します。

WebLogic 9.2

次の手順では、WebLogic 9.2 でサーブレットのサンプルを設定して実行する方法を示します。手順では、WebLogic 9.2 サーバがシステムにインストールされていることを前提としています WebLogic インストールの場所は**<WL_INSTALL_DIR>**として参照され、EspressReport インストールの場所は**<ER_INSTALL_DIR>**として参照されます。

サーブレットのサンプルをセットアップして実行するには、まず **<WL_INSTALL_DIR>/samples/domains/wl_server/bin** ディレクトリに移動します。次の手順に従って **setExamplesEnv.cmd** ファイルを変更します。

- **set ES_REPORT=<ER_INSTALL_DIR>**を追加します。**ES_REPORT** 変数には、マシン内の EspressReport ホームディレクトリへのパスが含まれています。マシンのパスに対応するパス値を変更します。また、**WL_HOME** 変数と **JAVA_HOME** 変数がコンピュータ上の正しいパスに対応していることを確認してください。
- **%ES_REPORT%¥lib¥ReportAPIWithChart.jar;%ES_REPORT%¥lib¥qblicense.jar;**を追加します。同じファイルの **CLASSPATH** フィールドに設定します。

次に、以下の手順に従います
（ファイルは**<ER_INSTALL_DIR>¥help¥examples¥servlet¥StreamingReport** ディレクトリにあります）。

- html ファイル（**columnar.html、servlet.html、index.html**）を **<WL_INSTALL_DIR>¥samples¥server¥examples¥build¥examplesWebApp** ディレクトリに配置します。
- **servlet.html** ファイルを編集し、コード内のマシン名を **127.0.0.1:8080/servlet** から **yourMachineName:7001/examplesWebApp** に変更します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- **streamingReportServlet.java** ファイルを **<WL_INSTALL_DIR>¥samples¥server¥examples¥build¥examplesWebApp¥WEB-INF¥classes** ディレクトリに配置します。
- **<WL_INSTALL_DIR>¥samples¥server¥examples¥build¥examplesWebApp¥WEB-INF** ディレクトリにある **web.xml** ファイルに次のコードフラグメントを挿入します。**<servlet>**コードはファイルの **<servlet>** コードセクションにあり、**<servlet-mapping>** セクションの **<servlet-mapping>**に入ります。

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/streamingReportServlet/*</URL-pattern>

</servlet-mapping>
```

- 次に、**<WL_INSTALL_DIR>¥samples¥server¥examples¥build¥examplesWebApp¥WEB-INF¥classes** ディレクトリに移動し、**streamingReportServlet.java** をコンパイルします。クラスパスに **includeReportAPIWithChart.jar** と **javax.servlet.jar** を入れてください。
  （**javax.servlet.jar** は **bea ¥jrockit90_150_06¥mercuryprofiler¥lib** ディレクトリにあります）。
- コマンドプロンプトウィンドウで、**<WL_INSTALL_DIR>/samples/domains/wl_server** ディレクトリに戻り、**startWebLogic.cmd** と入力して Enter キーを押して WebLogic Server を起動します。
- Web ブラウザを開き、**http://yourMachineName:7001/examplesWebApp/index.html** に移動して、サーブレットの例を表示します。

トラブルシューティングのために、入力ミスをチェックしてください。

4.5.1.3.5 WebSphere 3.5

- WebSphere Administrator's Console を開き、ノード（通常はマシン名）＞Default Servlet Engine ＞ Default Host ＞ Default App を選択し、**Advanced** タブをクリックして、**ReportAPIWithChart.jar** をクラスパスに追加します。次に、**Apply** をクリックします。
- **Servlet.html** の **FORM ACTION** タグを次のように変更します：

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
<FORM ACTION=http://<machine name>/servlet/streamingReportServlet method=POST>
```

- **streamingReportServlet.java** をコンパイルし、クラスファイルを **WebSphere/AppServer/servlets** ディレクトリに移動します。
- ノードの下にあるデフォルトサーバを開始/再起動します。

WebSphere 6.1

次の手順では、WebSphere 6.1 でサーブレットのサンプルを設定して実行する方法を示します。この手順では、WebSphere 6.1 サーバがシステムにインストールされていることを前提としています。WebSphere インストールの場所は**<WS_INSTALL_DIR>**として参照され、EspressReport インストールの場所は**<ER_INSTALL_DIR>**として参照されます。

サーブレットの例を設定して実行するには、まず**<ER_INSTALL_DIR>¥lib¥ReportAPIWithChart.jar** および**<ER_INSTALL_DIR>¥lib¥qblicense.jar** ファイルを**<WS_INSTALL_DIR>¥AppServer¥lib** ディレクトリにコピーします。

次に、以下の手順に従います
（ファイルは**<ER_INSTALL_DIR>¥help¥examples¥servlet¥StreamingReport** ディレクトリにあります）。

- **streamingReportServlet.java** ファイルを **<WS_INSTALL_DIR>¥AppServer¥profiles¥AppSrv01¥installedApps¥<YourMachineName>Node01Cell¥ivtApp.ear¥ivt_app.war¥WEB-INF¥classes** ディレクトリに配置します。
- **servlet.html** ファイルを編集し、コード内のマシン名を **127.0.0.1:8080/servlet** から **yourMachineName:9080/ivt/servlet** に変更します。
- html ファイル（**columnar.html、servlet.html、index.html**）を **<WS_INSTALL_DIR>¥AppServer¥profiles¥AppSrv01¥installedApps¥<YourMachineName>Node01Cell¥ivtApp.ear¥ivt_app.war** ディレクトリに入れます。
- **<WS_INSTALL_DIR>¥AppServer¥profiles¥AppSrv01¥installedApps¥<YourMachineName>Node01Cell¥ivtApp.ear¥ivt_app.war¥WEB-INF** ディレクトリにある **web.xml** ファイルに、次のコードフラグメントを挿入します。**<servlet>**コードはファイルの**<servlet>**コードセクションにあり、**<servlet-mapping>**セクションの**<servlet-mapping>**に入ります。

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        <servlet-name>streamingReportServlet</servlet-name>
        <URL-pattern>streamingReportServlet</URL-pattern>

    </servlet-mapping>
```

- 次に、**<WS_INSTALL_DIR>¥AppServer¥profiles¥AppSrv01¥installedApps¥<YourMachineName>Node01Cell¥DefaultApplication.ear¥DefaultWebApplication.war¥WEB-INF¥classes** ディレクトリに移動し、**streamingReportServlet.java** をコンパイルします。**ReportAPIWithChart.jar** と **j2ee.jar** をクラスパスに含めてください（j2ee.jar は**<WS_INSTALL_DIR>¥AppServer¥lib** ディレクトリにあります)。
- WebSphere サーバを開始します。これを行うにはいくつかの方法があります。Windows ユーザーにとって最も簡単な方法は、Start > Programs > IBMWebSphere > Application Server v6.1 > Profiles > AppSrv01 > First Steps の順に選択して **First Steps** ツールを起動することです。First Steps ツールから管理コンソールを起動することもできます。
- Web ブラウザを開き、**http://yourMachineName:9080/ivt/index.html** に移動して、サーブレットの例を表示します。

トラブルシューティングの際は、入力ミスをチェックしてください。

4.5.1.3.6 WebSphere Express Server 5.0

- File > New > Web Project を選択します。
- **Project Name** テキストフィールドにプロジェクト名を入力します。この例では、**EspressReport** を使用します。
- **Finish** ボタンをクリックします。
- **J2EE Navigator** パネルの **EspressReport** プロジェクトフォルダに移動し、プロジェクトフォルダを展開します。
- **Java Source** サブフォルダを右クリックし、**Import** オプションを選択します。**File System** タイプのインポートを選択します。
- サーブレットと java bean 用の java ソースファイルをここでインポートします。この例では、**EspressReport/help/examples/servlet/StreamingReport** ディレクトリから **streamingReportServlet.java** をインポートします。
- **EspressReport** プロジェクトフォルダに戻ります。
- **Web Content** のサブフォルダに移動します。JSP ファイルと HTML ファイルをここにインポートします。この例では、**EspressReport/help/examples/servlet/StreamingReport** ディレクトリから **File System** タイプのインポートとインポート **index.html**、**Columnar.html** と **Servlet.html** ファイルを選択します。
- **WEB-INF/lib** サブフォルダに移動し、**Import** オプションを右クリックし、**EspressReport/lib** ディレクトリから **ExportLib.jar** および **ReportAPIWithChart.jar** ファイルをインポートします。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- **Servlet.html** ファイルをダブルクリックして開き、**Source** タブを押して編集します。
- **Servlet.html** の **FORM ACTION** タグを次のように変更します。

```
<FORM ACTION=http://<machine name>:7080
/EspressReport/servlet/streamingReportServlet method=POST>
```

7080 は HTTP ポート番号です。**7080** がポート番号でない場合は、適切な値に置き換えてください。

- この HTML ファイルを保存します。
- **EspressReport** ウェブプロジェクトフォルダを右クリックし、**Rebuild Project** を選択します。
- **EspressReport** ウェブプロジェクトフォルダを右クリックして、**Run on Server...**を選択します。
- サーバを選択し、**Finish** ボタンをクリックします。すでに構成されているサーバを選択することをお勧めします。
- 結果は Web ブラウザパネルに表示されます。

4.5.1.3.7 JBoss 4.0.5

次の手順は、JBoss 4.0.5 の下でサーブレットのサンプルを設定して実行する方法を示しています。この手順では、JBoss 4.0.5 サーバがシステムにインストールされていることを前提としています。JBoss 4.0.5 インストールの場所は**<JB_INSTALL_DIR>**と参照され、EspressReport インストールの場所は**<ER_INSTALL_DIR>**として参照されます。

サーブレットの例を設定して実行するには、まず**<JB_INSTALL_DIR>¥bin** ディレクトリに移動します。次の手順に従って **run.bat** ファイルを変更します。

- セットを追加する **ES_REPORT** = **<ER_INSTALL_DIR>**。
  ES_REPORT 変数には、マシン内の **EspressReport** ホームディレクトリへのパスが含まれています。

- **%ES_REPORT%¥ lib¥ReportAPIWithChart.jar;%ES_REPORT%¥ lib¥qblicense.jar;**を追加します。同じファイルの **JBOSS_CLASSPATH** フィールドに設定します。

次に、以下の手順に従います（ファイルは <ER_INSTALL_DIR>¥help¥examples¥servlet¥StreamingReport ディレクトリにあります）。

- **streamingReportServlet.java** ファイルを **<JB_INSTALL_DIR>¥server¥default¥deploy¥jmx-console.war¥WEB-INF¥classes** ディレクトリに配置します.y

- **servlet.html** ファイルを編集し、コード内のマシン名を **servlet.html** ファイルの **127.0.0.1:8080/servlet** から **yourMachineName:8080/jmx-console** に変更します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- html ファイル（**columnar.html**、**servlet.html** および **index.html**）を **<JB_INSTALL_DIR>¥server¥default¥deploy¥jmx-console.war** ディレクトリに配置します。

- **<JB_INSTALL_DIR>¥server¥default¥deploy¥jmx-console.war¥WEB-INF** ディレクトリにある **web.xml** ファイルに次のコード・フラグメントを挿入します。**<servlet>**コードはファイルの **<servlet>**コードセクションにあり、**<servlet-mapping>**セクションの**<servlet-mapping>**に入ります。

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/streamingReportServlet</URL-pattern>

</servlet-mapping>
```

- 次に、**<JB_INSTALL_DIR>¥server¥default¥deploy¥jmx-console.war¥WEB-INF¥classes** ディレクトリに移動し、**ExportServlet2.java** をコンパイルします。**ReportAPIWithChart.jar** と **javax.servlet.jar** をクラスパスに含めてください
（注：**javax.servlet.jar** は**<JB_INSTALL_DIR>¥server¥all¥lib** ディレクトリにあります)。

- コマンドプロンプトウィンドウで、**<JB_INSTALL_DIR>¥bin** ディレクトリに移動し、**run.bat** と入力して **Enter** キーを押して JBoss サーバを起動します。

- Web ブラウザを開き、**http://yourMachineName:8080/jmx-console/index.html** に移動して、サーブレットの例を表示します。

トラブルシューティングのために、入力ミスをチェックしてください。

4.5.1.3.8 Orion 2.0.7

次の手順は、Orion 2.0.7 の下でサーブレットのサンプルを設定して実行する方法を示しています。この手順では、Orion 2.0.7 サーバがシステムにインストールされていることを前提としています。Orion インストールの場所は**<OR_INSTALL_DIR>**と参照され、EspressReport インストールの場所は**<ER_INSTALL_DIR>**として参照されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次の手順に従います（ファイルは<**ER_INSTALL_DIR**>**¥help¥examples¥servlet¥StreamingReport** ディレクトリにあります)。

- **streamingReportServlet.java** ファイルを <**OR_INSTALL_DIR**>**¥default-web-app¥WEB-INF¥classes** ディレクトリに配置します。

- **servlet.html** ファイルを編集し、コード内の URL の部分を **127.0.0.1:8080/servlet** から **yourMachineName:80** に変更します。

- **index.html** ファイルの名前を **index2.html** に変更し、html ファイル(**columnar.html**、**servlet.html** および **index2.html**）を<**OR_INSTALL_DIR**>**¥default-web-app** ディレクトリに配置します。

- <**OR_INSTALL_DIR**>**¥default-web-app¥WEB-INF** ディレクトリにある **web.xml** ファイルに次のコードフラグメントを挿入します。<**servlet**>コードはファイルの<**servlet**>コードセクションにあり、<**servlet-mapping**>セクションの<**servlet-mapping**>に入ります。

```
<servlet>

        <servlet-name>streamingReportServlet</servlet-name>
        <servlet-class>streamingReportServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>streamingReportServlet</servlet-name>
        <URL-pattern>streamingReportServlet</URL-pattern>

    </servlet-mapping>
```

- 次に、<**OR_INSTALL_DIR**>**default-web-app¥WEB-INF¥classes** ディレクトリに移動し、**streamingReportServlet.java** をコンパイルします。**ReportAPIWithChart.jar** と **j2ee.jar** をクラスパスに含めてください。
  （<**J2EE_SDK_INSTALL_DIR**>**¥lib** ディレクトリにある **J2eeSDK** インストールから **j2ee.jar** を使用できます)。

- **ReportAPIWithChart.jar** および **qblicense.jar** ファイルを<**OR_INSTALL_DIR**>**¥bin** ディレクトリにコピーします。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- コマンドプロンプトウィンドウで、**<OR_INSTALL_DIR>**ディレクトリに移動し、**java -jar orion.jar** と入力して Enter キーを押して、Orion サーバを開始します。

- Web ブラウザを開き、**http://yourMachineName:80/index2.html** に移動して、サーブレットの例を表示します。

トラブルシューティングのために、入力ミスをチェックしてください。

4.5.1.3.9 JRun 4（Update 6 付き）

次の手順は、JRun 4（Update 6 を使用）でサーブレットのサンプルを設定して実行する方法を示しています。手順では、システムに JRun サーバがインストールされていることを前提としています。JRun インストールの場所は**<JR_INSTALL_DIR>**と参照され、EspressReport インストールの場所は**<ER_INSTALL_DIR>**として参照されます。

次の手順に従います
（ファイルは**<ER_INSTALL_DIR>¥help¥examples¥servlet¥StreamingReport** ディレクトリにあります）。

- **servlet.html** ファイルを編集し、コード内の URL の一部を **127.0.0.1:8080** から **yourMachineName:8100** に変更します。

- html ファイル（**columnar.html**、**servlet.html**、**index.html**）を **<JR_INSTALL_DIR>¥servers¥default¥default-ear¥default-war** ディレクトリに配置します。

- **streamingReportServlet.java** ファイルを **<JR_INSTALL_DIR>¥servers¥default¥default-ear¥default-war¥WEB-INF¥classes** ディレクトリに配置します。

- **<JR_INSTALL_DIR>¥servers¥default¥default-ear¥default-war¥WEB-INF** ディレクトリにある **web.xml** ファイルに次のコードフラグメントを挿入します。**<servlet>**コードはファイルの**<servlet>**コードセクションにあり、**<servlet-mapping>**セクションの**<servlet-mapping>**に入ります。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/servlet/streamingReportServlet</URL-pattern>

</servlet-mapping>
```

- 次に、**<JR_INSTALL_DIR>¥servers¥default¥default-ear¥default-war¥WEB-INF¥classes** ディレクトリに移動し、**streamingReportServlet.java** をコンパイルします。**ReportAPIWithChart.jar** と **j2ee.jar** をクラスパスに含めてください。
  （**<J2EE_SDK_INSTALL_DIR>¥lib** ディレクトリにある J2eeSDK インストールから **j2ee.jar** を使用できます）。

- **ReportAPIWithChart.jar** および **qblicense.jar** ファイルを**<JR_INSTALL_DIR>¥servers¥lib** ディレクトリにコピーします。

- **<JR_INSTALL_DIR>¥bin¥jrun.exe** を実行して、JRun 4 アプリケーションサーバを起動します。これにより、**JRun Launcher** ウィンドウが起動します。JRun Launcher から、デフォルトサーバを起動します。

- Web ブラウザを開き、**http://yourMachineName:8100/index.html** に移動して、サーブレットの例を表示します。

トラブルシューティングのために、入力ミスをチェックしてください。

4.5.1.3.10 Oracle 10g（10.1.3.1.0）

次の手順では、Oracle 10g（10.1.3.1.0）でサーブレットのサンプルを設定して実行する方法を示します。この手順では、Oracle 10g（10.1.3.1.0）サーバがシステムにインストールされていることを前提としています。Oracle 10g インストールの場所は**<ORA_INSTALL_DIR>**として参照され、EspressReport インストールの場所は**<ER_INSTALL_DIR>**として参照されます。

次の手順に従います
（ファイルは**<ER_INSTALL_DIR>¥help¥examples¥servlet¥StreamingReport** ディレクトリにあり

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ます)。

- **servlet.html** ファイルを編集し、コードの URL 部分を **127.0.0.1:8080** から **yourMachineName:8888** に変更します。

- **index.html** ファイルの名前を **index2.html** に変更し、html ファイル(**columnar.html**、**servlet.html** および **index2.html**) を<**ORA_INSTALL_DIR**>¥**j2ee**¥**home**¥**default-web-app** ディレクトリに配置します。

- **streamingReportServlet.java** ファイルを<**ORA_INSTALL_DIR**>¥**j2ee**¥**home**¥**default-web-app**¥**WEB-INF**¥**classes** ディレクトリに配置します。

- <**ORA_INSTALL_DIR**>¥**j2ee**¥**home**¥**default-web-app**¥**WEB-INF** ディレクトリにある **web.xml** ファイルに次のコードフラグメントを挿入します。<**servlet**>コードはファイルの<**servlet**>コードセクションにあり、<**servlet-mapping**>セクションの<**servlet-mapping**>に入ります。

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/servlet/streamingReportServlet</URL-pattern>

</servlet-mapping>
```

- 次に、<**ORA_INSTALL_DIR**>¥**j2ee**¥**home**¥**default-web-app**¥**WEB-INF**¥**classes** ディレクトリに移動し、**streamingReportServlet.java** をコンパイルします。**ReportAPIWithChart.jar** と **servlet.jar** をクラスパスに含めてください。
  (注:**servlet.jar** は<**ORA_INSTALL_DIR**>¥**j2ee**¥**home**¥**lib** ディレクトリにあります)。

- **ReportAPIWithChart.jar** および **qblicense.jar** ファイルを <**ORA_INSTALL_DIR**>¥**j2ee**¥**home**¥**applib** ディレクトリにコピーします。

- Oracle サーバを起動します。Windows ユーザーにとって最も簡単な方法は、**Start** > **Programs** > **Oracle** > ....から **Start Soa suite** を起動することです。また、<**ORA_INSTALL_DIR**>¥**j2ee**

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

¥home¥bin ディレクトリの **runstartupconsole.bat** を実行して Oracle サーバを起動することもできます。

- Web ブラウザを開き、**http://yourMachineName:8888/index2.html** に移動して、サーブレットの例を表示します。

トラブルシューティングのために、入力ミスをチェックしてください。

4.5.1.3.11 Sun Java System アプリケーション PE（9.0、8.2）

次の手順では、Sun Java System Application PE（9.0 または 8.2）でサーブレットのサンプルを設定および実行する方法を示します。手順では、システムに Sun Java System Application PE サーバがインストールされていることを前提としています。Sun App Server インストールの場所は**<SAP_INSTALL_DIR>**として参照され、EspressReport インストールの場所は**<ER_INSTALL_DIR>**として参照されます。

- まず、独自のディレクトリを作成します。ディレクトリの場所は**<USER_DIR>**として参照されます。

- 次に、**<SAP_INSTALL_DIR>¥samples¥quickstart** ディレクトリに移動し、**hello.war** ファイルを**<USER_DIR>**ディレクトリにコピーします。次に、hello.war ファイルを展開します。

  war ファイルを解凍するには、jar または unzip を使用できます。

  jar を使用してアンパックするには、まず **java¥bin** ディレクトリがパスに含まれていることを確認し、**<USER_DIR>**ディレクトリで次のコマンド **jar -xf hello.war** を実行します。

  次の手順に従います（ファイルは**<ER_INSTALL_DIR>¥help¥examples¥servlet¥StreamingReport** ディレクトリにあります）。

- **servlet.html** ファイルを編集し、コード内の URL の部分を **127.0.0.1:8080/servlet** から **yourMachineName:8080/hello/servlet** に変更します。

- html ファイル（**columnar.html**、**servlet.html** および **index.html**）を**<USER_DIR>**ディレクトリに配置します。

- **streamingReportServlet.java** ファイルを**<USER_DIR>¥WEB-INF¥classes** ディレクトリに配置します。

- **<USER_DIR>¥WEB-INF** ディレクトリにある **web.xml** ファイルに次のコードフラグメントを挿入します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/servlet/streamingReportServlet</URL-pattern>

</servlet-mapping>
```

- 次に、**<USER_DIR>¥WEB-INF¥classes** ディレクトリに移動し、**streamingReportServlet.java** をコンパイルします。**ReportAPIWithChart.jar** と **j2ee.jar** をクラスパスに含めてください（**j2ee.jar** ファイルは**<SAP_INSTALL_DIR>¥lib** ディレクトリにあります）。

- コマンドウィンドウで**<USER_DIR>**ディレクトリに移動し、次のコマンドを実行して war ファイルを作成します。**jar -cvf hello.war ***。

- **hello.war** ファイルを**<SAP_INSTALL_DIR>¥domains¥domain1¥autodeploy** ディレクトリに移動します。

- **ReportAPIWithChart.jar** および **qblicense.jar** ファイルを **<SAP_INSTALL_DIR>¥domains¥domain1¥lib¥applibs** ディレクトリにコピーします。

- **<SAP_INSTALL_DIR>¥bin¥asadmin start-domain domain1** を実行して、Sun App サーバを起動します。Windows ユーザーにとって最も簡単な方法は、**Start** > **Programs** > **Sun Microsystems** > **Application Server PE 9.**の **Start Default Server** を起動することです。

- Web ブラウザを開き、**http://yourMachineName:8080/hello/index.html** に移動して、サーブレットの例を表示します。

トラブルシューティングのために、入力ミスをチェックしてください。

4.5.1.3.12 Sun Java System WebServer 7.0

次の手順は、Sun Java System WebServer 7.0 の下でサーブレットのサンプルを設定して実行する方法を示しています。手順では、Sun Java System WebServer 7.0 がシステムにインストールされていることを前提としています。Sun WebServer インストールの場所は**<SWS_INSTALL_DIR>**と参照され、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

EspressReport インストールの場所は<**ER_INSTALL_DIR**>として参照されます。

- まず、独自のディレクトリを作成します。
  ディレクトリの場所は<**USER_DIR**>として参照されます。

- 次に、<**SWS_INSTALL_DIR**>**¥samples¥java¥webapps¥simple** ディレクトリに移動し、**webapps-simple.war** ファイルを<**USER_DIR**>ディレクトリにコピーします。
  次に、**webapps-simple.war** ファイルを展開します。

war ファイルを展開するには、jar または unzip を使用します。jar を使用してアンパックするには、まず、**java¥bin** ディレクトリがパスに含まれていることを確認し、<**USER_DIR**>ディレクトリの次のコマンド **jar -xf webapps-simple.war** を実行します。

次の手順に従います（ファイルは<**ER_INSTALL_DIR**>**¥help¥examples¥servlet¥StreamingReport** ディレクトリにあります)。

- **servlet.html** ファイルを編集し、コード内の URL の部分を **127.0.0.1:8080/servlet** から **yourMachineName:81/webapps-simple/servlet** に変更します。

- **index.html** ファイルの名前を **index2.html** に変更し、html ファイル(**columnar.html**、**servlet.html** および **index.html**）を<**USER_DIR**>ディレクトリに配置します。

- **streamingReportServlet.java** ファイルを<**USER_DIR**>**¥WEB-INF¥classes** ディレクトリに配置します。

- <**USER_DIR**>**¥WEB-INF** ディレクトリにある **web.xml** ファイルに次のコードフラグメントを挿入します。<**servlet**>コードはファイルの<**servlet**>コードセクションにあり、<**servlet-mapping**>セクションの<**servlet-mapping**>に入ります。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
<servlet>

    <servlet-name>streamingReportServlet</servlet-name>
    <servlet-class>streamingReportServlet</servlet-class>

</servlet>

<servlet-mapping>

    <servlet-name>streamingReportServlet</servlet-name>
    <URL-pattern>/servlet/streamingReportServlet</URL-pattern>

</servlet-mapping>
```

次に、**<USER_DIR>¥WEB-INF¥classes** ディレクトリに移動し、**ExportServlet2.java** をコンパイルします。**ReportAPIWithChart.jar** と **j2ee.jar** をクラスパスに含めてください

<J2EE_SDK_INSTALL_DIR>¥lib ディレクトリにある **J2eeSDK** インストールから **j2ee.jar** を使用できます。

コマンドウィンドウの**<USER_DIR>**ディレクトリに移動し、次のコマンドを実行して war ファイルを作成します。**jar -cvf webapps-simple.war ***。

ReportAPIWithChart.jar および **qblicense.jar** ファイルを**<SWS_INSTALL_DIR>¥https-yourMachineName¥lib** ディレクトリにコピーします。

<SWS_INSTALL_DIR>¥admin-server¥bin¥startserv.bat を実行して管理サーバを起動します。その後、Web ブラウザを起動し、**http://yourMachineName:8989** という Admin Console ページに移動します。管理コンソールのページで、管理サーバにログインして、仮想サーバタスクの **Add web application** リンクをクリックします。これにより、[Web アプリケーションの追加]ウィンドウが表示されます。

「Web アプリケーションの追加」ウィンドウから **Browse** ボタンをクリックし、**<USER_DIR>/webapps-simple.war** ファイルにナビゲートします。アプリケーションのコンテキストルートを表し、サーバホストに関連する URI（デフォルトでは/ **webapps-simple**）を指定します。次に、**OK** ボタンをクリックします。

次の画面から、webapps-simple アプリケーションが有効になっているはずです。**Save** ボタンをクリックします。その後、画面の右上に **Deployment Pending** 警告リンクが表示されます。リンクをクリックし、**Deploy** ボタンを押します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

デプロイが成功した場合は、"The configuration has been deployed successfully to all available nodes"「構成がすべての使用可能なノードに正常にデプロイされました」という結果ウィンドウが表示されます。

Web ブラウザを開き、**http://yourMachineName:81/webapps-simple/index2.html** に移動して、サーブレットの例を表示します。

トラブルシューティングのために、入力ミスをチェックしてください。

4.5.1.3.13 Resin 3.1.0

次の手順は、Resin 3.1.0 の下でサーブレットのサンプルを設定して実行する方法を示しています。手順は、システムに Resin 3.1.0 がインストールされていることを前提としています。Resin インストールの場所は<**RES_INSTALL_DIR**>と参照され、EspressReport インストールの場所は<**ER_INSTALL_DIR**>として参照されます。

- まず、独自のディレクトリを作成します。
  ディレクトリの場所は<**USER_DIR**>として参照されます。

- 次に、<**RES_INSTALL_DIR**>**¥webapps** ディレクトリに移動し、**resin-doc.war** ファイルを<**USER_DIR**>ディレクトリにコピーします。次に、**resin-doc.war** ファイルを展開します

war ファイルを展開するには、jar または unzip を使用します。jar を使用してアンパックするには、**java¥bin** ディレクトリがパスに含まれていることを確認し、<**USER_DIR**>ディレクトリで次のコマンド **jar -xf resin-doc.war** を実行します。

- <**USER_DIR**>**¥tutorial** ディレクトリに移動し、**servlet-hello** ディレクトリを<**RES_INSTALL_DIR**>**¥webapp** にコピーします。次に、**servlet-hello** ディレクトリの名前を **test** に変更します。

次の手順に従います（ファイルは<**EC_INSTALL_DIR**>**¥help¥examples¥servlet¥DataFile** ディレクトリにあります）。

- **servlet.html** ファイルを編集し、コード内の URL の一部を **127.0.0.1:8080/servlet** から **yourMachineName:8080/test/servlet** に変更します。

- <**RES_INSTALL_DIR**>**¥webapps¥test** ディレクトリに html ファイル（**columnar.html**、**servlet.html** および **index.html**）を配置します。

- **streamingReportServlet.java** ファイルを<**RES_INSTALL_DIR**>**¥webapps¥test¥WEB-INF¥classes** ディレクトリに配置します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- **<RES_INSTALL_DIR>¥webapps¥test¥WEB-INF** ディレクトリにある **resin-web.xml** ファイルに次のコードフラグメントを挿入します。

```
<servlet-name="streamingReportServlet" servlet-class="streamingReportServlet"/>

<servlet-mapping URL-pattern="/servlet/streamingReportServlet"
servlet-name="streamingReportServlet"/>
```

- 次に、**<RES_INSTALL_DIR>¥webapps¥test¥WEB-INF¥classes** ディレクトリに移動し、**streamingReportServlet.java** をコンパイルします。**ReportAPIWithChart.jar** と **j2ee.jar** をクラスパスに含めてください。

  <J2EE_SDK_INSTALL_DIR>¥lib ディレクトリにある J2eeSDK インストールから **j2ee.jar** を使用できます。

- **ReportAPIWithChart.jar** および **qblicense.jar** ファイルを**<RES_INSTALL_DIR>¥lib** ディレクトリにコピーします。

- **<RES_INSTALL_DIR>**ディレクトリの下にある **httpd.exe** を実行して、Resin サーバを起動します。

- Web ブラウザを開き、**http://yourMachineName:8080/test/index.html** に移動して、サーブレットの例を表示します。

トラブルシューティングのために、入力ミスをチェックしてください。

4.5.1.3.14 ColdFusion MX 7.02

次の手順では、ColdFusion MX 7.02 でサーブレットのサンプルを設定して実行する方法を示します。この手順では、ColdFusion MX 7 アプリケーションサーバがシステムにインストールされていることを前提としています。ColdFusion MX 7.02 インストールの場所は**<CF_INSTALL_DIR>**として参照され、EspressReport インストールの場所は**<ER_INSTALL_DIR>**として参照されます。

次の手順に従います（ファイルは**<ER_INSTALL_DIR>¥help¥examples¥servlet¥StreamingReport** ディレクトリにあります）。

- **servlet.html** ファイルを編集し、コード内の URL の部分を **127.0.0.1:8080** から **yourMachineName:8500** に変更します。

- **<CF_INSTALL_DIR>¥wwwroot** ディレクトリに html ファイル（**columnar.html**、**servlet.html**

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

および **index.html**）を配置します。

- **streamingReportServlet.java** ファイルを**<CF_INSTALL_DIR>¥wwwroot¥WEB-INF¥classes** ディレクトリに配置します。

- **<CF_INSTALL_DIR>¥wwwroot¥WEB-INF** ディレクトリにある **web.xml** ファイルに次のコードフラグメントを挿入します。**<servlet>**コードはファイルの**<servlet>**コードセクションにあり、**<servlet-mapping>**セクションの**<servlet-mapping>**に入ります。

```
<servlet>

        <servlet-name>streamingReportServlet</servlet-name>
        <servlet-class>streamingReportServlet</servlet-class>

    </servlet>

    <servlet-mapping>

        <servlet-name>streamingReportServlet</servlet-name>
        <URL-pattern>/servlet/streamingReportServlet</URL-pattern>

    </servlet-mapping>
```

- 次に、**<CF_INSTALL_DIR>¥wwwroot¥WEB-INF¥classes** ディレクトリに移動し、**streamingReportServlet.java** をコンパイルします。**ReportAPIWithChart.jar と j2ee.jar** をクラスパスに含めてください

<J2EE_SDK_INSTALL_DIR>¥lib ディレクトリにある J2eeSDK インストールの **j2ee.jar** を使用できます。

- **ReportAPIWithChart.jar** および **qblicense.jar** ファイルを**<CF_INSTALL_DIR>¥runtime¥lib** ディレクトリにコピーします。

- ColdFusion アプリケーションサーバを起動します。Windows プラットフォームの場合、サービスとして動作するように ColdFusion アプリケーションサーバがインストールされています。自動的に起動する必要があります。サーバが動作していない場合は、Windows サービスに移動し、**ColdFusion MX 7** アプリケーションサーバサービスを開始します。必要に応じて、アプリケーションサーバを再起動します。

- Web ブラウザを開き、**http://yourMachineName:8500/index.html** に移動して、サーブレットの

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

例を表示します。

トラブルシューティングのために、入力ミスをチェックしてください。

4.5.1.4 サーブレットの実行

上記の手順に従ってサーブレットを設定したら、ブラウザから **index.html**（**EspressReport/help/examples/servlet/StreamingReport** ディレクトリにあります）を開きます。指定されたパラメータから選択し、**Get Report** をクリックします。

4.5.2 Java Server Pages（JSP）

4.5.2.1 前書き

Java Server Pages では、静的 HTML から動的コンテンツを分離することができます。JSP では、従来の方法で HTML を記述することができます。その後、動的コンテンツのコードは特殊タグ内で HTML 内に囲まれます。特殊タグは通常「<%」で始まり、「%>」で終わります。

JSP は通常の HTML ファイルと一緒に配置することができ、さまざまな形で HTML ファイルのように見えます。ただし、実行時には、JSP は通常のサーブレットに変換され、静的な HTML ページが印刷されます。

Java Server Pages は基本的に Servlet テクノロジーの拡張版です。しかし、JSP の使用には大きな利点が 1 つあります。Java Server Page の技術は、ユーザインタフェースとコンテンツ生成を分離して、デザイナーが基本的な動的コンテンツを変更せずに全体のページレイアウトを変更できるようにします。

EspressReport はいくつかの JSP の例を提供しています。この例では、**EspressReport/help/examples/jsp/StreamingReport** を見ていきます。この例は、他の JSP サンプルと独自の JSP コードを実行するためのガイドとしても使用できます。

4.5.2.2 ランニングアンダー

4.5.2.2.1 Apache Tomcat 4.0

- **ReportAPIWithChart.jar** と **servlet.jar** をクラスパスに含めてください。例えば、

```
set classpath=c:¥EspressReport¥lib¥ReportAPIWithChart.jar;c:¥Apache Tomcat 4.0¥common¥lib¥servlet.jar;
```

- **webapps¥ROOT ¥**の下に report というサブディレクトリを作成します。したがって、Tomcat が **c:¥ Apache Tomcat 4.0** にインストールされている場合は、新しいディレクトリ **c:¥ Apache Tomcat 4.0¥webapps¥ROOT¥report** が作成されます。

- **StreamingReport.jsp** および **streamerror.jsp を webapps/ROOT/report** ディレクトリにコピー

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

します。

- **webapps/ROOT/WEB-INF/classes** ディレクトリの下にある report という別のディレクトリを作成します。これで、**c:¥Apache Tomcat 4.0¥webapps¥ROOT¥WEB-INF¥classes¥report** になりました。

- **CreateStreamingReport.java** ファイルを **c:¥Apache Tomcat 4.0¥webapps¥ROOT¥WEB-INF¥classes¥report** ディレクトリに配置します。

- **CreateStreamingReport.java** をコンパイルします。**CreateStreamingReport.class** が同じディレクトリに存在するはずです。

- Tomcat サーバを起動します。（**startup.bat** を実行して）**EspressManager** を起動します。

- Web ブラウザを開き、**http://yourMachineName:8080/report/StreamingReport.jsp** に移動します（実際のマシン名を yourMachineName に置き換えてください）。

- 必要なオプションを指定して送信し、レポートを取得します。

4.5.2.2.2 WebSphere 3.5

- **ReportAPIWithChart.jar** をノードの下の従属クラスパス（通常はマシン名で呼び出されます）に追加します。

- ツールバーの **Wizards** ボタン（最後のボタン）をクリックし、**Create a Web Application** を選択します。

- Web アプリケーションの名前として **Quadbase**（引用符なし）を入力します。**Enable File Servlet** オプションをオンにし、**Server Servlet by classname** をオンにして **Next** をクリックします。

- デフォルトの **Default Servlet Engine** をノードとして選択し（デフォルトのサーブレットエンジンを選択できるまでツリーを展開したままにする）、**Next** をクリックします。

- Web アプリケーションの Web パスを/**Quadbase** に変更し、**Next** をクリックします。

- **ReportAPIWithChart.jar** をクラスパスに追加します。

- ドキュメントルートに注意してクラスパスに追加します（存在しない場合は、このパスのディレクトリを作成する必要があります）。**Finish** をクリックします。

- 既定のサーバを再起動します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- **Quadbase** と呼ばれる Web サーバのドキュメントルートの下にディレクトリを作成し（パスが Web アプリケーションのドキュメントルートと一致することを確認してください）、そこに.jsp ファイルと Java ファイルを移動します。

- **CreateStreamingReport.java** を変更してパッケージを削除（またはコメントアウト）し、Java ファイルをコンパイルします。クラスファイルは、ドキュメントルートにある必要があります。

- .jsp ファイルを変更して、**report.CreateStreamingReport** を **CreateStreamingReport** に置き換えます。

- ブラウザを起動し、**http://<machine name>/Quadbase/StreamingReport.jsp** を開きます。パラメータを入力し、ボタンをクリックしてレポートを取得します。**<machine name>**を WebSphere 管理コンソールにあるホスト別名設定に置き換えてください。

4.5.3 ファイルへのレポートの保存とブラウザへのレポートの送信

サーブレットと JSP を使用して、レポートをブラウザに戻すことができます。レポートは、アプレットで表示することができ、JPEG や PNG などの静止画像として表示することもできます。ネットトラフィックと帯域幅の可用性は、アプレットを使用するかスタティックイメージを使用するかを決める重要な要素です。

ブラウザに返されるレポートは、ファイルに保存するか、ブラウザに直接送る、2 つの方法で実行できます。最初のシナリオでは、レポートはサーバ側に保存され、ファイルの場所は、クライアントのブラウザに表示されて、レポートが表示されます。2 番目のシナリオでは、グラフ自体が文字列または出力ストリームとして、クライアントブラウザに送信されます。

4.5.3.1 レポートをファイルに保存する

サーブレットと JSP は、表示されたグラフを再利用またはコピーが必要な場合、グラフをファイルに保存することができます。ここで、レポートはサーバ側のファイルに保存されます。しかし、これらのタイプのサーブレットと JSP は慎重に構築する必要があります。複数のクライアントがページをヒットした場合、あるクライアントが別のクライアントのグラフを見ることができます。ブラウザは、以前の静的画像のグラフをキャッシュすることがあり、リフレッシュ、リロードしない限り、新しいグラフを表示しないことがあります。ファイルは、サーバ側に保存されるため、クライアントブラウザがグラフを表示する前に、ファイルをオーバライドしないように注意する必要があります。サーバスペースを占有しないようにするには、定期的にパージする必要があります。

どちらの場合も、グラフは目的の形式にエクスポートされます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
QbReport report = new QbReport (.....);
....
....
....
int format = QbReport.RPT;      //Here set to an interactive format.  For a static format change to QbReport.HTML
report.export(format, "report");
```

HTML ファイルはブラウザに返され、HTML ファイル内のチャートの場所がブラウザに表示されます。

```
    <applet code="quadbase.reportdesigner.ReportViewer.Viewer.class" width=100% height=100% archive="http://<machineName>/EspressReport/lib/ReportViewer.jar">
        <PARAM name="filename" value="report.rpt">

    </applet>
```

ここではレポートビューアが使用されています。上記のコードでレポートビューアを使用する場合、EspressManager を実行している必要があります。

静的フォーマットの場合、HTML ファイルを呼び出す（または HTML ファイル内の PDF ファイルへのリンクを提供する）だけで、レポートを表示することができます。

この方法を使用すると、レポートを返すページをサーブレットまたは JSP から分離し、サーブレットまたは JSP の外で事前に設計することができます。データベースまたはハッシュテーブルを使用して、生成されたレポートを追跡し、必要に応じてそれらを再度表示したり、レポートを定期的に再生成したりすることができます。

サーバ側に HTML または DHTML ファイルとしてレポートを保存するサーブレットの例を見るには、**EspressReport/help/examples/servlet/Report** にアクセスしてください。

チャートを含むレポートを HTML または DHTML 形式でエクスポートする場合は、**setHTMLParameters** メソッドを使用してチャートをエクスポートするディレクトリ、そのディレクトリにマップされた http URL、およびエクスポートチャートの名前を設定する必要があります。したがって、エクスポートされたレポートでは、**<img src>**タグは指定された URL とファイル名（拡張子付き）を指しています。

たとえば、3 つのチャートと次のメソッド呼び出しを持つレポートがあるとします。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
reportChartObject.setHTMLParameters("D:/inetpub/wwwroot/EspressReport/chartimages/",
"http://<machine name>/EspressReport/chartimages/",
"ReportWithChart");
```

レポートがエクスポートされると、上記のコードでは、グラフが **D:/inetpub/wwwroot/EspressReport/chartimages/**ディレクトリにエクスポートされ、グラフの名前は **ReportWithChart_0、ReportWithChart_1、ReportWithChart_2** になります（拡張子は画像セットの種類によって異なります）。HTML/DHTML レポートには、次のコード行が表示されます。

```
<img src = "http://<machine name>/EspressReport/chartimages/ReportWithChart_0">
<img src = "http://<machine name>/EspressReport/chartimages/ReportWithChart_1">
<img src = "http://<machine name>/EspressReport/chartimages/ReportWithChart_2">
```

レポートのチャートの代わりに使用します。

チャートを含むレポートをサーバ側に HTML または DHTML ファイルとして保存するサーブレットの例を見るには、**EspressReport/help/examples/servlet/ReportWithChart** を参照してください。

ハイパーリンクを含むグラフをレポートに追加することもできます。これらのタイプのレポートを DHTML または HTML ファイルにエクスポートすると、デザイナでマップファイルのオプションが選択されているか、レポート API でオプションが設定されている限り、DHTML/HTML ファイル内にマップファイルが自動的に生成されます。このオプションは、**ReportChartObject** へのハンドルを取得し、**setExportMapFile** メソッドを使用して設定します。

```
ReportChartObject reportChartObject = .....;
reportChartObject.setExportMapFile(true);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.5.3.2 レポートをブラウザに直接送信する

サーブレットと JSP はブラウザに直接グラフを送信できるため、サーバ側にファイルとして保存する必要はありません。したがって、サーバ上のディスク領域がいっぱいになっておらず、ファイルのメンテナンスを行う必要もありません。ただし、ブラウザが印刷されていない限り、グラフの永久コピーは作成できません。したがって、再度必要な場合は、チャートを再生成する必要があります。

ここでは、チャートは静的画像の場合は出力ストリーム、またはインタラクティブチャートの場合は出力ストリームにエクスポートされます。

```
public class OutputStreamServlet extends HttpServlet  {

        public void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {

            //first, set the "content type" header of the response
            response.setContentType("text/html");
            //where response is the response to the servlet

            OutputStream toClient = res.getOutputStream();
            QbReport report = new QbReport (.....);
            ....
            ....
            ....
            report.export(QbReport.HTML, toClient);

        }

    }
```

ブラウザでは、サーブレット URL を使用して、ストリームされた HTML ページを表示できます。 たとえば、URL **http://machine_name:8080/servlet/OutputStreamServlet** を入力してレポートを生成し、HTML ページとしてブラウザにストリーミングすることができます。

対話型レポートの場合、レポートは文字列としてエクスポートされます。

```
QbReport report = new QbReport (.....);
String buffer = report.exportReportToString();
....
....
....
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

HTML 側では、レポートビューアを使用してインタラクティブレポートを表示できます。
ここでは、HTML は以下のようにサーブレット・コード自体に組み込まれています。

```
toClient.println("<applet code=¥"quadbase.reportdesigner.ReportViewer.Viewer¥" width=600
height=600 " + "archive=¥"http://<machineName>/EspressReport/lib/ReportViewer.jar¥">");
toClient.println("<PARAM name=¥"ReportData¥" value=¥"" + report.exportReportToString()
+ "¥">");
toClient.println("</applet>");
```

レポートを HTML または DHTML ファイルとしてストリーミングするサーブレットの例を見るには、**EspressReport/help/examples/servlet/StreamingReport** を参照してください。

JSP では、Java Bean オブジェクト（**myStreamReport**）を作成する必要があります。このオブジェクトはレポートを作成し、それを **String** オブジェクトとして返します。次のように JSP ページを変更する必要があります。

```
<jsp:useBean id="myStreamReport" scope="page" class="streamingReport.CreateReport" />
<jsp:setProperty name="myStreamReport" property="*"/>
<applet code="quadbase.reportdesigner.ReportViewer.Viewer" width=600 height=600
archive="http://<machineName>/EspressReport/lib/ReportViewer.jar">
<PARAM name="ReportData" value="<%= myStreamReport.export()%>">
</applet>
```

myStreamReport Java Bean には次のコードが含まれます。

```
public string export() throws Exception {

        QbReport report = new QbReport(....);
        return report.exportReportToString();

 }
```

レポートを HTML ファイルとしてストリーミングする JSP の例を見るには、**EspressReport/help/examples/jsp/StreamingHTML** を参照してください。

チャートを含むレポートを（HTML または DHTML 形式で）ストリーミングする場合は、チャートをサーバ側の物理的な場所に保存する **setHTMLParameters** メソッドを使用するか、提供されている **RPTImageGenerator** サーブレットを使用してチャートをストリーミングできます。このサーブレット

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

を使用するには、サーブレットランナーまたはアプリケーションサーバからアクセス可能にする必要があります。このサーブレットにアクセスする URL は、**http://<machine name>:<port number>/servlet/RPTImageGenerator** でなければなりません。このサーブレットのソースコードは、**EspressReport/ImageGenerator** ディレクトリにあります。次に、サーブレット/JSP コードで、**QbReport.setDynamicExport**（**boolean state**、**String serverName**、**int servletRunnerPort**）を使用
する必要があります。

たとえば、3 つのチャートと次のメソッド呼び出しを持つレポートがあるとします。

```
report.setDynamicExport(true, "Aphrodite", 8080);
```

レポートがエクスポートされると、上記のコードでマシン Aphrodite とポート 8080 を使用してチャートがストリーミングされます。RPTImageGenerator サーブレットは URL **http://Aphrodite:8080/servlet/RPTImageGenerator** を使用して呼び出され、チャートをブラウザに戻します。

このサーブレットの URL は、QbReport クラスの **setServletDirectory**（**String**）メソッドを使用して変更することもできます。たとえば、上記の例では、次のコード行が追加されたとします。

```
report.setServletDirectory("EspressReport/Test/");
```

リンクにはデフォルトのサーブレットの代わりに
http://Aphrodite:8080/EspressReport/Test/RPTImageGenerator がリンク内に表示されます。

さらに、SSL を使用している場合、または相対リンクを使用して **RPTImageGenerator** を検索する場合は、次のメソッドを使用して、**RPTImageGenerator** の検索に使用される IMG タグの SRC 属性を構成できます。

```
/** use a relative URL for the SRC tag. */
    report.setDynamicExport(boolean state, boolean relativeURLToImageGenerator);

    /** use https protocol along with specified server name and port number for the SRC tag. */
    public void setHttpsDynamicExport(boolean state, String serverName, int
servletRunnerPort);
```

グラフを含むサーブレットのサンプルを HTML または DHTML ファイルとして表示するには、**EspressReport/help/examples/servlet/StreamingReportWithChart** を参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ハイパーリンクを含むチャートをレポートに追加することもできます。これらのタイプのレポートを DHTML または HTML ファイルにエクスポートすると、デザイナでマップファイルのオプションが選択されているか、レポート API でオプションが設定されている限り、DHTML/HTML ファイル内にマップファイルが自動的に生成されます。このオプションは、**ReportChartObject** へのハンドルを取得し、**setExportMapFile** メソッドを使用して設定します。

```
ReportChartObject reportChartObject = .....;
reportChartObject.setExportMapFile(true);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.6 WebObjects を使った作業

4.6.1 前書き

EspressReport for WebObjects は、WebObjects とのシームレスな統合を実現します。WOComponent クラスを使用して動的ページを返すだけで、標準の EspressReportAPI と簡単に統合できるだけでなく、WebObjects の非 Java 標準コレクションクラス NSArray でレポートを生成することもできます。これにより、エンタープライズオブジェクトを使用するユーザーがデータを取得するのが非常に簡単になります。

EspressReport for WebObjects は、チャートによるストリーミングレポート、ドリルダウンレポート、パラメータ化されたレポートなどの高度なレポート機能もサポートしています。

WebObjects 用の EspressReport は、個別の jar ファイル **EspressReportForWebObjects.jar** として利用できます。WebObjects プロジェクトとの統合は非常に簡単です。プロジェクトフレームワークに追加するだけです。ソースは利用可能です。ソースファイルを変更してプロジェクトに追加することができます。

この章では、WebObjects アプリケーション環境での EspressReport の基本的な統合について説明します。また、EspressReport のさまざまな機能を使用するためのステップバイステップの例も説明します。

4.6.2 基本 API

このセクションでは、WebObjects アプリケーション内で基本的な Report API 機能の一部を統合する方法について説明します。

4.6.2.1 EspressReport 標準 API との統合

Step 1：QbReport オブジェクトを作成する

次のコードは、ゼロから QbReport オブジェクトを作成する方法を示しています。さまざまなデータソースからレポートを作成する方法の詳細については、EspressReport レポート API を参照してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
//not using EspressManager, we will use a datasource from file
QbReport.setEspressManagerUsed(false);

//Set up column mapping
ColInfo colInfo[] = new ColInfo[14];
for (int i = 0; i < colInfo.length; i++)

    colInfo[i] = new ColInfo(i);

//allocate the QBReport Object using the specified parameters
QbReport report = new QbReport((Applet)null, QbReport.COLUMNAR,
"/EspressReport/help/examples/DataSources/text/sample.dat", colInfo, null);
```

Step 2: メソッドが WOComponent を返すように、コードを上記のコードに追加します。これによりメソッドのコード全体は次のようになります。

Copyright(C) 2018 Climb Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
public WOComponent returnReport() {

    //allocates a new WOComponent with static type WOReport
    //this is possible because WOReport extends WOComponent
    WOReport nextPage = (WOReport)pageWithName("WOReport");

    QbReport.setEspressManagerUsed(false);
    ColInfo colInfo[] = new ColInfo[14];
    for (int i = 0; i < colInfo.length; i++)
        colInfo[i] = new ColInfo(i);

    QbReport report = new QbReport((Applet)null, QbReport.COLUMNAR,
"/EspressReport/help/examples/DataSources/text/sample.dat", colInfo, null);

    //load in the WBReport into the WOReport
    nextPage.setReport(report);

    //configure to export the report in DHTML format
    nextPage.setExportFormat(QbReport.DHTML);

    //return the WOComponent (super-class of WOReport)
    return nextPage;

}
```

4.6.2.2 WebObjects コレクションクラスとの統合 NSArray

WOQbReport クラスは、ReportAPI 内の **QbReport** を拡張し、WebObjects をサポートするために使用されるメインクラスです。そのコンストラクタは、データソースとして **EOGenericRecord** の **NSArray** を取得し、それに基づいてレポートを作成します。

レポートを作成する前に、レポートで使用するデータソースが必要です。ここでは、EOModel をマップしたデータベースを使用していると仮定します。EOModel には、エンティティ（テーブル）、属性（フィールド）、および関係（外部キー）のデータベースへのマッピングが含まれます。EOModel の設定方法については、EOModel と EO フェッチの仕様を参照してください。EOModel と EO Fetch Specification の理解が強い場合は、このセクションを安全にスキップすることができます。

4.6.2.2.1 EOModel と EO フェッチの仕様

EOModel は、ビジネスロジックからデータベース層を分離する抽象です。EOModel を使用して、データベースデータをオブジェクトのように扱うことができます。ユーザーが EOModeler を使用して適切

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

なEOModelをセットアップした後、EOModelの属性に基づいてデータをフェッチする準備ができます。

EOModel には、エンティティ（SQL のテーブルに似ています）と属性（SQL のテーブルの列に似ています）があります。EOModel のエンティティは、データベースのテーブルとビューにマップされます。EOModel 内の属性は、データベース内のテーブルおよびビューの列にマップされます。さらに、EOModel には、エンティティ間のキーベースの論理的関連である関係も含まれています。独自の EOModel を設定する前に、エンティティ、リレーションシップ、およびアトリビュートの概念を理解することが重要です。あなたの EOModel を設定する前に、EOModel の関連トピックを見直してください。

データベース用の EOModel をセットアップしたら、そこからデータを取得する準備が整いました。**NSArray** としてデータを取得する方法は、フェッチ仕様を記述することであることを思い出してください。**EOFetchSpecification** は、必要なデータの資格が何であるかを指定するオブジェクトです。ある意味では、EOModel の Fetch Specification の記述は、データベースの Query の記述と似ています。実際、EOModel のためのフェッチ仕様があれば、常にデータベースの対応するクエリを書くことができます。しかし、その逆は真ではありません。すべてのクエリがフェッチ指定として書き込まれるわけではありません。このように、フェッチ仕様は、プログラマがオブジェクト指向の方法でデータを処理するのに役立つ抽象として機能しますが、データベースデータの制御に対してクエリが持つ力を失います。

これについて言えば、データベースの SQL クエリをフェッチ仕様に変換する例を見てみましょう。これは必ずしも可能ではないことを覚えておいてください。しかし、私たちの例では、それがどのようにできるかを説明するために私たちの仕事を制限しています

Orders、**OrderDetails**、**Categories**、および **Products** の 4 つのテーブルを持つデータベーススキーマを考えてみましょう。

Orders：
顧客の注文に関する情報を格納します。

- 属性 - OrderID、ShipCity、および ShipState。
- 主キー - OrderID

OrderDetails：
特定の注文に関する詳細。

- 属性 - OrderID、ProductID、Quantity
- 主キー – ProductID

Categories：

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

製品のカテゴリ

- 属性 - CategoryID、CategoryName
- 主キー – CategoryID

Products：

製品に関する情報

- 属性 - CategoryID、ProductID、ProductName、UnitsInStock
- 主キー - ProductID

次に、クエリを考えてみましょう。

```
SELECT Orders.OrderID, Orders.ShipCity, Orders.ShipState, Categories.CategoryName,
Products.ProductName, OrderDetails.Quantity
FROM Orders, OrderDetails, Categories, Products
WHERE Orders.ShipState = 'NY'
AND Products.CategoryID = Categories.CategoryID
AND OrderDetails.OrderID = Orders.OrderID
AND OrderDetails.ProductID = Products.ProductID
ORDER BY Orders.OrderID DESC, Orders.ShipCity ASC;
```

このクエリは 4 つのテーブルを結合し、NY にのみ出荷される注文を取得し、OrderID 列によって降順でレコードを注文し、データレコードを返す前に数量列で昇順に再度注文します。

ここで、この SQL クエリをフェッチ仕様に変換する方法を見てみましょう。

クエリに制限を設けるには、**EOQualifier** オブジェクトを **EOFetchSpecification** コンストラクタに渡す必要があります。EOQualifier の static メソッド **qualifierWithQualifierFormat()**は、EOQualifier を返します。qualifierWithQualifierFormat()メソッドの最初の引数の SQL Where 句に似た String を単に渡すと、フェッチされるデータを制限する有効な **EOQualifier** オブジェクトを作成できます。**EOQualifier** を構築するには、次のコードを記述するだけです。

```
EOQualifier qual = EOQualifier.qualifierWithQualifierFormat

    ("shipstate = NY", null); //we set the binding to null for simplicity
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

上記のコードでは、SQL クエリの Where 句は String 引数を渡して EOQualifier オブジェクトに変換されます。また、EOQualifier の引数に 1 つ以上のブール演算子を組み合わせるために AND と OR を含めることもできます。さらに、EOQualifier クラスには、クエリの他の制限用に設計されたサブクラスのコレクションもあります。これらのサブクラスの中には、ブール修飾子、EOAndQualifier、EOOrQualifier、および EONotQualifier があります。

資格があるので、Order ID と ShipCity の注文が必要です。EOSortOrdering オブジェクトの NSArray を EOFetchSpecification コンストラクタに渡すことで、このタスクを実行できます。
以下のコードです：

```
EOSortOrdering shipdateOrdering = EOSortOrdering.sortOrderingWithKey

    ("shipcity", EOSortOrdering.CompareAscending);

EOSortOrdering orderidOrdering = EOSortOrdering.sortOrderingWithKey

    ("orderid", EOSortOrdering.CompareDescending);

NSArray sortOrderings = new NSArray

    (new Object[] {shipdateOrdering, orderidOrdering});

Finally, construct the EOFetchSpecification Object using an entity name, the previously
constructed EOQualifier, and the NSArray of EOSortOrdering Objects:

EOFetchSpecification fetchSpec = new EOFetchSpecification

    ("Orders", qual, sortOrderings);
```

SQL Statement アプローチまたは Fetch Specification アプローチを使用してデータベースからデータを取得すると、同じ結果が得られるはずです。

また、フェッチ仕様が正しく機能するためには、関係が正しく定義された EOModel を設定する必要があります。つまり、アイデンティティーを自動的に結合するためには、EOModel のアイデンティティーをデータベースに追加する必要があります。EOModeler の使用方法については、Apple の Web サイトで EOModeler のオンラインマニュアルを参照してください。

4.6.2.2.2 NSArray をデータソースとして使用して WOQbReport を作成する

EOModel が作成された後、EOModel を使用してフェッチ仕様を作成し、データベースからデータオブ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ジェクトを取得する準備が整いました。最後に、これらのデータオブジェクトを **NSArray** オブジェクトの形式で **WOQbReport** コンストラクタのパラメータとして使用して、レポートを作成します。
次のコードは、この概念を示しています。

```
EOQualifier qual = EOQualifier.qualifierWithQualifierFormat

    ("shipstate = 'NY'", null);

EOSortOrdering shipdateOrdering = EOSortOrdering.sortOrderingWithKey

    ("shipcity", EOSortOrdering.CompareAscending);

EOSortOrdering orderidOrdering = EOSortOrdering.sortOrderingWithKey

    ("orderid", EOSortOrdering.CompareDescending);

NSArray sortOrderings = new NSArray

    (new Object[] {shipdateOrdering, orderidOrdering});

EOFetchSpecification fetchSpec = new EOFetchSpecification

    ("Orders", qual, sortOrderings);

EOEditingContext myEditingContext = this.session().defaultEditingContext();
NSArray data = myEditingContext.objectsWithFetchSpecification(fetchSpec);

String[] entities = new String[]{"Orders", "OrderDetails", "Products", "Categories"};
WOQbReport report = new WOQbReport(null, QbReport.SUMMARY, data, entities, colInfo,
null);
```

上記のコードでは、最初に **EOFetchSpecification** オブジェクトをインスタンス化し、EOModel を介してデータベースからデータを取得しました。次に、デフォルトの編集コンテキストを使用してデータをフェッチします。最後に、**WOQbReport** コンストラクタを呼び出し、**NSArray** をデータソースとして渡してレポートを作成します。また、**WOQbReport** のコンストラクタに Entity 名の String 配列を渡すことに注意してください。これは、**Orders** エンティティに関連するエンティティの属性も表示する必要があるためです。エンティティ **OrderDetails**、**Products**、および **Categories** はすべて、EOModel 内の何らかのタイプの関係設定で **Orders** エンティティに「関連」していることに注意することも重要です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

NSArray をデータソースとして使用して **WOQbReport** を作成する WebObjects プロジェクトの例については、**<EspressReport installation dir>/help/examples/webobjects/**フォルダの **EspressReportExamples** プロジェクトを使用してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.6.3 WebObjects アプリケーションのデプロイ

このセクションでは、Eclipse を使用して EspressReport を WebObjects アプリケーションに統合する方法を段階的に説明します。

4.6.3.1 Xcode をインストールする

Xcode は、Apple が開発した Mac OS X 上でソフトウェアを開発するためのツール群です。これには、コンパイラと他の必要なツールが含まれます。Xcode には OS X インストール CD が付属しています(デフォルトではインストールされていません)。

4.6.3.2 Eclipse をインストールする

Eclipse は Java IDE で、Mac 用の最新の Eclipse アプリケーションは **http://www.eclipse.org/downloads** からダウンロードできます。Eclipse を起動する前に、**<eclipse folder> /Eclipse.app/Contents/MacOS/eclipse.ini** を編集し、次のパラメータをファイルに追加して、Eclipse アプリケーションのメモリとヒープスペースを増やしてください。

- **-XX：MaxPermSize = 128m**
- **-Xmx512m**

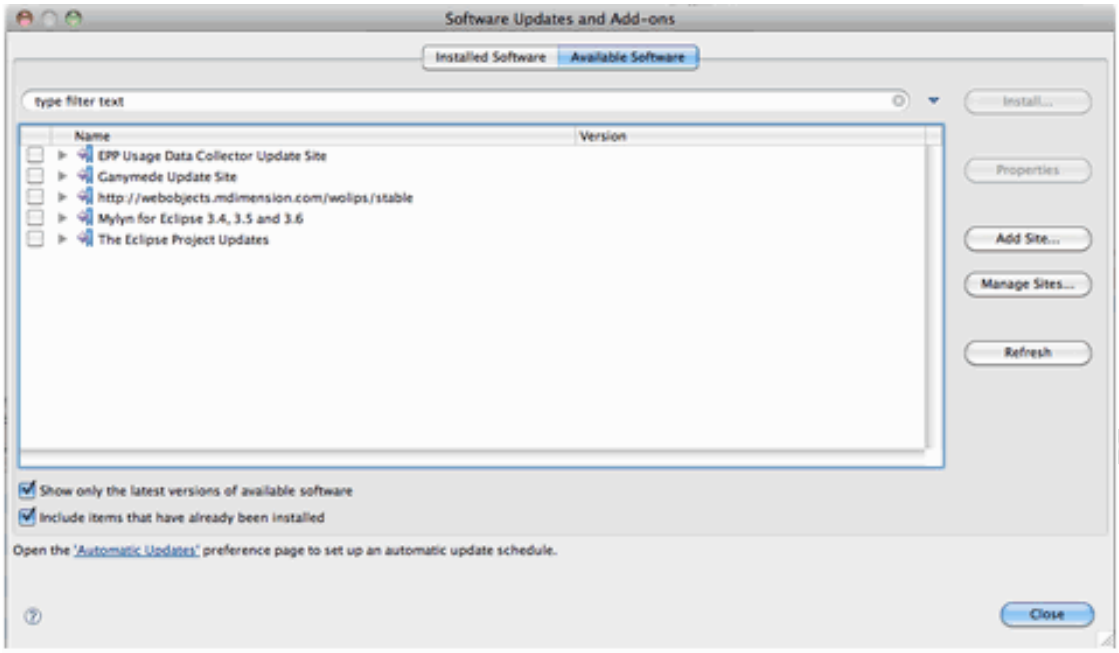
ファイルを保存したら、Eclipse を起動します。

4.6.3.3 WOLips をインストールする

WOLips は、WebObjects 開発のための一連の Eclipse ツールです。「ヘルプ」 > 「ソフトウェアの更新とアドオン」に移動して、プラグイン（Eclipse 経由）をダウンロードしてインストールすることができます。**Available Software** パネルが開きます。初めてこのプロセスを実行するときは、WOLips サイトをリストに追加する必要があります。**Available Software Sites** をクリックすると、新しいダイアログがポップアップします。ダイアログの右側にある **Add Site ...**を選択します。表示される **Add Site** パネルで、名前フィールドに **WOLips** と入力し、場所フィールドに WOLips の安定版である **http://webojects.mdimension.com/wolips/stable** の URL を入力して **OK** をクリックします。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



WOLips サイトをリストに追加したら、リストを完全に展開してすべてのオファリングを表示します。標準インストールのみを選択します。インストール後、Eclipse を再起動します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

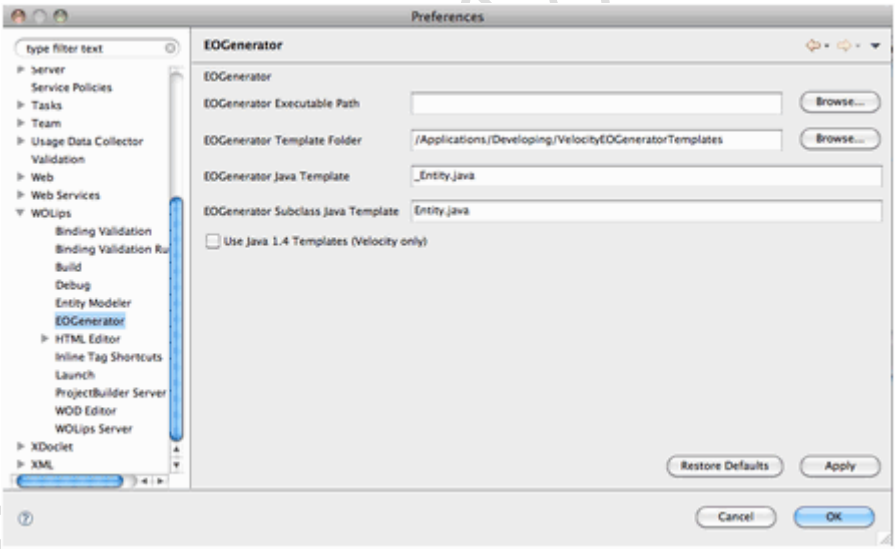
4.6.3.4 Velocity EGenerator をインストールする

これはプログラマが EO 用の Java ソースを生成するのを助ける外部の Java プログラムです。**http://webobjects.mdimension.com/wolips/EOGenerator/Velocity%20EOGenerator%20Templates/**からテンプレートをダウンロードし、**/Applications/Developing/VelocityEOGeneratorTemplates** フォルダに配置します。

このテンプレートフォルダ内には、**_Entity.java** と **Entity.java** の 2 つの標準テンプレートがあります。Eclipse で、**Preferences** > **WOLips** > **EGenerator** を選択し、次の情報を入力します。

- EGenerator テンプレートフォルダ:/Applications/Developing/VelocityEOGeneratorTemplates
- EOGenerator Java テンプレート：_Entity.java
- EOGenerator サブクラス Java テンプレート：Entity.java

EspressReport で最初の WebObjects サンプルを作成する準備が整いました。



4.6.3.5 新しい WO アプリケーションを作成する

Eclipse で、WOLips パースペクティブを選択します。これはウィンドウの右上隅に表示されます。WOLips ボタンが見つからない場合は、ウィンドウの一番端にある二重右矢印をクリックし、その他を選択して、WOLips パースペクティブをリストから選択し、**OK** をクリックします。

WOLips パースペクティブでは、パッケージエクスプローラタブで **Ctrl** +クリックし、**New** > **Other** を選択します。**Select a ..**ウィンドウが表示されます。ポップアップダイアログから **WOLips** フォルダの下にある **WedObjects** アプリケーションを選択します。**New WebObjects Project** ダイアログが開いたら、プロジェクトの名前として **EspressReportApp** と入力し、**Use default location** チェックボ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ックスをオンにしてデフォルトのワークスペースの場所を使用し、**Next** をクリックしてプロジェクトを作成します。

次のウィンドウの Base Package and Component Package フィールドをクリアし、**Finish** をクリックします。パッケージエクスプローラで作成された **EspressReportApp** プロジェクトが表示されます。プロジェクトを展開し、**Sources**、**JRE System Library**、**build**、**Components**（およびコンポーネント、**Main WO**）、および **build.xml** が含まれていることを確認します。

また、Sources を展開してから（デフォルトパッケージ）、存在する各 Java ファイル（**Application.java**、**DirectAction.java**、**Main.java**、および **Session.java**）を編集してパッケージを削除します。
主要な行をインポートします。編集後、ファイルを保存します。

4.6.3.6 メイン WO コンポーネントの編集

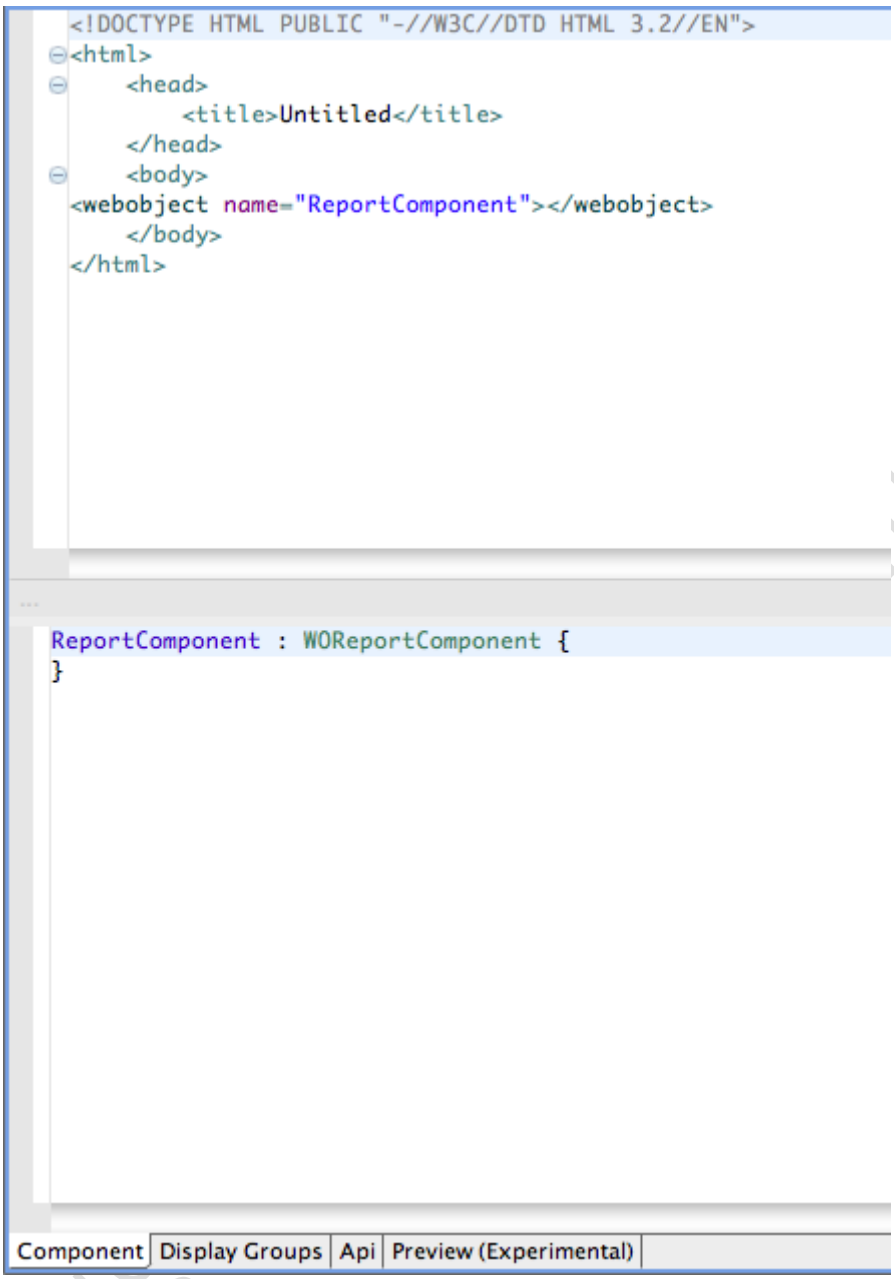
パッケージエクスプローラで、ツリーから **Components/Main WO** を選択します。メインエディタの下部にある **Component** タブをクリックし、HTML 領域（上部パネル）に次のコードを追加します。

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">

  <html>
      <head>
          <title>Untitled</title> </head>

      <body>
          <webobject name = "ReportComponent"></webobject> </body>

  </html>
```

コンポーネントエディタの下部パネルをクリックして WOD ファイルを編集し、次のように入力します。

```
ReportComponent : WOReportComponent {
}
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.



変更を保存します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.6.3.7 新しい WO コンポーネントを作成する

パッケージエクスプローラから **EspressReportApp/Components** を選択し、Ctrl +クリックでポップアップダイアログを表示します。**New > Other** を選択し、**WOLips** の下の **WOComponent** を選択します。**New WebObjects Component** ダイアログが開きます。**WOReportComponent** をコンポーネント名として入力します。**Create HTML contents** および **Create api file** の選択を解除し、**Finish** をクリックします。

ソースフォルダには新しい **WOReportComponent.java** が表示され、コンポーネントフォルダには WOReportComponent WO フォルダが表示されます。パッケージエクスプローラツリーから WOReportComponent WO を選択し、**component** タブに切り替えます。**HTML** パネル（上部パネル）に次のように入力します。

```
<webobject name = "ReportComponent"></webobject>
```

次に、WOD 定義ウィンドウ（LOWER PANEL）に次のように入力して保存します。

```
ReportComponent : WOEmbeddedObject {

    value = reportComponent;
    width = 1500;
    height = 1800;

}
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
<webobject name = "ReportComponent"></webobject>
```

WOEmbeddedObject

```
ReportComponent : WOEmbeddedObject {
    value = reportComponent;
    width = 1500;
    height = 1800;
}
```

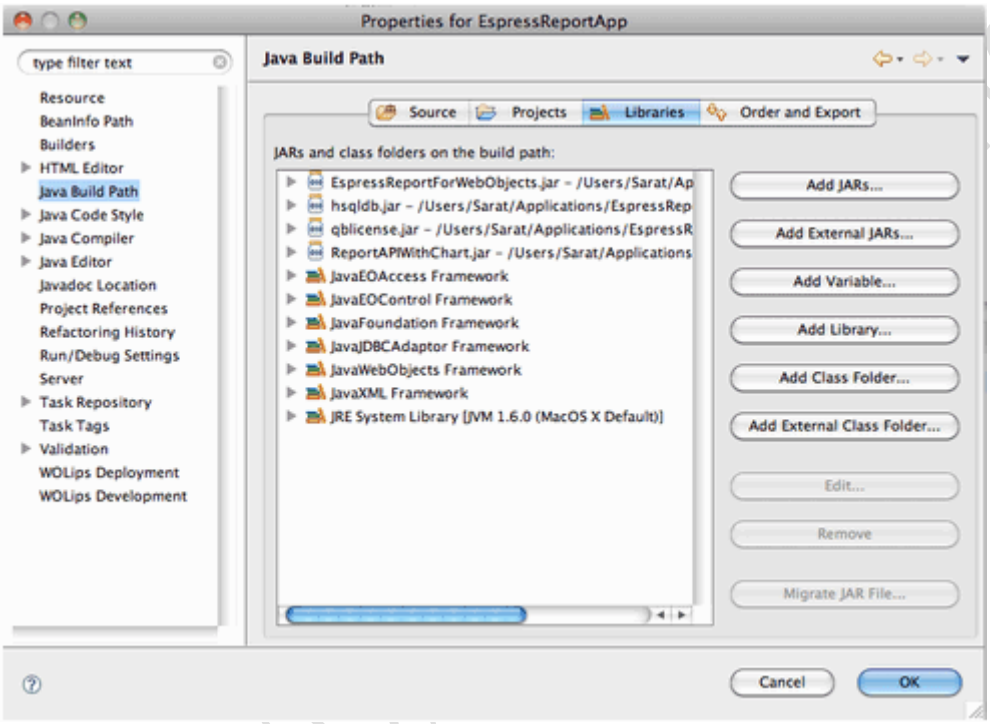
Component | Display Groups | Api | Preview (Experimental)

Copyright(C) Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.6.3.8 ビルドパスを変更する

Ctrl キーを押しながら Package Explorer ツリーから **EspressReportApp** フォルダをクリックし、ポップアップメニューから **Properties** を選択します。次に、新しいポップアップダイアログの左側の **Java Build Path** をクリックします。**Add External Jars** を選択し、**<EspressReport InstallDir>/lib** フォルダから **EspressReportForWebObjects.jar**、**ReportAPIWithChart.jar**、**hsqldb.jar**、**qblicense.jar** を参照して追加します。

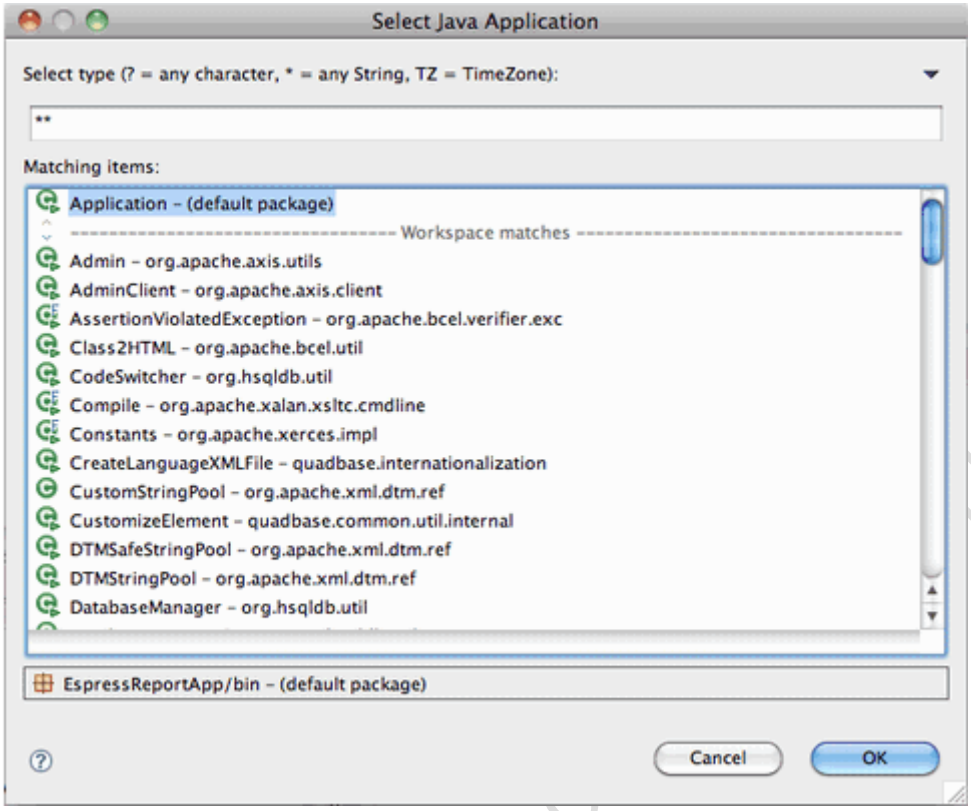


<EspressReport InstallDir>/help/examples/WebObjects フォルダから **ParameterizedNSArray.java** と **WOReportComponent.java** をパッケージエクスプローラの **EspressReportApp/Sources/(default package)**にドラッグします。

Sources フォルダの下にある **WOReportComponent.java** ファイルをクリックし、**<EspressReport InstallDir> /help/examples/DataSources/text/sample.dat** を指すように **sample.dat** へのパスを変更します。

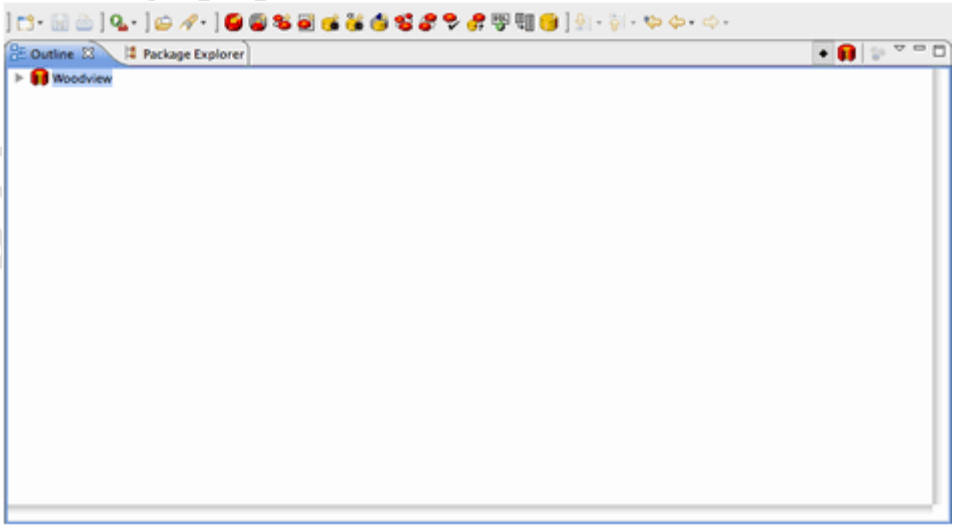
CTRL +プロジェクト名をクリックし、**Run As > WOApplication** を選択します。**Select Java Application** ダイアログで **Application** - (default package)を選択し、**OK** をクリックします。次に、デフォルトの Web ブラウザに表示されるレポートが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



4.6.3.9 エンティティモデラを使用して EOModel を作成する

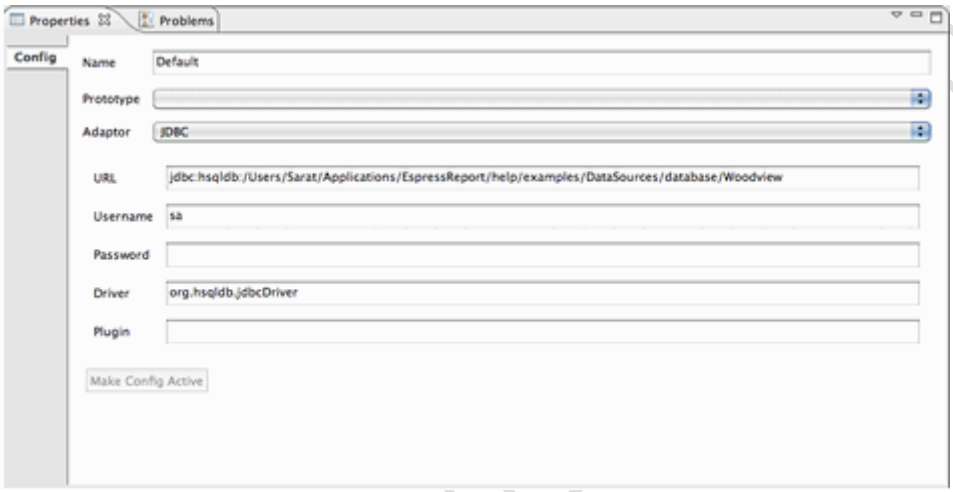
CTRL ＋ **EspressReportApp/build/EspressReportApp.woa/Contents/Resources** をクリックし、New：Other を選択します。ポップアップダイアログの **WOLips** フォルダから **EOModel** を選択します。EOModel の名前として **WoodView** を入力し、アダプタタイプとして **JDBC** を選択します。Eogenerator File を使用していることを確認して、**Finish** をクリックしてください。Entity Modeler Preserve に切り替えます。



Copyright(C) 2018 Climb.Inc. All Rights Reserved.

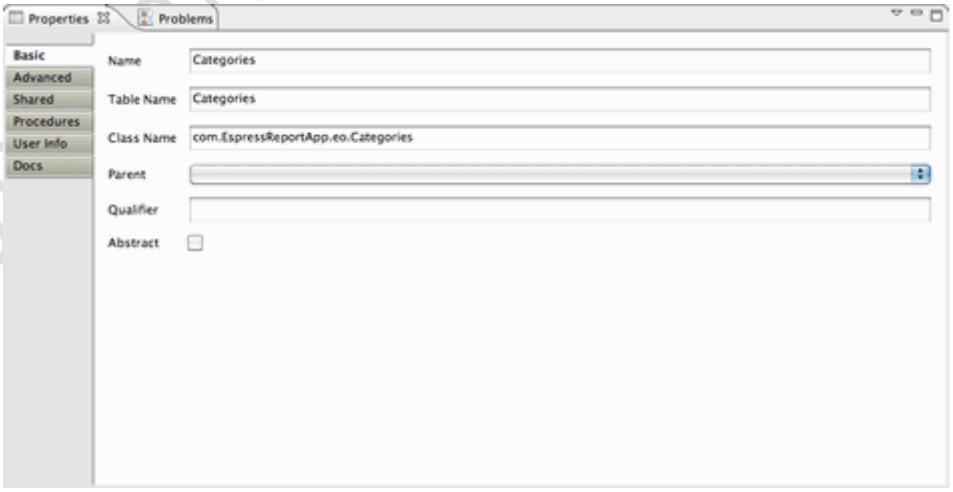
4.6.3.10 データベース情報の設定

エンティティモデラで、左のタブ付きパネルから **Outline** を選択し、**WoodView** フォルダを展開します。**Default** を選択し、下のパネルの **Properties** をクリックします。ユーザー名（**sa**）とパスワードを入力（空白）します。
URL には、
jjdbc:hsqldb:<EspressReport InstallDir>help/examples/DataSources/database/woodview と入力し、Driver に **org.hsqldb.jdbcDriver** と入力します。



4.6.3.11 エンティティの作成

新しいエンティティを作成するには、ツールバーボタンの **New Entity** をクリックします。Properties ペインの Basic タブをクリックし、新しいエンティティの名前を **Orders** に変更し、テーブル名を **Orders** に変更し、クラス名を **com.EspressReportApp.eo.Orders** に設定します。



Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Name: OrderDetails
Table Name: Order_Details
Class Name: com.EspressReportApp.eo.OrderDetails

Name: Categories
Table Name: Categories
Class Name: com.EspressReportApp.eo.Categories

Name: Products
Table Name: Products
Class Name: com.EspressReportApp.eo.Products

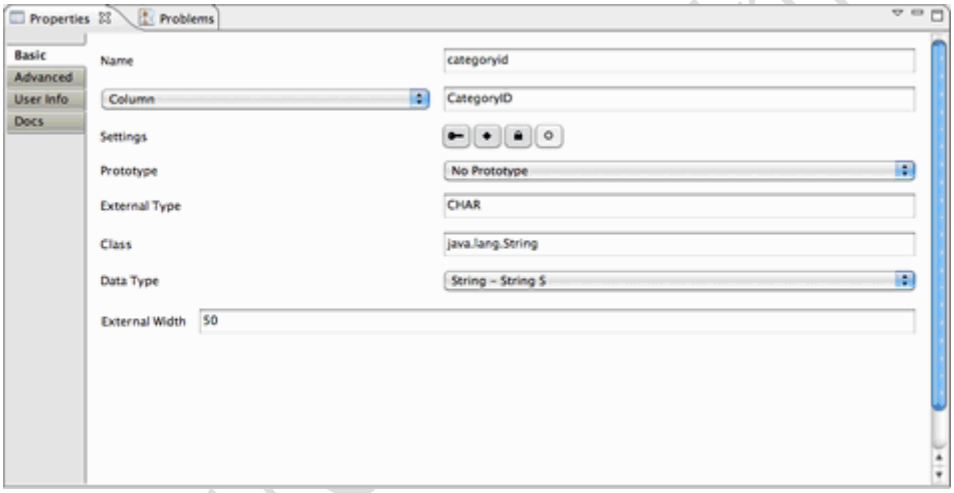
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.6.3.12 属性定義を作成する

Orders エンティティを選択し、ツールバーボタンから **New Attribute** をクリックします。
新しい属性が作成されたら、**Properties pane** を選択して次のように入力します。

Name: orderid Column: OrderID External Type: INTEGER Data Type: Integer - Integer Key icon (Primary Key): checked Diamond icon (Class Property): checked Lock icon (Used for Locking): checked



上記の手順を繰り返して、**Orders** エンティティの次の属性を作成します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Name: shipcity
Column: ShipCity
External Type: CHAR
Data Type: String - String S
External Width: 50

Key icon (Primary Key): unchecked
Diamond icon (Class Property): checked
Lock icon (Used **for** Locking): checked

Name: shipstate
Column: ShipState
External Type: CHAR
Data Type: String - String S
External Width: 50
Key icon (Primary Key): unchecked
Diamond icon (Class Property): checked
Lock icon (Used **for** Locking): checked

Copyright(C) 2018

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

OrdersDetails エンティティに次の属性を追加します。

Name: orderid
Column: OrderID
External Type: INTEGER

Data Type: Integer - Integer
Key icon (Primary Key): unchecked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name:　productid
Column: ProductID
External Type: INTEGER
Data Type: Integer - Integer
Key icon (Primary Key): checked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name: quantity
Column: Quantity
External Type: INTEGER
Data Type: Integer - Integer
Key icon (Primary Key): unchecked
Diamond icon (Class Property): checked

Lock icon (Used for Locking): checked

Copyright

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

エンティティ **Categories** を選択し、次の属性を作成します。

Name: categoryid
Column: CategoryID
External Type: CHAR
Data Type: String - String S
External Width: 50
Key icon (Primary Key): checked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name: categoryname
Column: CategoryName
External Type: CHAR
Data Type: String - String S
External Width: 50 Key icon (Primary Key): unchecked

Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Copyright(C) 2018

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Products エンティティに次の属性を追加します。

Name: categoryid
Column: CategoryID
External Type: CHAR
Data Type: String - String S
External Width: 50
Key icon (Primary Key):un checked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name: productid
Column: ProductID
External Type: INTEGER
Data Type: Integer - Integer

Key icon (Primary Key):checked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name: productname
Column: ProductName
External Type: CHAR
Data Type: String - String S
External Width: 50
Key icon (Primary Key):un checked
Diamond icon (Class Property): checked
Lock icon (Used for Locking): checked

Name: unitsinstock
Column: UnitsInStock
External Type: INTEGER
Data Type: Integer - Integer
Key icon (Primary Key):un checked
Diamond icon (Class Property): checked

Lock icon (Used for Locking): checked

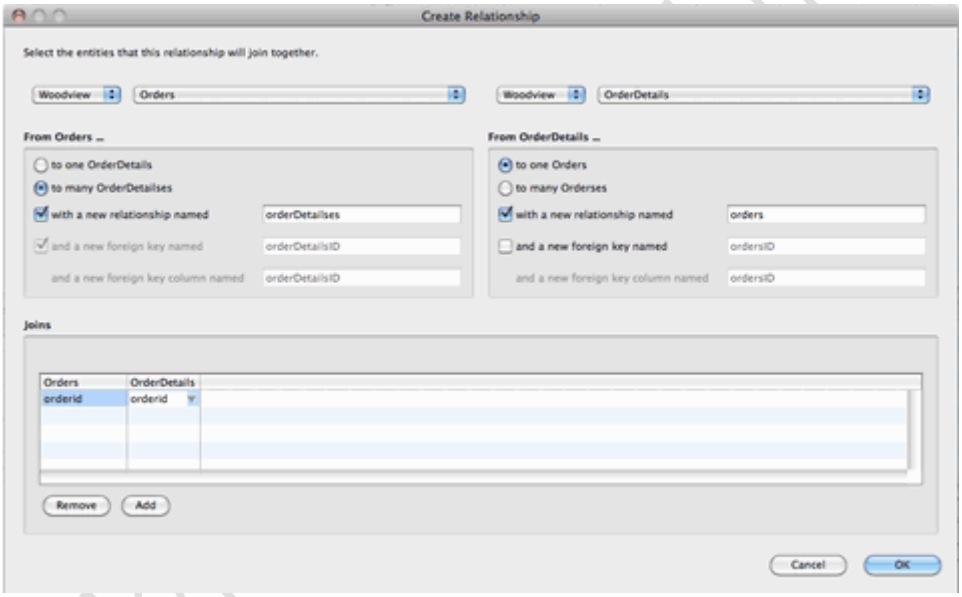
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.6.3.13 エンティティ間の関係の作成

4.6.3.13.1 「注文」と「注文明細」の関係を作成する

CTRL + **Orders** をクリックし、ポップアップメニューから **New Relationship** を選択します。
新規リレーションシップの作成ダイアログが開きます。
Woodview.Orders と **Woodview.OrderDetails** を一緒にマップします。

左パネルの **Orders ...**で OrderDetails を多数選択し、名前を付けた新しい関係を作成するチェックボックスをオンにして、外部キーのチェックボックスをオフにします。右パネルの OrderDetails から...を選択して、新しい関係を作成するチェックボックスをオンにし、外部キーのチェックボックスをオフにします。**Joins** パネルで、**Orders.orderid** と **OrderDetails.orderid** を一緒にマップし、**OK** をクリックします。



4.6.3.13.2 "OrderDetails"と "Products"の関係を作成する

CTRL + **OrderDetails** をクリックし、ポップアップメニューから New Relationship を選択します。
Woodview.OrderDetails と Woodview.Products を一緒にマップします。

左パネルの OrderDetails から... Products を 1 つ選択し、名前を付けた新しい関係を作成するチェックボックスをオンにし、外部キーのチェックボックスをオフにします。右側のプロダクトから...を選択し、多くの OrderDetails を選択し、名前の付いた新しい関係を作成するためのチェックボックスをオンにして、外部キーのチェックボックスをオフにします。

Joins パネルで、**OrderDetails.productid** と **Products.productid** を一緒にマップし、**OK** をクリックします。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.6.3.13.3 「製品」と「カテゴリ」の関係を作成する

製品を **CTRL** +クリックし、ポップアップメニューから新規関係を選択します。Woodview.Products と Woodview.Categories をマップします。

左パネルの From Products ...で、1 つのカテゴリを選択し、新しい関係を作成するためのチェックボックスをオンにし、外部キーのチェックボックスをオフにします。右パネルのカテゴリから...を選択し、多くの製品を選択し、名前を付けた新しい関係を作成するチェックボックスをオンにして、外部キーのチェックボックスをオフにします。

Joins パネルで、**Products.categoryid** と **Products.categoryid** をマップして **OK** をクリックします。

4.6.3.14 確認して保存する

エンティティ、属性、およびリレーションシップを作成したら、ツールバーの **Verify Model** ボタンをクリックします。ウィンドウにエラーメッセージは表示されません。

ビューモデルを保存するには、ツールバーの **Save** ボタンを選択します。

4.6.3.15 Java コードを作成する

EspressReportApp/build/EspressReportApp.woa/Contents/Resources の **woodview.ecogen** ファイル（存在する場合）を削除します。パッケージエクスプローラから Woodview EOModel を CTRL +クリックし、ポップアップメニューから **WOLips Tools** > **Create EOGenerater File ...**を選択します。生成された **woodview.ecogen** を **CTRL** +クリックし、ポップアップメニューから **EOGenerate** を選択します。エラーなしで戻る必要があります.Sources フォルダの下に新しい java ファイルが作成されているはずです。

4.6.3.16 NSArray を使用したテストケースの実行

Package Explorer ツリーから WOReportComponent WO を選択し、Component タブに切り替えます。WOD 定義ウィンドウ（LOWER PANEL）で、値属性を **generateReportFromNSArray** に置き換えます。

```
ReportComponent : WOEmbeddedObject {

    value = generateReportFromNSArray;
    width = 1500;
    height = 1800;

}
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

パッケージエクスプローラーペインで **Ctrl** + プロジェクト **EspressReportApp** をクリックし、ポップアップメニューから **Run As** > **WOApplication** を選択します。前と同様に、リストから **Application** -（デフォルトパッケージ）を選択します。レポートはデフォルトの Web ブラウザからポップアップするはずです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.6.3.17 NSArray を使用したパラメータ化されたレポートのテストケースの実行

Package Explorer ツリーから WOReportComponent WO を選択し、Component タブに切り替えます。WOD 定義ウィンドウ（LOWER PANEL）で、value 属性を **generateParamNSArrayReport** に置き換えます

```
ReportComponent : WOEmbeddedObject {

    value = generateParamNSArrayReport;
    width = 1500;
    height = 1800;

}
```

パッケージエクスプローラーペインで **Ctrl** + プロジェクト **EspressReportApp** をクリックし、ポップアップメニューから **Run As** > **WOApplication** を選択します。ここでも、リストから **Application** -（デフォルトパッケージ）を選択します。パラメータ入力ダイアログが表示されます。都市の **NY** を入力し、**OK** をクリックします。レポートはデフォルトの Web ブラウザからポップアップされます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
<webobject name = "ReportComponent"></webobject>
```

WOEmbeddedObject

```
ReportComponent : WOEmbeddedObject {
    value = generateParamNSArrayReport;
    width = 1500;
    height = 1800;
}
```

Component | Display Groups | Api | Preview (Experimental)

Copy

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.6.4 ノート

すべてのファイル（**EspressReportsForWebObjects.jar** ファイルのファイルと 2 つの追加ファイル）は、<**EspressReport InstallDir**>/**help/examples/WebObjects** フォルダにあります。

デフォルトでは、EspressManager は使用されません。 EspressManager を起動して実行したい場合は、必要に応じてソースコードを変更する必要があります。 EspressManager がドリルダウンレポート、サブレポート付きレポート、またはチャート付きレポートで実行されていない場合は、次の方法を使用してパスを指定する必要があります。

- report.setDrillDownPath（...）;
- report.setChartPath（...）;
- report.setSubReportPath（..）;

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.7 デプロイメント

4.7.1 前書き

EspressReport は、Report Designer、Report Viewer、Page Viewer、Chart Viewer、Scheduler、EspressManager、Report API の各コンポーネントで構成されています。Report Designer は、GUI 環境でレポートを作成するために使用されます。レポートビューアは、既に作成されたレポート（.rpt/.xml/.pak 形式で保存）を表示するために使用されるアプレットです。Page Viewer は、レポートページをページ単位で表示するために使用されるアプレットです（ページ形式で保存されます）。Chart Viewer はチャート（.cht/.tpl 形式で保存）を表示するために使用されるアプレットです。レポート API は、プログラムでレポートを作成するために使用されます。スケジューラは、レポートのエクスポートをスケジュールするために使用されます。最後に、EspressManager はユーザー管理者として機能し、データとデータのバッファリングを処理します。

qblicense.jar は、コードを実行するために CLASSPATH または HTML ページにアーカイブとして（他の追加の jar ファイルとともに）含めなければなりません。

Report Designer とスケジューラは EspressManager と組み合わせて使用する必要がありますが、レポートビューア、Page Viewer、チャートビューア、レポート API は EspressManager に接続せずに動作するオプションが用意されています。

EspressManager への接続方法に関する情報を指定する必要があります。EspressManager がアプリケーションとして実行されている場合は、API メソッドを使用して、EspressManager が配置されている IP アドレス/マシン名と EspressManager がリッスンしているポート番号を指定できます。

接続情報を設定するには、次の 2 つの API メソッドを使用します。

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

たとえば、次のコード行があります。

```
QbReport.setServerAddress("someMachine");
QbReport.setServerPortNumber(somePortNumber);
```

someMachine で動作し、**somePortNumber** でリッスンしている **EspressManager** に接続します。

EspressManager の接続情報が指定されていない場合、コードはローカルマシンの EspressManager に接続し、デフォルトのポート番号（22071）をリッスンします。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

EspressManager がサーブレットとして実行されている場合は、次の方法を使用できます。

```
public static void useServlet(boolean b);
public static void setServletRunner(String comm_URL);
public static void setServletContext(String context);
```

たとえば、次のコード行があります。

```
QbReport.useServlet(true);
QbReport.setServletRunner("http://someMachine:somePortNumber");
QbReport.setServletContext("EspressReport/servlet");
```

http://someMachine:somePortNumber/EspressReport/servlet で動作する EspressManager に接続します。

これらのメソッドは、**QbChart**、**QbReportDesigner**、**QbChartDesigner**、**QbScheduler**、および **ScheduleModifier** クラスにも存在します。

Page Viewer と Report Viewer（アプレット内）では、次のパラメータを渡すことで、EspressManager に接続情報を設定できます。

```
server_address (server address), server_port_number (port number)
```

上記のパラメータは、EspressManager がアプリケーションとして実行されている場合に設定されます。EspressManager がサーブレットとして実行されている場合は、次のパラメータが使用されます。

```
comm_protocol (servlet), comm_URL (machine name/IP address and port number),
servlet_context (servlet context)
```

以下のセクションでは、遭遇する可能性のあるさまざまな展開シナリオと各シナリオの結果について説明します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.7.2 EspressManager でデプロイする

このセクションでは、EspressManager と連携して動作するレポートコンポーネントのデプロイについて説明します。

この相互作用の詳細については、EspressReport レポート API の概要の章の「EspressManager との相互作用」セクションを参照してください。

相対 URL 参照（**help¥examples¥data¥sample.dat**）を使用するデータソースまたはレポートファイルへの参照は、EspressManager が実行されているディレクトリからの相対パスです。

4.7.2.1 Report Designer

前述のとおり、Report Designer は EspressManager と組み合わせて使用する必要があります。Report Designer は API を使用して呼び出すことができます。このシナリオでは、**ReportDesigner.jar** を CLASSPATH に追加し、レポート画像（**EspressReport/reportimages**）、画像（**EspressReport/images**）、および backgroundImages（**EspressReport/backgroundImages**）ディレクトリを.class ファイルの作業ディレクトリの下に配置する必要があります。

API から Report Designer を呼び出すときに作業ディレクトリを基準にディレクトリをコピーする代わりに、**images**/、**location/reportimages**/および **backgroundImages**/ディレクトリの場所を設定するオプションも使用できます。単純に **QbReportDesigner** コンストラクタをパラメータ **imagesPath** とともに使用します。**QbReportDesigner** コンストラクタまたは **imagesPath** パラメータの詳細については、EspressReport Java API ドキュメントを参照してください。

アプリケーションとして動作する EspressManager に接続するには、**QbReportDesigner** で次の 2 つの API メソッドを使用して接続情報を設定します。

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

サーブレットとして動作する EspressManager に接続するには、**QbReportDesigner** で次の 3 つの API メソッドを使用して接続情報を設定します。

```
static void useServlet(boolean b);
static void setServletRunner(String comm_URL);
static void setServletContext(String context);
```

EspressManager への接続情報の指定は、**QbReportDesigner** コンストラクタを呼び出す前に行う必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.7.2.2 Report Viewer

レポートビューアをアプレットまたはアプリケーション環境で使用すると、**.rpt/.xml/.pak** ファイル（Report Designer またはレポート API によって生成されたファイル）に保存されたレポートを表示できます。

Report Viewer をアプレット環境に配備するには、**ReportViewerWithChart.jar** を HTML ファイルにアーカイブファイルとして含める必要があります。

レポートビューアを使用するには、適切なアプレットコードを使用して HTML ページを作成します（詳細は、レポートビューアの章を参照してください）。レポートの場所やデータへの相対参照は、EspressManager が開始されたディレクトリからの相対パスであることに注意してください。

アプリケーションとして動作する EspressManager に接続するには、Report Viewer アプレットに次のパラメータを渡す必要があります。

server_address (server address), server_port_number (port number)

サーブレットとして動作する EspressManager に接続するには、Report Viewer アプレットに次のパラメータを渡します。

comm_protocol (servlet), comm_URL (machine name/IP address and port number), servlet_context (servlet context)

4.7.2.3 Page Viewer

アプレットまたはアプリケーション環境で、Page Viewer を使用して、レポートの特定のページを一度に表示することができます。これらの PAGE ファイルは**.page** ファイル拡張子であり、Report Designer または Report API によって生成されます。

アプレット環境に Page Viewer をデプロイするには、PageViewer.jar を HTML ファイルにアーカイブとして含める必要があります。

Page Viewer を使用するには、適切なアプレットコードで HTML ページを作成します（詳細については、Page Viewer の章を参照してください）。レポートの場所やデータへの相対参照は、EspressManager が開始されたディレクトリからの相対パスであることに注意してください。

アプリケーションとして動作する EspressManager に接続するには、Report Viewer アプレットに次のパラメータを渡します。

server_address (server address), server_port_number (port number)

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

サーブレットとして動作する EspressManager に接続するには、Report Viewer アプレットに次のパラメータを渡す必要があります。

comm_protocol (servlet), comm_URL (machine name/IP address and port number), servlet_context (servlet context)

4.7.2.4 チャートビューア

Chart Viewer の展開は、Report Viewer の展開と同様です。 Chart Viewer は、アプレットまたはアプリケーション環境で、**.cht** ファイルまたは**.tpl** ファイル（Designer または API によって生成されたファイル）に保存されたグラフを表示するために使用できます。

Chart Viewer をアプレット環境にデプロイするには、**EspressViewer.jar** を HTML ファイルにアーカイブとして含める必要があります。

Chart Viewer を使用するには、適切なアプレットコードを使用して HTML ページを作成します（詳細については、Chart Viewer の章を参照してください）。チャートの場所やデータへの相対参照は、EspressManager が開始されたディレクトリからの相対パスであることに注意してください。

アプリケーションとして動作する EspressManager に接続するには、Report Viewer アプレットに次のパラメータを渡す必要があります。

server_address (server address), server_port_number (port number)

サーブレットとして動作する EspressManager に接続するには、Report Viewer アプレットに次のパラメータを渡す必要があります。

comm_protocol (servlet), comm_URL (machine name/IP address and port number), servlet_context (servlet context)

4.7.2.5 レポート API

Report API は、アプレット環境またはアプリケーションとして使用できます。 Report API は、サーブレットまたは JSP 環境で使用して、サーバ側のレポートを生成することもできます。

Report API をデプロイするには、**ReportAPIWithChart.jar** および **ExportLib.jar** が CLASSPATH（アプリケーションおよびサーブレット/JSP 環境の場合）または HTML ファイル（アプレットの場合）に含まれている必要があります。

アプリケーションとして動作する EspressManager に接続するには、**QbReport** で次のメソッドを使用して接続情報を設定します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

サーブレットとして動作する EspressManager に接続するには、**QbReport** で以下のメソッドを使用して接続情報を設定します。

```
static void useServlet(boolean b);
static void setServletRunner(String comm_URL);
static void setServletContext(String context);
```

EspressManager への接続情報の指定は、**QbReport** コンストラクタを呼び出す前に行う必要があります。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.7.2.6 スケジューラ

EspressReport にはスケジューラも含まれています。このスケジューラは、特定の間隔でソースファイルからレポートをエクスポートするために使用できます。スケジューラの詳細については、スケジューラの章を参照してください。

Report Designer と同様に、スケジューラは EspressManager と組み合わせて使用する必要があります。スケジューラは、API を使用して呼び出すことができます。このシナリオでは、CLASSPATH に **Scheduler.jar** を追加する必要があります。

アプリケーションとして動作する EspressManager に接続するには、**QbScheduler/ScheduleModifier** で次の 2 つの API メソッドを使用して接続情報を設定します。

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

サーブレットとして動作する EspressManager に接続するには、**QbScheduler/ScheduleModifier** で次の 3 つの API メソッドを使用して接続情報を設定します。

```
static void useServlet(boolean b);
static void setServletRunner(String comm_URL);
static void setServletContext(String context);
```

EspressManager への接続情報の指定は、**QbScheduler/ScheduleModifier** コンストラクタを呼び出す前に行う必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.7.3 EspressManager なしでのデプロイ

このセクションでは、EspressReport とは独立して動作するコンポーネントのデプロイについて説明します。この相互作用の詳細については、EspressReport API の概要の章の「EspressManager との相互作用」セクションを参照してください。

EspressManager とは無関係にレポート/チャートを生成すると、EspressManager とレポート/チャートコンポーネントの対話が削除されるため、パフォーマンスが少し向上します。

重要な考慮事項は、データソースまたはレポートソースへの参照（つまり **help¥examples¥data¥sample.dat** など）は、HTML ファイルの場合の作業ディレクトリからの相対パスです。アプレット）または（アプリケーション内の）クラスファイルが実行されています。サーブレット/ JSP 環境では、通常、作業ディレクトリはクラスが存在するディレクトリとは異なります。作業ディレクトリを調べるには、サーブレット/ jsp に次のコード行を追加します。

```
System.out.println("The working directory is " + System.getProperty("user.dir") + ". ");
```

上記のコード行を持ち、**servlet/jsp** を実行することにより、作業ディレクトリが確認され、コードによって指示されるように、データファイルとレポートテンプレートを作業ディレクトリに対する相対位置に移動することができます。

4.7.3.1 Report Viewer

Report Viewer をデプロイするには、**ReportViewerWithChart.jar** を HTML ファイルにアーカイブとして含める必要があります。使用している機能に応じて、必要に応じて追加の jar を含める必要があります。

Report Viewer は、EspressManager とは独立して、アプレットコードに 1 つ以上のパラメータを追加して使用できます。

```
<param name="EspressManagerUsed" value="false">
```

レポートの場所および/またはデータへの相対参照は、HTML ファイルが置かれているディレクトリからの相対的なものであることに注意してください。たとえば、.rpt ファイルで指定されたデータソースが **help¥examples¥data¥sample.dat** で、HTML ファイルが **D:¥ EspressReport¥TestApplet** にある場合は、レポートが正常に表示されるように **help¥examples¥data¥sample.dat** は、D:¥ EspressReport¥TestApplet 内に存在する必要があります（つまり、D:¥ EspressReport¥TestApplet¥help¥examples¥data¥sample.dat が存在する必要があります）。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.7.3.2 Page Viewer

Page Viewer をデプロイするには、**PageViewer.jar** をアーカイブとして HTML ファイルに含める必要があります。使用している機能に応じて、必要に応じて追加の jar を含める必要があります。

Page Viewer は EspressManager とは独立して使用することができ、アプレットコードにはさらに 1 つのパラメータが追加されています：

```
<param name="EspressManagerUsed" value="false">
```

レポートの場所および/またはデータへの相対参照は、HTML ファイルが置かれているディレクトリからの相対的なものであることに注意してください。たとえば.rpt ファイルで指定されたデータソースが **help¥examples¥data¥sample.dat** で、HTML ファイルが **D:¥ EspressReport¥TestApplet** にある場合は、レポートが正常に表示されるように **help¥examples¥data¥sample.dat** は、**D:¥ EspressReport¥TestApplet** 内に存在する必要があります。
（つまり、D:¥ EspressReport¥TestApplet¥help¥examples¥data¥sample.dat が存在する必要があります）。

EspressManager に接続せずに Page Viewer を使用する場合、テンプレートを直接渡すことはできません。代わりにページファイルを渡す必要があります。

4.7.3.3 チャートビューア

Chart Viewer をデプロイするには、**EspressViewer.jar** を HTML ファイルにアーカイブとして含める必要があります。使用している機能に応じて、追加の jar を含める必要があります。

Chart Viewer は、EspressManager とは独立して、アプレットのコードに 1 つ以上のパラメータを追加して使用できます。

```
<param name="EspressManagerUsed" value="false">
```

レポートの場所および/またはデータへの相対参照は、HTML ファイルが置かれているディレクトリからの相対的なものであることに注意してください。たとえば.rpt ファイルで指定されたデータソースが **help¥examples¥data¥sample.dat** で、HTML ファイルが **D:¥ EspressReport¥TestApplet** にある場合は、レポートが正常に表示されるように **help¥examples¥data¥sample.dat** は、**D:¥ EspressReport¥TestApplet** 内に存在する必要があります（つまり、D:¥ EspressReport¥TestApplet¥help¥examples¥data¥sample.dat が存在する必要があります）。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.7.3.4 レポート API

Report API をデプロイするには、**ReportAPIWithChart.jar** および **ExportLib.jar** が CLASSPATH（アプリケーションおよびサーブレット/JSP 環境の場合）または HTML ファイル（アプレットの場合）に含まれている必要があります。

次のコード行を追加することで、EspressManager に接続せずに Report API を使用することもできます。

```
QbReport.setEspressManagerUsed(false);
```

このメソッドは、QbReport インスタンス化の前に呼び出さなければなりません。

レポートビューアと同様に、レポートの場所やデータへの相対参照は、クラスファイルが実行されている作業ディレクトリからの相対パスです。たとえば、指定されたデータソースが **help¥examples¥data¥sample.dat** で、クラスファイルが **D:¥EspressReport¥TestApplication** にある場合は、レポートが正常に表示されるよう **help¥examples¥data¥sample** になります。**D:¥ EspressReport¥TestApplication**
（つまり、**D:¥ EspressReport¥TestApplicationhelp¥examples¥data¥sample.dat** が存在する必要があります）内に存在する必要があります。

Report Designer で作成されたグラフには、テンプレートへの相対パスが含まれます。そのため、ディレクトリ構造を作業ディレクトリの下にミラーリングする必要があります（つまり、作業ディレクトリの下にテンプレートを含むチャートサブディレクトリを作成する）か、テンプレートを開き、次のコードを使用して絶対パスでパスを変更する必要があります。

```
//Get the Chart from the footer
ReportChartObject obj = (ReportChartObject)report.getTable().getFooter().getData(0);
String originalLocation = obj.getText();
String newLocation = new
String("http://128.0.0.1/EspressEnterprise/EspressReport/test/chart/chart/EspressReport1.1_0.tpl");
obj.setText(newLocation);
```

servlet/jsp 環境では、クラスファイルと作業ディレクトリを持つディレクトリの場所が異なる場合があります。その場合、相対参照はすべて作業ディレクトリに関するものです。たとえば、サーブレット・クラス・ファイルが **D:¥<some servlet engine>¥webapp¥examples¥quadbase¥someservlet.class** にあり、データソースが **help¥examples¥data¥sample.dat** であり、作業ディレクトリが **D:¥ <some servlet engine>¥webapp**、次に **help¥examples¥data¥sample.dat** が **D:¥ <some servlet engine>¥webapp**（D:¥ <some servlet engine>¥webapp¥help¥examples¥data¥sample.dat が存在しなければなりません）。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

この API では、特定のファイルが配置されている可能性のあるチャート、ドリルダウン、イメージ、およびサブレポートのディレクトリの場所を指定できるメソッドも用意されています。たとえば、レポートにテンプレート **EspressReport1.1_0.tpl** が存在する場合、次のコードを使用してこのテンプレートの場所を指定できます。

```
report.setChartPath("d:/EspressReport/Chart/");          //where report is an object of type QbReport
```

上記のコードは、テンプレートファイルの場所を設定するので、クラスファイルの作業ディレクトリの下にチャートのサブディレクトリを持つ必要はありません。

次の方法も同様です。

```
QbReport.setDrillDownPath(String directory);
QbReport.setImagePath(String directory);
QbReport.setSubReportPath(String directory);
```

RPT 形式でレポートを展開する代わりに、PAK 形式でレポートをパックし、PAK ファイルとして展開することもできます。この代替方法は、RPT ファイルに関連付けられているすべてのチャート、サブレポート、およびドリルダウンファイルを含めることによって、展開手順を簡略化します。API でこれを行うには、次の方法を使用してレポートファイルをパックします。

```
pack(java.lang.String filename, java.lang.String subReportPath, java.lang.String drillDownPath, java.lang.String chartPath);
```

ターゲットマシンへのデプロイメント中に、**QbReport** コンストラクタを使用して、テンプレートパラメーター内の PAK ファイル名を使用してレポートを開くだけです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.7.4 非 Windows 環境での展開

EspressReport は、レポートを生成するための Pure Java ツールです。そのため、色やフォントなどの AWT 情報を生成するためのグラフィカルライブラリは含まれていません。そのために、Java は、その情報を提供するためのシステム（レポートが生成されている）ライブラリに依存しています。したがって、AWT 情報を提供できる環境と、静的フォーマットにエクスポートするためのグラフィックスカードが必要です。

Windows 環境では、既に存在するような環境を設定するために特別な作業を行う必要はありません。GUI インタフェースはすでに実行されており、グラフィックスカードはすでに存在します。

Windows 以外の環境（Unix や Linux など）では、そうではありません。このようなシステムでは、X や何らかの形式の X を実行し、X を実行しているマシンを表示する必要があります（例えば、シェルで DISPLAY = 192.168.0.16：0.0 コマンドを実行するなど)。最高のパフォーマンスを得るには、マシン上で X を実行することをお勧めします（または、X を実行している別のマシンを指すように DISPLAY を設定する)。しかし、それが許容可能な解決策ではない場合、代替の解決策が利用可能である。

4.7.4.1 Xvfb（X バーチャルフレームバッファ）

Xvfb は､ディスプレイハードウェアと物理的な入力デバイスのないマシンで実行できる X サーバです。仮想メモリを使用してダム端末フレームバッファをエミュレートします。

このサーバの主な用途はサーバテストを目的としています｡Xvfb の色数やフォント数にも制限があります。

Xvfb はダウンロードできます（システムによって異なるバージョンが利用可能です)。Xvfb の実行後、**DISPLAY** 変数を設定して、アプリケーション（またはサーブレットエンジン）に渡す必要があります。

4.7.4.2 ヘッドレスモードの JVM 1.5+

EspressReport を使用してアプリケーションを実行するには、1.5 以上の JVM 実行時パラメータを使用できます。実行時パラメータは **java.awt.headless** であり、これを true に設定すると、Java がグラフィックス情報の X 接続を行わなくなります。たとえば、グラフを jpg として生成する chartExport というアプリケーションがある場合、通常はそれを実行するには X に接続する必要があります。ただし、次のコマンドを使用します。

```
java -Djava.awt.headless=true exportChart
```

同じアプリケーションが X に接続せずに実行されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.7.5 プラットフォーム固有の問題

4.7.5.1 AS/400

4.7.5.1.1 NAWT で実行

NAWT を使用する AS/400 環境で EspressReport を使用してアプリケーションを実行するには、以下を実行する必要があります。

- DISPLAY 環境変数をシステム名と表示番号に設定します。表示番号は、VNC サーバの表示番号です。
- XAUTHORITY 変数を**/home/VNCProfile/.Xauthority** に設定します。ここで、**VNCProfile** は VNC サーバを起動したプロファイルです。VNC サーバと Java 仮想マシンの両方が同じユーザープロファイルで実行されている場合は、XAUTHORITY を設定する必要はありません。
- Java を実行する前に Java システムプロパティを設定する

```
os400.awt.native=true
```

4.7.5.1.2 ヘッドレスモードで実行

ヘッドレスモードを使用する AS/400 環境で EspressReport を使用してアプリケーションを実行するには、以下を実行する必要があります。

- Java を実行する前に Java システムプロパティを設定する

```
java.awt.headless=true
```

- 任意の QbChart コンストラクタを呼び出す前に、コードに次のコードが含まれていることを確認してください。

```
QbChart.setForExportOnly(true);
```

4.7.5.2 Linux/Unix

4.7.5.2.1 X で実行

X を使用して Linux/Unix 環境で EspressReport を使用してアプリケーションを実行するには、以下を実行する必要があります。

- DISPLAY 環境変数を X クライアント・システム名と表示番号に設定します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.7.5.2.2 ヘッドレスモードで実行

ヘッドレスモードを使用している Linux/Unix 環境で EspressReport を使用してアプリケーションを実行するには、以下を実行する必要があります。

- Java を実行する前に Java システムプロパティを設定する

```
java.awt.headless=true
```

- 任意の QbChart コンストラクタを呼び出す前に、コードに次のコードが含まれていることを確認してください。

```
QbChart.setForExportOnly(true);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.8 付録 A. パラメータサーバ

4.8.1 パラメータサーバの作成

Report Viewer は、パラメータサーバと対話することでプッシュテクノロジをサポートします。プログラマは、**Quadbase.reportdesigner.ReportViewer.ParamServer** クラスを使用してパラメータサーバを作成して、Report Viewer がレポートをプロットするために更新されたデータを継続的に供給することができます。

クライアント側では、HTML 構文は次のとおりです。

```
<applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class" width=640 height=480>
<PARAM name="filename" value="example.rpt">
<PARAM name="ParameterServer" value="machine:portno">
</applet>
```

ブラウザが Report Viewer クラスとレポートデータを Web サーバからロードすると、Report Viewer はそのパラメータで特定された特定のマシンとポート番号に接続しようとします。セキュリティ上の理由から、信頼できないアプレットは他のマシンへの接続を開くことができません。パラメータビューアは、Report Viewer が保持するレコードを更新および操作するために、Report Viewer と対話できます。

サーバ側では、ポート番号をリッスンするサーバプログラムを作成する必要があります。このサーバは、クラス **quadbase.reportdesigner.ReportViewer.ParamServer** を使用して記述できます。
接続を開くと、サーバは Report Viewer にデータを提供します。

サーバのコンストラクタは次のとおりです。

```
public ParamServer(DataInputStream in, DataOutputStream out)
```

ソケットの入出力はサーバとの通信に使用されます。

ParamServer には、Report Viewer でレコードを操作する基本的な方法が 3 つあります。

```
int addRecord(Object record[]);
int deleteRecord(int recordNo);
int updateRecord(Object record[], int recordNo);
```

一連のレコード更新の後、最終コール

```
void repaint();
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

新しいデータを使用してレポートを再描画するように Report Viewer に要求を送信します。

次の Java プログラムでは、パラメータサーバクラスを使用して新しいレポートを生成する例を示します **ParamServer** 用の Report API は、オンライン API ドキュメントで提供されています。この例では、12 番目のレコードの 1 つのフィールドが、ランダムに選択された整数で単純に更新されます。実際のアプリケーションでは、プログラマは、パラメータサーバがデータソースと対話できるようにするコードを記述する必要があります。

```
import java.io.*;
import java.net.*;
import java.util.*;
import quadbase.reportdesigner.ReportViewer.ParamServer;

//Sample Program to update the data in EspressReport Viewer
//using Parameter Server
public class pserver extends Thread {
protected int port = 1997; //port no for report viewer to connect

protected ServerSocket listen_socket;

public static void fail(Exception e, String msg) {
System.err.println(msg + ":" + e);
System.exit(1);
}

public pserver() {
try {
listen_socket = new ServerSocket(port);
}
catch (IOException e) {
fail(e, "Exception creating server socket");
}
this.start();
}
public void run() {
try {
while(true) {
//create a thread for each connection to handle the
//request
Socket client_socket = listen_socket.accept();
Connection c = new Connection(client_socket);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
}
}
catch (IOException e) {
fail(e, "Exception while listening for connections");
}
}
public static void main(String[] args) {
new pserver();
}
}

class Connection extends Thread {
protected Socket client;
protected DataInputStream in;
protected DataOutputStream out;

public Connection(Socket client_socket) {
client = client_socket;
try { //get the input and output stream
in = new DataInputStream(client.getInputStream());
out = new
DataOutputStream(client.getOutputStream());
}
catch (IOException e) {
try {
client.close();
}
catch (IOException e2) {}
return;
}
this.start();
}

public void run() {
ParamServer paramserver;
Random random = new Random(System.currentTimeMillis());
try {
paramserver = new ParamServer(in, out);
System.out.println("Total no of record = " + paramserver.getRecordNo());

//get record 12 from the report viewer
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
Object rec[] = paramserver.getRecord(12);
for (int i=0; i < 100; i++) {
//update the third field of record 12
rec[3] = new Integer(random.nextInt() % 30);
paramserver.updateRecord(rec, 12);
//tell the report viewer to repaint //after every tenth record is updated
if (i % 10 == 0)
paramserver.repaint();
}
}
catch (IOException e) {}
finally {
try {
client.close();
}
catch (IOException e2) {}
}
}
}
```

Report Viewer は次の形式の HTML ファイルを読み込んでいました。

```
<html>
<title>Sample Report</title>
<applet code = "quadbase.reportdesigner.ReportViewer.Viewer.class" width=600 height=450>
<PARAM name="filename" value="PServer.cht">
<PARAM name="ParameterServer" value=":1997">
</applet>
</html>
```

Report Viewer が読み込む Java クラスは、レポートファイル **Pserver.rpt** の名前であり、次の行は Report Viewer が接続を開くマシンとポートを指定します。EspressManager は、更新されたレコードなどの定期的な間隔で Report Viewer に情報を提供し、Report Viewer がレポートを更新できるようにします。

4.8.1.1 変数

UPDATE_RECORD: public final static int **UPDAATE_RECORD**
INSERT_RECORD: public final static int **INSERT_RECORD**
DELETE_RECORD: public final static int **DELETE_RECORD**
GET_RECORD: public final static int **GET_RECORD**
GET_RECORDNO: public final static int **GET_RECORDNO**
GET_RECORD_DATATYPE: public final static int **GET_RECORD_DATATYPE**

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

REPAINT: public final static int **REPAINT**

OK: public final static int **OK**

ERROR: public final static int **ERROR**

4.8.1.2 コンストラクタ

public ParamServer(DataInputStream in, DataOutputStream out) throws IOException

パラメータサーバのコンストラクタ

4.8.1.3 メソッド

updateRecord

public int updateRecord(String rec[], int recorddb) throws IOException

レコードを更新します。

パラメータ:

rec – レコードが文字列配列として渡されます。文字列配列の形式については、クラス **quadbase.reportdesigner.ReportAPI.DbData** を参照してください。

recordNo – 更新されるレコード番号です。

戻り値:

成功時は **OK** が、レコード番号またはレコードタイプが違う場合は **ERROR** が返されます。

updateRecord

public int updateRecord(Object rec[], int recordNo) throws IOException

レコードを更新します。

パラメータ:

rec – レコードがオブジェクト配列として渡されます。

recordNo – 更新されるレコード番号です。

戻り値:

成功時は **OK** が、レコード番号またはレコードタイプが違う場合は **ERROR** が返されます。

addRecord

public int addRecord(String rec[])throws IOException

レコードを追加します。

パラメータ:

rec – レコードが文字列配列として渡されます。

戻り値:

成功の場合は **OK** が、レコードタイプが違う場合は **ERROR** が返されます。

addRecord

public int addRecord(Object rec[]) throws IO Exception

レコードを追加します。

パラメータ:

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

rec – レコードがオブジェクト配列として渡されます。

戻り値:

成功の場合は **OK** が、レコードタイプが違う場合は **ERROR** が返されます。

deleteRecord

public int deleteRecord(int recordNo) throws IOException

レコードを削除します。

パラメータ:

recordNo – 削除するレコード番号です。

戻り値:

成功の場合は **OK** が、レコードタイプが違う場合は **ERROR** が返されます。

repaint

public void repaint() throws IOException

Report Viewer にレポートを再描画するように通知します。

checkRecord:

public Boolean checkRecord(Object rec[])

指定されたレコードタイプが Report Viewer で使用されているレコードタイプと一致するかどうかを確認します。

パラメータ:

rec – チェックする入力レコードです。

戻り値:

レコードが一致する場合は **true** を、そうでない場合は **false** を返します。

getRecordNo:

public int getRecondNo() throws IOException

使用するレコードのレコード番号を取得します。

戻り値:

レコードのレコード番号を返します。

getDataType:

public int[] getDataType() throws IOException

使用するレコードのデータタイプを取得します。

戻り値: jdbc/Types.class に定義されているデータ型の配列。配列のサイズはレコードのフィールドの数に等しくなります。

getRecordSize:

public int getRecordSize()

レコードのフィールドの数を取得します。

戻り値:

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

レコード内のフィールドの数を返します。

getRecord:

public Object[] getRecord(int recordNo) throws IOException

レコードを取得します。

パラメータ:

recordNo – 取得するレコードの番号です。

戻り値:

レコードのオブジェクト配列が返されます。

convertRecord:

public Object[] convertRecord(String rec[])

文字列配列形式のレコードをオブジェクト配列形式に変換します。

戻り値:

レコードのオブジェクト配列が返されます。データタイプが違う場合は **null** が返されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.9 付録 B. レポートデータの取得

この章の 2 つの付録（レポートデータの取得とレポートの作成）には、デザイナを使用しないで最初からレポートを作成するための情報が含まれています。この方法は、通常、レポートを展開および保守するのをより困難にするため、お勧めしません。しかし、状況によっては、このような方法でレポートを作成する必要があるかもしれません。

以下は、**QbReport** コンストラクタの例です。多くのバリエーションがありますが、一般的な **QbReport** コンストラクタには少なくとも 3 つのパラメータが必要です。新しいレポートを作成するには、レポートタイプ、入力データソース情報、およびレポートの各列へのデータ列のマッピングを指定する必要があります。

```
QbReport(java.lang.Object parent, int reportType, IResultSet data, ColInfo[] mapping, java.lang.String template)
```

レポートテンプレートは必須ではなく（null でもかまいません）、親オブジェクトはレポートがアプレットで使用されている場合にのみ必要ですが、他の 3 つのパラメータには常に意味のある情報が含まれている必要があります。この付録では、データパラメータと、さまざまなデータソースに接続するためのメソッドについて詳しく説明します。レポートの作成では、レポートの種類、列のマッピング、および実際のレポートの作成について説明します。レポートの作成には、実践的な例も含まれています。

レポートを作成するための最初のステップは、データを取得することです。データは、いくつかの異なるタイプのソースのうちの 1 つから取り出すことができます。これらのソースは次のようになります。

- JDBC ドライバを使用して、ローカルまたはリモートデータベースからデータをフェッチします。必要な作業は、データベースへの接続方法とクエリの正確な形式に関する情報を提供することだけです。レポートは自動的に結果を取得します。
- プレーンテキスト（ASCII）形式または XML 形式のデータベースレコードを含むプレーンテキストファイルからデータを読み取ります。この形式のファイルは、ほとんどのデータベースプログラムで作成できます。
- EJB を介して、またはクラスファイルとして、独自の動的データを API 経由で渡します。
- データセットをメモリ内の配列として渡します。

以下のセクションでは、これらの各データソースとそれらからデータを取得するために使用されたメソッドを詳しく見ていきます。

4.9.1 データベースのデータ

Report API の強力な機能の 1 つは、JDBC を介してデータベースから直接データをフェッチする機能です。このアプローチでは、データベースへの接続に必要な情報と SQL 文の正確な形式を指定するだけ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

です。したがって、JDBC ドライバが利用可能である場合、プログラムは事実上すべてのデータベースに接続できます。

データベースとクエリ情報は、レポートを作成する前に **DBInfo** オブジェクトに格納する必要があります。次のコードを使用して、**DBInfo** オブジェクトをインスタンス化します。

```
DBInfo dbinfo = new DBInfo(

        "jdbc:hsqldb:help/examples/DataSources/database/woodview",       //URL
        "org.hsqldb.jdbcDriver",                                        //JDBC driver
        "myName",                                                       //Username
        "myPassword",                                                   //Password
        "select * from sales");                                         //SQL Query
```

DBInfo はインタフェース IDatabaseInfo を実装しているため、次の QbReport コンストラクタで DBInfo オブジェクトを使用できます。

DBInfo はインタフェース **IDatabaseInfo** を実装しているため、次の **QbReport** コンストラクタで **DBInfo** オブジェクトを使用できます。

```
QbReport(java.lang.Object parent, int reportType, IDatabaseInfo dbinfo, ColInfo[] mapping, java.lang.String template)
```

場合によっては、EspressReport にデータベース情報（ユーザーID、パスワード、場所、ドライバ、クエリなど）全体を渡したくない場合もあります。自分で接続し、クエリの結果セットを API に直接提供したい場合もあります。これは、生成した **ResultSet** オブジェクトから **QueryResultSet** オブジェクトを作成し、その **QueryResultSet** オブジェクトを **DBInfo** オブジェクトではなくデータソースとして渡すことによって実行できます。

```
//Create a QueryResult object from the ResultSet object resultSet
QueryResultSet queryResultSet = new QueryResultSet(resultSet);

//QueryResultSet implements IResultSet, use it in the following QbReport constructor
QbReport(java.lang.Object parent, int reportType, IResultSet data, ColInfo[] mapping, java.lang.String template)
```

上記の例では、**QueryResultSet** クラスのインスタンスが API に渡される **ResultSet** オブジェクトから作成され、このインスタンスが **QbReport** コンストラクタに渡されてレポートオブジェクトが作成されます。このシナリオでは、レポートは更新できません。レポートを更新するには、再度データベースへの接続を行い、**ResultSet** オブジェクトをレポートに渡して自分で更新する必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

データベースからのデータを使用した実際の例については、レポートの作成を参照してください。

4.9.1.1 JNDI からのデータ

EspressReport では、JNDI ソースからデータを取得することもできます。JNDI データソースは、データベースデータソースのように扱われ、同じ機能をサポートします。JNDI データソースを使用すると、異なる環境間でレポートを簡単に移行できます。両方の環境のデータソースが同じルックアップ名でセットアップされている場合、レポートは変更なしで移行できます。

JNDI データソースに接続するには、アプリケーションサーバにデータソースをデプロイする必要があります。接続を成功させるには、**INITIAL_CONTEXT_FACTORY** および **PROVIDER_URL** も指定する必要があります。Tomcat にデプロイされた JNDI データソースに接続する場合、**INITIAL_CONTEXT_FACTORY** と **PROVIDER_URL** を指定する必要はありません（ただし、EspressManager を Tomcat 環境で実行している必要があります）。

```
//create a DBInfo object with JNDI lookup name and query
String JNDIName = "java:comp/env/jdbc/TestDB";
String query = "select * from testdata";

//The environment hashtable is empty for tomcat because EspressManager is
//running inside Tomcat context. If other application server is used,
//need to set INITIAL_CONTEXT_FACTORY and PROVIDER_URL.
Hashtable env = new Hashtable();
DBInfo dbInfo = new DBInfo(JNDIName, query, env);
```

上記の例では、クラス **DBInfo** のインスタンスは、プログラムが JNDI データソースに接続してデータを取得するために必要な情報を提供します。**IDatabaseInfo** を含む次のコンストラクタを使用して、レポートを作成します。

```
public QbReport(Object parent, int reportType, IDatabaseInfo dbinfo, ColInfo[] colMap, String templateFile);
```

4.9.2 データファイルのデータ（TXT/DAT/XML）

レポートを生成するためのデータは、テキストファイルでも XML 形式のファイルでもデータファイルからインポートすることができます。レポートデータファイル（**.dat** 拡張子付き）は、それぞれの行が 1 つのレコードを表し、データ型とフィールド名をそれぞれ含む最初の 2 行を除いたプレーンテキストファイルです。次に、4 つのレコードを含むデータファイルの例を示します。各レコードには 3 つのフィールドがあります。最初の行は各フィールドのデータ型を指定し、2 行目はフィールド名を指定します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
string, date, decimal
Name, Day, Volume
"John", "1997-10-3", 32.3
"John", "1997-4-3", 20.2
"Mary", "1997-9-3", 10.2
"Mary", "1997-10-04", 18.6
```

コンマ文字の使用はオプションですが、ほとんどのデータベースプログラムは、テキスト形式の表からレコードをエクスポートするときに、この形式のファイル（最初の 2 行を除く）を出力できます。その結果、データベース用の JDBC ドライバがなくても、すぐにレポートを作成することができます。データベースのデータをテキスト形式でエクスポートするだけで、プログラムで Report API を使用してデータを読み取ることができます。

XML 形式の場合、仕様は次のとおりです。

```
<EspressData>

        <DataType>string</DataType>
        <DataType>date</DataType>
        <DataType>decimal</DataType>

        <FieldName>Name</FieldName>
        <FieldName>Day</FieldName>
        <FieldName>Volume</FieldName>

        <Row>
                <Data>John</Data>
                <Data>1997-10-3</Data>
                <Data>32.3</Data> </Row>

        <Row>
                <Data>John</Data>
                <Data>1997-4-3</Data>
                <Data>20.2</Data> </Row>

        <Row>
                <Data>Mary</Data>
                <Data>1997-9-3</Data>
                <Data>10.2</Data> </Row>
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        <Row>
                <Data>Mary</Data>
                <Data>1997-10-4</Data>
                <Data>18.6</Data> </Row>

</EspressData>
```

XML データの詳細については XML エンコーディングを参照してください。

使用するテキストファイルを指定するのは非常に単純です。次のコンストラクタを使用して、変数 **dataFile** をデータファイルのパス（相対パスまたはフルパス）で置き換えます。サーバに接続する場合は、相対パスは EspressReport のルートディレクトリを基準にする必要があります。それ以外の場合、パスは現在の作業ディレクトリからの相対パスになります。

```
QbReport(java.lang.Object parent, int reportType, java.lang.String dataFile, ColInfo[] mapping, java.lang.String template)
```

テキストファイルのデータを使用する例についてはレポートの作成を参照してください。

4.9.3 XML データソースのデータ

上記に加えて、EspressReport では、データを検索して XML ファイルをクエリすることができます。XML データは事実上どんなフォーマットでも構いませんが、XML データと共に DTD ファイルまたは XML スキーマを指定する必要があります。次のコードは、XML クエリを設定する方法を示しています。

```
//Set up the XML Data Source
String xmlfilename = "Inventory.xml";
String dtdfilename = "Inventory.dtd";
String xmlcondition = "/Inventory/Category/Product/ProductID < 45";

XMLFieldInfo[] fields = new XMLFieldInfo[5];
fields[0] = new XMLFieldInfo(new String[] {"Inventory", "Category", "Product"}, "ProductID");
fields[0].setAttributeDataType(DTDDataType.INT);

fields[1] = new XMLFieldInfo(new String[] {"Inventory", "Category", "Product", "ProductName"});

fields[2] = new XMLFieldInfo(new String[] {"Inventory", "Category", "Product", "UnitPrice"});
fields[2].setElementDataType(DTDDataType.DOUBLE);

fields[3] = new XMLFieldInfo(new String[] {"Inventory", "Category", "Product", "UnitsInStock"});
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
fields[3].setElementDataType(DTDDataType.INT);

fields[4] = new XMLFieldInfo(new String[] {"Inventory", "Category", "Product", "ShipDate"});
fields[4].setElementDataType(DTDDataType.DATE);
fields[4].setDateFormat(XMLDataTypeUtil.YYYY_MM_DD);

XMLFileQueryInfo xmlInfo = new XMLFileQueryInfo(xmlfilename, fields, xmlcondition, fields, dtdfilename, false, null);
```

XMLFieldInfo インスタンスは、2 つのコンストラクタのいずれかを使用して作成されます。xml フィールドを選択するために使用される最初のコンストラクタには、1 つのパラメータが含まれています。このコンストラクタでは、ターゲットフィールドにつながる階層の各 xml タグを指定する String 配列を渡す必要があります。上記の例では、フィールド[1-4]は最初のコンストラクタを使用して作成されます。XML 属性を選択するために使用される 2 番目のコンストラクタには、2 つのパラメータが含まれています。String 配列パラメータに加えて、2 番目のコンストラクタには、属性名に別の String も必要です。上記の例では、**ProductID** は **Product** の属性であるため、このコンストラクタを使用してフィールド[0]が作成されています。

また、String 以外のフィールドについては、データ型を明示的に設定する必要があることに気づいたかもしれません。**XMLFileQueryInfo** インスタンスを作成したら、次のコンストラクタを使用して **QbReport** を作成できます。

```
public QbReport(Object parent, int reportType, XMLFileQueryInfo xmlInfo, ColInfo[] colMap, String templateFile, boolean sideBySideLayout);
```

XML データをデータソースとして使用する場合、XML ファイルの代わりに XML ストリームを渡すこともできます。XML ストリームを渡すには、XML データファイル名の代わりに XML データを含むバイト配列を渡します。

上記の例では、**XMLFileQueryInfo** コンストラクタでバイト配列（**xmlByteArray** というバイト配列など）を介して XML ストリームを渡すことができます。

```
XMLFileQueryInfo xmlInfo = new XMLFileQueryInfo(xmlByteArray, fields, xmlCondition, fields);
```

XML ファイルソースの実例についてはレポートの作成を参照してください。

4.9.4 メモリ内の配列に渡されたデータ

API を使用すると、入力データを配列としてメモリに直接渡すことができます。これはインタフェース **IResultSet**（**quadbase.reportdesigner.util** パッケージで定義されています）によって可能になります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

このインタフェースはテーブル形式でデータを読み取るために使用され、JDBC 結果セットで使用される **java.sql.ResultSet** インタフェースと非常によく似ています（インタフェースよりもはるかに簡単です）。ユーザーは **IResultSet** の独自の実装を提供することも、EspressReport が提供する **IResultSet** を使用することもできます。最も簡単な実装は **DbData** クラスです（**IResultSet** 実装を提供する他のクラスは **QueryResultSet** と **StreamResultSet** です）。レポートに必要なすべてのデータをメモリに格納できる場合は、1 行のコードで配列を **DbData** の引数として渡すだけで済みます。DbData には 3 つのコンストラクタがあります。

```
DbData(java.lang.String s)
        Construct DbData by parsing the data value argument from an HTML page

DbData(java.lang.String[] fieldName, java.lang.Object[][] records)
        Construct a new DbData class

DbData(java.lang.String[] dataType, java.lang.String[] fieldName, java.lang.String[][] records)
        Construct a new DbData class
```

このアプローチを使用してデータを渡すときに、サマリブレークまたはクロス集計レポートの場合、データが適用可能であり、すでにグループ化されてソートされていることを確認して、パフォーマンスを向上させてからレポートを生成してください。

この例では、次のコンストラクタを使用します。

```
public DbData(String dataType[], String fieldName[], String records[][])
```

ここでは、最初の引数はデータ型（データファイルの最初の行）を表し、2 番目の引数はフィールド名（2 行目）を表します。3 番目の引数、**records [] []**は、レコードの配列を提供します。レコード[i]は i 番目のレコードです。以下は、どのように動作するかを示しています。

```
String dataType[] = {"varchar", "decimal"};
String fieldName[] = {"People", "Sales"};
String records[][] = {{"Peter", "93"}, {"Peter", "124"},
                      {"John", "110"}, {"John", "130"},
                      {"Mary", "103"}, {"Mary", "129"}};

DbData data = new DbData(dataType, fieldName, records);
```

レポートを作成するには、次の **QbReport** コンストラクタを使用します。

```
public QbReport(Object parent, int reportType, IResultSet data,
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                ColInfo[] colMap, String templateFile);
```

ソート済みのデータを EspressReport に渡すと、パフォーマンスが大幅に向上します。データがすでにソートされている場合は、コンストラクタでフラグを **true** に設定して、EspressReport によるデータの並べ替えを回避できます。コンストラクタは次のとおりです。

```
public QbReport(Object parent, int reportType, IResultSet data,
                ColInfo[] colMap, String templateFile, boolean sideBySideLayout, boolean
isDataSorted);
```

特に、複数のブレークレベルを持つサマリブレークレポートを生成する場合は、これが重要です。メモリ内のデータが既にソートされている場合（たとえば、データベースエンジンによってソートされている場合)、この機能を利用してこの機能を使用して、レポートエンジンが不要な内部データ構造を構築したりソートを実行することを防ぎ、大量のメモリとプロセッサリソースの消費を抑えることができます。したがって、性能を大幅に向上させることができます。

メモリ内の配列に渡されたデータを使用する実際の例については、レポートの作成を参照してください。

4.9.5 カスタム実装で渡されるデータ

柔軟性を最大限に高めるために、必要な方法でデータセットを取得して準備し、レポートエンジンに渡すことができます。クラスファイルをデータソースとして渡すには、クラスファイルで **IDataSource** インタフェースを実装する必要があります。**IDataSource** を実装する簡単な例を以下に示します。

```
public class CustomClassData extends Applet implements IDataSource {

        //Setting DbData for passing data as arguments
        String dataType[] = {"string", "String", "double"};
        String fieldName[] = {"Destination", "Time", "Price"};
        String records[][] = {{"Mayfair", "13:43", "3.50"},
                {"Bond Street", "13:37", "3.75"},
                {"RickmansWorth", "13:12", "5.25"},
                {"Picadilly", "13:24", "3.00"}};
        DbData data = new DbData(dataType, fieldName, records);

        public IResultSet getResultSet()
        {
                return data; }

}
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フルソースコード

上記の例では、データ（**DbData** インスタンス）が作成され、メモリに格納されます。**getResultSet**() メソッドが呼び出されると、IResultSet を実装する **DbData** オブジェクトが返されます。この方法でデータを作成する必要はないことに注意してください。**IResultSet** を実装するオブジェクトを返すことができる限り、任意のデータソースからデータを取得できます。レポートを作成するには、次のコンストラクタを使用します。

```
QbReport(java.lang.Object parent, int reportType, int fileType, java.lang.String filename, ColInfo[] mapping, java.lang.String template)
```

カスタムクラスファイルの場合は、**fileType** を **QbReport.CLASSFILE** に、ファイル名をクラスファイルの名前に設定します。

データソースとして独自のクラスファイルを渡していて、EspressManager を使用している場合、クラスファイルは EspressManager の CLASSPATH からアクセス可能でなければならないことに注意してください。

パラメータ化されたクラスファイルをレポートのデータソースとして渡すこともできます。パラメータは、実行時にユーザーから取得され、データがフェッチされ、レポートの生成に使用されます。パラメータ化されたクラスファイルの例を次に示します。これは、特定のクラスファイルデータソースを持つ Report Designer を開くで使用されているのと同じファイルです。

```
public class ParamClassFile implements IParameterizedDataSource
{
	String url = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
	String driver = "org.hsqldb.jdbcDriver";
	String username = "sa";
	String password = "";

	//	Specify what the parameter properties are.
	public IQueryInParam[] getParameters()
	{
		SimpleQueryInParam[] params = new SimpleQueryInParam[2];
		//		SimpleQueryInParam(name, promptString, mapToColumn, TableName, ColumnName, dataType, defaultValue, actualValue)
		params[0] = new SimpleQueryInParam("price", "Max price:", false, null, null, Types.DOUBLE, new Double(500.00), null);
		params[1] = new MySimpleQueryInParam("popular", "Popular Items Only:", false, null, null, Types.VARCHAR, new String("NO"), null);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            return params;
        }

        private class MySimpleQueryInParam extends SimpleQueryInParam implements IQueryParamValuesProvider
        {
            public MySimpleQueryInParam(String paramName, String promptName, boolean mapToColumn,
                                String tableName, String columnName, int sqlType, Object defaultValue, Object value)  {
                    super(paramName, promptName, mapToColumn, tableName, columnName, sqlType, defaultValue, value);
            }
            public Vector getSelectionChoices() {
                    Vector choices = new Vector();
                    choices.add(new String("Yes"));
                    choices.add(new String("No"));
                    return choices;
            }
        }

        //       Specify what data is to be returned
        public IResultSet getResultSet(IQueryInParam[] params)
        {
            double price = 500.00;
            int units = 999;

            if ((params != null) && (params.length >= 1))
            {
                Object obj = params[0].getValue();
                if ((obj != null) && (obj instanceof Double)) price = ((Double)obj).doubleValue();
                obj = params[1].getValue();
                if((obj != null) && (obj instanceof String) && ((String)obj).equalsIgnoreCase("yes")) units = 15;
            }

            try{
                Class.forName(driver);
                Connection conn = DriverManager.getConnection(url, username, password);
                String query = "SELECT ProductName, Description, UnitPrice, UnitsInStock FROM Products WHERE UnitPrice < "
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                    + price + " AND UnitsInStock < " + units;
            Statement stmt = conn.createStatement();

            ResultSet rs = stmt.executeQuery(query);
            QueryResultSet qry = new QueryResultSet(rs);

            return qry;
        } catch(Exception e) {
            e.printStackTrace();
        }
        return null;
    }

}
```

フルソースコード

パラメータ化されたクラスファイルは、**IParameterizedDataSource** を実装する必要があります。上記の例では、WoodView HSQL データベースからデータを取得するため、クラスパスにデータベース HSQL JDBC ドライバ（**hsqldb.jar**）を追加する必要があります。ドライバは**<EspressReportInstall>/lib** ディレクトリにあります。

クエリには 2 つのパラメータが含まれています。最初のパラメータを使用すると、クエリ結果の最大価格を設定できます。パラメータは **double** 型で、デフォルトは 500.00 に設定されています。2 番目のパラメータはカスタムパラメータです。これは、人気のあるアイテムのみを表示するかどうかを決定する **yes** または **no** オプションをユーザーに与えます。カスタムパラメータを作成するには、**SimpleQueryInParam** クラスを拡張し、**IQueryParamValuesProvider** インタフェースを実装し、**getSelectionChoices**()メソッドを上書きするクラスを作成します。ユーザーが **Yes** を選択した場合、**getResultSet**()で UnitsInStock が 15 未満の項目のみが返されます。

これまでと同じコンストラクタを使用して、このパラメータ化されたクラスを使用してレポートを作成できます。パラメータ化されたクラスファイルをデータソースとして使用する場合は、列に直接マップすることはできません（つまり、**SimpleQueryInParam** コンストラクタでテーブル名と列名を設定することはできません）。パラメータを列にマップするには、上記のようなカスタムパラメータを使用する必要があります。ただし、**getSelectionChoices()**メソッドは次のようになります。

```
public Vector getSelectionChoices() {
    System.out.println("getSelectionChoices called");
    try {
        Class.forName("org.hsqldb.jdbcDriver");
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            String url = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
            Connection conn = DriverManager.getConnection(url, "sa", "");
            Statement stmt = conn.createStatement();
            String query = "SELECT DISTINCT " + getColumnName() + " FROM " +
getTableName();
            ResultSet rs = stmt.executeQuery(query);
            Vector v = new Vector();

            while (rs.next()) {
                switch (getSqlType()) {
                case Types.INTEGER:
                    v.add(new Integer(rs.getInt(1)));
                    break;

                case Types.VARCHAR:
                    v.add(rs.getString(1));
                    break;
                }
            }

            stmt.close();
            conn.close();

            return v;
        } catch (Exception ex) {
            ex.printStackTrace();
        }
        return null;
}
```

フルソースコード

この例では、パラメータを列にマッピングするだけでなく、複数値のパラメータも使用します。プロセスは単一の値パラメータを使用するのと似ていますが、コードが標準的なものから逸脱している箇所があります。詳細は、上記のソースコードのコメントを参照してください。また、API のパラメータの使用に関する詳細は、パラメータ化されたレポートを参照してください。

カスタムクラスファイルから渡されたデータを使用した実例については、レポートの作成を参照してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.9.6 Enterprise Java Bean（EJB）のデータ

entity Bean から直接データをクエリできるようにすることで、EJB データソースからレポートにデータを渡すことができます。EJB をデータソースとして追加するには、最初に EJB をアプリケーションサーバにデプロイする必要があります。また、適切なスタブクラスを含むクライアント JAR ファイルをクラスパスに追加する必要があります（または EspressManager と連携して API を使用する場合は EspressManager バッチファイルの**-classpath** 引数）。

EJB をデータソースとして使用してレポートを作成するには、まず EJB に接続するための情報を含む **EJBInfo** オブジェクトを構築する必要があります。コンストラクタを以下に示します。

```
public EJBInfo(java.lang.String jndiName,
                java.lang.String homeName,
                java.lang.String remoteName,
                java.lang.String selectedMethodName,
                java.lang.Object[] selectedMethodParamVal)
```

例えば、次の行は ProductEJB データソースに接続する EJBInfo インスタンスを作成します。

```
Object[] vals = new Object[1];
vals[0] = new String("Printer");

EJBInfo ejbInfo = new EJBInfo("ProductEJB", "ejb.ProductHome",
                              "ejb.Product", "findByCategory", vals);
```

最後に、次のコンストラクタを使用して **QbReport** オブジェクトを作成します。

```
public QbReport(Object parent, int reportType, EJBInfo ejbInfo, ColInfo[] mapping, String template);
```

4.9.7 SOAP データソースのデータ

EspressReport では、SOAP サービスからデータを取得できます。SOAP データソースを使用するには、必要なすべての情報を含む WSDL ファイルの場所を指定する必要があります。場所は、サーバ上の絶対パスか、EspressReport インストールディレクトリまたは URL からの相対パスのいずれかです。

SOAP データソースのコンストラクタは次のようになります。

```
SOAPQueryFileInfo(String wsdlURI, QName serviceName, String portName, String operationName,
XMLFieldInfo[] xmlFieldInfo, SOAPParam[] parameters)
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

wsdlURI は WSDL ファイルへの絶対パスまたは相対パスです。QName 型の serviceName は SOAP サービスの名前です。次のコンストラクタが使用されます。

```
QName(String namespaceURI, String localPart),
```

namespaceURI は URI 名前空間（中括弧内の **serviceName** 文字列の一部）であり、**localPart** はサービス名（残りの文字列）です。たとえば、次の **servicename {http://www.webservicex.net} WeatherForecast** の場合、**http://www.webservicex.net** は **namespaceURI** で、**WeatherForecast** はサービス名です。

SOAPQueryFileInfo コンストラクタの **portName** および **operationName** は、ポートおよび操作の名前です。**XMLFieldInfo** 配列は XML データソースと同じです（XML データソースのデータを参照）。最後に、**SOAPParam** フィールドはパラメータを扱い、次のコンストラクタを使用します。

```
SOAPParam(String paramName, int sqlType, String paramPrompt, Object defaultValue, Object value, boolean alwaysUseDefault),
```

paramName はパラメータ名、**sqlType** はデータ型のパラメータを表します。これは **java.sql.Types** の定数です。**paramPrompt** はパラメータプロンプト文字列、**defaultValue** はパラメータのデフォルト値、値はパラメータ値、**alwaysUseDefault** は **boolean** 型で、デフォルト値を常に使用するかどうかを指定します。つまり、このパラメータの値は固定され、ユーザーにはこのパラメータの値を入力する必要はありません。

QbReport/QbChart オブジェクトを作成するには、以下のコンストラクタを使用します。

```
QbReport(Object parent, int reportType, SOAPQueryFileInfo soapInfo, ColInfo[] mapping, String template, Properties props)

QbChart(Applet applet, int dimension, int chartType, SOAPQueryFileInfo soapInfo, boolean doTransposeData, int[] transposeCol, IColumnMap cmap, String template)
```

フルソースコード

4.9.8 複数のデータソースからのデータ

EspressReport は、複数のデータソースを併合する機能も提供します。**DataSheet** オブジェクトは、データファイル、データベース、**IResultSet** などのデータソースの任意の組み合わせから作成できます。**QbReport** コンストラクタを使用すると、**DataSheet** オブジェクトの配列からレポートオブジェクトを作成できます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次のプログラム例は、上記のセクションの例のデータを結合する方法を示しています。

```
//Declaration of DataSheet object to be used to merge data
DataSheet dataSheet[] = new DataSheet[3];

//Declaration For Database (Data from Database section)
DBInfo dbinfo = new DBInfo("jdbc:hsqldb:help/examples/DataSources/database/woodview",
"org.hsqldb.jdbcDriver", "sa","","select * from Orders");

//Declaration For Data Passed in from Memory (Data Passed in from Memory section)
String dataType[] = {"varchar", "decimal"};
String fieldName[] = {"People", "Sales"};
String records[][] = {{"Peter", "93"}, {"Peter", "124"},
                      {"John", "110"}, {"John", "130"},
                      {"Mary", "103"}, {"Mary", "129"}};
DbData data = new DbData(dataType, fieldName, records);

//Create DataSheet from data file (Data from a Text File section)
//DataSheet(Applet applet, String dataFile)
dataSheet[0] = new DataSheet(this, "../data/1_A_7_TextData.dat");

//Create DataSheet from database (Data from a Database section)
//DataSheet(Applet applet, IDatabaseInfo dbInfo)
dataSheet[1] = new DataSheet(this, dbinfo);

//Create DataSheet from IResultSet  (Data Passed in from Memory section)
//DataSheet(Applet applet, IResultSet data)
dataSheet[2] = new DataSheet(this, data);
```

レポートの作成に、次の QbReport コンストラクタを使用します。

```
QbReport(java.lang.Object parent, int reportType, DataSheet[] dataSheet, ColInfo[] mapping, java.lang.String template)
```

DataSheet オブジェクトを作成した後は、DataSheet API を使用して変更（行の値の追加、削除、または更新）することができます。**IResultSet** からデータをマージすると、データを更新できません。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.10 付録 C. レポートの作成

新しいレポートを作成するには、レポートタイプ、入力データソース情報、およびレポートの各列へのデータ列のマッピングを指定する必要があります。この付録では、さまざまなレポートの種類と列のマップに使用される方法について説明します。この付録には、いくつかの完全に機能的な例もあります。

特に指定のない限り、すべての例では、**<EspressReportInstall>/help/examples/DataSources/database** ディレクトリにある Woodview HSQL データベースを使用しています。例を実行するには、データベース HSQL JDBC ドライバ（**hsqldb.jar**）をクラスパスに追加する必要があります。ドライバは<EspressReportInstall>/lib ディレクトリにあります。

基本的なレポートタイプには、シンプルカラムナ、サマリブレーク、クロス集計、マスタ&詳細、メーリングラベルレポートの 5 つがあります。すべてのレポートには、レポートの種類に固有のプロパティと同様に定義された列マッピングがあります。

列のマッピングは、**ColInfo** クラス（quadbase.reportdesigner.util パッケージにあります）を使用して API で実行されます。**ColInfo** 配列オブジェクトを宣言することによって、列マッピングを定義します。配列内の各要素は、列のそれぞれの列を表します。また、必要なレポートの種類に応じて、**ColInfo** 配列オブジェクトの各要素に固有のプロパティを割り当てて、レポートを作成することもできます。

QbReport クラスには、多数のコンストラクタが含まれています。しかし、ほとんどのコンストラクタは 3 つのパラメータセットに分解できます。最初のセットは親オブジェクトだけです。アプレットのコンテキストでレポートを使用する場合は、親オブジェクトを設定する必要があります。そうでない場合は、親オブジェクトを設定する必要があります。パラメータの第 2 セットは常に必須であり、構成するレポートのタイプ、データソース、およびマッピングを決定します。このグループのパラメータに関する 2 つの付録には大量の情報があります。最後に、3 番目のパラメータセットはオプションとプロパティです。たとえば、ほとんどのコンストラクタにはテンプレートを指定するオプションがありますが、これは必須ではありません。一部のオプションは特定のレポートタイプ固有の機能です。たとえば、boolean パラメータの **sideBySideLayout** は、マスタ&詳細レポートを作成する場合にのみ関連します。いくつかのコンストラクタには、複数のプロパティを設定して 1 つのプロパティオブジェクトにグループ化できる **Properties** パラメータが含まれています。さまざまなオプションとプロパティの詳細については、**APIDocs** を参照してください。

Java でオブジェクトを作成することは、リソースを集中的に使用します。常に多くのオブジェクトを作成しないことをお勧めします。リソースを節約する 1 つの方法は、**QbReport** オブジェクトを可能な限り再利用することです。たとえば、多くのユーザーが Web サイトでシンプルカラムナレポートを要求した場合、要求を受け取ったときに新しい **QbReport** オブジェクトを作成するのではなく、**QbReport** オブジェクトを 1 つ（または限られた数）作成し、特定の要求ごとにレポートのデータと属性を変更することで、再利用できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.10.1 シンプルカラムナ

Simple Columnar（シンプルカラムナ）レポートは、EspressReport でサポートされている中で最も基本的なレポートです。このレポートは単一テーブルのカラムナデータを表示します。グルーピングまたはブレークは含みません。

次に一例を示します。

Product Type	Product Name	Price
12	Chair	$20
12	Table	$30
14	Cabinet	$20
14	Table	$50

4.10.1.1 列マッピング

シンプルカラムナレポートは、データソースに対して **ColInfo** 配列オブジェクトを定義するだけです。たとえば、以下のデータセットが与えられているとします。

```
String dataType[] = {"String", "String", "String"};
String fieldName[] = {"Product Type", "Product Name", "Price"};
String records[][] = {{"12", "Chair", "$20"},
                      {"12", "Table", "$30"},
                      {"14", "Cabinet", "$20"},
                      {"14", "Table", "$50"}};
```

列マッピングは、データソースの列がレポートにマップされるように設定されます。したがって、レポートの列 0 は、データソースの列 0、レポートの列 1、データソースの列 1 などに対応します。したがって、次のレポートが作成されます。

Product Type	Product Name	Product Price
12	Chair	$20
12	Table	$30
14	Cabinet	$20
14	Table	$50

マッピングおよびテーブルヘッダー名設定後のレポート

前述のレポートを生成するには、API を使用して次の列マッピングを設定する必要があります。

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(0);
colInfo[0].setName("Product Type");
colInfo[1] = new ColInfo(1);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
colInfo[1].setName("Product Name");
colInfo[2] = new ColInfo(2);
colInfo[2].setName("Product Price");
```

上記のコードでは、元のデータに定義されているヘッダー名を保持するのではなく、列の名前もそれぞれ製品タイプ、製品名および製品価格に変更されます。

列マッピングは、順序通りである必要はなく、データと同じ順序に従います。また、希望する列を選択して選択することもできます。たとえば、複数の列を持つデータソースがある場合、次のコードを記述できます。

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(2);
colInfo[0].setName("Product Name");
colInfo[1] = new ColInfo(7);
colInfo[1].setName("Units Sold");
colInfo[2] = new ColInfo(4);
colInfo[2].setName("Unit Price");
```

上記の例では、レポートの列 0 はデータソースの列 2 にマップされ、レポートの列 1 はデータソースの列 7 にマップされ、レポートの列 2 はデータソースの列 4 にマップされます。列のマッピングを設定するだけでなく、各列のテーブルヘッダーも設定しています。

4.10.1.2 レポートの作成

シンプルカラムナレポートを作成するのは比較的簡単です。すでに **ColInfo** 配列を設定する方法と、前のセクションでデータを取得する方法についてはすでに説明しました。次のコードは、**シンプルカラムナ**レポートを作成する方法を示しています。

```
//Data passed in an array in memory
DbData data = new DbData(dataType, fieldName, records);

//Set Column Mapping
ColInfo colInfo[] = new ColInfo[3];
colInfo[0] = new ColInfo(0);
colInfo[0].setName("Product Type");
colInfo[1] = new ColInfo(1);
colInfo[1].setName("Product Name");
colInfo[2] = new ColInfo(2);
colInfo[2].setName("Product Price");

//Create Report
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
QbReport report = new QbReport
                  (parent, //Parent
                  QbReport.COLUMNAR, //Type of Report
                  data, //Data
                  colInfo, //Column Mapping - use the column mapping in the template
                  "SimpleColumnar.rpt"); //Template Name
```

フルソースコード
エクスポートされた未フォーマットの結果
エクスポートされたフォーマット済の結果

エクスポートされた未フォーマットの結果は、書式設定やテンプレートなしで初めてレポートを作成したときのレポートの表示方法を示します。エクスポートされたフォーマット済の結果には、テンプレートを適用した後の結果が表示されます。

この例は最初からレポートを作成しますが、テンプレートを適用することをお勧めします。ソースコードを見ると、QbReport コンストラクタの後に大きなコメントセクションがあります。このセクションでは、テンプレートの適用とまったく同じ方法で、レポートの外観を変更します。コードの記述には、位置と大きさを測るのに役立つ視覚的補助がないので、はるかに多くの労力が必要なのは想像に難くありません。

テンプレートを適用すると、テンプレートに格納されている列マッピングを使用することができます。上記の例では、コンストラクタ内の **colInfo** を **null** に置き換えることができ、それは同じレポートを生成します。これは、テンプレート内の列マッピングがまったく同じであるためです。**ColInfo** を設定してテンプレートを適用すると、作成したレポートは指定した **ColInfo** を使用し、テンプレートの列マッピングを無視します。

4.10.2 サマリブレーク

サマリブレークレポートの場合は、行ブレークを定義する必要があります。行ブレークは、レポートを中断してサマリー列を挿入する時期を定義します。このレポートは、ブレーク列のデータが変更されるたびに、本質的に **Broken** 状態になります。行が非数値であっても、行ブレークとして指定されていない列については、集計を設定する必要があります（集計を **NONE** に設定する）。これは、集計に基づいて列要約を生成できるようにするためです。たとえば、以下のデータセットが与えられているとします。

Order#	Product	Quantity
12	Chair	2
12	Table	3
14	Cabinet	2
14	Table	5

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

元データ

このデータに対して列 0（**Order#**）に**行ブレーク**（**Row Breaks**）、列 1（**Product**）の集計（**Aggregation**）には **NONE** を、列 2（**Quantity**）の集計には **SUM** を適用した場合、次のようなレポートを得ることができます。

Order#	Product	Quantity
12	Chair	2
	Table	3
		5
14	Cabinet	2
	Table	5
		7

行ブレークおよび集計設定後のレポート

4.10.2.1 列マッピング

サマリブレークレポートの列マッピングを設定するには、**シンプルカラムナ**レポートよりもさらに 2 つのプロパティを設定する必要があります。まず、行グループ列に対して行ブレーク列を指定する必要があります。第 2 に、行ブレーク列ではない列は、集約タイプを指定する必要があります。たとえば、前のセクションでのレポートを生成するには、次のコード行を設定する必要があります。

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(0);
colInfo[0].setRowBreak(true);
colInfo[1] = new ColInfo(1);
colInfo[1].setAggregation(false, ColInfo.NONE);
colInfo[2] = new ColInfo(2);
colInfo[2].setAggregation(false, ColInfo.SUM);
```

ここでも、最初の手順は、レポートの列とデータソースの列を対応付けることです。次に、列 0 が行ブレーク列に設定されます。列 1 と列 2 は行ブレーク列として設定されていないため、**NONE** であっても集計を設定する必要があります。**setAggregation** メソッドには 2 つのパラメータがあり、最初のパラメータはレポートで列集計を使用するかどうかを決定します。列集計を使用すると、データの個々の行が非表示になり、集計だけが表示されます。最初のパラメータを true に設定すると、列の集計を使用するために、もう一方の非行ブレーク列も設定する必要があります。2 番目のパラメータは集計のタイプを決定します。**ColInfo** クラスの利用可能なすべての集計のリストを次に示します。

AVG	**MAX**	**STDDEV**
COUNT	**MEDIAN**	**SUM**
COUNTDISTINCT	**MIN**	**SUMSQUARE**

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

FIRST	NONE	VARIANCE
LAST		

4.10.2.2 レポートの作成

次の例は**サマリブレーク**レポートを作成するコードです。

```
ColInfo[] colInfo = new ColInfo[5];
colInfo[0] = new ColInfo(1);
colInfo[0].setRowBreak(true);
colInfo[1] = new ColInfo(9);
colInfo[1].setRowBreak(true);
colInfo[2] = new ColInfo(4);
colInfo[2].setAggregation(true, ColInfo.SUM);
colInfo[3] = new ColInfo(5);
colInfo[3].setAggregation(true, ColInfo.SUM);
colInfo[4] = new ColInfo(6);
colInfo[4].setAggregation(true, ColInfo.SUM);

QbReport report = new QbReport
                    (parent, //parent
                    QbReport.SUMMARY, //Sumamry Break    Report
                    "sample.dat", //filename
                    colInfo, //column information
                    null);
try {
        //Apply template including scripts and formula
        report.applyTemplate("SummaryBreak.rpt", true);
} catch (Exception e) {
        e.printStackTrace();
}
```

フルソースコード
エクスポートされた未フォーマットの結果
エクスポートされたフォーマット済の結果

エクスポートされた未フォーマットの結果は、書式設定やテンプレートなしで初めてレポートを作成したときのレポートの表示方法を示します。エクスポートされたフォーマット済の結果には、テンプレートを適用した後の結果が表示されます。

わかりやすい変化として挙げられるのは、列マッピングが前の例とは異なることです。これは、テキストファイルをデータソースとして使用する場合によく使用されます。この例では、レポートの列 0 をテ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

キストファイルの列 1、レポートの列 1 をテキストファイルの列 9 などにマッピングしています。

また、この例では、テンプレートを使用してレポートの外観を整形して、過剰な量のコードを書く必要がなくなりました。この例では、コンストラクタにテンプレートを追加することはできません。デフォルトでは、テンプレートをコンストラクタに追加すると、数式とスクリプトがインポートされないためです。式とスクリプトの両方をインポートする方法は、**QbReport** オブジェクトを作成した後に **applyTemplate** メソッドを呼び出すことです。渡す必要がある 2 つの引数は、テンプレートのファイルパスと、式とスクリプトをインポートするかどうかを指定するブール値です。

サマリブレークレポートを作成する別の方法は、データソースがすでにソートされていることを指定することです。こうすることで、EspressReport は、データベースデータと同じ方法でサマリブレークレポートを作成します。この機能は、テキスト、XML、およびクラスデータを含むすべてのデータソースで使用できます。これを行うには、boolean パラメータ **isDataSorted** を **true** に設定して **QbReport** コンストラクタを呼び出します。

```
new QbReport(parent, reportType, data, mapping, template, sideBySideLayout, isDataSorted);
```

4.10.3 クロス集計

クロス集計レポートは、列状のデータを行列形式で表示および要約するレポート形式です。クロス集計レポートは、スプレッドシートに似ています。行と列の両方が要約され、多次元データを 2 次元形式で表示することができます。

行ブレーク（**Row Breaks**）を定義するだけでなく、**列ブレーク**（**Column Break**）として機能する列と、**列ブレーク値**（**Column Break Value**）として機能する列を指定する必要があります。列ブレーク列は、選択した列の一意のエントリごとに個別の列を含むテーブルヘッダーに含まれ、列ブレーク値列はレポートに要約されるフィールドを表します。また、列ブレーク値列の集計タイプを指定する必要があります。たとえば、以下のデータセットが与えられているとします。

Region	Product	Total Sales
East	Chair	14500
Midwest	Chair	13250
South	Chair	15252
East	Table	10550
Midwest	Table	9150
South	Table	11250

元のデータ

列ブレークとして列 0（**Region**）が割り当てられ、列ブレークとして列 1（**Product**）が割り当てられ、**SUM** 集計で列ブレーク値として割り当てられた列 2（**Total Sales**）がある場合、次のようになります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Region	Chair	Table	Total Sales
East	14500	10550	25050
Midwest	13250	9150	22400
South	15252	11250	26502
	43002	30950	73952

列ブレークおよび列ブレーク値集計適用後のレポート

この例は、クロス集計レポートで頻繁に表示されるいくつかの特性を示しています。異なる製品名が列見出しになるように変換されていることに注意してください。つまり、レポートの列数はデータソースに応じて変化します。また、数値は、製品と地域の特定の組み合わせの合計売上を表します。データの右と下のセルは集計セルです。各列の末尾の **Total Sales** は各地域の **Total Sales** を合計し、各列の下部の値は各製品の **Total Sales** を合計します。

4.10.3.1 列マッピング

前述のレポートを生成するには、API を使用して次の列マッピングを設定する必要があります。

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(0);
//use this column as the row break. The number of
//new groups would be equal to the number of unique
//fields present.
colInfo[0].setRowBreak(true);

colInfo[1]=new ColInfo(1);
//use this column as the column break. The number of
//new columns would be equal to the number of unique
//column fields present.
colInfo[1].setColumnBreak(true);

colInfo[2] = new ColInfo(2);
//use this column as the value fields for the
//newly created chair and table columns
colInfo[2].setColumnBreakValue(true);
//aggregate by adding up the individual fields to
//get the summary field.
colInfo[2].setAggregation(ColInfo.SUM);
```

上記のコードでは、列の名前は、十分なデータソースからの列の元の名前として設定されていません。

上記の例では、1 つの列ブレークと 1 つの列ブレーク値のみが指定されています。ただし、行ブレーク

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

と同様に、複数の列ブレークと列ブレーク値を指定できます。

次は一例になります。

```
ColInfo[] colInfo = new ColInfo[7];
colInfo[0] = new ColInfo(0);
colInfo[0].setRowBreak(true);

colInfo[1]=new ColInfo(1);
colInfo[1].setRowBreak(true);

colInfo[2]=new ColInfo(2);
colInfo.setColumnBreak(true);

colInfo[3]=new ColInfo(3);
colInfo[3].setColumnBreak(true);

colInfo[4]=new ColInfo(4);
colInfo[4].setColumnBreakValue(true);
colInfo[4].setAggregation(ColInfo.SUM);

colInfo[5]=new ColInfo(5);
colInfo[5].setColumnBreakValue(true);
colInfo[5].setAggregation(ColInfo.SUM);

colInfo[6]=new ColInfo(6);
colInfo[6].setColumnBreakValue(true);
colInfo[6].setAggregation(ColInfo.SUM);
```

上記のコードでは、データソースの列 2 と 3 は列ブレークとして設定され、列 4、5、6 は列ブレーク値の列として設定されています。

4.10.3.2 レポートの作成

クロス集計レポートを作成するコードは次のとおりです。

```
ColInfo[] colInfo = new ColInfo[4];
colInfo[0] = new ColInfo(0);
colInfo[0].setRowBreak(true);

colInfo[1] = new ColInfo(1);
colInfo[1].setColumnBreak(true);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
colInfo[2] = new ColInfo(2);
colInfo[2].setColumnBreakValue(true);
colInfo[2].setAggregation(ColInfo.AVG);

QbReport report = new QbReport
                (parent, //Parent
                QbReport.CROSSTAB, //Type of Report
                QbReport.CLASSFILE, //Data
                "ClassFile",
                colInfo, //Column Mapping
                "CrossTab"); //Template Name
```

フルソースコード
エクスポートされた未フォーマットの結果
エクスポートされたフォーマット済の結果

エクスポートされた未フォーマットの結果は、書式設定やテンプレートなしで初めてレポートを作成したときのレポートの表示方法を示します。エクスポートされたフォーマット済の結果には、テンプレートを適用した後の結果が表示されます。

この例では、クラスファイルをデータソースとして使用しています。クラスファイルは、プログラムを実行するたびに温度を生成し、結果が変化します。クラスファイルのデータソースの詳細については、カスタム実装で渡されるデータを参照してください。

4.10.3.3 固定フィールドクロス集計レポート

固定フィールドクロス集計オプションを使用すると、変化するデータを使用して拡張および縮小できるクロス集計レポートをユーザーが簡単に設計できるようになり、フリーフォームクロス集計の実装のいくつかの制限が改善されます。

固定フィールドとフリーフォームのクロス集計の主な違いは、デザインビューでは、クロス集計レポートの個々の列を配置または制御できないことです。代わりに、要素のグループ（行ブレーク、ヘッダー、フッター、および数式）の書式を設定できます。実際のレポートは、実行中/プレビュー中にのみ作成されます。クロス集計テーブルは、レポートが実行されるたびにゼロから構築されるため、スムーズに縮小および拡大するクロス集計テーブルを作成する方が簡単です。

4.10.3.3.1 レポートの作成

次の例は、3 列のブレークで多次元（オプション行/列ブレーク）の固定フィールドクロス集計レポートを作成する方法を示しています。

```
ColInfo colInfo[] = new ColInfo[4];
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
colInfo[0] = new ColInfo(0);
colInfo[1] = new ColInfo(1);
colInfo[2] = new ColInfo(2);
colInfo[3] = new ColInfo(3);

colInfo[0].setRowBreak(true);
colInfo[1].setColumnBreak(true);
colInfo[2].setColumnBreak(true);
colInfo[3].setColumnBreakValue(true);
colInfo[3].setAggregation(ColInfo.SUM);

Properties props = new Properties();
props.put("crossTabFreeForm", "false"); //Using fixed-field form
props.put("crossTabSummaryPositionL", "false"); //Draw summary column to the right of the details matrix
props.put("crossTabFormulaOnHeader", "true"); //Draw formula column in the header section
props.put("crossTabColBkValAlignH", "false"); //Align Column Break Value vertically

QbReport report = new QbReport
                (parent, //Parent
                QbReport.CROSSTAB, //Report Type
                QbReport.DATAFILE, //Data Type
                "sample.dat", //Filename
                colInfo, //Column Mapping
                "field.rpt", //Template
                props); //Properties
```

フルソースコード
エクスポートされた結果

この例では、テキストファイルの列 0 のレポートの列 0、テキストファイルの列 1 のレポートの列 1 などをマッピングしています。上記のコードでは、データソースの列 1,2 および 3 は列ブレーク列に設定され、列 4 は **SUM** 集計付きの列ブレーク値列に設定されます。クロス集計オプションは、固定フィールドレイアウトでクロス集計オプションを個別に書式設定または配置することができないため、さまざまなオプションがあります。この例では、サマリー列はレポートデータの右側に描画されるように設定され、数式列はヘッダーセクションに描画されるように設定され、列ブレーク値は垂直方向に設定されます。

クロス集計の詳細は、クロス集計レポートを参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.10.4 マスタ&詳細

マスタ&詳細レポートは、マスターフィールドに従ってグループ化された一連のテーブル形式のデータです。このレポートタイプは、データテーブル内の一部のフィールドが他の多くのフィールドと関連している場合に最も一般的に使用されます。これの良い例は請求書です。各注文番号には、顧客情報と価格情報付きのいくつかの項目があります。マスタ&詳細レポートは、注文番号に従って情報をグループ化するために使用されます。

マスタ&詳細レポートでは、データは **Primary** または **Master** 列に基づいてグループ化されます。ここでは、**Primary Key** 列を定義する必要があります。**Primary Key** 列（請求書の例の注文番号）によって、レポートのグループが決定されます。選択した列（**Primary Key** 列）に値が入力されるたびに、新しいグループが作成されます。主キーとして使用できる列は 1 つだけです。また、**Master** フィールドまたは複数の **Master** フィールドを選択することもできます。**Master** フィールド（顧客情報）として選択された列は、レポートのデータセクションの代わりにグループヘッダーに列の値を配置します。定義により、**Primary Key** 値は、**Master** フィールド列の各グループに一意でなければなりません。たとえば、以下のデータセットが与えられているとします。

Order#	Customer Name	Product	Unit Price	Quantity
12	Paul Campbell	Chair	$24.95	4
12	Paul Campbell	Table	$127.50	1
14	Sally Hayes	Cabinet	$227.25	2
14	Sally Hayes	Chair	$24.95	2
14	Sally Hayes	Table	$127.50	1

元のデータ

列 **0**（**Order#**）を **Primary Key**、列 **1**（**Customer Name**）を **Master** フィールドとして設定した場合、次のレポートが得られます。

Order#	12	
Customer Name	Paul Campbell	
Product	Unit Price	Quantity
Chair	$24.95	4
Table	$127.50	1
Order#	14	
Customer Name	Sally Hayes	
Product	Unit Price	Quantity
Cabinet	$227.25	2
Chair	$24.95	2
Table	$127.50	1

主キー（Primary Key）とマスター（Master）フィールド適用後のレポート

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.10.4.1 列マッピング

前述のレポートを生成するには、API を使用して次の列マッピングを設定する必要があります。

```
ColInfo[] colInfo = new ColInfo[5];
colInfo[0] = new ColInfo(0);
colInfo[0].setPrimaryKey(true);
colInfo[1]=new ColInfo(1);
colInfo[1].setMaster(true);
colInfo[2] = new ColInfo(2);
colInfo[3] = new ColInfo(3);
colInfo[4] = new ColInfo(4);
```

上記のコードでは、列の名前は、十分なデータソースからの列の元の名前として設定されていません。

マスターセクション内の任意の数の列を選択できます。唯一の制限は、主キーが自動的にマスターセクションに置かれることです。これが望ましくない場合は、その列を非表示に設定できます。

4.10.4.2 レポートの作成

次の例は、マスタ&詳細レポートを作成する方法を示しています。

```
String url = "jdbc:hsqldb:database/woodview";
String driver = "org.hsqldb.jdbcDriver";
String query = "...";

DBInfo dbInfo = new DBInfo(url, driver, "sa", "", query);

ColInfo[] colInfo = new ColInfo[16];

for(int i = 0; i < colInfo.length; i++){
        colInfo[i] = new ColInfo(i);
        if(i < 8)
                colInfo[i].setMaster(true);
}
colInfo[0].setPrimaryKey(true);
colInfo[15].setVisible(false);

QbReport report = new QbReport
                (parent,                    //Parent
                QbReport.MASTERDETAILS, //Type of Report
                dbInfo,                      //Database Info
                colInfo,                   //Column Mapping
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                null);                    //Template Name
try {
        //Apply Template with Formula and Script
        report.applyTemplate("MasterDetails", true);
} catch( Exception e ) {
        e.printStackTrace();
}
```

フルソースコード

クエリの詳細については、フルソースコードを参照してください。エクスポートされた未フォーマット結果は、書式設定やテンプレートなしで初めてレポートを作成したときのレポートの表示方法を示します。エクスポートされたフォーマット済の結果には、テンプレートを適用した後の結果が表示されます。

この例では WoodView HSQL を使用しているため、クラスパスにデータベース HSQL JDBC ドライバ（**hsqldb.jar**）を追加する必要があります。ドライバは<**EspressReportInstall**>/**lib** ディレクトリにあります。データベースデータソースの使用の詳細については、データベースのデータを参照してください。

この例では合計 16 個の列があるので、for ループ文を使用して列情報を設定します。最初の 8 列はマスター列として設定され、グループヘッダーセクションに表示されます。列 **0**（**OrderID**）が **Primary Key**（**主キー**）として設定されます。

テンプレートに格納された数式とスクリプトをインポートするために、**QbReport** オブジェクトを作成した後にテンプレートを適用します。

マスターセクションを残りの列と並べて配置することもできます。これを行うには、次の QbReport コンストラクタを使用し、**sideBySideLayout** パラメータに **true** を渡します。

```
QbReport(java.lang.Object parent, int reportType, IDatabaseInfo dbinfo, ColInfo[] mapping, java.lang.String template, boolean sideBySideLayout)
```

4.10.5 メーリングリスト

メーリングラベルレポートでは、データがグループ化されずに区切られた列形式で表示されます。**シンプルカラムナ**レポートに似ています。データソースに対して ColInfo 配列オブジェクトを定義するだけです。たとえば、以下のデータセットが与えられているとします。

Name	Company	Region
Mary	ABC Inc.	New York

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Peter	GHI Ltd	London

元のデータ

列マッピングは、データソースの列がレポートにマップされるように設定されます。したがって、レポートの列 0 は、データソースの列 0、レポートの列 1 はデータソースの列 1 などに対応します。したがって、次のレポートが作成されます。

Name	Mary
Company	ABC Inc.
City	New York
Name	Peter
Company	GHI Ltd
City	London

テーブルヘッダー名のマッピングと設定後のレポート

4.10.5.1 列マッピング

前述のレポートを生成するには、API を使用して次の列マッピングを設定する必要があります。

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(0);
colInfo[1] = new ColInfo(1);
colInfo[2] = new ColInfo(2);
colInfo[2].setName("City");
```

上記のコードでは、元のデータに定義されているヘッダー名を保持するのではなく、3 列目の名前も **City** に変更されています。

列マッピングは、順序通りである必要はなく、データと同じ順序に従います。また、希望する列を選択して選択することもできます。たとえば、複数の列を持つデータソースがある場合、次のコードを記述できます。

```
ColInfo[] colInfo = new ColInfo[3];
colInfo[0] = new ColInfo(2);
colInfo[1] = new ColInfo(7);
colInfo[2] = new ColInfo(4);
```

この例では、列 **0** にはデータソースの列 **2** が、列 **1** にはデータソースの列 **7** が、列 **2** にはデータソースの列 **4** がマッピングされています。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

4.10.5.2 レポートの作成

次の例は、**メーリングラベル**レポートを作成する方法を示しています。

```
String xmlfilename = "Inventory.xml";
String dtdfilename = "Inventory.dtd";
String xmlcondition = "/Inventory/Category/Product/UnitsInStock < 15";

XMLFieldInfo[] fields = new XMLFieldInfo[8];
fields[0] = new XMLFieldInfo(new String[] { "Inventory", "Category" }, "CategoryName");
fields[1] = new XMLFieldInfo(new String[] { "Inventory", "Category", "Product" },
                "ProductID");
fields[2] = new XMLFieldInfo(new String[] { "Inventory", "Category", "Product",
                "ProductName" });

fields[3] = new XMLFieldInfo(
                new String[] { "Inventory", "Category", "Product", "UnitPrice" });
fields[3].setElementDataType(DTDDataType.DOUBLE);

fields[4] = new XMLFieldInfo(
                new String[] { "Inventory", "Category", "Product", "Material" });
fields[5] = new XMLFieldInfo(new String[] { "Inventory", "Category", "Product", "Material",
                "Units" });
fields[5].setElementDataType(DTDDataType.INT);

fields[6] = new XMLFieldInfo(new String[] { "Inventory", "Category", "Product", "Material",
                "CostPerUnit" });
fields[6].setElementDataType(DTDDataType.DOUBLE);

fields[7] = new XMLFieldInfo(new String[] { "Inventory", "Category", "Product",
                "UnitsInStock" });
fields[7].setElementDataType(DTDDataType.INT);

XMLFileQueryInfo xmlInfo = new XMLFileQueryInfo(xmlfilename, fields, xmlcondition, fields,
                dtdfilename, false, null);

ColInfo[] colInfo = new ColInfo[8];
for (int i = 0; i < 8; i++) {
        colInfo[i] = new ColInfo(i);
}

QbReport report = new QbReport
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                        (parent, //Parent
                        QbReport.MAILINGLABELS, //Type of Report
                        xmlInfo, //XML Info
                        colInfo, //Column Mapping
                        null, //Template Name
                        false); //Side By Side

try {
        //        Apply Template with Formula and Script
        report.applyTemplate("MailingLabels.rpt", true);

} catch (Exception e) {
        e.printStackTrace();
}
```

フルソースコード
エクスポートされた未フォーマットの結果
エクスポートされたフォーマット済の結果

エクスポートされた未フォーマット結果は、書式設定やテンプレートなしで初めてレポートを作成したときのレポートの表示方法を示します。エクスポートされたフォーマット済の結果には、テンプレートを適用した後の結果が表示されます。

テンプレートに保存された数式とスクリプトをインポートするために、**QbReport** オブジェクトを作成した後にテンプレートを適用します。

4.10.6 式

元のデータを使用してレポートの列を生成するだけではありません。独自の値で独自の列を作成したり、数式を使用して 1 つ以上の列を計算したりすることができます。たとえば、以下のデータセットが与えられているとします。

Units Sold	Product	Unit Price
12	Chair	$25
2	Table	$38
18	Cabinet	$21
4	Sofa	$57

元のデータ

列 **0**（**Units Sold**）と列 **2**（**Unit Price**）をかけた結果の新しい列（**Total Revenue**）を作成する場合、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次のようになります。

Product	Units Sold	Unit Price	Total Revenue
Chair	12	$25	$300.00
Table	2	$38	$76.00
Cabinet	18	$21	$378.00
Sofa	4	$57	$228.00

式の適用および新しい列の作成後のレポート

上述したレポートを作成するには、API を使用して次の列マッピングを設定する必要があります。

```
ColInfo[] colInfo = new ColInfo[4];
colInfo[0] = new ColInfo(1);
colInfo[1] = new ColInfo(0);
colInfo[2] = new ColInfo(2);
IObject salesColumn = NumericObject.multiply(NumericObject.getColumnValue(1),
                                NumericObject.getColumnValue(2));
colInfo[3] = new ColInfo(salesColumn, "Total Revenue", Types.DOUBLE);
```

上記のコードでは、列の名前は、十分なデータソースからの列の元の名前として設定されていません。新しく作成された列の場合、**Total Revenue** がヘッダーとして指定されました。

式の詳細については数式および式ビルダの使用を参照してください。

4.10.7 パラメータ化されたレポート

通常のクエリに加えて、パラメータを持つクエリを渡すことができ、レポートを生成する前に、レポートにパラメータの値を入力させることができます。各クエリで許可されるパラメータの数に制限はありません。

パラメータ付きクエリをレポートのデータソースとして使用するには、**IQueryFileInfo** を実装する **SimpleQueryFileInfo** クラスを使用してクエリ情報を渡す必要があります。SimpleQueryFileInfo は **DBInfo** のサブクラスなので、同じパラメータを使用して **SimpleQueryFileInfo** オブジェクトを作成できます。

```
SimpleQueryFileInfo(java.lang.String url, java.lang.String driver, java.lang.String username, java.lang.String password, java.lang.String query)
```

クエリオブジェクトの作成に加えて、単一値のパラメータの場合は **SimpleQueryInParam** の配列を、複数値のパラメータの場合は **SimpleQueryMultiValueInParam** の配列のいずれかを使用してパラメー

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

タを作成する必要があります。最後に、**setInParam(ParamSet)**メソッドを使用して、パラメータセットのインスタンスをクエリオブジェクトに渡す必要があります。

単一値のパラメータと複数値のパラメータのコンストラクタは、1 つが他のサブクラスであるため同一です。

```
SimpleQueryInParam(java.lang.String paramName, java.lang.String promptName, boolean mapToColumn, java.lang.String tableName, java.lang.String columnName, int sqlType, java.lang.Object defaultValue, java.lang.Object value)
```

paramName はパラメータの名前で、別々のパラメータインスタンスで作成されている場合は、同じ名前の 2 つのパラメータが許可されます。**promptName** は、パラメータの値の入力を求められたときに表示されるメッセージです。

パラメータ **mapToColumn**、**tableName** および **columnName** は、パラメータ入力に値が使用されるデータベースから列を指定するかどうかを決定します。**true** を選択すると、Report Viewer でレポートをプレビューまたは実行するときにエンドユーザーに表示される **prompt** パラメータが変更されます。パラメータをデータベース列にマップすると、パラメータ値を選択するための異なる値のドロップダウンリストが表示されます。マップしない場合は、**tableName** および **columnName** を **null** に設定します。ユーザーは特定のパラメータ値を入力する必要があります。パラメータ化されたクラスファイルからデータベースをクエリし、クラスファイルをデータソースとして使用する場合、**mapToColumn** を **true** に設定できないことに注意してください。パラメータ化されたクラスファイルの列への別のマッピング方法については、カスタム実装で渡されるデータを参照してください。

sqlType は、パラメータのデータ型を決定します。**java.sql.Types** クラスのデータ型に一致する定数を選択します。プロンプトが最初に表示されたら、**defaultValue** が入力ボックスに表示されます。列にマップし、存在しないデフォルト値を指定すると、ドロップダウンリストの最初の要素が表示されます。最後のパラメータはオブジェクト型で、このパラメータが **null** でない場合、パラメータプロンプトは表示されません。ユーザーが値を指定できるようにする代わりに、このパラメータの値をクエリに使用します。

4.10.7.1 単一値パラメータを使用したレポートの作成

次の例では単一値パラメータを使用したレポートを作成します。

```
SimpleQueryInParam inParam = new SimpleQueryInParam(
                    "param", "Please select", true, "Categories",
                    "CategoryName", Types.VARCHAR, "Arm Chairs", null);
SimpleQueryInParam[] paramSet = { inParam };

SimpleQueryFileInfo reportInfo = new SimpleQueryFileInfo(
                    "jdbc:hsqldb:woodview",
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                    "org.hsqldb.jdbcDriver",
                    "sa", "",
                    "...");
reportInfo.setInParam(paramSet);
//                    End Code : Adding Parameter Info

//Begin Code : Setting up Column Mapping
ColInfo colInfo[] = new ColInfo[4];
for (int i = 0; i < colInfo.length; i++) {
        colInfo[i] = new ColInfo(i);
}
colInfo[0].setPrimaryKey(true);
colInfo[1].setName("Product Name");
colInfo[2].setName("Price");
colInfo[3].setName("Units In Stock");
//End Code : Setting up Column Mapping

QbReport report = new QbReport(parent,
                    QbReport.MASTERDETAILS,
                    reportInfo,
                    colInfo,
                    "SingleValueParameter.rpt");
```

フルソースコード
エクスポートされた結果

この例では WoodView HSQL データベースを使用しているため、クラスパスにデータベース HSQL JDBC ドライバ（**hsqldb.jar**）を追加する必要があります。ドライバは**<EspressReportInstall>/lib** ディレクトリにあります。データベースデータソースの使用の詳細については、データベースのデータを参照してください。

ここで、パラメータは **Category** テーブルの **CategoryName** カラムにマッピングされます。プロンプトに **Please Select** と表示され、ユーザーが選択するための値がドロップダウンリストに表示されます。

4.10.7.2 複数値パラメータを使用したレポートの作成

たとえば、ユーザーが単一の値ではなく値の範囲に対してチェックしたい場合など、複数の値を持つようにパラメータを割り当てることもできます。値の範囲は、通常、SQL クエリの **IN** 句内で指定されます。EspressReport は、**IN** 句内のパラメータを複数値と見なすだけであることに注意してください。

次の例は、複数値のパラメータを使用してレポートを作成する方法を示しています。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
SimpleQueryMultiValueInParam inParam = new SimpleQueryMultiValueInParam("param",
                "Please select", true, "products", "productid", Types.INTEGER, new Integer(1), null);
SimpleQueryMultiValueInParam[] paramSet = { inParam };

SimpleQueryFileInfo rootInfo = new SimpleQueryFileInfo(
                "jdbc:hsqldb:woodview",
                "org.hsqldb.jdbcDriver",
                "sa", "",
                "select productid, productname " +
                "from Products where productid in (:param)");
rootInfo.setInParam(paramSet);
//                End Code : Adding Parameter Info

//                Begin Code : Setting up Column Mapping
ColInfo rootColInfo[] = new ColInfo[2];
rootColInfo[0] = new ColInfo(0);
rootColInfo[0].setName("Product ID");
rootColInfo[1] = new ColInfo(1);
rootColInfo[1].setName("Product Name");

//                End Code : Setting up Column Mapping

QbReport rootReport = new QbReport(
                parent,
                QbReport.MAILINGLABELS,
                rootInfo,
                rootColInfo,
                "MultiValueParameter.rpt");
```

フルソースコード
エクスポートされた結果

4.10.8 サブレポート

メインレポートにサブレポートを追加することもできます。これは、2 つのレポートを 1 つに表示できるので、より多くのデータを表示するのに役立ちます。サブレポートの詳細については、サブレポートを参照してください。

サブレポートを含むレポートを作成するには、サブレポートオブジェクトを作成し、そのサブレポートオブジェクトをレポートセルオブジェクトとしてメインレポートに挿入する必要があります。 サブレ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ポートはメインレポートのどこにでも配置できます。**SubReportObject** を作成するには、次の静的メソッドを使用します。

SubReport.createSubReport(QbReport parentReport, int reportType, IDatabaseInfo dbinfo, ColInfo[] mapping, java.lang.String template)

QbReport コンストラクタと同様に、さまざまなデータソースを補正するためのパラメータにはさまざまなバリエーションがあります。上記のメソッドは、データベースからデータを取得します。サブレポートにはパラメータも含めることができ、API からのパラメータ共有も設定できます。**SubReportObject** が作成されると、**addData(subReportObject)**メソッドを使用してレポートに挿入できます。

4.10.8.1 レポートの作成

次の例は、サブレポートを含むレポートを作成する方法を示しています。

```
//                Begin Code : Creating the sub report
DBInfo subReportInfo = new DBInfo(
                "jdbc:hsqldb:woodview",
                "org.hsqldb.jdbcDriver",
                "sa",
                "",
                "SELECT Employees.FirstName + ' ' + Employees.LastName AS Employee, "
                                +
                                "Count(Order_Details.OrderID) AS Orders, "
                                +
                                "Sum(  (  Products.UnitPrice  +  Order_Details.StainCost  )  *
Order_Details.Quantity) AS Sales "
                                +
                                "FROM Employees, Products, Orders, Order_Details " +
                                "WHERE ((Orders.EmployeeID = Employees.EmployeeID) " +
                                "AND (Orders.OrderID = Order_Details.OrderID) " +
                                "AND (Products.ProductID = Order_Details.ProductID)) " +
                                "AND (((Orders.OrderDate BETWEEN '2003-01-10' AND '2003-31-12'))) "
+
                                "GROUP BY Employees.FirstName + ' ' + Employees.LastName;");

ColInfo subReportColInfo[] = new ColInfo[3];
for (int i = 0; i < subReportColInfo.length; i++) {
        subReportColInfo[i] = new ColInfo(i);
}
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
try {
	//			createSubReport(root report, report type, sub report data source, sub report column mapping, template)
	SubReportObject reportCell = quadbase.reportdesigner.ReportAPI.SubReport
			.createSubReport(
				rootReport,
				QbReport.COLUMNAR,
				subReportInfo,
				subReportColInfo,
				"SubReportSub.rpt");
	reportCell.setWidth(rootReport.getTable().getHeader().getWidth() - 2);
	reportCell.setHeight(2);
	//			Or use the following method if the height of the subreport may change
	//			reportCell.setResizeToFitContent(true);

	reportCell.setY(0);
	reportCell.setX(2);

	//			Add SubReport to the Table Header
	rootReport.getTable().getHeader().addData(reportCell);
	rootReport.getTable().getHeader().setHeight(1.8);
} catch (Exception ex) {
	ex.printStackTrace();
}
//		End Code : Creating the sub report
```

フルソースコード
エクスポートされた結果

この例のメインレポートでは、**DBInfo** オブジェクトを使用してデータを取得し、サマリーレポートを作成します。どちらのレポートでも、この例のコード量を減らすためのテンプレートが適用されます。サブレポートが作成され、メインレポートのテーブルヘッダーセクションに挿入されます。サブレポート全体を表示するには、SubReportObject とテーブルヘッダーセクションのサイズを調整するために、いくつかのコード行が必要であることに注意してください。

また、メインレポートのパラメータを使用するサブレポートを含めることもできます。サブレポートを使用すると、サブレポート内のデータを取り込むことができます。これは、メインレポート内のセクションに関連するデータを入れるのに役立ちます。パラメータ化されたサブレポートは、サブレポートのパラメータ化されたクエリを渡し、サブレポートのクエリのパラメータを提供するメインレポートから列を指定することによって作成されます。サブレポートに渡されるパラメータは、指定された列の最初の値です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

サブレポートの配置に応じて、列の個別の値ごとにリンクされたサブレポートを作成できます。たとえば、サマリブレークレポートでは、パラメータが行ブレーク列にリンクされたサブレポートを作成できます。このサブレポートをグループヘッダーセクションに配置すると、行ブレーク列の各個別値（主な値ごとに列が分割されるため）ごとに、メインレポートにサブレポートが生成されます。

リンクされたサブレポートは、常にそのパラメータにマップされた列の最初の値をとります。ただし、サブレポートの配置によっては、グループ内の列の最初の値を取得するたびにレポート内に複数回表示されることがあります。

以下は、メインレポートからパラメータを取り込むグループヘッダーに配置されたサブレポートの例です。

```
QbReport.setEspressManagerUsed(false);

//                Begin Code : Creating the root Report
DBInfo rootInfo = new DBInfo("jdbc:hsqldb:woodview",
                "org.hsqldb.jdbcDriver",
                "sa",
                "",
                "SELECT Customers.Company, Orders.OrderID, " +
                                "Products.ProductName, Products.UnitPrice, " +
                                "Order_Details.Quantity " +
                                "FROM Order_Details, Products, Orders, Customers " +
                                "WHERE (Orders.CustomerID = Customers.CustomerID) " +
                                "AND (Order_Details.OrderID = Orders.OrderID) " +
                                "AND (Products.ProductID = Order_Details.ProductID);");

ColInfo rootColInfo[] = new ColInfo[5];
rootColInfo[0] = new ColInfo(0);
rootColInfo[0].setRowBreak(true);
ootColInfo[1] = new ColInfo(1);
rootColInfo[1].setRowBreak(true);
rootColInfo[2] = new ColInfo(2);
rootColInfo[2].setAggregation(false, ColInfo.NONE);
rootColInfo[3] = new ColInfo(3);
rootColInfo[3].setAggregation(false, ColInfo.AVG);
rootColInfo[4] = new ColInfo(4);
rootColInfo[4].setAggregation(false, ColInfo.SUM);

QbReport rootReport = new QbReport(
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                parent,
                QbReport.SUMMARY,
                rootInfo,
                rootColInfo,
                "ParameterizedSubReport.rpt");
//                End Code : Creating the root Report

int numberOfTableHeaderCells = rootReport.getTable().getHeader().getData().length;

//                Transferring Table Header cells to Row Break Zero Header
for (int i = 0; i < numberOfTableHeaderCells; i++)
{

        rootReport.getTable().getRowBreakHeader(0)
                        .addData(rootReport.getTable().getHeader().getData(i));
        rootReport.getTable().getRowBreakHeader(0).getData(i).setY(1);
}

//                Deleting Table Header cells
for (int i = 0; i < numberOfTableHeaderCells; i++) {
        rootReport.getTable().getHeader().removeData(0);
}

//                Begin Code : Creating the SubReport
//                 SimpleQueryInParam(name of Parameter, String to be displayed, MapToColumn?,
tableName, ColumnName, SQL Type, DefaultValue, value)
SimpleQueryInParam inParam = new SimpleQueryInParam(
                "Company", "Please select", true, "Customers",
                "Company", Types.VARCHAR, "All Unfinished Furniture",
                "All Unfinished Furniture");
SimpleQueryInParam[] paramSet = { inParam };

SimpleQueryFileInfo subReportInfo = new SimpleQueryFileInfo(
                "jdbc:hsqldb:woodview",
                "org.hsqldb.jdbcDriver",
                "sa", "",
                "SELECT Customers.CustomerID, Customers.ContactName, " +
                                "Customers.Address, Customers.City, " +
                                "Customers.State, Customers.Zip " +
                                "FROM Customers " +
                                "WHERE (Customers.Company =:Company);");
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
subReportInfo.setInParam(paramSet);

//              Begin Code : Setting up Column Mapping
ColInfo subReportColInfo[] = new ColInfo[6];
for (int i = 0; i < subReportColInfo.length; i++) {
        subReportColInfo[i] = new ColInfo(i);
}

        //              End Code : Setting up Column Mapping

        try {

        //                      createSubReport(root report, report type, sub report data source, sub report column mapping, template)
        SubReportObject reportCell = SubReport.createSubReport(
                        rootReport,
                        QbReport.COLUMNAR,
                        subReportInfo,
                        subReportColInfo,
                        null);
        ((QbReport) reportCell.getSubReport()).applyTemplate(
                        "ParameterizedSubReportSub.rpt", true);

        //                      Get Parameter from root report column
        reportCell.setParameterMap(new String[] { rootReport.getTable().getColumn(0).getID() });

        //                      Set SubReport cell's height and width
        reportCell.setWidth(7.2);
        reportCell.setResizeToFitContent(true);
        reportCell.setY(.3);
        reportCell.setX(0);

        //                      Add SubReport to the Group Zero Header
        rootReport.getTable().getRowBreakHeader(0).addData(reportCell);

} catch (Exception ex) {
        ex.printStackTrace();
}
//              End Code : Creating the sub report

return (new Viewer().getComponent(rootReport));
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フルソースコード
エクスポートされた結果

メインレポートとサブレポートの両方に同じ名前のパラメータがある場合、メインレポートからサブレポートのパラメータに列をリンクする必要はありません。

4.10.9 ドリルダウン

パラメータ化されたクエリまたはクラスから、ドリルダウンレポートを作成することができます。大量のデータを含むレポートを表示するのではなく、必要な最小限のデータを示す最上位レベルのレポートを表示して、選択したデータをさらに詳しく調べることができます。ドリルダウンレポートの詳細については、ドリルダウンを参照してください。

ドリルダウンレポートを作成するには、さまざまな **QbReport** オブジェクトを作成する必要があります（ルートレポートオブジェクト以外のすべての **QbReport** オブジェクトは、データソースとしてパラメータ化されたクエリを持つことに注意してください）。ドリルダウンの順序とレポートの列を指定しドリルダウンレポートの次のレベルを添付します。ドリルダウンレポートを作成するには、次のメソッドを使用します。

```
rootReport.createDrillDownReport(java.lang.String name, int reportType, IDatabaseInfo dbInfo, ColInfo[] mapping, java.lang.String template, int[] columnMapping)
```

サブレポートとは異なり、ドリルダウンレポートはパラメータ化されたクエリでなければならないため、ドリルダウンレポートの作成方法は少なくなります。

4.10.9.1 レポートの作成

次の例は、ドリルダウンレポートを含むレポートを作成する方法を示しています。

```
Component doDrillDownReport(Object parent) {
QbReport.setEspressManagerUsed(false);

//              Begin Code : Creating Report 1   - the Root Report
DBInfo rootInfo = new DBInfo("jdbc:hsqldb:woodview",
                "org.hsqldb.jdbcDriver",
                "sa",
                "",
                "select Categories.CategoryName, sum(Products.UnitsInStock) " +
                        "from Categories, Products " +
                        "where Categories.CategoryID = Products.CategoryID " +
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                        "group by Categories.CategoryName " +
                        "order by Categories.CategoryName;"
                );

ColInfo rootColInfo[] = new ColInfo[2];
for (int i = 0; i < rootColInfo.length; i++) {
        rootColInfo[i] = new ColInfo(i);
}

rootColInfo[0].setName("Product Name");
rootColInfo[1].setName("Total Units In Stock");
QbReport rootReport = new QbReport(
                parent,
                QbReport.COLUMNAR,
                rootInfo,
                rootColInfo,
                "DrillDown.rpt");
//               End Code : Creating Report 1    - the Root Report
//               Begin Code : Creating Report 2    - the drill-down Report
SimpleQueryInParam inParam = new SimpleQueryInParam("param", "Please select", true,
                "Categories", "CategoryName", Types.VARCHAR, "Arm Chairs", null);

SimpleQueryInParam[] paramSet = { inParam };
SimpleQueryFileInfo levelOneReportInfo = new SimpleQueryFileInfo(
                "jdbc:hsqldb:woodview",
                "org.hsqldb.jdbcDriver",
                "sa",
                "",
                "select Categories.CategoryName, Products.ProductName, Products.UnitPrice,
Products.UnitsInStock from Categories, Products where
Categories.CategoryID=Products.CategoryID and Categories.CategoryName=:param order by
Categories.CategoryName, Products.ProductName;");
levelOneReportInfo.setInParam(paramSet);
ColInfo levelOneReportColInfo[] = new ColInfo[4];
for (int i = 0; i < levelOneReportColInfo.length; i++) {
        levelOneReportColInfo[i] = new ColInfo(i);
}

levelOneReportColInfo[0].setName("Product Type");
levelOneReportColInfo[1].setName("Product Name");
levelOneReportColInfo[2].setName("Price");
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
levelOneReportColInfo[3].setName("Units In Stock");

try {
        //                      createDrillDownReport(Name, Type of Report, database information,
column mapping, template, column of root report to be mapped to)
        rootReport.createDrillDownReport("TestDrillDownReport",
                        QbReport.COLUMNAR,
                        levelOneReportInfo,
                        levelOneReportColInfo,
                        "lvl1.rpt", new int[] { 0 });
} catch (Exception ex) {
        ex.printStackTrace();
}

rootReport.getTable().getColumn(0).setDrillDownName("TestDrillDownReport");
//              End Code : Creating Report 2    - the drill-down Report
return (new Viewer().getComponent(rootReport));
```

フルソースコード
エクスポートされた結果

EspressManager を使用せずにドリルダウンレポートを生成する場合は**.class** ファイルの作業ディレクトリの下に **DrillDown** というサブディレクトリが必要です。

上記のエクスポート結果には、**Oval Tables** のルートレポートと**ドリルダウン**レポートのみが含まれています。

ドリルダウンレポートを DHTML または PDF 形式にエクスポートする場合は、**DrillDownReportServlet** サーブレットを用意する必要があります。このサーブレットは、レポート内のリンクを次のレベルに提供します。クリックされたリンクに応じて、このサーブレットは適切な次のレベルのレポートを配信します。このサーブレットを使用するには、サーブレットランナー/アプリケーションサーバからアクセスできるようにする必要があります。このサーブレットにアクセスする URL は、**http://<machine name>:<port number>/servlet/DrillDownReportServlet** でなければなりません。次に、アプリケーションコードで、次のメソッドを使用する必要があります。

```
QbReport.setDynamicExport(boolean state, String serverName, int servletRunnerPort);
```

たとえば、ドリルダウンレポートと次のメソッド呼び出しがあるとします。

```
report.setDynamicExport(true, "Aphrodite", 8080);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

レポートがエクスポートされると、リンクはマシン **Aphrodite** とポート **8080** を指していることがわかります。**DrillDownReportServlet** サーブレットは URL:**http://Aphrodite:8080/servlet/DrillDownReportServlet** を使用して呼び出され、次のレベルはクリックされたリンクに基づいており、次のレベルを示しています。

このサーブレットの URL は、**QbReport** クラスの **setServletDirectory(String)**メソッドを使用して変更することもできます。たとえば、上記の例では、次のコード行が追加されたとします。

```
report.setServletDirectory("ER/Test/");
```

リンクにデフォルトのサーブレットの代わりに **http://Aphrodite:8080/ER/Test/DrillDownReportServlet** が表示されます。

HTML、DHTML、または PDF 以外の形式にエクスポートされたドリルダウンレポートには、現在のレベルのみが表示されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5 チャートガイド

5.1 概要

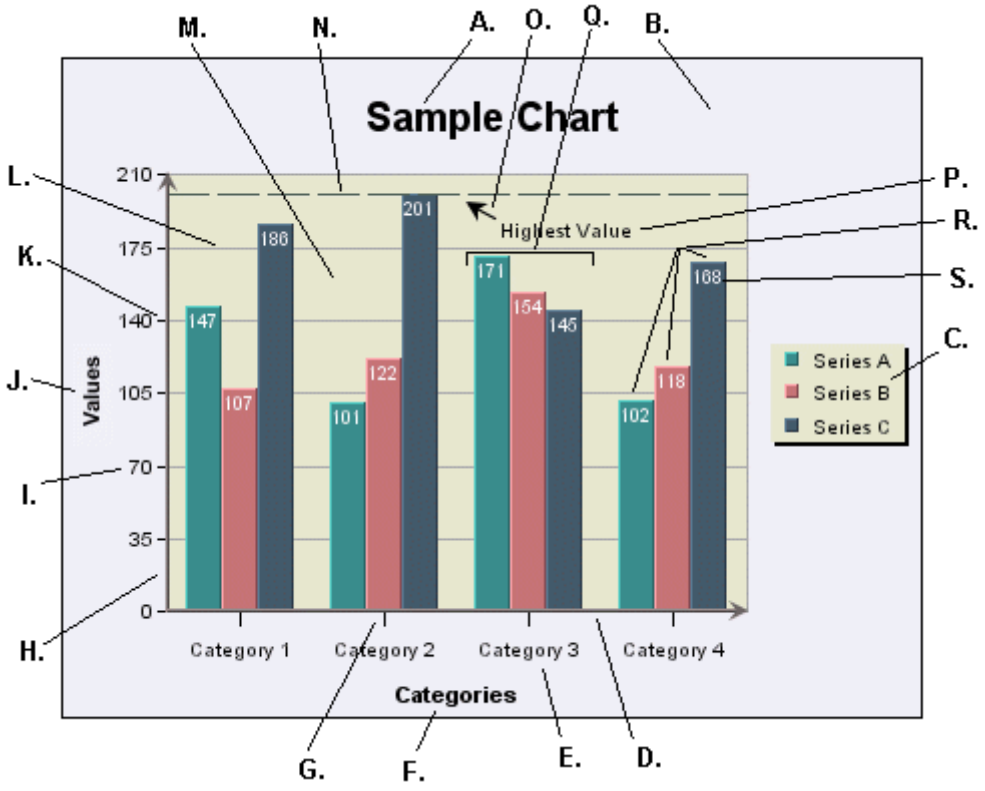
EspressReport には洗練された強力なチャートエンジンが組み込まれており、さまざまなタイプのビジュアルデータ表現をレポートに含めることができます。スタンドアロングラフを Web ページやアプリケーションに追加して、レポートから独立してグラフを展開することもできます。グラフは、埋め込まれた Chart Designer インタフェースを使用するか、チャート API を使用して作成できます。このガイドでは、Chart Designer と API を使用してチャートを設計する方法と、レポートを使用してチャートを独立して配置する方法の両方について説明します。

5.2 チャート作成の基礎

このセクションでは、チャートのさまざまな部分を含むチャートの基本と、レポートまたは別のデータソースのいずれかのテーブルデータをチャートにマップする方法について説明します。

5.2.1 チャートとは

次の図は、チャートを構成するさまざまなコンポーネントを示しています。また、この図では、このガイド全体で使用されるさまざまな用語についても言及しています。



Copyright(C) 2018 Climb.Inc. All Rights Reserved.

A. メインタイトル(Main Title)

チャートのメインタイトルです。

B. チャートキャンバス(Chart Canvas)

グラフが描画される背景です。キャンバスはチャートのサイズ/境界として機能し、一般にエクスポート後のイメージと同じサイズです。キャンバスの色を変更することや、画像ファイルを背景として追加/追加することができます。

C. 凡例(Legend)

凡例の部分です。凡例には、カラーポイントとともにカテゴリまたはシリーズ名が表示されます。セカンダリ値、追加されたトレンド/コントロールライン、コントロールエリアは、凡例に表示することもできます。

D. カテゴリ(X)軸(Category (X)Axis)

チャートの X 軸またはカテゴリ軸です。一般的にカテゴリ軸は、グラフの値をプロットするデータセットからの別個のエントリをプロットします。(値は一般に Y 軸にプロットされます)。棒グラフや Gantt などの特定のグラフタイプでは、Y 軸にカテゴリを描画し、X 軸に値をプロットすることでこれを取り消します。2D または 3D 空間にポイントを作成するために各軸に値をプロットする散布図やバブルプロットなどのグラフタイプがあります。

E. X 軸ラベル(X-Axis Labels)

X 軸の要素またはカテゴリのラベルです。

F. X 軸タイトル(X-Axis Title)

X 軸のタイトルです。

G. X 軸ティッカー(X Ticker)

X 軸ティッカーです。デフォルトでは、ティッカーはチャートの各データポイントと一致します。

H. 値(Y)軸(Value (Y) Axis)

チャートの Y または値の軸です。一般的に Y 軸は各カテゴリの値をプロットします。デフォルトでは、Y 軸のスケールはデータセットに最適なものを生成するために生成されます。ただし、手動で調整することができます。コンビネーションチャートでは、プロットの右側に 2 番目の値の軸が描画されます。

I. Y 軸ラベル(Y-Axis Labels)

Y 軸の値のラベルです。

J. Y 軸タイトル(Y-Axis Title)

Y 軸のタイトルです。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

K. Y 軸ティッカー(Y-Axis Ticker)

Y 軸のティッカーです。

L. Y グリッド(Y Grid)

Y 軸の各スケールステップに沿って描かれたグリッド線です。グリッド線は、X 軸（3D チャートの場合は Z 軸）の点についても描画できます。

M. プロットエリア(Plot Area)

すべてのデータ点がプロットされている軸によって境界が定められたエリアです。エリアを色で塗りつぶすことや、周囲の枠線を描くことができます。他のオプションもあります。プロットエリアは、チャートのキャンバス上で移動およびサイズ変更できます。

N. コントロールライン(Control Line)

グラフに追加できる特別な種類の線の 1 つです。この例では、シリーズ内の最高値に線を描画することや標準偏差の平均および倍数に関してコントロールラインを描画することができます。 また、さまざまなトレンドラインをグラフに追加することもできます。

O. フローティングライン(Floating Line)

フローティングラインは、チャートに追加された任意の線です。この場合、矢印付きのポインタとして使用されています。フローティングラインはチャートのプロットと相対的な位置に移動します。塗りつぶされた図形を作成するために使用することもできます。

P. 注釈テキスト(Annotation Text)

これはチャートに追加する任意のテキストです（ラベルやタイトルではありません)。チャートのキャンバスのどこにでもテキストを配置できます。フローティングラインように、注釈テキストは相対位置でチャートのプロットに配置します。

Q. カテゴリ要素(Category Elements)

これはカテゴリ要素のプロットです。このチャートにはデータシリーズが 3 つあるため、各カテゴリに 3 つの点がプロットされています。

R. データシリーズ要素(Data Series Elements)

これらは、カテゴリを構成する個々のポイントです。一連のデータを 1 つのチャートにプロットすることができます。カテゴリとシリーズについての詳細は、基本的データマッピングを参照してください。

S. データトップラベル(Data Top Labels)

これらは、チャートの各データポイントに配置され、各ポイントの正確な値を表示するラベルです。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

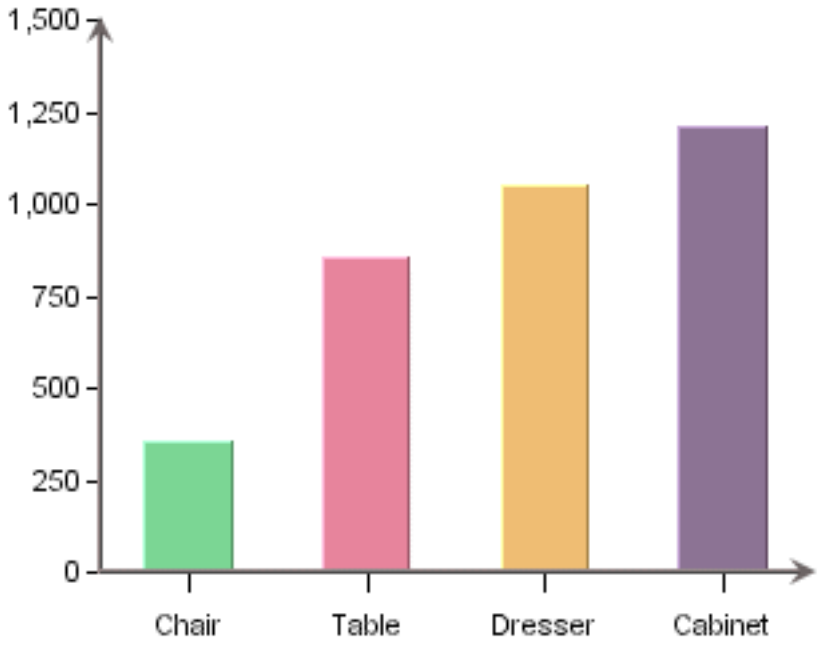
5.2.2 基本的データマッピング

データマッピングは、Raw データをチャートに描画する方法です。データは多くのソースから取り出すことができますが、グラフにするためにはテーブルの形式でデータの基本構造を持つ必要があります。そのため、レポートや XML ファイルから引数として渡されたデータを、マッピングの前にテーブル構造に変換します。

基本的なデータセットは次のようになります。

Product	Sales
Chair	362
Table	862
Dresser	1052
Cabinet	1211

このデータをグラフにプロットするには、**Product** 列に各エントリの **Sales** 値をプロットします。したがって、**Product** はカテゴリであり、**Sales** は値です。チャートでは、**Product** 列を X（カテゴリ）軸に、**Sales** 列を Y（値）軸にマッピングします。結果のプロットは縦棒グラフで次のようになります。

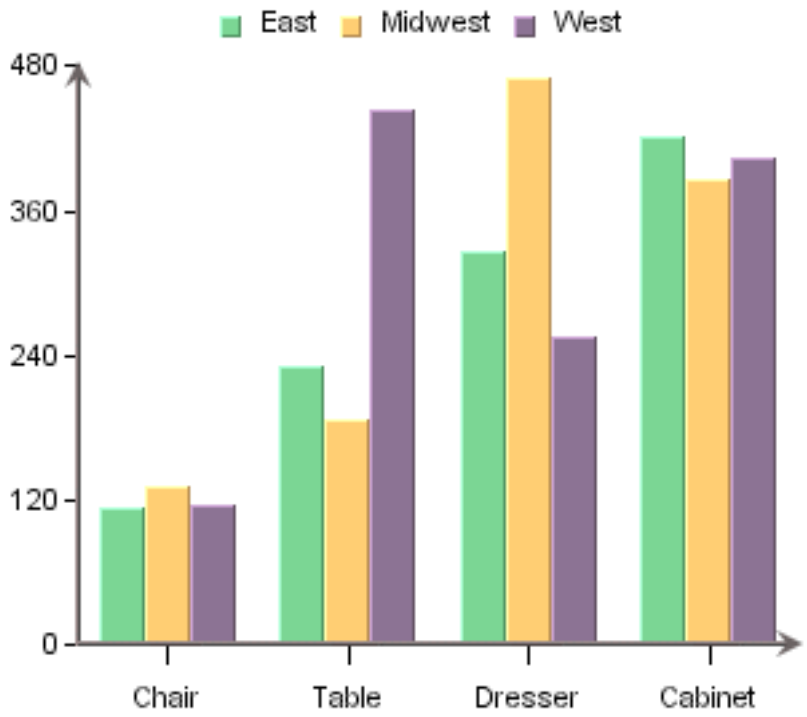


ここでは、カテゴリ列の各個別要素の値を示す列が描画されます。基本カテゴリ値の上に、追加情報をデータシリーズの形式で表示することができます。たとえば、データだけでなくリージョン別の販売データも表示する要素が他にもあるとします。調整後のテーブルは次のようになります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

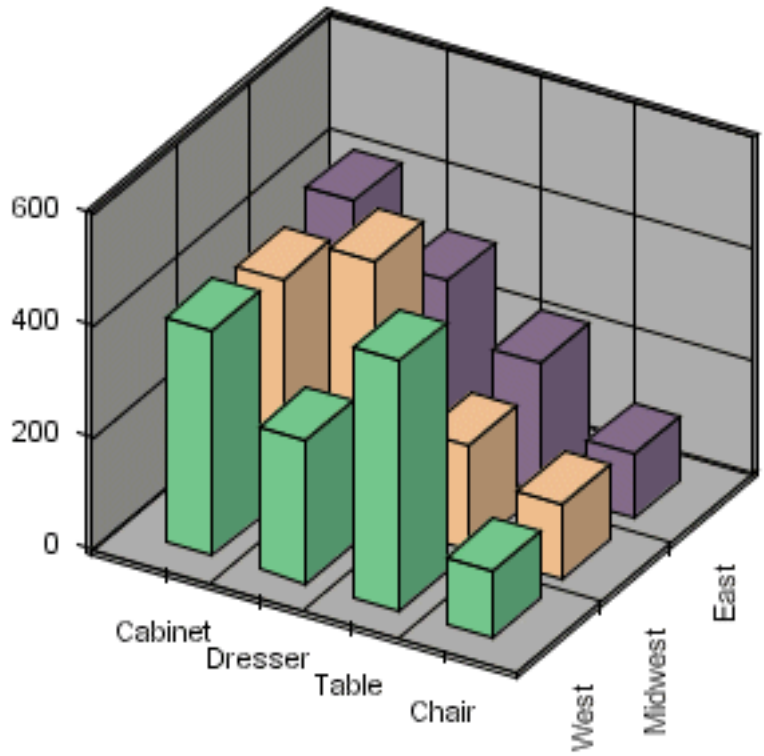
Product	Region	Sales
Chair	East	114
Chair	Midwest	131
Chair	West	117
Table	East	231
Table	Midwest	187
Table	West	444
Dresser	East	327
Dresser	Midwest	469
Dresser	West	256
Cabinet	East	422
Cabinet	Midwest	386
Cabinet	West	403

製品ごとに各地域の値を表示するために、**Region** 列をデータシリーズとしてデータマッピングに追加することができます。そうすることで、次の図が得られます。



現在、各カテゴリには 3 つのデータポイントがあり、各データポイントはリージョンごとに 1 つずつあります。2D グラフの場合、シリーズは常にインラインで表示されます。3D グラフでは、シリーズはデフォルトで Z 軸に描画されますが、インラインで描画することもできます。以下は 3D での同様なチャートです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



このグラフでは、データシリーズは Z 軸に沿って描かれています。カテゴリの順序が変更され、データがよりよく表示されるようになりました。これがデータマッピングの基本概念です。ほとんどのチャートのタイプは、このようなマッピング手法やマッピングオプションを使用します。各グラフタイプの詳細なデータマッピング手順は、チャートの種類ごとのデータマッピングを参照してください。

Copyright(C) 2018 Climb.Inc.

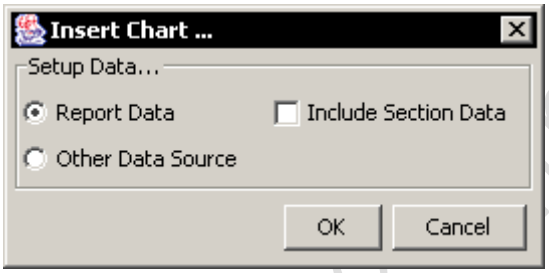
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.3 チャートの作成

グラフを作成するには、挿入メニューから **Insert Chart** を選択するか、ツールバーの **Insert Chart** アイコンを選択します。小さい矩形が、デザインウィンドウの周りのカーソルの後に表示されます。 グラフを追加する位置にカーソルを合わせ、チャートをクリックします。これにより、チャートのデータソースオプションを指定するダイアログが表示されます。

5.3.1 データソースオプション

チャートのデータソースを取得するには、次のオプションを使用できます。



チャートデータオプション

Report Data:

これにより、レポートに含まれるデータを使用してグラフがプロットされます。このデータには、レポート関数を使用して追加されたすべてのレポート列と計算列が含まれます。チャートのレポートデータを使用する場合は、セクションデータを含めることもできます。これは、チャートが配置されているレポートセクションから任意の数式またはラベルを取得し、チャートで使用できるようにします。ラベルと数式のデータは、チャートの入力データに追加の列としてインポートされます。セル ID が列名として使用されます。

ただし、レポートデータを使用すると、チャートをレポートとは別にデプロイすることはできず、時間ベースのズームやヒストグラムなどの特定のチャート機能は使用できません。

Other Data Source:

これにより、レポートデータとは異なるデータソースを選択することができます。このオプションを選択すると、グラフのデータソースを作成または選択できる **Data Source Manager** ウィンドウに戻ります。レポートの場合はできるだけ多くのソースのデータを使用できます。データソースの詳細については、Data Source の動作を参照してください。

独立したデータソースを使用すると、チャートは後でレポートから独立して配備できます。また、すべてのチャート機能が利用可能です。ドリルダウンなどの一部の機能は、チャートをレポートとは別にデプロイする場合にのみ機能することに注意してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.3.1.1 チャートパラメータリンク

チャートデータソースとしてパラメータ化されたクエリを使用する場合、デフォルトでチャートはレポートで実行されるときにデフォルト値で実行されます。ただし、チャートのチャートパラメータリンク機能を使用して、レポートのカラム値に基づいてパラメータ値を取得することができます。この機能は、リンクされたサブレポートで説明されているサブレポートリンク機能と同じ方法で機能します。パラメータ化されたグラフを列にリンクするには、まずパラメータ化されたグラフをレポートに挿入します。次に、**Data** メニューから **Chart Parameter Mapping** を選択します。これにより、レポートの列フィールドをグラフのクエリパラメータにマップできるダイアログが表示されます。

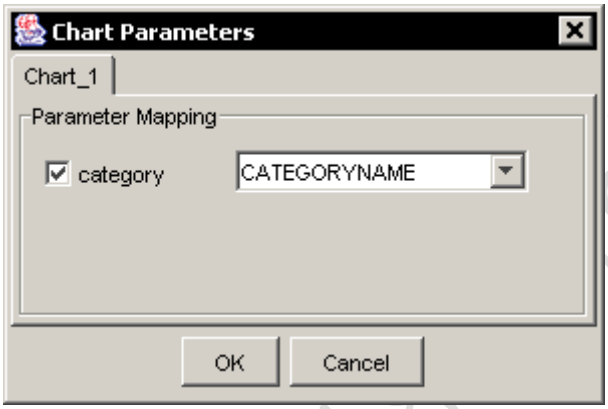


Chart Parameter Mapping ダイアログ

ダイアログには、パラメータ化された各チャートのタブが含まれています。グラフの各パラメータは、レポートの別の列フィールドにマップできます。

5.3.2 複数のデータソース

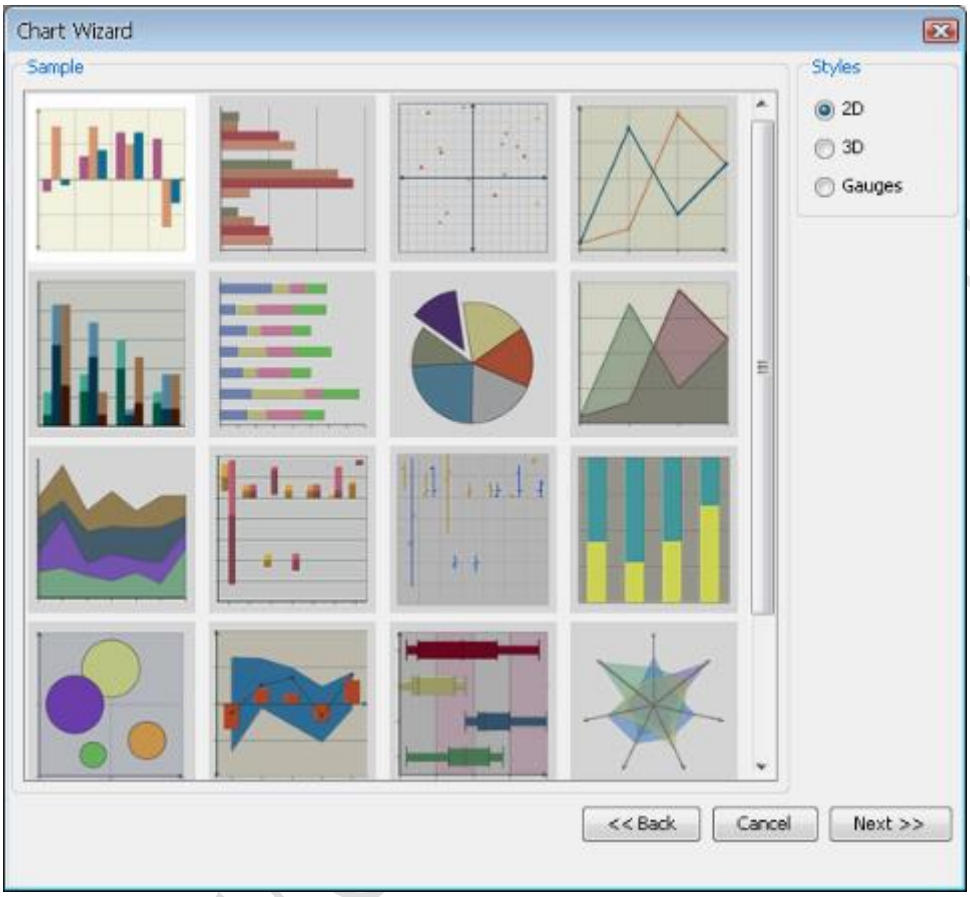
チャートに独立したデータソースを使用することを選択した場合は、複数のデータソースを使用して、レポートと同じようにコンポジットテーブルを構築できます。この機能は、レポートウィザードと同じ方法で機能します。詳細は、Salesforce のデータを参照してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.4 チャートの種類ごとのデータマッピング

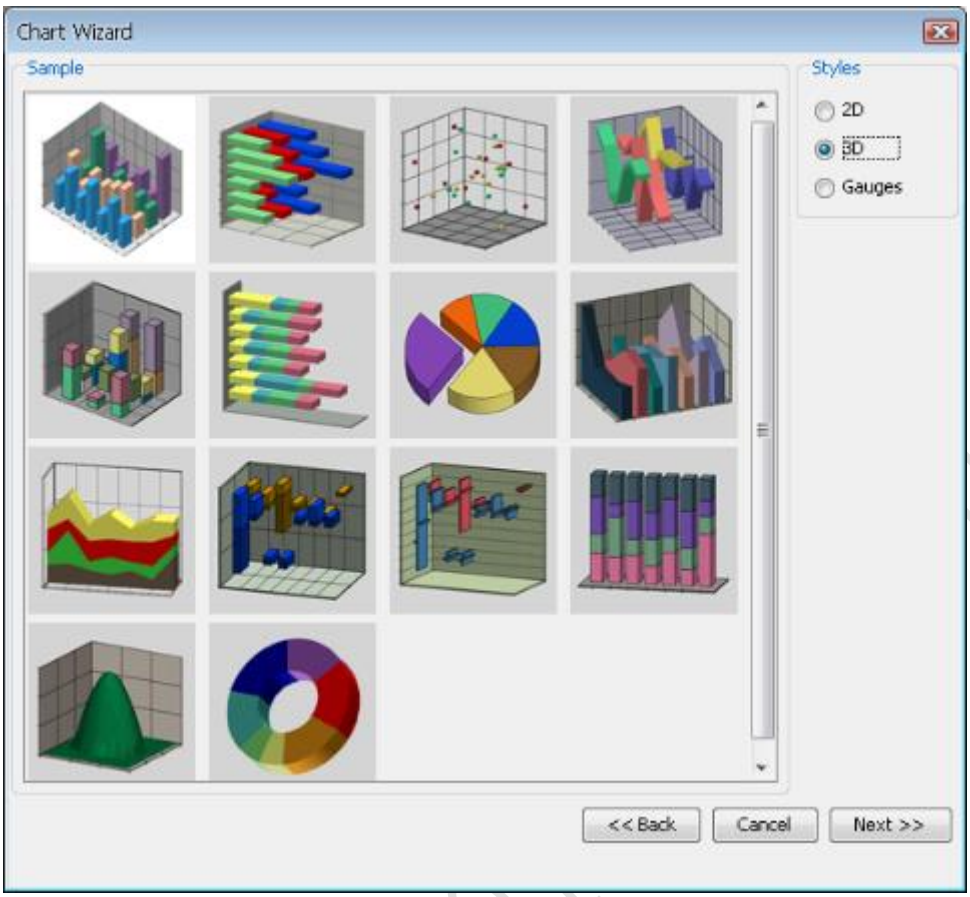
チャートに使用するデータを選択した後に、チャートウィザードの下記ステップで、使用するチャートタイプを選択します。グラフの種類を指定するためのダイアログが表示されます。



2D Chart Types Selection ダイアログ

各グラフタイプは、データポイントをプロットしてあらゆる種類のデータに適切な表現を与える異なる方法を表します。さまざまな種類のグラフが 2D または 3D に分類されています。基本的なチャートタイプに加えて、チャートに第 2 値/シリーズを追加することにより、さまざまな種類のコンポジット/コンビネーションチャートを作成できます。グラフの種類を切り替えるには、グラフの種類ダイアログの右側にある **2D**、**3D** または **Gauges** (ゲージ)のいずれかを選択します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



3D Chart Types Selection ダイアログ

このダイアログでは、イメージを選択して **Next** をクリックするか、チャートイメージをダブルクリックして、グラフの種類を選択できます。次のいずれかのチャートタイプから選択できます：

- 縦棒(Column)
- 横棒（Bar)
- 散布図（Scatter)
- 折れ線（Line)
- スタック縦棒（Stack Column)
- スタック横棒（Stack Bar)
- 円グラフ（Pie)
- エリアグラフ（Area)
- スタックエリア（Stack Area)
- High Low (High-Low)
- HLCO (HLCO)
- 100% (100% Column)
- ドーナツ（Doughnut)
- サーフェス（Surface) (3D のみ)
- バブル（Bubble) (2D のみ)
- オーバーレイ(Overlay) (2D のみ)
- ボックス（Box) (2D のみ)

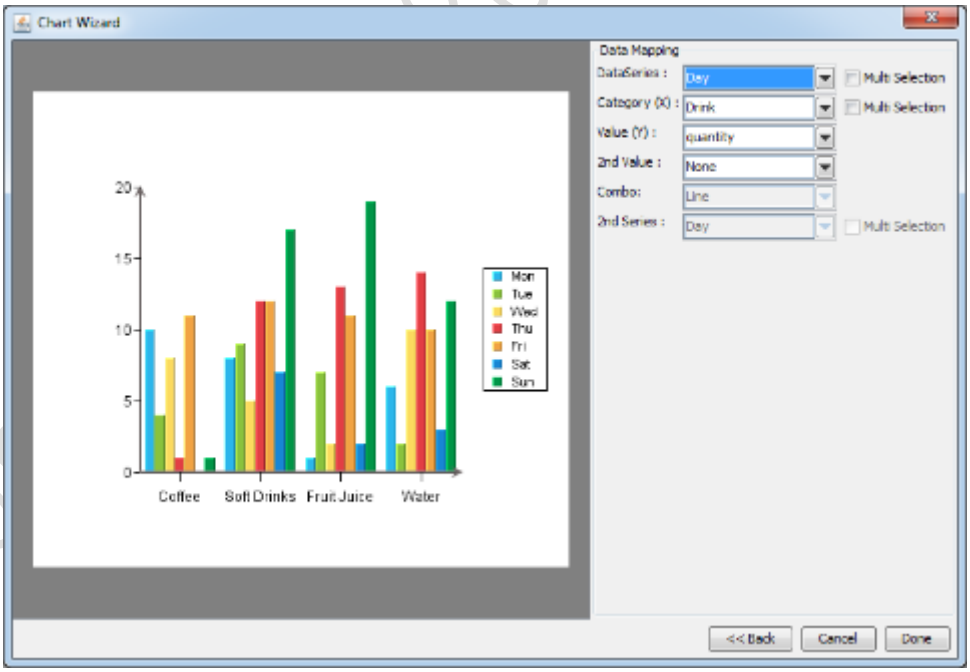
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- レーダー（Radar）(2D のみ)
- ダイヤル（Dial）(2D のみ)
- ガント（Gantt）(2D のみ)
- 極座標（Polar）(2D のみ)
- 円形ゲージ
- 正方形ゲージ
- 半円形ゲージ
- 正方形ゲージ
- 4 分割円形ゲージ

各グラフの種類については、この章の後半の縦棒グラフで詳しく説明します。
ゲージについては、ゲージを参照してください。

5.4.1 データマッピング

グラフの種類を選択した後に、グラフのデータマッピングを指定します。データマッピングは、選択したデータソースをグラフの各要素に対応付けします。データマッピングの基本については、基本的データマッピングで説明しています。ウィザードのデータマッピング画面では、マッピングオプションを設定することや、結果をプレビューすることができます。



Data Mapping ダイアログ

ダイアログの左側には、チャートのプレビューが表示されます。右側では、データソースのどの列が、グラフ要素（シリーズ、カテゴリ、値など）に対応してプロットするか選択できます。デフォルトで、EspressChart はデータ型に基づいて最初に使用可能な列を選択します。データマッピングの変更は簡単です。データフィールドの横にある下向き矢印をクリックし、別のフィールドを選択します。データ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

マッピングダイアログの左側のチャートプレビューは直ちに更新されます。

特定のマッピングオプションは、各グラフの種類によって異なります。この章の後半で、縦棒グラフから紹介しています。

5.4.1.1 データ変換

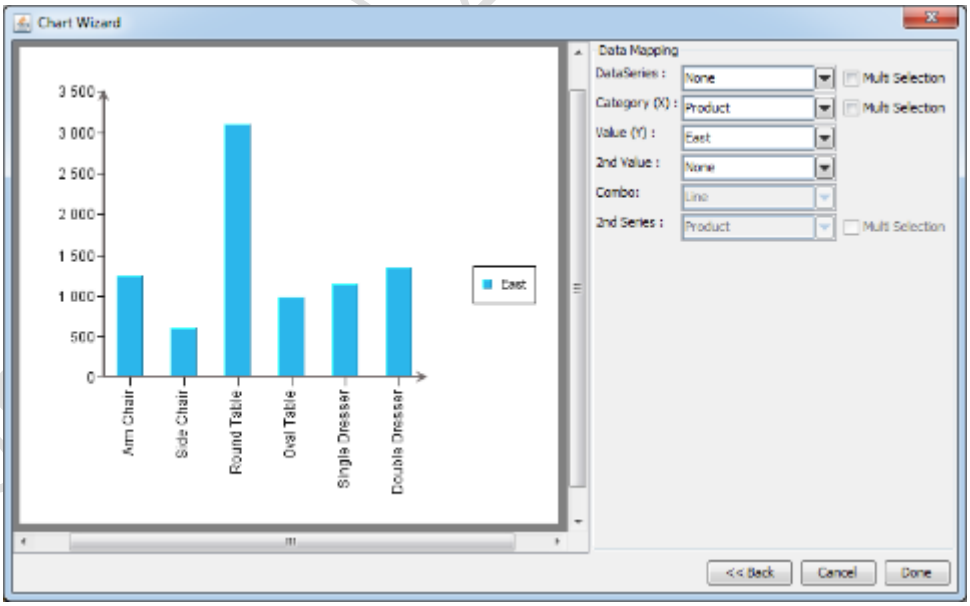
データマッピングダイアログからデータ変換を設定することもできます。変換を使用すると、データソースを変更せずに、複数のデータソース列を 1 つのグラフにプロットできます。

たとえば、次のようなデータソースがあります。

INDEX	Product	East	Midwest	West
TYPE	Varchar	Integer	Integer	Integer
1	Arm Chair	1241	2100	800
2	Side Chair	600	2940	1150
3	Round Table	3100	2500	2630
4	Oval Table	980	1660	1210
5	Single Dresser	1145	2340	1970
6	Double Dresser	1345	3560	1200

データソースの例

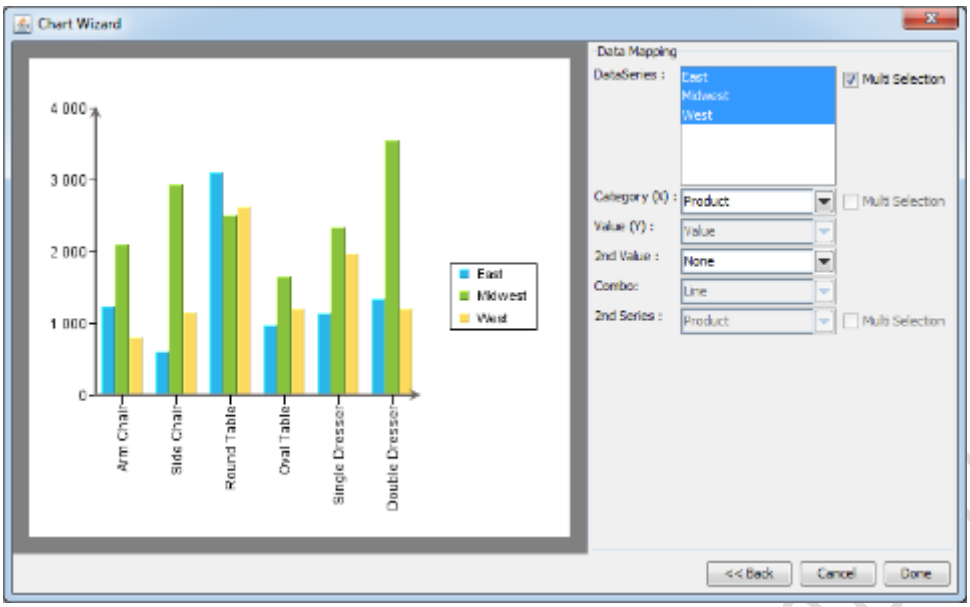
すべてのリージョンのデータを 1 つのチャートにプロットしたい。データ変換のないデフォルトマッピングでは、これを行うことはできません。グラフには 1 つのエリアのみをプロットするか、データソースを変更する必要があるため、データソースを変換する必要があります。



Default Chart Data マッピング

データを変換するには、**Data Series** フィールドの横にある **Multi Selection** オプションを選択します。フィールドはリストに変わり、同時に複数の列を選択することができます。すべての region 列を選択するだけで、チャートは次のようになります：

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

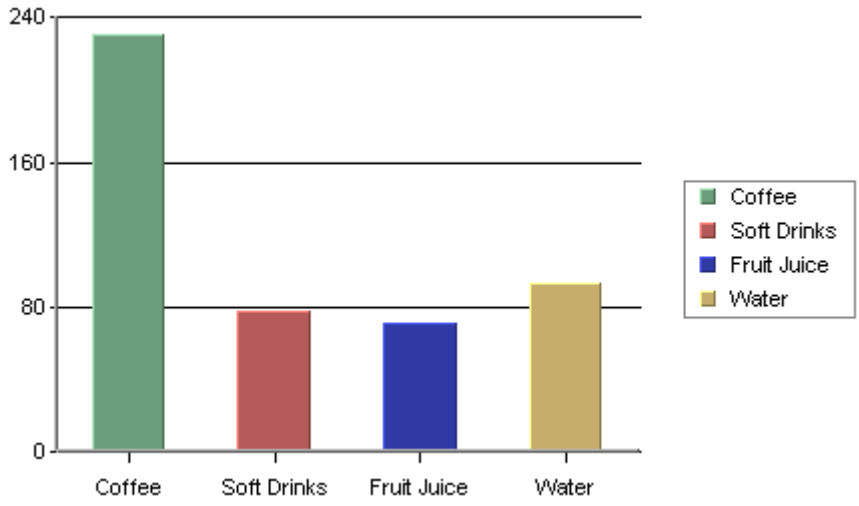


Default Chart Data マッピング

一度に 1 つのチャート要素（データシリーズ、カテゴリ、値、またはセカンドシリーズなど）のみを変換することができます。1つのチャート要素に対して **Multi Selection** チェックボックスを選択すると、他のチャート要素に対して **Multi Selection** チェックボックスが無効になります。

変換はドリルダウンチャートでは使用できません。

5.4.2 縦棒グラフ



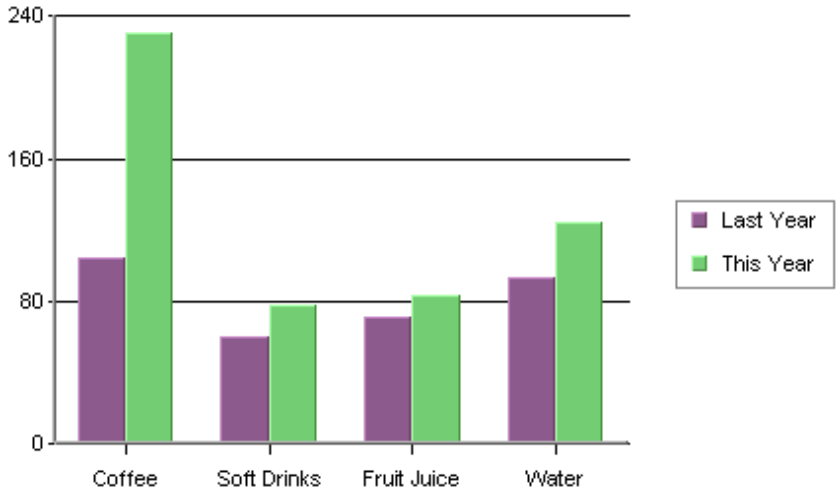
縦棒グラフ（データシリーズ無し）

縦棒グラフは、縦棒としてデータテーブルの各行を表示します。カテゴリは X 軸に沿ってリストされ、値は Y 軸に沿ってプロットされます。縦棒グラフは、異なるグループの離散値を比較するのに適しています。各グループは異なる色で表されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2D グラフでは、データシリーズが選択されている場合、1 つのカテゴリのシリーズ全体が XY 平面に表示されます。3D 縦棒グラフが使用されている場合、与えられたデータシリーズのすべての列は、Z 軸に沿って同じ色を使用して描画されます。グラフにデータシリーズがない場合、カテゴリはデフォルトで同じ色で表示されます。カテゴリの色を変更するには、ツールバーの **Change Chart options** アイコン をクリックし（または **Format** > **Chart Options** を選択）、**Single Color For All Categories** オプションのチェックを外します。

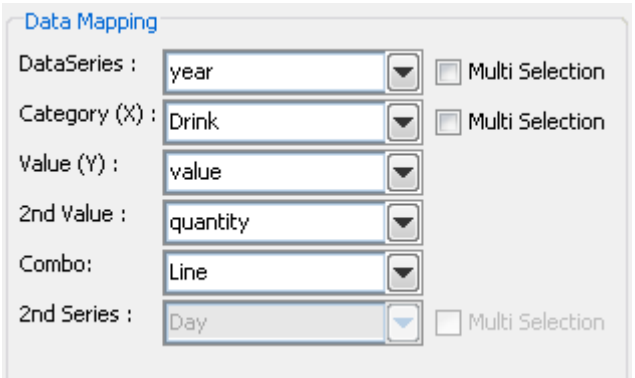
与えられた例では、カテゴリ変数として **Drink** を、値変数として **Value** を選択しました（ほとんどの例では、EspressChart インストールに含まれる **Sample.dat** ファイルが使用されています）。下のグラフのデータシリーズ変数として **Year** が選択されています。したがって、各名前には、データシリーズの列に存在する唯一の 2 つの値である **Last Year** と **This Year** の 2 つの縦棒が表示されます。



縦棒グラフ（データシリーズ有り）

5.4.2.1 データマッピング

縦棒グラフのデータマッピングは簡単です。基本的データマッピングで最初に示された例と似ています。縦棒グラフの場合、次のオプションが使用できます。



縦棒グラフのマッピングオプション

Data Series:

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

別個の値としてチャート内のデータシリーズ数を決定するデータシリーズを選択します。データシリーズの各要素は、同じ色セットと属性を使用して描画されます。

Category (X):
カテゴリを決定するデータ列を選択します。

Value (Y):
各カテゴリの値を提供するデータ列を選択します。

2nd value:
組み合わせチャートを作成するための 2 番目の値を追加します。

2nd Series:
セカンダリチャートのシリーズとなる別の列を選択します。このオプションは、セカンダリチャートが組み合わせチャートである場合にのみ適用されます。

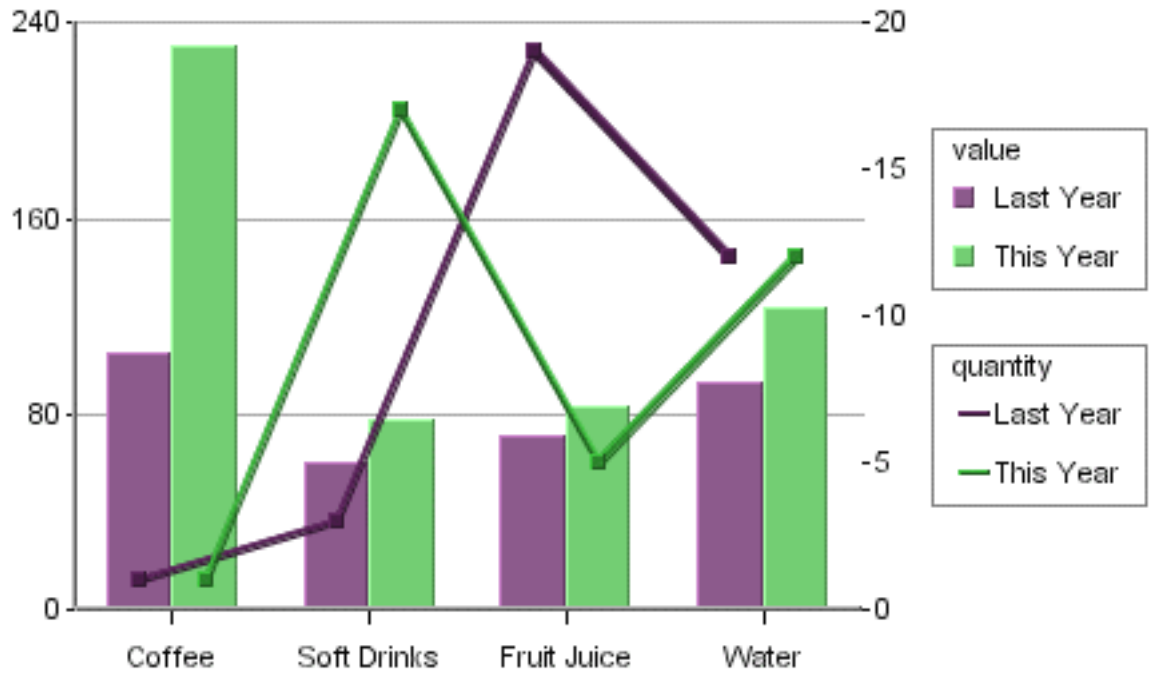
Combo:
セカンダリグラフのグラフの種類を選択します。縦棒グラフの場合、コンボオプションは折れ線グラフとオーバーレイチャートです。

データマッピングでは、データを変換することもできます (単一のカテゴリで複数の列を選択するなど)。データ変換の詳細については、データ変換を参照してください。

最後の 3 つのデータマッピングオプションを使用すると、グラフに第 2 値を追加できます。EspressChart は、円グラフ、レーダーチャート、バブルチャート、ダイヤルチャート、サーフェスチャート、散布図以外のすべてのグラフタイプで第 2 値をサポートしています。下の例では、第 2 値が縦棒グラフに追加されています。

Copyright(C) 2018 Climb.Inc.

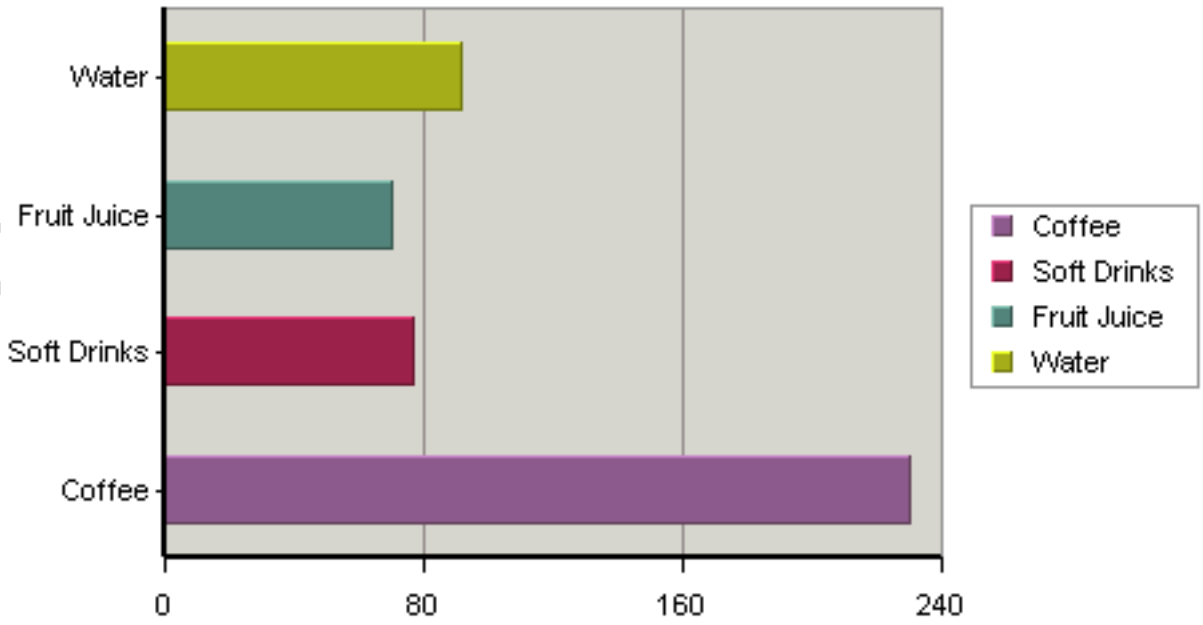
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



第 2 値付き縦棒グラフ

ご覧のように、第 2 軸のラベルは、プロットエリアの右側に描画されます（この軸を表示させることができます)。通常、第 2 値は第 1 値と同じカテゴリとシリーズを共有します。ただし、2D 縦棒、スタック縦棒グラフ、スタックエリアチャート、High Low チャート、HLCO チャート、100%スタック縦棒グラフでは、組み合わせチャートを指定できます。これにより、2 番目のシリーズを指定できます。オーバーレイチャートの詳細については、オーバーレイチャートを参照してください。

5.4.3 横棒グラフ



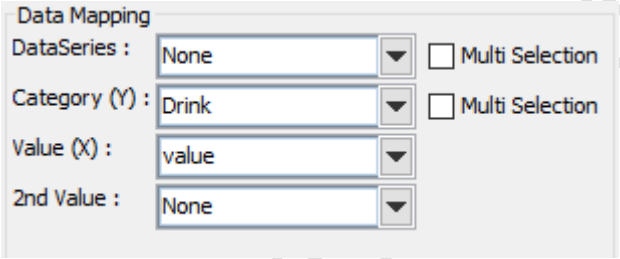
横棒グラフ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

横棒グラフは、基本的に縦棒グラフと同じですが、縦棒グラフで使用される縦棒とは対照的に、横棒がグラフに描かれています。横棒グラフでは、カテゴリは Y 軸に沿ってプロットされ、値は X 軸に沿ってプロットされます。

縦棒グラフの場合と同様に、データシリーズが選択されている場合は、単一カテゴリのシリーズ全体が XY 平面に表示されます。3D 棒グラフを使用している場合は、データシリーズ内のすべての水平棒が同じ色で描画されます。各カテゴリは異なる色で描画されます。チャートにデータシリーズが存在しない場合、カテゴリはデフォルトで同じ色で表示されます。カテゴリの色を変更するには、ツールバーの **Change Chart options** アイコン をクリック（または **Format** > **Chart Options** を選択）し、**Single Color For All Categories** オプションのチェックを外します。

5.4.3.1 データマッピング



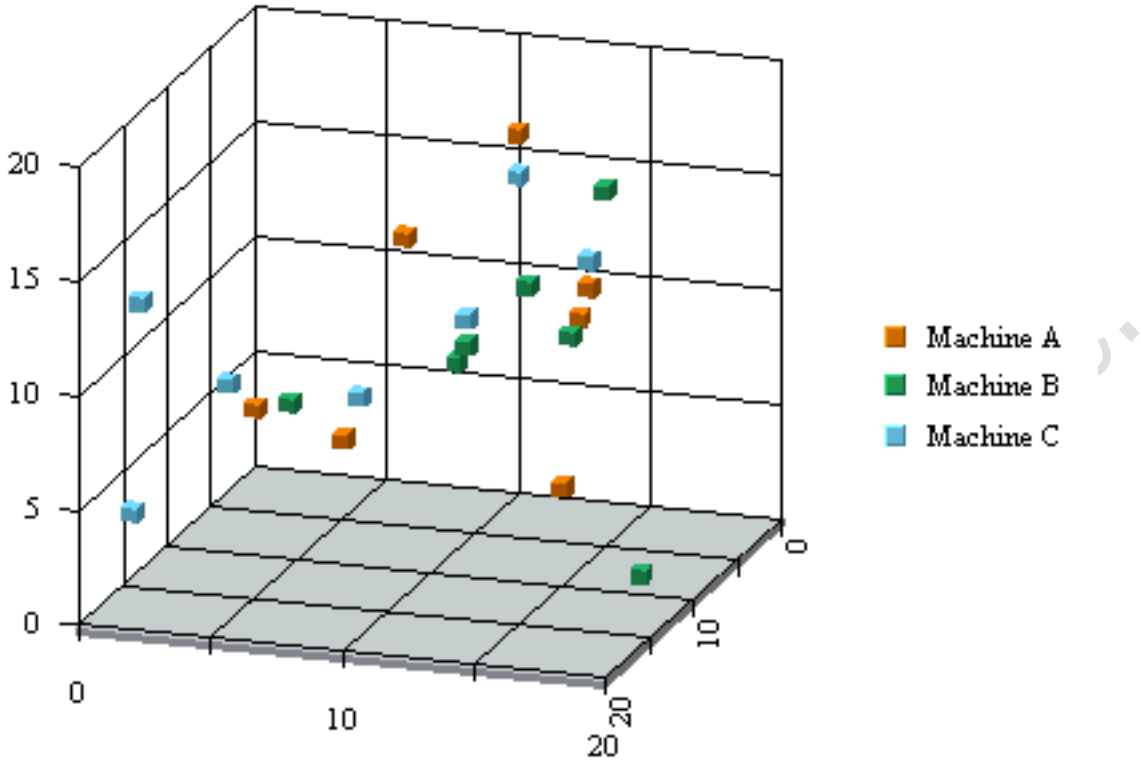
横棒グラフのマッピングオプション

このチャートタイプのマッピングは、縦棒グラフのカテゴリ（X）と値（Y）がそれぞれカテゴリ（Y）と値（X）になる点を除いて、縦棒グラフのマッピングと同様です。これは、値が縦棒グラフでは縦に、横棒グラフでは横に表示されるためです。第 2 シリーズとコンボオプションは横棒グラフでは使用できません。これは、横棒グラフで使用できる唯一の組み合わせが折れ線グラフであるためです。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.4.4 XY(Z)散布図

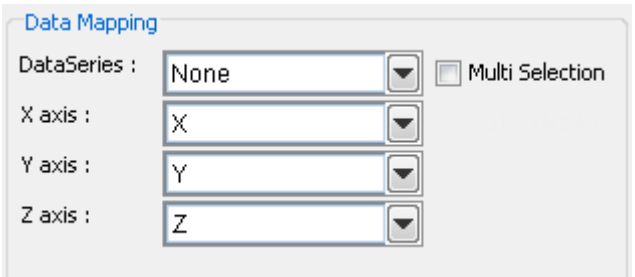


XY(Z)散布図

XY（Z）散布図では、データテーブル内の選択された各行は、2D または 3D 空間内の点を定義します。したがって、各列には数値または日付/時刻値が含まれていなければなりません。マーカーは各点を表します。データテーブルの各行にあるデータ列は、マーカーの空間的位置を決定します。

必要に応じて、別のデータテーブル列を選択して、マーカーをグループに分けることができます。各グループの要素は、データ列と呼ばれるこの列に同じ値を持ちます。同じグループ内のマーカーは、同じ描画属性を使用して描画されます。言い換えれば、同じ形と色を使用しています。散布図の X 軸スケールは線形です。これは、他のチャートタイプとは異なり、データポイントが X 軸に沿って均等に配置されている場合とされていない場合があります。

5.4.4.1 データマッピング



散布図のマッピングオプション

散布図では、X 軸、Y 軸、Z 軸の値によって点の X、Y、Z 座標がそれぞれ決定されます。データシリーズボックスでは、チャート内のデータシリーズ数を区別する値を持つデータ列を選択できます。データ

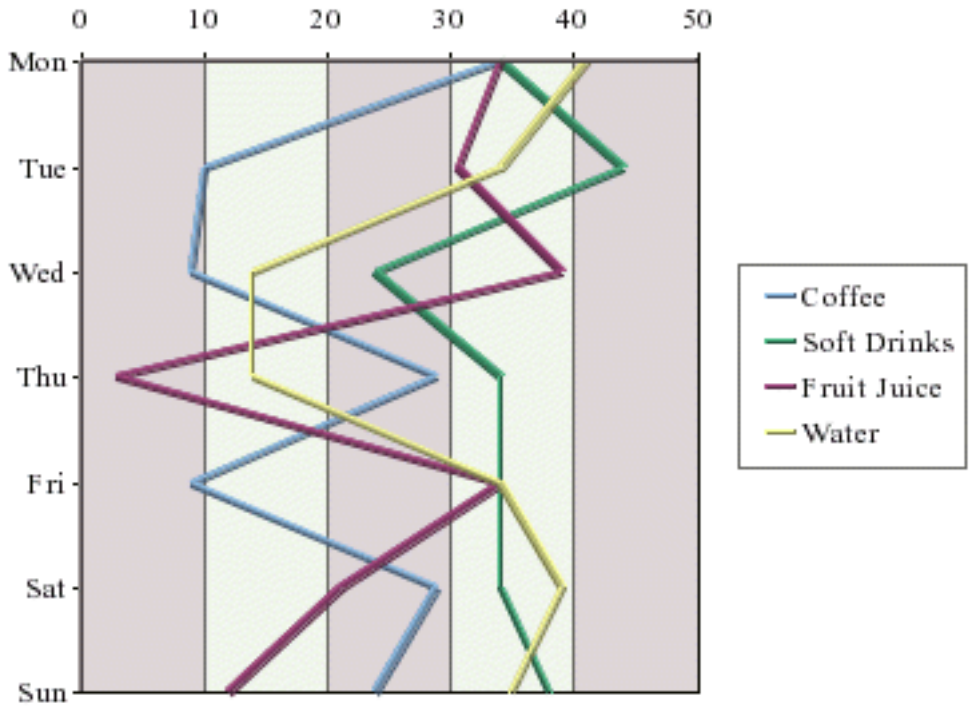
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

シリーズの各要素は、同じ描画属性セット（色やマーカーなど）を使用して描画されます。

5.4.5 折れ線グラフ

折れ線グラフは、縦棒グラフと同様の方法でデータを表示します。2D 折れ線グラフの場合、マーカーは各カテゴリのデータポイントを示します。値列は数値でなければならず、マーカーの高さを決定します。各マーカーは線で結合されています。グラフにデータシリーズがある場合は、シリーズ内の各要素に対して別々の線が描画されます。マーカー（ポイント）はデフォルトでは表示されませんのでご注意ください。**Line and Point** ダイアログでマーカー（ポイント）を表示することができます。このダイアログは、**Format** メニューから開くか、またはツールバーの **Format line and points** アイコン をクリックして開くことができます。

EspressChart では、デフォルトの水平設定に加えて、2D 折れ線グラフを垂直方向に表示することができます。チャートデザイナでこの機能を使用するには、折れ線グラフを作成し、**Format** > **Chart Options** を選択します。次に、**Vertical** を選択します。チャートは時計回りに 90 度回転します。

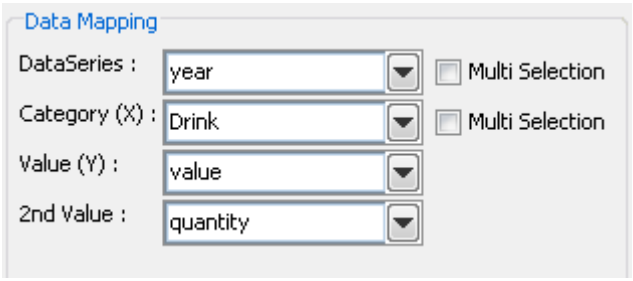


垂直折れ線グラフ

3D 折れ線グラフは、2D の対応部分の拡張です。追加情報はありません。マーカーは消えます（3D 折れ線グラフでは使用できません）。また、Z 軸上に間隔を置いて太い線を置きます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

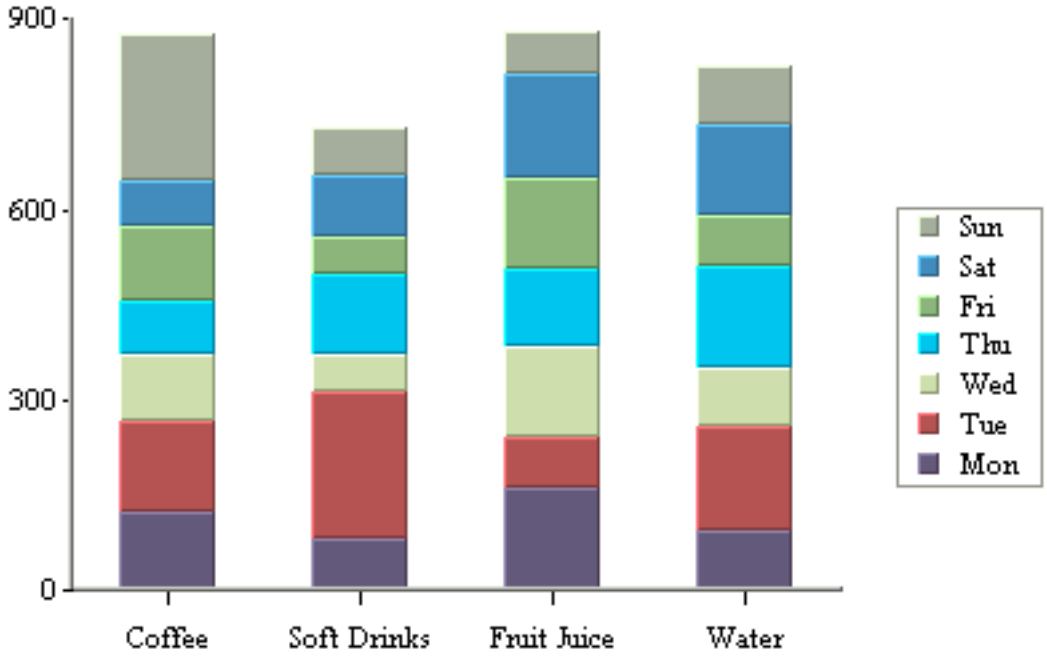
5.4.5.1 データマッピング



折れ線グラフのマッピングオプション

折れ線グラフのデータマッピングは、縦棒グラフとほぼ同じです (縦棒グラフのデータマッピング参照)。ただし、折れ線グラフには第 2 シリーズとコンボオプションはありません。これは、折れ線グラフで使用できる唯一の組み合わせが折れ線であるためです。他のグラフタイプ（棒、柱、スタック横棒など）と折れ線の組み合わせを作成するには、他のグラフの種類を主グラフとして選択し、折れ線をコンボオプションとして選択します。

5.4.6 スタック縦棒グラフ



スタック縦棒グラフ

スタック横棒グラフは縦棒の各柱が棒の束を構成する縦棒グラフの派生です。データテーブル内の選択された各行は、チャート縦棒のコンポーネントで表されます。2D スタック横棒グラフの場合、カテゴリは X 軸にプロットされ、Y 軸には値がプロットされます。数値でなければならない value 列は、各棒成分の長さを決定します。合計列と呼ばれる第 3 列は、各カテゴリの値のグループを表します。指定されたカテゴリ値の積み上げが、そのカテゴリ値の合計値を積み重ねて構築されます。チャート縦棒の各積み上げコンポーネントには、別個の色で示される別個の合計値があります。データテーブルの各行には、カテゴリと合計列の両方で固有の値のペアが必要です。負の値を持つコンポーネントは、正の値を

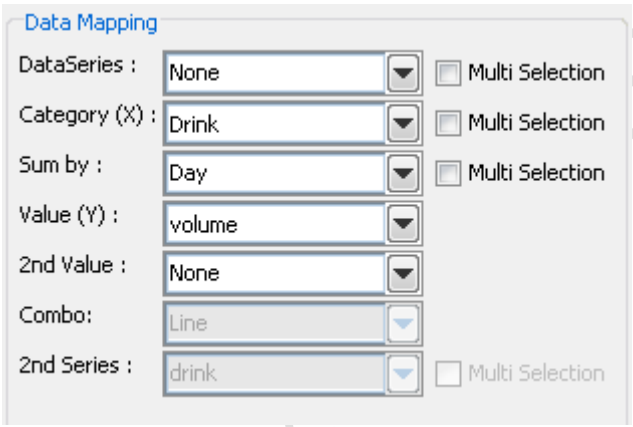
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

持つコンポーネントから分離され、X 軸の下のマイナス方向に積み上げされます。

上記の例では、カテゴリ値と合計値を表す日として飲料タイプを選択しました。別の独立したカテゴリ（Z 軸上に表示される）は、データシリーズとして第 4 の列に基づいて任意に指定することができる。このような場合、データテーブルの各行は、カテゴリ、合計、およびデータシリーズの列に一意の 3 つの値を持たなければなりません。2D チャートでは、1 つのカテゴリのデータシリーズ全体が XY 平面に並べて表示されます。3D スタック横棒グラフでは、データシリーズが Z 軸に沿って表示されます。

5.4.6.1 データマッピング

スタック縦棒グラフでは、各カテゴリはさらにコンポーネントに細分されています。したがって、チャートの合計値を決定するために使用される sum-by オプションが追加されています。



スタック縦棒グラフのマッピングオプション

データマッピングのオプションは次のとおりです：

Data Series:

別個の値がチャート内のデータシリーズ数を決定するデータ列を選択します。データシリーズの各要素は、同じ色のセットと属性を使用して描画されます。

Category (X):

異なる値によってカテゴリが決定されるデータ列を選択できます。このデータ列の値は、X 軸を較正するために使用されます。データシリーズの各カテゴリは、チャート内の個別の縦棒として描画されます。

Sum-by:

個別の値によって各カテゴリのコンポーネントが決定されるデータ列を選択できます。この列の個々の値によって、縦棒の個別のスタックコンポーネントが決まります。

Value (Y):

各カテゴリの値を提供するデータ列を選択します。

2nd value:

オーバーレイチャートを作成するための 2 番目の値を追加します。

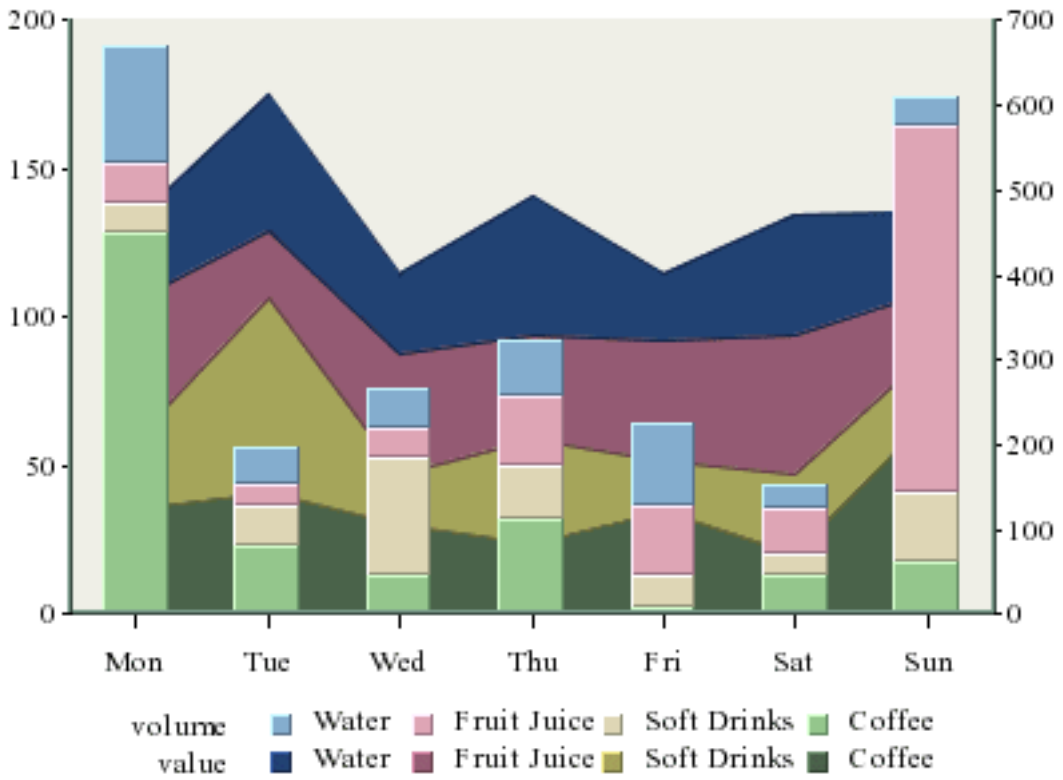
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2nd Series:
セカンダリチャートのシリーズとなる別の列を選択します。このオプションは、セカンリチャートがオーバーレイチャートである場合にのみ適用されます。

Combo:
セカンダリグラフのグラフの種類を選択します。スタック縦棒グラフの場合、コンボオプションは折れ線、スタックエリア、およびオーバーレイです。

データマッピングでは、データを変換することもできます（言い換えれば、単一のカテゴリの複数の列を選択すること）。データ変換の詳細については、データ変換を参照してください。

スタック縦棒グラフには、ユーザがスタックエリアチャートとして第 2 値の値をプロットするためのユニークな組み合わせオプションがあります。両方の値は同じカテゴリと合計値を共有します。このチャートは、データの異なる値に対するコンポーネントベースのカテゴリを比較できます。

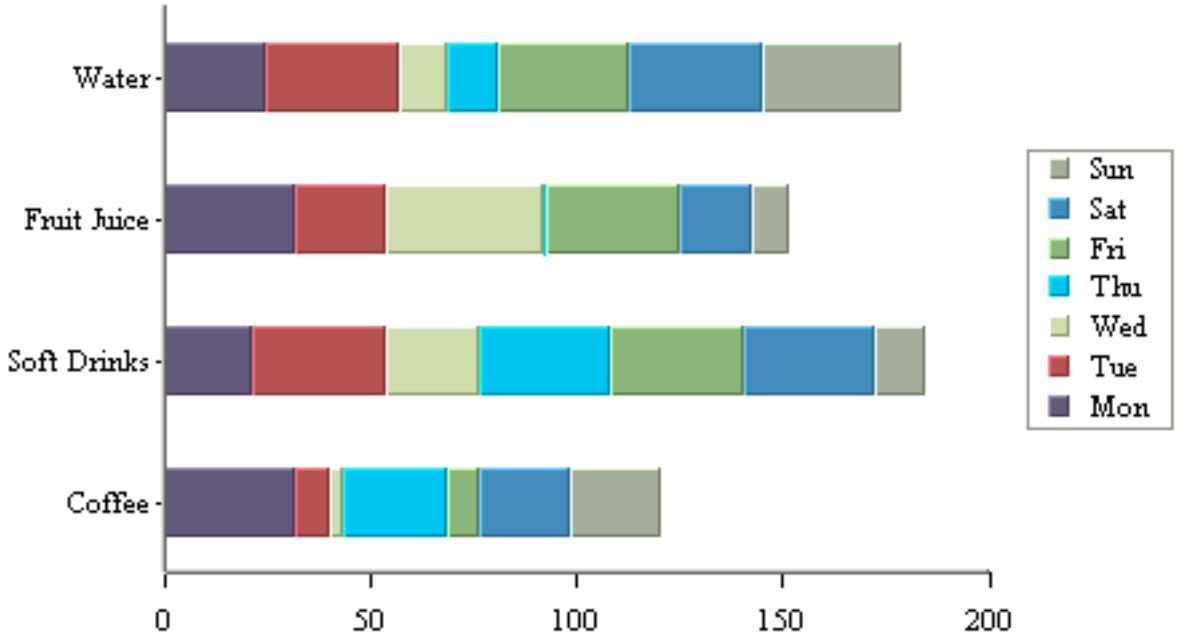


スタック縦棒グラフ-スタックエリアチャートのオーバーレイチャート

このコンビネーションチャートでは、一方のチャートをできるだけ隠さないように、もう一方のチャートが配置されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

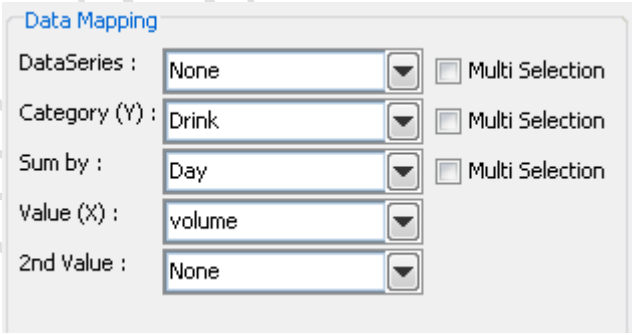
5.4.7 スタック横棒グラフ



スタック横棒グラフ

スタック縦棒グラフと同様に、スタック横棒グラフは、異なる積み重ねられた値の合計（データソース内の合計列）としてチャートカテゴリを表示します。スタック横棒グラフの違いは、カテゴリが Y 軸にプロットされ、値が X 軸に沿ってプロットされていることです。

5.4.7.1 データマッピング

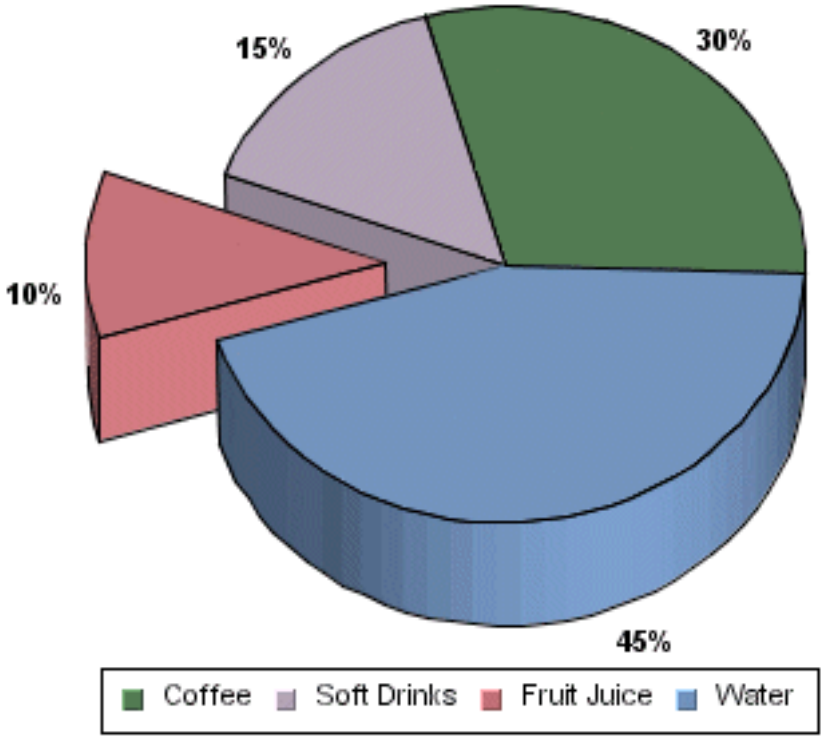


スタック横棒グラフのデータマッピングオプション

このグラフタイプのマッピングは、縦棒グラフの下のカテゴリ（X）と値（Y）がそれぞれカテゴリ（Y）と値（X）になる点を除き、スタック縦棒グラフのマッピングに似ています。これは、値が縦棒グラフでは縦に、棒グラフでは横に表示されるためです。セカンダリシリーズとコンボオプションはスタック横棒グラフでは使用できません。これは、スタック横棒グラフで使用できる唯一の組み合わせが折れ線グラフであるためです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.4.8 円グラフ



円グラフ

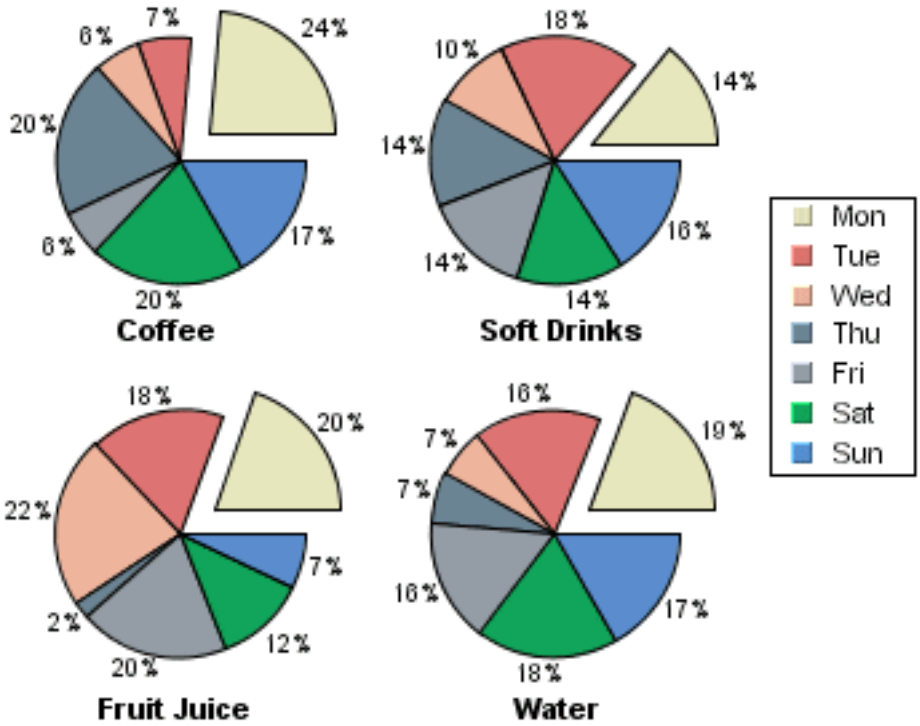
円グラフでは、データの各行は扇形として表されます。円グラフには、カテゴリ列と値列が必要です。値の列は数値でなければなりません。カテゴリ列内の種類の数によって、グラフ内の扇形の数が決まります。すべての値が合計され、各スライスに合計のシェアに応じた角度が割り当てられます。したがって、各カテゴリの値の列は、そのカテゴリを表すスライスのサイズを決定します。

3D 円グラフは、単に 2D 円グラフの拡張であり、新たな情報は含まれません。

円グラフにはデータシリーズも指定できます。データマッピングでシリーズを指定すると、シリーズ要素で構成される各カテゴリ要素で別々の円が描画されます。データシリーズの円グラフは次のように表示されます。

Copyright(C) 2018 Climb.Inc.

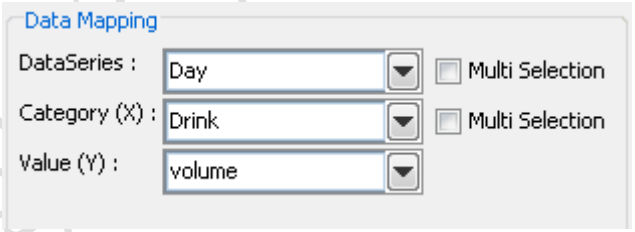
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



データシリーズによる円グラフ

シリーズが存在する場合、それぞれの円は 1 つのプロットエリアに描画され、プロットに比例してサイズが変更されます。各円の配置は一つの行にいくつ並べるかで指定が可能です。

5.4.8.1 データマッピング



円グラフのマッピングオプション

円グラフの場合、マッピングは次のようになります。

Data Series:
チャート内のデータシリーズ数を区別する値を持つデータ列を選択できます。データシリーズの各要素は同じ色で描画されます。

Category (X):
異なる値がさまざまなカテゴリを決定するデータ列を選択できます。

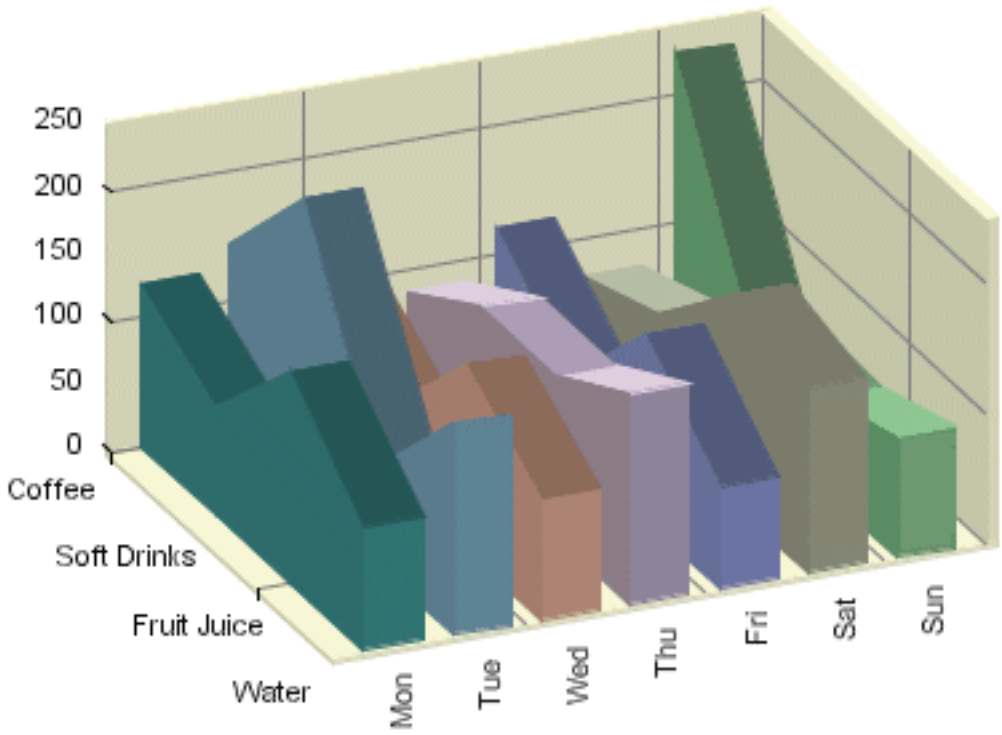
Value (Y):
各カテゴリの値を提供するデータ列を選択できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

円グラフを組み合わせたチャートは作成できないため、第 2 の値オプションはありません。

データマッピングでは、データを変換することもできます (単一のカテゴリの複数の列を選択すること)。データ変換の詳細については、データマッピングのデータ変換を参照してください。

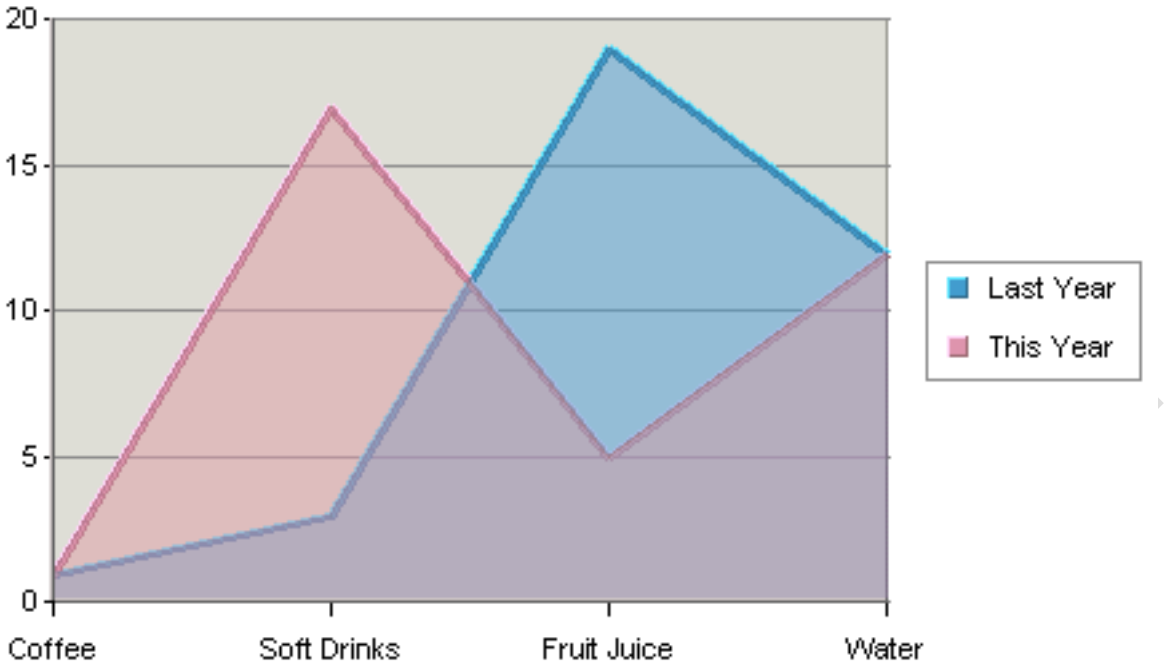
5.4.9 エリアチャート



3D エリアグラフ

3D エリアグラフは、縦棒グラフの派生品とみなすことができます。3D エリアチャートは、以下のようにして 3D の柱状図から構成することができます。データシリーズ（Z 軸）の値ごとに、すべての列の上端が太線で結合されています。列が削除され、各行の下のエリア（XY 平面内）が別の色で塗りつぶされます。

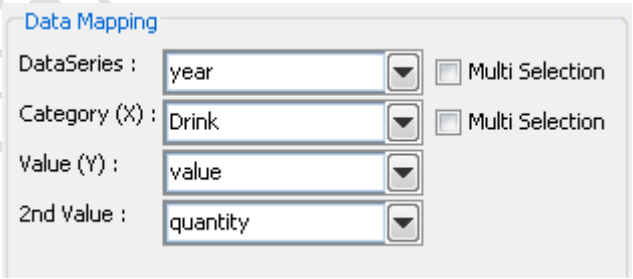
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



2D エリアグラフ

2D エリアグラフは、基本的に、Z 軸に沿って表示される 3D の対応物の投影です。結果として、2D エリアグラフは、データシリーズ値を完全に隠す場合があります。そのため、注意して使用する必要があります。２D ディスプレイでは、各シリーズの一部が一連の正面によって隠されています。この問題を改善するには、上記の例のように、データシリーズの順序を変更（データの順序付けを参照）や、エリアを半透明に設定します（フォーマットメニューを参照)。

5.4.9.1 データマッピング

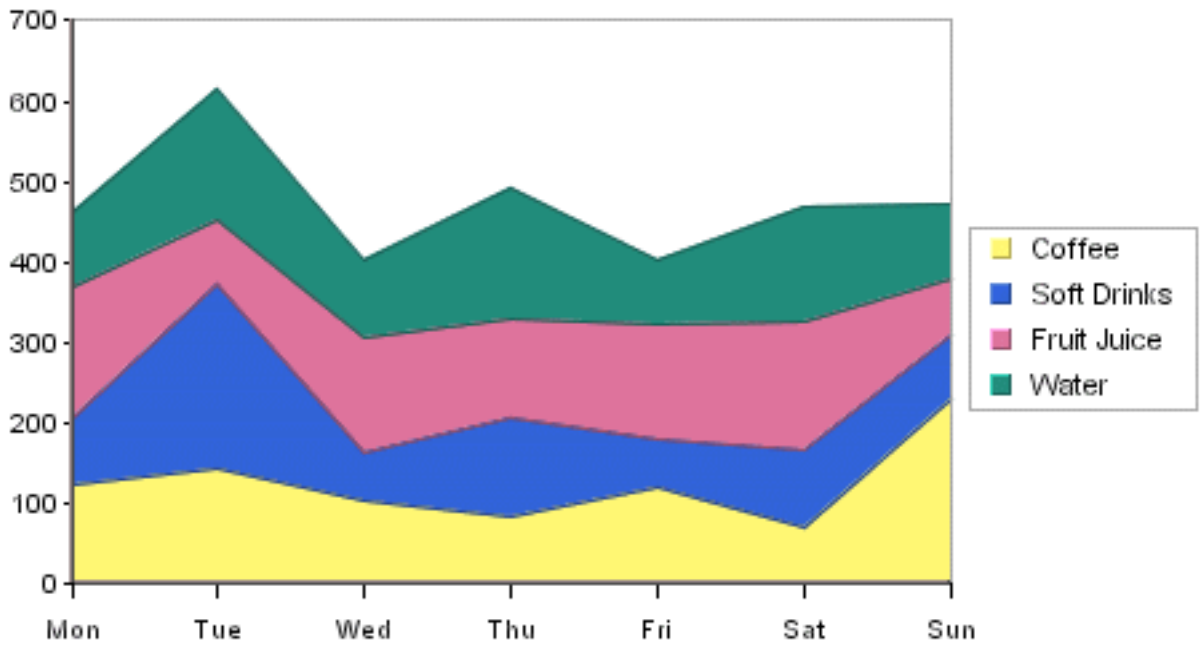


エリアグラフのマッピングオプション

エリアグラフのデータマッピングは、縦棒グラフのチャートとほぼ同じです（データマッピングで説明しています)。ただし、エリアグラフには第 2 シリーズとコンボオプションはありません。これは、エリアグラフで使用できる唯一の組み合わせが折れ線グラフであるためです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.4.10 スタックエリアチャート



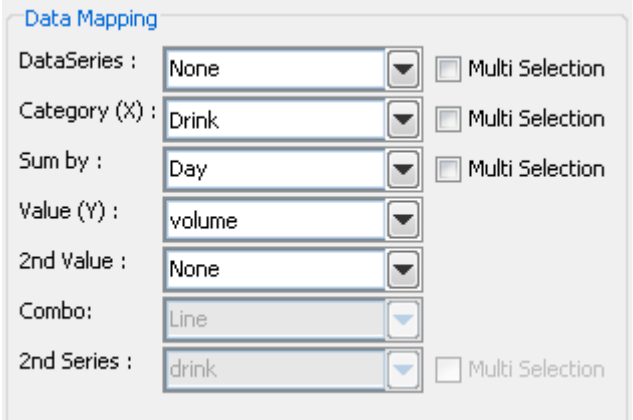
スタックエリアチャート

２D スタックエリアチャートは、2D スタック縦棒グラフの派生物ですが、データシリーズはありません。このチャートは、次のようにスタック縦棒グラフから作成できます。同じ色を持つ各スタックコンポーネントの上端が、線で結合されています。スタック縦棒が削除され、各行のセット間のエリアが別の色で塗りつぶされます。

3D スタックエリアチャートは、データシリーズを有することができ、上述のプロセスを使用して 3D スタック縦棒チャートから導出することもできます。この場合、個別のデータシリーズ値（固定 Z 軸値）ごとの積み上げ列が別々に結合され、複数の積み上げが生成されます。

2D/3D スタックエリアチャートは、それぞれスタック縦棒チャートの対応に必要とされる同様のデータセットを使用して構築することができます。先述した、2D スタックエリアチャートにはデータシリーズを持たないことに注意してください。

5.4.10.1 データマッピング

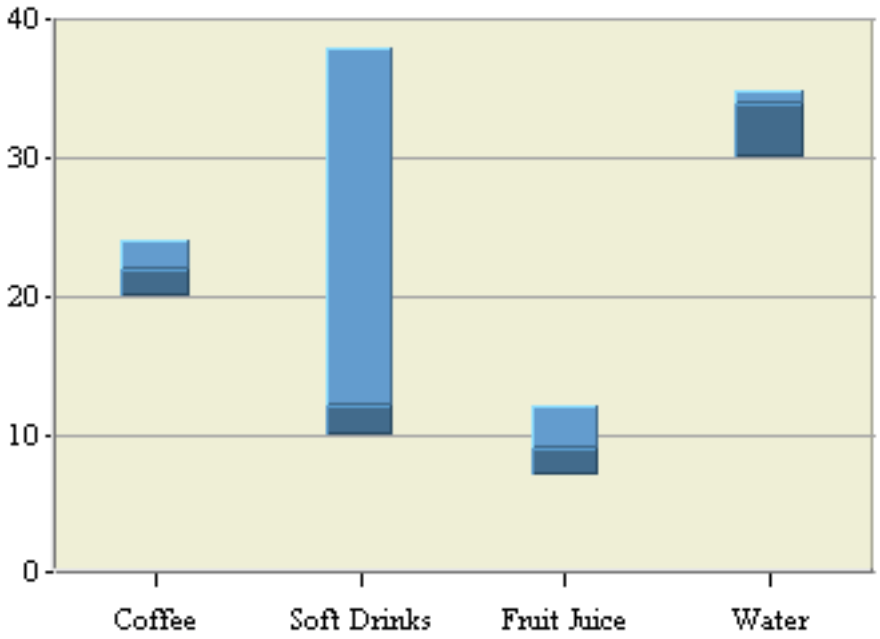


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

スタックエリアチャートのマッピングオプション

スタックエリアチャートのデータマッピングは、スタック縦棒グラフ（スタック縦棒グラフのデータマッピングで説明）とほぼ同じです。ただし、2D スタックチャートはデータシリーズを持つことができず、コンボオプションは**折れ線（Line）**と**オーバーレイ（Overlay）**です。

5.4.11 High-Low チャート



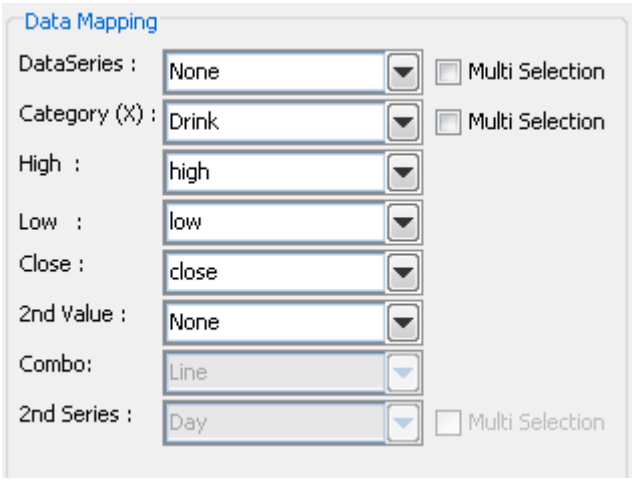
HighLow チャート

HighLow チャートは、棒グラフの派生系でもあります。一つの値の列の代わりに、High と Low の 2 つの列を使用します。グラフのデータには、カテゴリ、高値、安値の少なくとも 3 つの列が含まれている必要があります。カテゴリ列には任意の型の値を含めることができますが、High と Low の列は数値でなければなりません。

HighLow チャートの別のオプションは、クローズ列と呼ばれるデータ列です。クローズ列も使用されている場合、クローズ値は High 値と Low 値の間になります。Low 点とクローズ点との間の部分は、クローズ点と High 点との間の部分と同じ色のより暗い形態でレンダリングされます。他の多くのタイプのチャートと同様に、HighLow チャートにはデータシリーズの列が含まれている場合があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

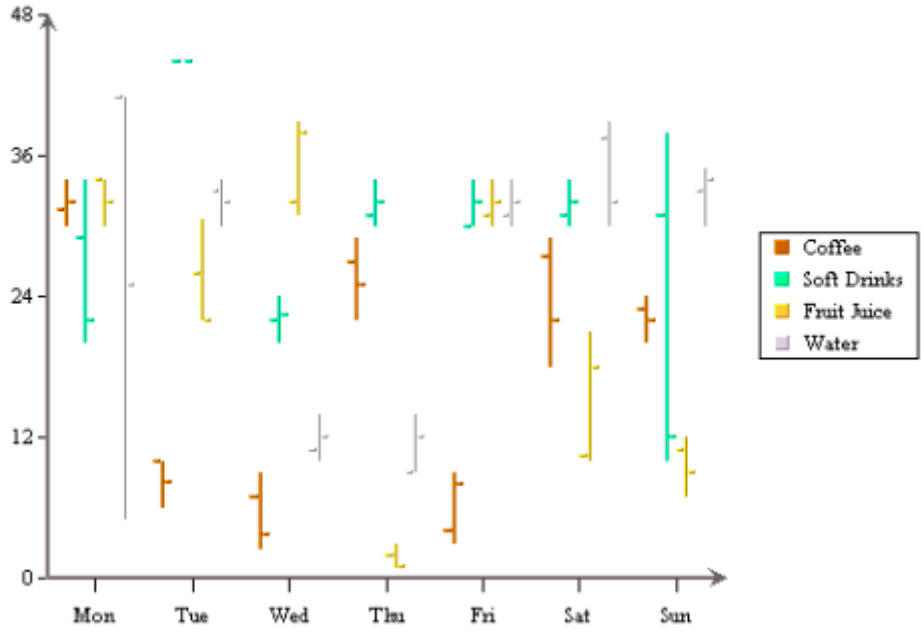
5.4.11.1 データマッピング



HighLow チャートのマッピングオプション

HighLow チャートのデータマッピングは、縦棒グラフに似ており、同じ方法でシリーズ列とカテゴリ列を選択できます。HighLow チャートの違いは、少なくとも 2 つの値の列、高値と安値を指定する必要があることです。これらは、各カテゴリにプロットされた 2 つのポイントです。**Close** と呼ばれる 3 番目の値も指定できます。HighLow チャートのコンボオプションは、**折れ線（Line）、縦棒（Column）、オーバーレイ（Overlay）**です。

5.4.12 HLCO チャート



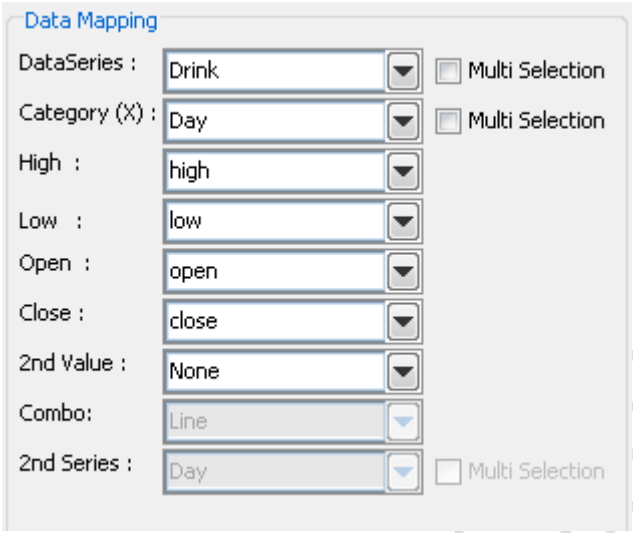
HLCO チャート

HLCO（High Low Close Open）チャートは、前述の HighLow チャートに似ていますが、**Open** 列も含まれています。これは株価や日中の気温のような不連続な期間に亘って変動するデータを提示するのに

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

有用です。HLCO チャートは、2D チャートと 3D チャートの両方に対してろうそく足形式で表示することもできます。この機能については、デザイナーインタフェースで説明しています。

5.4.12.1 データマッピング



HLCO チャートのマッピングオプション

HLCO チャートのデータマッピングは、HighLow チャートのデータマッピングに似ています。唯一の違いは、2 つの異なる値の列を指定する代わりに、4 つの異なる列（**High**, **Low**, **Open**, **Close**）を指定する必要があることです。コンボオプションには、**折れ線**（**Line**）、**縦棒**（**Column**）、**オーバーレイ**（**Overlay**）があります。

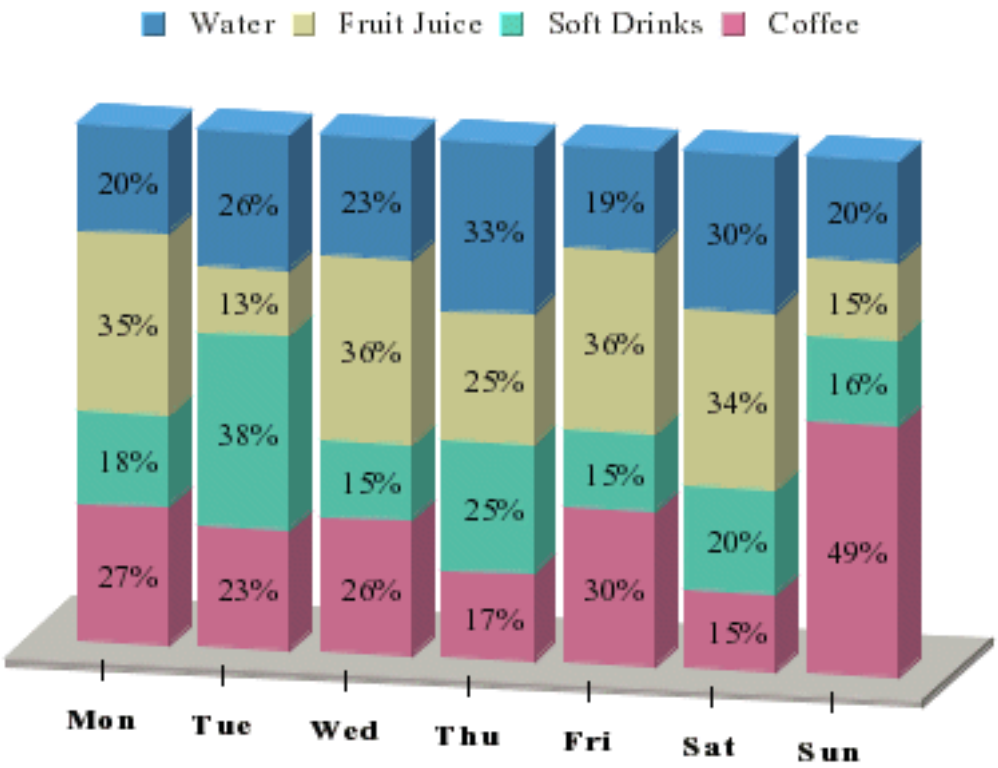
一般的な種類のチャートは、同じチャートで株価と出来高の両方をプロットしたものです。これは、EspressChart の HLCO 列の組み合わせを使用して実行できます。

HLCO-カラムコンビネーションチャート

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

この例では、株価と出来高データを 3 か月間にわたってプロットしています。

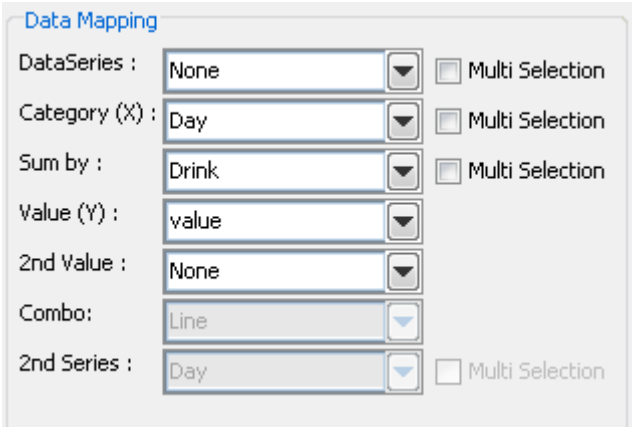
5.4.13 100%スタック横棒グラフ



100%スタック横棒グラフ

100%スタック横棒グラフは、円グラフと縦棒グラフを派生させたものです。グラフの各列は 1 つの円に対応しています。パーセンテージ列チャートには、X 軸値に対応するカテゴリ列があります。 この例では、sum-by 列がカテゴリを表し、value 列は円グラフの列と同じです。

5.4.13.1 データマッピング



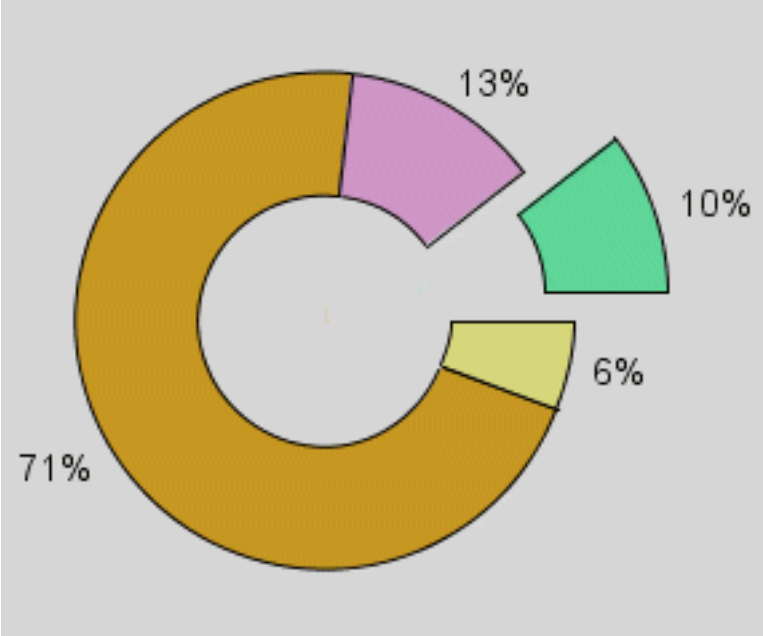
100%スタック横棒グラフのマッピングオプション

100%スタック横棒グラフのデータマッピングは、スタック縦棒チャート（データマッピングでの説明）

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

と同じです。唯一の違いは、合計コンポーネントが離散値ではなく、値列のパーセンテージとしてプロットされていることです。100%スタック横棒グラフのコンボオプションは、**折れ線（Line）**と**オーバーレイ（Overlay）**です。

5.4.14 ドーナツチャート

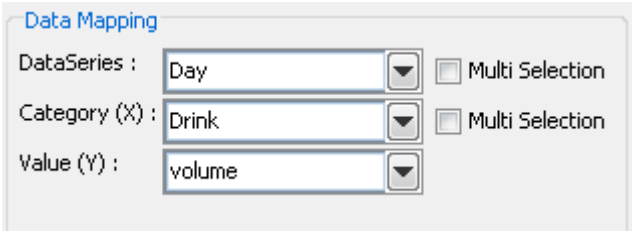


ドーナツグラフ

ドーナツグラフは中央の「穴」を除いて円グラフと同じです。

円グラフの場合と同様に、データシリーズが選択されると、各カテゴリ要素に対して個別のドーナツが描画され、各ドーナツはシリーズ要素で構成されます。シリーズが存在する場合、すべての別々のドーナツが単一のプロットエリアに描画され、プロットに比例してサイズが変更されます。異なるドーナツの配置は一つの行にいくつ並べるかで指定が可能です。

5.4.14.1 データマッピング

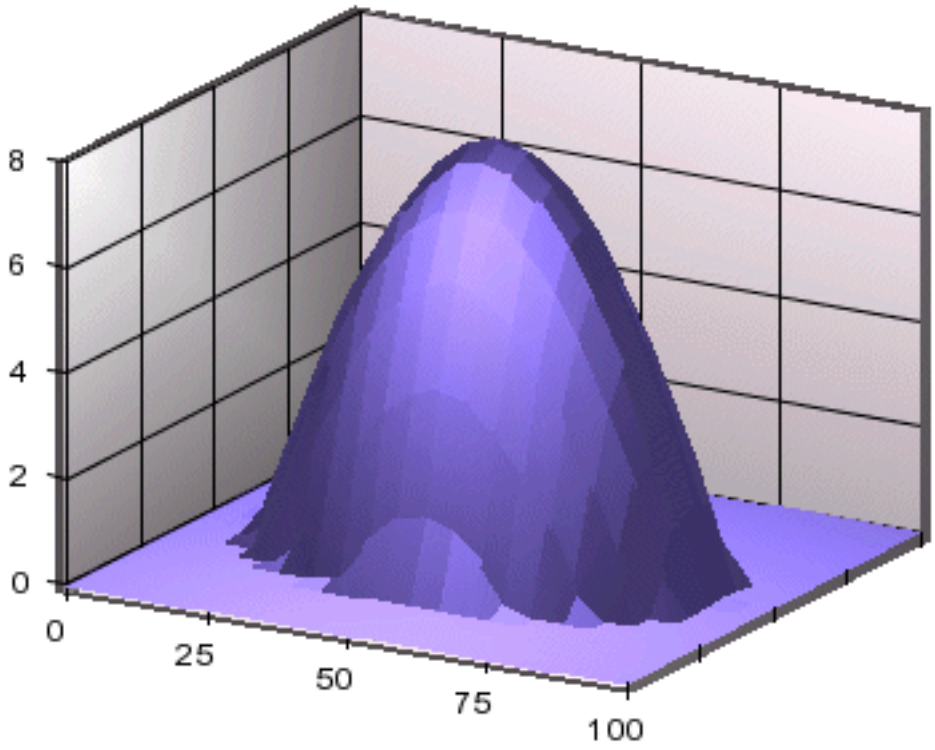


ドーナツグラフのマッピングオプション

ドーナツグラフのデータマッピングは、円グラフの場合とまったく同じです（データマッピングを参照）。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.4.15 サーフェスチャート



サーフェスチャート

3D サーフェスチャートは科学/ビジネスチャートであり、3 つの数値データセットを使用して 3D 空間で滑らかなサーフェスを形成します。各データ点について、3 つの値のうち 2 つは水平面上の座標を表し、3 つ目の値はデータ点の垂直位置を決定するために使用されます。入力データは、(以下に示すような) 矩形行列の形式でもよいし、各列が軸を表す 3 つの列に配置されてもよいです。次のサンプルデータでは、上の行（列ヘッダー）と左の列（行ヘッダー）は、サーフェスを描画する平面を形成する軸の座標値を表します。チャートを平面の上から見ると、隣接する 4 つの点を 1 つずつ結合して形成されたグリッドが表示されます。このグリッドを描画できない場合、指定されたデータセットでサーフェスチャートをプロットすることはできません。

サーフェスチャートデータのサンプル：

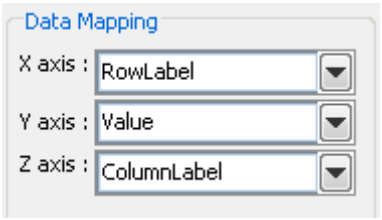
	0	20	40	60	70	80	100
20	0	0	0	0	0	0	0
30	0	10	10	10	10	10	10
40	0	10	25	25	25	2	10
80	0	10	25	30	30	30	25
90	0	10	25	30	30	30	25
100	0	10	10	25	25	25	10

サーフェスチャートが水平距離を適切に拡大縮小できるため、列ヘッダーと行ヘッダーの値は等しく分割する必要はありません。このデータセットは歪んだ逆円錐を作成します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

サーフェスチャートは、X、Y、Z 座標のセットを使用して 3D チャートをプロットします。縦軸を Y 軸、各 Y 値を X、Z、f（X、Z）の関数で表します。XY 平面（水平面）のデータは、単に正方行列のように見えます。サーフェスチャートはデータシリーズをサポートしておらず、2D チャートに変換することはできません。

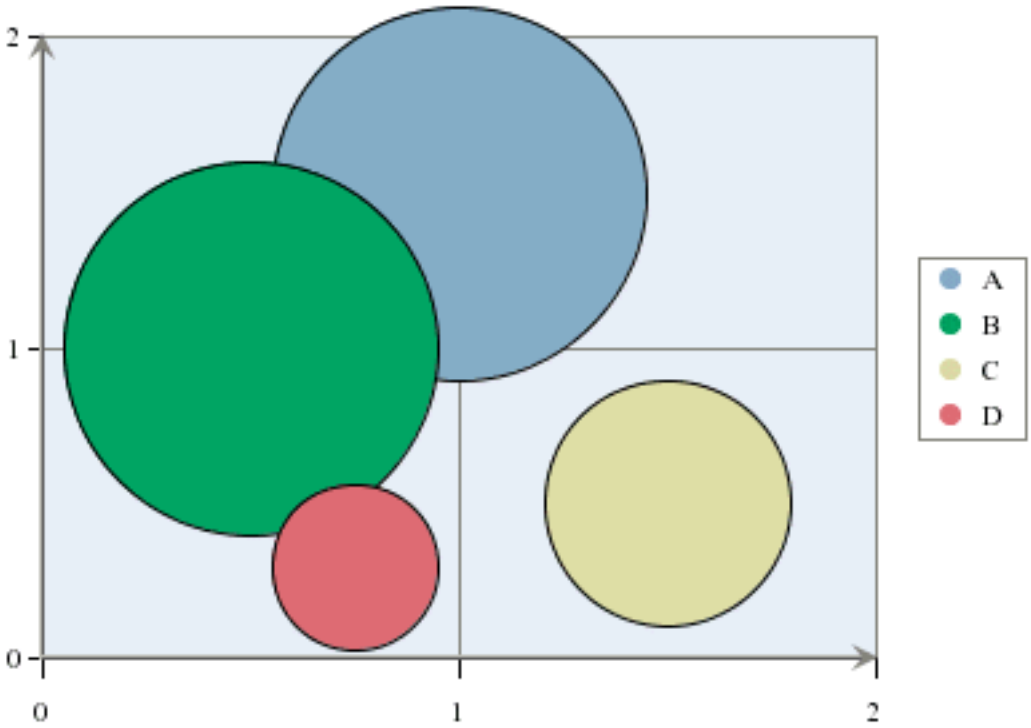
5.4.15.1 データマッピング



サーフェスチャートのマッピングオプション

サーフェスチャートのデータマッピングは、3D の散布図に似ています。X 軸、Y 軸、および Z 軸の値によって、ポイントの X、Y、および Z 座標がそれぞれ決定されます。ただし、散布図とは異なり、サーフェスチャートはデータシリーズをサポートしておらず、2D グラフに変換することはできません。

5.4.16 バブルチャート



バブルチャート

バブルチャートは、バブルの大きさが情報を提供するデータを表すために使用されます。データポイン

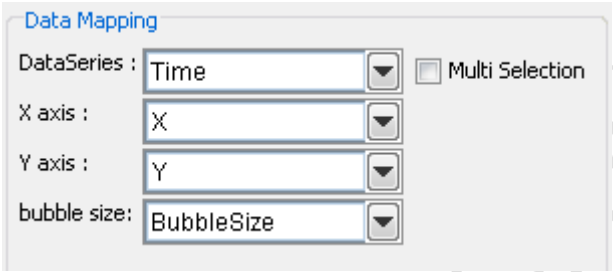
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

トは、XY 座標で表され、円またはバブルの半径を第 3 の値とします。

このグラフは、2D 形式でのみ利用できます。バブルチャートとその表示オプションの詳細については、バブルチャートを参照してください。

5.4.16.1 データマッピング

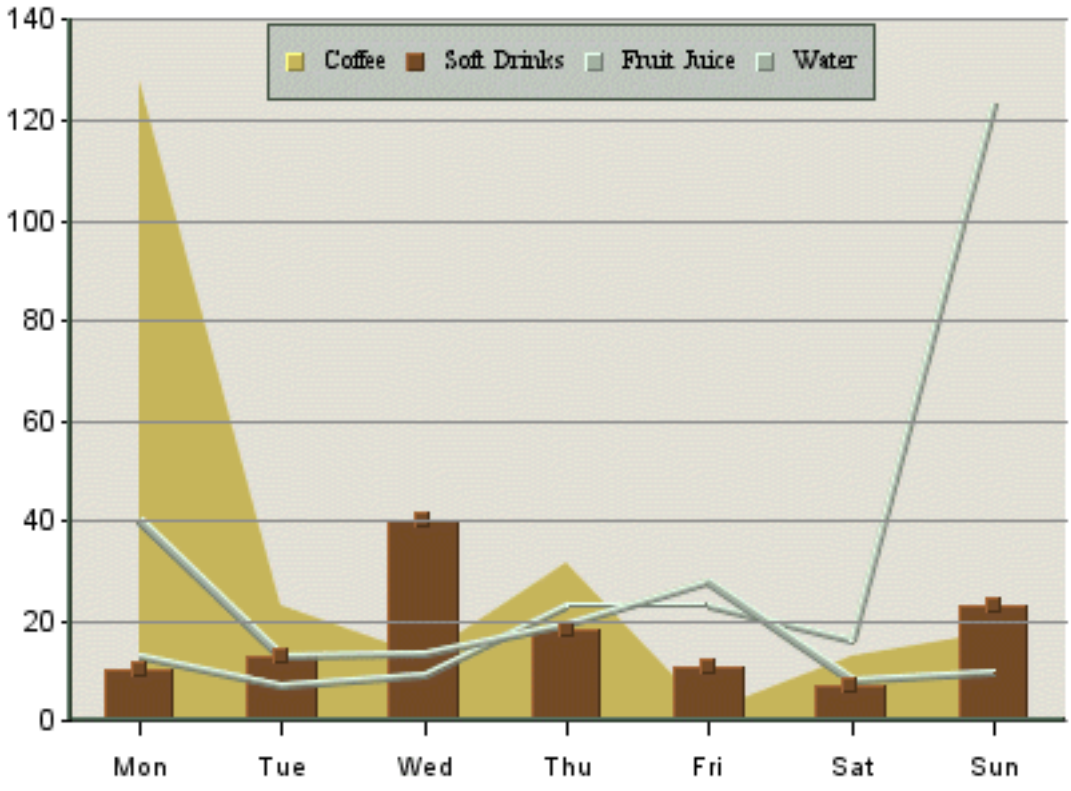
バブルチャートのデータマッピングは、3D 散布図に似ています。ただし、Z 軸の位置をプロットする代わりに、3 番目の値がバブルのサイズ（具体的には半径）を決定します。



バブルチャートのマッピングオプション

散布図と同様に、列は正常にマップされるためには数値でなければならないことに注意してください。

5.4.17 オーバーレイチャート

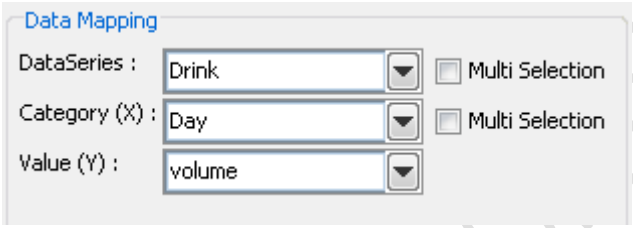


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

オーバーレイチャート

EspressChart は、オーバーレイチャートと呼ばれる特殊なチャートタイプをサポートしています。これにより、共通のカテゴリ軸を持つ 2 つ以上のチャートを重ね合わせることができます。そのため異なるチャートを使用して、データシリーズの各要素を表すことができます。これにより、チャート作成中の自由度が増し、より多くの情報を表現することができます。オーバーレイチャートでサポートされているチャートタイプは、縦棒グラフ、エリアチャート、折れ線グラフです。オーバーレイチャートには、2D チャートのみを含めることができます。

5.4.17.1 データマッピング



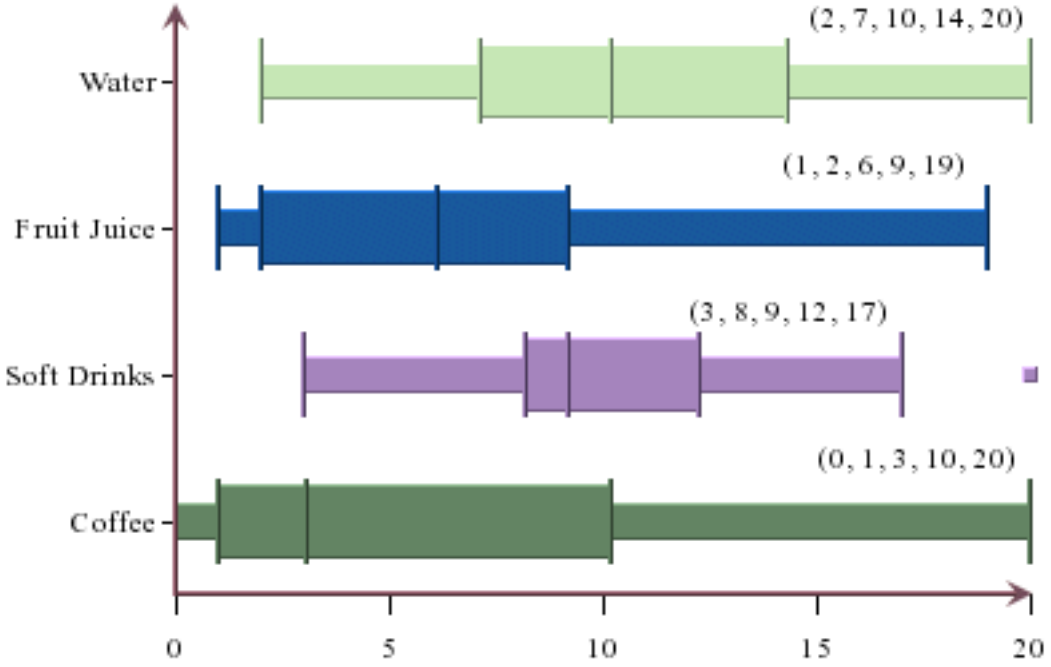
オーバーレイチャートのマッピングオプション

オーバーレイチャートのデータマッピングは、データチャートの要素を別々のチャートとしてプロットすることを除けば、縦棒グラフ（データマッピングでの説明）と同様です。第 2 値、第 2 シリーズ、コンボオプションはオーバーレイチャートには適用されませんのでご注意ください。オーバーレイチャートは第 2 値をサポートしていません。しかし、異なるシリーズ要素を異なる軸にプロットすることができます。オーバーレイチャートのオプションをプロットする方法については、オーバーレイチャートを参照してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.4.18 ボックスチャート



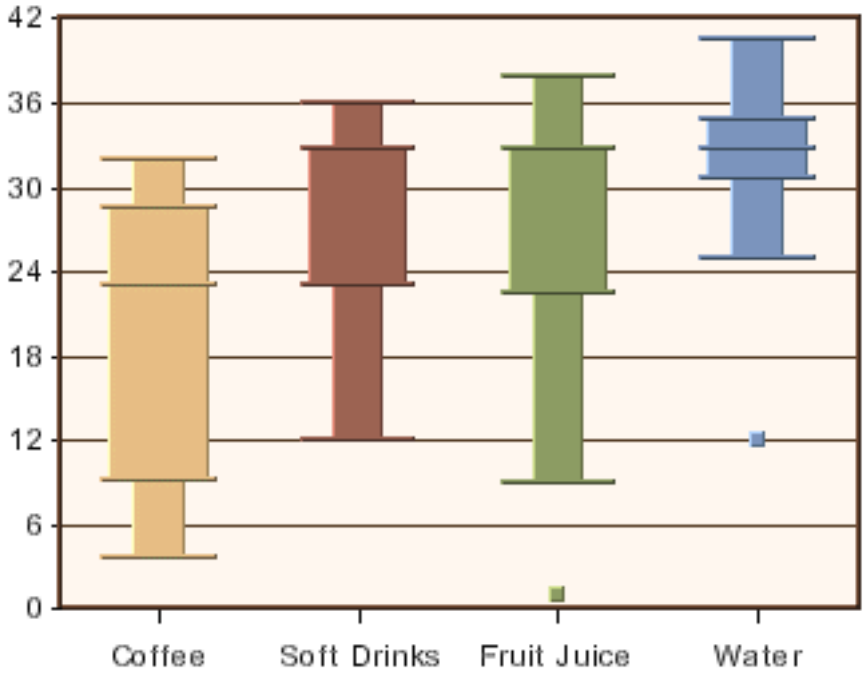
ボックスチャート

EspressChart には、統計チャートタイプであるボックスチャート（Box and Whiskers Chart）があります。基本的に、ボックスチャートは、要約統計量を素早く視覚化します。ヒストグラムは観測値の分布を表示し、ピーク値、最小値、最大値、およびデータのクラスタリングを識別できます。ボックスチャートでは、4 分割したデータ（最小データ・ポイント、25 パーセンタイル、中央値、75 パーセンタイル、最大データポイント）でのデータ分布の要約が表示されます。

ボックスの中央の線は中央値を表します。他の線は、25 パーセンタイルの増分（減分）を表します。最小および最大点は、異常値ではない最小値および最大値です。ボックスの端から最小値/最大値までの線は、ひげ(whiskers)と呼ばれることがあります。そのためボックスチャートはひげチャートと呼ばれることもあります。異常値は、ボックスの長さの 1.5 倍（またはそれ以上）の値で、75%と 25%の差です。異常値は計算され、ボックスから離れた点として表示されます。ボックスプロットについてのもう 1 つの素晴らしい点は、異なるデータセットのサマリを比較するために、1 つのチャートにボックスを積み重ねることができることです。

EspressChart では、デフォルトの水平設定に加えてボックスチャートを垂直に表示することができます。チャートデザイナでこの機能を使用するには、ボックスチャートを作成し、**Format** > **Chart Options** を選択し、**Vertical** を選択します。

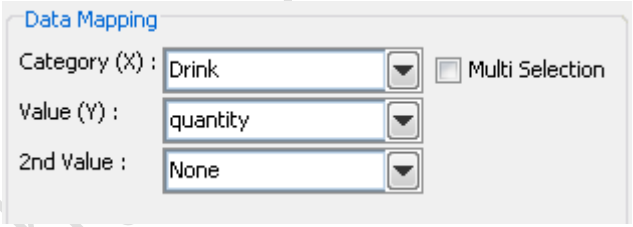
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



縦型ボックスチャート

このグラフは、2D でのみ利用できます。

5.4.18.1 データマッピング



ボックスチャートのマッピングオプション

ボックスチャートの場合、マッピングは次のとおりです。

Category (X):
異なる値がさまざまなカテゴリを決定するデータ列を選択できます。

Value (Y):
各カテゴリの値を提供するデータ列を選択できます。これらの値は、最小データポイント、25 パーセンタイル、中央値、75 パーセンタイル、および最大データポイントを計算するために使用されます。

2nd value:
オーバーレイチャートを作成するための 2 番目の値を追加します。

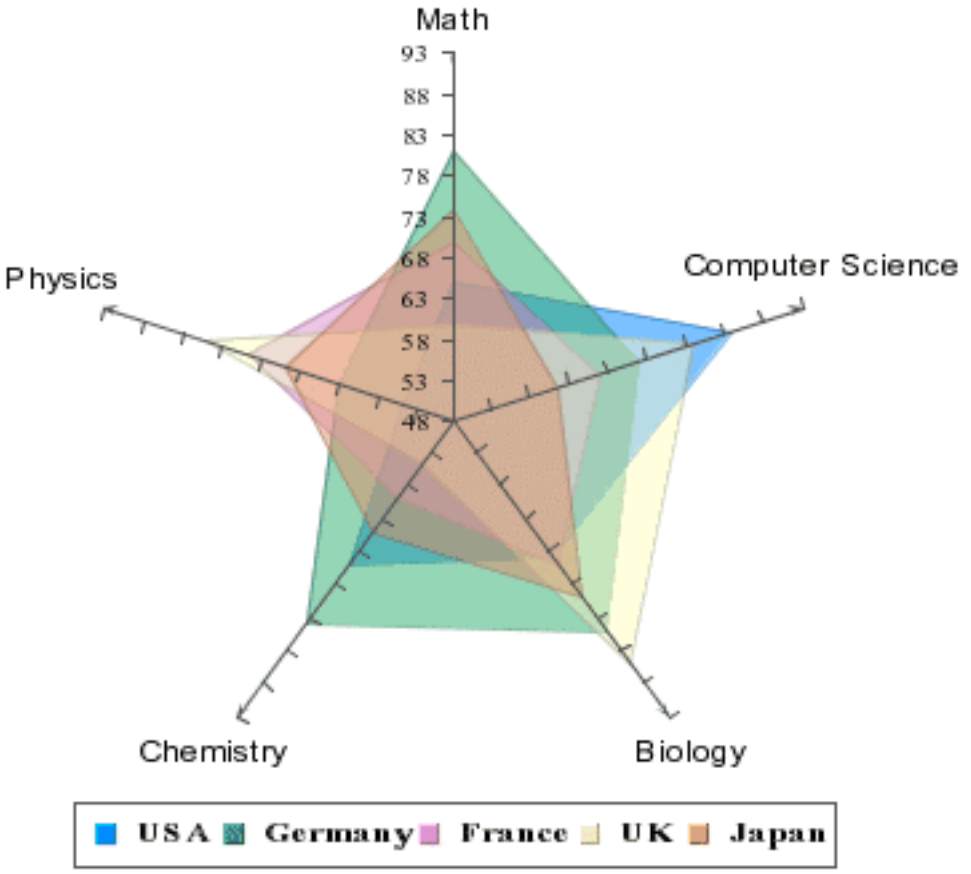
第 2 シリーズとコンボオプションはボックスチャートでは使用できません。これは、ボックスチャート

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

で使用できる唯一の組み合わせが折れ線グラフ（Line）であるためです。

データマッピングでは、データを変換することもできます（言い換えれば、単一のカテゴリの複数の列を選択すること）。データ変換の詳細については、データ変換を参照してください。

5.4.19 レーダーチャート



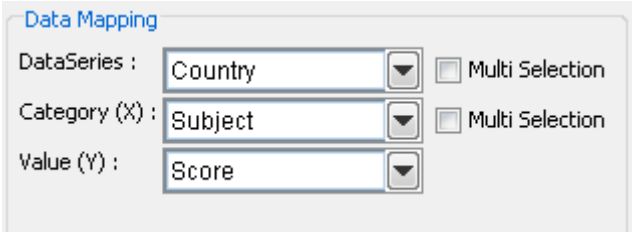
レーダーチャート

レーダーチャートは、さまざまなソースからのパフォーマンス/測定結果、統計などを比較する必要がある場合に便利です。例えば、数学、コンピュータサイエンス、生物学などの科目について、子供のテストのスコアを中央値と比較することができます。レーダーチャートには、データをプロットできる複数の軸があります。各軸はカテゴリです。データは軸上の点として表示されます。1 つのデータシリーズに属するポイントは、結合することも、囲まれたエリアを満たすこともできます。いずれの軸上の中心に近い点は低い値を示し、一方、終点に近い点は高い値を示す。場合によっては、低い値が高い値よりも望ましい場合があります。例えば、価格/収益、価格/売上、価格/株価情報などです。

このチャートは 2D でのみ利用可能です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.4.19.1 データマッピング



レーダーチャートのマッピングオプション

レーダーチャートの場合、マッピングは次のとおりです。

Data Series:
チャート内のデータシリーズ数を区別する値を持つデータ列を選択できます。データシリーズの各要素は、各軸に沿って同じ色を使用して描画されます。

Category (X):
異なる値によってカテゴリが決定されるデータ列を選択できます。この列の固有の要素の数によって、レーダーチャート内の軸の数が決まります。

Value (Y):
カテゴリに対してプロットする数値を提供するデータ列を選択できます。

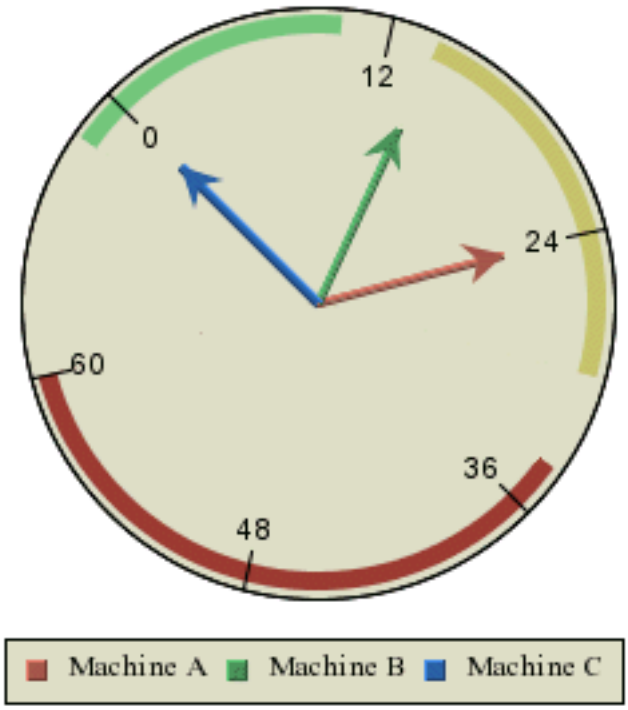
第 2 値、第 2 シリーズ、コンボのオプションはレーダーチャートには適用されません。レーダーチャートは二次値をサポートしていません。

データマッピングでは、データを変換することもできます（言い換えれば、単一のカテゴリの複数の列を選択すること）。データ変換の詳細については、データ変換を参照してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.4.20 ダイヤルチャート

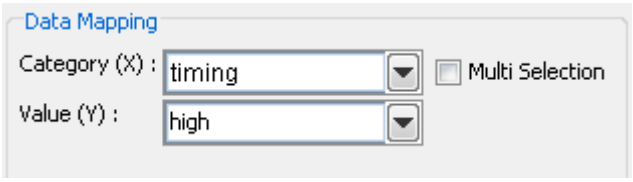


ダイヤルチャート

ダイヤルチャートは、円形のチャート（温度ゲージ、速度計、時計など）の作成、ダッシュボードの作成、バランススコアカードアプリケーションの作成に使用されます。ここでは、データは「ダイヤル」の「手」として表されます。ダイヤル全体（時計など）またはその一部（ガソリンゲージなどの半円形の図）を使用できます。ダイヤルチャートでは、ポップアップラベル、ドリルダウン、ダイヤルの手やセクタを回転させるなどのユーザ操作をサポートしています。

このチャートは、2D でのみ使用できます。

5.4.20.1 データマッピング



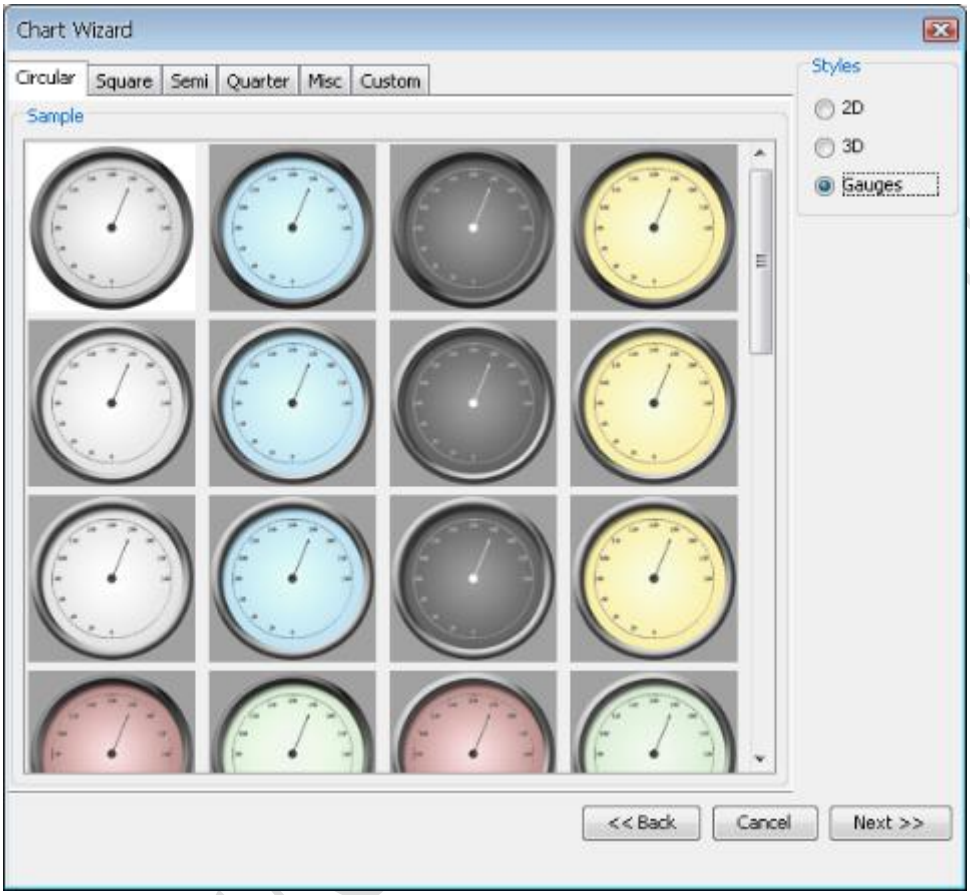
ダイヤルチャートのマッピングオプション

ダイヤルチャートのデータマッピングは、円グラフ（データマッピングで詳述）に似ています。しかし、円グラフの扇形の代わりに、カテゴリはダイヤルハンドになります。また、ダイヤルチャートでは、データシリーズや第 2 値がサポートされていません。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.4.20.2 ゲージ

ゲージは、プロット背景画像またはプロットフォアグラウンド画像を含む特別なタイプのダイヤルチャートです。新しいゲージを作成するには、**Chart Type Selection** ダイアログで **Gauges** ラジオボタンを選択します。

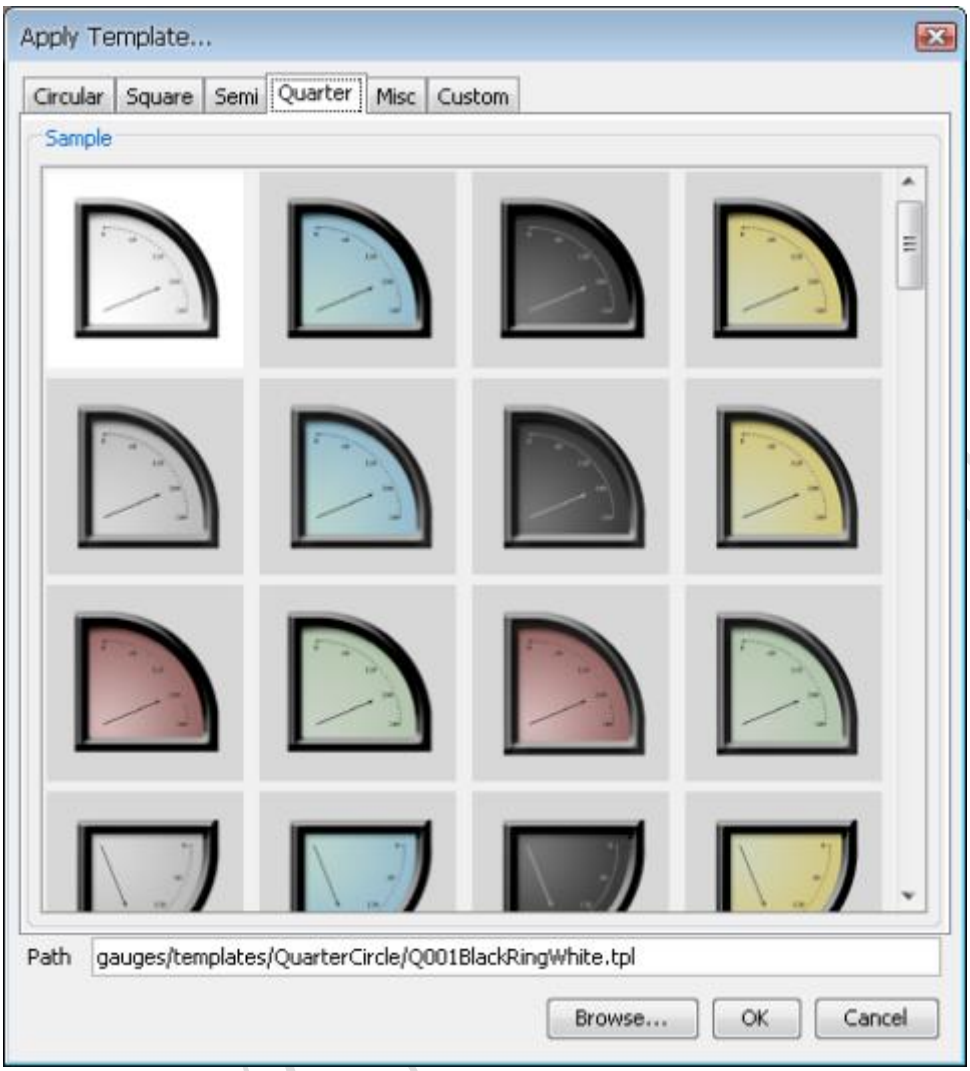


ゲージテンプレートの選択

また、テンプレート（tpl ファイル）を**<EspressChartInstallDir>/gauges/templates/Custom/**フォルダに保存することで独自のゲージを作成することもできます（Custom フォルダは EspressChart インストーラによって自動的に作成されませんので、任意のファイルマネージャを使用して手動で作成してください）。このテンプレートをカスタムタブに追加するには、**<EspressChartInstallDir>/gauges/screenshots/selected/Custom/**と、**<EspressChartInstallDir>/gauges/screenshots/unselected/Custom/**に計 2 枚、解像度の高いものと低いものを配置する必要があります。画像を作成する簡単な方法は、通常の画像を gif に書き出し、100 ピクセル×100 ピクセルにサイズ変更することです。その後、背景を暗い色に変更し、参照のために再度エクスポートしてサイズを変更します。

既存のダイヤルチャートにゲージテンプレートを適用することもできます。現在のチャートがダイヤルチャートのときに適用テンプレートを選択すると、新しいチャートを作成するときと同様にゲージタブが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

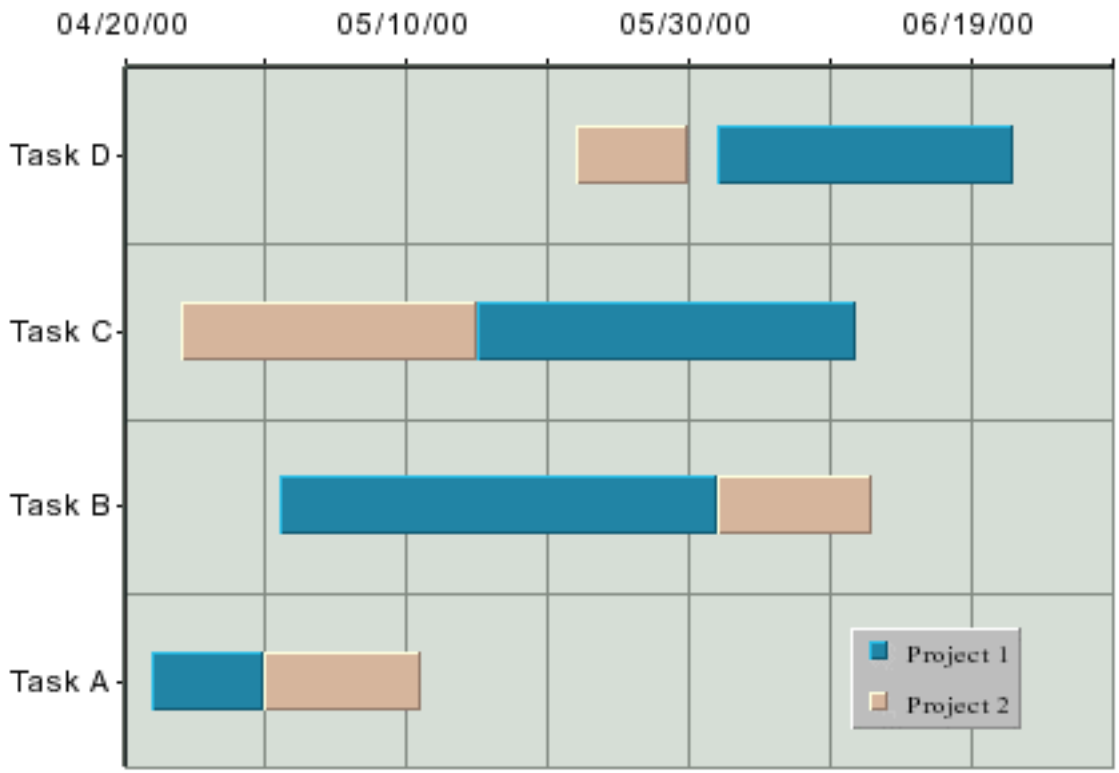


ゲージテンプレートの適用

ダイヤルのバックグラウンドおよびフォアグラウンドイメージの詳細については、ダイヤルチャートを参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.4.21 ガントチャート



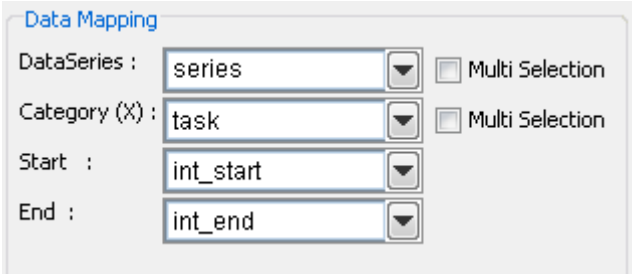
ガントチャート

ガントチャートまたはタイムチャートは、時間係数が測定されたデータを表すために使用されます。これは、プロジェクトまたは時間管理の状況でよく使用されます。ガントチャートは、Y 軸にカテゴリ軸、X 軸に値軸を持つ棒グラフに似ています。値の軸は、日付、時刻、またはタイムスタンプのデータを使用します（数値も使用できます）。各データポイントには、開始時刻と終了時刻が関連付けられています。

このチャートは、2D フォームでのみ使用できます。

5.4.21.1 データマッピング

ガントチャートのデータマッピングは、HighLow チャートに似ています。ただし、高値と低値の列を選択する代わりに、開始値と終了値を選択します。



ガントチャートのマッピングオプション

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

マッピングは次のとおりです。

Data Series:
チャート内のデータシリーズ数を区別する値を持つデータ列を選択できます。

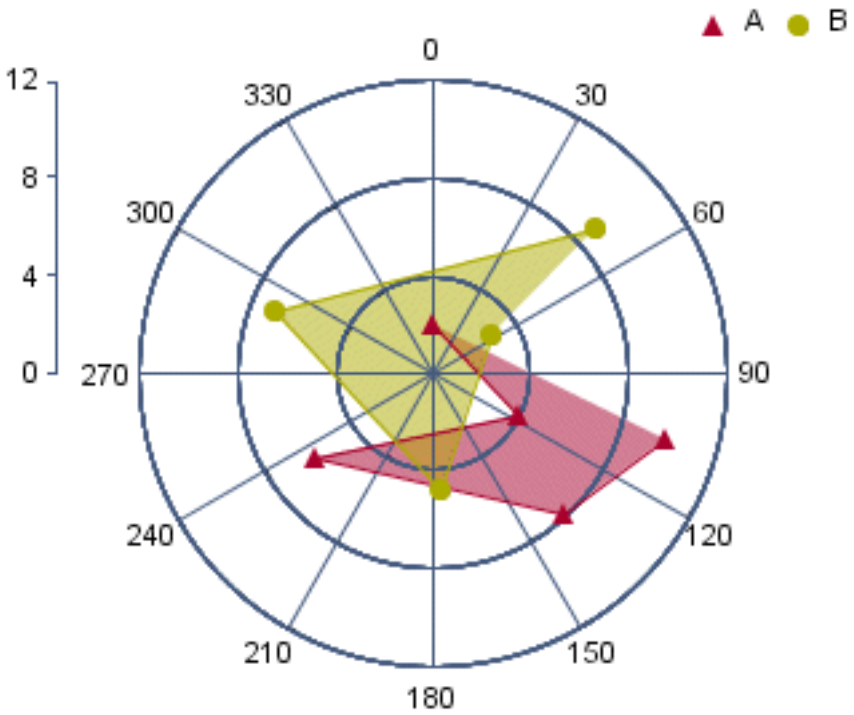
Category (X):
異なる値でカテゴリを決定するデータ列を選択できます。

Start:
識別可能な値がカテゴリの開始時間間隔を表すデータ列を選択できます。

End:
カテゴリの終了時間間隔を表す個別の値を持つデータ列を選択できます。

データマッピングでは、データを変換することもできます（言い換えれば、単一のカテゴリの複数の列を選択すること）。データ変換の詳細については、データ変換を参照してください。

5.4.22 極座標



極座標

2D 散布図と同様に、極座標グラフは平面上の点をプロットします。2D 散布図と同様に、極座標グラフは平面上の点をプロットします。ただし、直交座標の代わりに、極座標グラフは極座標（r、θ）を使用して点をプロットします。ここで、r は円プロットの中心からの距離を表し、θ はプロットの中心からその点を通過する光線の角度を表します。散布図と同様に、極座標グラフの座標データは数値でなけれ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ばなりません。θ値は度またはラジアンで指定できます。第3のデータ列を使用して、データポイントをグループに分けることができます。

5.4.22.1 データマッピング

極座標グラフのマッピングオプション

極座標グラフのデータマッピングは散布図に似ています。angleオプションとradiusオプションを使用し、値が極座標を構成する列を選択できます。これらは両方とも数値でなければなりません。データシリーズボックスでは、チャート内のデータシリーズ数を区別する値を持つデータ列を選択できます。データシリーズの各要素は、同じ色、マーカーなどの描画属性の同じセットを使用して描画されます。

5.4.23 データマッピング/データソースの変更

マッピングオプションの設定が完了したら、メインのChart Designerインタフェースに移動します。Chart Designerに入ったら、**Data** > **Modify Data Mapping**を選択するか、**Change data mapping**アイコン をクリックして、データマッピングウィンドウに戻ることができます。これにより、データマッピング画面に戻り、グラフのマッピングを調整できます。

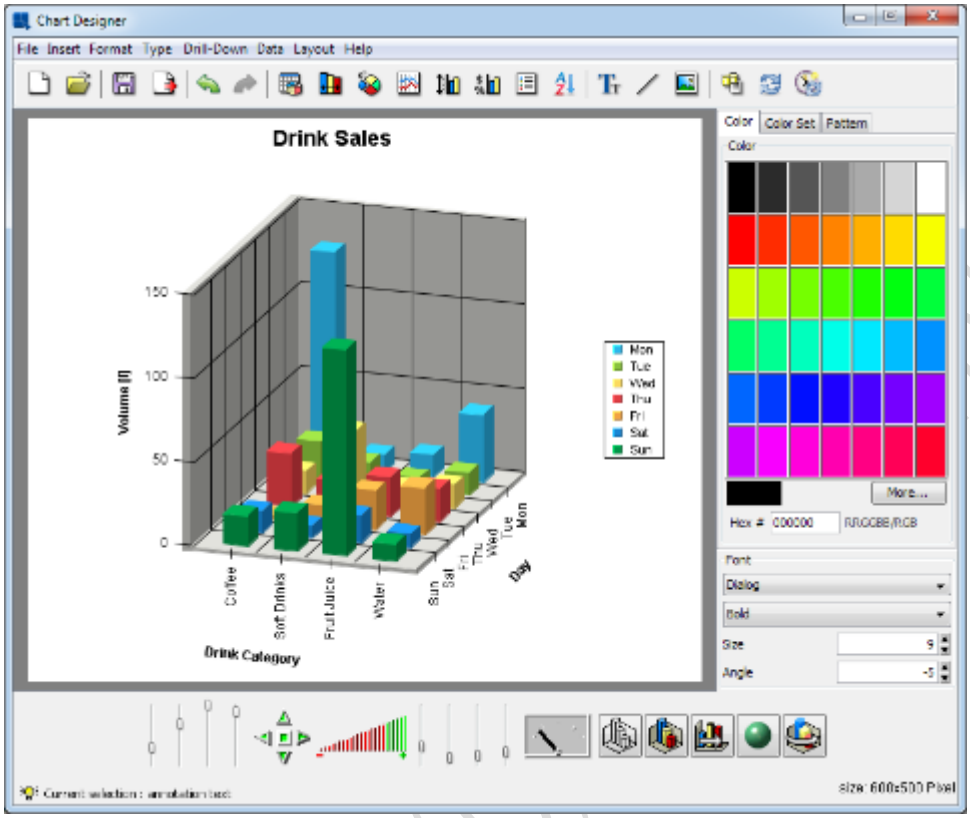
Data > **Modify Data Source**を選択するか、ツールバーの**Modify data source**アイコン をクリックして、Chart Designerから直接データソースを変更することもできます。これにより、Data Source Managerが開き、新しいデータソースを選択することができます。このオプションはレポートデータをデータソースとして使用する場合には使用できません。また、データソースを変更する場合は、再度データマッピングを実行する必要があります。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5 デザイナーインタフェース

ウィザードのすべての手順を完了すると、メインの Chart Designer インタフェースが表示されます。ここでは、プロパティを変更して新しい要素を追加することによって、チャートのデザインをカスタマイズすることができます。



5.5.1 デザイナーメニュー

Chart Designer のほとんどの機能は、デザイナウィンドウの上部にあるメニューバーから制御できます。このセクションでは、使用可能なオプションの概要を説明します。すべての機能については、この章の後半で説明します。

5.5.1.1 ファイルメニュー

このメニューは、ファイルのオープン、クローズ、および保存などの基本的なファイル操作を実行します。ファイルオプションの詳細については、グラフの保存およびエクスポートを参照してください。

New:

Data Source Manager に遷移し、新しいチャートの作成が開始されます。現在のチャートを保存していない場合は、保存するよう指示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Open:
保存されたチャート定義を開くことができます。Chart Designer によってロードできるファイルには、**.cht**、**.tpl**、および**.xml** のチャート定義ファイルがあります。

Close:
現在のチャートを閉じます。

Apply Template:
このオプションを使用すると、現在のチャートにテンプレートを適用できます。任意のチャートテンプレートを**.tpl** または**.xml**形式で適用できます。チャートテンプレートを使用した作業の詳細については、テンプレートの使用を参照してください。

Save:
現在のチャートを保存します。

Save As:
現在のチャートを別名保存することができます。チャートを**.cht** または**.tpl**（チャートまたはテンプレートファイル）として保存することができます。その他のオプションを使用すると、Chart Viewer アプレットを埋め込んだ XML チャート定義ファイルまたは HTML ページを作成できます。

Export:
チャートをいくつかの画像フォーマットにエクスポートすることができます。チャートデータは XML ファイルとしてエクスポートすることもできます。

Exit:
Chart Designer を閉じます。可能であれば現在のファイルは保存されます。

5.5.1.2 挿入メニュー

このメニューでは、さまざまな要素をチャートに追加できます。

Titles:
このオプションを使用すると、チャートに自動的にタイトルを追加できます。各軸のタイトルだけでなく、メインタイトルも指定できます。注釈テキストとは異なり、タイトルはチャートとともに自動的にサイズと位置が決まります。

Text:
これにより、テキストまたは注釈をグラフに追加することができます。テキストはチャートの任意の場所に追加することができ、さまざまな書式プロパティを持つことができます。実行時置換のためのテキストを変数として追加することができます。グラフにテキストを追加する方法の詳細は、テキストを追

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

加を参照してください。

Background:
これにより、チャートの背景として使用するイメージを選択できます。背景画像は、キャンバス全体に合わせてタイル、センタリング、または引き伸ばすことができます。

Dial Foreground:
このオプションは、ダイヤルチャートでのみ使用できます。これにより、ダイヤルチャートの前景画像として使用するイメージを選択できます。画像は引き伸ばすことができます。

Dial Background:
このオプションは、ダイヤルチャートでのみ使用することができます。ダイヤルチャートの背景画像に使用するイメージを選択できます。画像は引き伸ばすことができます。

Link:
チャート内のデータポイントまたはデータポイントのコレクションへのハイパーリンクを追加できます。グラフでハイパーリンクを使用するには、マップファイルをイメージと共にエクスポートする必要があります。

Line:
これにより、チャートに任意のフローティングラインを追加することができます。これらの線はどこにでも配置でき、枠線の描画にも使用できます。フローティングラインはポインタとして使用されることもでき、矢印として生成することもできます。

TrendLine:
チャートにトレンドラインを追加することができます。EspressChart を使用すると、線形、任意の次数の多項式、指数関数、対数関数、移動平均、指数移動平均、三角移動平均、三次 B スプライン曲線、正規分布曲線など、さまざまな種類のトレンドラインを描画できます。

Horz/Vert Line:
グラフに固定された水平線または垂直線を追加することができます。定数線（X 軸または Y 軸の固定値を描画する線）またはコントロールライン（平均、最小、最大、または標準の倍数のいずれかの値の範囲に基づいて線を描画する）かのいずれかを選択できます。

Control Area:
固定されたエリア（2D チャートのプロットエリアまたはダイヤルチャートの面）を描画して、チャートのデータ値と比較することができます。

5.5.1.3 フォーマットメニュー
このメニューでは、さまざまなチャートコンポーネントのプロパティを編集および変更できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Undo:
Designer で実行された最後の操作をキャンセルし、元の状態に戻します。Designer は 10 のアクションまで戻すことができます。

Redo:
Undo コマンドの動作を元に戻します。たとえば、フォントの色を黒から赤に変更した場合に、Undo コマンドをすることで黒に戻すことができ、Redo コマンドをすることで、再度赤に変えることができます。

Data Properties:
このオプションを使用すると、データの表示方法に関するいくつかのオプションを制御できます。横棒、縦棒の太さ、データの表示方法、データのトップラベルを描画するかどうか、ネガティブトップラベルのカラーを有効にしたり選択したりすることができます。

Histogram Options:
チャートをヒストグラムとして描画するかどうかを指定し、頻度カウントを指定するための追加オプションを表示することができます。

Aggregation Options:
グラフを描画する前にデータを集計するかどうかを指定したり、集計のタイプを指定する追加オプションを表示したりすることができます。

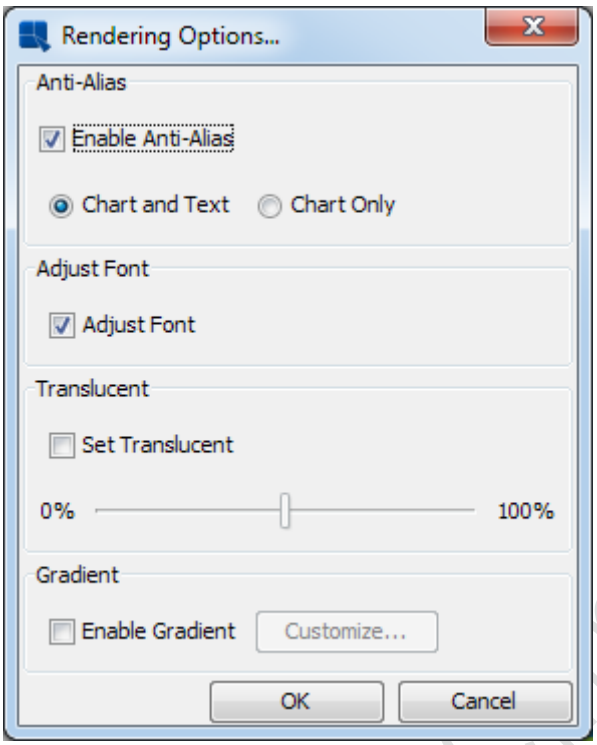
Zoom Options:
時間ベースのズームのオプションを有効/無効にしたり、オプションを設定したりすることができます。このオプションは、日付/時刻データがカテゴリ軸にマップされている場合にのみ適用されます。

3D Display Options:
3D チャートの表示を制御するいくつかのオプションを設定できます。3D 棒（または同様の）チャートのインラインシリーズを指定し、3D スキャッターチャートおよびサーフェスチャートのレンダリング近似を指定することができます（これにより、多くのデータポイントを持つチャートのパフォーマンスが向上します）。

Rendering Options:
より良いプレゼンテーションのためにチャートにさまざまなレンダリングオプションを指定できます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Rendering Options ダイアログ

さまざまなレンダリングオプションを指定できます。

Enable Anti-Alias:

グラフのアンチエイリアスを指定できます。アンチエイリアシングは、チャート内のテキストと線を滑らかにし、スムーズな外観を作ります。この機能は、ファイル全体に適用することも、チャートにのみ適用することもできます。この場合、テキストは影響を受けません。

Adjust Font:

グラフのテキストのフォントサイズを画面の解像度に比例して指定することができます。この機能により、チャートをさまざまなエクスポート形式および環境でより正確に変換できます。

Translucent:

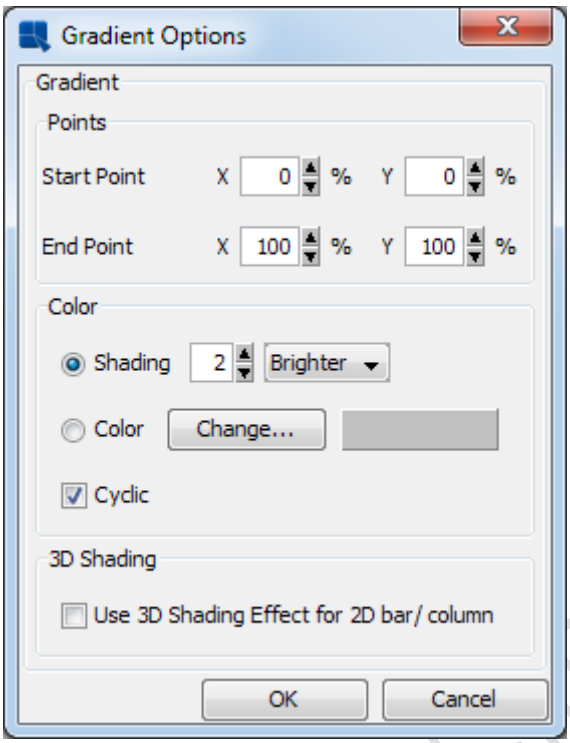
列/棒/エリアを半透明にするかどうかを指定することができます。これにより、背後にある見えない列/棒/エリアを表示することができます。また、スライダーを調整することで不透明度を指定することもできます。ここでは、0%は完全に不透明で 100%は完全に透明です。このオプションは、すべての 3D チャート、およびデータシリーズの 2D エリア、レーダー、ガントチャートで使用できます。

Gradient:

グラデーションを有効にすることができます。グラデーションは、キャンバス全体に適用することも、チャートのデータポイントのみに適用することもできます。グラデーションオプションを変更するには、Customize をクリックして次のダイアログを表示します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Gradient Options ダイアログ

このダイアログでは、グラデーションの開始点と終了点（x-y 平面に基づきます)、グラデーションのカラースキーム、グラデーションの周期性を指定することができます。開始点と終了点は、キャンバスの左上隅を（0,0)、キャンバスの右下隅を（100,100）としたパーセンテージで表されます。グラデーションの変化が濃くなる（濃くなるか明るくなる）か、別の色にするかを指定することもできます。最後に、勾配を繰り返すように指定することができます。つまり、勾配内の 2 つの反対側の間を切り替えることができます。勾配が周期的である場合、開始点と終了点は最初のセグメントを表すことに注意してください。

2D 横棒/縦棒に 3D シェーディング効果を使用するオプションを選択することで、特定の 2D チャート（縦棒、横棒、スタック横棒、スタック横棒、100%縦棒、オーバーレイチャート）に対して 3D シェーディングを有効にすることもできます（このオプションは 2D 横棒グラフまたは縦棒グラフのみ)。

Font Mapping:

PDF エクスポート用のシステムフォントファイルをマップすることができます。この機能の詳細については、PDF フォントマッピングを参照してください。

Chart Options:

使用しているチャートタイプの特定のオプションが表示されます。オプションはチャートの種類によって異なります。

Axis Scale:

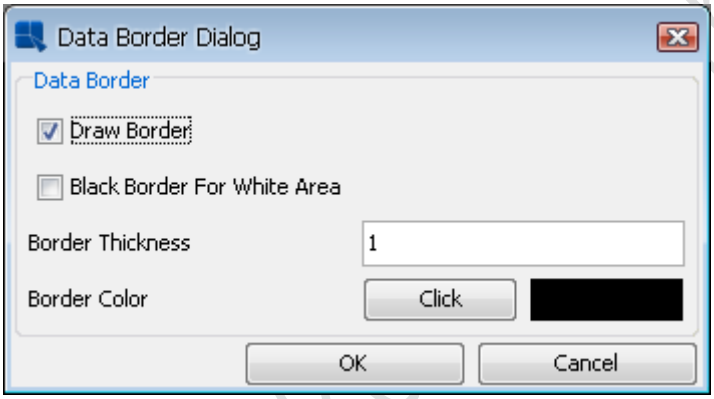
任意の数値軸のスケールを調整できます。自動的に（最適な）スケーリングがデフォルトで使用されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Axis Elements:
軸と軸ラベルの外観を変更することができます。ここでのオプションには、軸の太さ、グリッド線、ラベルステップ、およびデータフォーマットが含まれます。

Canvas:
背景キャンバスのサイズを調整することができます。キャンバスのサイズがビューポートを超えたときにスクロールバーを使用するかどうかを指定することもできます。

Data Border:
データ要素（縦棒、横棒など）に枠線を追加します。



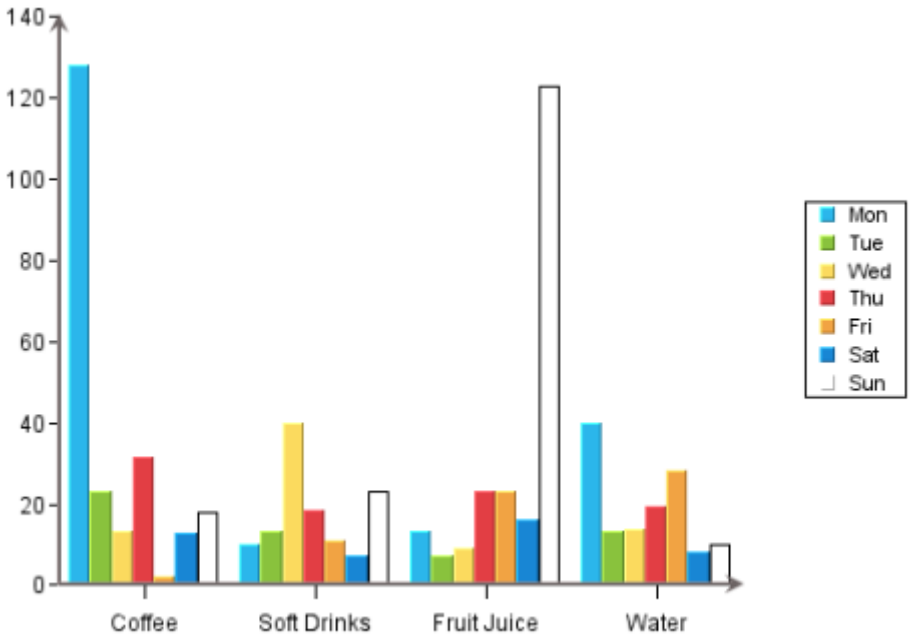
Data Border ダイアログ

枠線の描画オプションを選択すると、枠線のプロパティを構成できるように、枠線の太さと枠線の色フィールドが有効になります。

Black border for white area オプションは、**Draw border** が無効の場合でも選択できます。これを行うと、単純な黒い枠線が白のデータ要素にのみ追加されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



***Black border for white area** が有効になっている縦棒グラフ*

このオプションは、サーフェスチャート、散布図、折れ線グラフ、バブルチャート、ボックスチャート、レーダーチャート、ダイヤルチャート、極座標グラフでは利用できません。

Legend:
チャートの凡例表示プロパティを変更します。

Lighting Model:
3D チャートの照明オプションのいくつかを変更することができます。ライトアンビエントカラーと強度の両方を変更できます。

Line and Point:
チャートの任意の線の表示プロパティを変更することができます。2D チャートでは、任意のデータセットの線や点を表示したり、外観をカスタマイズしたりすることができます。このメニュー項目を使用して、トレンド、フローティング、または水平/垂直線の表示プロパティを変更することもできます。

Plot Area:
このオプションを使用すると、2D チャートの X 軸と Y 軸で囲まれたエリアの外観をカスタマイズできます。

No Data To Plot Message:
このオプションを使用すると、グラフを描画するためのデータが不十分な場合に表示されるメッセージを設定できます。デフォルトでは、"**No Data To Plot**"メッセージが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Flash Hintbox Customization:
このオプションを使用すると、フラッシュのエクスポートでヒントボックスのフォントプロパティと枠線と背景色を指定できます。

NULL Data Properties:
このオプションを使用すると、Null データを含むカテゴリ軸のポイントを表示し、それを別の文字列に置き換えることができます。デフォルトでは、Null データのカテゴリポイントはスキップされます。

Table:
チャートデータを表示するテーブルを追加および設定することができます。

Text Properties:
このオプションを使用すると、グラフ内の任意のテキストのリサイズ率を設定できます。このオプションから、Java 2D の回転テキストを使用するように指定することもできます。このオプションを使用すると、回転したテキストがより鮮明に表示されます。任意のテキスト置換オプションを指定することもできます。

Viewer Options:
このメニュー項目では、チャートを表示したときにチャートビュアーアプレットの設定オプションを指定できます。ビュアーのポップアップメニューでは、使用可能なオプションを制御できます。これらのオプションは HTML パラメータでも制御できます。

5.5.1.4 タイプメニュー

このメニューでは、現在のチャートとそのディメンションを変更できます。各ディメンションの表現をサポートするグラフタイプの 2D と 3D を切り替えることができます。グラフの種類を変更することもできます。グラフの種類を切り替えると、一部の書式情報が失われることに注意してください。また、ガントチャートと他のチャートタイプを切り替えることもできません。

5.5.1.5 ドリルダウンメニュー

このメニューには、チャートのドリルダウンレイヤーの追加と移動を可能にするオプションがあります。ドリルダウン機能については、ドリルダウンで説明しています。ドリルダウンは、チャートをレポートから独立して配備する場合にのみ使用可能（機能的）であることに注意してください。

Add:
ドリルダウンしたデータのレイヤーを追加します。

Remove This:
現在のレベルのデータドリルダウンが削除されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Remove All:
全てのレベルのデータドリルダウンが削除されます。

Previous:
前のレイヤーのデータドリルダウンに移動します。

Next:
次のレイヤーのデータドリルダウンに移動します。

Go To Top Level:
データドリルダウンのトップレベルチャートに移動します。

Dynamic:
ドリルダウン用の動的データを有効にすることができます。

Parameter Drill-Down:
パラメータドリルダウン ナビゲーションウィンドウが表示され、パラメータドリルダウンのレイヤーを編集、追加、および削除することができます。

5.5.1.6 データメニュー

このメニューには、チャートデータを更新、並べ替え、または完全に変更するためのオプションが含まれています。

Modify Data Mapping:
データマッピングウィンドウが表示され、グラフのデータマッピングを変更できます。

Modify Data Source:
データソースマネージャに戻り、チャートの新しいデータソースを選択できるようになります。

Modify Database:
グラフでデータベースをデータソースとして使用する場合、このオプションを使用すると、Report Designer のように、グラフが使用するデータベース接続を変更できます。この機能の詳細については、データベース接続の編集を参照してください。

Modify Query:
グラフでデータベースをグラフのデータソースとして使用する場合は、このオプションを使用すると、Report Designer のようにグラフデータを取得するために使用されるクエリを変更できます。この機能の詳細については、クエリの編集を参照してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Query Parameters:

クエリパラメータを再初期化し、現在のグラフのパラメータ値を変更することができます。このオプションは、グラフでパラメータ化されたクエリをデータソースとして使用する場合にのみ使用できます。

Ordering:

チャート内のデータポイントを並べ替えることができます。任意のカテゴリ要素の順序を任意に変更することも、値でカテゴリを並べ替えることもできます。

Refresh:

現在のグラフが最新のデータに更新されます。このオプションが機能するには、元のデータソースが使用可能でなければなりません。

Schedule Refresh:

データを更新するスケジュールを設定できます。このオプションは、Chart Viewer アプレットにチャートを配置するためのオプションです。チャートが最新のデータで更新されるように定期的な更新間隔を設定することができます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

View Table:
このオプションは、現在のチャートが生成されるデータテーブルを含むウィンドウを表示します。最初の 20 レコードのみがテーブルに表示されます。**Show All Records** チェックボックスをクリックすると、すべてのレコードが表示されます。

View Chart Data:
データテーブル全体を表示するのではなく、現在のチャートにプロットされているデータポイントを表示するよう変更します。

View Data Source Info:
グラフに独立したデータソースが含まれている場合、このオプションを使用すると、グラフの作成に使用されたデータソースに関する情報を含むダイアログが表示されます。データソースの種類と場所が表示されます。

Go Back:
リンクをたどった場合、元のグラフに戻ります。

5.5.1.7 ヘルプメニュー

このメニューでは、プログラムのバージョンを表示することや、ドキュメントを開くことができます。

About:
プログラムのバージョンに関する情報が表示されます。

Contents:
EspressReport のユーザガイドが開きます。

5.5.1.8 レイアウトメニュー

このメニューのオプションを使用すると、チャートデザイナインタフェースのさまざまな要素を切り替えることができます。このメニューから、フォントとカラーパネル、Chart Designer のツールバー、および 3D チャートのナビゲーションパネルのオン/オフを切り替えることができます。

5.5.2 Designer のツールバー

Chart Designer ウィンドウの上部にあるツールバーを使用すると、Chart Designer で最も一般的に使用される機能に簡単にアクセスできます。ボタンは次の機能を実行します。

アイコン	説明

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

	現在のチャートを保存します。
	最後の操作を取り消します。
	取り消した操作をもう一度行います。
	現在のチャートのデータマッピングを変更します。
	データ表示プロパティを変更します。
	チャート特有のオプションを変更します。
	ラインとポイントの属性を変更します。
	軸のスケールを変更します。
	軸要素/表示プロパティを変更します。
	凡例表示を変更します。
	データの表示順序を変更します。
	注釈のテキストを挿入します。
	フロートラインを挿入します。
	背景画像を追加または変更します。
	チャートデータソースを変更します。

5.5.3 カラー/カラーセット/パターン/フォントパネル

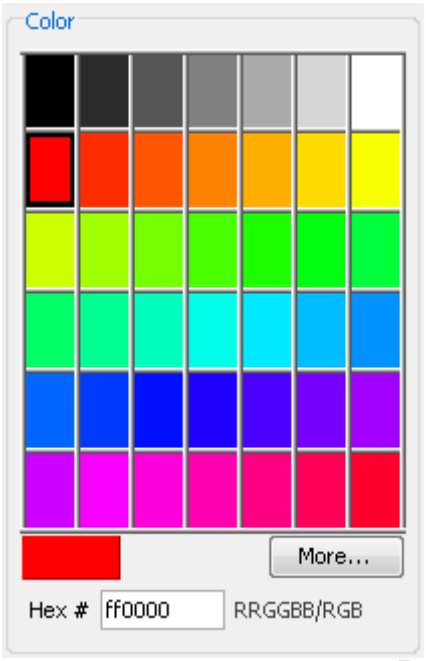
Chart Designer ウィンドウの右側にある色、カラーセット、およびフォントパネルを使用すると、チャートオブジェクトの色や、グラフの任意のテキスト/ラベルのフォントサイズやスタイルを変更することができます。**Layout** > **Show font/color** パネルオプションをトグルすることにより、これらのパネルを表示しないように選択できます。

5.5.3.1 カラーパネル

カラーパネルを使用して、チャートの任意の要素の色を変更することができます。要素の色を変更するには、まず要素の色をクリックします。チャートデザイナウィンドウの下部にあるステータスバーには、現在選択されている要素が示されます。オブジェクトを選択した後、カラーパネルのいずれかのパネルをクリックすると、選択した色が反映されてオブジェクトの色が変わります。

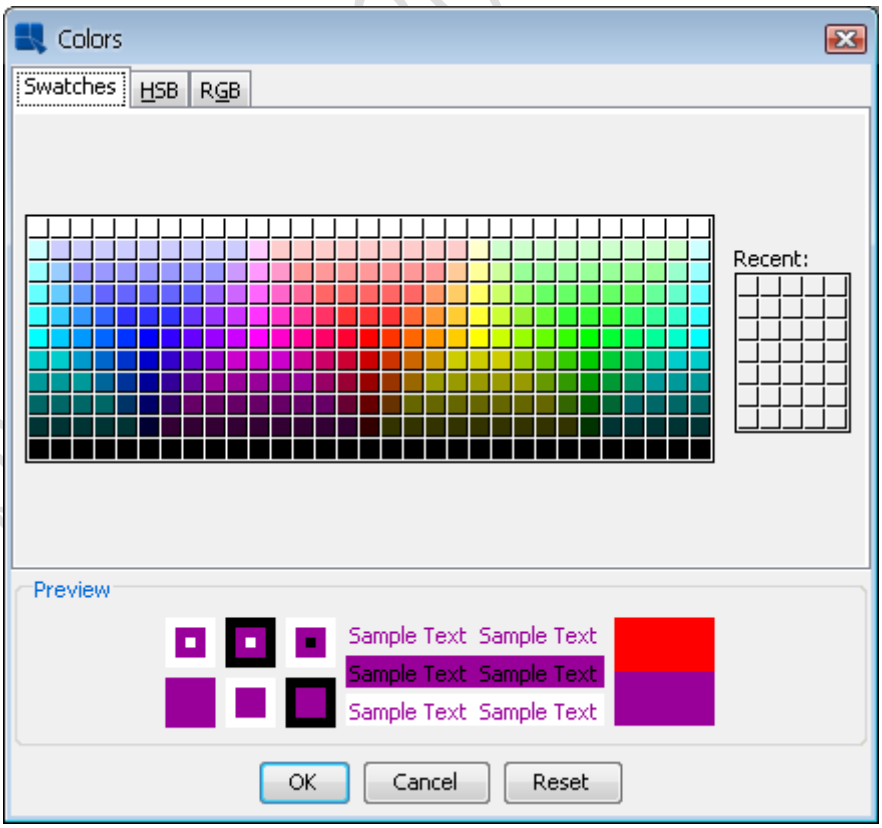
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Color Panel

オブジェクトのカスタムカラーを作成するには、まずそれを選択して、**More** ボタンをクリックします。これにより、新しいダイアログが表示され、大きなパレットから色を選択したり、新しい色を作成したりすることができます。



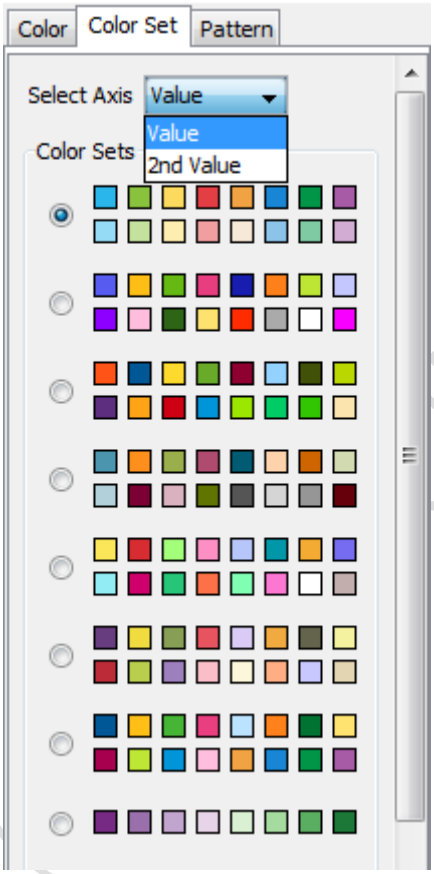
Additional Colors ダイアログ

このダイアログでは、見本から新しい色を選択することや、HSB 値または RGB 値を使用してカスタム色を設定することができます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.3.2 カラーセットパネル

カラーセットパネルでは、チャートのカラースキームを選択できます。これは、値または第 2 の値の軸上にデータポイントをチャート化するために適用することができる予め定義されたカラーセットを含みます。



Color Set パネル

値または第 2 の値の軸のカラーセットを変更するには、まず **Color Set** タブを選択します。次に、**Select Axis** ボックスから軸（Value または 2nd Value）を選択し、適切なカラーセットラジオボタンをクリックします。チャートのデータポイントは選択したカラーセットのカラーを取得します。**Select Axis** ボックスは、グラフに 2 番目の値の軸がある場合にのみ表示されることに注意してください。デフォルトでは、値軸は最初のカラーセットを使用し、2 番目の値軸は 7 番目のカラーセットを使用します。

データポイントの色を手動で変更すると、選択したカラーセットが自動的に選択解除されます。これは、グラフのデータポイントの色に対応しないカラーセットが原因です。また、チャートのデータポイントのカラーがカラーセットのカラーよりも異なる場合、自動的に次のカラーセットの先頭からカラーを使用することに注意してください。利用可能な次のカラーセットがない場合、代わりに最初のカラーセットが使用されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.3.2.1 カテゴリ機能の色を保存

データポイントの色は、**Data Properties** ダイアログで使用できる **Save Colors for Categories** 機能と密接に関連しています。このダイアログは、ツールバーの アイコンをクリックするか、**Format** > **Data Properties** を使用して開くことができます（この機能はスタックチャートでは使用できません。また、チャートにはすべてのカテゴリでシングルカラーがある場合は無効になります)。この機能がオンになっている場合は、チャートデータポイントの色がカテゴリの名前（またはシリーズのあるチャートのシリーズ）に割り当てられます。カテゴリ（またはシリーズ）がチャートに表示されると、割り当てられた色が自動的に使用されます。機能がオフ（デフォルト）の場合、データポイント順に色が割り当てられます（つまり、最初のデータポイントがカラーセットから最初に使用可能な色を取得し、2 番目の色が 2 番目の色などになります)。カテゴリの色を保存する設定は自動的に.pac ファイルに保存されます。次の例は、この機能がオンとオフのシナリオを示しています。

チャートには、次のカラーセット（青、緑、黄、赤）から色分けされた 3 つのカテゴリ（ "A"、 "B"、"C"）があるとします。

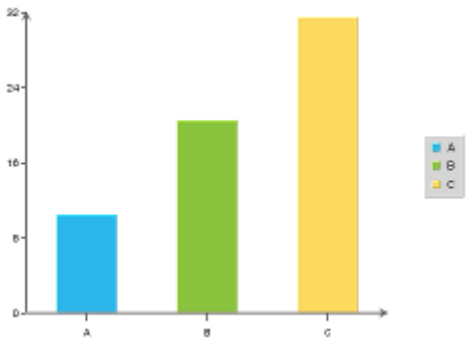


図 1 チャートの例

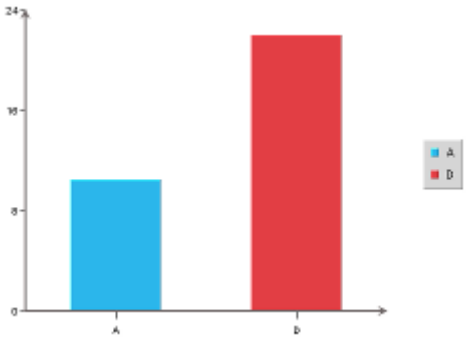


図 2 Save Colors for Categories 機能がオンのとき

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

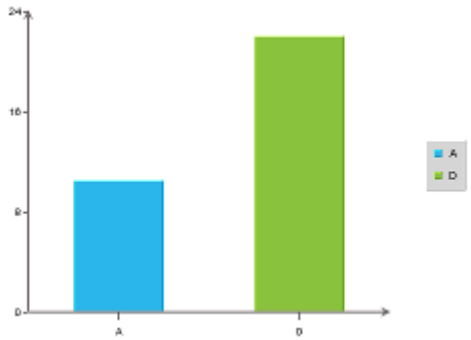


図 3 Save Colors for Categories 機能がオフのとき

Save Colors for Categories 機能がオンのとき

Save Colors for Categories 機能を有効にしてグラフを保存したので（図 1 を参照）、保存されたカテゴリのリストに次のカテゴリと色が保存されました：

カテゴリ A ...青色

カテゴリ B ...緑色

カテゴリ C ...黄色

これで、チャートとデータの変更（たとえば、データにカテゴリ A と D のみが表示 – 図 2 を参照）を開くと、カテゴリの色は次のようになります。

カテゴリ A ...青色（カテゴリ A は保存されたカテゴリのリストに存在するため、カテゴリ A は青色である）

カテゴリ D ...赤色（カテゴリ D は、保存されたカテゴリのリストに存在しないため、カラーセット内で次の利用可能な色を使用します。この場合、カテゴリは青、緑、黄はカテゴリ A、B、C にすでに割り当てられています。

このシナリオでは、データセットの色がカラーセットに対応していないため、**Color Set** タブではカラーセットは選択されません。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Save Colors for Categories 機能がオフのとき

同じ状況ですが、グラフの **Save Colors for Categories** 機能がオフになっているとします（図 3 を参照）。今回は、保存されたカテゴリのリストが空になります。

チャートを開くと、カテゴリの色は次のようになります。

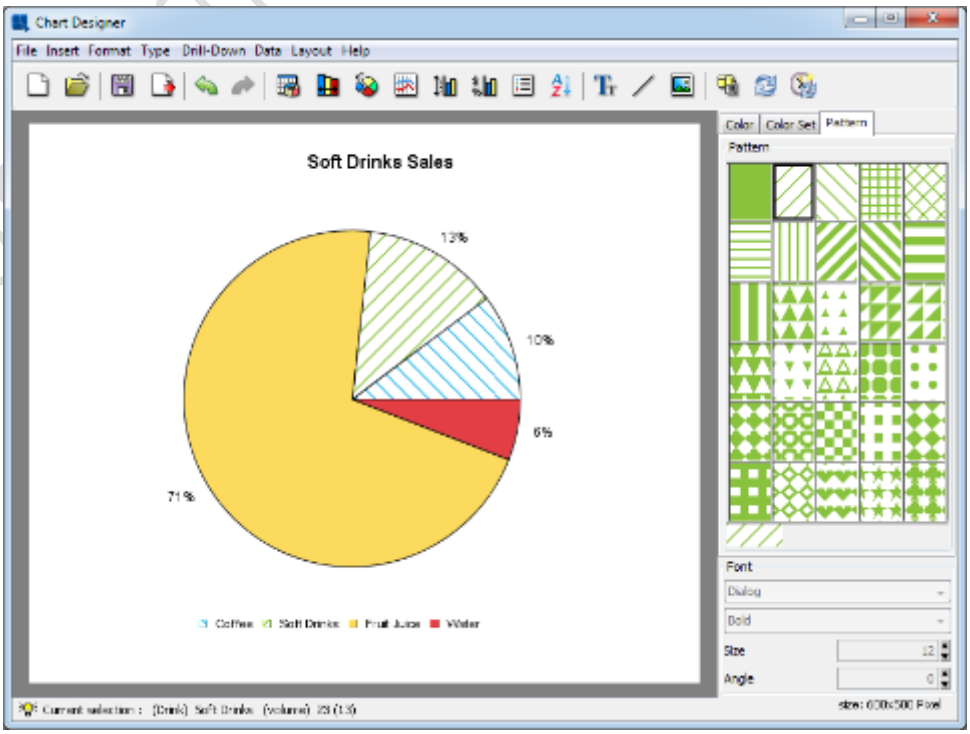
カテゴリ A ...青色（カテゴリ A は、カラーセットから最初に利用可能な色であるため、青色を有する）
カテゴリ D ...緑色（カテゴリ D は、最初にカテゴリ A が割り当てられているため、カラーセットの 2 番目の色を持ちます）

このシナリオでは、データセットの色がカラーセットに対応するため、**Color Set** タブでカラーセットが選択されます。

5.5.3.3 パターンパネル

色およびフォントパネルとは異なり、パターンパネルはデータポイントにのみ適用できます。使用できる定義済みのパターンパレットがあります。カラーパネルの使用方法と同様に、まず変更したいデータポイントを選択し、パターンパレットから任意のパターンを選択する必要があります。パターンはデータポイントに直接適用されます。

データポイントに対してパターンがすでに定義されている場合でも、ユーザはカラーパネルタブを選択し、カラーパレットから別のカラーを選択することによって、カラーを変更することができます。パターンの色はすぐに新しい色に変更されます。パターンパレットに表示されるパターンも新しい色に変更されます。

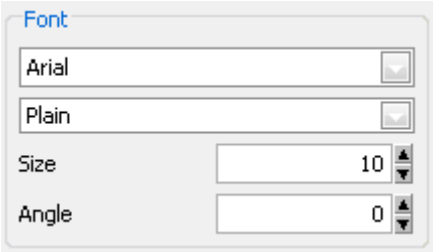


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

パターン例

5.5.3.4 フォントパネル

フォントパネル、フォントスタイル、フォントサイズ、ラベル、タイトル、またはその他のテキストのフォント角度をグラフの中で変更するには、フォントパネルを使用します。フォントを変更するには、まずフォントをクリックして変更したいテキストオブジェクトを選択する必要があります。チャートデザイナウィンドウの下部にあるステータスバーには、現在選択されているオブジェクトが表示されます。



Font パネル

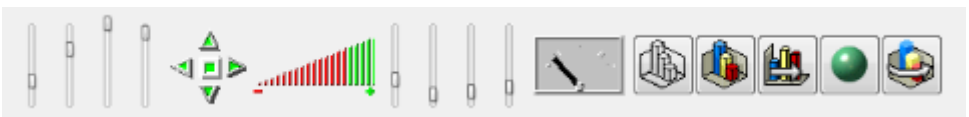
最初のドロップダウンボックスでは、使用するフォントを選択できます。2 番目のドロップダウンボックスでは、フォントスタイルをプレーン、ボールド、イタリック、太字のいずれかで選択できます。 3 番目のボックスでは、フォントサイズを選択できます。 最後のボックスでは、テキストの角度を指定できます。

特定のテキストグループ（軸ラベルまたはデータトップラベル）はすべて同じプロパティを持ちます。したがって、テキストの 1 つを選択してフォントのプロパティを変更すると、それらのすべてに適用されます。

Report Designer では、Java グラフィックスライブラリを使用してテキストをよりクリーンな外観にすることができます。通常のテキストの場合は、**Format > Rendering Options** を選択して、グラフのアンチエイリアス機能を使用できます。回転テキスト（テキストが 0 度ではない）の場合は、**Format > Text Properties** を選択して、Java 2D 回転テキスト機能を使用できます。これらのメソッドには、Java 1.2 以上が必要です。

5.5.4 ナビゲーションパネル

ナビゲーションパネルには、3D チャートに固有のいくつかのプロパティを制御するオプションがあります。2D グラフには表示されず、**Layout > Show Navigation Panel** の表示オプションをトグルすることで 3D グラフのために非表示にすることができます。



ナビゲーションパネル

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ナビゲーションパネルには 6 つのコントロールと 5 つのボタンがあります。左から順に、ステップサイズ、X、Y、Z 軸のライトポジション、ナビゲーション、ズーム、スケール、ナビゲーションの 6 つのコントロールがあります。右のボタンは、ワイヤーフレーム/ソリッドモード、オン/オフ枠線描画、オン/オフインラインシリーズ（シリーズの柱状図と棒グラフ用）、Gouraud シェーディングのオン/オフ、オン/オフアニメーションを制御します。

Step size:

このスライドバーでは、ナビゲーションコントロールが 3D チャートを回転または移動するために使用する増分量を設定できます。水平バーの位置が高いほど、各ステップの増分が大きくなります。

Light position for X, Y, and Z:

ユーザ定義の光の位置は、光の位置を制御するのに役立ちます。したがって、各軸で光が当たる方向を制御することができます。

Navigation:

このコントロールを使用すると、チャートを回転または平行移動できます。中央のボタンはトグルスイッチです。押された状態（ボタンが赤色）にあるとき、4 つの三角形のボタンのいずれかをクリックすると、そのボタンによって示される方向にチャートが移動します。中央のボタンが押されていない状態にあるとき、4 つの三角形のボタンのいずれかをクリックすると、そのボタンによって示される方向にチャートが回転します。ナビゲーション速度は、ナビゲーション速度制御によって制御することができます。

Zoom:

グラフを効果的にビュアーに近づけたり遠ざけたりすることができます。操作するには、マウスをコントロールに向け、マウスの左ボタンをクリックします。次に、ボタンを押しながらマウスを左右にドラッグします。マウスを右にドラッグすると、赤い部分が右に広がり、グラフが近くに表示されます（グラフを拡大するためにズームイン）。マウスを左にドラッグすると、赤色のエリアが左に移動し、グラフがあなたから遠ざかるように表示されます（縮小してグラフを縮小します）。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Scale:
これは 4 つのスライドバーのセットです。最初の 3 つのバーでは、それぞれ X、Y、Z 軸のスケールファクタを変更できます。4 番目のバーは、3D の横棒、縦棒、折れ線、または円グラフの太さを決定します（グラフの種類によって異なります）。バーの位置が高いほど値は大きくなります。2D チャートでは、**Format > Data Properties** を選択して、横棒、縦棒、積み重ね横棒、積み重ね縦棒、HighLow、HLCO チャートのバー幅を調整できます。

Animation speed:
チャートナビゲーションの速度を設定できます。このコントロールは、ナビゲーションコントロールとアニメーションオン/オフスイッチで便利です。 チャートの移動速度または回転速度を決定します。 操作するには、指示針をクリックして目的の位置までドラッグします。 インジケータ針を右に動かすと平行移動または回転が加速し、針を左に動かすと速度が遅くなります。

これに加えて、5 つのボタンが使用可能です。

アイコン	名前	説明
	Wire frame/solid mode On/Off	スイッチがオンのときワイヤーフレームとして、スイッチがオフのときは立体オブジェクトとして 3D チャートを表示できます。
	Border Drawing On/Off	スイッチがオンのときにグラフの各端に黒い枠線を描画します。
	InlineSeries On/Off	シリーズの列を同じ XY 平面に描画します。このオプションは、データシリーズを含む縦棒グラフと棒グラフでのみ使用でき、チャートタイプが適用されない場合はナビゲーションパネルに表示されません。
	Gouraud Shading On/Off	Gouraud シェーディングは、3D チャートの洗練された非常に現実的なシェーディング機能です。スイッチがオンになると、チャートを各サーフェスのシェードにレンダリングし始めます。
	Animation On/Off	3D グラフのアニメーションを開始/停止することができます。アニメーションの速度は、ナビゲーションスピードコントロールを使用して設定できます。 アニメーション中、アニメーション速度コントロールを除くすべてのパネルコントロールは無効になります。

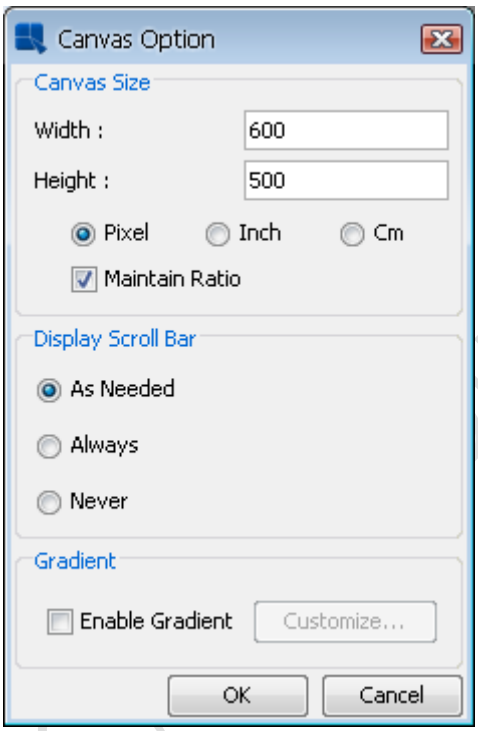
5.5.5 ビューポート

ビューポートは、Chart Designer ウィンドウの中央部分を傷つけます。ビューポート内でキャンバス上のさまざまなチャート要素すべてを選択、移動、およびサイズ設定できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.5.1 チャートキャンバス

チャートキャンバスは、すべてのチャート要素が描画される背景です。その寸法は完成したチャートのサイズです。**Format** > **Canvas** を選択して、チャートキャンバスのサイズを変更できます。新しいキャンバスの寸法を指定するダイアログが表示されます。



Canvas Formatting ダイアログ

キャンバスのサイズは、ピクセル、インチ、またはセンチメートルで指定できます。このダイアログから、ビューポートでスクロールバーを使用するタイミングを指定することもできます。デフォルトでは、キャンバスがウィンドウより大きい場合、ビューポートにスクロールバーが表示されます。キャンバスがビューポートウィンドウより小さい場合は、周囲に濃いグレーのエリアが表示されます。

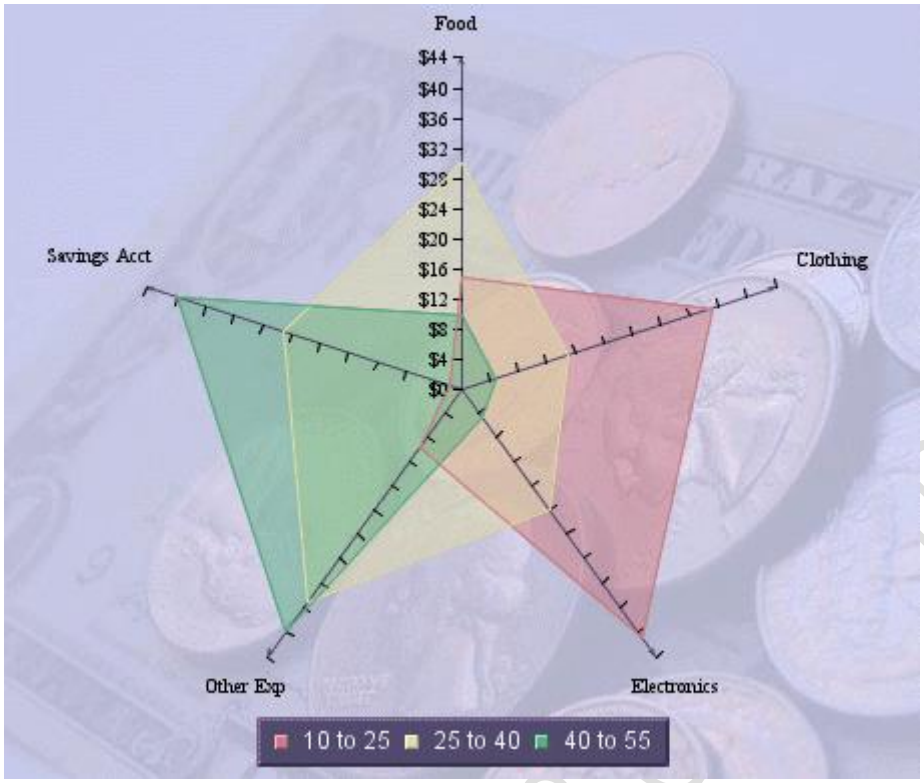
レポートに埋め込みチャートを作成する場合、キャンバスは Report Designer のチャートに定義したスペースに合わせてサイズが変更されます。たとえば、レポートのスペースを 3 インチ×4 インチに設定した場合、次にチャートを編集するときに、キャンバスのサイズは自動的に 3 インチ×4 インチになります。

このダイアログでは、キャンバスのグラデーションの背景を設定することもできます。グラデーションの設定は、フォーマットメニューで説明しているレンダリングオプションと同じです。

5.5.5.1.1 背景画像

プレーンまたはカラーのキャンバスではなく、チャートの背景としてイメージを追加できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

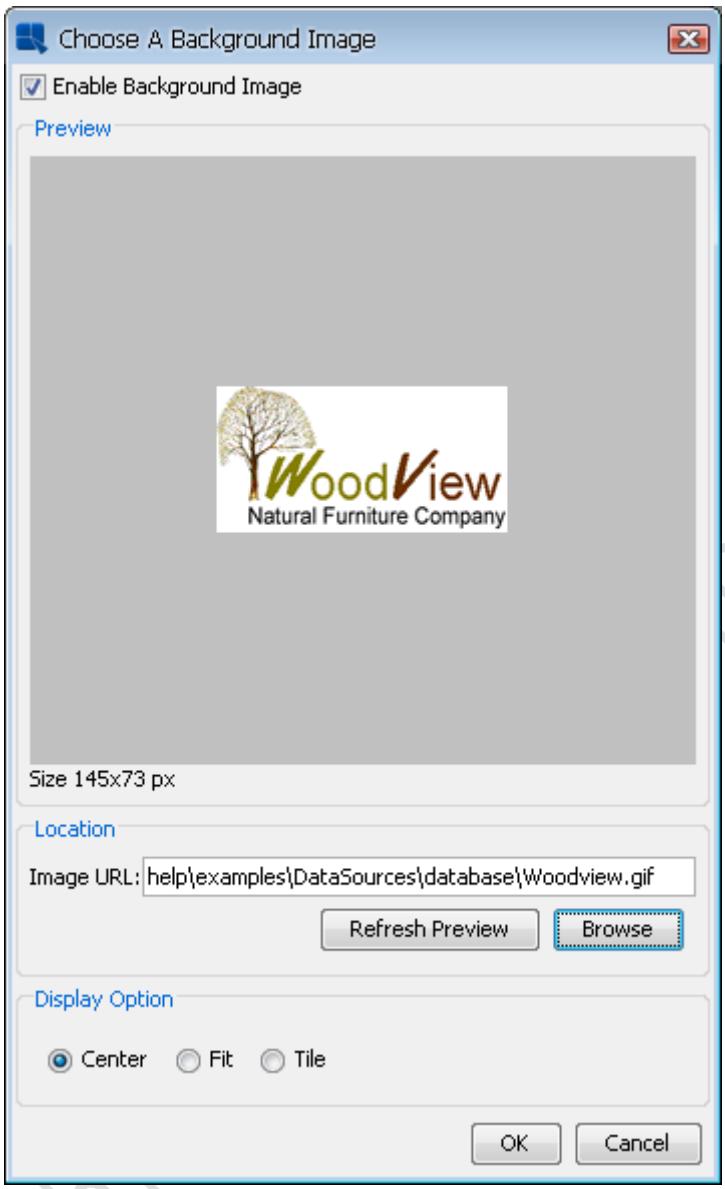


背景つきレーダーチャート

Insert > **Background** を選択するか、ツールバーの背景ボタン をクリックして、背景画像を追加できます。これにより、グラフの背景画像を指定できるダイアログが表示されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Add Background Image ダイアログ

新しい画像を挿入するには、**Enable background** オプションを選択します。既存の背景画像を削除し、単純な背景色を使用する場合は、このオプションの選択を解除します。

背景画像を挿入するには、ローカルからイメージを配置するか、URL からイメージを取得する 2 つの方法があります。

ローカルの画像を見つけるには、**Browse** ボタンをクリックします。

URL からイメージを取得するには、**Image URL** テキストフィールドに URL を入力します。その後、**Refresh Preview** ボタンをクリックして URL を確認します。URL の画像が **Preview** セクションに表示される場合は、URL が正しいことを示します。

背景画像を追加してチャートを TPL または CHT として保存すると、イメージ自体がチャートに保存されません。パスまたは URL のみが保存されます。TPL チャートまたは CHT チャートを移動する場合は、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

指定されたパスに沿ってイメージにアクセスできることを確認する必要があります。チャートを PAC として保存すると、背景画像はチャートと共に PAC ファイルに保存されます。

5.5.5.2 チャート要素の移動とサイズ変更

チャート内の任意の要素をクリックして選択することができます。Chart Designer ウィンドウの下部にあるステータスバーには、現在選択されているオブジェクトが表示されます。オブジェクトをクリックしてドラッグすると、チャートキャンバスの周りを移動します。凡例のような他のオブジェクトは独立して動くのに対し、軸やデータのトップラベルのようなオブジェクトは連動して移動することに注意してください。

プロット領域をクリックしてマウスをドラッグすると、グラフプロット全体を移動できます。キャンバスの周りのチャート全体が移動します。グラフのサイズを変更するには、プロット領域をクリックし、プロット領域の隅と辺に 8 点を表示します。マウスを使用してこれらのポイントのいずれかをドラッグして、プロットエリアを拡大または縮小します。ナビゲーションパネルのズームコントロールを使用して、3D チャートのサイズを変更することもできます。

5.5.6 チャート要素の追加

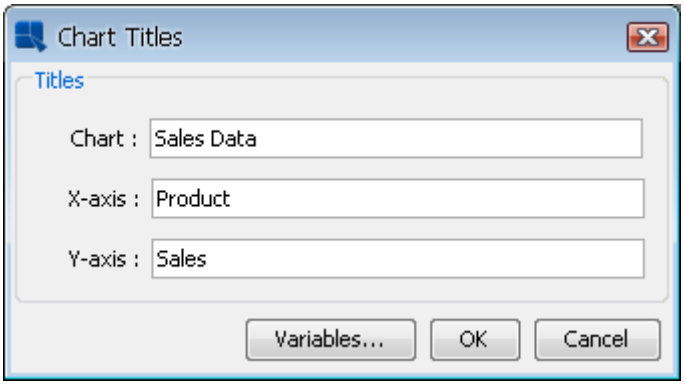
デフォルトのチャート要素に加えて、EspressChart には、チャートに追加できるいくつかの追加要素が用意されています。

5.5.6.1 テキストを追加

チャートにテキストを追加するには、タイトル、またはプレーンテキストとしての 2 つの方法があります。

タイトルを追加

チャートにタイトルを追加するには、**Insert** > **Titles** を選択します。これにより、チャートのタイトルを入力するダイアログが表示されます。



Add Titles ダイアログ

このダイアログでは、チャートのメインタイトルと各軸のタイトルを指定できます。軸を持たない円グ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

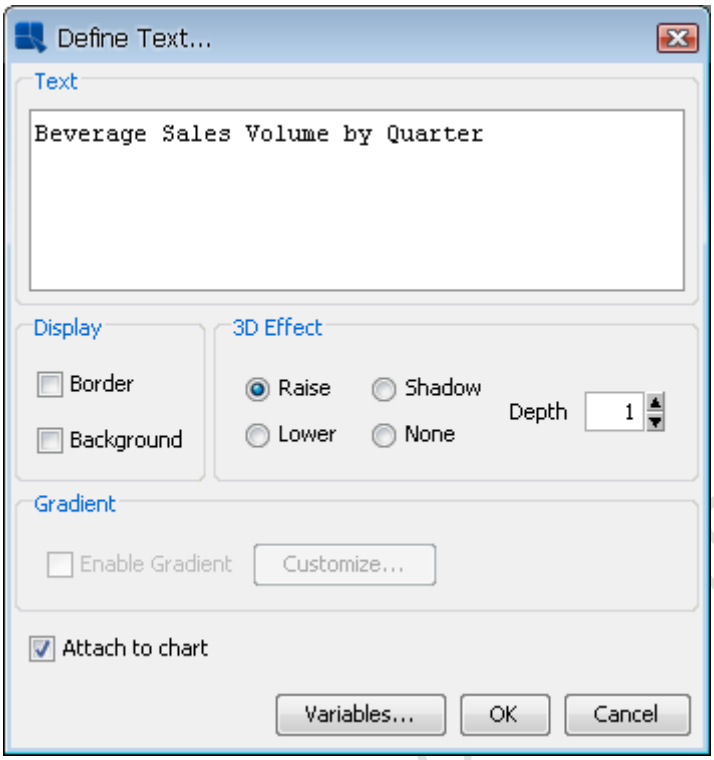
ラフチャートとダイヤルチャートは、チャートのタイトルのみを表示します。タイトルの指定が終了したら、**OK** をクリックすると、チャートに追加されます。タイトルは自動的に配置され、サイズが決まります。タイトルの移動やフォントの変更も行うことができます。

テキストを追加

個々のテキストフィールドをチャートに追加するには、**Insert** > **Text** を選択するか、ツールバーの **Add Text** ボタン をクリックします。グラフにテキストを追加するように促すダイアログが表示されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Add Text ダイアログ

このダイアログでは、テキストを指定したり、表示オプションを設定したりできます。テキストの周りに、または背景の周りに枠線を描画するかどうかを指定できます。また、背景に適用する効果を指定することもできます。

このダイアログでは、テキストラベルのグラデーションの背景を設定することもできます。グラデーションの設定は、フォーマットメニューで説明しているレンダリングオプションと同じです。

テキストの指定が完了したら、**OK** をクリックします。これで、グラフにテキストを配置することができます。小さいキャンバスは、グラフキャンバスの周りのポインタの後ろにあります。テキストを配置するマウスをクリックします。

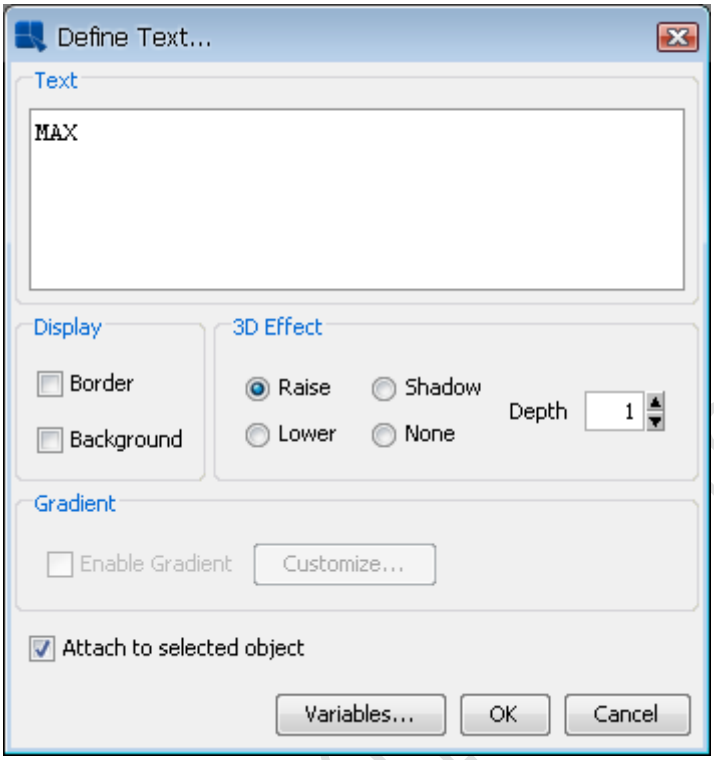
注釈テキスト:

ChartDesigner は注釈もサポートしています。Annotation では、ラベルやテキストフィールドを線やチャートプロットなどの特定の要素に添付することができます。たとえば、最大値を示すコントロールラインをグラフに挿入し、このコントロールラインに **MAX** というテキストラベルを付けることができます。最大値が変わるたびに、ラベルはコントロールラインとともにその位置を調整します。コントロールラインの詳細については、固定水平/垂直ラインを参照してください。

注釈にするテキストは 2 つの方法で指定することができます。グラフプロットにテキストを添付するには、テキストを追加するときに **Attach to Chart** オプションを選択します。コントロール行のような特定のオブジェクトにテキストを添付するには、まずオブジェクトを選択し、**Insert** > **Text** を選択するか、ツールバーの **Add Text** ボタン をクリックします。**Attach to selected object** オプションが自

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

動的にチェックされます。チェックしたままにすると、追加したテキストは自動的にオブジェクトに添付されます。



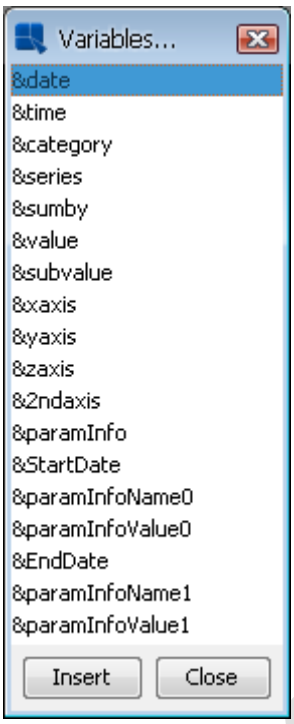
Add Text(Annotation) ダイアログ

5.5.6.1.1 テキストの変数

ChartDesigner では、チャート内の特定の値/オブジェクトに基づいて実行時の置換を可能にするテキスト内の特定の変数を指定することができます。たとえば、グラフでパラメータ化されたクエリをデータソースとして使用する場合は、**&paramInfo** 変数を使用して、実行時に選択されたパラメータ値を表示できます。

Insert titles ダイアログと **add text** ダイアログには、下に **Variables** と表示されたボタンがあります。これにより、使用できる変数のリストを含むダイアログが表示され、タイトルまたはテキストに追加する変数を選択することができます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Variables ダイアログ

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

以下のテキスト変数がサポートされています。

変数	説明
&drillInfo	ドリルダウンチャートでどのデータポイントがドリルダウンされているかが表示されます。この変数は、パラメータドリルダウンチャートでは機能しません。
&paramInfo	選択されたパラメータ値が表示されます。パラメータドリルダウンチャートに&drillInfoの代わりにこの変数を使用できます。
&date	チャートが最後に描画/再描画された日付を表示します。
&time	チャートが最後に描画/再描画された時間を表示します。
&category	カテゴリ列の名前を表示します。
&series	シリーズ列の名前を表示します。
&sumby	sum-by列の名前を表示します。
&value	値列の名前を表示します。
&subvalue	2つ目の値の列の名前を表示します。
&xaxis	X軸にマップされている列の名前を表示します。カテゴリの代わりに値をスキャッターチャートやバブルチャートのようなX軸にマップするチャートのために使用されます。
&yaxis	Y軸にマップされている列の名前を表示します。スキャッター、バブル、サーフェスチャートに使用されます。
&zaxis	Z軸にマップされている列の名前を表示します。スキャッター、バブル、サーフェスチャートに使用されます。
&2ndaxis	第2列にマップされている列の名前を表示します。
&paramInfoName<index>	チャートにパラメータが含まれている場合は、選択したインデックスのパラメータ名が表示されます。
paramInfoValue<index>	チャートにパラメータが含まれている場合は、選択した値のパラメータ名が表示されます。
&<<paramName>>	チャートにパラメータが含まれている場合は、そのパラメータに選択された値が表示されます。

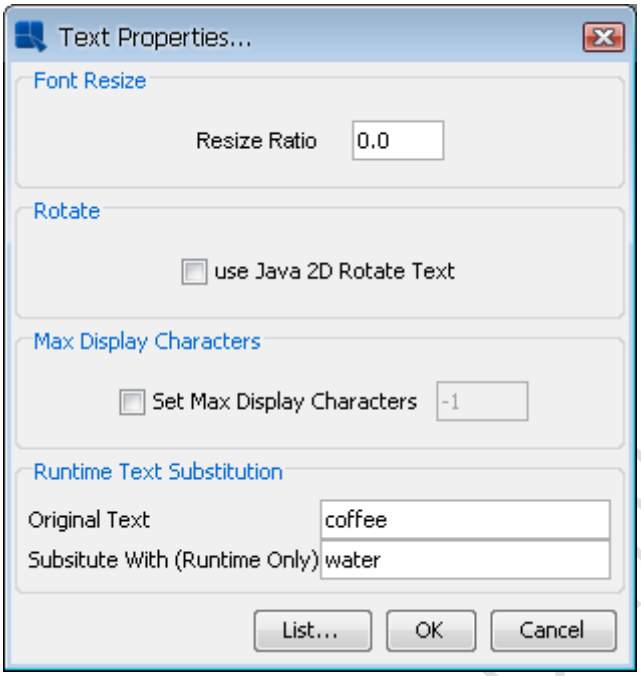
5.5.6.1.2 テキストの置換

ChartDesignerでは、チャート内の特定のテキストを上書きすることができます。これは、データソースが特に直感的な列名を使用しない場合に便利です。この機能は、テキストのすべてのインスタンスを置き換えることに注意してください。たとえば、X軸ラベルと凡例項目の列名を表示するシリーズがない縦棒グラフがある場合、テキスト置換を使用して凡例項目だけでなくラベルのみを変更することはできません。テキスト置換機能は文字列全体だけを変更し、部分的に一致するインスタンスは変更しません。

テキストの置換を使用するには、**Format** > **Text Properties** を選択します。置換されたテキストを指

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

定するためのダイアログが表示されます。



Text Properties ダイアログ

同じテキストを連続して変更する場合は、元のテキストを使用する必要があります。たとえば、**coffee** という単語を **water** に置き換えたとします。ここで、**water** を **soft drink** に変更したい場合は、元のテキスト（「**coffee**」、次に「**soft drink**」）をテキストに置き換える必要があります。テキスト置換を削除するには、元の文字列を置換するだけです。したがって、同じ例を使用して、「**coffee**」を「」に置き換えます。

リストボタンをクリックすると、元のテキストと置換えのリストがすべて表示されます。 これにより、チャート内のすべてのテキスト置換をリストするダイアログが表示されます。

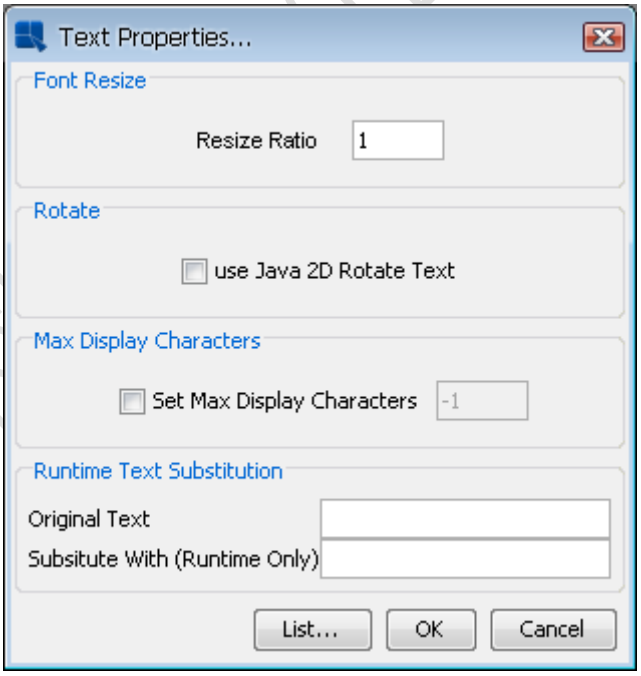
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

置換テキストリスト

このダイアログでは、置き換えられたテキストのいずれかを選択し、ダイアログの下部にあるボタンをクリックして置換を変更または元に戻すことができます。

5.5.6.1.3 テキストの自動リサイズ

ChartDesigner には、サイズ変更時にチャート内のテキストのフォントサイズを自動的に調整する機能があります。これは、同じチャートテンプレートを使用して、さまざまなサイズのチャートを作成する場合に便利です。チャートキャンバスの変更に基づいて調整するフォントサイズの比率を指定できます。サイズ変更比を指定するには、**Format** > **Text Properties** を選択します。



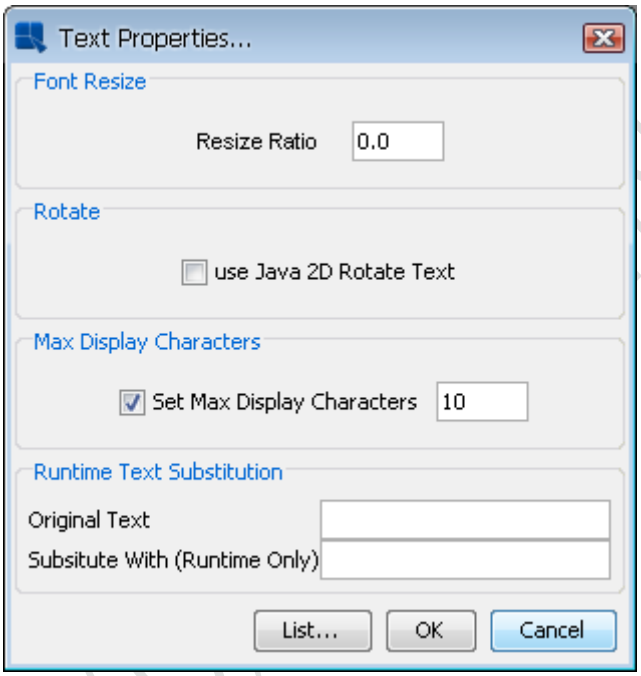
Text Properties ダイアログ

比率は、フォントがキャンバスに関してリサイズする相対的な割合を決定します。たとえば、500×500 ピクセルから 250×250 までのグラフのサイズを変更する場合、サイズ比が 1 の場合、12 ポイントフォントのテキストはキャンバスと同じパーセントで 6 ポイントに減少します。しかし、リサイズ率が 0.5 の場合、フォントはキャンバスの半分になりますので、12 ポイントのフォントは 9 ポイントに減少します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.6.1.4 テキストの切り取り

チャート内の長いラベルやテキストは、グラフプロット内で余りにも多くのエリアを占有することがあり、実際のグラフの余地はほとんどありません。このような状況に対して、EspressChart はチャートテキストのテキストクロッピング機能を提供します。ユーザ指定のしきい値よりも長いテキストは、「...」で切り捨てられます。チャートのヒントボックスには、ラベル全体が表示されます。テキストのクロッピングを指定するには、**Format** > **Text Properties** を選択します。



Text Properties ダイアログ

テキストのクロッピングを有効にするには、**Set Max Display Characters** オプションをオンにし、ダイアログ内の最大文字長を指定します。指定された文字数より長いテキストはクロッピングされます。

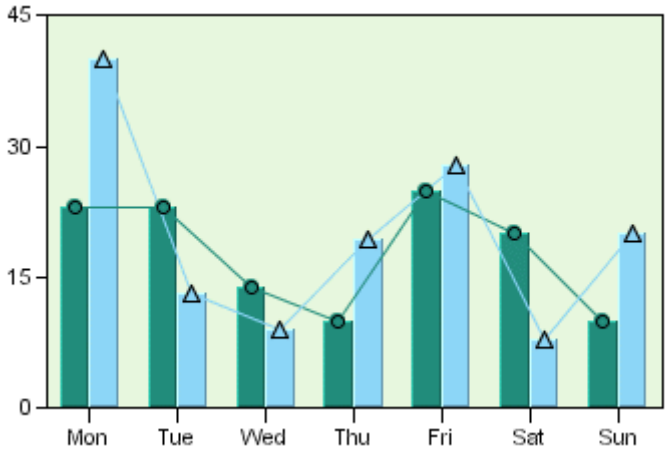
5.5.6.2 ラインの追加

EspressChart では、チャートのさまざまな種類の行を追加および書式設定できます。

5.5.6.2.1 線と点の書式設定

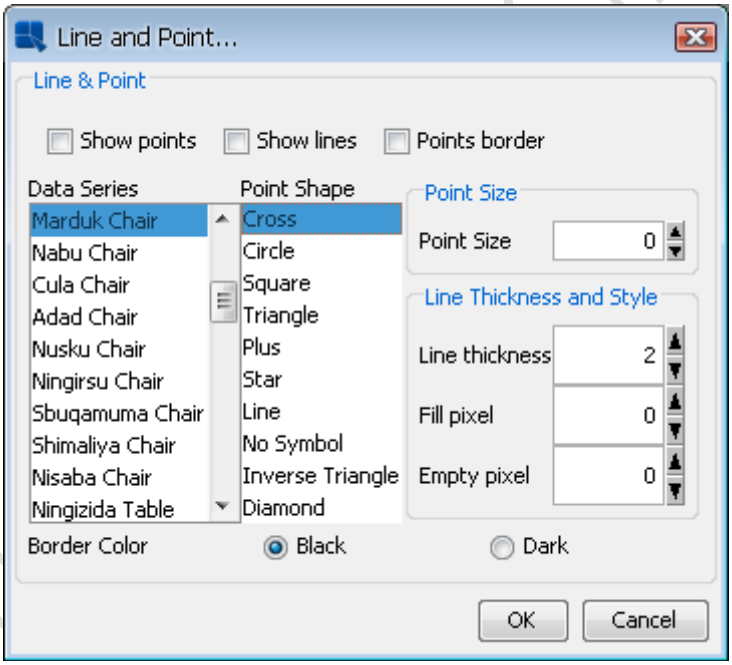
任意の 2D チャートタイプに対して、チャート内のすべてのデータポイントの線と点を表示するように選択できます。 一部のグラフタイプではすでにこの表現（折れ線グラフまたは散布図）が使用されています。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



ラインとポイントが定義された縦棒グラフ

線と点の表示は、**Format** > **Lne and Point** を選択するか、ツールバーの線と点のボタン をクリックして制御します。これにより、いくつかのオプションを提示するダイアログが表示されます。



Line and Point ダイアログ

最初の 3 つのオプションでは、グラフの線、点、および点の枠線を表示するかどうかを指定できます。レーダー、散布図、極座標チャートでは、エリアを表示するオプションもあります。レーダーと極座標の場合、エリアオプションはデータポイントで囲まれたエリアを埋めるでしょう。散布図では、X 軸からデータ点までの列を描画します。残りのオプションを使用すると、データシリーズの各要素の線と点の表示をカスタマイズできます。

データシリーズ要素ごとに、使用するポイントシェイプを指定できます。また、ポイントのサイズを制御することもできます。デフォルトのポイントサイズは 0 です。ポイントサイズは、それぞれ 0.75,0.5、および 0.25 のサイズを表す-1、-2、および-3 を指定できます。-3（0.25）では、選択したポイントの形状に関係なく、点が点として描画されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ラインの場合、ラインの太さを指定することや、ダッシュパターンをカスタマイズすることができます。ダッシュパターンは、塗りつぶされたピクセルの数と空のピクセルの数（0～255 の間）を指定することによって作成されます。その後、線分は、セグメントに分割されて描画されます。塗りつぶされたピクセルの数に続いて空のピクセルの数が続きます。両方に 0 を設定すると、実線になります。両方を 255 に設定すると、線が描画されません。

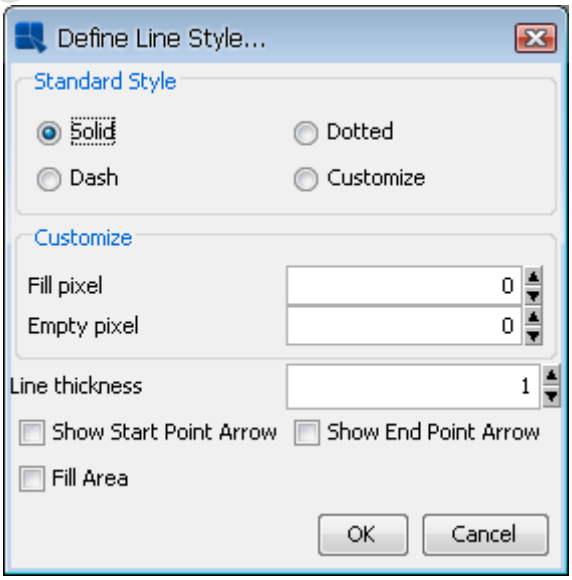
最後のオプションでは、データポイントのシンボルの枠線の色を黒または濃い色のシンボル色に変更できます。

5.5.6.2.2 浮動線

フローティングラインは、チャートキャンバスの任意の場所に任意に追加できる自由形式の線です。多くの場合、フローティングラインはチャート内の特定の要素を指すために使用されます。フローティングラインを追加するには、**Insert** > **Line** を選択するか、ツールバーの **Insert Line** ボタン をクリックします。

このオプションを選択すると、マウスポインタが十字に変わります。グラフキャンバス内をクリックして、行を開始します。クリックを追加するごとにポイントが追加され、別のセグメントを追加することができます。この方法でフローティングラインを使用して図形を描画することもできます。終了したら、右クリックして描画を停止します。その行が追加されます。

キャンバス上に線を配置すると、線を個別に移動することはできません。注釈テキストのようなチャートプロットとともに移動します。フローティングラインのオプションを指定するには、まずそれを選択し、**Format** > **Line and Point** を選択するか、ツールバーの **Line and Point** ボタンをクリックします。これにより、フローティングラインのさまざまなプロパティを書式設定できるダイアログが表示されます。



Copyright(C) 2018 Climb.Inc.

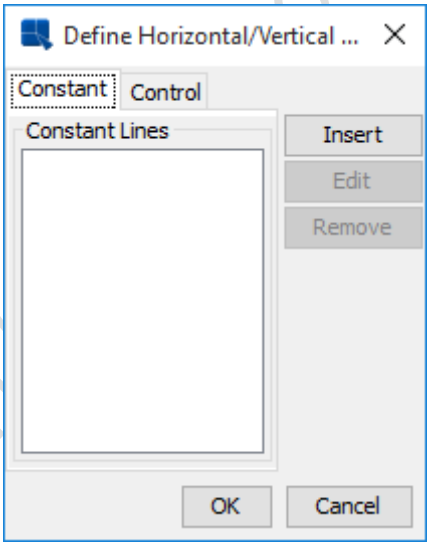
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フローティングラインの Line and Point ダイアログ

このダイアログでは、標準の線種を指定したり、線や点の書式設定と同じ方法でカスタムダッシュパターンを作成したりすることができます。線の太さをピクセル単位で指定することもできます。ダイアログの下部にあるチェックボックスを使用すると、線の始点および/または終点に矢印を配置し、線で囲まれたエリアを塗りつぶしてソリッドシェイプを作成することができます。

5.5.6.2.3 固定水平/垂直線

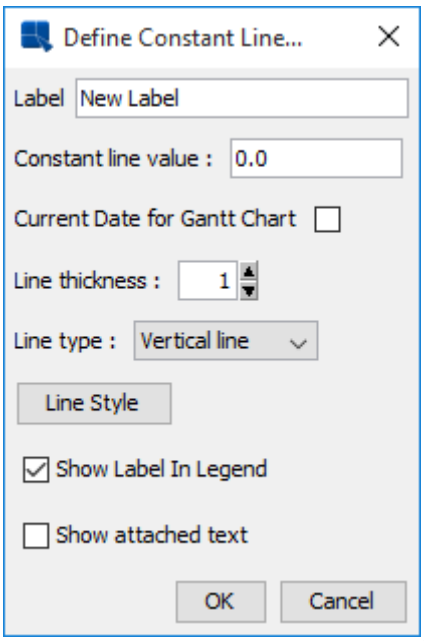
固定された水平線または垂直線は、グラフ軸の 1 つに描画される線です。これらの線は、平面として表示される 3D チャート上に描画することもできます。固定線には、定数線とコントロールラインの 2 種類があります。定数線は、軸の特定の値に固定された線です。コントロールラインは、特定の値の範囲外にあるデータ点を見つけることを可能にする計算値に基づいて描かれます。いずれかの種類の線をグラフに追加するには、**Insert** メニューから **Horz/Vert Line** を選択します。既存の水平/垂直線のリストを表示するダイアログが表示され、選択した線を編集したり、選択した線を削除したり、新しい線を作成することができます。



Define Horizonal/Vertical Lines ダイアログ

既存の行が選択されているときに **Insert** をクリックするか、**Edit** をクリックすると、選択した行を設定するダイアログが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Constant Line ダイアログ

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

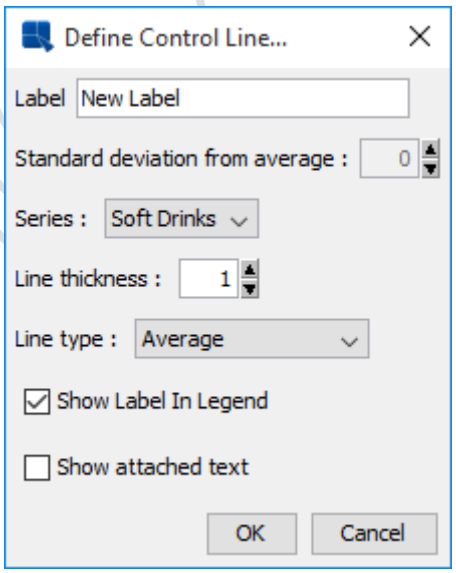
定数行の場合は、行に使用する数値だけでなく、行のラベルも指定する必要があります。カテゴリ軸の場合、データポイントは 0.5 から始まることに注意してください。線の太さと線が水平か垂直かを指定することもできます。最後の 2 つのオプションを使用すると、行のチャート凡例に項目を追加することや、その行に注釈を表示するかどうかを指定できます。

ガントチャートには、ガントチャートの現在の日付というオプションが 1 つあります。このオプションを選択すると、**Constant line value** フィールドは無効になり、現在の日付が行の値として使用されます（行の位置は、グラフを実行するたびに更新されます）。

レーダーチャートの場合、水平/垂直オプションは無効です。レーダーチャートは、（レーダーグリッドと同様の方法で）すべてのチャート軸の周りの同じポイントに一定/コントロールラインを描画します。さらに、レーダーチャートでは、**Circular Style** というオプションが追加されています。デフォルトでは、レーダーチャートの線分はセグメント化された形で描画されます。直線は各軸の点を結びます。このオプションを選択すると、定数/制御が円として描画されます。

その行に注釈を指定していない場合は、最後のオプションを選択した場合は none が表示されます。注釈をチャートに追加する方法の詳細は、テキストを追加を参照してください。

コントロールラインを追加するには、ダイアログの **Control Line** タブをクリックします。



Control Line ダイアログ

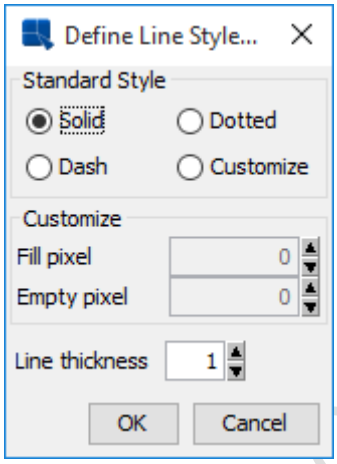
コントロールラインでは、値を計算するシリーズ要素を指定する必要があります（シリーズがない場合は表示されません）。また、値を計算する方法もあります。コントロールラインのオプションは、平均、最小、最大、および標準偏差の倍数です。

すべてのオプションを指定すると、その行がグラフに追加されるか、選択された行がそれぞれ変更されます。前のダイアログで指定されたプロパティを編集するには、もう一度 **Insert** > **Horz/Vert Line** を

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

選択し、リストから行を選択します。変更したい行をダブルクリックすることもできます。

最初に線を選択してから、**Format** > **Line and Point** を選択するか、ツールバーの **Line and Point** アイコンをクリックして、固定線のデザインを変更することができます。行をカスタマイズするためのダイアログが表示されます。

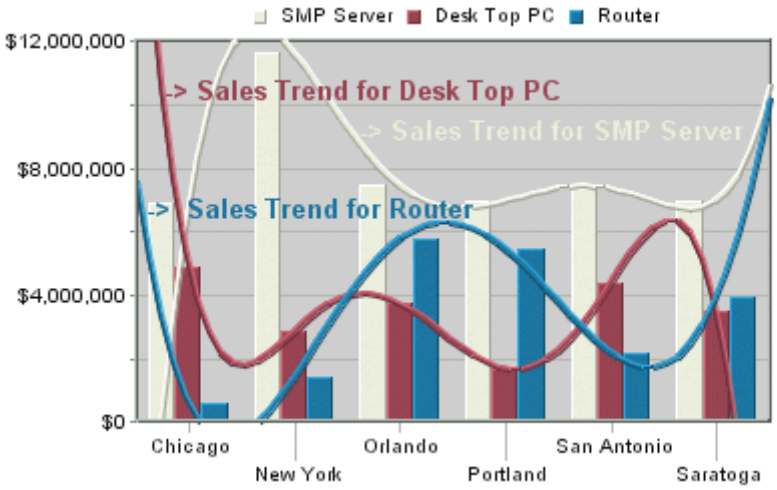


Line and Point ダイアログ

このダイアログでは、標準の線スタイルを指定したり、線や点の書式設定と同じ方法でカスタムダッシュパターンを作成したりすることができます。

5.5.6.2.4 トレンドライン

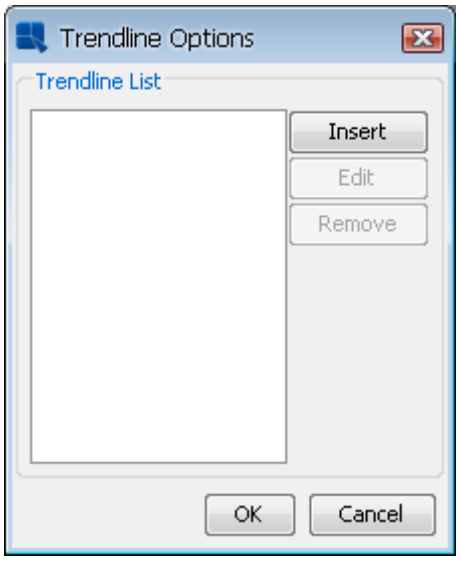
ChartDesigner の強力な機能は、チャートにトレンドラインを追加する機能です。トレンドラインは、特定のトレンドを公開し強調表示することにより、チャートのデータの詳細を表示するのに役立ちます。



トレンドラインチャート

グラフにトレンドラインを追加するには、**Insert** > **Trend Line** を選択します。既存のトレンドラインのリストを表示するダイアログが表示され、選択したトレンドラインを編集したり、選択したトレンドラインを削除したり、新しいトレンドラインを作成したりすることができます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

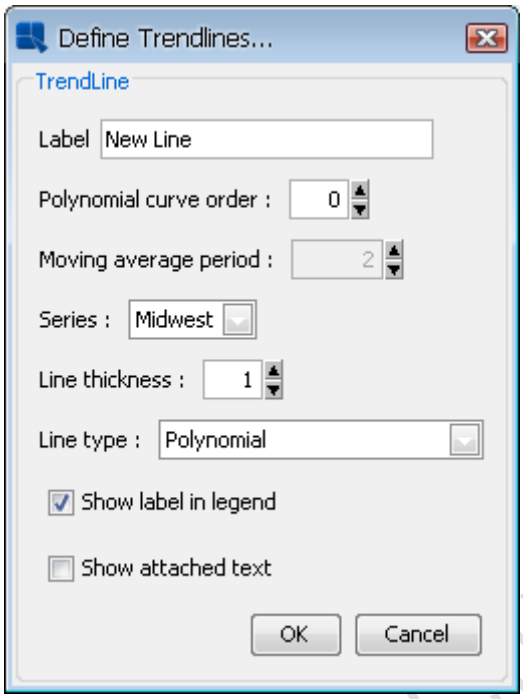


Trend Line Options ダイアログ

既存のトレンドラインが選択されているときに **Insert** をクリックするか、または **Edit** をクリックすると、選択したトレンドラインを設定するダイアログが表示されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Define Trend Lines ダイアログ

このダイアログでは、行のラベルと、計算の基礎となるデータシリーズの要素を指定することができます。以下のタイプのトレンドラインがサポートされています。

- 任意の次数の多項式（線形トレンドラインは 1 次の多項式トレンドラインです。つまり、多項式曲線次数オプションは 1 に設定する必要があります）
- 指数関数対数
- 移動平均
- 指数移動平均
- 三角移動平均
- 三次 B スプライン関数
- 正規分布曲線

移動平均の場合は平均期間を指定する必要があり、多項式の場合はカーブ順序を指定する必要があります。また、線の太さを指定して、凡例のラベルと添付テキストを表示するかどうかを設定することもできます。グラフにデータシリーズがある場合は、特定のシリーズのトレンドラインを設定できます。

すべてのオプションを指定すると、トレンドラインがチャートに追加されるか、選択されたトレンドラインがそれぞれ変更されます。前のダイアログで指定したプロパティを編集するには、挿入メニューから **TrendLine** を選択し、リストから行を選択します。トレンドラインのデザインは、まずそれを選択してから、**Format** > **Line and Points** を選択するか、ツールバーの **Line and Point** ボタンをクリックして変更できます。これにより、ラインをカスタマイズするためのダイアログが表示されます。

Copyright(C) 2018 Climb.Inc.

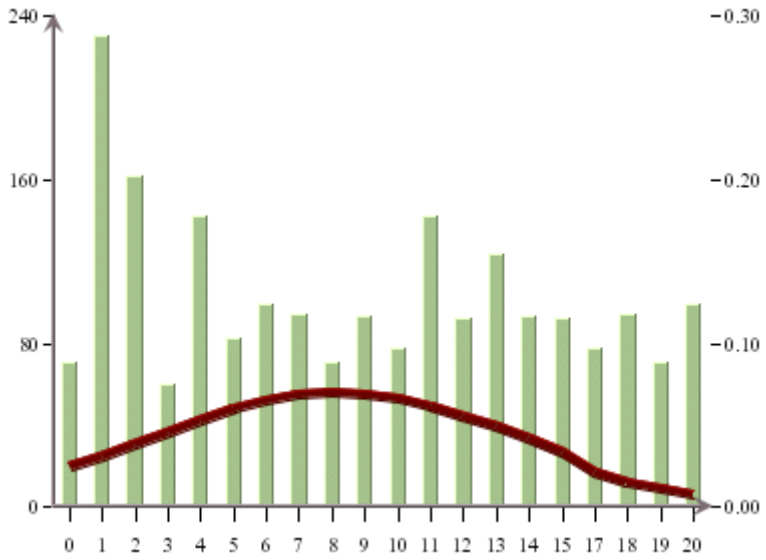
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

トレンドラインの Line and Point ダイアログ

このダイアログでは、ラインとポイントの書式設定と同じ方法でカスタムダッシュパターンを作成できます。

正規分布曲線

グラフのデータの正規分布曲線を描くことができる特別なタイプのトレンドラインです。正規分布曲線をプロットするには、グラフは 2D の縦棒グラフまたは棒グラフでなければならず、データシリーズを持つことができず、カテゴリは数値でなければなりません。これらの条件が満たされていると、トレンドラインオプションの 1 つとして正規分布曲線を指定できます。



正規分布曲線のチャート

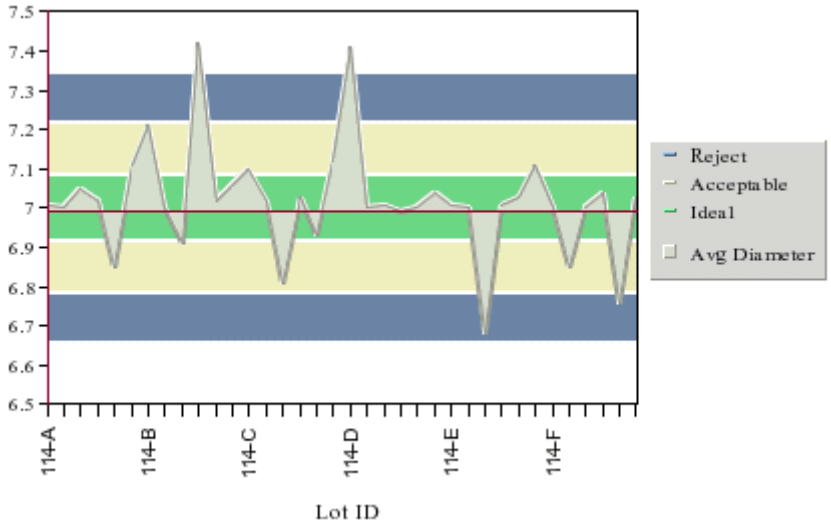
カーブのスケールは通常、値軸のスケールとは異なるため、カーブは 2 軸に表示されます。2 軸のスケールを変更することによって、スケールを変更できます。

5.5.6.3 コントロールエリアの追加

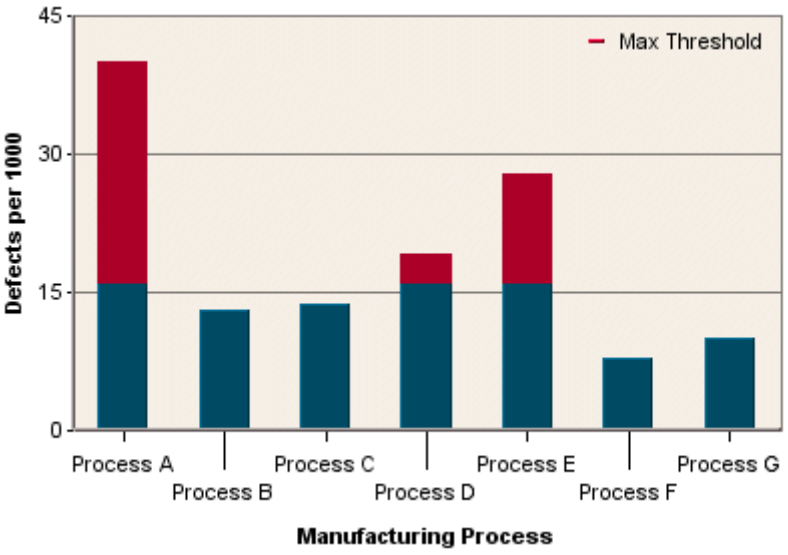
コントロールエリアは、チャートデータと特定の範囲のデータを比較するのに便利です。ほとんどの 2

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次元チャートでは、コントロールエリアは、チャートの値軸および/またはカテゴリ軸上の値の範囲の間のチャートプロット上の塗りつぶし領域として描画されます。次に、データ点がコントロールエリアの上に描画され、指定された範囲内にあるデータ点を簡単に視覚的に参照できます。プロット上に背景領域を描画するのではなく、データ点がコントロールエリアと交差する場合にのみコントロールエリアを表示することもできます。この機能により、特定のしきい値に達するとユーザに明確な視覚的参照が提供されます。



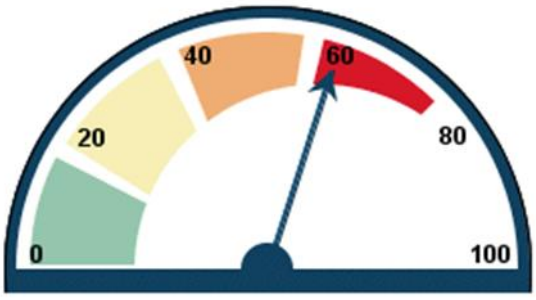
コントロールエリアを持つ２次元領域チャート



データポイント用のコントロールエリアを含む縦棒グラフ

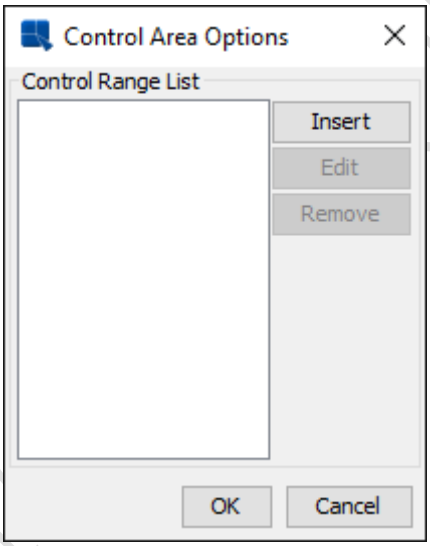
ダイヤルチャートには、コントロールエリアの特殊なインスタンスを使用できます。ダイヤルチャートの場合、コントロールエリアはダイヤルの面に弧として描かれ、ダイヤルハンド（データポイント）が範囲内にあるかどうかを確認できます。コントロールエリアはレーダーと円グラフでは使用できません。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



コントロールエリアを持つダイヤルチャート

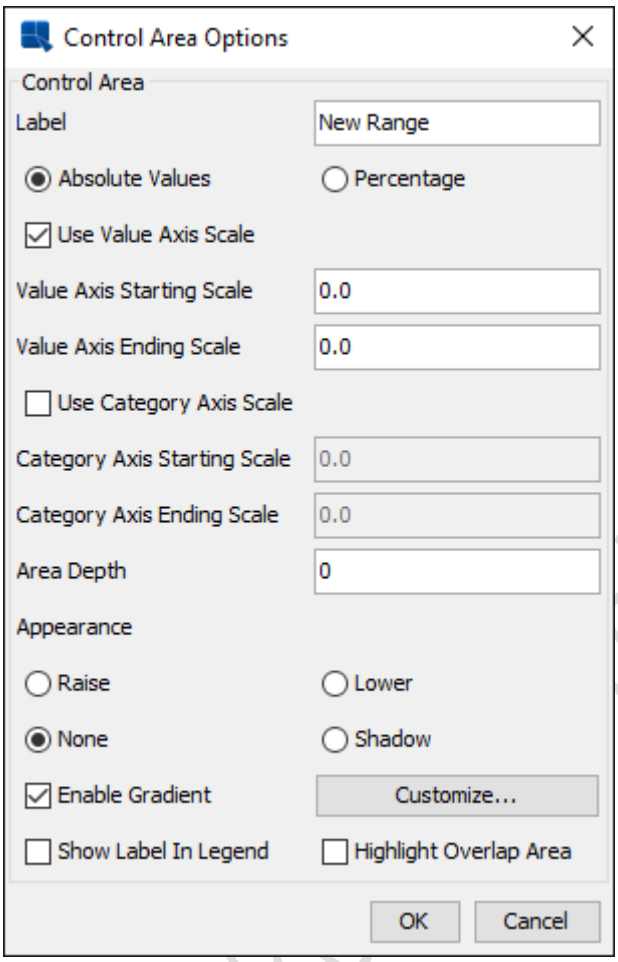
チャートにコントロールエリアを追加するには、**Insert** > **Contorol Area** を選択します。既存のコントロールエリアのリストを表示する次のダイアログが表示され、選択したコントロールエリアを編集したり、選択したコントロールエリアを削除したり、新しいコントロールエリアを作成したりすることができます。



コントロールエリアリスト

既存のコントロールエリアが選択されているときに **Insert** をクリックするか **Edit** をクリックすると、選択したコントロールエリアを設定するダイアログが表示されます。チャートがダイヤルチャートでない場合、次のダイアログが開きます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Contorol Area Configuration ダイアログ

Copyright(C) 2018 Climb.Inc.

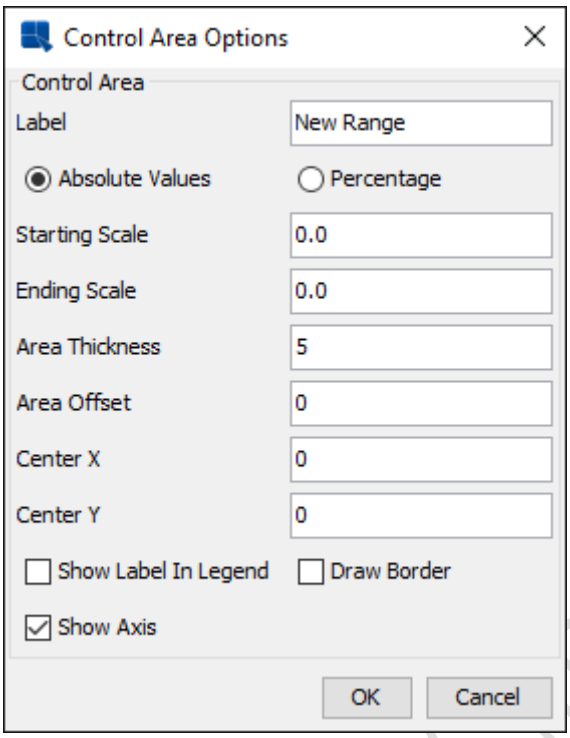
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

コントロールエリアには以下のオプションがあります。

名前	説明
Label	コントロールエリアのラベルを指定できます。
Absolute Values	スケールを絶対値で指定できます。
Percentage	スケールをパーセンテージで指定できます。
Use Value Axis Scale	このオプションでは、コントロールエリアをチャートの値軸上の値で囲むかどうかを指定できます。
Value Axis Starting Scale	値軸上のコントロールエリアの下限です。
Value Axis Ending Scale	値軸上のコントロールエリアの上限です。
Use Category Axis	このオプションでは、コントロールエリアをチャートのカテゴリ軸の値で囲むかどうかを指定できます。
Category Axis Starting Scale	カテゴリ軸のコントロールエリアの下限です。
Category Axis Ending Scale	カテゴリ軸のコントロールエリアの上限です。
Area Depth	このオプションは、任意の外観スタイルの奥行きを指定します。 スタイルが選択されていない場合、深さは効果を持ちません。
Appearance	このオプションでは、コントロール領域に 3D またはシャドウエフェクトを指定できます。領域の深さがゼロとして指定されている場合、外観は有効になりません。
Enable Gradient	コントロールの色グラデーションを有効にします。グラデーションの設定については、フォーマットメニューを参照してください。
Show Label In Legend	チャートの凡例にコントロールエリアのラベルを表示するかどうかを指定します。
Highlight Overlap Area	データポイントがコントロールエリアの境界と重なるコントロールエリアのみを表示します。

チャートがダイヤルチャートの場合は、**Insert** > **Control Area** の順に選択し、**Insert** ボタンまたは **Edit** ボタンをクリックすると、別のダイアログが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



ダイヤルチャートの Control Area ダイアログ

ダイヤルチャートのコントロールエリアダイアログは次のオプションを提供しています。

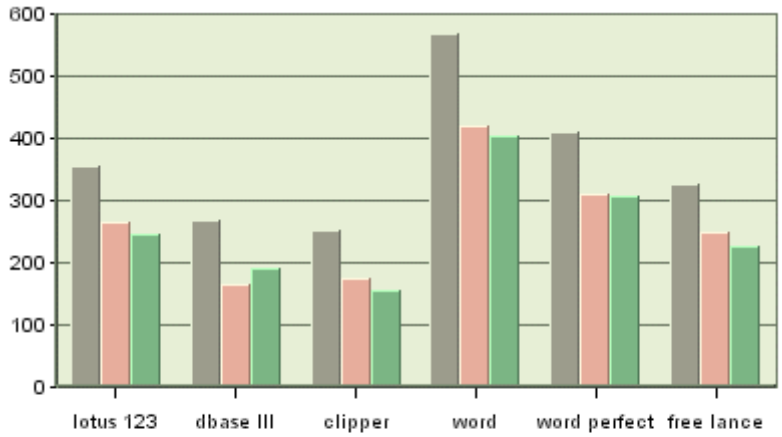
名前	説明
Label	コントロールエリアのラベルを指定します。
Absolute Values	スケールを絶対値で指定します。
Percentage	スケールをパーセンテージで指定します。
Starting Scale	コントロールエリアの値の下限を指定します。
Ending Scale	コントロールエリアの値の上限を指定します。
Area Thickness	コントロールエリアの厚さを指定します。
Area Offset	ダイヤルチャートのエッジからオフセットをピクセル単位で指定します。
Center X	コントロールエリアの中心の X 座標を設定します。0 はダイヤルフェイスと同じ中心点を共有します。ダイヤルの中央からオフセット位置を指定するには、新しい番号（負数または正数のいずれか）のピクセルを指定します。
Center Y	これは、コントロール領域の中心の Y 座標を設定します。以前のオプションと同じように動作します。
Show Label in Legend	チャートの凡例にコントロールエリアのラベルを表示するかどうかを指定します。
Draw Border	コントロールエリアの周りに境界線を描くことができます。
Show Axis	コントロールエリアでカバーされていない残りの領域の軸を表示または非表示にすることができます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

すべてのオプションを指定すると、コントロールエリアがチャートに追加されるか、選択されたコントロールエリアがそれぞれ変更されます。前のダイアログで指定されたプロパティを編集するには、もう一度 **Insert** > **Control Area** を選択し、リストからコントロールエリアを選択します。変更したいコントロールエリアをダブルクリックすることもできます。

5.5.6.4 テーブルの追加

チャートの表示に加えて、チャートに表示されるデータポイントを示すテーブルを表示することもできます。表はチャートプロットの下または右に配置できます。



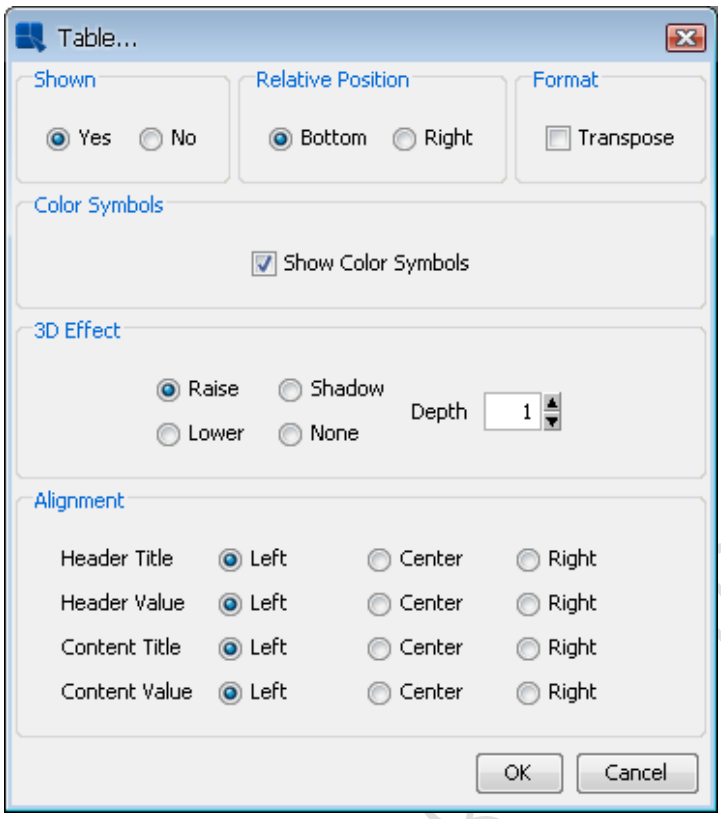
PERIOD_END	lotus 123	dbase III	clipper	word	word perfect	free lance
1987-03-31	355	270	253	569	410	327
1987-02-28	267	165	177	420	311	251
1987-01-31	248	193	155	405	307	229

表付きチャート

チャートを表に追加するか、表のさまざまな表示オプションを変更するには、**Format** > **Table** を選択します。これにより、テーブル表示用のさまざまなオプションをカスタマイズできるダイアログが表示されます。

Copyright(C) Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Format Table ダイアログ

このダイアログでは、表を表示するかどうか、およびチャートをグラフのプロットの右端または右端に与える相対的な位置を指定できます。テーブルとその奥行きに 3D エフェクトを指定することもできます。

Transpose チェックボックスを使用すると、テーブルの列と行を入れ替えることができます。デフォルトでは、カテゴリ要素は列として、データ系列要素は行として描画されます。

Show Color Symbols オプションでは、表のデータ系列のカラーボックスの表示/非表示を切り替えることができます。

quarter\drink	Coffee	Fruit Juice	Soft Drinks	Tea	Water
Q1	181	89	264	114	303
Q2	149	144	212	144	156
Q3	169	70	498	162	275
Q4	195	171	230	144	186

カラーボックス付きチャートテーブル

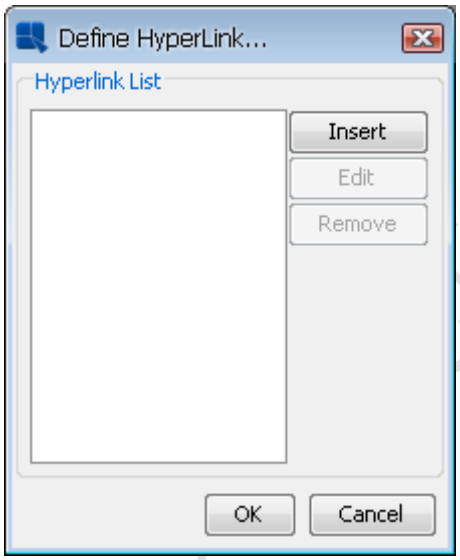
Alignment オプションを使用すると、表のセル内のテキストの水平方向の配置を左、中央、または右のいずれかに指定できます。行ヘッダー、列ヘッダー、および内部テーブルセルに対して配置を設定できます。

チャートキャンバスに十分なスペースがない場合、すべてのデータポイントがテーブルに表示されるわけではありません。表のサイズは、キャンバスのサイズと表のセルのフォントのサイズで調整されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

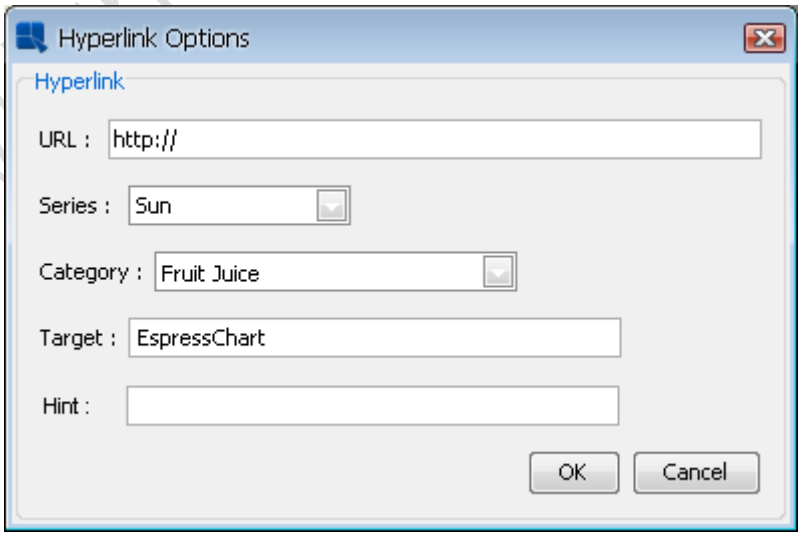
5.5.6.5 ハイパーリンクの追加

ChartDesigner には、チャートの任意のデータポイントにハイパーリンクを追加する機能があります。リンクは、単一のデータポイントまたは複数の要素のいずれかに指定できます。追加されたハイパーリンクは、チャートのデータポイントと凡例のそれぞれのフィールドに適用されます。グラフにハイパーリンクを追加するには、**Insert** > **Link** を選択するか、データポイントを右クリックし、ポップアップメニューから **Insert Link** を選択します。既存のハイパーリンクのリストを表示するダイアログが表示され、新しいハイパーリンクの設定、削除、および/または新規作成ができます。



Inset Link ダイアログ

既存のハイパーリンクが選択されているときに **Insert** をクリックするか、**Edit** をクリックすると、選択したハイパーリンクを設定できるダイアログが表示されます。



Define Link ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

URL フィールドでは、リンクする Web ページを指定できます。

Series および **Category** ドロップダウンメニューでは、ハイパーリンクのデータ系列とカテゴリ要素の要素を選択できます。すべてのデータ系列要素またはすべてのカテゴリ要素にリンクすることもできます。

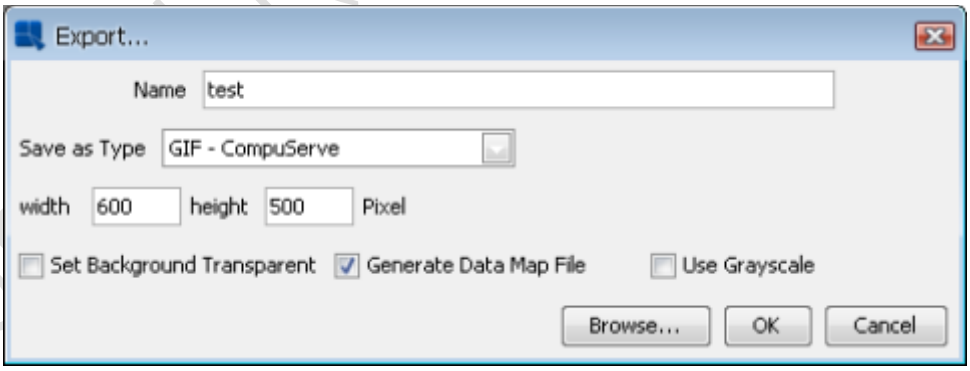
データポイントまたはデータ系列に接続するハイパーリンクを指定するときに、HTML で認識される **Target** パラメータを指定できます。これは、新しい HTML ページを新しいブラウザウィンドウで開くか、同じブラウザウィンドウで開くか、または新しいページが現在のページと同じページの部分を占めるかどうかを判断するために使用できます。

Hint フィールドでは、マウスカーソルをデータポイント上に移動するとポップアップするテキストを入力できます。ハイパーリンクなしでポップアップラベルを作成する場合は、URL フィールドを空白のままにしてヒントのみを指定できます。

5.5.6.5.1 ハイパーリンクの表示

グラフのハイパーリンクを指定すると、グラフが Flash 形式*にエクスポートされた場合にのみアクティブになります。PNG、JPG、GIF などのほとんどの画像フォーマットでは、チャートのエクスポート時にリンクの情報を含むイメージマップファイルが自動的に生成されます。画像ファイルと画像マップを HTML ファイルに挿入して、クリック可能なリンクで画像を表示することができます。

* Flash のエクスポートでは、ユーザーがハイパーリンクをクリックすると、安全でない可能性のある操作があったという Flash の警告メッセージが表示される可能性があります。設定をクリックし、チャートを信頼できる場所のリストに追加することで、この警告をオフにすることができます。



Export ダイアログ

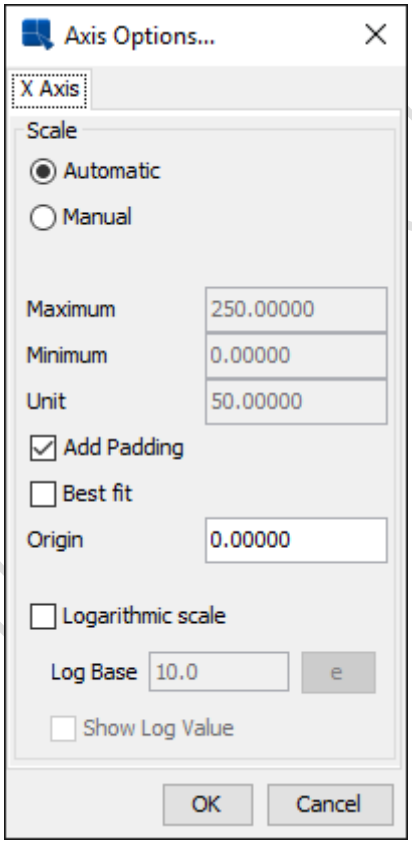
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.7 グラフ軸の書式設定

ChartDesigner は、チャート軸のための豊富なフォーマット機能を提供しています。ユーザーは、軸スケールから軸ラベルの表示方法まですべてカスタマイズできます。

5.5.7.1 軸スケール

デフォルトでは、チャート内の任意の数値軸のスケールが計算され、プロットされるデータに最適なフィッティングが提供されます。表示されているデータが大幅に変更される可能性がある場合、便利な機能です。ただし、手動で軸のスケールを設定することがよくあります。軸スケールを変更するには、**Format** > **Axis Scale** を選択するか、ツールバーの **Axis Scale** ボタン をクリックします。チャート内の任意の値軸のスケールをフォーマットするためのダイアログが表示されます。



Axis Scale ダイアログ

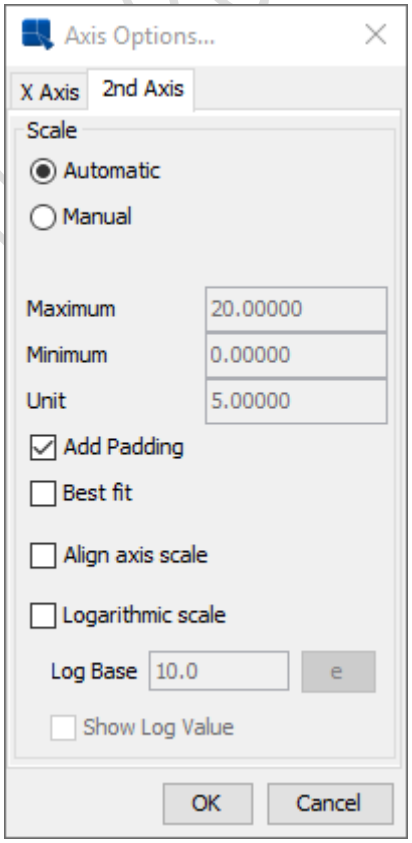
Copyright(C) Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次のオプションが提供されています。

名前	説明
Automatic	軸の自動スケーリングがオンになります。これはデフォルトのオプションです。
Manual	手動スケーリングがオンになります。軸の縮尺を好きに合わせることができます。
Maximum	軸スケールの最大値です。
Unit	軸スケールの最小値です。
Add Padding	連続ラベル間のステップ間隔です。
Best fit	軸の最高値が上昇し、データの最大値とグラフの最上部の間にクッションが作成されます。
Origin	データの最小値と最大値に基づいてチャートの原点が自動的に配置されます。
Logarithmic Scale	X 軸と Y 軸が交差する場所を指定することができます。これは通常 0 に設定されます。
Log Base	ログ値のベースを指定することができます。
Show Log Value	軸ラベルにログ値を表示するかどうかを指定します。

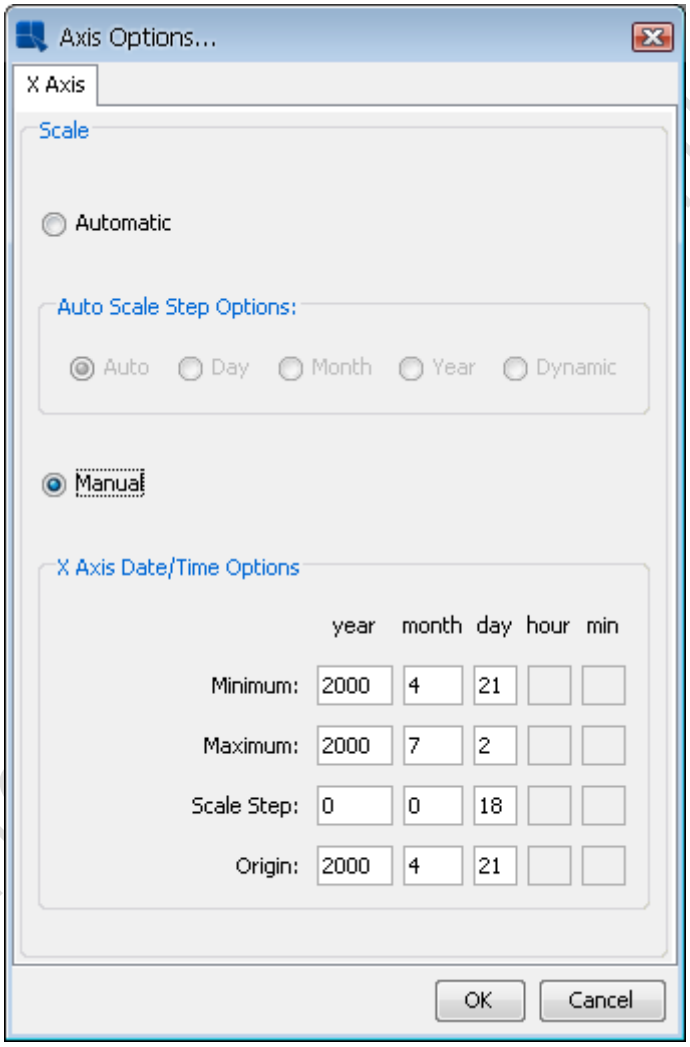
軸スケールダイアログには、チャート内の各軸のタブがあります。2 軸にはユニークなオプションがあり、軸のスケールを揃えることができます。1 軸から 2 軸までのすべてのオプションが適用され、両方の軸にまったく同じスケールが与えられます。



2 軸の Axis Scale ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ガントチャートでは、軸スケールのダイアログが異なります。自動オプションを選択してスケールを自動的に設定するか、手動オプションを選択してスケールを手動で設定することもできます。自動軸スケールには、自動、曜日、月、年、および動的の、いくつかの自動スケールステップオプションがあります。自動オプションは常に最適なものを見つけます。動的オプションも最適ですが、自動オプションとは異なり、標準的なステップ間隔のみを使用します（たとえば、1 か月、1 年など...）。手動軸スケールを選択すると、Maximum、Minimum、Unit、Origin がそれぞれ Maximum Date、Minimum Date、Scale Step、Origin Date に置き換えられ、これらの新しい設定には Date/Time（yyyy、MM、dd、 hh、mm）が用いられます。

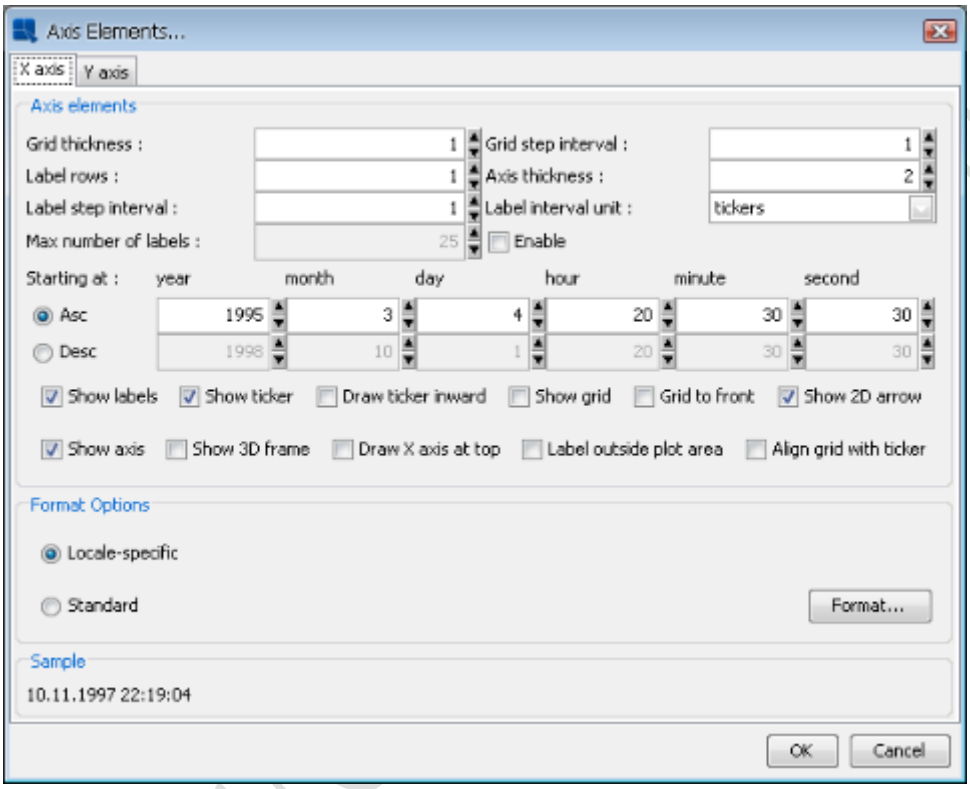


ガントチャートの Axis Scale ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.7.2 軸要素

軸の外観プロパティと軸ラベルは、**Axis elements** ダイアログによって制御されます。軸要素ダイアログを呼び出すには、**Format** > **Axis Elements** を選択するか、ツールバーの **Format Value Elements ボタン**をクリックします。これにより、次のダイアログが表示され、すべてのチャート軸の要素をカスタマイズできます。また、このダイアログからダイヤルと円グラフのラベルの外観をカスタマイズすることもできます。



Axis Elements ダイアログ

チャート内の各軸のタブがこのダイアログに表示されます。このダイアログでは、次のオプションを実行できます。一部のオプションは、特定のチャートタイプ、特定のデータタイプ、および特定の軸でのみ使用可能であることに注意してください。

Copyright(C) Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

名前	説明
Grid thickness	軸に沿ったグリッド線の太さを指定することができます。
Grid step interval	軸に沿ったグリッド線のグリッドステップ間隔を設定できます。
Label rows	ラベルを交互に表示することができます。重複を防ぐことができます。このオプションは、X 軸でのみ使用できます。
Axis thickness	軸の太さをピクセル単位で設定できます。この設定はチャートのすべての軸に適用されることに注意してください。
Label step interval	データのラベルステップ間隔を設定できます。たとえば、これを 2 に設定すると、チャート内の他のすべてのデータポイントのラベルが描画されます。
Label interval unit	時間ベースのデータ（日付、時刻、タイムスタンプ）を並べ替えて表示するときに使用する単位を選択できます。ティッカーを選択すると、Chart Designer が読み取ったデータが使用されます。時間ベースのデータに対して動的を選択することもできます。これはデータポイントの数と範囲に応じて適切なスケールを選択します。
Max number of labels	軸に表示されるラベルとティッカーの最大数を選択できます。ラベルの数が最大数を超えると、ラベルステップが再計算されます。
Ascending/Descending	時間ベースのデータを並べ替えることができます。データを昇順または降順に並べ替えるだけでなく、データの開始（または終了）ポイントを指定することもできます。
Show labels	各軸のラベルを削除または表示できます。
Show ticker	軸ティッカーを表示または削除できます。
Draw ticker inward	軸のテロップを外側（デフォルト）ではなくプロット領域の内側に描画します。
Show sub/tickers	軸のスケールが 10 のログベースで対数に設定されている場合にのみ、値軸で使用できます。値軸上の点間に間隔ティッカー（非一様）が描画されます。
Show grid	各軸のグリッドを削除または表示できます。
Grid to front	グラフのデータ要素の上にグリッド線を描画できます。 デフォルトでは、データポイントはグリッドの上に描画されます。
Show 2D arrow	軸の終わりにある矢印を削除または表示できます。 すべてのチャート軸に適用され、2 次元チャートでのみ使用可能であることに注意してください。
Show axis	軸（2 次元チャートの場合）または壁面（3 次元チャートの場合）を削除または表示できます。
Show 3D frame	チャートの周りのフレームを削除または表示できます。 すべてのチャート軸に適用され、3 次元チャートでのみ使用可能であることに注意してください。
Label outside plot area	軸がどこにあるかにかかわらず、プロット領域の外側に配置されるラベル

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

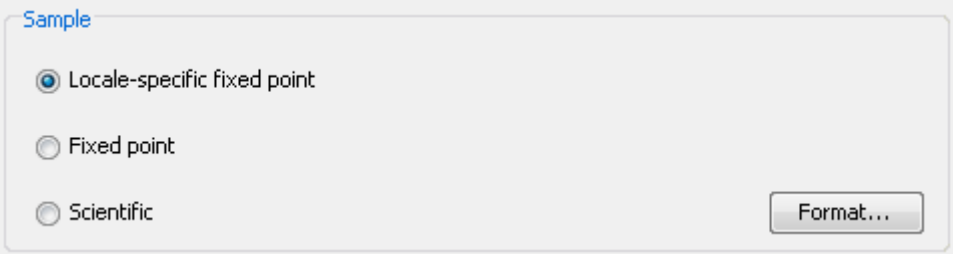
	を設定します。 正の値と負の値の両方を使用してデータをプロットする場合、カテゴリ軸ラベルに役立ちます。
Align grid with ticker	グリッドラインをティッカーの間に配置するのではなく、ティッカーと整列させます。 ティッカーと対応するグリッド線が同じ線に沿って配置されます。 列タイプチャートのカテゴリ軸にのみ適用されます。
Swap Y-axis position	プライマリとセカンダリの値の軸を入れ替えます。2 軸タブの下にのみ表示されます。
Draw X-axis at top	X 軸がデフォルトの下部位置の代わりにチャートの上部に配置されます。2 次元の列、棒、散布、高低、HLCO、バブル、ガントチャートで使用できます。

5.5.7.2.1 軸ラベルの書式設定

軸要素ダイアログでは、軸上にプロットされているデータの種類に応じて、軸ラベルの外観を書式設定することもできます。ダイアログのフォーマットオプション部分には、ラベルの書式設定ダイアログが含まれています。

数値データの書式設定

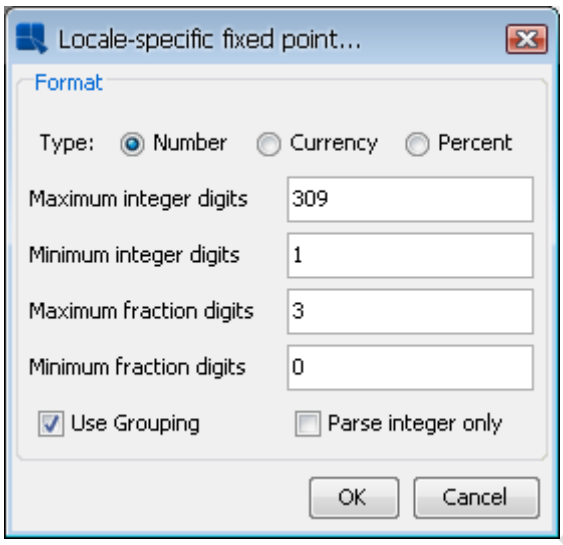
数値データの場合、表示フォーマットにはロケール固有の固定小数点、固定小数点、および科学という 3 つの主要なオプションがあります。**Format** ボタンをクリックすると、追加のオプションが表示されます。



数値データ形式のオプション

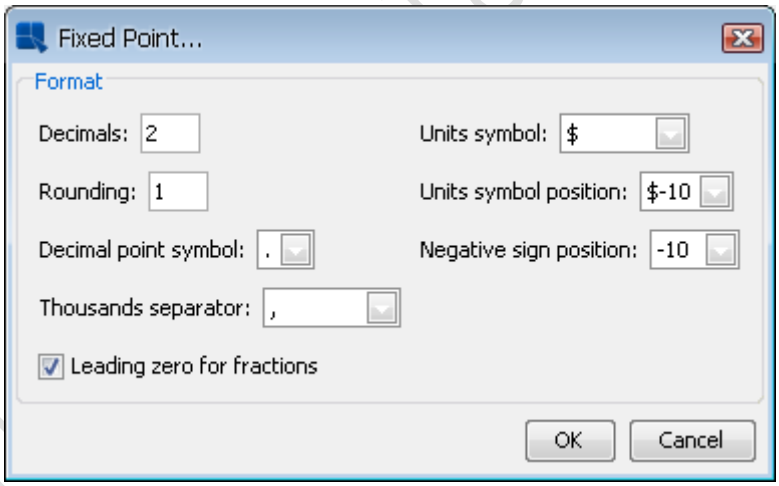
- **Locale-Specific Fixed Point：**
  これにより、表示されているロケールに応じてデータのフォーマットが変更されます。 このオプションの追加の書式設定では、データを数値、通貨、またはパーセンテージとして表示するかどうかを指定できます。また、整数桁数と小数点桁数の最大値と最小値を設定できます。 他の表示属性はロケールによって異なります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



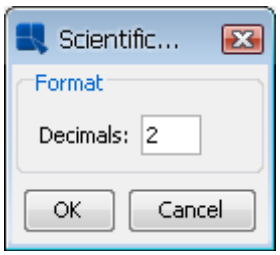
Locale-Specific Formatting オプション

- **Fixed Point：**
  これにより、ロケールに関係なくデータフォーマットが一貫して保持されます。このオプションの追加の書式設定では、小数点以下の桁数、小数点以下の桁数、単位記号、負の記号の位置、小数点と桁区切り記号を設定し、小数点の先頭桁を有効にすることができます。



Fixed Point Formatting オプション

- **Scientific:**
  これにより、科学的表記法でデータが表示されます。このオプションの追加の書式設定では、小数点以下の桁数を設定できます。

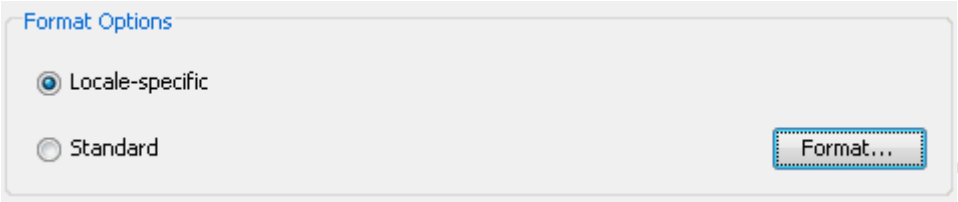


Scientific Formatting オプション

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

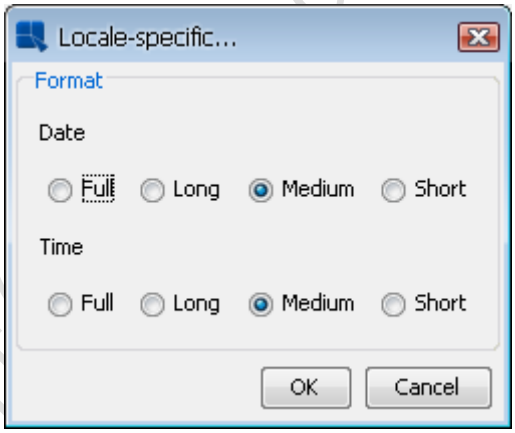
日付/時刻データの書式設定

日付/時刻データの場合、表示フォーマットの主な 2 つのオプションがあります：ロケール固有と標準。Format ボタンをクリックすると、追加のオプションが表示されます。 使用可能なオプションは、データの性質によって異なります。日付、時刻、およびタイムスタンプのデータはそれぞれ、日付、時刻、および日付と時刻のオプションを表示します。



Date/Time Data Format オプション

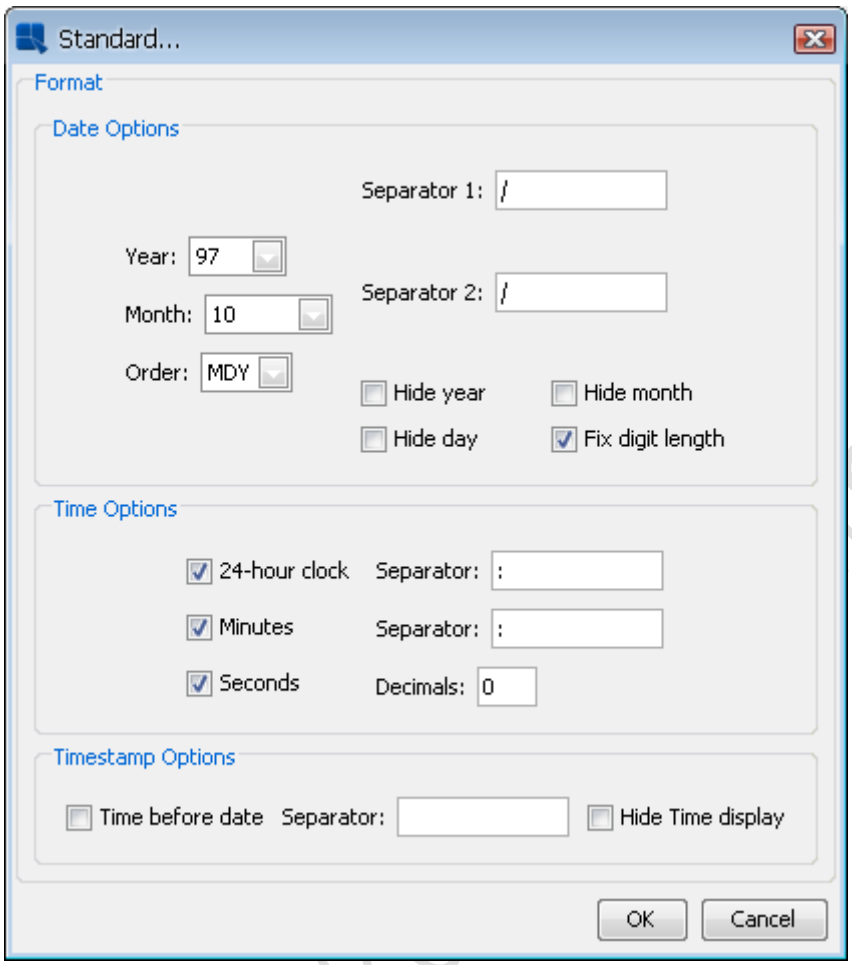
- **Locale-Specific：**
  これにより、表示されているロケールに応じてデータのフォーマットが変更されます。 このオプションの追加の書式設定では、日付と時刻情報の完全、長、中、または短い表記を選択できます。他の表示属性はロケールによって異なります。



Locale-Specific Formatting オプション

- **Standard：**
  これにより、ロケールに関係なくデータフォーマットが一貫して保持されます。 このオプションの追加の書式設定では、年月表示、および月、日、年の情報の表示順序を選択できます。セパレータとして使用する文字を選択することもできます。時間オプションを使用すると、時間、分、および秒を表示し、それらの間にセパレータを選択することもできます。タイムスタンプデータの場合は、日付の前後の時刻と、それらの間で使用される区切り文字を表示するように選択できます。

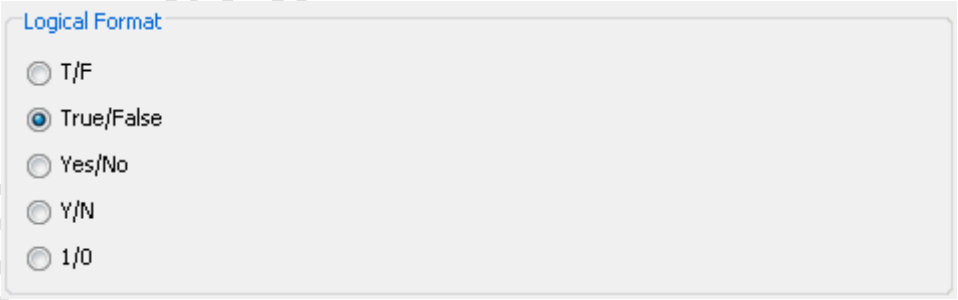
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



Standard Formatting オプション

論理データのフォーマット

T/F、True/False、Yes/No、Y/N、および 1/0 の 5 つのオプションがあります。



Logical Data Formatting オプション

データの書式を変更すると、軸要素ダイアログで **OK** ボタンをクリックすると、その変更が有効になります。文字列データの追加の書式設定オプションはありません。

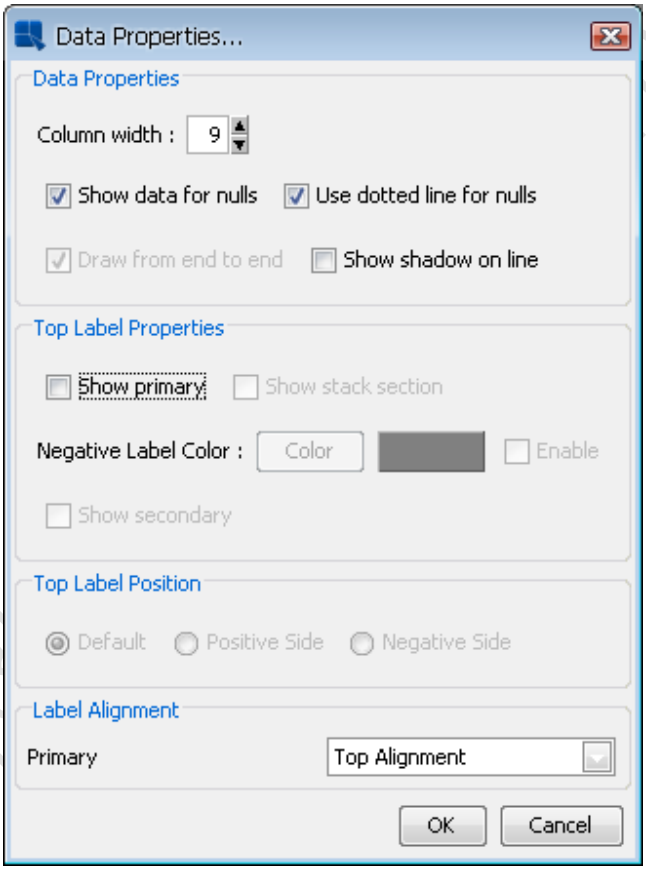
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.8 プロット/データ要素の書式設定

ChartDesigner には、グラフ上にデータ点が描かれ注釈される方法やチャートプロット自体をカスタマイズして設定するためのさまざまな方法が用意されています。

5.5.8.1 データプロパティ

多くのデータ表示オプションは、データプロパティダイアログで制御されます。このダイアログでは、バー/カラムのサイズを制御したり、Null 値の表示オプションを設定したり、データラベルのオプションを指定することができます。データプロパティダイアログを呼び出すには、Format > Data Properties を選択するか、ツールバーの **Data Properties** ボタンをクリックします。これにより、次のダイアログが表示されます。



Data Properties ダイアログ

この Data Properties ダイアログには、次のオプションがあります。

Column width：

チャート内の連続するバーの間のギャップに対するバー/カラムの幅の比率を指定します。各単位は、データポイント間のスペースの 1/10 を表します。したがって、**9** を入力するとデータポイント間のスペースの 10%が空白になり、**10** はバー/カラム間のスペースをすべて無くします。このオプションは、2D 横棒、縦棒、スタックバ横棒、スタック縦棒、HighLow、HLCO、ガントチャートにのみ関連します。3 次元グラフで列の太さを制御するには、ナビゲーションパネルのシェイプスライダーの太さを使用します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Show data for nulls：
null データが存在するときに行を接続します。たとえば、3 つのポイントがあり、ポイント 1 の値が 4、ポイント 2 が null、ポイント 3 が 6 の場合、ラインは 4 から 6 まで描画されます。このオプションは、折れ線グラフと他の 2 つの線を含む次元グラフのみ使用可能です。その他のすべてのグラフタイプは、nul データをプロットしません。

Use dotted lines for nulls：
フルラインを点線で置き換えることができます。null の表示データオプションと同様に、このプロパティはラインでのみ使用できます。

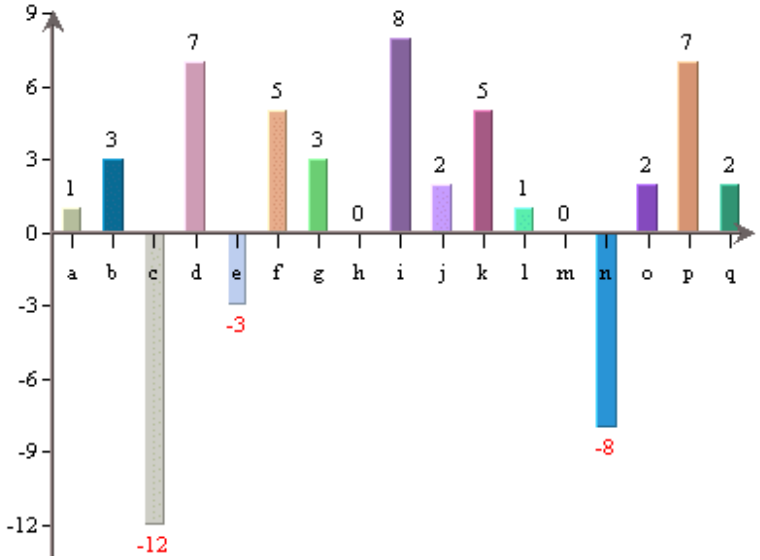
Draw from end to end：
チャート上の最初と最後のデータポイントにオフセットするのではなく、プロットエリア全体に 2 次元の線グラフと面グラフを描画できます。

Show shadow on line：
2 次元線で陰影を使用できます。ただし、線は 1 ピクセルよりも厚くなければなりません。

Show primary：
チャートのプライマリ値のデータのトップラベルを表示します。

Show stack section：
スタックバー、スタックカラム、およびスタックエリアチャートの各スタックセクションに個別のラベルを表示します。

Negative Label Color：
原点の値よりも小さい値を持つ最上位のラベルが、機能を有効にした後に **Color** ボタンを使用して選択できる別の色で表示されます。



色付きネガティブトップラベルのグラフ

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Show secondary：
グラフ内のセカンダリ値のデータのトップラベルが表示されます。

Top label position：
データのトップラベルを描画する場所を指定できます。デフォルトでは、データ点が正の場合はデータ点の上に、負の場合はデータ点の下に描画されます。他のオプションでは、ラベルを正側または負側のいずれかに描画できます。

Label Alignment：
データトップラベルの配置を設定できます。データ点の上、下、または中央に描画することができます。また、上または下のデータポイント内にラベルを描画するように選択することもできます。追加のオプションスタックチャートでは、スタックセクションラベルのアラインメントを設定できます。

5.5.8.2 日付/時間ベースズーム

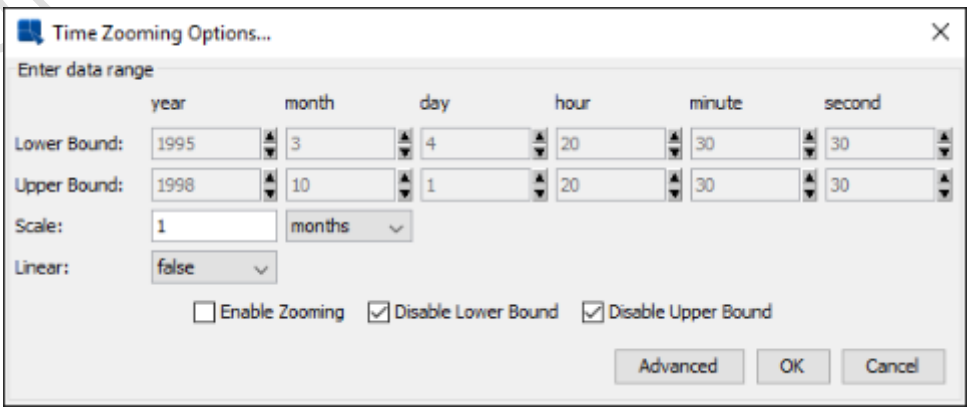
カテゴリ軸上の日付または時刻データを表示するチャートの場合、ChartDesigner はユーザーが日時ベースのズームを実行できるユニークな機能を提供します。この機能を使用すると、カテゴリ要素をユーザー定義の間隔にグループ化し、各グループのポイントを集約することができます。また、結果の上限と下限を指定してデータをフィルタリングすることもできます。

たとえば、あなたのデータに過去 2 年間の日々の販売数量が含まれているとします。ズームを使用すると、データを集計して月、四半期、または年の平均音量を確認できます。上限と下限を使用すると、特定の四半期内の週単位の販売数量を調べることができます。

拡大縮小は、散布図、サーフェス、ボックス、ダイヤル、極座標、レーダー、バブル、ガント以外のすべてのチャートタイプで利用できます。

5.5.8.2.1 ズームの追加

カテゴリ軸に日付、時刻、またはタイムスタンプのデータを含む新しいチャートを作成するときは、Format > Time Zooming Options を選択してズームオプションを指定できます。



Zoom Options ダイアログ

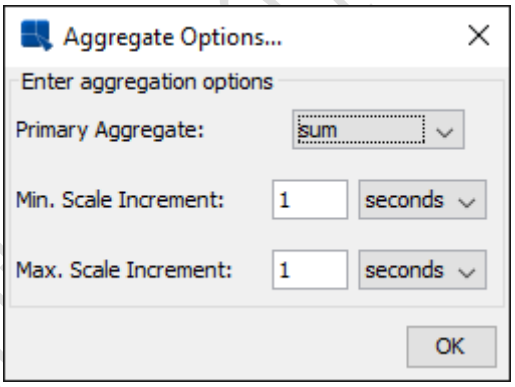
Copyright(C) 2018 Climb.Inc. All Rights Reserved.

このダイアログでは、データの下限と上限、およびデータをグループ化する間隔を指定できます。ここで指定する縮尺は、集約オプションダイアログで指定した最大縮尺と最小縮尺の範囲内でなければなりません。

このダイアログでは、チャートの線形尺度を保持することもできます。**Linear** オプションを true に設定すると、特定のグループに関連付けられたデータがなくても、チャートにはグループ化された間隔のポイントが常に表示されます。たとえば、4 月、5 月、6 月の 3 か月間の販売数量を測定しているとします。入力データに 5 月のレコードがなく、**Linear** オプションを true に設定すると、値が 0 の点が 5 月に描画されます。**Linear** オプションを false に設定すると、4 月のデータポイントの直後に 6 月が続きます。

ダイアログの下部にあるチェックボックスを使用することで、ズームや上限と下限の制限を無効/有効にすることができます。

ズームを有効にすると（**Enable Zooming** チェックボックスをオンにした場合）、ダイアログボックスの **Aggregate Options** が表示され、グループ化されたデータポイントの集約オプションを指定するよう求められます。



Aggregation Options ダイアログ

このダイアログでは、データをズームするときに使用できる最大および最小スケールインクリメントと同様に、Primary Aggregation を指定できます。必要なオプションを指定したら、**OK** ボタンをクリックしてズームオプションに戻ります。

すべてのオプションの指定が完了したら、**OK** ボタンをクリックすると、グラフにズームが適用されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

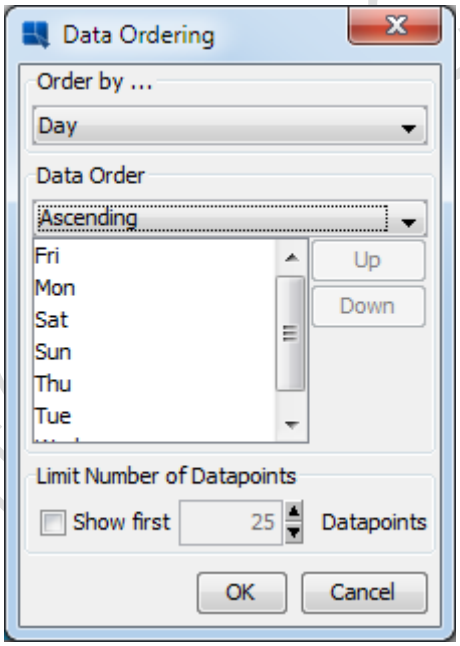
5.5.8.2.2 チャートビューアをズームイン

Chart Viewer を使用してチャートを配置する場合、エンドユーザーは動的なズームを実行できます。チャートビューアで時系列ズームを実行するには、チャート上のポイントを **Ctrl +クリック**してチャート内の別のポイントにドラッグします。これにより、マウスを使用して選択した下限および上限に基づいて自動的にズームインされます。集計は、設計時に設定されたオプションに従って実行されます。**Ctrl + 右クリック**でズームを元に戻すことができます。

スケール間隔は、データと選択された範囲に応じて自動的に選択されます（最小 2 データポイントを表示できる限り）。スケール間隔は、**Alt + Z** を押して、チャートビューアで変更することもできます。 これにより、ユーザーがズーム設定を変更できるダイアログが表示されます。

5.5.8.3 データの順序付け

ChartDesigner では、カテゴリとシリーズ要素の順序を変更できます。順序を変更するには、Data > Ordering を選択するか、ツールバーの **Change Data Ordering** ボタンをクリックします。 これにより、次のダイアログが表示されます。



Data Ordering ダイアログ

カテゴリ要素、シリーズ要素、および **VALUES** とマークされたオプションを含む Order By リストがあります。カテゴリおよびシリーズ要素には、次のオプションがあります。

DataSource Order：

順序をオフにします。カテゴリ/シリーズの注文はデータソースのみに依存し、EspressChart では変更されません。

Ascending：

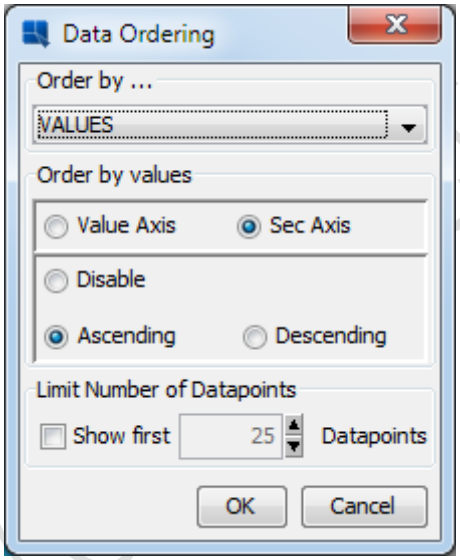
カテゴリ/シリーズ要素を昇順に並べ替えます。たとえば、カテゴリ要素が文字列の場合、アルファベット順に並べ替えられます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Descending：
カテゴリ/シリーズ要素を降順で並べ替えます。

Customize：
このオプションでは、カテゴリ/シリーズの順序をカスタマイズできます。順序をカスタマイズするには、カテゴリ/シリーズ項目のリストから項目を選択し、**Up** または **Down** ボタンをクリックしてリスト内を上下に移動します（このボタンは **Customize** オプションを選択するまで非アクティブです）。

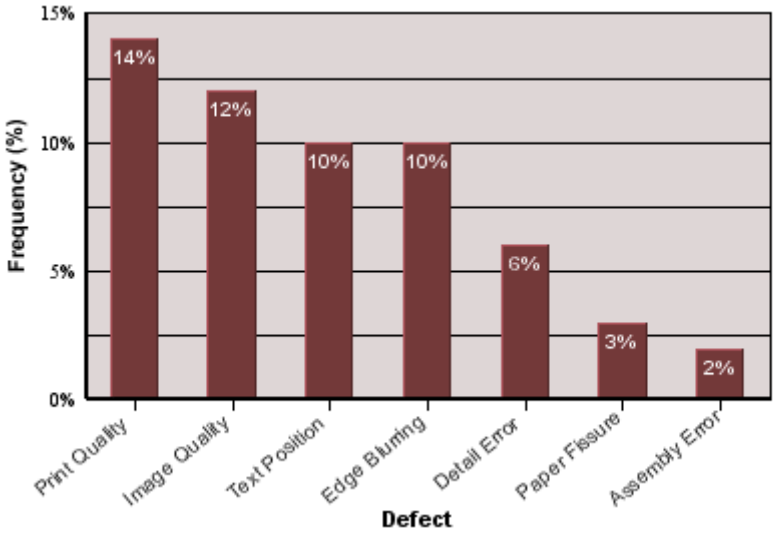
カテゴリ要素は、対応する値に基づいてソートすることもできます。これを行うには、データ順序付けダイアログで **VALUES** オプションを選択します。



Value Data Ordering ダイアログ

VALUES オプションを選択すると、ダイアログ全体が変更されます。このダイアログでは、値またはセカンダリ値の軸に対応する値に基づいてカテゴリ要素をソートするように指定できます。 昇順または降順でソートするかどうかを指定することもできます。このタイプの順序付けは**パレート**図と呼ばれ、プロセス制御アプリケーションでよく使用されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



パレート図

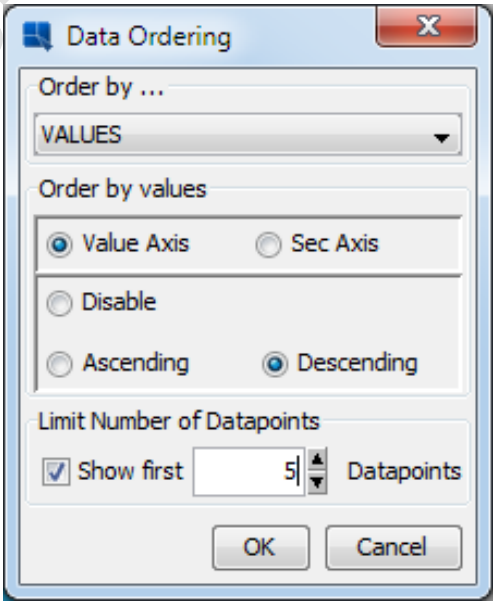
チャートが更新された場合、および/またはデータが変更された場合は、ソートセットが再度適用されます。

5.5.8.3.1 トップ/ボトム N チャート

場合によっては、いくつかの最高値または最低値のみをプロットしたい場合もあります。 これを行うには、**Top/Bottom N** 機能を使用します。

この機能を有効にするには、**Data** > **Ordering** を選択するか、ツールバーの **Change Data Ordering** ボタンをクリックします。

グラフにデータ系列がない場合、**Ordering** ダイアログには **LimitPoint Of DataPoints** オプションが表示されます。



Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

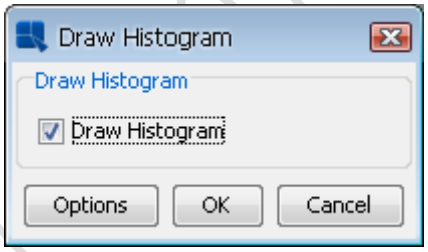
このオプションは、カテゴリまたは値、または昇順または降順の値でグラフをソートする場合にのみ有効にすることができます。このようなグラフがある場合は、**Show first** オプションを選択してこの機能を有効にすることができます。次に、グラフに表示される項目の最大数を指定できます。データソースがこのオプションで指定した以上のアイテムを返した場合、余分なアイテムは、存在しないかのようにチャートに表示されません。

5.5.8.4 ヒストグラム

ヒストグラムは、イベントが発生する頻度や、データのセットが特定の範囲に収まる時期と方法を追跡できる便利な分析ツールです。ChartDesigner では、チャートのカテゴリ要素に基づいてヒストグラムをプロットすることができます。時間ベースのデータ（日付、時刻、またはタイムスタンプ）を除くすべてのカテゴリのデータタイプのヒストグラムをプロットすることができます。

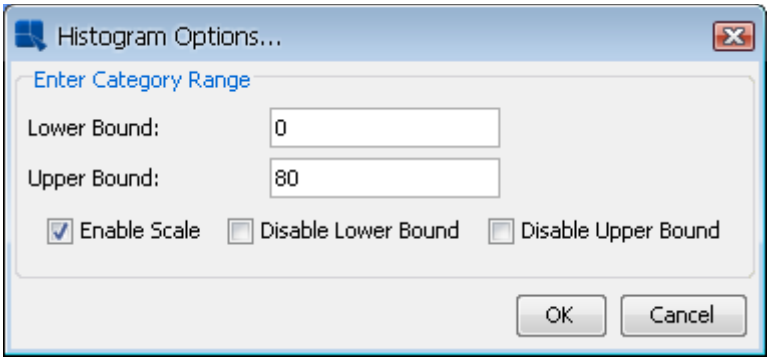
ヒストグラムは、各カテゴリ要素のデータ点またはインスタンスを数えることによって計算されます。数値カテゴリの場合は、さらに上限と下限を指定できます。また、頻度カウントの範囲を作成するためのスケールも指定できます。

ヒストグラムを作成するには、**Format** > **Histogram Options** を選択します。ヒストグラムプロットを選択できるダイアログが表示されます。



Select Histogram ダイアログ

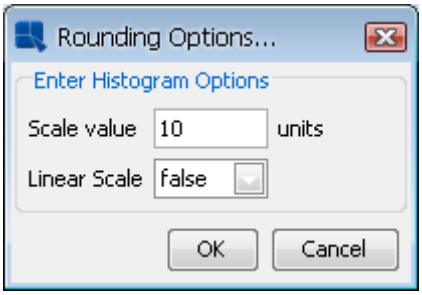
Draw Histogram オプションを選択すると、別のダイアログが表示され、ヒストグラムプロットのオプションを指定できます。このダイアログから、プロットされるデータの下限または上限を選択できます。制限付きの制限を設定すると、ヒストグラムは上限と下限で指定された範囲外のデータをカウントしません。一般的に、数値カテゴリ要素には制限付きの制限を設けることが理にかなっています。



Histogram Options ダイアログ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

数値カテゴリ要素の場合は、頻度数を計算する値の範囲またはグループを提供するために位取りを指定することもできます。スケールを指定するには、**Enable Scale** オプションを選択します。これにより、スケールを指定できる新しいダイアログが表示されます。

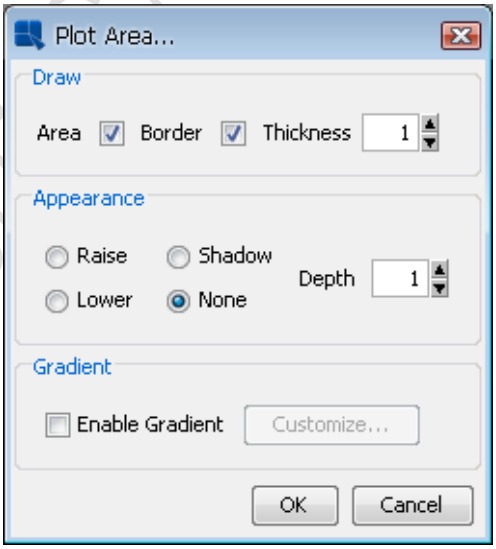


Histogram Scale オプション

ここでは、各スケールステップの単位数を指定できます。リニアスケールを保存するかどうかを指定することもできます。対応するデータポイントがない場合でも、カテゴリ軸に範囲を描画します。例えば、頻度カウントが 20 と 30 との間でゼロである場合、"30"要素は依然としてカテゴリ軸に 0 のカウントで現れる。ただし、リニアスケールがオフの場合、"30"要素はグラフに描画されません。

5.5.8.5 プロットエリアの書式設定

プロット領域は、2 次元グラフ用にデータポイントが描画される平面です。Format > Plot Area を選択すると、プロット領域の外観をカスタマイズできます。現在のチャートが 2 次元チャートであると仮定すると、次のダイアログが表示されます。



Plot Area ダイアログ

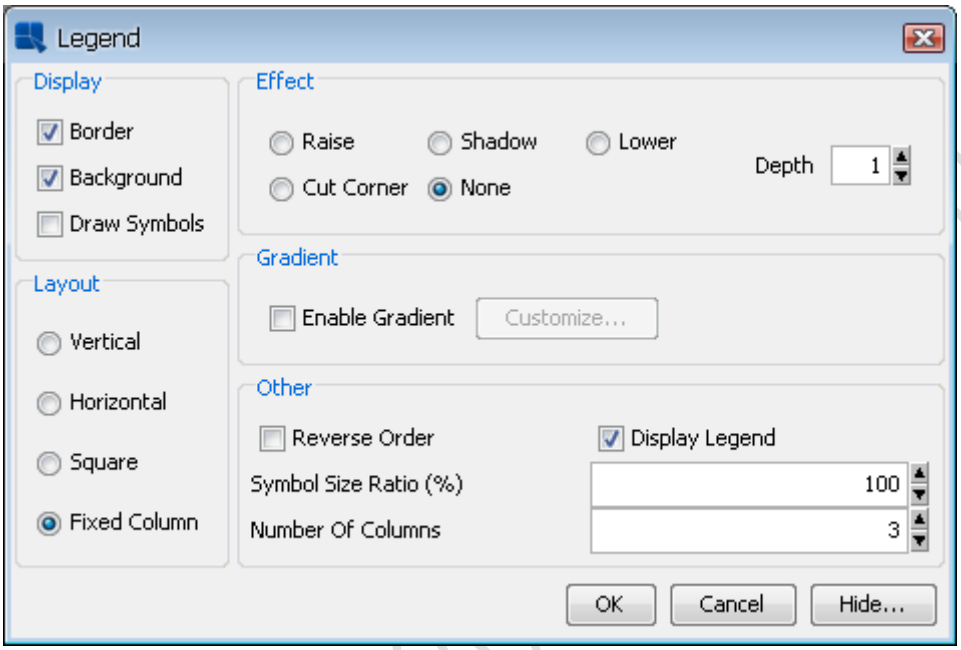
このダイアログでは、プロット領域の周りに境界線を描くことも、背景色を塗りつぶすこともできます。エリアを塗りつぶした場合は、上げたり下げたり影をかけるような特定の 3D 効果を指定することもできます。

このダイアログでは、プロット領域のグラデーションの背景を設定することもできます。グラデーションの設定は、フォーマットメニューで説明している **Rendering オプション**と同様です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.8.6 フォーマットチャートの凡例

Format > Legend を選択するか、またはツールバーの **Format Legend** ボタンをクリックするか、または凡例ポップアップメニュー(凡例で右クリックしたときのポップアップ)から **Legend プロパティ** を選択することによって、チャート凡例の表示を制御および変更できます。これにより、次のダイアログが表示され、凡例のプロパティをカスタマイズできます。



Format Legend ダイアログ

このダイアログには次のオプションがあります：

Display：
これらのオプションを使用すると、凡例の境界線と背景をオンまたはオフにすることができます。これにより、凡例に線またはブロックの代わりにポイントシンボルを表示することもできます。

Effect：
3D 効果を凡例に追加することができます。 上げる、下げる、または影を描くことができます。3D エフェクトに加えて、カットコーナーで凡例を表示することもできます。

Layout：
凡例を垂直、水平、正方形、または固定の列レイアウトから変更することができます。

Gradient：
凡例の背景のグラデーションを設定できます。グラデーションの設定は、フォーマットメニューで説明している **Rendering オプション**と同様です。

Other：
これにより、凡例を表示するかどうかを選択したり、凡例を逆順に描画したりすることができます。 凡

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

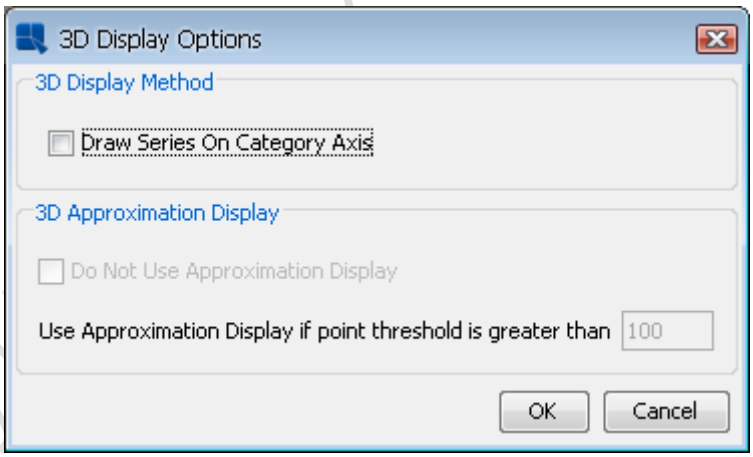
例の固定列数を**列数**フィールドに設定できます。このフィールドは、**レイアウト**セクションで**固定列**レイアウトオプションを選択した場合にのみアクティブになります。また、凡例のシンボルのサイズを変更することもできます。

さらに、凡例から特定のカテゴリ/シリーズ要素を削除するには、**Hide** ボタンをクリックします。 これにより、凡例項目のリストが表示され、非表示にする要素を選択できます。

5.5.8.7 3D 表示オプション

EspressChart は、3D チャートを 3D でレンダリングし、光源の変更、パン、ズーム、回転を可能にします。しかし、3D レンダリングは非常にメモリと CPU を消費します。チャートにデータポイント（3D 散布図やサーフェスチャートなど）が多い場合は、グラフを生成するときにメモリが不足する可能性があります。この問題を解決するために、レンダリング近似機能が提供されています。このアルゴリズムを使用すると、図は完全にはレンダリングされませんが、多くの点を表示する必要がある場合にある程度の許容範囲で効率的に表示されます。

デフォルトでは、しきい値の点の数は 100 です。100 で近似がオンになります。つまり、3D チャートに 100 を超えるデータポイントがある場合、近似値が使用されます。この機能をオフにするか、**Format > 3D Display Options** を選択してしきい値を変更することができます。これにより、以下のダイアログが表示されます。



3D Display Options ダイアログ

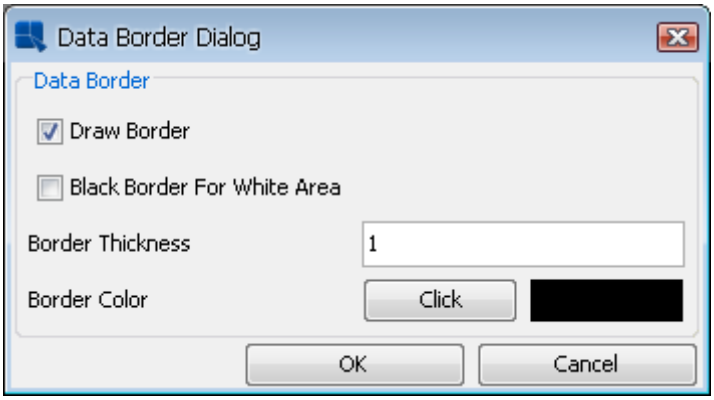
3D 近似の 2 つのオプションは、近似をオンまたはオフにし、しきい値を設定することを可能にします。このダイアログのもう 1 つのオプションでは、シリーズをインラインで描画することができます（同じオプションはナビゲーションパネルにあります）。

5.5.8.8 境界線のオプション設定

縦棒グラフ、横棒グラフ、スタック縦棒グラフ、スタック横棒グラフ、および HLCO チャートの場合、EspressChart を使用してカラムの周りに境界線を設定できます。枠線オプションを設定するには、**Format > Data Border** を選択します。これにより、境界線のオプションを設定できるダイアログが表

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

示されます。



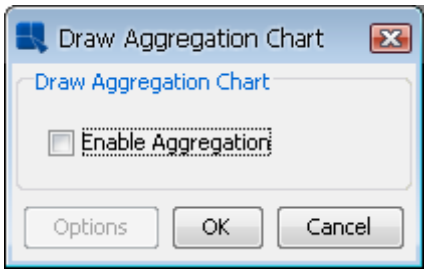
Data Border ダイアログ

Draw Border : データの境界線をオンまたはオフにすることができます。
Black Border For White Area : チャート内の白い部分の黒い枠線を設定できます。**Draw Border** がチェックされておらず、チャートの白い部分の周りにのみ表示される場合に限って、境界線は黒であることに注意してください。
Border Thickness および **Border Color** では、境界線の太さと境界線の色を設定できます。**Click** をクリックすると、新しい色を選択または入力するための新しいダイアログが表示されます。

5.5.8.9 集約オプション

ChartDesigner では、特定のカテゴリ（シリーズやスタックが存在する場合はそのシリーズやスタック）に複数のデータポイントが関連付けられている場合、データを集約できます。これにより、単一のデータポイント（多数のデータポイント）だけではなく、データを幅広く見ることができます。

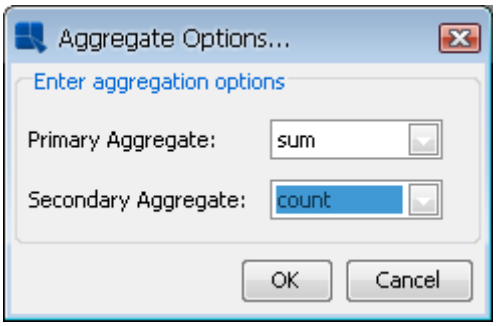
データを集約するには、**Format** > **Aggregation** を選択します。集約を有効にするためのダイアログが表示されます。



Select Aggregation ダイアログ

Enable Aggregation を選択すると、適用する集約のタイプを決定する 2 番目のダイアログボックスが表示されます。集約の最小、最大、平均、合計、カウント、最初、最後、合計四角、分散、標準偏差、およびカウントディストリクトから選択できます。セカンダリ軸が存在する場合、セカンダリ集約（セカンダリ軸にマップされた列に適用された集約）だけでなく、プライマリ集約（プライマリ軸にマップされた列に適用された集約）を指定できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



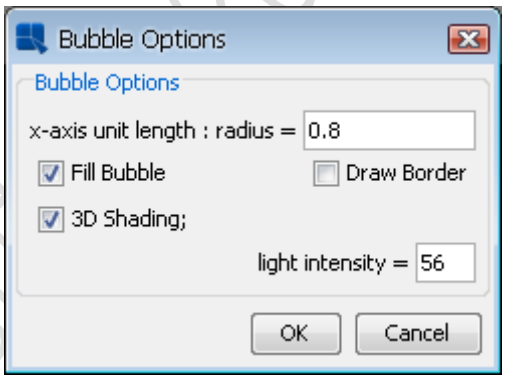
Aggregate Options ダイアログ

5.5.9 グラフ固有のオプション

特定のチャートタイプには、固有の多数の書式設定オプションがあります。これらのオプションは、**Format** > **Chart Options** を選択するか、ツールバーの **Chart Option** ボタン をクリックして変更できます。これにより、現在のチャートのタイプによって異なるダイアログが表示されます。一部のグラフタイプには追加オプションはありません。

5.5.9.1 バブルチャート

バブルチャートの場合は、次のダイアログが表示されます。



Bubble Options ダイアログ

バブルチャートでは以下のオプションが利用可能です。

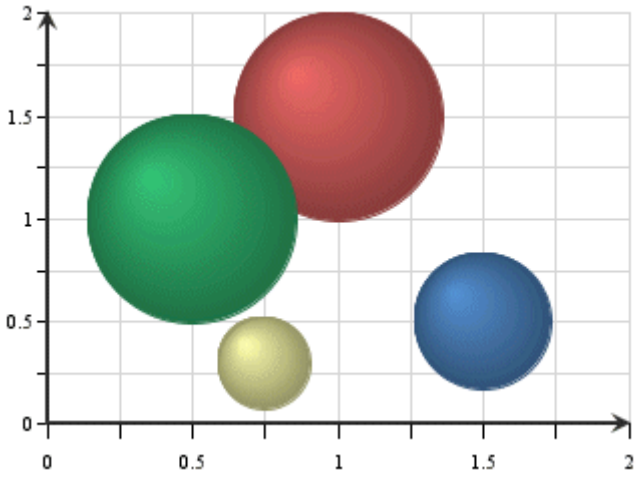
x-axis unit length : radius: バブルの半径に対する X 軸単位の長さの比を指定します。

Fill Bubble: バブル領域を埋めるかどうか指定します。

Draw Border: バブルの境界線を描画するか指定します。

3D Shading: 3D シェーディングをバブルに追加します。シェーディングの光強度を調整することもできます。

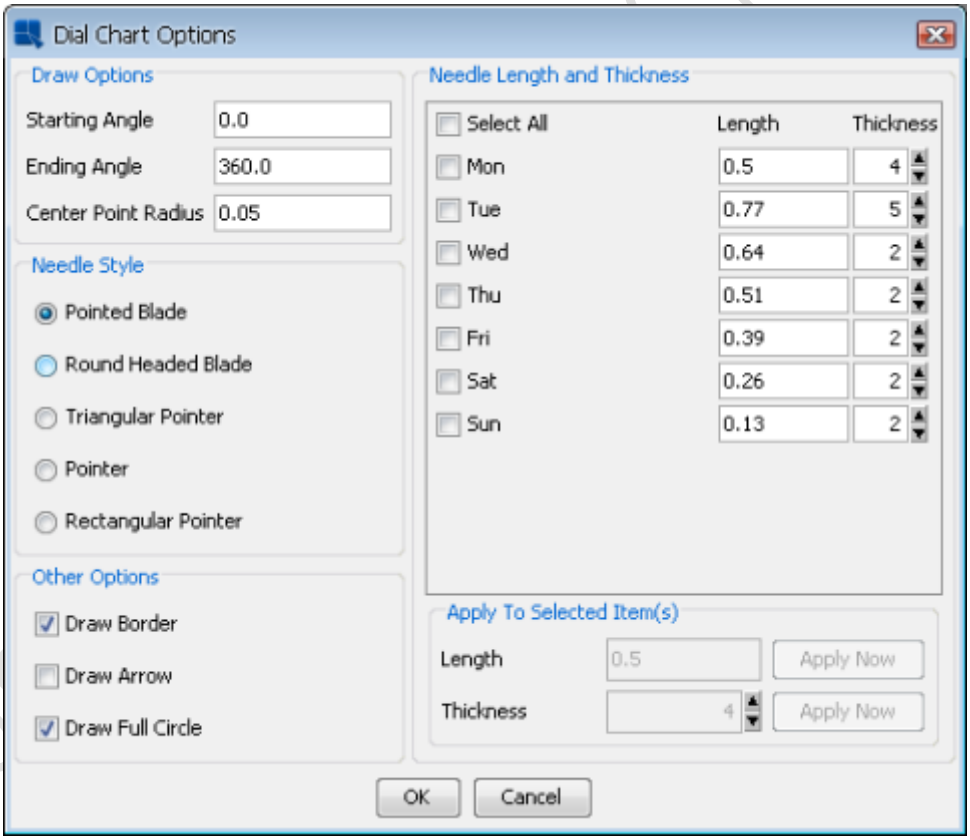
Copyright(C) 2018 Climb.Inc. All Rights Reserved.



バブルチャートの 3D シェーディング

5.5.9.2 ダイヤルチャート

ダイヤルチャートでは次のダイアログが表示されます。



Dial Options ダイアログ

ダイヤルチャートでは以下のオプションが利用可能です。

Starting Angle:

最初の軸ラベルを設定する角度を指定します。このプロパティは、**Draw Full Circle** オプションがオフの場合に、境界線とダイヤル領域の開始位置も決定します。角度は度で表され、デフォルトでは 0 です。ダイヤルチャートを時計に見立てると、0 度は 12 時になります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Enidng Angle:

最後の軸ラベルを設定する角度を指定します。**Draw Full Circle** オプションをオフにすると、境界線とダイヤル領域の終了位置も決まります。角度は度で表され、デフォルトでは 360 です。したがって、デフォルトでは、ラベル（およびデータポイント）はダイヤルの全周を囲んでいます。

Center Point Radius:

内部円の半径を指定します。これは、ダイヤルチャートの中心から始まります。半径は、ダイヤルプロットの半径に対する比率で指定します。したがって、値 1 を指定すると、内側の円がダイヤル全体を囲みます。**Draw Full Circle** オプションをオフにすると、始点と終点の範囲内にある中心点の部分だけが表示されます。

Dial Chart の中心点半径

Needle Style: 描画する針の種類を指定します。

針の種類	名前	説明
	点刃	（デフォルト）根元が太く徐々に細くなり、先端は非常に鋭い形になっています。
	丸刃	先端以外は点刃に似ています。先端が少し丸い形になっています。
	三角点	三角形のような形になっています。
	ポインタ	3 つのセグメントを持つステップラダーポインタです。
	長方形ポインタ	単純なまっすぐな針です。

Draw Border:

ダイヤルの境界線を描画するか指定します。

Draw Arrow:

ダイヤルの針の端に矢印を描くかどうか指定します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Draw Full Circle:

ダイヤルを円として 360 度すべて描画するか、開始角度から終了角度までのみを円として描画するか指定します。

Needle Length and Thickness:

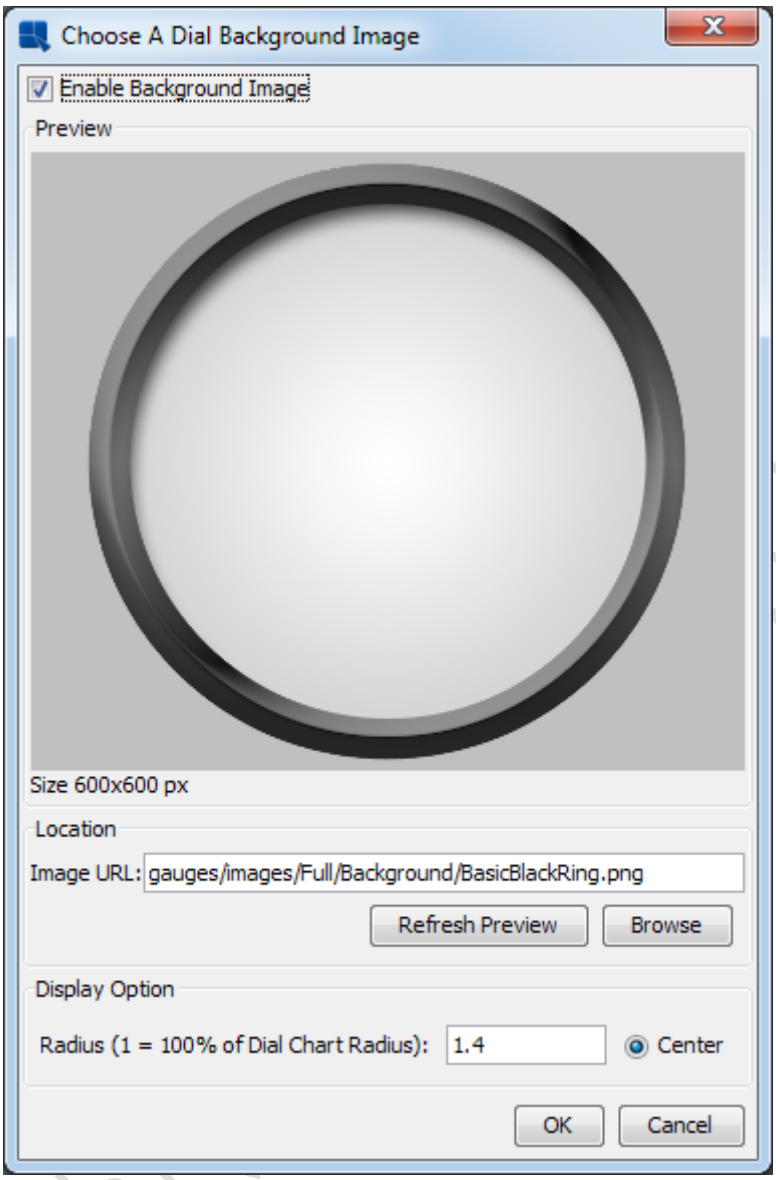
針の長さと厚さを指定します。針の長さは、ダイヤルの中央からの長さです。範囲は 0（ダイヤルの中央）から 1（ダイヤルの最後）までです。厚さによって針の幅が決定され、値が大きくなるとより広いポインタになります。チャートを作成するとき、針の長さはランダムに生成されます。デフォルトの太さは 2 です。各カテゴリ要素は異なる針で表されます。プロパティを個別に変更することも、複数のカテゴリを一度に変更することもできます。各針のプロパティを個別に変更するには、値をカテゴリの右側に直接変更します。複数の針を変更するには、各カテゴリをチェックするか、**Select All** オプションをチェックし、右下隅にあるプロパティを設定し、変更されたプロパティごとに **Apply Now** ボタンをクリックします。これにより、チェックされたすべてのカテゴリが新しい値に変更されます。

5.5.9.2.1 ゲージ画像

ダイヤルチャートには、ダイヤルプロット領域の前景または背景画像を表示するオプションが追加されています。前景または背景を追加するには、**Insert > Dial Forgroound...**または **Insert > Dial Background...**を選択します。これらのオプションは、ダイヤルチャートでのみ有効です。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



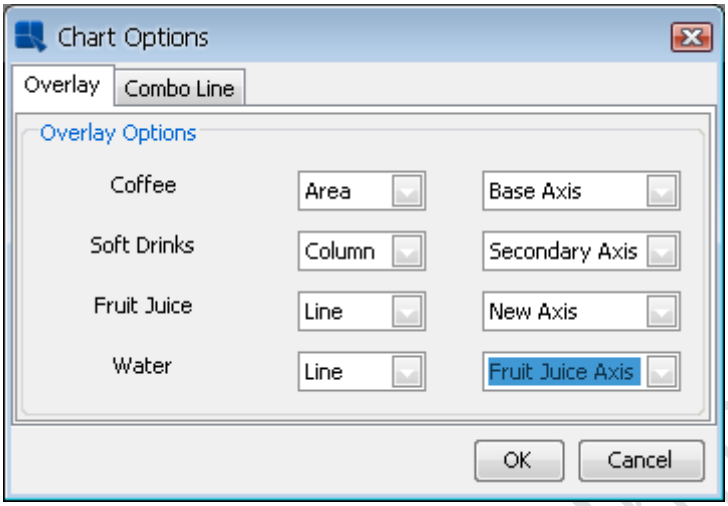
Dial Chart Background Image ダイアログ

イメージの選択は、Background image ダイアログと同じ方法で動作します（背景画像を参照してください）。Diak Chart Background Image ダイアログには、画像の半径を指定するオプションもあります。半径を 1 に指定すると、イメージはプロット領域と同じ幅と高さになります。この値を増減すると画像が拡大または縮小されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.9.3 オーバーレイチャート

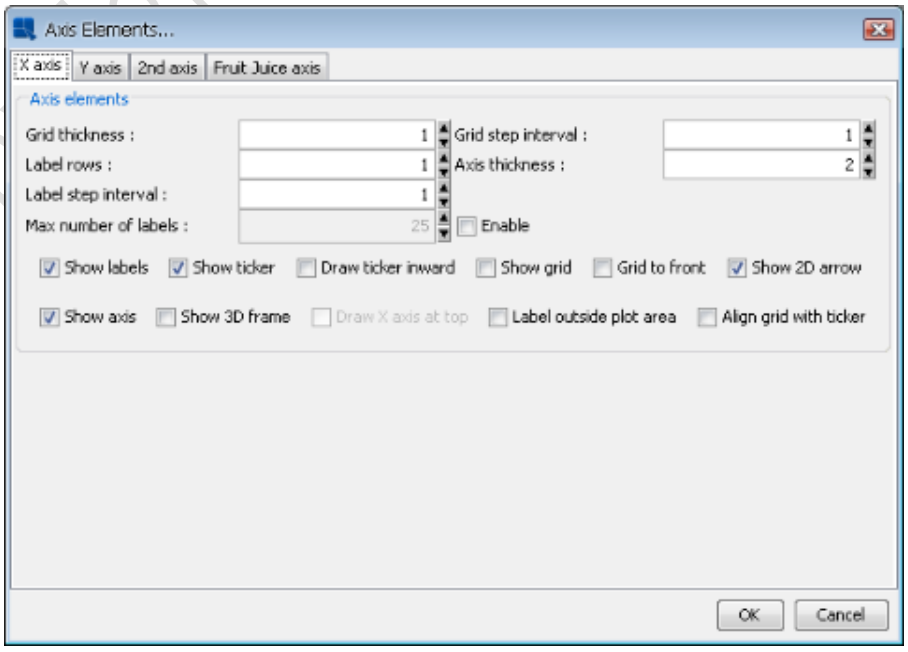
オーバーレイチャートでは、次のダイアログが表示されます。



Overlay Options ダイアログ

このダイアログでは、データ系列の各要素に使用するグラフの種類を指定できます。オーバーレイチャートの系列要素に使用できるグラフの種類は、列、領域、および行です。特定のシリーズ要素を表示しないように選択することもできます。

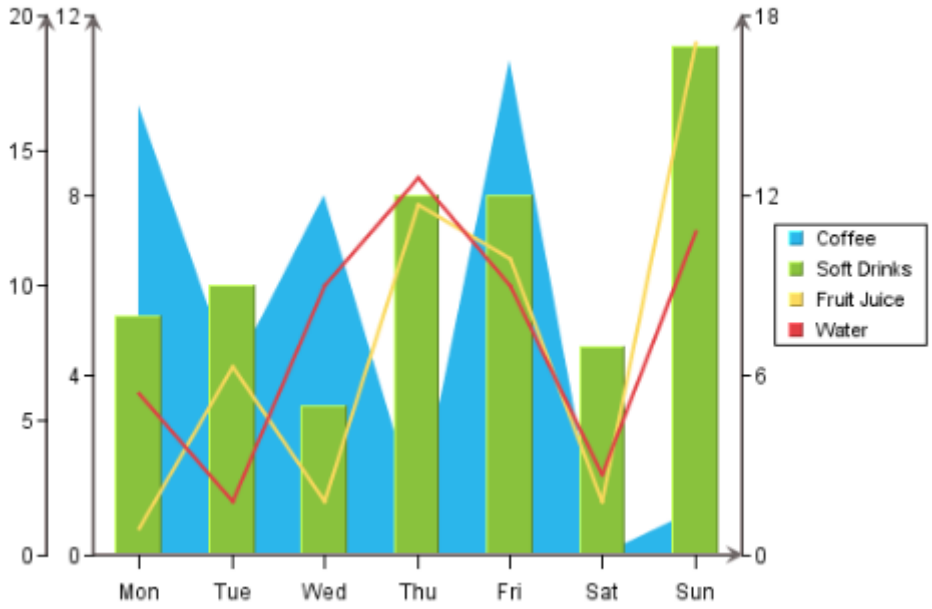
また、系列要素のプロットに使用する軸を指定することもできます。主軸または第 2 軸に要素を配置することも、シリーズ要素の新しい値軸を作成することもできます。新しい値の軸を作成するには、ドロップダウンメニューから **New Axis** を選択します。新しい軸を使用するように指定すると、他のデータ系列要素のドロップダウンメニューに新しいオプションが追加され、以前指定した軸と同じ軸にその軸を描くことができます。さまざまな軸を使用すると、さまざまなシリーズ要素が使用するスケールを正確に調整できます。これらの各軸には Axis Elements ウィンドウに独自のタブがあり、各軸のラベルステップ間隔を他と独立して変更できます。



複数の値の軸の Axis Options ダイアログ

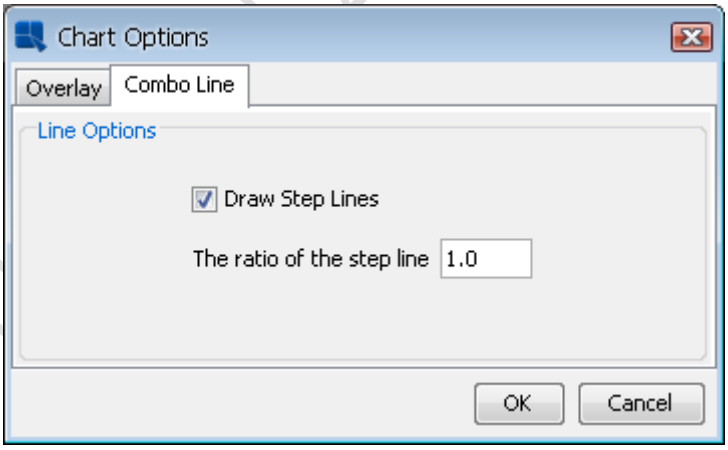
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



複数の値の軸を用いたオーバーレイチャート

オーバーレイチャートで使用されるチャートタイプの 1 つがラインである場合、**Combo Line** タブを使用してラインをステップラインとして描画するかどうかを指定できます。ステップラインの詳細については、線グラフを参照してください。

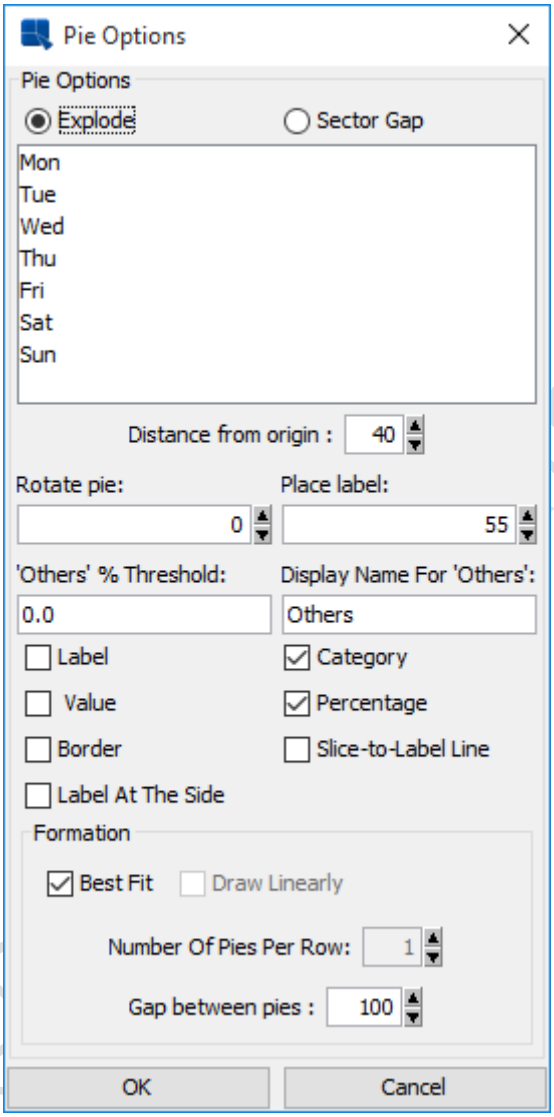


オーバーレイチャートの Combo Line オプション

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.9.4 円グラフ

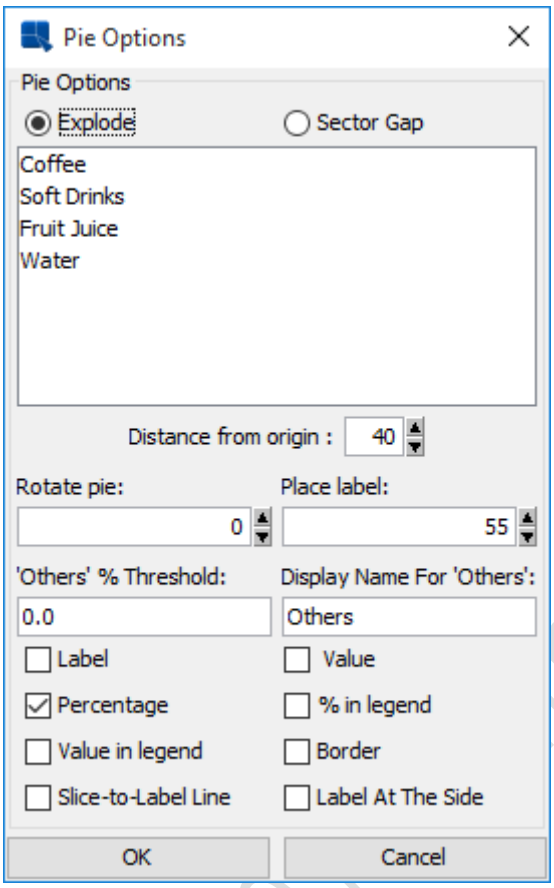
円グラフの場合は、次のダイアログが表示されます。(グラフにシリーズがあるかどうか、2D または 3D チャートであるかどうかによって、さまざまなオプションが表示されたり消えたりすることに注意してください。) シリーズのある 2D チャートで使用できるオプションは次のとおりです。



シリーズを持つ Pie Options ダイアログ

シリーズなしの 2D チャートで使用できるオプションは次のとおりです。編成オプションが削除され、凡例の%と凡例の値が追加されていることに注意してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



シリーズを持たない Pie Options ダイアログ

パイチャートでは次のオプションが利用可能です。

Explode:

セクションが円の中心から一定の距離を置いて描画される 1 つ以上のカテゴリ/シリーズ要素を選択できます。

Sector Gap:

セクションが円の中心から一定の間隔を置いて描画され、円の境界線と円の境界の間に同じ間隔を維持する 1 つ以上のカテゴリ/シリーズ要素を選択できます。

Distance from origin

exploded/sector ギャップセクションを中心からどれだけ引き離すかを指定できます。半径のパーセンテージで表されるこの数値は、エクスポートされる扇形の中心と先端の間の距離を示します。

Rotate pie:

チャートを時計回りに回転させる角度を指定できます。使用可能な値は 0～360 です

Place label:

円の中心からのラベルの距離を示します。個々のラベルの位置は、テキストをドラッグして調整することもできます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

“Others” % Threshold:
多くの小さなカテゴリを持つ円グラフに役立ちます。各カテゴリのスライスを描画するのではなく、値列のパーセンテージが指定したしきい値未満のカテゴリを、**Others** のスライスにまとめます。

Display Name for “Others”:
指定したしきい値を下回るカテゴリに対して作成された **Others** スライスの表示名を設定できます。このラベルは、凡例およびスライスラベルに表示されます。

Label:
各円のスライスに対してカテゴリ/系列ラベルを描画するかどうかを決定します。デフォルトでは、これらは凡例項目としてのみ表示されます。スライスのデータが 0 または null の場合、ラベルは表示されません。

Value:
このオプションでは、各円グラフの実際の値を表示するかどうかを指定できます。スライスのデータが 0 または null の場合、値は表示されません。

Category:
各スライスにカテゴリ名を表示するか指定します。

% in legend:
このオプションを使用すると、凡例の円の各スライスで表される割合を表示できます。円のスライスが薄いと、見やすくなります。このオプションは、円グラフにデータ系列がない場合にのみ使用できます。

Value in legend:
凡例の円の各スライスで表される値を表示できます。円のスライスが薄いと、見やすくなります。このオプションは、円グラフにデータ系列がない場合にのみ使用できます。

Border:
各円グラフの周りに枠線を描画するかどうかを指定します。このオプションは、2 次元パイチャートでのみ使用できます。3 次元パイチャートの場合は、ナビゲーションパネルの枠線描画オプションを使用できます。スライスのデータが 0 または null の場合、境界線は表示されません。

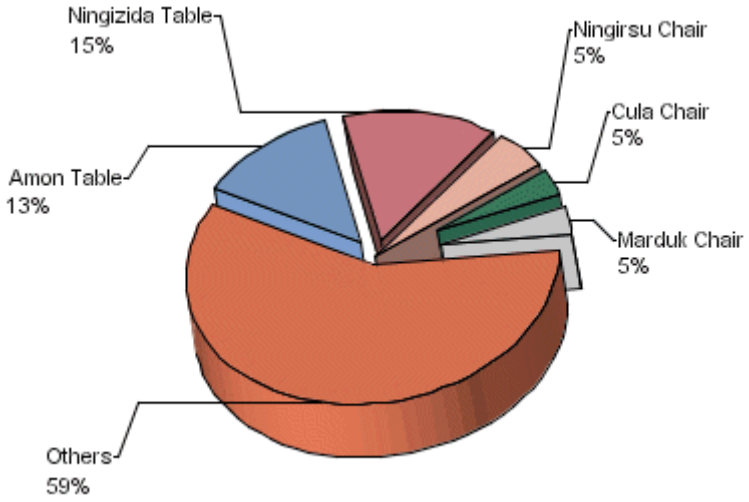
Slice to Label Line:
任意のラベルから対応する円のスライスに線を描画します。スライスのデータが 0 または null の場合、スライスからラベルへの行は表示されません。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Label at the Side:

円グラフのラベルをグラフの外側のプロットから離して配置します。**Slick-to-Label Line** オプションを使用すると、テキストが重複せずに多数の小さなカテゴリを持つチャートの円のラベルを表示することができます。スライスのデータが 0 または null の場合、ラベルは表示されません。



サイドラベルと線のあるパイチャート

Best Fit:

複数の円をチャートキャンバスに合わせて最適な構成に整列させます。これは、データ系列のある円にのみ使用できます。

Draw Linearly:

複数の円を直線の水平線で並べます。これは、データ系列のあるパイにのみ使用できます。

Number of Pies Per Row:

配列の各行に描画する円の数を指定することで、複数の円のカスタム配列を作成できます。

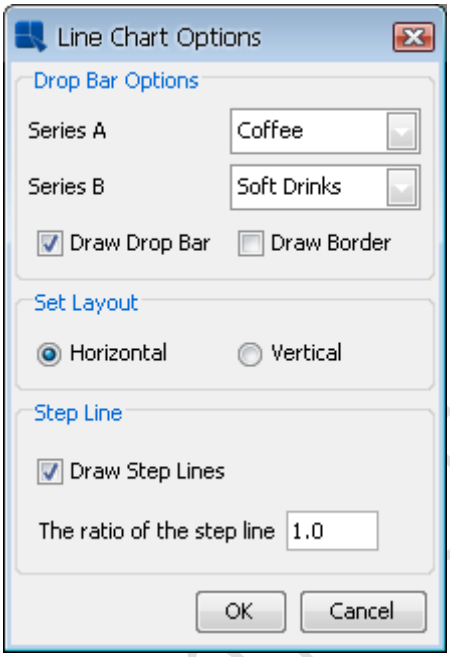
Gap between pies:

複数の円の間隔を指定できます。数値は円の半径の倍数なので、間隔はグラフプロットのサイズに合わせて調整されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.9.5 折れ線グラフ

2 次元線グラフの場合、グラフにデータ系列があるかどうかによって 2 つのうちどちらかのダイアログが表示されます。グラフにデータ系列がある場合、次のダイアログが表示されます。



Line Option ダイアログ（データ系列あり）

データ系列を持つ線グラフには、2 つの系列要素間にドロップバーを描画するオプションがあります。ダイアログのオプションは次の通りです。

Series A:
ドロップバーの最初の系列要素を指定します。

Series B:
ドロップバーの 2 番目以降の系列要素を指定します。

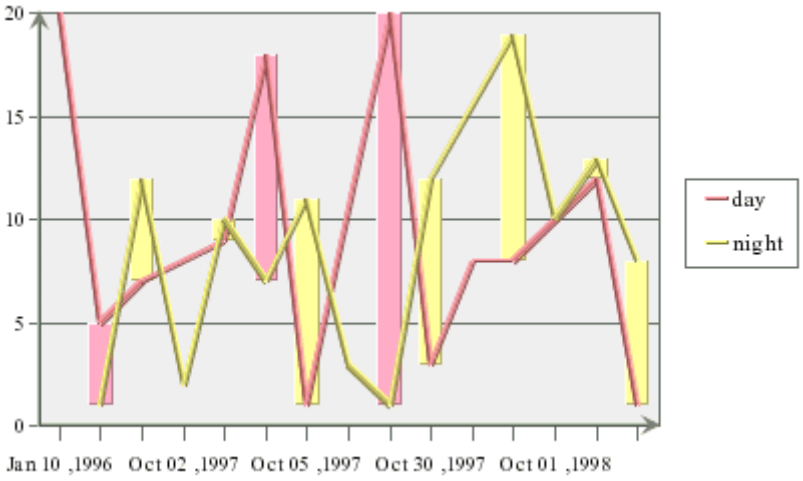
Draw Drop Bar:
ドロップバーを描画するか指定します。

Draw Border:
ドロップバーの枠線を描画するか指定します。

Set Layout:
縦横どちらの向きで折れ線グラフを描画するかどうかを指定します。

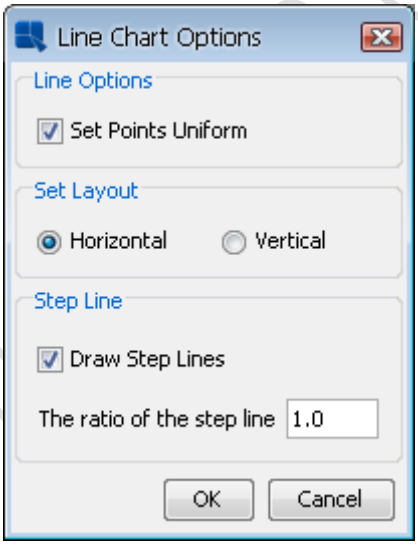
Step Line:
折れ線グラフをステップラインとして描くことができます。ステップライン比の指定も可能です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



ドロップバーありの線グラフ

ドロップバーの色は、どの系列が与えられた点でより高い値を持つかによって変わることに注意してください。折れ線グラフに系列がない場合、次のダイアログが表示されます。



Line Options ダイアログ（データ系列なし）

このダイアログでは次のオプションを指定します。

Set Points Uniform:

点の形や色を統一するか指定します。このオプションのチェックを外すと、データポイントに複数の色とプロットの形を設定できます。点は、line and point ダイアログでカスタマイズすることができます。

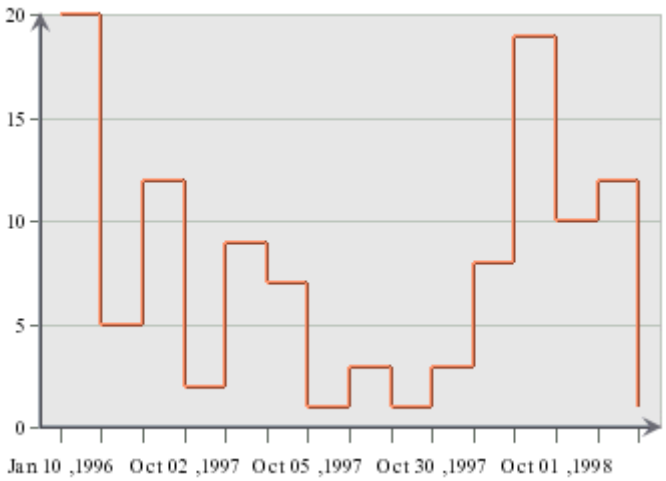
Set Layout:

縦横どちらの向きで折れ線グラフを描画するかどうかを指定します。

Step Line:

折れ線グラフをステップラインとして描くことができます。使用するステップライン比を指定することもできます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



ステップラインとして描画した線グラフ

ステップライン比を使用すると、ステップラインの水平部分を描画するポイント間の距離を指定できます。比率が 1 の場合、グラフ上の次の点に水平に描画され、その点に垂直に描画されます。比率が 0.5 の場合、2 つの点の中間に水平部分が描画されます。比率が 0 の場合、線の垂直部分が最初に描画され、次に水平方向に次の点に接続されます。比率は 0 と 1 の間の値を指定します。

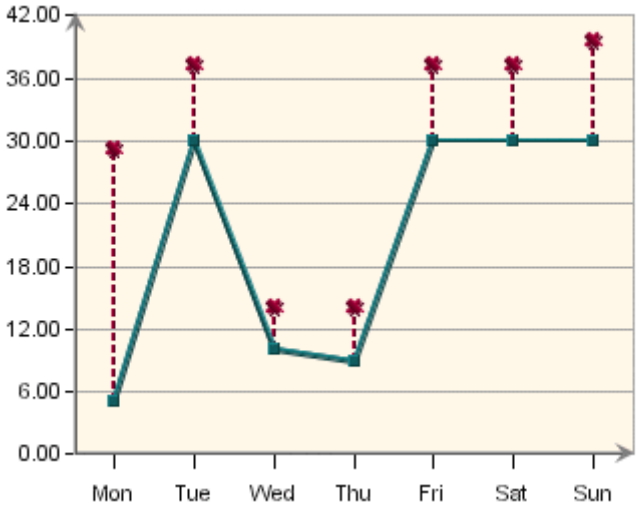
線グラフの 3D 表示の際の追加オプションはありません。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

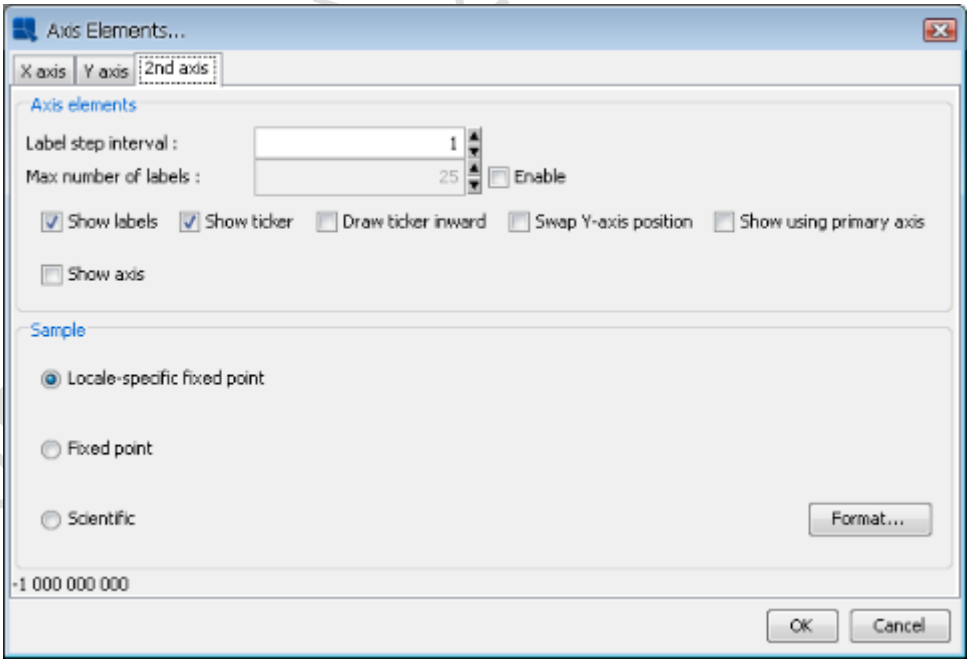
5.5.9.5.1 2 つの値を用いる折れ線グラフ

ChartDesigner には、同じ行に 2 つの値を表示できるようにする線グラフの特別なオプションがあります。ここでは、2 軸を使用して第 2 の値をプロットし（折れ線の組み合わせのように）、主軸の線と結合します。



2 つの値を用いる線グラフ

2 値の線グラフを作成するには、線グラフ（一次値と二次値を持つ線グラフ）を設計します。**Format** ＞ **Axis Elements** を選択します。これにより、axis elements ダイアログが表示されます。



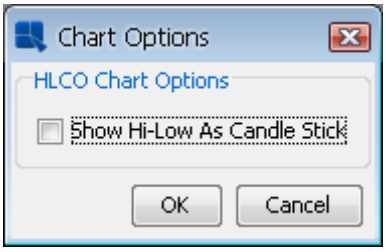
2 値の線グラフの Axis Elements ダイアログ

2nd axis タブの下に、**Show using primary axis** のチェックボックスがあります。このチェックボックスをオンにして、**OK** をクリックします。チャートが二重線の線グラフとして描画されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

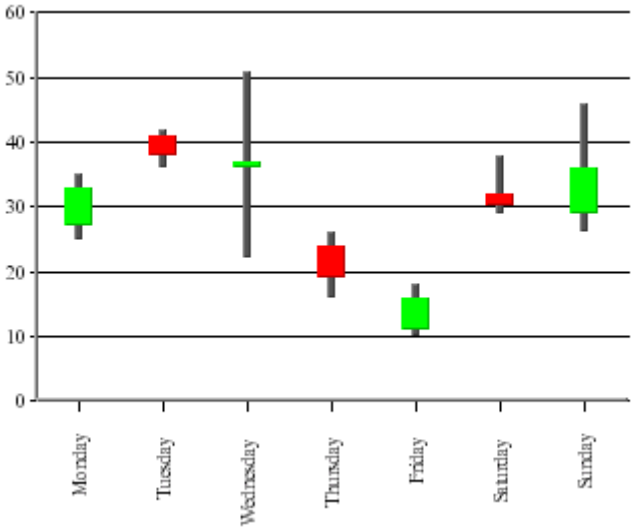
5.5.9.6 HLCO チャート

HLCO チャートでは次のダイアログが表示されます。



HLCO Options ダイアログ

Show Hi-Low As Candle Stick オプションは、HLCO チャートをろうそく足に変えます。ろうそく足の HLCO チャートは、ローソク足に似ている単一のオブジェクトに始値、安値、高値、終値のデータを縦線で表示します。



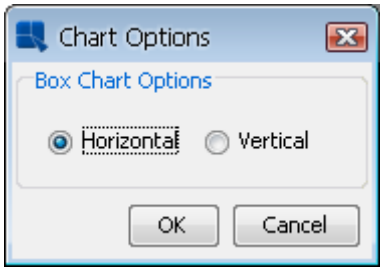
ろうそく足 HLCO チャート

Copyright(C) Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.9.7 ボックスチャート

ボックスチャートでは次のダイアログが表示されます。

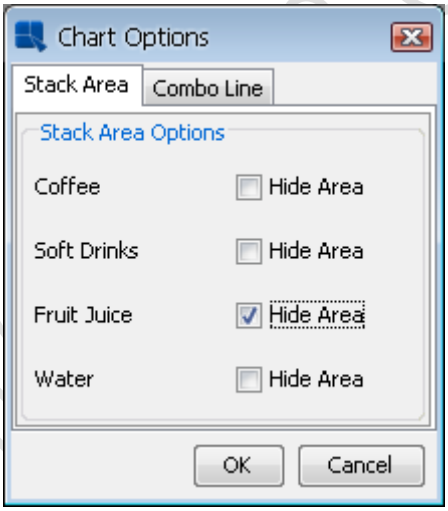


Box Options ダイアログ

このダイアログでは、ボックスチャートを水平または垂直方向に表示するかどうかを指定できます。

5.5.9.8 スタックエリアチャート

スタックエリアチャートでは次のダイアログが表示されます。



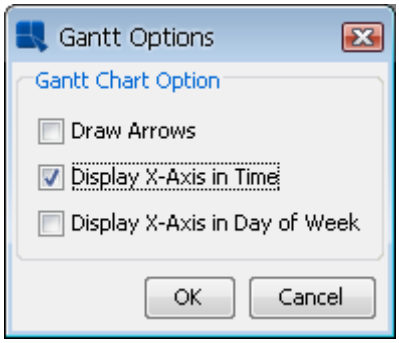
Stack Area Options ダイアログ

対応するチェックボックスをクリックすることで、チャート内の積み上げを非表示にすることができます。**Combo Line** タブでは、チャートがラインスタック領域の組み合わせである場合にステップラインを指定できます。3D スタックエリアチャートの追加オプションはありません。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.9.9 ガントチャート

ガントチャートでは次のダイアログが表示されます。

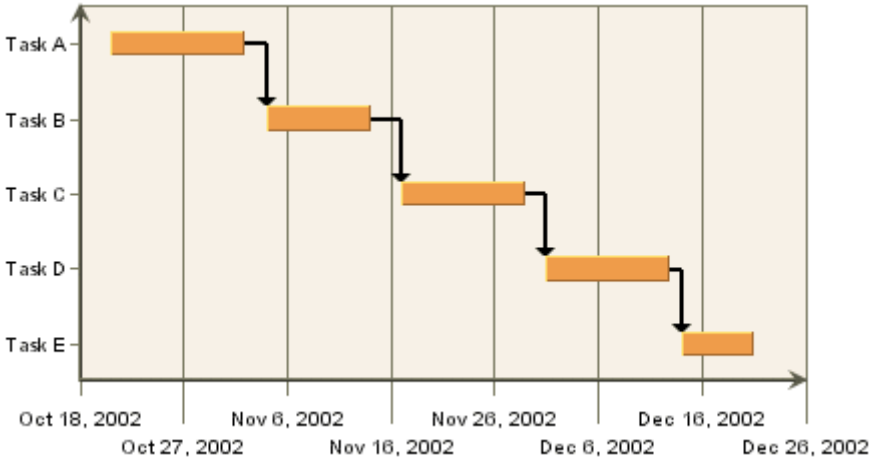


Gantt Options ダイアログ

ガントチャートでは次のオプションが利用可能です。

Draw Arrows:

ガントチャートのカテゴリ要素の間に接続矢印を描画するか指定します。これにより、スケジュールされたイベント間のシーケンスを説明することができます。矢印は、カテゴリ要素がデータソースに表示される順序で表示されます。



矢印のあるガントチャート

Display X-Axis in Time:

X 軸を X 軸の数値ではなく時間値として表示します。

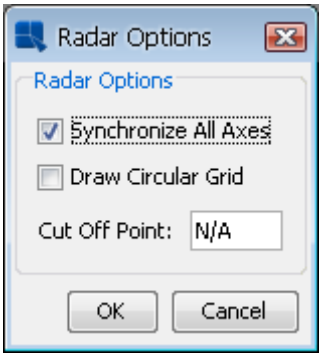
Display X-Axis in Day of Week:

X 軸を曜日およびその日付として表示します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.9.10 レーダーチャート

レーダーチャートでは次のダイアログが表示されます。



Radar Options ダイアログ

デフォルトでは、スケールはレーダーチャートのすべての軸で同じです。**Synchronize All Axes** のチェックを外すと、レーダーチャートの各軸を個別にスケーリングできます。各軸に自動スケーリングを使用するか、axis scale ダイアログを呼び出して手動でスケールを設定するかを選択できます。

Draw Circular Grid では、レーダーチャートの描画方法を設定できます。デフォルトでは、グリッドが有効な場合、グリッドは各軸の点を結ぶ直線で描画されます。**Draw Circular Grid** を有効にすると、極座標グラフに似た円でグリッドが描画されます。

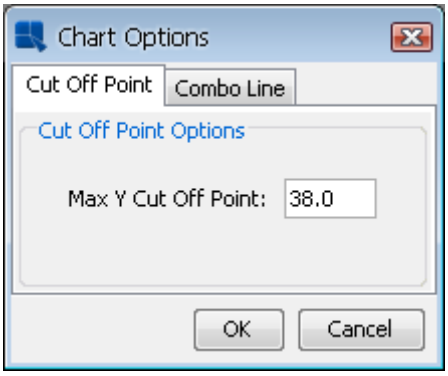
Cut Off Point では、レーダーチャートのデータポイント（区域）のカットオフポイントを指定できます。グラフに表示する最大値を入力することができます。データポイントで囲まれた領域は、指定されたカットオフポイントを超えて描画されません。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.9.11 散布図

散布図では次のダイアログが表示されます。



Scatter Options ダイアログ

Max Y Cut Off Point オプションを使用すると、散布図の Y 点の最大値を指定できます。このしきい値を超える座標はプロットされません。 接続線はしきい値の端まで引き上げられ、次のデータ点に進みます。

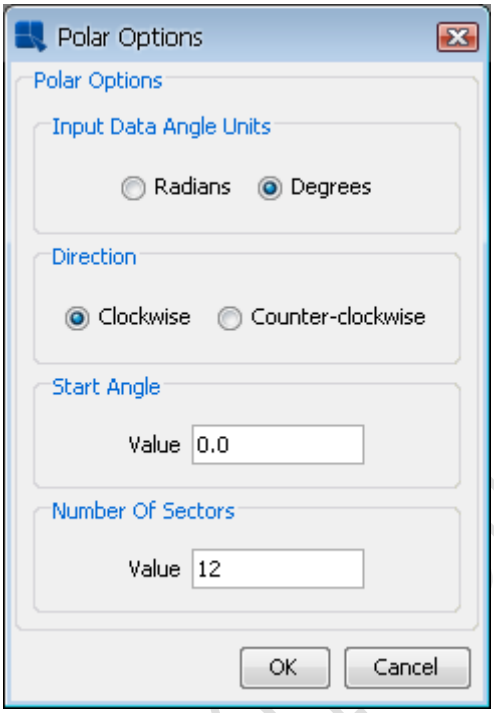
Combo Line オプションを使用すると、接続ラインをステップラインとして描画し、ステップライン比を指定することができます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.9.12 極座標グラフ

極座標グラフでは次のダイアログが表示されます。



Polar Options ダイアログ

Scale:
データポイントの角度部分の入力データを弧度法か度数法の選択ができます。グラフには常に 0～360 の角度が表示されます。入力データがラジアンの場合においても度数として表示されます。

Direction:
プロットを時計回りか反時計回りに描画するか指定します。

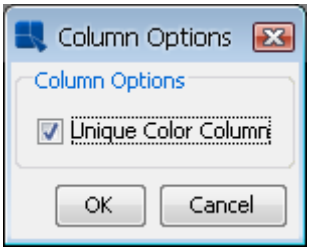
Start Angle:
デフォルトでは、極座標グラフプロットの上端は 0 度です。このオプションを使用すると、プロットの上端に異なる角度を指定できます。この角度の引数は、選択したスケールに応じて度またはラジアンで指定します。

Number of Sectors:
チャートに表示するセクタの数を選択できます。セクタは、指定された角度間隔で追加の極軸線を描画することによって作成されます。デフォルトでは、4 つのセクタが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.9.13 データ系列を持つ縦棒グラフ

データ系列を持つ列グラフでは次のダイアログが表示されます。

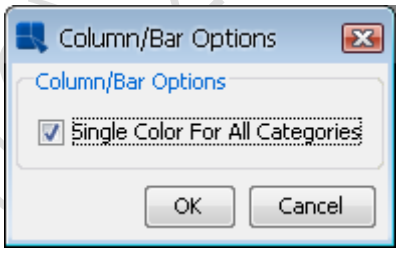


Column Options ダイアログ

通常、縦棒グラフにデータ系列がある場合、各系列にはチャートの各カテゴリに適用される独自の色があります。系列に関係なくチャートの列に異なる色を割り当てる場合は、このダイアログで **Unique Color Column** を有効にします。これをオンにすると、チャート内の各列の色を個別に設定できます。

5.5.9.14 データ系列のない横棒グラフ

データ系列を持たない列/横棒グラフでは次のダイアログが表示されます。



Column/Bar Options ダイアログ

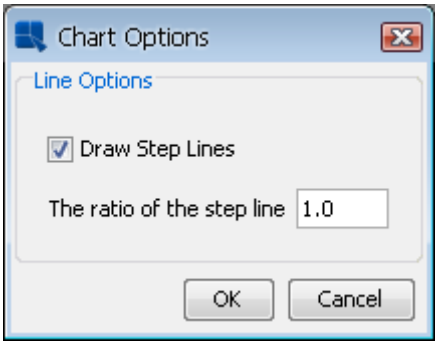
通常、列/棒グラフにデータ系列がない場合、チャート内のすべてのカテゴリは単一色です。チャートのカテゴリに異なる色を割り当てる場合は、このダイアログで **Single Color For All Categories** オプションのチェックを外します。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.9.15 ２D の組み合わせ線グラフ

他の 2D 組み合わせ線グラフでは次のダイアログが表示されます。



Line Combination Chart Options ダイアログ

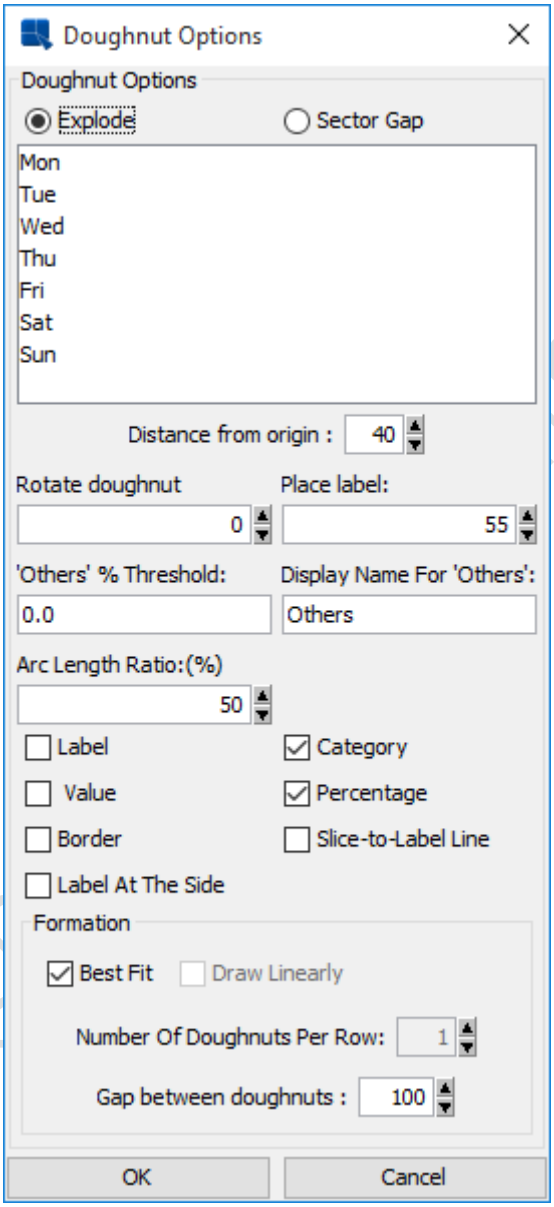
コンボラインをステップラインとして描画するか、また、ステップライン比を指定することができます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.5.9.16 ドーナツチャート

ドーナツチャートのグラフオプションは、パイチャートのオプションとほぼ同じです。詳細は円グラフを参照してください。



Doughnut Chart Options ダイアログ

ドーナツチャート固有のオプションは、チャートの中央にある穴のサイズを指定する比率（%）です。数値が大きいほど穴が小さくなります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.6 ドリルダウン

レポートと同様に、チャートにドリルダウンのレイヤーを追加することもできます。追加されたすべてのドリルダウンは、プロット領域のデータポイントと凡例のそれぞれのフィールドに適用されます。チャートのドリルダウンを使用すると、要約データを表示するトップレベルのチャートを作成し、特定のデータポイントをクリックして基礎となる詳細を表示することができます。

EspressReport のチャートエンジンには、さまざまな組み込みのドリルダウンメカニズムが含まれており、ドリルダウンの各レベルに対して 1 つのチャートテンプレートのみを作成できます。ドリルダウンオプションはすべて、Chart Designer のドリルダウンメニューから使用できます。

チャートがレポートとは独立して配備されている場合（Chart Viewer、サーブレットなど）は、ドリルダウンのみをチャートで使用できることに注意してください。ドリルダウンを含むチャートテンプレートがレポートに配置されている場合は、最上位のチャートのみが表示されます。ユーザーは、下位レベルの図をクリックすることはできません。

5.6.1 データドリルダウン

データドリルダウンを使用すると、1 つのデータソースに基づいて情報をグループ化して表示できます。このドリルダウンの利点は、どのデータソースでも機能することです。ただし、ドリルダウンのすべてのレベルが入力データと同じ値の列を共有するため、関連性の低い情報を表示することはできません。

たとえば、チャートデータとして以下の表があるとします。

カテゴリ	製品	販売
Chairs	Elm Arm Chair	$8,216
Chairs	Pine Side Chair	$7,611
Chairs	Redwood Arm Chair	$8,625
Tables	Elm Round Table	$10,241
Tables	Pine Oval Table	$9,663
Tables	Oak Oval Table	$11,261
Dressers	Oak Single Dresser	$16,442
Dressers	Elm Double Dresser	$17,148

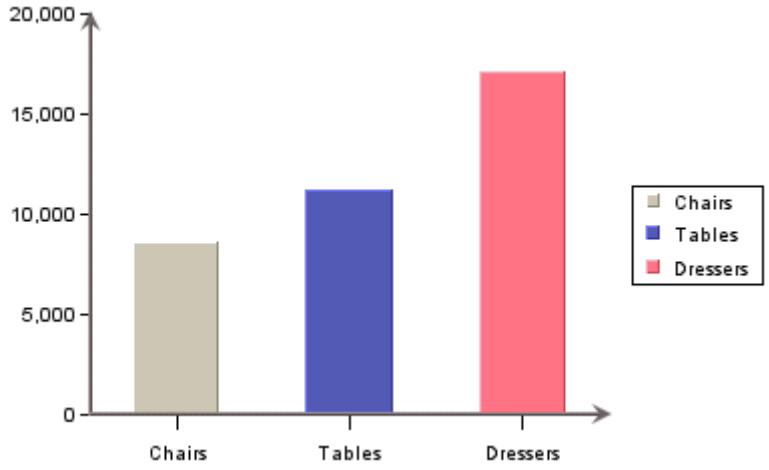
このデータを使用して、個別の製品カテゴリごとの合計売上高を表示し、カテゴリ内の各製品の個別の売上高を示す下位レベルのグラフを作成するトップレベルのグラフを作成できます。利用可能なドリルダウンレベルの数は、入力データに存在するグループ化の数によって異なります。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

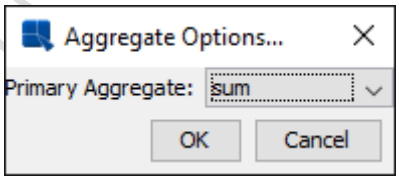
5.6.1.1 データドリルダウンの追加

ドリルダウンしたデータのレイヤーをチャートに追加するには、まず最上位のチャートを作成する必要があります。上記の例では、カテゴリ軸にマップされたカテゴリと値軸にマップされた売り上げのチャートを作成します。縦棒グラフでは、以下のようになります。



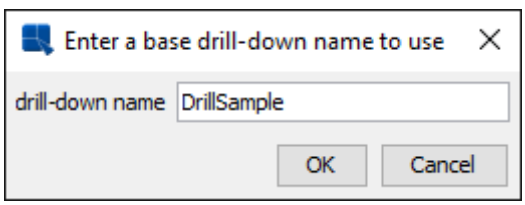
トップレベルチャート

ドリルダウンのレイヤーを追加するには、**Drill-Down** > **Add** を選択します。これにより、値軸に使用する集約を指定するダイアログが表示されます。たとえば、合計を選択すると、最上位のグラフに各カテゴリの合計が表示されます。使用可能な集計関数は、最小（min)、最大（max)、平均（average)、合計（sum)、カウント（count)、最初（First)、最後（Last)、平方根（sumsquare)、分散（variance)、標準偏差（stddev)、および countdistinct です。



Aggregate ダイアログ

使用する集約を選択したら **OK** ボタンをクリックします。ドリルダウンチャートの名前を指定するための新しいダイアログが表示されます。この名前は、ドリルダウンプロセスによって作成されたチャートテンプレートはすべて、**/drilltemplates/**ディレクトリに保存されます。

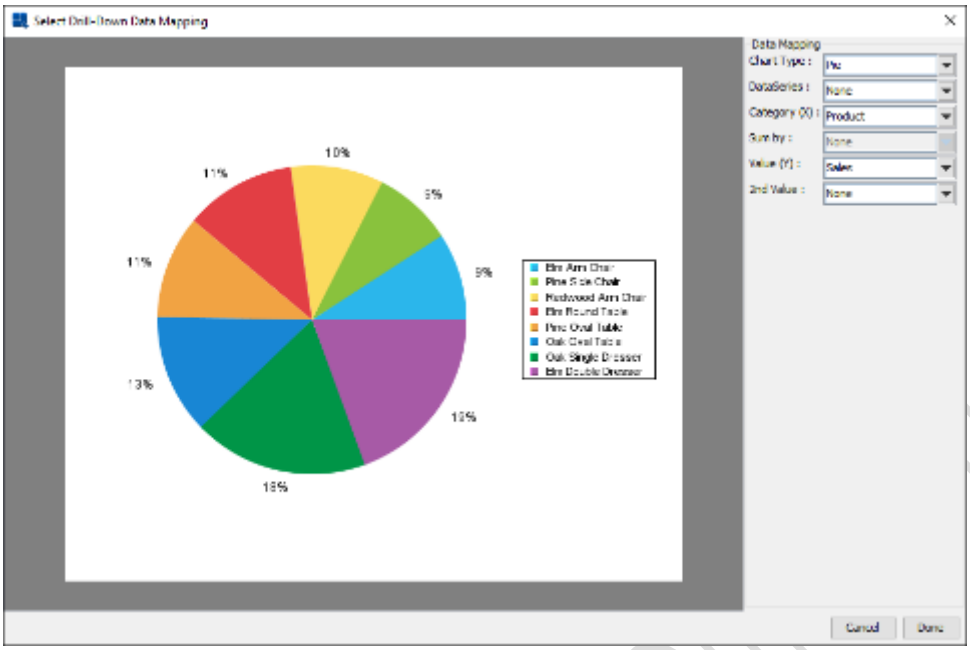


Drill-Down Name ダイアログ

使用する名前を指定したら、**OK** ボタンをクリックします。次に、サブレベルのグラフの種類とデータ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

のマッピングを指定するよう求められます。



Drill-Down Data Mapping ダイアログ

ダイアログのオプションは、通常のデータマッピングのオプションに似ています。主な違いは、最初のオプションではグラフの種類を選択できることです。データドリルダウンに使用できるチャートの種類は、列、棒、線、スタックの列、スタックバー、円、面積、ドーナツ、オーバーレイ、レーダー、およびダイヤルです。データマッピングオプションは、選択したチャートのタイプによって変わります。

マッピングオプションの指定が完了したら、**Done** ボタンをクリックすると、サブレベルのチャートをカスタマイズして変更できる Chart Designer に戻ります。**Drill-Down** > **Add** を再度選択して、ドリルダウンのレイヤーを追加することができます。

Drill-Down > **Previous** または **Drill-Down** > **Next** を選択するか、データポイントをダブルクリックして、特定のドリルダウンチャートに移動できます。**Previous** を選択するとレベルが上がり、**Next** は下に移動し、左端のカテゴリをドリル対象のデータとして使用します。また、**Drill-Down** > **Top Level** に移動を選択することによって、ドリルダウンチャートからトップレベルチャートに移動することもできます。ドリルダウンチャートは、トップレベルチャートと同じ方法でカスタマイズ（つまり、色の割り当て、タイトルの追加、軸ラベル、背景画像の追加など）することができます。変更してレベルを離れるたびに、グラフを保存するように求められます。ドリルダウンレベルの属性を保存する場合は、**yes** と答えてください。そうでない場合は、すべての変更が失われます。

上位レベルのドリルダウンチャートに移動し、**Drill-Down** > **Add** を選択することにより、他の 2 つのドリルダウンチャートの間にドリルダウンチャートを挿入することができます。作成したドリルダウンチャートは、前の 2 つのドリルダウンチャートの間に挿入されます。

ドリルダウンのレベルを削除するには、そのレベルに移動して **Drill-Down** > **Remove** を選択します。**Drill-Down** > **Remove All** を選択することによって、ドリルダウンチャート全体を削除することもでき

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ます。**Drill-Down** > **Previous** への選択は、ドリルダウンチャートをより高いレベルにします。これは、右クリックしてポップアップメニューから **Back** を選択するのと同じです。**Next** を選択すると、ドリルダウンチャートが下位レベルになります。これは、チャート内の項目をダブルクリックするのと同じ機能を持ちます。

ドリルダウンチャートのすべてのレベルの作成と編集が終了したら、最上位のチャートに移動して **File** > **Save または File** > **Save as** を選択して保存することができます。

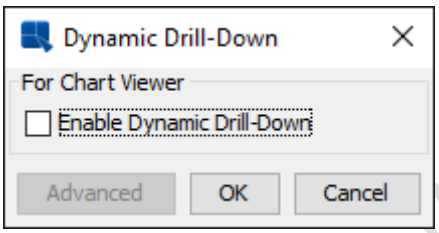
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.6.2 動的データドリルダウン

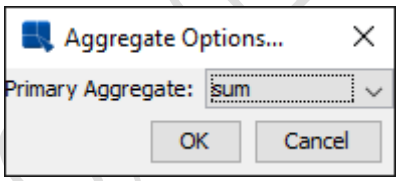
通常、ドリルダウンされるデータのマッピングオプションとドリルオプションは、設計時に固定されます。動的ドリルダウンは、マッピングを選択できる追加のオプションです。設計時に指定されるのは、最上位のグラフと集計だけです。

動的ドリルダウンを使用してチャートを作成するには、まずトップレベルチャートとして使用するチャートを作成します(たとえば、前と同じトップレベルチャートを作成できます)。次に、**Drill-Down** > **Dynamic** を選択します。動的ドリルダウンを有効にするためのダイアログが表示されます。



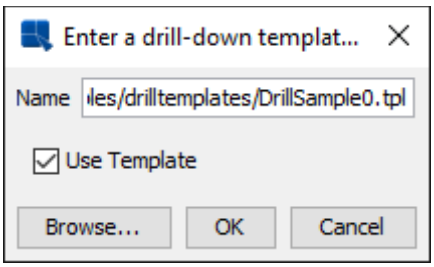
Enable Dynamic Drill-Down ダイアログ

Enable Dynamic Drill-Down チェックボックスをオンにすると、集計を選択するように促す新しいダイアログがポップアップします。



Aggregate ダイアログ

このダイアログのオプションは、通常のデータドリルダウンと同じです。使用する集約を指定したら、**OK** ボタンをクリックします。別のダイアログがポップアップし、サブレベルのグラフに使用するテンプレートを指定することができます。



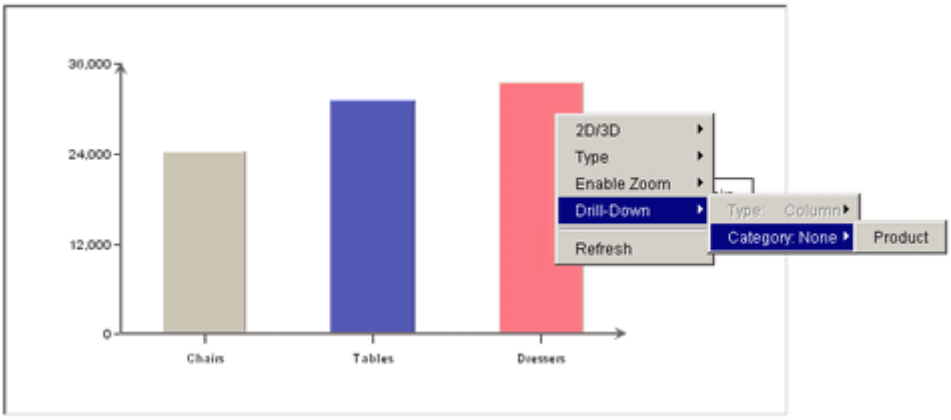
Enter a drill-down template ダイアログ

テンプレートを使用するように選択しなかった場合は、デフォルトの外観プロパティを使用してサブレベルのグラフが生成されます。これらのオプションの選択が完了すると、チャートは値軸に集約データを表示するように変更されます。**Drill-Down** > **Dynamic** をもう一度選択し、次に **Advanced** ボタンをクリックしてオプションを変更できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ダイナミックドリルダウンチャートは、チャートビューアアプレットでのみ表示できます。そこでは、デスクトップ領域を右クリックしてポップアップメニューを表示することによって、次のドリルダウンチャートの設定を構成することができます。動的ドリルダウンが有効で、グラフにプロットする未使用のフィールド（列）がまだある場合は、項目ドリルダウンがポップアップメニューに表示されます。ポップアップメニューで **Drill-Down** を選択すると、次のドリルダウンの現在の設定を表示できます。**Category** が **None** の場合は、まだ次のドリルダウンチャートを設定していないことを意味します。ドリルダウンチャートを次のレベルに設定するには、カテゴリを選択する必要があります（**Category** が設定されていれば **Type**、**Series** と **SumBy** を選択できます)。ドリルダウンチャートを作成した後、チャートビューアのさまざまなレベルに移動するには、データポイントを左クリックして 1 レベル下に移動するか、右クリックしてポップアップメニューから **Back** を選択します 1 レベル上に移動したい場合や、前の手順を繰り返して次のレベルの別のドリルダウンチャートを作成する場合があります。

ドリルダウンの下位レベルに移動した後、データポイントを右クリックすると、トップレベルのチャートに戻ります。ポップアップメニューを表示するには、チャートデータポイントから離れたチャートキャンバスを右クリックします。



チャートビューアの動的データドリルダウン

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

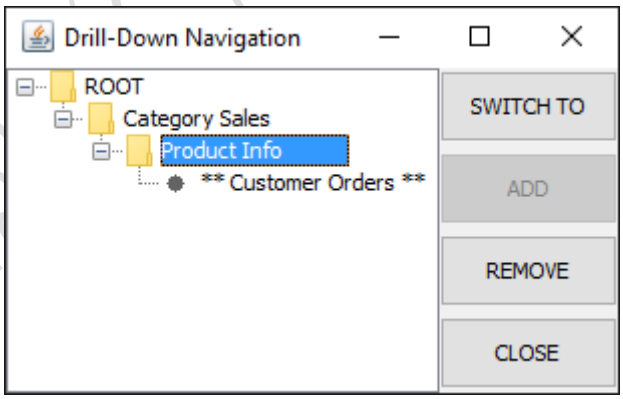
5.6.3 パラメータドリルダウン

チャートに使用できるドリルダウンの 3 つ目のタイプは、パラメータのドリルダウンです。ドリルダウンレベル間のデータは、任意の方法で関連付けることができるため、パラメータドリルダウンは最も柔軟な実装です。具体的には、各レベルに同じ値要素を使用する必要はありません。その代わりに、パラメータドリルダウンではパラメータ化されたクエリ機能を使用してさまざまなチャートレベルを関連付けます。クエリを使用するため、サブレベルのグラフのデータソースはデータベースまたはパラメータ化されたクラスファイルでなければなりません。

たとえば、前回のシナリオでは、常に売り上げを見るのではなく、トップレベルのチャートでカテゴリ別に集計された売り上げを調べ、次のレベルでは各商品の販売数量を調べるそこから、各製品の在庫レベルを地域ごとに調べる必要があります。これは、パラメータのドリルダウンで実行できます。
パラメータのドリルダウンを使用すると、ドリルオンするためにクリックした要素のカテゴリ値が、サブレベルグラフにパラメータ値として渡されます。したがって、チェア列をクリックすると、チェアの値がクエリに渡されます。したがって、カテゴリ名でフィルタリングできるデータベース内のすべてのものを取得できます。

5.6.3.1 パラメータドリルダウンの追加

パラメータドリルダウンのさまざまなレイヤーを追加および編集するには、**Drill-Down** > **Parameter Drill-Down** を選択します。これにより、ドリルダウンのさまざまなレベルを示すナビゲーションウィンドウが表示されます。



パラメータドリルダウンナビゲーションウィンドウ

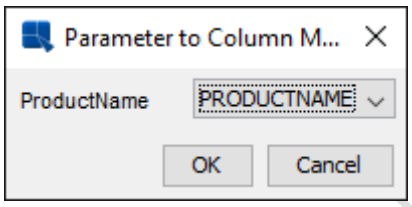
ナビゲーションウィンドウの左側にドリルダウンレベルの階層が表示されます。ROOT ノードは、トップレベルのチャートです。現在編集しているレベルには、**が表示されます。別のレベルを編集するには、そのレベルを選択し、右側の **SWITCH TO** ボタンをクリックします。デザイナーでグラフが開きます。ドリルダウンのレベルは、ナビゲーションウィンドウでノードを選択し、**REMOVE** ボタンをクリックすることによっても削除できます。

新しいレベルのドリルダウンを追加するには、レイヤーを追加するチャートを選択します(トップレベルチャートのみの場合は、ROOT ノードのみが表示されます)。**ADD** をクリックします。新しいチャート

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

を作成するか、ドリルダウンレイヤに既存のチャートを使用するかどうかを尋ねるプロンプトが表示されます。既存のグラフを使用することができます。ただし、ドリルダウンレイヤのチャートを作成することを選択すると、データソースマネージャが再び開き、チャートのデータソースを選択して、チャートウィザードの手順を実行できます。

次のステップでは、トップレベルチャートのカテゴリおよび、またはシリーズの列をサブレベルチャートのクエリパラメータにマップします。ドリルダウンレイヤの既存のチャートを選択するとき、またはドリルダウンレイヤの新しいチャートのデータソースを選択するときにこれを行うように求められます。いずれの場合も、フィールドのマッピングを促すダイアログが表示されます。



パラメータマッピングウィンドウ

ドロップダウンメニューで使用できるオプションは、データタイプに基づいています。たとえば、パラメータ化されたクエリのパラメータが string の場合、文字列データを含むフィールドのみをマップできます。したがって、カテゴリまたはシリーズのデータ型が正しくない場合や、渡すフィールドが不足している場合は、最上位のグラフから下位のグラフに渡すデータの種類を検討することが重要です。下位レベルでは、ドリルダウンが正しく機能しません。

パラメータマッピングを正しく指定すると、ドリルダウンレベルの表示名を指定するよう求められます。名前を選択すると、サブレベルのグラフが Designer に表示され、カスタマイズすることができます。ナビゲーションウィンドウを使用してドリルダウンのレイヤーをナビゲートするか、データドリルダウンのようにデータポイントをダブルクリックしてレベルを下げて右クリックし、ポップアップメニューから **Back** を選択してレベルを上げることができます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.7 チャートの保存およびエクスポート

チャートの設計が完了したら、定義をチャートまたはチャートテンプレートとして保存し、両方に XML 定義を提供できます。チャートの静的イメージのエクスポートをいくつか生成したり、Chart Viewer アプレットを埋め込んだ HTML ページを作成することもできます。

5.7.1 レポートデータによるチャートの保存

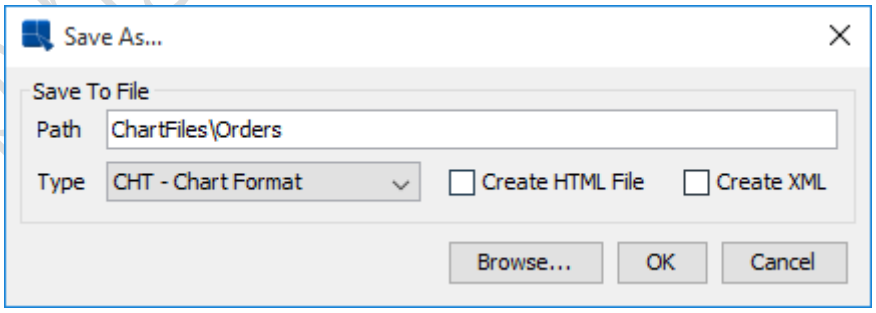
レポートデータを使用してチャートを設計する場合、チャートはレポート内以外の場所では実行/表示できません。このため、ほとんどのファイル操作は Chart Designer で無効になっています。チャートを保存することを選択すると、インストールの/ chart /ディレクトリにレポートと同じ名前で自動的に保存されます。たとえば、レポートの名前が SalesReport の場合、レポートデータを使用するレポート内のチャートは、SalesReport_0.tpl、SalesReport_1.tpl などの次の規則で保存されます。

レポートを実行するときに、これらのチャートを別の相対的な場所に配置することは可能です。ランタイム API オプションを使用すると、グラフのパスを変更できます。

グラフのレポートデータを使用する場合、ほとんどのファイルオプションは無効になっていますが、別のグラフの外観プロパティを継承するためにテンプレートを適用できます。チャートテンプレートについては、テンプレートの使用を参照してください。

5.7.2 レポートデータなしのチャートの保存

チャートがレポートの独立したデータソースを使用する場合は、任意のファイル名で任意の場所に保存できます。現在のチャートを保存するには、**File** > **Save or File** > **Save As** を選択するか、ツールバーの保存ボタン をクリックします。以前にグラフを保存していなかったと仮定するか、**File** > **Save As** を選択すると、次のダイアログが表示されます。



Save As ダイアログ

最初のオプションでは、保存したグラフの名前とファイルパスを指定できます。ダイアログの下部にある **Browse** ボタンをクリックすると、適切なファイルパスを参照できます。2 番目のオプションでは、チャートの保存時に使用するフォーマットを指定できます。チャートの基礎で詳述しているように、チャートの定義を保存するには 2 つの原則があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

グラフの形式

チャートファイルは、グラフを **filename.cht** というバイナリファイルに保存します。チャートファイルには、チャートの定義(タイプ、ディメンションなど)とチャートの作成に使用されたデータの両方が格納されます。

テンプレート形式

テンプレートファイルは、グラフを **filename.tpl** というバイナリファイルに保存します。テンプレートファイルはチャート定義のみを格納し、チャートデータは保存されません。したがって、テンプレートファイルを開くたびに、元のデータソースに接続してデータを取得しようとします。

PAC 形式

PAC ファイルは、ハイチの準備が整うようにチャートを保存します。**.PAC** ファイルは、バックグラウンドイメージ、動的ドリルダウン、またはパラメトリックドリルダウンなど、チャートとそれに関連するすべての補足ファイルを取得し、1 つのバイナリファイルに配置します。

使用するフォーマットを選択したら、**OK** ボタンをクリックすると、チャートが保存されます。

レポートが実行されると、EspressReport はチャートが保存されている相対パスに沿ってチャートをロードしようとします。グラフが見つからない場合、グラフは表示されません。ただし、API で使用できるオプションを使用して、グラフの別のパスを指定できます。

5.7.2.1 テンプレートの使用

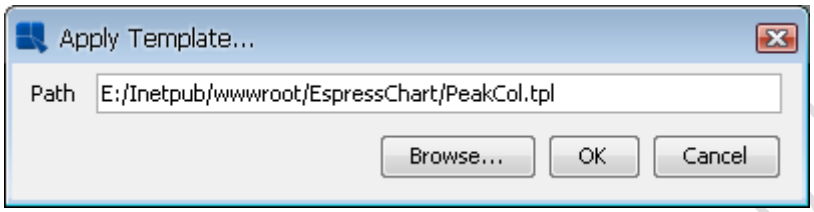
チャートテンプレートは、チャート定義を保持できる特殊なフォーマットです。グラフ定義とチャートの両方のデータを保存するチャートファイル(**.cht** 形式)とは異なり、テンプレートは定義(チャート属性と個々のコンポーネントのレイアウト)とデータソース情報のみを保存します。つまり、テンプレートには、グラフの作成に使用されたデータソースの場所・接続情報が格納されますが、実際のデータはファイルに保存されません。(テンプレートはバックアップを 10 レコード保存し、data ソースが存在しないときに開くことができます)。

テンプレートファイルは、2 つの方法で使用できます。まず開いたり、表示したり、エクスポートしたりすることができます。これにより、新鮮なデータを含むグラフを表示できます。この方法で、グラフのデータは、ファイルで指定されたデータソースに基づいて取得されます。元のデータソースにアクセスできない場合、このメソッドは使用できません。
テンプレートを使用する 1 つ目の方法は、その属性を他のグラフに適用することです。このシナリオでは、テンプレートが適用されるチャートは、テンプレートの外観プロパティ(色、フォント、チャートコンポーネントのサイズ、凡例の位置など)を継承します。この機能を使用すると、チャート間で一貫した外観を作成できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

2 つ目の方法は、API を使用して JDBC コードやほかのデータソースを使用してプログラムでチャートを生成する場合に非常に便利です。次に、事前に定義されたテンプレートを使用して、コード内の外観プロパティに依存することなく、生成されたチャートの外観を制御できます。API でテンプレートを適用する方法の詳細については、グラフテンプレートの適用を参照してください。

File > **Apply Template** を選択して、Chart Designer でテンプレートを適用できます。これにより、使用するテンプレートファイルを指定するダイアログが表示されます。



Apply Template ダイアログ

このダイアログでは、テンプレートファイルとその場所を指定できます。**Browse** ボタンをクリックすると、ファイルを参照できます。

チャートと適用されるテンプレートのサイズ(チャートキャンバスのサイズなど)が異なる場合、結果のグラフが正しく表示されないことがあります。これは、適用されるテンプレートとチャートの間でテキストサイズが変更されないためです。他のコンポーネントは新しいキャンバスのサイズに合わせて調整されますが、フォントは変更されません。一貫した外観を維持するには、テンプレートのサイズを適用するチャートのサイズに近づける必要があります。

テンプレートファイルで定義されているハイパーリンク、浮動線、浮動小数点数、および軸の取りが引き継がれます。必要に応じて、チャートでそれらを再定義する必要があるかもしれません。グラフの種類と寸法は、テンプレートによって変更されません。グラフの種類と寸法はテンプレートによって変更されません。たとえば、テンプレートが 2 次元チャートである場合、3 次元チャートは 2 次元チャートに変更されません。同様に、棒グラフのテンプレートが適用されるとき、円グラフは円グラフのままです。テンプレートを適用するとグラフの種類は変わりませんが、一部の外観プロパティは別のグラフの種類のテンプレートからはうまく翻訳されません。最良の結果を得るには、グラフに同じタイプとディメンションのテンプレートを適用してみてください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.7.2.2 XML テンプレートの保存

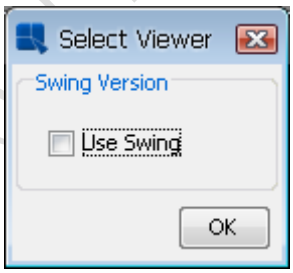
2 つのバイナリチャート表現に加えて、チャート定義を XML 形式で保存することもできます。このオプションを使用すると、Chart Designer または API の外部でグラフプロパティを変更するために使用できるテキストベースのグラフテンプレートを作成できます。

グラフを保存するときに XML ファイルを作成するには、グラフを保存するときに saves as ダイアログボックスの **Create XML file** ボックスをオンにします。これにより、グラフの XML ファイルが作成されます。XML ファイルは、**.cht** 形式ではなく、**.tpl** 形式の表現であることに注意してください。

5.7.2.3 ビューアページの作成

指定するもう 1 つのオプションは、チャートを表示する HTML ページを作成するかどうかです。HTML ページには、チャートビューアアプレットが含まれており、エンドユーザはチャートを表示して操作することができます。Chart Viewer の機能については、Chart Viewer を参照してください。

グラフを保存するときに HTML ページを作成するには、グラフを保存するときに save as ダイアログボックスの **Create HTML File** ボックスをオンにします。これにより、EspressReport インストールの **html**//ディレクトリの下にあるチャートファイルと同じ名前の HTML ページが作成されます。ファイルが書き込まれる前に、使用したい Viewer のバージョンを選択するように求められます。



Select Viewer ダイアログ

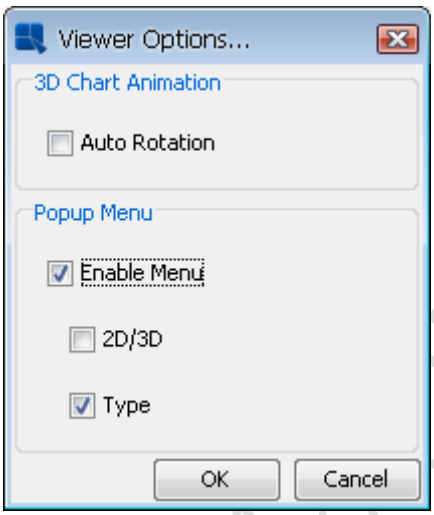
EspressReport は、AWT とスイングバージョンのビューアをサポートしています。ビューアを使用する場合は、クライアントに Java プラグインが必要であることに注意してください。

HTML ページを作成したら、ブラウザでページを参照してチャートを表示できます。グラフが表示されるようにするには、
HTML ページの vial http プロトコルをロードする必要があります。ファイルプロトコル上で読み込む(ファイルを参照して開く)ことはできません。つまり、EspressReport は Web サーバにインストールされている必要があり、**http://machinename:port/EspressChart/yourfile.html** を介してページをアクセスして正しく読み込む必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.7.2.3.1 ビューアオプション

チャートデザイナ内から構成できるいくつかのチャートビューアオプションがあります。これらはチャートとともに保存されるプロパティで、チャートがビューアに表示されたときに表示されます。ビューアオプションを変更するには、**Format** > **Viewer Options** を選択します。これにより以下のダイアログが表示されます。



Viewer Options ダイアログ

以下のオプションを使用できます。

自動回転
これにより、3 次元チャートの自動回転が可能になります。アプレットがロードされると、チャートが回転し始めます。ナビゲーションは、ナビゲーションパネルの適切なボタンを使用して停止することができます。

メニューを有効にする
これにより、ユーザーが Chart Viewer でチャートヲ表示しているときにポップアップメニューをオンまたはオフにすることができます。

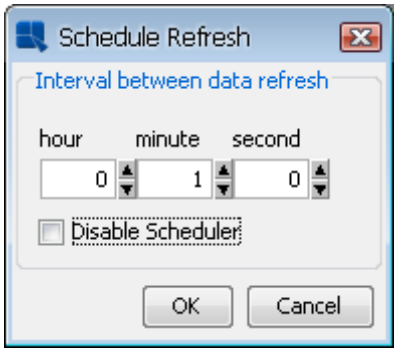
2D/3D
これにより、ポップアップメニューの寸法トグルをオンまたはオフにすることができます。これをオフにすると、ユーザーはグラフのディメンションを変更できなくなります。

タイプ
これにより、ポップアップメニューのタイプサブメニューをオンまたはオフにすることができます。これをオフにすると、ユーザーはグラフの種類を変更できなくなります。

.cht ファイルの場合は、スケジュール更新を設定することもできます。これにより、チャートがチャートビューアで表示されているときにチャートが定期的にデータをリフレッシュすることができます。リ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フレッシュレートをスケジュールするには、**Data** > **Schedule refresh** を選択します。ダイアログが表示され、スケジュールオプションを設定できます。



Schedule Refresh ダイアログ

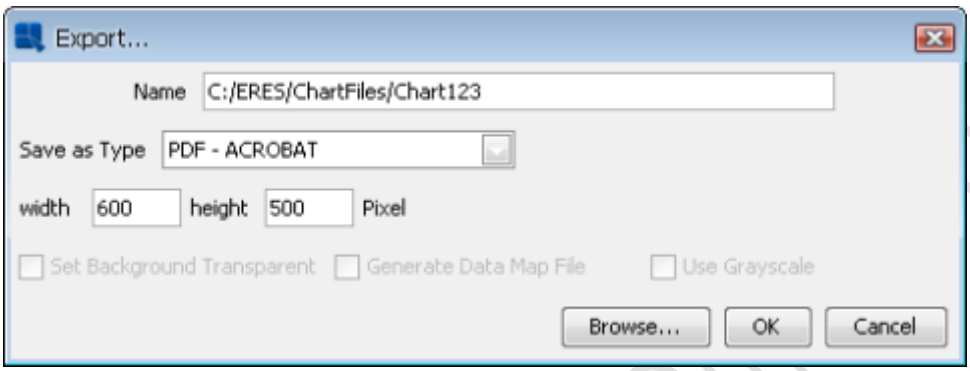
このダイアログから、リフレッシュ間隔の時、分、秒を設定できます。**Disable Scheduler** チェックボックスをオンにすると、スケジューラがオフになります。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.7.3 チャートのエクスポート

チャートデザイナから、あなたのチャートのいくつかの静止画のエクスポートを生成することができます。現在のグラフをエクスポートするには、**File** > **Export** を選択するか、ツールバーの **Export** ボタン をクリックします。これにより、生成されたファイルのオプションを指定できるダイアログが表示されます。



Export Graph ダイアログ

最初のオプションでは、生成されたファイルの名前とファイルパスを指定できます。ダイアログの下部にある **Browse** ボタンをクリックすると、適切なファイルパスを参照できます。2 番目のオプションではどのタイプのエクスポートを希望するかを選択できます。以下のオプションを使用できます。

GIF

EspressReport は GIF 画像を生成することができます。GIF には画像のファイルサイズを小さく保つ 256 の色制限があります。

JPEG

JPEG のほかのポピュラーな画像フォーマットです。これは GIF よりも高解像度の画像フォーマットであり、特許保護されていません。JPEG ファイルを生成するときに、ファイルの品質・圧縮を指定できます。品質が高いほど、ファイルは大きくなります。

エクスポートフォーマットとして JPEG を選択した場合、OK をクリックすると、画質を指定するよう求められます。選択した画質が高いほど、生成されるファイルサイズは大きくなります。JPEG エクスポートでは、低品質の結果が望ましくない可能性が高いため、高品質の画像を指定することをお勧めします。

PNG

PNG はあまり一般的ではありませんが、ほとんどのブラウザで表示できる画像形式です。JPEG よりもファイルサイズが小さい高画質の画像です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

SVG

SVG(スケータブルベクターグラフィックス)は比較的新しい画像フォーマットで、XML ベースのテキスト形式でベクトルとして画像を保存します。一般的にこれらの画像を表示するには、ブラウザプラグインが必要です。

SWF

SWF は Adobe Flash ファイルです。フラッシュフォーマットはベクトルベースであり、エクスポート後にチャートのサイズを変更することができます。また、フラッシュは高解像度の印刷を可能にし、小さなファイルサイズを生成する。このエクスポートタイプを選択するときは、フレームカウント(アニメーションのフレーム数)とフレームレート(1 秒当たりのフレーム数)を指定するか、アニメーションを無効にすることもできます。

BMP

これは、Windows のビットマップ形式です。

WMF

WMF は Windows メタファイル形式です。これは、MS Office ドキュメントへのインポート・エクスポートに使用できます。

PDF

これにより、Adobe Portable Document Format でチャートが生成されます。チャートは、1 ページの PDF ドキュメントとして生成されます。

XML

これにより、チャートのデータを含む XML データファイルが生成されます。XML チャートの属性ファイルは生成されません。データは XML 形式のみに書き込まれます。

XLS

XLS(MS Excel)ファイルを生成し、チャートを画像として最初の XLS シートに挿入します。

TXT

これにより、チャートのデータを含むテキストデータファイルが生成されます。

また、チャートにハイパーリンクが含まれている場合、マップファイルを生成されえたイメージと共にエクスポートすることもできます。マップファイルには、生成チャートのリンクを含む HTML イメージマップが含まれています。地図ファイルは、画像ファイルと同じ場所に生成され、同じファイル名を持ちます。マップファイルを生成するには、エクスポートする前に **Generate Data Map File** ボックスをオンにします。

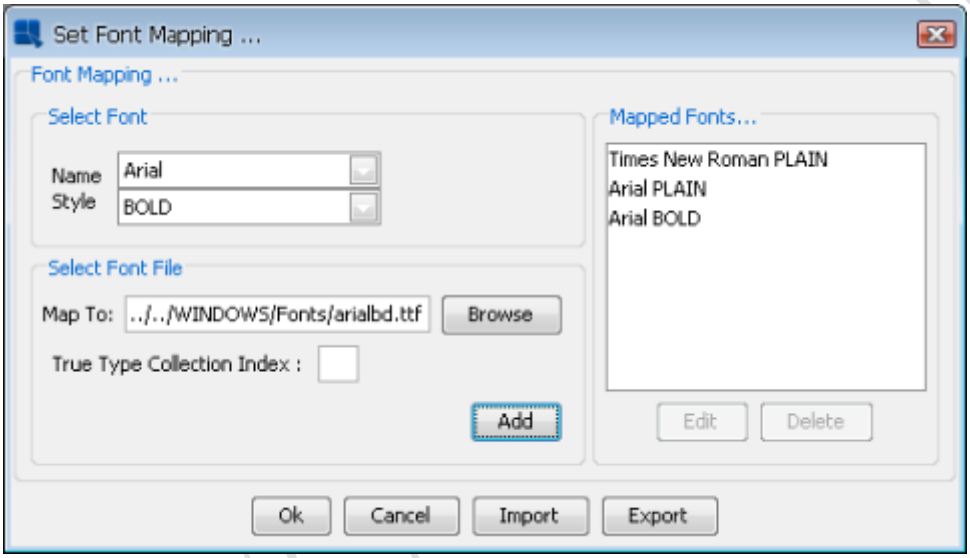
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.7.3.1 PDF フォントマッピング

レポートの場合と同様に、システム内の任意のフォントをチャート内のラベル、テキスト、およびタイトル用に使用できます。ほとんどのイメージのエクスポートでは、テキストがイメージに書き込まれ、追加の構成は必要ありません。ただし、PDF エクスポートの場合は、チャートで使用する任意のシステムフォントに対して、**.ttf**（真字型フォント）、**.ttc**（真型コレクション）、**.pfb**、または**.afm** ファイルを指定する必要があります。

PDF エクスポート尾のフォントマッピングを設定するには、**Format** > **Font Mapping** を選択します。これにより、フォントファイルを指定できるダイアログが表示されます。



Font Mapping ダイアログ

フォントとスタイルの組み合わせごとに、そのフォントの特定の**.ttf**、**.ttc**、**.pfb**、または**.afm** ファイルを選択することができます。フルパスを入力するか、フォントファイルを参照することができます。**.ttc** ファイルを使用している場合は、指定されたボックスにフォントインデックスを指定する必要があります(**.ttc** ファイルには複数のフォントが含まれています)。正しいファイルを指定したら、**Add** ボタンをクリックしてマッピングをリストに保存します。既存のマッピングを編集または削除するには、リストでそれらのマッピングを選択し、適切なボタンをクリックします。

チャートを単独でエクスポートするときは、フォントマッピングを適用する必要があります。
グラフがレポート内にある場合は、レポートに定義されているフォントマッピングを継承します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.7.3.1.1 PDF フォントマッピングのインポート/エクスポート

インポート・エクスポート機能を使用して、あるチャートから別のチャートへのフォントマッピングを渡すことができます。フォントマッピングをエクスポートするには、フォントマッピングダイアログの **Export** ボタンをクリックします。これにより、ファイル名を指定するダイアログボックスが表示されます。フォントマッピングは XML ファイルとして保存されます。ダイアログから **Import** オプションを選択すると、フォントマッピング XML ファイルを読み込むことができます。これにより、インポートする XML ファイルを指定するダイアログボックスが表示されます。**OK** をクリックすると、XML ファイルに格納されているマッピングが現在のチャートに適用されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.8 チャートビューア

Chart Designer で作成されたすべてのチャートは、ドキュメントに貼り付けることのできるファイル形式の範囲で保存できます。これらの形式には、BMP、JPG、PNG、PDF、SVG、SWF、WMF、GIF などがあります。さらに、Chart Designer には、グラフを**.cht**、**.tpl**、または**.xml** ファイルとして保存するオプションが用意されています。

Chart Viewer は、Web ブラウザを使用してチャートを動的に表示および操作できるアプレットです。ビューアーは、Chart Designer または API によって出力されたファイル（.cht または.tpl 形式）を読み取り、グラフを表示します。データファイルのサイズが小さいため、チャートイメージを Web 上に配布するのに適しています。データは EspressManager から Chart Viewer に転送される間に暗号化されるので、機密情報にある程度のセキュリティが確保されます。

Chart Viewer アプレットを使用して、ユーザーが**.cht** ファイルをインタラクティブに表示することができます。チャートビューアを使用すると、基礎となるデータを変更することなく、チャートを表示および操作できます。

.tpl 形式のファイルは、Chart Viewer を使用して表示することもできます。.tpl ファイルを含む Web ページをブラウザで表示すると、Chart Viewer によって新しいデータが自動的に取得されます。したがって、1 つのチャートテンプレートを使用して、最新のチャートをリアルタイムでユーザーに提供することができます。

チャートビューア内で、チャート、凡例、タイトル、またはラベルをドラッグして、オブジェクトの位置を調整できます。チャートのサイズを変更し、データポイントまたは一連のデータをドリルダウンすることもできます。 3 次元チャートでは、ナビゲーションパネルを使用して、パン、ズーム、各方向への回転、および翻訳を行うことができます。また、個々の x 軸、y 軸、z 軸のスケーリング、厚さ比率の調整、リアルタイムの 3 次元アニメーションなどはすべて簡単に実行できます。ユーザーがデータ要素をクリックして基礎となるデータを表示したり、関連する URL にジャンプしたりできるようにする組み込みのコールバックメカニズムがあります。Chart Viewer は、Java をサポートするすべてのプラットフォーム上で実行される純粋な Java で書かれています。また、Chart Viewer はスケジュールされたリフレッシュ（デザイナーが指定した一定の間隔でチャートのデータが更新される）と、ロード時にチャートのパラメータが提供されるパラメータサービングをサポートします。

以下の構文を使用して、**.cht** または**.tpl** 形式のファイルを Web ページに埋め込むことができます。

```
<applet codebase = ".." code = "quadbase.chartviewer.Viewer.class"width = 640 height = 480 archive = "EspressViewer.jar">

    <PARAM name = "filename" value = "yourchart.cht">

</applet>
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

パラメータ **filename** は、チャートデータを含むファイルのファイル名を指定し、リモートデータファイルにアクセスするために **http://**でプレフィックスすることができます。Chart Viewer で表示すると、グラフ形式（**.cht**）で保存されたグラフは、そのファイルに格納されているデータをプロット用に使用します。

テンプレート形式（**.tpl**）で保存されたグラフを使用すると、Chart Designer を使用してテンプレートを作成するときにグラフのデータソースを指定する場所（データベース名、ユーザー名、パスワードなどはすべて**.tpl** ファイルに保存されます)。

5.8.1 チャートビューアのパラメータ

また、チャートデザイナを使用せずに、データをチャートビューアプレットに渡すこともできます。 つまり、Chart Viewer を使用して、データファイルを直接表示したり、HTML コード内の他の制御情報とともに、データ行の形式でデータを直接渡すことができます。デフォルトでは、パラメータが真/偽の型の場合は true になります。以下にパラメータの一覧を示します。

チャートパラメータ（2 次元および 3 次元チャートに共通）

mainTitle：チャートのメインタイトル

xTitle：X 軸タイトル

yTitle：y 軸のタイトル

zTitle：z 軸タイトル

RefreshInterval：スケジュールされたリフレッシュ間隔（秒単位）

DragLegend：false の場合、凡例は移動できません

DragChart：false の場合、グラフの位置を変更またはサイズ変更することはできません。

ShowDataHint：false の場合、チャートデータを左クリックしたときにデータ情報ボックスは表示されません

ShowLinkHint：false の場合、チャートデータを右クリックするとハイパーリンク情報ボックスは表示されません

DataHintBgColor：データ情報ボックスの背景色を設定する

LinkHintBgColor：ハイパーリンク情報ボックスの背景色を設定する

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

DataHintFontColor：データ情報ボックスのフォント色を設定する

LinkHintFontColor：ハイパーリンク情報ボックスのフォント色を設定する

DataHintFont：データ情報ボックスのフォントを設定する

LinkHintFont：リンク情報ボックスのフォントを設定する

DataHintOffsetX：データ情報ボックスの x オフセットを設定する

DataHintOffsetY：データ情報ボックスの y オフセットを設定する

LinkHintOffsetX：リンク情報ボックスの x オフセットを設定する

LinkHintOffsetY：リンク情報ボックスの y オフセットを設定する

Printing：false の場合、ブラウザで（Ctrl + P および/または Ctrl + J を使用して）グラフをエクスポートする機能が無効になります

filename：チャートに適用されるテンプレートファイルの名前

xAxisRuler：true の場合は、x 軸ルーラーを表示します（2D グラフの場合のみ）。

yAxisRuler：true の場合は、y 軸ルーラーを表示します（2D チャートの場合のみ）

sAxisRuler：true の場合、セカンダリ軸ルーラを表示します（2D グラフの場合のみ）

ResizeChart：false の場合、グラフのサイズを変更することはできません。

ResizeCanvas：false の場合、キャンバスのサイズを変更することはできません。

comm_protocol：ファイアウォールの場合に使用されるプロトコル

comm_url：ファイアウォールの場合、EspressManager に接続する URL

RefreshData：false の場合、チャートデータはリフレッシュできません。

PopupMenu：false の場合、ポップアップメニューは表示されません。

TypeMenu：false の場合、タイプサブメニューはポップアップメニューに表示されません

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

DimensionMenu：false の場合、ディメンションサブメニューはポップアップメニューに表示されません

3 次元グラフのみ

Toggle3Dpanel：false の場合、ナビゲーションパネルを表示または非表示に切り替えることはできません

Drawmode：3D チャートを描画する異なるモードを設定します。使用可能な描画モードは、フラット（デフォルト）、**WireFrame**、**Flat Border**（フラットシェーディングモデルの周囲に黒い境界線を描く）、**Gouraud**、**Gouraud Border**

NavColor：ナビゲーションパネルの色を設定する

navpanel：false の場合、3D チャートが表示されているときにナビゲーションパネルは表示されません。2D チャートを表示しているときにナビゲーションパネルが表示されることはありません

GouraudButton：false の場合、ナビゲーションパネルの Gouraud シェーディングボタンは非表示になります

AnimateButton：false の場合、ナビゲーションパネルのアニメーション速度コントロールは非表示になります

SpeedControlButton：false の場合、ナビゲーションパネルの速度コントロールボタンは非表示になります

データ入力パラメータ

sourceDB：チャートを生成するためにデータベース情報を設定する

sourceData：チャートを生成するためにデータ情報を設定する

sourceFile：グラフを生成するためにデータファイル情報を設定する

datamap：チャートの列マッピングを設定する

TransposeData：チャートを生成するために使用する前に変換されるようにデータを設定する

chartType：生成されたチャートのチャートタイプを設定する

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

EspressManagerUsed：使用する EspressManager を設定する

ParameterServer：チャート内のデータを動的に更新する

transposeData：true の場合は、データを変換する

server_address：Espress Manager 接続の IP アドレス。

server_port_number：Espress Manager 接続のポート番号。

パラメータ **mainTitle**、**xTitle**、**yTitle**、および **zTitle** は、チャートのメインタイトルと軸タイトルを指定するために使用され、テンプレートで定義されているタイトルと軸のタイトルを上書きします。

<PARAM name="mainTitle" value="This is the main Title">
<PARAM name="xTitle" value="x axis title">
<PARAM name="yTitle" value="y axis title">
<PARAM name="zTitle" value="z axis title">

パラメータ **RefreshInterval** を使用して、以下を指定できます。

<PARAM name="RefreshInterval" value="60">

上記の例では、アプレットは（EspressManager の助けを借りて）データベースまたはデータファイルからデータをフェッチし、60 秒ごとにグラフを透明にすべて再描画します。データが頻繁に変更されるデータベースにアクセスする場合に便利です。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.8.2 チャートビューアのデータソースの指定

パラメータを使用すると、チャートテンプレートで異なるデータを表示したり、最初からチャートを作成したりするために、チャートビューアのデータソースを指定できます。

5.8.2.1 データベースから読み取ったデータ

これは、チャートビューアを使用して、データベースから抽出されたデータを使用して描画されたグラフを表示するサンプル HTML コードです。

```
<applet code = "quadbase.chartviewer.Viewer.class" width=640 height=480>

    <PARAM name="sourceDB" value="jdbc:odbc:DataSource, sun.jdbc.odbc.JdbcOdbcDriver, username, password, select * from products">
    <PARAM name="dataMap" value="0 1 -1 3">
    <PARAM name="chartType" value="3D Column">

</applet>
```

dataMap の引数は、チャートが入力データの異なる鉄を使用してグラフをプロットする方法を指定します。散布図の場合は、series、x-value、y-value、z-value です。高いローオープンクローズまたはローローチャートの場合、数値はシリーズ、category、high、low、open、close です。他のすべてのチャートでは、引数は series、category、sumBy、および value です。詳細については、Chart API リファレンスの Column Mapping に関しての章を参照してください。引数 chartType は、ひょじされるチャートのタイプを指定し、以下のいずれかになります。

- 2D Column
- 3D Column
- 2D bar
- 3D bar
- 2D stack bar
- 3D stack bar
- 2D stack column
- 3D stack column

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- 2D area
- 3D area
- 2D stack area
- 3D stack area
- 2D line
- 3D line
- 2D pie
- 3D pie
- 2D scatter
- 3D scatter
- 2D High Low
- 3D High Low
- 2D HLCO
- 3D HLCO
- 2D 100% Column
- 3D 100% Column
- 3D Surface
- 2D Bubble
- 2D Overlay
- 2D Box
- 2D Radar

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- 2D Dial
- 2D Gantt
- 2D Polar

5.8.2.2 データファイルから読み取ったデータ

この HTML コードは、データファイルのデータを使用してグラフを描画します。

```
<applet code = "quadbase.chartviewer.Viewer.class" width=640 height=480>

        <PARAM name="sourceFile" value="http://.../test.dat">
        <PARAM name="dataMap" value="-1 0 -1 1">
        <PARAM name="chartType" value="3D Pie">

</applet>
```

5.8.2.3 引数から読み取ったデータ

チャートビューアをデータファイルまたはデータベースからではなく、HTML ファイルから直接読み込むことができます。

```
<applet code = "quadbase.chartviewer.Viewer.class" width=640 height=480>

        <PARAM name="sourceData" value="int, string, int | value, name, vol | 10, 'John', 20 | 3, 'Mary', 30 | 8, 'Kevin', 3 | 9, 'James', 22">
        <PARAM name="dataMap" value="-1 1 -1 2">
        <PARAM name="chartType" value="3D Bar">

</applet>
```

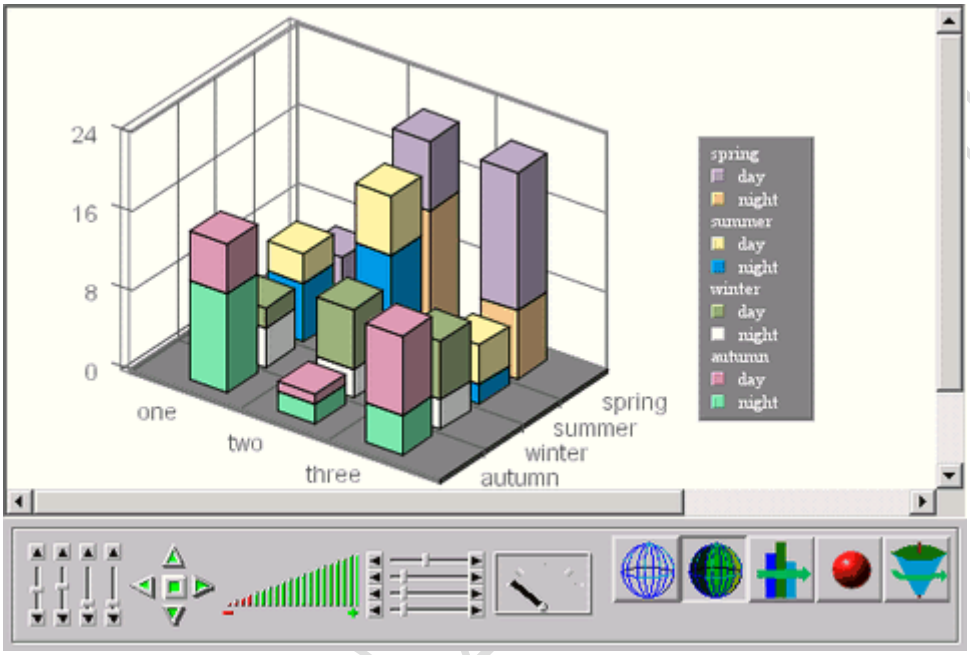
パラメータ **sourceData** の形式は、データファイル形式と同じですが、各行は垂直バー”|”で終わる点が異なります。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.8.3 チャートビューアの使用

チャートビューアでマウスを使用するだけで、チャートの外観を操作できます。3 次元グラフの場合、チャートビューア画面は、**drawing panel** と **navigation panel** の 2 つのセクションで構成されています。2 次元グラフでは、**drawing panel** のみが表示されます。描画パネル内では、チャート自体がプロット領域に配置されています。



サンプルチャートビューアの表示

以下は、チャートビューアを使用しているときに可能なチャート操作のリストです（順序または重要度ではありません）。

- ナビゲーションパネルを使用して、以下のことを行うことができます。
  - ライトポジションを変更する
  - チャートを回転する
  - チャートを翻訳する
  - チャートの拡大/縮小
  - x、y、z 次元でグラフをスケールする
  - バー/ライン/ポイント/パイの太さ比を調整する
  - チャートアニメーションを開始し、アニメーションの速度を制御する

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

- ワイヤーフレームモードとソリッドモードの切り替え
- グラフのすべての辺に黒いアウトラインを描く
- グーローシェーディングを実行する

- チャート、メインタイトル、x、y、z ラベル、または凡例をマウスの左ボタンでドラッグして項目を移動する
- プロット領域の上でマウスの右ボタンをドラッグすると、グラフのサイズが変更されます
- **Alt** キーを押しながらプロットエリアの上でマウスの右ボタンをドラッグすると、キャンバスのサイズが変更されます
- ナビゲーションパネル（表示と非表示）を切り替えるには、描画パネル上でマウスの左ボタンを**ダブルクリック**します
- マウスを使用すると、データポイントを個別に照会することもできます
- データ点を**左クリックす**ると、その点に関連付けられたデータを表示します。
- データを**右クリック**すると、データポイントに関連付けられたハイパーリンクの名前が提供されます
- **左のダブルクリック**すると、データポイントに関連付けられているハイパーリンクにジャンプします。（これにより、新しいブラウザウィンドウが作成されることに注意してください。ユーザは、ブラウザの **Back** ボタンを使用して前のチャートに戻ることはできません）
- **右のダブルクリック**で、前のチャートに戻ります（該当する場合）
- **Alt** + **Z** を使用してズームインパラメータを指定する
- 手動でデータを更新するには、**Ctrl** + **R** を使用します。
- **Ctrl** +**左クリック**で拡大する境界を選択し、**Ctrl** +**右をクリック**して縮小します

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.8.4 軸ルーラー

チャートビューアに軸ルーラーを表示するオプションが追加されました(アプレットタグにパラメータ{**axisRuler**}を設定することにより)。Chart API（API ドキュメント quadbase.util.IAxisRuler を参照してください)。これにより、スクロールが有効になっているときに参照点が提供され、グラフの軸が表示されないようにチャートが移動されます。この機能は 2D チャートのみに適用されます。

5.8.5 パラメータサーバ

パラメータサーバを使用すると、チャートビューアは TCP/IP を使用して指定されたポートからの要求をリッスンし、データを動的に更新できます。構文は以下の通りです。

```
<applet code = "quadbase.chartviewer.Viewer.class" width=640 height=480>
        <PARAM name="ParameterServer" value="machine:portno">
</applet>
```

セキュリティ上の理由からマシンは通常、Web サーバマシンと同じです。したがって、マシンを空のままにすることができます（**value = ":portno"**)。

パラメータサーバの詳細については、パラメータサーバを参照してください。

5.8.6 ポップアップメニュー

チャートの作成・編集中に、ポップアップメニューオプションを選択した場合は、チャートを変更できます。メニューは、チャートを右クリックすると開きます。そこにあるオプションは、強調表示して左クリックすることで選択できます。使用可能なオプションは、動的データドリルダウン、チャートタイプの変更、軸ズーミング、時系列データズームなどがあります。

5.8.6.1 チャートの次元/種類の変更

チャートディメンションとチャートタイプは、ポップアップメニューを使用して **2D/3D** オプションまたは **Type** オプションを選択することによって変更できます。

チャートが 3 次元の場合、**Overlay**、**Box**、および **Dial** チャートオプションは表示されません。

5.8.6.2 軸ズーム

ビューポートのサイズを超える大きなチャートの場合（チャートが十分に大きくてウィンドウ内に完全に見えない場合)、軸のズームを可能にするオプションがあります。これにより、すべての軸を移動し、軸をズームイン・ズームアウトができます。

このオプションは、2 次元チャートのみ使用できます。バブル、ダイアル、パイ、極座標、レーダー、散布図、スタック領域、および垂直ボックスチャートは、この機能をサポートしていません。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

このオプションを有効にするには、ポップメニューを開き、**Enable Zoom** を選択します。x 軸、y 軸、またはその両方のズームインから選択できます。

ズームインするには、目的のエリアを左クリックしてドラッグします。y 軸ズームのみを選択した場合は、x 軸の長さや位置を気にする必要はありません。
スクロールはズームされた軸に対してのみ有効です。軸のズームを有効にすると、特定の軸にスクロールバーが表示されます。このスクロールバーを左クリックしてドラッグすると、アプレットのビューポートを移動できます。

左クリックしながらコントロールキーを押し続けると、アプレットは前のズームに戻ります。以前のズームは 1 つだけメモリに保存されているため、1 ステップ前に戻ることができます。デフォルトの x と y スケールに戻すには、キーボードの **Home** キーを押します。

5.8.6.3 ズーム

表示されているチャートがズーム可能なチャートの場合、ズームオプションはポップアップメニューから利用できます。ズームオプションを使用すると、カテゴリ要素をユーザ定義の間隔にグループ化し、各グループのポイントを集約することができます。日付・時刻に基づくズームの詳細は、日付・時刻ベースズームを参照してください。ズームを有効にすると、加減、上限などのさまざまなズームオプションを入力でき（境界を無効にして、最初のデータポイントから最後のデータポイントに移動することもできます）、タイムスケール、および x 軸を線形にするかどうかを指定します。

5.8.6.4 動的データドリルダウン

ポップアップメニューからドリルダウンを選択すると、次の **drill-down** を設定できます。このオプションは、グラフに対して動的データドリルダウンが有効な場合にのみ使用できます。ポップアップメニューの **Drill-Down** オプションを選択すると、次のドリルダウンチャートの現在の設定を表示できます。**Category** が **None** の場合、次のレベルはまだ設定されていません。グラフを生成するために使用されるデータに未使用の列がある場合は、その列の列を選択できます。**Category** が選択されると、次のレベルのチャート **Type** を（**Series** と **SumBy** と共に）設定することができます。データポイントをクリックしてレベルを下に移動し、右クリックしてレベルを上げることができます。

5.8.6.5 クエリパラメータ

パラメータ化されたグラフを表示するときは、ポップメニューを開き、**Query Parameter** オプションを選択することで、パラメータ（または数値に応じたパラメータ）に異なる値を入力できます。次に、選択されたパラメータに基づいて、新しいデータに再描画されます。

5.8.7 利用可能な Swing バージョン

Chart Viewer の JFC/Swing バージョンは、**EspressReport/lib** ディレクトリの **EspressViewer.jar** の代わりに **SwingEspressViewer.jar** を参照するためにも使用できます。Swing ビューアを使用するときは、次のクラスを呼び出します：**quadbase.chartviewer.swing.Viewer.class**。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.9 EspressReport Chart API

5.9.1 概要およびセットアップ

EspressReport デザイナーでチャートテンプレートを設計し作成することに加えて、EspressReport は使いやすいアプリケーションプログラミングインタフェース（API）を提供し、ユーザーはアプリケーションやアプレット内で 2D チャートや 3D チャートを作成しカスタマイズすることができます。これは 100%Pure Java で書かれているため、ほとんどまたはまったく修正を必要としない任意のプラットフォーム上で実行できます。チャートテンプレートは、API を使用して完全にカスタマイズできますわずか 4 行のコードを使用して、別のグラフにグラフを追加することができます。EspressReport デザイナーでチャートテンプレートを設計し作成することに加えて、EspressReport は使いやすいアプリケーションプログラミングインタフェース（API）を提供し、ユーザーはアプリケーションやアプレット内で 2D チャートや 3D チャートを作成しカスタマイズすることができます。これは 100%Pure Java で書かれているため、ほとんどまたはまったく修正を必要としない任意のプラットフォーム上で実行できます。チャートテンプレートは、API を使用して完全にカスタマイズできます。わずか 4 行のコードを使用して、別のグラフにグラフを追加することができます。

チャートのデータは、2 つのソースから取得できます。

- データは、チャートテンプレートが設定されているレポートセクションから取得できます。たとえば、レポートのフッターにはグラフテンプレートがあり、レポートのデータ全体が表示されます。ただし、同じテンプレートがグループフッターに割り当てられている場合は、グループ内の部分データのみがグラフに表示されます。したがって、チャートに正しいデータが表示され、グラフテンプレートがレポートの適切な場所に設定されていることを確認する必要があります。
- データは、独立したデータソースから取得できます。このデータソースは、レポートの作成に使用されたデータソースと同じでも、データソースとは異なるデータソースでもかまいません。チャートデータは、レポート内の配置に応じて変化しないことに注意してください。ただし、チャートが追加されたレポートのセクションによっては、複数のチャートが表示されることがあります。

レポートにチャートを追加するだけでなく、レポートとは独立してスタンドアロンチャートを生成してデプロイすることもできます。グラフを表示するために、レポートにグラフを追加する必要はありません（空白の場合も同様）。

メインクラス **QbChart** と **ChartObject**（**quadbase.ChartAPI.QbChart** を拡張）は、チャートオブジェクトを作成してレポートに追加するために使用されます。このコンポーネントには、**quadbase.ChartAPI** と **quadbase.util** の 2 つのパッケージに含まれる一連の補助クラスがあります。このドキュメントの残りの部分では、API の構成要素とその使用方法について説明します。

完全な API ドキュメントは、こちらにあります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

API を使用するには、**EspressReport/lib** ディレクトリにある **EspressAPI.jar** と **ExportLib.jar** を CLASSPATH に追加します。 XML を使用している場合（データの出力またはデータの読み込みを行う場合)、**AxercesImpl.jar** および **xml-apis.jar**（同じディレクトリにもあります）を CLASSPATH に追加する必要があります。チャートを SVG または Flash にエクスポートする場合は、SVGExport.jar または FlashExport.jar（同じディレクトリにもあります）を CLASSPATH に追加する必要があります。パラメータ化されたデータベースクエリをデータソースとして使用する場合は、**jsqlparser.jar** を CLASSPATH に追加します。CLASSPATH に **qblicense.jar** も含める必要があります。アプリケーションが Windows または Solaris マシン上にある場合、（プラットフォームに応じて）次の環境変数を追加する必要があります。

Windows の場合

```
set PATH=%PATH%;<path to EspressReport root directory>¥lib
```

Solaris の場合

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<path to EspressReport root directory>/lib
```

5.9.2 Chart API を使用するために推奨されるアプローチ

EspressReport は、プログラムによるスクラッチからチャートを生成する API を提供します。しかし、これには大量のコードが含まれています。Designer を使用し、チャートテンプレート（**.TPL** ファイル）を作成することをお勧めします。チャートテンプレートは、**QbChart** コンストラクタで渡すことができ、そのため、新しく作成されたチャートオブジェクトに外観が適用されます。したがって、外観のほとんどは、チャート生成時に設定されます。プロパティを変更、追加、または削除するコードを記述することができます。この方法は、コーディング時間を節約し、パフォーマンスを向上させます。

EspressReport Chart API を使用する場合、EspressManager に接続したり、接続したりすることができます。 Designer を使用する場合は接続が必要ですが、EspressReport Chart API を使用するアプリケーションを実行する場合は、EspressManager に接続する必要はありません。EspressManager に接続せず、アーキテクチャ内の別のレイヤーを避けることをお勧めします。詳細については、次のセクションを参照してください。

マニュアルに記載されているすべての例とコードは、テンプレートベースのアプローチと EspressManager への接続の 2 つの推奨事項に従います。特に明記しない限り、すべてのコード例でテンプレートを使用し（対応する章でダウンロードできます)、EspressManager に接続しません。

また、EspressReport Chart API を使用するアプレットを使用する場合、ブラウザには少なくとも 1.5 JVM プラグインが必要です。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.9.3 EspressManager とのインタラクション

EspressReport は、一般的に EspressManager と組み合わせて使用されます。チャートコンポーネントは、ファイルの読み書き、データベースへのアクセス、特定の高度な機能（集計など）に必要なデータ前処理を実行するために、EspressManager に接続します。ただし、チャートコンポーネントは、スタンドアロンモードでも使用できます。スタンドアロンモードでは、EspressManager を使用せずにファイル I/O とデータベースアクセスを直接実行します。

両方の方法にはそれぞれ独自の利点があります。チャートがアプレット内に含まれている場合、セキュリティの制限によりファイルの入出力を直接実行できなくなります。クライアント側に JDBC ドライバがなければ、データベースへのアクセスも困難です。そのような場合、EspressManager は上記のサービスをチャートに提供します。一方、チャートがアプリケーションで使用される場合、そのような制限はなく、直接アクセスによってパフォーマンスを向上させることができます。EspressManager をよく知られた場所で実行する必要なく、アプリケーションを実行できます。

たとえば、アプレットまたはアプリケーションがマシン**クライアント**上で実行されている可能性があり、マシン **DbMachine** 上のデータベースからのデータが必要な場合があります。ただし、マシン **DbMachine** はファイアウォールまたは直接接続の背後にある可能性があり、セキュリティ制限のためにマシンの**クライアント**から **DbMachine** をマシン化することが許可されないことがあります。EspressManager はマシン**サーバ**上で実行でき、アプレット/アプリケーションはマシン**サーバ**上の EspressManager に接続できます。JDBC を使用してマシン**サーバ**からマシン **DbMachine** に接続し、データを取得できます。次に、データがマシン**クライアント**に渡され、チャートが生成されます。これは、クライアントマシンからデータを保護し、アクセスできないようにし、すべての接続をサーバマシン（EspressManager を実行しているマシン）を経由させたい場合に便利です。このオプションを利用して、すべてのクライアントのログに EspressReport を介してデータを保存することもできます（EspressManager の起動時にログファイルを作成することができます。ログファイルは espressmanager.log です）。この機能にはコストがかかります。コードが EspressManager を介してデータに接続している（つまり、別のレイヤーが追加されている）ため、少しオーバーヘッドになります。

デフォルトでは、チャートコンポーネントには EspressManager が必要です。モードを変更するには、（**QbChart** オブジェクトが作成される前に）アプレット/アプリケーションの先頭で **QbChart** クラスの静的メソッドを使用します。

```
public void setEspressManagerUsed（boolean b）
```

アプリケーションとアプレットの両方を、EspressManager にアクセスするかどうかにかかわらず実行できます。通信は http プロトコルを使用して行われます。サーバの場所は、API コードで渡される IP アドレスとポート番号によって決まります。以下は EspressManager への接続方法です：

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.9.4 EspressManager に接続する

5.9.4.1 アプリケーションとして実行する EspressManager

EspressManager は主にアプリケーションとして実行されます。 ChartAPI を使用してアプリケーションとして動作する EspressManager に接続する場合は、API メソッドを使用して、EspressManager が存在する IP アドレスまたはマシン名、および EspressManager がリッスンしているポート番号を指定できます。

接続情報を設定するには、次の 2 つの API メソッドを使用します。

```
static void setServerAddress（java.lang.String address）；
static void setServerPortNumber（int port）；
```

たとえば、次のコード行があります。

```
QbChart.setServerAddress（"someMachine"）；
QbChart.setServerPortNumber（somePortNumber）；
```

someMachine で動作し、**somePortNumber** でリッスンしている EspressManager に接続します。

EspressManager の接続情報が指定されていない場合、コードはローカルマシン上の EspressManager に接続し、デフォルトポート番号（22071）を聞き取ります。

これらのメソッドは、**QbReport**、**QbChart**、**QbReportDesigner**、および **QbScheduler** クラスに存在します。

5.9.4.2 サーブレットとして実行する EspressManager

EspressManager はサーブレットとしても実行できます。Chart API を使用してサーブレットとして動作する EspressManager に接続する場合は、次のメソッドを使用する必要があります。

```
public static void useServlet（boolean）；
public static void setServletRunner（文字列 comm_url）；
public static void setServletContext（文字列コンテキスト）
```

たとえば、次のコード行があります。

```
QbChart.useServlet（true）；
QbChart.setServletRunner（ "http://someMachine:somePortNumber"）；
QbChart.setServletContext（ "EspressReport/servlet"）；
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

こちらから動作する EspressManager に接続します。

これらのメソッドは、**QbReport**、**QbChart**、**QbReportDesigner**、および **QbScheduler** クラスに存在します。

5.9.5 API の使用

次のセクションでは、API の使用方法について詳しく説明します。ここでも、API の例とコードは上記の推奨事項(テンプレートベース/EspressReport サーバなし)を使用して設計されています。

特に記載のない限り、すべての例では、<**EspressReportInstall**> **/help/samples/DataSources/database** ディレクトリにある Woodview HSQL データベースを使用しています。例を実行するには、データベース HSQL JDBC ドライバ（**hsqldb.jar**）をクラスパスに追加する必要があります。

また、すべての API の例では、マニュアルの中核コードが表示されます。 例をコンパイルするには、CLASSPATH に **ReportAPIWithChart.jar** と **qblicense.jar** が含まれていることを確認してください。

API メソッドの詳細については、API のドキュメントを参照してください。ディレクトリにあります。

5.9.5.1 チャートの読み込み

チャートは、CHT または TPL 形式（どちらも独自の形式）を使用してファイルに保存できます。 TPL ファイルには、実際のデータを除くすべてのチャート情報が格納されます。 CHT ファイルには実際のデータも格納されます。これらの形式は、チャートオブジェクトを再構成するために使用できます。データは、TPL ファイルを開くたびに元のデータソースから自動的に再読み込みされます。 CHT ファイルを開くと、CHT ファイルを作成するために使用されたデータが表示されます（ただし、データソースからデータをフェッチするオプションもあります)。

デフォルトでは、TPL ファイルにはデータが含まれていないことに注意することが重要です。これには、チャートテンプレート情報（すなわち、チャートの外観)、指定されたデータソースが含まれる。したがって、TPL ファイルをロードするときに、データソースを照会することによってチャートのデータが取得されます。一方、CHT ファイルは、データソースが開かれたときにそのデータソースを照会しません。それは最初にチャートを作成するために使用されるデータを示します。両方の形式は、**QbChart** クラスで提供されている **Designer** または **export()**メソッドを使用して取得できます。

アプレットまたはアプリケーションとして実行できる次の例は、CHT ファイルを読み込んでチャートを再構成します：

```
//EspressManager は使用しない
QbChart.setEspressManagerUsed(false);

//テンプレートを開く
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        QbChart chart = new QbChart (parent, //container
                                "../templates/OpeningChartTemplate.cht"); //template

        //ビューアにレポートを表示

        return chart;

}
```

フルソースコード
エクスポート後の結果

このサンプルでのコンストラクタは次の通りです。

```
public QbChart(Applet applet, String file);
```

applet はアプレットコンテナで、file は CHT/TPL ファイル名です。

ファイル名は、URL 文字列または相対/絶対パスを使用して指定できます。パス名は、ERES サーバの現在のディレクトリ、または EspressReport Manager が使用されていない場合は現在のアプリケーションに相対的なものとして解釈されます。

5.9.5.1.1 オブジェクト配列

グラフには、データを何らかの形で制限するためのパラメータも含めることができます。通常、データソースによってクエリ（または IN）パラメータが使用され、制限されます。

クエリパラメータ単一値型と、複数値型の両方のパラメータ型にすることができます。

パラメータ化されたテンプレートが次のコンストラクタを使用して開かれた場合：

```
QbChart(Applet parent, String templateName);
```

ダイアログボックスが表示され、パラメータの値を要求します。このダイアログボックスは、テンプレートを開いたときに Designer に表示されるダイアログボックスと同じです。

オブジェクト配列

パラメータ化されたグラフは、任意の値を求めるダイアログを表示せずに開くこともできます。これは、オブジェクト配列を渡すことによって行うことができます。配列の各要素は、その特定のパラメータの値を表します。

配列の順序は、Designer でパラメータが作成された順序と一致する必要があります。正しい結果を得る

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

には、値のデータ型と一致する必要があります。

クエリパラメータは、複数値のパラメータタイプでもあります。多値パラメータの場合、ベクトルがオブジェクトと配列に渡されます。ベクトルには、多値パラメータの全ての値が含まれます。

アプレットまたはアプリケーションとして実行できる次の例は、チャートテンプレートを開くときにパラメータ値を渡します。

```
Component doObjectArray(Applet parent) {

        //EspressManager を使用しない
        QbChart.setEspressManagerUsed(false);

        //クエリパラメータのオブジェクト配列
        Vector vec = new Vector();
        vec.addElement("CA");
        vec.addElement("NY");

        //クエリパラメータのオブジェクト配列
        Object queryParams[] = new Object[3];
        queryParams[0] = vec;
        queryParams[1] = new Date(101, 0, 4);
        queryParams[2] = new Date(103, 01, 12);

        //テンプレートを開く
        QbChart chart = new QbChart (parent, //container
                        "../templates/ObjectArray.cht", //template
                                queryParams); //Query Parameters

        //ビューアにグラフを表示する「

        return chart;

}
```

フルソースコード
エクスポート後の結果

5.9.5.2 チャートテンプレートの適用

QbChart オブジェクトが最初から作成されると（チャートの作成を参照)、デフォルトの属性を使用してチャートが作成されます。ただし、グラフテンプレート（.tpl）を使用して、チャートの作成時にユ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ーザ定義の属性を指定することができます。ほとんどの属性（データソースとグラフタイプを除く）はテンプレートから抽出され、QbChart オブジェクトに適用されます。テンプレート名は通常、QbChart コンストラクタの最後の引数として表示されます。

また、**QbChart** クラスの **applyTemplate filename(String filename)**メソッドを使用してテンプレート名を指定することもできます。

以下の例は、アプレットまたはアプリケーションとして実行でき、**QbChart** オブジェクトにテンプレートを適用します。

```
Component doApplyingTemplate(Applet parent) {

        //EspressManager を使用しない
        QbChart.setEspressManagerUsed(false);

        String templateLocation = "..";

        //テンプレートを適用する
        QbChart chart = new QbChart (parent, //コンテナ
                        QbChart.VIEW2D, //グラフの寸法
                        QbChart.BAR, //チャートタイプ
                        data, //データ
                        columnMapping, //カラムマッピング
                        templateLocation); //template

        //Show chart in Viewer
        return chart;

}
```

フルソースコード
エクスポート後の結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

また、テンプレートから列マッピングを取得して、作成する **QbChart** オブジェクトに適用させることもできます。これは、**CollnfO** オブジェクトの代わりに **null** を渡すことによって行われます。

5.9.5.3 データソースの変更

Designer でレポートテンプレートを作成し、API を使用してそれらのテンプレートを開くことができま

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

す。作成された **QbChart** オブジェクトは、テンプレートと同じデータソースを使用し、データをフェッチしようとします。ただし、テンプレートの外観がすべて必要な場合でも、データソースが正しくない可能性があります。次のセクションでは、バックアップデータでテンプレートを開き、レポート全体を再作成せずにデータソースを切り替える方法を示します。

最良の結果を得るには、列数と各列のデータ型が、2 つのデータソース（つまり、Designer でテンプレートを作成するために使用されたものと新しいデータソース）の間で一致している必要があります。

データソースを切り替えた後、**QbChart** オブジェクトを強制的に新しいデータをフェッチする必要があります。これは、**QbChart** クラスの **refresh** メソッドを呼び出すことで実行できます。

5.9.5.3.1 データベースのデータ

データソースをデータベースに参照するように切り替えるのは簡単です。必要なのは、データベース接続情報と使用されるクエリの 2 つで、それを **QbChart** オブジェクトに渡すだけです。**DBInfo** オブジェクトにデータベース接続情報およびクエリを提供できます（**DBInfo** オブジェクトの作成の詳細については、データベースのデータを参照してください）。

以下の例は、アプレットまたはアプリケーションとして実行でき、**QbChart** オブジェクトのデータソースをデータベースに切り替えます。

```
Component doChartSwitchToDatabase(Applet parent) {

        //EspressManager を使用しない
        QbChart.setEspressManagerUsed(false);

        //テンプレートを開く
        QbChart chart = new QbChart (parent, //container
                        "SwitchToDatabase.cht"); //template

        //新しいデータベース接続情報
        DBInfo newDatabaseInfo = new DBInfo(.....);

        try {
                //データソースを切り替える
                chart.gethInputData().setDatabaseInfo(newDatabaseInfo);

                //グラフを更新する
                chart.refresh();

        } catch (Exception ex)
        {
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            ex.printStackTrace();

        }

        //ビューアにグラフを表示する
        return chart;

}
```

フルソースコード
エクスポート後の結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションが含まれています。

JNDI

データソースを JNDI データソースに変更することもできます。これは、**DBInfo** オブジェクト内の JNDI 接続情報を指定し、それを **QbChart** オブジェクトに渡すことによって行われます。

以下の例は、アプレットまたはアプリケーションとして実行でき、**QbChart** オブジェクトのデータソースを JNDI データベースに切り替えます：

```
Component doChartSwitchToDatabaseJNDI(Applet parent) {

        //EspressManager を使用しない
        QbChart.setEspressManagerUsed(false);

        //テンプレートを開く
        QbChart chart = new QbChart (parent, //container
                        "ChartSwitchToDatabaseJNDI.cht"); //template

        //新しいデータベース接続情報
        DBInfo newDatabaseInfo = new DBInfo(.....);

        try {
                //データソースを切り替える
                chart.gethInputData().setDatabaseInfo(newDatabaseInfo);

                //グラフを更新する
                chart.refresh();
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        } catch (Exception ex)
        {
                ex.printStackTrace();

        }

        //ビューアにグラフを表示する
        return chart;

}
```

フルソースコード
エクスポート後の結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。アプリケーションコードを実行するには、Woodview データベースを Tomcat 環境の JNDI データソースとして設定し、アプリケーションコードを接続情報に合わせて変更する必要があります。

5.9.5.3.2 データファイル（TXT/DAT/XML）のデータ

テキストファイルが Quadbase ガイドラインに従っている場合は、データソースをテキストファイルに切り替えることができます（詳細についてはデータファイル（TXT/DAT/XML）のデータを参照してください）。

アプレットおよびアプリケーションとして実行できる次の例は、**QbChart** オブジェクトのデータソースをテキストファイルに変更します：

```
Component doChartSwitchToDataFile(Applet parent) {

        //EspressManager を使用しない
        QbChart.setEspressManagerUsed(false);

        //テンプレートを開く
        QbChart chart = new QbChart (parent, //container
                        "../templates/ChartSwitchToDataFile.cht"); //template

        try {
                //データソースを切り替える
                chart.gethInputData().setDataFile(....);

                //グラフを更新する
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            chart.refresh();

        } catch (Exception ex)
        {
            ex.printStackTrace();

        }

        //ビューアにグラフを表示する
        return chart;

}
```

フルソースコード
エクスポート後の結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

5.9.5.3.3 XML データソースのデータ

データに付随する.dtd または.xml スキーマが存在する限り、データソースをカスタム XML データに切り替えることができます。XML データ情報が指定（詳細は XML データソースのデータを参照してください）された後、**QbChart** オブジェクトに渡されます。

以下の例は、アプレットまたはアプリケーションとして実行でき、**QbChart** オブジェクトのデータソースを XML データに切り替えます。

```
Component doSwitchToXMLData(Applet parent) {

        //EspressManager を使用しない
        QbChart.setEspressManagerUsed(false);

        //テンプレートを開く

        QbChart chart = new QbChart (parent, //container
                        "../templates/ChartSwitchToXMLData.cht"); //template

        //XML データソース情報
        XMLFileQueryInfo newData = new XMLFileQueryInfo(...);

        try {
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            //データソースを切り替える
            chart.gethInputData().setXMLFileQueryInfo(newData);

            //グラフを更新する
            chart.refresh();

        } catch (Exception ex)
        {
            ex.printStackTrace();

        }

        //ビューアにグラフを表示する
        return chart;

}
```

フルソースコード
エクスポート後の結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

5.9.5.3.4 カスタム実装のデータ

通常のデータソースに加えて、独自のカスタムデータを渡すこともできます。カスタムデータは、**IDataSource** インタフェースを使用して **QbReport** オブジェクトに渡されます（詳細については、カスタム実装で渡されたデータを参照してください）。

以下の例は、アプレットまたはアプリケーションとして実行でき、データソースをカスタム実装に切り替えます。

```
Component doSwitchToCustomData(Applet parent) {

        //EspressManager を使用しない
        QbChart.setEspressManagerUsed(false);

        //テンプレートを開く
        QbChart chart = new QbChart (parent, //container
                        "ChartSwitchToCustomData.cht"); //template

        try {
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            //データソースを切り替える
            chart.gethInputData().setClassFile(..);

            //グラフを更新する
            chart.refresh();

        } catch (Exception ex)
        {
            ex.printStackTrace();

        }

        //グラフにビューアを表示する
        return chart;

}
```

フルソースコード
エクスポート後の結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

5.9.5.3.5 メモリ内の配列に渡されたデータ

配列を使用してデータを渡すこともできます。配列データは通常メモリに格納され、**QbChart** オブジェクトに渡されます（詳細はメモリ内の配列に渡されたデータを参照してください）。

アプレットまたはアプリケーションとして実行できる次の例は、データソースをメモリ内の配列に切り替えます。

```
Component doSwitchToArrayData(Applet parent) {

        //EspressManager を使用しない
        QbChart.setEspressManagerUsed(false);

        //テンプレートを開く
        QbChart chart = new QbChart (parent, //container
                        "ChartSwitchToArrayData.cht"); //template

        //配列データを作成する
        DbData newData = new DbData(...);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        try {
                //データソースを切り替える
                chart.gethInputData().setData(newData);

                //グラフを更新する
                chart.refresh();

        } catch (Exception ex)
        {
                ex.printStackTrace();

        }

        //ビューアにグラフを表示する
        return chart;

}
```

フルソースコード
エクスポート後の結果

上記のコードは完全ではなく、ガイドとしてありますのでご注意ください。ただし、リンクには実行可能な完全なアプリケーションコードが含まれています。

5.9.5.4 グラフ属性の変更

EspressReport には何百ものプロパティがあり、チャートのさまざまな要素を細かく制御できます。使いやすさを容易にするために、ほとんどのプロパティはグループに分類され、インタフェースの形式で公開されています。アプリケーションは、まず **gethXXX** メソッドを使用してグループインタフェースのハンドルを取得し、そのインタフェース上のメソッドを呼びたすことによって、チャートのプロパティを直接操作します。ほとんどのインタフェースは、quadbase.util および quadase.ChartAPI パッケージに含まれています。

5.9.5.4.1 色/フォントなどの変更

チャートはさまざまな要素（キャンバス、軸、凡例、チャートデータなど）で構成されています。それぞれの要素は、適切なインタフェースを取得し、その中のメソッドを呼びだすことによって、他と独立して変更することができます。

たとえば、次のコードは、チャートの背景色を白に設定します：

```
chart.gethCanvas().setBackgroundColor(Color.white);      //QbChart 型オブジェクトのオブジェクト
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
であるチャート
```

次のコードは、X 軸の色を黒に変更します。

```
chart.gethXAxis().setColor(Color.black);                    //QbChart 型のオブジェクトであるチャート
```

系列に基づいて、またはカテゴリに基づいてチャートのデータ要素の色を設定することもできます（系列が定義されていない場合）。たとえば、チャートが系列内に 5 つの要素含む縦棒グラフである場合、次のコードは列の色を黄、オレンジ、赤、緑、青に設定します。

```
Color dataColors[] = {Color.yellow, Color.orange, Color.red, Color.green, Color.blue};
chart.gethDataPoints().setColors(dataColors);          //QbChart 型のオブジェクトであるチャート
```

定義された色の数は、系列内の固有の要素の数と一致しなければならないことに注意してください（系列が定義されていない場合はカテゴリ）。

同様にフォントスタイルは、適切なインタフェースを呼び出して、そのメソッドを使用して設定できます。次のコードは、凡例に使用されるテキストを Arial、太字タイプおよびサイズ 15 に設定します。

```
Font legendFont = new Font("Arial", Font.BOLD, 15);
chart.gethLegend().setFont(legendFont);                 //QbChart 型のオブジェクトであるチャート
```

次のコードは、Y 軸で使用されるラベルのフォントと色を設定します。

```
Font YAxisFont = new Font("Helvetica", Font.ITALIC, 15);
chart.gethYAxis().gethLabel().setFont(YAxisFont);      //QbChart 型のオブジェクトであるチャート
chart.gethYAxis().gethLabel().setColor(Color.blue);   //QbChart 型のオブジェクトであるチャート
```

同様に他のプロパティは、インタフェースのメソッドを使用して設定できます。

5.9.5.4.2 定義済みパターンの設定

クラス **QbPattern** は、**java.awt.Color** のサブクラスなので、パターンを明示的に設定するための新しいメソッドはありません。最初に **QbPattern** オブジェクトを作成し、それを **Color** オブジェクトであるかのように使用します。パターンフィーチャは、データポイントにのみ適用されます。他のチャート要素（軸、キャンバスなど）の場合、EspressReport は **QbPattern** のプロパティを無視し、**Color** と同じように使用します。

アプレット、またはアプリケーションとして実行できる次の例は、パターンをデータポイントに設定する方法を示しています。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
Color[] dataColors = chart.gethDataPoints().getColors();
QbPattern dataPattern = new QbPattern(colors[2], QbChart.PATTERN_THICK_FORWARD_DIAGONAL);
dataColors[2] = dataPattern;
chart.gethDataPoints().setColors(dataColors);
```

フルソースコード
エクスポート後の結果

使用できるパターンは 30 種類あります。パターン ID の範囲は 0～29 です。ユーザがこの範囲を超えて整数を渡すと、ID = 0 とみなされます。これは、単色のブロックです。パターン ID は、quadbase.ChartAPI.IMiscConstants クラスで定義されています。QbChart は、IMiscConstants インタフェースを実装しているので、これらの ID は QbChart から直接取得できます。パターンパレットデザイナで、各パターンの外観を確認することもできます。

5.9.5.4.3 カスタマイズパターンの設定

独自のパターン（テクスチャ）イメージを設定することもできます。この機能は、API を介してのみ利用可能であり、変更はテンプレートに保存させません。この機能は、主にランタイム中にグラフを変更したい場合に使用します。

アプレット、またはアプリケーションとして実行できる次の例は、カスタムパターンをデータポイントに設定する方法を示しています。

```
Color[] dataColors = chart.gethDataPoints().getColors();
QbPattern dataPattern = new QbPattern(dataColors[2]);
File customImageFile = new File("Quadbase_Logo.png");
BufferedImage customImage = ImageIO.read(customImageFile);
dataPattern.setPatternImage(customImage);
dataColors[2] = dataPattern;
chart.gethDataPoints().setColors(dataColors);
```

フルソースコード
エクスポート後の結果

上記のコードでは、パターン ID を指定せずに QbPattern オブジェクトが作成されます。新しく作成されたオブジェクトは、java.awt.Color オブジェクトに相当します。開発者は、プロシージャで NullPointerException プロンプトが表示されないように、イメージを提供する必要があります。次に、ファイルからの既存のイメージが、データポイント 2 のパターンとして渡されます。

5.9.5.4.4 サイズの変更

チャートを作成する際に考慮すべき 2 つのサイズがあります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

相対サイズ

このサイズは、チャートキャンバスに関連して定義されます。たとえば、チャートプロットの高さを 7、チャートプロットの幅を 8 に設定することができます。キャンバスが 300×300 ピクセルの場合、グラフプロットの高さは 210（.7×300）ピクセルになり、幅は 240 ピクセル（または.8×300）になります。キャンバスサイズを 500×600 ピクセルに増やすと、チャートプロットの高さは 420 ピクセル（.7*600）になり、幅は 400 ピクセル（.8*500）になります。相対的なサイズは、キャンバスの高さと幅に依存します。

絶対サイズ

このサイズは、キャンバスの絶対サイズをピクセル単位で表します。

チャートの size パラメータが使用される全ての要素は、チャートキャンバスに関連して定義されます。float は、チャート要素の相対的なサイズを表すために使用されます。たとえば、次のコードでは、グラフのプロットの高さを.65 に設定しています。

```
chart.gethChartPlot().setRelativeHeight(.85f);      //QbChart 型のオブジェクトであるチャート
chart.gethChartPlot().setRelativeWidth(.65f);       //QbChart 型のオブジェクトであるチャート
```

次のコードは、グラフのキャンバスサイズを設定します。

```
Dimension canvasSize = new Dimension(500, 450);
chart.gethCanvas().setSize(canvasSize);              //QbChart 型のオブジェクトであるチャート
```

同様に他のサイズは、インタフェースのメソッドを使用して設定することができます。

5.9.5.4.5 日付/時刻ズームチャートの変更

デザイナーで作成された日付/時刻ズームテンプレートのプロパティ（スケール、開始/終了時間など）を変更することができます。これは、ズームプロパティ（IzoomInfo インタフェースを使用）のハンドルを取得し、そこにあるさまざまなメソッドを使用して、プレゼンテーションを変更することによって行われます。

アプレットまたはアプリケーションとして実行できる次の例は、日付/時刻ズームチャートを変更する方法を示しています。

```
//開始と終了の期間とスケールを変更する
IZoomInfo zoomInfo = chart.gethZoomInfo();

//開始日
Calendar beginDate = new GregorianCalendar(2001, 0, 1);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
//終了日
Calendar endDate = new GregorianCalendar(2003, 11, 31);

zoomInfo.setLowerBound(beginDate.getTime());
zoomInfo.setUpperBound(endDate.getTime());

zoomInfo.setScale(6, IZoomInfo.MONTH);

chart.refresh();
```

フルソースコード
エクスポート後の結果

上のコードでは、グラフは QbChart 型のオブジェクトです。グラフのズームプロパティを有効にするには、グラフを更新する必要があります。

5.9.5.4.6 レポートでのチャートの変更

スタンドアロンチャートは API を使用して直接変更できますが、レポート内のチャート（チャートが独立しているかどうかに関係なく）を簡単に変更することはできません。IChartModefier インタフェースを実装して、目的の変更を呼び出すクラスを作成することによって、レポート内のグラフの外観を変更できます。実際のチャートテンプレートは変更されておらず、レポート内のグラフの外観だけが変更されています。

アプレットまたはアプリケーションとして実行できる次の例は、レポート内のグラフを変更する方法を示しています。

```
ReportChartObject[] reportCharts = report.getReportChartObjects();
for (int i = 0; i < reportCharts.length; ++i)
{

        reportCharts[i].setChartModifier(new ModifyChart());

}

public class ModifyChart implements IChartModifier {

        public IChart modifyChart(Object chartInfo) {
                ChartObject chart = new ChartObject(chartInfo); //実際の ChartObject を取得する
                chart.gethCanvas().setBackgroundColor(Color.white);
                return chart; }
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
}
```

フルソースコード
エクスポート前の結果
エクスポート後の結果

5.9.5.5 チャートのエクスポート

レポートに含まれるチャートは、JPEG、GIF、PNG フォーマットフォーマットにエクスポートすることもできます。デフォルトでは、レポートのグラフは JPEG 形式でエクスポートされます。グラフのフォーマットは変更できます。**ReportChartObject** オブジェクトの作成時には、イメージタイプを指定して、**setImageType**（**int**）メソッドを使用します。

EspressReport API には、スタンドアロンチャートをさまざまな形式でエクスポートする機能があります。これらには、GIF、JPEG、PNG、BMP、PDF 形式があります。さらに、グラフを独自の.cht または.tpl 形式にエクスポートすることもできます。.cht はすべてのグラフ情報を格納し、静止画像のファイルの EspressReport に相当するものとみなすことができます。.cht ファイルは、データソースからデータを更新でき、ズーム、ドリルダウンなどの追加機能が可能であるという点で静止画像ファイルとは異なります。.tpl ファイルは、.cht ファイルと同じですが、実際のデータを格納します。.tpl ファイルの場合、チャートのリロード時に元のデータソースからデータが自動的にリロードされます。.cht ファイルと.tpl ファイルは、EspressReport Viewer または EspressReport API を使用して表示できます。スタンドアロンチャートを使用する場合は、**QbChart** オブジェクトを作成し、そのクラス内のさまざまなエクスポートメソッドを使用して、書式を指定できます。

スタンドアロンチャートを、さまざまな形式やオプションでエクスポートするには、いくつかの方法があります。もっとも簡単な方法は、次のとおりです。

```
public export(int format, String filename)
```

上記の方法では、format は書式定数の 1 つであり、filename は出力ファイル名（拡張子あり、またはなし）です。

次は、グラフコンポーネントのエクスポートに使用される定数のリストです。

ビットマップ	QbChart.BMP
GIF	QbChart.GIF
JPEG	QbChart.JPEG
PNG	QbChart.PNG

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

PDF	QbChart.PDF
Flash	QbChart.FLASH
スケーラブルなベクトルグラフィックス	QbChart.SVG
テキストデータ	QbChart.TXTFORMAT
XML データ	QbChart.XMLFORMAT
Windows メタファイル	QbChart.WMF

選択した形式に応じて、追加オプションも使用できます。たとえば、グラフオブジェクトを jpeg としてエクスポートすると、画質を選択できます（0～99 の数値で表されます）。PNG 形式でグラフオブジェクトをエクスポートすると、使用する圧縮を指定することができます。オプションは、次のメソッドを指定できます。

```
public export(int format, String filename, int option)
```

アプレットまたはアプリケーションとして実行できる次のコードは、グラフを作成およびエクスポートする方法を示しています。

```
//テンプレートを開く
QbChart chart = new QbChart(parent, "ExportChart.cht");

try {

        chart.export(QbChart.PNG, "ExportChart");

} catch (Exception ex) {

        ex.printStackTrace();

}
```

フルソースコード
エクスポート後の結果

チャートをテキストファイルにエクスポートすると、デフォルトの区切り文字はタブスペースになります。ただし、次の方法を使用して別の区切り文字を設定することもできます（区切り文字は“,”、“;”、“”です）。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
chart.exportDataFile("TextData", QbChart.COMMA, QbChart.TXTFORMAT);     //ここで、グラフは QbChart 型のオブジェクトです
```

グラフオブジェクトを表示せずにエクスポートする場合は、次の静的メソッドを trueni 設定すると、パフォーマンスがわずかに向上します。

```
QbChart.setForExportOnly(boolean);
```

QbChart オブジェクトを作成する前にこのメソッドを呼び出すと、パフォーマンスが向上します。

次のメソッドを使用して、**QbChart** オブジェクトをバイト配列にエクスポートすることもできます。

```
public byte[] exportChartToByteArray();
```

これにより、.cht はバイト配列の形で与えられます。これは、**QbChart** オブジェクトをシリアライズする必要があり場合に便利です。

5.9.5.5.1 レコードファイルのエクスポート

EspressReport は、大量のデータを処理するためのレコードファイルの、エクスポート機能も備えています。レコードファイルのエクスポートでは、メモリに保持するレコード数と一時ディレクトリにディスク上のデータが格納されます。

この機能を使用するには、次の条件を満たす必要があります。

- EspressManager が使用されていないことを確認してください。
- データが XML ファイルでないことを確認してください。
- データがソートされていることを確認してください。

レコードファイルのエクスポートを使用するには、QbChart コンストラクタを呼び出す前に、次のコードを追加する必要があります。

```
QbChart.setEspressManagerUsed(false);
QbChart.setTempDirectory(<specify the temp directory to store data. Default is ./temp>);
QbChart.setMaxCharForRecordFile(<specify the maximum character length. Default is 40>);
QbChart.setMaxRecordInMemory(<specify the maximum number of rows to be kept in memory. Default is -1 i.e., store all records in memory>);
QbChart.setFileRecordBufferSize(<specify the number of rows to be retrieved at one time from disk. Default is 10,000>);
```

5.9.5.5.2 ストリーミングチャート

ローカルドライブへのエクスポートに加えて、チャートをバイトストリームとしてエクスポートして、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Web ブラウザに直接ストリーミングすることもできます。サーバ側のコードは通常、バイナリ形式でのみイメージをエクスポートします。エクスポートされたいイメージの周りにラッパー（DHTML/HTML コンテンツ）を作成する責任があります。

以下は、グラフを PNG にエクスポートして、ブラウザにストリームする例です。この例を実行するには、ソースコードを設定してコンパイルし、サーブレットランナーのサーブレットディレクトリにクラスファイルをデプロイする必要があります。**chartTemplate** 変数を絶対変数、またはアプリケーションサーバの作業ディレクトリに対する相対パスに置き換えます。

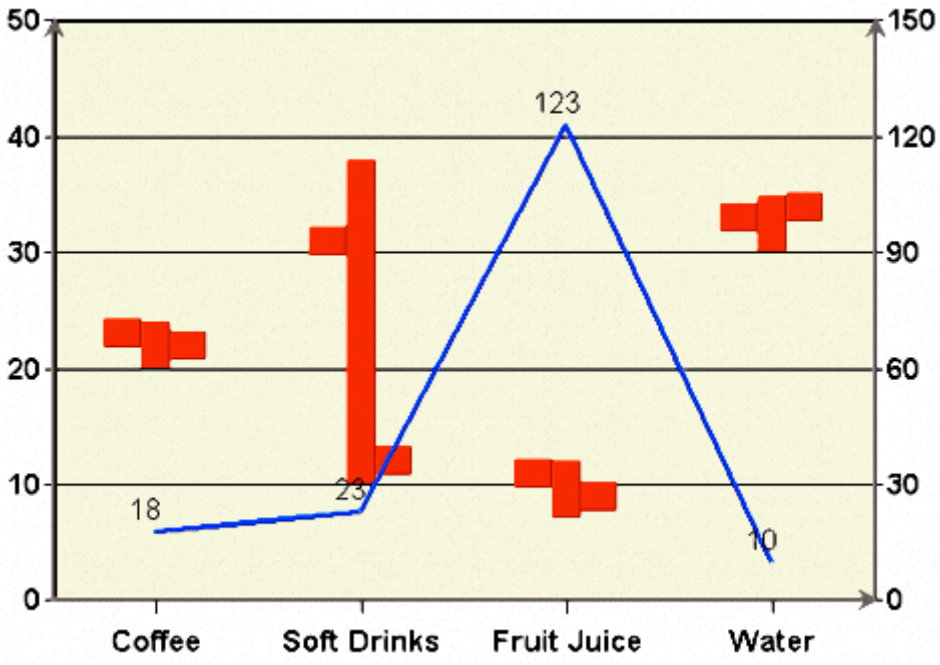
```
public void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

	//EspressManager を使用しない
	QbChart.setEspressManagerUsed(false);

	String chartTemplate = "StreamingChart.cht";

	//テンプレートを開く
	QbChart chart = new QbChart((Applet)null, chartTemplate);

	//グラフを PNG にエクスポートし、バイトをストリームします。
	ByteArrayOutputStream chartBytes = new ByteArrayOutputStream();

	try {
		chart.export(QbChart.PNG, chartBytes);

	} catch (Exception ex)
	{
		ex.printStackTrace();

	}

	resp.setContentType("image/x-png");
	resp.setContentLength(chartBytes.size());

	OutputStream toClient = resp.getOutputStream();
	chartBytes.writeTo(toClient);
	toClient.flush();
	toClient.close();

}
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フルソースコード

この例を実行するには、コードによってアクセスされる場所にグラフテンプレートをコピーしないでください。相対パスを使用する場合、現在のディレクトリはアプリケーションサーバの作業ディレクトリであることに注意してください。たとえば、Tomcat の作業ディレクトリは**<Tomcat>/bin** なので、**<Tomcat>/templates**/ディレクトリを作成し、そこにグラフテンプレートをコピーして、コードを実行する必要があります。結果のグラフを以下に示します。



ストリーミングチャート

5.9.6 API のみの機能

EspressReport API は Chart Designer の全ての機能を再現できますが、API には追加の機能もあります。これらの追加の特徴について、このセクションで説明します。一部はスタンドアロンチャート専用です。

提供されるコードのスニペットでは、グラフは **QbChart** 型のオブジェクトです。

5.9.6.1 ビジュアル

このセクションに示すフィーチャは、チャートとそのコンポーネントの表示を処理します。このセクションの各機能は、グラフを何らかの方法で視覚的に変更します。これらの変更は、静止画像にエクスポートされたときにも実行されます。

5.9.6.1.1 キャンバス/プロット

次の機能は、API を使用してグラフプロットや、キャンバスに行うことができるさまざまな視覚的な変更について説明しています。以下の機能は、グラフプロットやキャンバスの一般的な変更に対応してい

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ます。コンポーネント固有の変更については、該当するセクションを参照してください。

プロットなしのデータのためのカスタマイズ可能なメッセージ

EspressReport は、データがない時やプロットするデータが不足しているときに表示されるメッセージを変更できるようにします。これは、**INoDataToPlotMessage** へのハンドルを取得し、新しいメッセージを指定することによって実行できます。

```
INoDataToPlotMessage noData = chart.gethNoDataToPlotMessage();
noData.setMessage("Not enough data to plot chart");
```

詳細は、オンライン API のドキュメントを参照してください。

キャンバスに合わせてグラフを設定

EspressReport では、グラフに関連付けられたすべてのテキストとラベルを考慮して、グラフのサイズを変更してキャンバスに正しく収めることができます。グラフをキャンバスに正しくフィットさせるには、ICanvas インタフェースで **setFitOnCanvas**（**boolean b**）メソッドを使用します。

```
ICanvas canvas = chart.gethCanvas();
canvas.setFitOnCanvas(true);
```

詳細は、オンライン API のドキュメントを参照してください。

チャートの不可視化

表形式のプロットデータのみが表示されるように、グラフを非表示にすることができます。これは、**ICanvas** インタフェースへのハンドルを取得し、**setChartVisible** メソッドを使用して行います。

```
ICanvas canvas = chart.gethCanvas();
canvas.setChartVisible(false);
```

詳細は、オンライン API のドキュメントを参照してください。

異なるグラフィックレンダリングの適用

EspressReport では、さまざまなレンダリング手法（アンチエイリアシングなど）をグラフに適用することができます。これにより、自在にグラフを描くことができます。

API でこの機能を利用するには、**QbChart** の **setRenderingHint** メソッドを呼び出し、ヒントキーとヒント値のパラメータを渡します。

```
chart.setRenderingHint(java.awt.RenderingHints.KEY_ANTIALIASING,
java.awt.RenderingHints.VALUE_ANTIALIAS_ON);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

アンチエイリアス処理される水平方向のテキストだけを指定することもできます。

```
chart.forceApplyAntiAliasToHorizontalText(true);
```

詳細は、オンライン API のドキュメントを参照してください。

この機能は、スタンドアロンチャートでのみ使用できます。

チャートプロット位置

EspressReport では、グラフの位置を-0.5（キャンバスを基準にした x と y の位置）にすることができます。これにより、グラフをキャンバスの端に沿って、正しく配置することができます。

API を使用してこの機能を利用するには、IPlot のハンドルを取得し、**setPosition** メソッドを使用します。

```
IPlot chartPlot = chart.gethChartPlot();
chartPlot.setPosition(new Position(-0.5, -0.5));
```

詳細は、オンライン API のドキュメントを参照してください。

同じプロットで複数の図の描画

EspressReport は、複数のグラフを重ねて表示することができます（プライマリチャートの軸を使用）。プライマリチャートのキャンバスが使用されるため、2D チャートを 3D チャートに追加することはできず、その逆もできません。プライマリチャートに重なっているすべてのグラフには、テキストとキャンバスの情報がすべて削除されます。結果チャートでは、プライマリチャートのカテゴリと縮尺が使用されるため、追加チャートでも同様のカテゴリと縮尺を正しく表示する必要があります。散布図とサーフェスチャートでは、x 軸と y 軸の両方のスケールを考慮する必要があります。

API を使用してこの機能を利用するには、**QbChart** で **setAddOnChart** メソッドを使用します。

アプレットまたはアプリケーションとして実行でいる次のコードは、3 つのグラフを 1 つのプロットでオーバーラップさせる方法を示しています：

```
//テンプレートを開く
QbChart primaryChart = new QbChart(parent,
//container
                                   "../templates/AddOnChart1.cht"); //template

//他の 2 つのテンプレートを開く
QbChart chart2 = new QbChart(parent, "../templates/AddOnChart2.cht");
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
QbChart chart3 = new QbChart(parent, "../templates/AddOnChart3.cht");

primaryChart.setAddOnChart(new QbChart[] {chart2, chart3});
```

フルソースコード
エクスポート後の結果

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.1.2 ヒントボックス

このセクションの機能では、API を使用して、グラフのヒントボックスに加えることができる様々な視覚的な変更を示しています。これには、ヒントボックスの外観だけでなく、コンテンツの変更も含まれています。

ヒントボックスの変更

EspressReport は、ヒントボックスを修正することができます。つまり、ヒントボックスが表示された時に、ヒントボックスの内容を指定することができます。これは、**IHintBoxInfo** を実装するクラスを作成し、そのクラスを **IHint** の **setHintBoxInfo** メソッドを使用して割り当てることによって行われます。

アプレット、またはアプリケーションとして実行できるコードは、ヒントボックス情報を変更する方法を示しています。

```
//テンプレートを開く
QbChart chart = new QbChart(parent,                                    //コンテナ
                            "../templates/ModifyHintBox.cht"); //テンプレート

IHintBoxInfo hintBoxInfo = new Hint();
chart.gethDataPoints().gethHint().setHintBoxInfo(hintBoxInfo);

...

class Hint implements IHintBoxInfo {

        public Vector getHint(PickData pickData) {
                Vector vec = new Vector();

                if (pickData.series != null)
                        vec.addElement("(" + pickData.seriesName+ ") " + pickData.s_series);

                if (pickData.category != null)
                        vec.addElement("(" + pickData.categoryName + ") " + pickData.s_category);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
            if (pickData.s_value != null)
                vec.addElement("(" + pickData.valueName + ") " + pickData.s_value +
(pickData.value > 7 ? ":Good" : ":Restock"));

            return vec;

        }

}
```

フルソースコード
エクスポート後の結果

詳細は、オンライン API のドキュメントを参照してください。

この機能は、スタンドアロンチャートでのみ使用できます。

データとハイパーリンクのヒントボックスのオフセット

EspressReport では、グラフ内のグラフや他のコンポーネントと重ならないように、データと HyperLink のヒントボックスにオフセットを設定することができます。

API でこの機能を利用するには、グラフオブジェクトまたは **IHyperLinkSet** オブジェクトのいずれかから **IHint** へのハンドルを取得し、**setOffset** メソッドを使用します。

```
IHint dataHints = chart.gethDataPoints().gethHint();
dataHints.setoffset(new Dimension (30, 20));
```

詳細は、オンライン API のドキュメントを参照してください。

ヒントボックスの境界の色

EspressReport では、API を使用してヒントボックスの境界線の色を設定できます。これは、**IHint** のハンドルを取得し、**setBorderColor** メソッドを使用することで実行できます。

```
IHint chartHintBox = chart.gethHint();
chartHintBox.setBorderColor(Color.red);
```

詳細は、オンライン API のドキュメントを参照してください。

イメージマップのヒントボックスのカスタマイズ

EspressReport では、グラフをマップファイルにエクスポートするときに、ヒントボクステキストをカ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

スタマイズすることができます。マップファイルが DHTML/HTML ファイルに含まれている場合、カスタマイズしたヒントボックスは、マウスチャートのデータポイント上に移動すると表示されます。**ICustomizedImageDataMapHintBox** を実装するクラスを作成し、そのクラスを **QbChart** の **setImageMapDataHintBoxHandle** メソッドに割り当てることによって、カスタマイズされたイメージマップのヒントボックスを作成出来ます。

```
//テンプレートを開く
QbChart chart = new QbChart(parent,                                    //コンテナ
                        "../templates/CustomizeImageMapHintBox.cht"); //テンプレート

chart.setImageMapDataHintBoxHandle(new customizeImageMap());

try {

        chart.export(QbChart.PNG,                    "../export/CustomizeImageMapHintBox",
"../export/CustomizeImageMapHintBox", 0, 0, true);

} catch (Exception ex) {

        ex.printStackTrace();

}

...

class customizeImageMap implements ICustomizeImageMapDataHintBox {

        public String customizeHintBox(String str) {
                return str.substring(6, 11) + " " + str.substring(str.length()-3);

        }
}
```

フルソースコード
エクスポート後のマップ
エクスポート後のイメージ

詳細は、オンライン API ドキュメントを参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.9.6.1.3 凡例/注釈

このセクションでは、API を使用してグラフの凡例、および任意の注釈に加えられるさまざまな視覚的変化を記述します。これには、グラフの凡例/注釈の外観だけでなく、内容の変更も含まれます。

記号による注釈

EspressReport では、シンボルをアノテーションに含めることができるため、シンボルや文字列をグラフに配置することができます。この機能のアプリケーションの 1 つは、EspressReport によって作成されたデフォルトの凡例の代わりに、独自の凡例を作成することです。

IAnnotation 用に、新しいコンストラクタが作成されました。シンボルや、テキストの追加に使用できます。

アプレットまたはアプリケーションとして実行できる次のコードは、記号と注釈を組み合わせる方法を示しています。

```
//テンプレートを開く
QbChart chart = new QbChart(parent,                                    //コンテナ
                                    "../templates/AnnotationWithSymbol.cht"); //テンプレート

//カスタム凡例を作成する
IAnnotationSet set = chart.gethAnnotations();
String[] text = {"New Legend", "Hello World", "ABC", "I got It"};
int[] shape = {QbChart.PLUS, QbChart.NOSYMBOL, QbChart.SQUARE, QbChart.DASH};
Color[] color = {Color.red, Color.black, Color.blue, Color.white};
IAnnotation anno = set.newAnnotation(text, shape, color);
Point_2D newPosition = new Point_2D(.7f, .7f);
anno.setRelativePosition(newPosition);
set.addAnnotation(anno);
```

フルソースコード
エクスポート後のイメージ

詳細は、こちらとこちらのオンライン API のドキュメントを参照してください。

凡例と注釈テキストの参照位置の設定

EspressReport では、凡例と注釈テキストの参照位置を左上隅、またはデフォルト位置（左下隅）のいずれかに設定できます。基準位置を左上の位置に変更するには、**ICanvas** で **setReferenceAtTop** メソッドを使用します。

```
ICanvas canvas = chart.gethCanvas();
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
canvas.setReferenceAtTop(true);
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.1.4 その他

次の機能は、グラフのさまざまなコンポーネントに対して、さまざまな視覚的な変更を行うことができます。これには、グラフとその要素の外観だけでなく、内容の変更も含まれます。

ティッカーラベルの変更

EspressReport では、ティッカーラベルをユーザー定義の文字列に置き換えることができます。これにより、ユーザーは自分のティッカー値を入力することができます。

ティッカーラベルを置き換えるには、**IAxis** の **setTickerLabels** メソッドを使用します。

```
IAxis hXAxis=chart.gethXAxis();
hXAxis.setTickerLabels(new String[] {new String("a"), new String("b"), new String("c")});
```

詳細は、オンライン API のドキュメントを参照してください。

カスタマイズ可能なデータトップラベル

EspressReport は、プライマリチャートとセカンダリチャートの両方のデータトップラベルをカスタマイズすることができます。これを行うには、**IDataLabelInfo** を実装するクラスを作成し、**IDataPointSet** の **setDataLabelInfo** メソッドを使用して、そのクラスを渡します。

アプレット、またはアプリケーションとして実行できる次のコードは、記号と注釈を組み合わせる方法を示しています。

```
//テンプレートを開く
QbChart chart = new QbChart(parent,                                    //コンテナ
                            "../templates/CustomizeDataTopLabel.cht"); //テンプレート

chart.gethDataPoints().setDataLabelInfo(new customizeDataTopLabel());

...

class customizeDataTopLabel implements IDataLabelInfo {

        public String getDataLabel(PickData pickData, String originalDataLabel) {
                return (pickData.toString() + " (" + originalDataLabel + ")")
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        }

}
```

フルソースコード
エクスポート後のイメージ

データのトップラベルへのカスタマイズは、表示されない限り明白ではありません。

詳細は、オンライン API のドキュメントを参照してください。

軸を日付/時刻/タイムスタンプとして表示

EspressReport では、数値軸の代わりに日付/時刻の単位で価軸を表示できます。これは、価軸を持つ任意のグラフです。値軸のデータは依然として数値でなければならないことに注意してください。

API を使用してこの機能を利用するには、**IAxis** へのハンドルを取得し、**setDisplayLabelAsDate** メソッドを使用します。

```
IAxis chartYAxis = chart.gethXAxis();
chartXAxis.setDisplayLabelAsDate(true);
```

詳細は、オンライン API のドキュメントを参照してください。

選択文字列レンダリング

アンチエイリアス処理されていない時は、水平または垂直に描画される文字列がよく見えます。ただし、アンチエイリアス処理を行うと、他の角度の文字列がより見やすくなります。EspressReport では、特定のテキストにアンチエイリアスを適用したり、アンチエイリアスを行わずにほかの文字列（0 と 90 度の角度）を描画することができます。

API を使用してこの機能を利用するには、**IDataPointSet** のハンドルを取得し、**setDisableJava2DForStraightText** メソッドを使用します。

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setDisableJava2DForStraightText(true);
```

詳細は、オンライン API のドキュメントを参照してください。

水平/垂直/トレンド線よりも上のデータポイントを描画

EspressReport は、データ点を任意の水平/垂直/トレンド線の上に描画することを可能にします。つま

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

り、データ点が線で覆われずに線の上に置かれているように見えます。

API を使用してこの機能を利用するには、**ILninePropertySet** のハンドルを取得し、**setDataDrawnOnTop** メソッドを使用します。

```
ILinePropertySet lineProperties = chart.gethLineProperties();
lineProperties.setDataDrawnOnTop(true);
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.2 データ

以下に示すフィーチャは、グラフとそのコンポーネントのデータ処理をします。このセクションの各機能は、表示されたデータ（通常はデータの表示）を変更したり、グラフ内のデータを取得したりします。これらの変更は、静止画像にエクスポートされた時にも実行されます。

5.9.6.2.1 座標の取得

EspressReport では、指定されたピクセル位置に基づいて、データポイントの情報を取得できます。これは、指定されたピクセル位置のデータポイントのカテゴリ、値、系列、および sumby（存在する場合）を返し、そうでない場合は null を返します。

pickData を取得するには、**IHint** の **getPickData**（**width**、**height**）メソッドを使用します。

```
IDataPointSet dataPoints=chart.gethDataPoints();
IHint hint=dataPoints.gethHint();
PickData pickdata=hint.getPickData(200, 300)          //幅 200 ピクセル、高さ 300 ピクセル
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.2.2 軸スケールでのデータ制限の設定

手動軸が適用されると、EspressReport は、軸上の最大値を超えているデータポイントを切り捨てるオプションを提供します。**IDataPointSet** の **setLimitAtAxisScale** メソッドを使用して、データ制限を設定できます。

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setLimitAtAxisScale(true);
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.2.3 Null データを 0 として設定

EspressReport は、0 データ（すなわち、関連する 0 値を有するデータポイント）として、任意の null

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

データを表すことができます。これを行うには、**IdataPointSet** で **setNullDataAsZero** メソッドを使用します。

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setNullDataAsZero(true);
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.2.4 追加のトレンドラインオプション

EspressReport では、**ITrendLine** を呼び出して適切な方法を使用することによって、標準曲線偏差線の平均値に加えて、最小値、最大値、確率、逆正規率を得ることができます。

この機能を API で利用するには、、**ITrendLine** を呼び出して適切なメソッドを使用する必要があります。

グラフが **setLinearScale** および **setRounded** true ヒストグラムグラフである場合にのみ、標準曲線の標準偏差傾向線を描画します。

アプレット、またはアプリケーションとして実行できる次のコードは、グラフ内の通常の曲線のトレンドラインから、情報を取得する方法を示しています。

```
ITrendLine normalCurve = (ITrendLine)chart.gethDataLines().elements().nextElement();
System.out.println("Mean = " + normalCurve.getMean());
System.out.println("St. Dev = " + normalCurve.getStandardDev());
System.out.println("Min = " + normalCurve.getMin());
System.out.println("Max = " + normalCurve.getMax());
System.out.println("Dev of 1.0, Prob = " + normalCurve.getProbability(1.0));
System.out.println("Back Calculated Dev = " +
normalCurve.getInverseNorm(normalCurve.getProbability(1.0)));
```

フルソースコード
エクスポート後の結果

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.3 グラフ固有のオプション

このセクションのフィーチャは、特定のグラフタイプを処理します。このセクションの各機能はグラフタイプによって、グラフで使用できる追加の機能を示しています。これらの変更は、静止画像にエクスポートされた時にも実行されます。

5.9.6.3.1 縦棒/棒グラフ

次の機能は、API を使用して列/棒グラフに行うことのできる様々な変更点を示しています。これには、

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

グラフの外観だけでなく、コンテンツの変更も含まれます。

カラーセパレータ

EspressReport では、定義されたカテゴリ値に基づいて列の色を表示することができます。カラーセパレータを使用するには、**IDataPointSet** で **setColorSeparators** メソッドを使用します。

アプリケーション、またはアプレットとして実行できる次のコードは、カラーセパレータの設定方法を示しています。

```
IDataPointSet hDataPoints=chart.gethDataPoints();
hDataPoints.setColorSeparators(new Color[]{Color.green, Color.red, Color.blue},
                               new Integer[]{new Integer((int)Math.rint(9)), new Integer((int)Math.rint(14))},
                               QbChart.ASCENDING);
```

フルソースコード
エクスポート後のイメージ

詳細は、オンライン API のドキュメントを参照してください。

影を無効化

EspressReport は、列/バーに表示される影を表示、または非表示にすることができます。この機能を使用するには、**IDataPointSet** のハンドルを取得し、**setShowShadowOnPoint** メソッドを使用します。

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setShowShadowOnPoint(false);
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.3.2 円グラフ

このセクションの機能は、API を使用して円グラフに行うことのできる様々な変更点について説明しています。これには、グラフの外観だけでなく、コンテンツの変更も含まれています。

円グラフ/右回りのパイスライスの描画

EspressReport は、円グラフのスライスを時計回りと、反時計回りの順序で描画できるようになりました。時計回り/反時計回りのオプションを設定するには、**IDataPointSet** の **reverseOrder** メソッドを使用します。

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.reverseOrder(QbChart.CATEGORY);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

詳細は、オンライン API のドキュメントを参照してください。

0%スライスと 100%スライスの円ボーダー

EspressReport は、円全体の 100%であるスライスの円の枠線を削除するオプションを提供します。これを行うには、**IPiePropertySet** のハンドルを取得し、**setRadialBorderDrawnForZero** メソッドを使用します。

```
IPiePropertySet pieProperties = chart.gethPieProperties();
pieProperties.setRadialBorderForZero(true);
```

詳細は、オンライン API のドキュメントを参照してください。

円の凡例のカテゴリと割合値の間のセパレータのカスタマイズ

EspressReport では、円グラフの凡例のカテゴリと%値の文字列の間に、ユーザー定義の区切り記号を配置できます。これは、**IPiePropertySet** へのハンドルを取得し、**setSepSymbol** メソッドを使用することで行うことができます。

```
IPiePropertySet pieProperties = chart.gethPieProperties();
pieProperties.setSepSymbol(",");
```

詳細は、オンライン API のドキュメントを参照してください。

円ボーダーの色のカスタマイズ

EspressReport では、円の境界線の色を指定することができます。これは、**IPiePropertySet** で **setBorderColor** メソッドを使用して行います。

```
IPiePropertySet pieProperties = chart.gethPieProperties();
pieProperties.setBorderColor(Color.red);
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.3.3 折れ線グラフ

このセクションの機能では、API を使用して折れ線グラフに加えられるさまざまな変更点を説明しています。これには、グラフの外観だけでなく、コンテンツの変更も含まれます。

ラインエリア

EspressReport では、水平線とデータ線の間に線の領域を作成して、変更を表すことができます。たとえば、25 の水平線とデータ線があります。データラインと水平ライン、また水平ラインの上に囲まれた全ての領域は 1 色にすることができ、水平ラインとデータラインで囲まれた領域、また水平ラインの下

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

の領域は別の色にすることができます。この機能は、シリーズのない 2D ラインチャートでのみ使用できます。また、この機能を使用するには、水平線の上下に色を設定する必要があります。

API でこの機能を使用するには、**ILinePropertySet** のハンドルを取得し、**setAreaVisible** メソッドと **setAreaColors** メソッドを使用する必要があります。

アプレット、またはアプリケーションとして実行できる次のコードは、行領域の使用方法を示しています。

```
ILinePropertySet lineProperties = chart.gethLineProperties();
lineProperties.setAreaVisible(true);
lineProperties.setAreaColors(Color.green, Color.yellow);
```

フルソースコード
エクスポート後のイメージ

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.3.4 散布図

このセクションの機能では、API を使用して散布図に加えられるさまざまな変更点を説明しています。これには、グラフの外観だけでなく、コンテンツの変更も含まれます。

トップラベルにシリーズを表示

EspressReport は、シリーズを散布図の一番上のラベルに表示することができます。IDataPointset の **showSeriesInTopLabel** メソッドを使用すると、これを行うことができます。

```
IDataPointSet dataPoints=chart.gethDataPoints();
dataPoints.showSeriesInTopLabel(true);
```

詳細は、オンライン API のドキュメントを参照してください。

図面順序

EspressReport では、散布図の接続線をデータセットの順に描画できます。たとえば、データに次の点（0,2)、(3,4)、(1,2)、(2,5）が含まれている場合、接続線のデフォルトの表示は（0,2）から（1,2）から（2,5)、最後に（3,4）となります。ただし、データの順番で接続線を生成することができます。

API を使用してこの機能を利用するには、**IDataPointSet** のハンドルを取得し、**setConnectLinesInOriginalOrder** メソッドを使用します。

```
IDataPointSet dataPoints = chart.gethDataPoints();
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
dataPoints.setConnectLinesInOriginalOrder(true);
```

詳細は、オンライン API のドキュメントを参照してください。

散布図のキューブ幅

EspressReport では、3D 散布図のキューブ幅を任意のサイズに変更できます。キューの幅を変更するには、まず **IDataPointSet** のハンドルを取得し、**setScatterCubeWidth** メソッドを使用します。

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setScatterCubeWidth(15);
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.3.5 オーバーレイチャート

このセクションの機能は、API を使用してオーバーレイチャートに加えられるさまざまな変更点を説明しています。これには、グラフの外観だけでなく、コンテンツの変更も含まれます。

複数の軸のタイトル

EspressReport では、オーバーレイチャートの各軸に異なるタイトルを設定することができます。これは、個々の軸ごとにハンドルを取得し、**gethTitle().setValue** メソッドを使用して行うことができます。レイヤーを指定することによって、個々の軸のハンドルを取得します。

```
IAxis axis0 = chart.gethYAxis();
axis0.gethTitle().setValue("XYZ");

IAxis axis1 = chart.gethAxis(1); //レイヤー1 の軸を取得する
axis1.gethTitle().setValue("ABC");

IAxis axis2 = chart.gethAxis(2); //レイヤー2 の軸を取得する
axis2.gethTitle().setValue("DEF");
```

タイトルを設定する前に、バックグラウンドでグラフを描画する必要があることに注意してください。
詳細は、オンライン API のドキュメントを参照してください。

5.9.6.3.6 ダイヤルチャート

このセクションの機能は、API を使用してダイヤルチャートに加えられるさまざまな変更点を示しています。これには、グラフの外観だけでなく、コンテンツの変更も含まれます。

コントロールエリアのスケールラベル

EspressReport では、コントロール領域の開始と終了のスケールをラベルとして表示できます。これは、コントロール領域のハンドルを取得し、**setShowLabel** メソッドを使用することで実行できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
ControlRange cr1 = chart.gethControlRanges().elementAt(0);
cr1.setShowLabel(true);
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.3.7 HLCO チャート

このセクションの機能は、API を使用して HLCO チャートに加えられるさまざまな変更点を説明しています。これには、グラフの外観だけでなく、コンテンツの変更も含まれます。

キャンドルスティックの色の変更

EspressReport では、API のみを使用して HLCO チャートのロウソク足の色を変更することができます。これを行うには、**IDataPointSet** のハンドルを取得し、**setCandleStickColors** メソッドを使用する必要があります。

```
IDataPointSet dataPoints = chart.gethDataPoints();
//上の色は緑、下の色は赤
dataPoints.setCandleStickColors(Color.green, Color.red);
```

詳細は、オンライン API のドキュメントを参照してください。

キャンドルスティックの枝幅の変更

EspressReport では、HLCO チャートの API のみを使用して、ロウソク足の（ロウソク足の上下の部分）を変更することができます。これを行うには、**IDataPointSet** のハンドルを取得し、**setCanIdeStickWidth** メソッドを使用する必要があります。渡された数は、ロウソク足の幅とロウソク足の幅の比率です。

```
IDataPointSet dataPoints = chart.gethDataPoints();
//幅を 0.5 に設定する
dataPoints.setCandleStickWidth((float)0.5);
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.3.8 サーフェスチャート

このセクションの機能は、API を使用してサーフェスチャートに加えられるさまざまな変更点を説明しています。これには、グラフの外観だけでなく、コンテンツの変更も含まれます。

ヒートマップ

EspressReport では、等高線図のようなサーフェスチャートを描画することができます。基本的に、サーフェスチャートは、指定されたしきい値にしたがって異なる色で、セクションで描画できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

アプレット、またはアプリケーションとして実行できる次のコードは、ヒートマップを使用してサーフェスチャートを作成する方法を示しています。

```
double [] heatMapValues = {3, 6};
Color [] heatMapColors = { Color.green, Color.yellow, Color.red};
ColorSpectrum heatMapColorSpectrum = new ColorSpectrum(heatMapColors, heatMapValues);
I3DPropertySet set = chart.geth3DProperties();
set.setColorSpectrum(heatMapColorSpectrum);
```

上のコードは、3 つの色のセクション、緑色、黄色、赤色、を持つ 3D サーフェスチャートを作成します。色を決定するしきい値は 3 と 6 です。

フルソースコード
エクスポート後のイメージ

詳細は、こちらとこちらのオンライン API のドキュメントを参照してください。

5.9.6.4 パフォーマンス

このセクションで説明するフィーチャは、グラフの生成とエクスポートのパフォーマンスを改善します。このセクションの各機能は、グラフで使用できる追加機能を示し、グラフの生成に必要なメモリリソースと時間を削減します。

5.9.6.4.1 バッファイメージ/フレーム

EspressReport では、エクスポート中に **java.awt.image.BufferedImage** または、**java.awt.Frame** のいずれかを使用して、パフォーマンスを向上させることができます。デフォルトでは、EspressReport は **java.awt.Frame** を使用して、グラフオブジェクトを作成します。**java.awt.Frame** を使用すると、データポイントの数が増えるにつれてパフォーマンスが向上し、**java.awt.image.BufferedImage** を使用すると、大きなグラフのディメンションでより良いパフォーマンスが得られます。これは、**QbChart** で **setBufferedImageUsed** メソッドを使用して行います。

```
chart.setBufferedImageUsed(true);
```

詳細は、オンライン API のドキュメントを参照してください。

この機能は、スタンドアロンチャートでのみ使用できます。

5.9.6.4.2 チャート生成順序

EspressReport では、チャートの生成順序を選択したり、API のみを使用してチャートの生成に要素を追加したりすることができます。たとえば、背景を描き、円を描き、円の中心にグラフを作成してグラフを作成することができます。これを行うには、**IChartGraphics** インタフェースを実装するクラスを

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

作成し、そのクラスを **QbChart** の **setChartGraphics** メソッドに割り当てます。本質的に、**IChartGraphics** インタフェースを使用すると、グラフを描画する前後に任意のグラフィックス情報を追加、または変更できます。

アプレット、またはアプリケーションとして実行できる次のコードは、チャートとグラフィックを特定の順序で生成する方法を示しています。

```
//テンプレートを開く
QbChart chart = new QbChart(parent,                                    //コンテナ
                            "../templates/ChartGeneration.cht"); //テンプレート

chart.setChartGraphics(new chartGenerationGraphics());;

...

class chartGenerationGraphics implements IChartGraphics {

        public void initializeGraphics(Graphics g, int w, int h) {
                g.setColor(Color.red);
                g.fillOval(50, 50, 400, 400);

        }

        public void finalizeGraphics(Graphics g, int w, int h) {
                g.setColor(Color.white);
                g.fillOval(125, 225, 50, 50);
                g.setColor(Color.orange);
                g.drawString("HELLO WORLD", 150, 250);

        }
}
```

フルソースコード
エクスポート後のイメージ

詳細は、オンライン API のドキュメントを参照してください。

この機能はスタンドアロンチャートでのみ使用できます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.9.6.5 ビューア

このセクションで説明されているフィーチャーは、Java アプリケーションまたは Java アプレットのいずれかで、グラフを表示するときにビューアを処理します。このセクションの各機能は、チャートビューアで使用できる追加の機能を示しています。

5.9.6.5.1 コールバックメカニズム

EspressReport には、イベントを処理するための高レベルのコールバック機構があります。チャート内のデータオブジェクトがビューアによって選択されると、アクションイベントが生成されます。イベント引数には、選択したデータポイントの系列、カテゴリ、値などの情報を提供する **PickData** のインスタンスが含まれます。

アプレット、またはアプリケーションイベントをキャプチャする方法を示しています。

```
static TextField textField;

//テンプレートを開く
QbChart chart = new QbChart(parent,                                    //コンテナ
                            "../templates/CallBack.cht"); //テンプレート

chart.setActionListener(new callBackActionListener());;

...

class callBackActionListener implements ActionListener {

        public void actionPerformed(ActionEvent e) {
                Object arg = ((QbChart) e.getSource()).getArgument();

                String click;

                switch (e.getModifiers()) {

                case QbChart.LEFT_SINGLECLICK:
                        click = "Left single click";
                        break;

                case QbChart.LEFT_DOUBLECLICK:
                        click = "Left double click";
                        break;

                case QbChart.RIGHT_SINGLECLICK:
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                click = "Right single click";
                break;

            case QbChart.RIGHT_DOUBLECLICK:
                click = "Right double click";
                break;

            default: //起こらない
                click = "Error !";

            }

            if (arg instanceof PickData)
                textField.setText(((PickData) arg).toString() + " " + click);

            else
                textField.setText((String) arg + " " + click);

        }

}
```

フルソースコード
エクスポート後のイメージ

この機能はスタンドアロンチャートでのみ使用できます。

5.9.6.5.2 ツールのヒントテキストの無効/有効化

EspressReport を使用すると、ナビゲーションパネルのツールヒントのテキストを有効、または無効にすることができます。これは、**I3DControlPanel** で **setToolTipEnabled** メソッドを使用して行うことができます。このオプションは、3D チャートのみに使用されます。

```
I3DControlPanel controlPanel = chart.geth3DControlPanel();
controlPanel.setToolTipEnabled(true);
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.6.5.3 キャンバスエリア

EspressReport は、より多くのイベントプロパティ（すなわち、ユーザー定義のイベントプロパティ）を追加できるように、キャンバスを含むビューパネルを選択することを可能にします。**ICanvas** へのハンドルを取得し、**getCanvasArea()**メソッドを使用してコンポーネントを返すことで行うことができま

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

す。

```
ICanvas chartCanvas = chart.gethCanvas();
Component chartCanvasComponent = chartCanvas.getCanvasArea();
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.7 チャートビューアオプションの変更

チャートビューアを使用してグラフを表示しているときに、ユーザーが実行できる操作とできない操作を構成することができます。

ユーザーがチャートビューアを使用してグラフを表示しているときに、グラフを右クリックすると、メニューがポップアップ表示されます。このメニューを使用して、ユーザーはグラフタイプの変更、チャートメンディションの変更などのグラフオプションを選択できます。グラフや静止画像のタイプをエクスポートすることもできます。デフォルトのポップアプメニューには、利用可能なすべての選択肢がリストされていますが、チャットビューアのポップアップメニューで使用可能なオプションを制御する API メソッドがあります。

これらの API 呼び出しは、**IPopupMenu** で使用できます。

```
IPopupMenu popupMenu = chart.gethPopupMenu();
popupMenu.setDimMenuEnabled(boolean b);
popupMenu.setPopupMenuEnabled(boolean b);
popupMenu.setTypeMenuEnabled(boolean b);
```

詳細は、オンライン API のドキュメントを参照してください。

5.9.8 Javadoc

API 全体の Javadoc は、EspressReport と共に提供されています。この API は、Chart API も利用できません。詳細は、こちらを参照してください。

5.9.9 スイングバージョン

1.1JFC/Swing 版の EspressReport グラフ API も利用できます。詳細は、こちらを参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

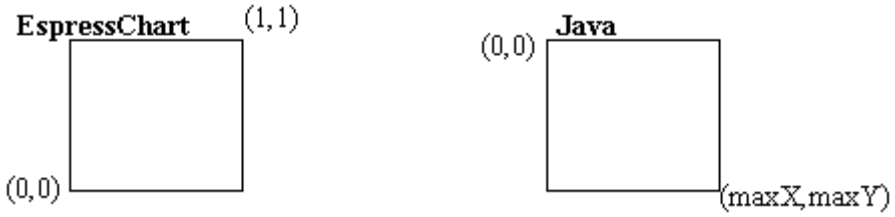
5.9.10 サマリ

EspressReport API は、ビジネスアプリケーション用の使いやすくて強力なチャーティングライブラリを提供します。Chart Designer と組み合わせることで、アプレットまたはアプリケーションに 1 行のコードを追加するだけで簡単にプログラミングできます。グラフのすべての属性は、EspressReport Designer で作成できるテンプレートファイルで設定できます。EspressReport API は、Netscape の Communicator(4.06 以上)、Microsoft の Internet Explorer(4.x 以上)、Windows 95、Windows NT/2000、Solaris、Linux、AIX と HP 上の Sun の Appletviewer（1.2 以上)。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.10 付録 A. チャートレイアウトのカスタマイズ



EspressChart と Java のコンポーネントレイアウトの違い

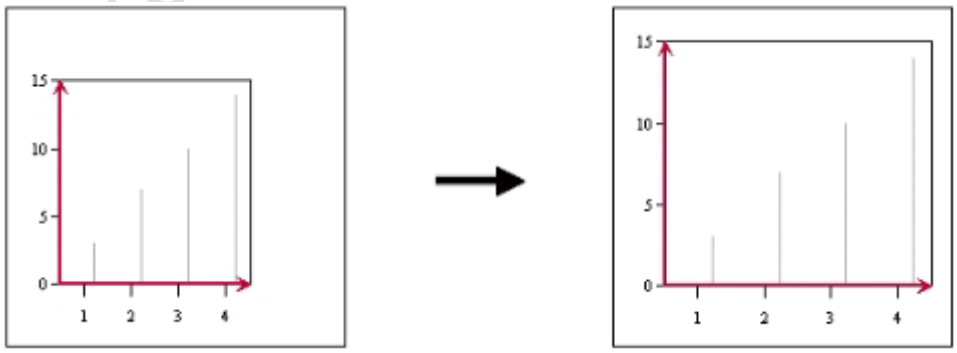
EspressReport は、コンポーネントレイアウトを相対座標で記述します。一方、Java はピクセル単位でレイアウト形式を使用します。下位互換性の問題により、EspressReport のレイアウト形式は変更されません。相対的な位置と絶対的な位置に関連する問題は、グラフィカルユーザのインタフェースがありため、Chart Designer ではあまり関係がありません。ただし、レイアウトの書式が Chart API にどのようになっているかを知ることは有益です。原点(0.0)はキャンバスの左下隅にあり、最大(1.1)は右上隅にあります。

Java レイアウト形式では、原点は左上隅にあり、最大ピクセル(maximumX、maximumY)は右下隅にあります。

IAxis クラスでは、メソッド setMaxScale、setMinScale、setScaleStep は、軸の範囲と間隔を参照します。したがって、これらのメソッドは軸の見かけ上の長さに影響を与えず、むしろ軸の範囲を指定します。

次のセクションでは、チャートコンポーネントのレイアウト、サイズ変更、およびデータポイントのラベリングに関連するカスタマイズテクニックの例について説明します。

5.10.1 キャンバスエリアの変更



キャンバス領域の拡大

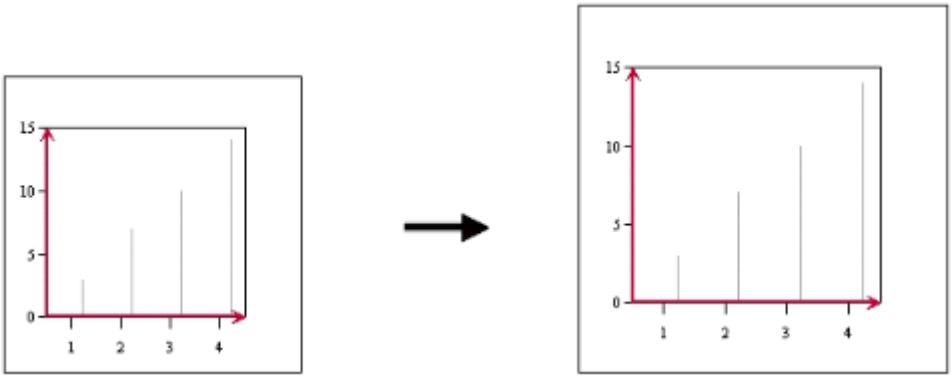
グラフプロット領域のデフォルトの相対的なサイズは、x ディメンションで 0.6、y ディメンションで 0.6 です。API を使用してこれらの比率を調整する場合は、**setRelativeHeight()** および **setRelativeWidth()**メソッドを使用できます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
//チャートのデフォルトの相対サイズは、0.6,0.6 です
IPlot chartPlot = chart.gethChartPlot();
chartPlot.setRelativeHeight( (float) 0.75);
chartPlot.setRelativeWidth( (float) 0.8);
```

チャートデザイナーでチャートプロットを調整するには、マウスカーソルがグラフ上にあるときにマウスの右ボタンをクリックしたままにします。マウスを右にドラッグすると、グラフプロット領域が拡大され、左にドラッグすると、グラフプロット領域が縮小されます。マウスを移動する方向（縦方向、横方向、斜め方向）に応じて、グラフプロットは一方または両方の次元で変化します。

5.10.2 キャンバス領域の変更



キャンバス領域の拡大

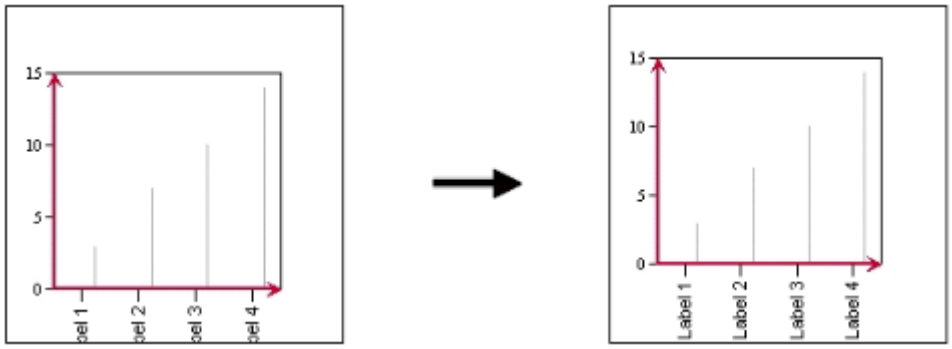
この例では、グラフのサイズをキャンバスに対して一定に保ちながら、キャンバスのサイズを調整する方法を示します。

```
//デフォルトのキャンバスサイズは、600 ピクセル×500 ピクセルです
int width = 500; //ピクセル単位
int height = 550; //ピクセル単位
chart.gethCanvas().setSize( new Dimension(width, height));
```

チャートデザイナでキャンバス領域を変更するには、Format > Canvas を選択し、そこで寸法を変更します。キャンバスは、常に Chart Designer のウィンドウサイズ以上であることに注意してください。小さいキャンバス領域が必要な場合は、最初に Chart Designer ウィンドウのサイズを小さくする必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.10.3 チャートにフィットする要素

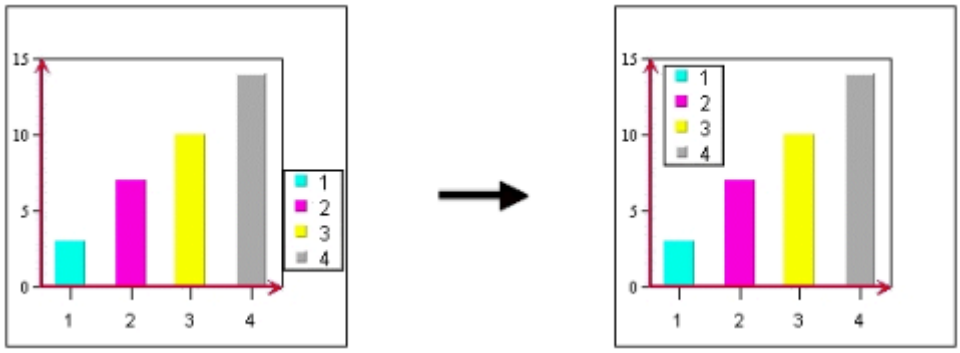


グラフ要素をキャンバスにフィットさせる

キャンバスで切り落とされるグラフ要素（カテゴリ軸のティッカーラベルなど）があり事が分かった場合は、**setFitOnCanvas()**メソッドを使用して、必要な調整を行うことができます。

```
//グラフの位置を変更したり、グラフのサイズを縮小したりする
//EspressReport はチャート全体に収まる
//キャンバス上で、デフォルト値は FALSE です
chart.gethCanvas()。setFitOnCanvas（true）；
```

5.10.4 凡例の位置の変更



凡例ボックスの位置を変更する

凡例の位置は、デフォルトの場所が望まない場合に調整することができます。基準点は凡例の中心です。

```
//凡例ボックスは指定された新しい相対位置で移動できます
ILegend hLegend = chart.gethLegend();
float x = 0.2;
float y = 0.7;
hLegend.setPosition( new Position( x, y) );
```

凡例のサイズは、テキストフォントとテキスト文字列のサイズに依存します。したがって、凡例のボッ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

クスサイズを直室設定することはできませんが、凡例ボックスのサイズを取得することはできます。

ここに、いくつかのサンプルコードがあります。

```
ILegend hLegend = chart.gethLegend()
float relWidth = hLegend.getRelativeWidth();
float relHeight = hLegend.getRelativeHeight();
```

チャートデザイナーで凡例の位置を変更するには、凡例を左クリックしてキャンバス上の目的の位置にドラッグします。

5.10.5 データポイントへのラベルの添付

必要に応じて、データポイントにラベルを付ける必要があることがあります。EspressReport には、これらの要件を満たすためのいくつかの機能があります。たとえば、注釈テキストをトレンドラインに添付することができます。トレンドラインがデータとともに移動すると、注釈テキストはトレンドラインと並行して移動します。トップラベルは、すべてのデータポイントの上部に表示されます。ただし、特定の状況では、選択されたデータポイントラベルを付けることができます。

既存のフィーチャを使用して、次のように表現できます。まず、希望のデータポイントの持つ水平線を描きます。次に、行に注釈テキストを添付し、行の凡例を非表示にします。最後に、点線のスタイルを選択し、塗りつぶしと空のピクセルの長さを 255 にして、水平線を見えなくします。

下記のコードは、この手法を示しています。下記の表は、コードで参照されているデータを示しています。目的は、グラフの右側の 2 つのデータ点（両方の系列の最大値）にラベルを添付し、グラフの左側のデータポイントに 1 つのラベルを添付することです（下図参照）。

最後のデータポイントの 1 つが $ 700 から $ 500 に変更されると、ラベルの位置は自動的に更新されます。

Day	Value1	Value2
1	100	100
2	150	300
3	200	500
4	250	700

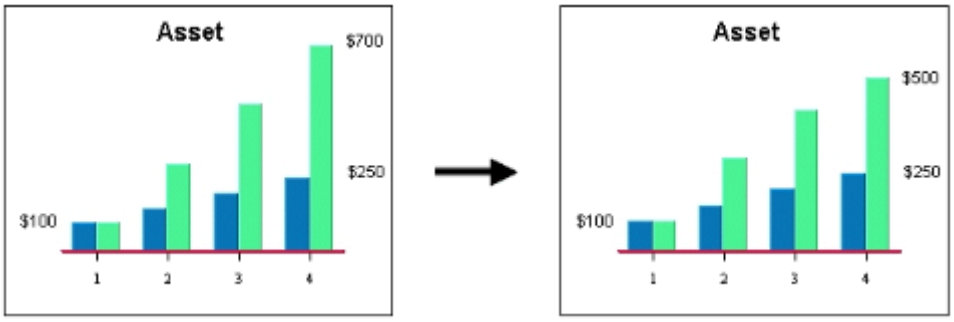
オリジナルデータ

Day	Value1	Value2
1	100	100
2	150	300

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

3	200	450
4	250	500

新しいデータ



元のデータを持つチャートを新しいチャートに変更

```
//example は、データベースクエリの最後の行の値ポイントの 1 つを検索します。
//特に最後の値にラベルを付ける必要はありません。
//プログラムでラベル付けする値、つまり最大値、最小値、
//または平均

//データポイントの最初のセットでハンドルを取得する
IRow rowInit = chart.gethInputData().getRow(0);
//初期値を取得します。最初の列が列 3 からのものである場合、index は 3 です
int valueInit = Integer.parseInt(rowInit.getObject(3).toString());

//データポイントの最後のセットでハンドルを取得する
int lastRownumber = chart.gethInputData().getRowCount() - 1;
IRow lastRow = chart.gethInputData().getRow(lastRownumber);
IRow lastButOneRow = chart.gethInputData().getRow(lastRownumber-1);

//最後の 2 番目の値が最後の行にある
int value1 = Integer.parseInt(lastRow.getObject(3).toString());
//最初のシリーズの最後の値は、最後の行に 1 つあります
int value2 = Integer.parseInt(lastButOneRow.getObject(3).toString());

String labelInit = "$" + valueInit;
String label1 = "$" + value1;
String label2 = "$" + value2;

//初期値の行を追加する
IDataLineSet hDataLines = chart.gethDataLines();
IHorzVertLine hvInit = hDataLines.newHorzVertLine(
IHorzVertLine.HORIZONTAL_LINE, labelInit);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

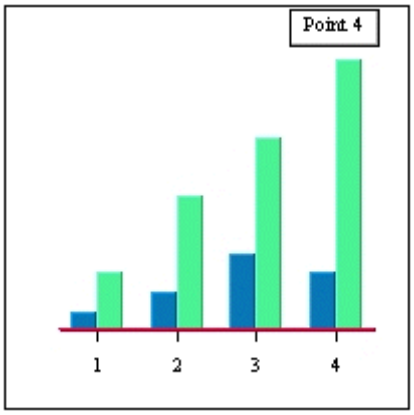
```
hvInit.setLineValue(valueInit);    //値 0 の水平線
hvInit.setLineStyle( ((255*256) + 255) *256); //行を非表示にする
hDataLines.add(hvInit);

//最初のシリーズの終了値の行を追加する
IHorzVertLine hv1 = hDataLines.newHorzVertLine(
IHorzVertLine.HORIZONTAL_LINE, label1);

hv1.setLineValue(value1);    //値 1 の水平線
hv1.setLineStyle( ((255*256) + 255) *256); //行を非表示にする
hDataLines.add(hv1);

//2 番目の系列の終了値の行を追加する
IHorzVertLine hv2 = hDataLines.newHorzVertLine(
IHorzVertLine.HORIZONTAL_LINE, label2);

hv2.setLineValue(value2);    //値 1 の水平線
hv2.setLineStyle( ((255*256) + 255) *256); //行を非表示にする
hDataLines.add(hv2);

//行に注釈を追加する
IAnnotationSet hAnnotation = chart.gethAnnotations();
IAnnotation annoInit = hAnnotation.newAnnotation(labelInit, hvInit);
IAnnotation anno1 = hAnnotation.newAnnotation(label1, hv1);
IAnnotation anno2 = hAnnotation.newAnnotation(label2, hv2);

//グラフプロット領域の左側に annoInit を追加します（100 ドル）
hvInit.addAnnotation(annoInit);
annoInit.setRelativePosition(new Point_2D( -(float)chart.gethChartPlot().getRelativeWidth(), 0f));

//アノテーション 1 を行に追加すると、グラフに表示されます
hv1.addAnnotation(anno1);

//アノテーション 2 をグラフに追加します。グラフに表示されます
hv2.addAnnotation(anno2);

//凡例ボックスを非表示にする
chart.gethLegend().setVisible(false);
```

個別のカテゴリトップラベルを選択して表示する場合は、対応するカテゴリ値を持つ垂直トレンドライ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ンを追加できます。EspressChart のグラフィカルな仕組みでは、カテゴリのデータポイントに位置は、通常数字ではなくても、常に数字として参照されます。最初のデータポイントは常に 0.5 と表示され、次の各ポイントは、1.0 が追加されます。したがって、1 組の点は常に 0.5,1.5,2.5,3.5 などと呼ばれます。唯一の例外は、散布図とバブルチャートです。

正しい位置に行を挿入した後、行のスタイルを設定して行を非表示にすることができ、アノテーションが追加されます。



グラフに Point4 を追加する

```
//4 番目のポイントの上にラベルを追加すると、 "ポイント 4"
ITrendLine tr1 = hDataLines.newTrendLine(ITrendLine.VERTICAL_LINE, 1, "Point 4");
tr1.setTitleVisibleInLegend(false);    //凡例にタイトルを表示しない
tr1.setLineValue(3.5);    //4 番目のデータポイントの垂直線
tr1.setLineStyle( ((255*256) + 255) *256); //行を非表示にする
hDataLines.add(tr1);

//行に注釈を追加する
IAnnotationSet hAnnotation = chart.gethAnnotations();
IAnnotation annoTr1 = hAnnotation.newAnnotation(“Point 4”, tr1);

//annoTr1 をその行に追加すると、グラフに表示されます
tr1.addAnnotation(annoTr1);
```

チャートまたはデータポイントの値のスケールが変わると、ラベルは、常にデータポイントの最上部に表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.11 付録 B. チャートデータの取得

付録 B と付録 C には、デザイナを使用せずに 0 からグラフを作成するための情報が含まれています。この方法は通常、グラフを展開および保守するのをより困難にするため、お勧めしません。しかし、状況によってはこの方法でグラフを作成する必要もあります。

次のコードは、**QbChart** コンストラクタの例です。レポートと同様に、多数のグラフコンストラクタが利用可能です。典型的な **QbChart** コンストラクタには、少なくとも 4 つパラメータが必要です。新しいグラフを作成するには、グラフの次元、グラフの種類、入力データソースの情報、およびデータ列とグラフのそれぞれの列のマッピングを指定する必要があります。

```
QbChart(java.applet.Applet applet, int dimension, int chartType, IDatabaseInfo dbinfo, IColumnMap cmap, java.lang.String template)
```

テンプレートは必須ではなく（null でも可能）、グラフをアプレットに表示する場合にのみアプレットが必要ですが、他の 4 つのパラメータには常に意味のある情報が含まれている必要があります。この付録では、データソースパラメータと、さまざまなデータソースへの接続に使用されるメソッドについて詳しく説明します。付録 C では、ディメンション、チャートのタイプ、カラムのマッピング、実際のチャートの作成について説明します。付録 C には、実践的な例も含まれています。

グラフのデータは、レポート（グラフがレポートに挿入されている場合）または独立したデータソースから取得できます。以下のセクションでは、独立したデータソースに接続してグラフを生成するためのデータを取得します。

グラフを作成するために使用されるデータは、いくつかの異なるタイプのソースのうちの 1 つからフェッチされ得ます。

これらのソースは、以下のとおりです。

- JDBC ドライバを使用して、ローカルまたはリモートデータベースからデータを取得します。
- プレーンテキスト（ASCII）形式のデータベースレコードを含む、プレーンテキストファイルからデータを読み取ります。この形式のファイルは、ほとんどのデータベースプログラムで生成できます。
- スプレッドシートモデルからデータを読み込みます。
- データセットをメモリ内の配列として渡します。
- API を使用して独自のデータを渡します。
- EJB データソースからデータを取得します。
- 複数のデータソースからマージします。

このセクションの残りの部分では、データ抽出の上記の方法をより詳細に説明します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.11.1 データベースのデータ

Chart API の強力な機能の 1 つは、JDBC を介してデータベースから、直接データをフェッチする機能です。このアプローチでは、データベースへの接続に必要な情報と SQL 文の正確な形式を指定するだけです。したがって、JDBC ドライバが使用可能である場合、プログラムは事実上すべてのデータベースに接続できます。

データベースとクエリ情報は、グラフを作成する前に **DBInfo** オブジェクトに格納する必要があります。次のコードを使用して、**DBInfo** オブジェクトをインスタンス化します。

```
DBInfo dbinfo = new DBInfo(

                "jdbc:odbc:ODBCDatabase",          //URL
                "sun.jdbc.odbc.JdbcOdbcDriver",    //JDBC ドライバ
                "myName",                          //ユーザ名
                "myPassword",                      //パスワード
                "select * from sales");            //SQL
```

DBInfo はインタフェース **IDatabaseInfo** を実装しているため、次の **QbChart** コンストラクタで **DBInfo** オブジェクトを使用できます。

```
public QbChart(Applet parent, int dimension, int chartType, IDatabaseInfo dbinfo, ColInfo colMap, String templateFile);
```

場合によっては、EspressReport にデータベース情報（ユーザ ID、パスワード、場所、ドライバ、クエリなど）、全体を渡すことを望まない場合もあります。自分で接続し、クエリの結果セットを直接 API に提供することができます。これは、生成した **ResultSet** オブジェクトから **QueryResultSet** オブジェクトを作成し、その **QueryResultSet** オブジェクトを **DBInfo** オブジェクトではなくデータソースとして渡すことによって実行できます。

```
//java.sql.ResultSet オブジェクトの resultSet から QueryResult オブジェクトを作成します
QueryResultSet queryResultSet = new QueryResultSet(resultSet);

//次のコンストラクタで IResultSet データの代わりに queryResultSet を使用します
QbChart(java.applet.Applet applet, int dimension, int chartType, IResultSet data, IColumnMap cmap, java.lang.String template)
```

上記の例では、**QueryResultSet** クラスのインスタンスが API に渡された **ResultSet** オブジェクトから作成され、このインスタンスが **QbChart** コンストラクタに渡されて、グラフオブジェクトが作成されます。このシナリオでは、グラフは更新できません。グラフを更新するには、データベースに再度接続し、**ResultSet** オブジェクトをグラフに渡して自分で更新する必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.11.1.1 JNDI

EspressReport では、JNDI ソースからデータを取得することもできます。JNDI データソースは、データベースデータソースのように扱われ、同じ機能をサポートします。JNDI データソースを使用すると、異なる環境間でグラフを簡単に移行できます。両方の環境のデータソースが、同じルックアップ名でセットアップされている場合、グラフは変更なしで移行できます。

JNDI データソースに接続するには、アプリケーションサーバにデータソースをデプロイする必要があります。接続を成功させるには、**INITIAL_CONTEXT_FACTORY** および **PROVIDER_URL** も指定する必要があります。Tomcat にデプロイされた JNDI データソースに接続する時は、**INITIAL_CONTEXT_FACTORY** および **PROVIDER_URL** を指定する必要はありません（ただし、EspressManager は Tomcat 環境下でサーブレットとして実行する必要があります）。

```
//JNDI ルックアップ名とクエリを使用して DBInfo オブジェクトを作成する
String JNDIName = "java:comp/env/jdbc/TestDB";
String query = "select * from testdata";

//EspressManager が存在するため環境ハッシュテーブルは Tomcat では空です
//Tomcat コンテキスト内で実行し、他のアプリケーションサーバが使用されている場合は、
//INITIAL_CONTEXT_FACTORY と PROVIDER_URL を設定する必要があります
Hashtable env = new Hashtable();
DBInfo dbInfo = new DBInfo(JNDIName, query, env);
```

上記の例では、クラス **DBInfo** のインスタンスは、プログラムが JNDI データソースに接続してデータを取得するために必要な情報を提供します。グラフを作成するには、**IDatabaseInfo** を含む次のコンストラクタを使用します。

```
public QbChart(Applet parent, int dimension, int chartType, IDatabaseInfo dbinfo, ColInfo colMap, String templateFile);
```

5.11.2 データファイル（TXT/DAT/XML）のデータ

グラフを生成するためのデータは、データファイルからインポートすることもできません。XML ファイルと同様にテキストファイルでも構いません。グラフデータファイル（.dat 拡張子付き）は、それぞれの行が 1 つのレコードを表し、データ型とフィールド名をそれぞれ含む最初の 2 行を除いて、プレーンテキストファイルです。次に、4 つのレコードを含むデータファイルの例を示します。各レコードには、3 つのフィールドがあります。最初の行は各フィールドのデータ型を指定し、2 行目はフィールド名を指定します。

```
string, date, decimal
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
Name, Day, Volume
"John", "1997-10-3", 32.3
"John", "1997-4-3",   20.2
"Mary", "1997-9-3", 10.2
"Mary", "1997-10-04", 18.6
```

コンマ文字の使用はオプションですが、ほとんどのデータベースプログラムは、テキスト形式の表からレコードをエクスポートするときに、最初の 2 行を除くこの形式のファイルを出力できます。その結果、データベース用の JDBC ドライバがなくても、グラフを迅速に作成することができます。データベースからデータをテキスト形式でエクスポートするだけで、API を使用してプログラムで読み取ることができます。

XML 形式の場合、使用は次のとおりです。

```
<EspressData>

        <DataType>string</DataType>
        <DataType>date</DataType>
        <DataType>decimal</DataType>
        <FieldName>Name</FieldName>
        <FieldName>Day</FieldName>
        <FieldName>Volume</FieldName>
        <Row>
                <Data>John</Data>
                <Data>1997-10-3</Data>
                <Data>32.3</Data> </Row>
        <Row>

          <Data>John</Data>
          <Data>1997-4-3</Data>
          <Data>20.2</Data>
    </Row>
    <Row>
          <Data>Mary</Data>
          <Data>1997-9-3</Data>
          <Data>10.2</Data>
    </Row>
    <Row>
          <Data>Mary</Data>
          <Data>1997-10-4</Data>
          <Data>18.6</Data>
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
    </Row>
</EspressData>
```

詳細は、XML および XBRL ファイルのデータを参照してください。

使用するテキストファイルを指定するのは、非常にシンプルです。次のコンストラクタを使用し、変数 filename をデータファイルのファイルパス（相対パスまたはフルパス）に置き換えます。サーバに接続する場合は、相対パスは EspressReport のルートディレクトリを基準にする必要があります。それ以外の場合、パスは現在の作業ディレクトリからの相対パスになります。また、**fileType** パラメータを、**QbChart.DATAFILE**、**QbChart.QUERYFILE**、または **QbChart.XMLFILE** のいずれかのオプションで置き換えます。

```
public QbChart(Applet parent, int dimension, int chartType, int fileType, String filename, boolean doTransposeData, ColInfo colMap, String templateFile);
```

同じコンストラクタは、サーブレットによって生成された XML データを渡すためにも使用できます。fileType を **QbChart.XMLFILE**、ファイル名をサーブレット URL（**http://localhost:8080/servlet/XMLDataGenerator** など）として指定できます。

5.11.3 XML データソースのデータ

上記に加えて、EspressReport では、データを検索して XML ファイルを照会することができます。XML データは事実上どんなフォーマットでも構いませんが、XML データと共に DTD ファイルまたは XML スキーマを指定する必要があります。次のコードは、XML クエリを設定する方法を示しています。

```
//XML データソースを設定する
String xmlfilename = "Inventory.xml";
String xmlcondition = "/Inventory/Category/Product/ProductID < 45";

XMLFieldInfo[] fields = new XMLFieldInfo[5];
fields[0] = new XMLFieldInfo(new String[] {"Inventory", "Category", "Product"}, "ProductID");
fields[0].setAttributeDataType(DTDDataType.INT);

fields[1] = new XMLFieldInfo(new String[] {"Inventory", "Category", "Product", "ProductName"});

fields[2] = new XMLFieldInfo(new String[] {"Inventory", "Category", "Product", "UnitPrice"});
fields[2].setElementDataType(DTDDataType.DOUBLE);

fields[3] = new XMLFieldInfo(new String[] {"Inventory", "Category", "Product", "UnitsInStock"});
fields[3].setElementDataType(DTDDataType.INT);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
fields[4] = new XMLFieldInfo(new String[] {"Inventory", "Category", "Product", "ShipDate"});
fields[4].setElementDataType(DTDDataType.DATE);
fields[4].setDateFormat(XMLDataTypeUtil.YYYY_MM_DD);

XMLFileQueryInfo xmlInfo = new XMLFileQueryInfo(xmlfilename, fields, xmlcondition, fields);
```

XMLFieldInfo インスタンスは、2 つのコンストラクタのいずれかを使用して作成されます。xml フィールドを選択するために使用される最初のコンストラクタでは、ターゲットフィールドにつながる階層の各 xml タグを指定する、String 配列を渡す必要があります。上記の例では、fields[1-4]は最初のコンストラクタを使用して作成されます。XML 属性を選択するために使用される 2 番目のコンストラクタには、2 つのパラメータが含まれています。String 配列パラメータに加えて、2 番目のコンストラクタには、属性名に別の String も必要です。上記の例では、**ProductID** が **Product** の属性であるため、fields[0]はこのコンストラクタを使用して作成されます。

また、String 以外のフィールドについては、データ型を明示的に設定する必要があります。**XMLFileQueryInfo** インスタンスを作成したら、次のコンストラクタを使用して **QbChart** を作成できます。

```
public QbChart(Applet applet, int dimension, int chartType, XMLFileQueryInfo xmlInfo, boolean doTranspose, int[] transposeColumns, IColumnMap colMap, String templateFile);
```

XML データを、データソースとして使用する場合、XML ファイルの代わりに XML ストリームを渡すこともできます。XML ストリームを渡すには、XML データファイル名の代わりに、XML データを含むバイト配列を渡します。

上記の例では、XMLFileQueryInfo コンストラクタでバイト配列(xmlByteArray というバイト配列など)を介して、XML ストリームを渡すことができます。

```
XMLFileQueryInfo xmlInfo = new XMLFileQueryInfo(xmlByteArray, fields, xmlCondition, fields);
```

5.11.4 メモリ内の配列に渡されたデータ

API を使用すると、入力で他を配列としてメモリに直接渡すことができます。これは、インタフェース IResultSet（quadbase.until パッケージで定義されています）によって可能になります。このインタフェースは、表形式でデータを読み取るために使用され、JDBC 結果セットに使用される、**java.sql.ResultSet** インタフェースとよく似ています（しかし、それに比べてはるかに簡単です）。ユーザは、**IResultSet**（次のセクションで説明します）の独自の実装を提供することも、EspressReport が提供する **IResultSet** を使用することもできます。最も簡単な実装は、**DbData** クラスによって提供されます（IResultSet 実装を提供する他のクラスは QueryResultSet と StreamResultSet です）。グラ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フに必要なすべてのデータをメモリに収めることができれば、1 行のコードで配列を **DbData** の引数として渡すことができます。**DbData** には、3 つのコンストラクタがあります。

- **DbData(java.lang.String s)** – HTML ページからデータ値の引数を解析して **DbData** を構築します
- **DbData(java.lang.String[] fieldName, java.lang.Object[][] records)** – 新しい **DbData** クラスを構築します
- **DbData(java.lang.String[] dataType, java.lang.String[] fieldName java.lang.String[][] records)** – 新しい **DbData** クラスを構築します

この例では、次のコンストラクタを使用します。

```
public DbData(String dataType[], String fieldName[], String records[][]);
```

これは、データファイルからデータを読み込むのと同様の構成です。ここでは、最初の引数はデータ型（データファイルの最初の行）を表し、2 番目の引数はフィールド名（2 行目）を表します。3 番目の引数、records[][]は、レコードの配列を提供します。次は、どのように動作するかを示しています。

```
String dataType[] = {"varchar", "decimal"};
String fieldName[] = {"People", "Sales"};
String records[][] = {{"Peter", "93"}, {"Peter", "124"},
                      {"John", "110"}, {"John", "130"},
                      {"Mary", "103"}, {"Mary", "129"}};

DbData data = new DbData(dataType, fieldName, records);
```

グラフを作成するには、次の **QbChart** コンストラクタを使用します。

```
public QbChart(Applet parent, int dimension, int chartType, IResultSet data, ColInfo colMap, String templateFile);
```

5.11.5 カスタム実装で渡されたデータ

柔軟性を最大限に高めるために、必要な方法でデータセットを取得して準備し、グラフエンジンに渡すことができます。クラスファイルをデータソースとして渡すには、クラスファイルで IDataSource インタフェースを実装する必要があります。**IDataSource** を実装し、クラスファイルにデータソースとして渡すコードの例を次に示します。

```
public class ChartCustomClassData extends Applet implements IDataSource {

    //引数としてデータを渡すための DbData の設定
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        String dataType[] = {"string", "String", "double"};
        String fieldName[] = {"Destination", "Time", "Price"};
        String records[][] = {{"Mayfair", "13:43", "3.50"},
                {"Bond Street", "13:37", "3.75"},
                {"RickmansWorth", "13:12", "5.25"},
                {"Picadilly", "13:24", "3.00"}};
        DbData data = new DbData(dataType, fieldName, records);

        public IResultSet getResultSet()
        {
                return data; }

}
```

フルソースコード

上記の例では、データ（**DbData** インスタンス）が作成され、メモリに格納されます。**getResult()**メソッドが呼び出されると、IResultSet インタフェースを実装する **DbData** オブジェクトが返されます。この方法でデータを作成する必要はないことに注意してください。**IResultSet** を実装するオブジェクトを返すことができる限り、任意のデータソースからデータを取得できます。グラフを作成するには、次のコンストラクタを使用します。

```
public QbChart(Applet parent, int dimension, int chartType, int fileType, String filename, ColInfo colMap, String templateFile);
```

カスタムクラスファイルの場合は、fileType を **QbChart.CLASSFILE** にファイル名をクラスファイルの名前に設定します。

データソースとして独自のクラスファイルを渡していて、EspressManager を使用している場合、クラスファイルは EspressManager の CLASSPATH からアクセス可能でなければならないことに注意してください。

また、グラフのデータソースとしてパラメータ化されたクラスファイルを渡すこともできます。パラメータは実行時にユーザから取得され、データがフェッチされ、グラフの生成に使用されます。これは、スタンドアロングラフ設定でのみ機能することに注意してください。

```
public class ParameterizedClassFile implements IParameterizedDataSource {
        public IQueryInParam[] getParameters()
        {
                SimpleQueryInParam[] params = new SimpleQueryInParam[1];
                params[0] = new SimpleQueryInParam("param2", "Enter the price:", false, null,
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
null, Types.INTEGER, new Integer(2), null);
                return params;
        }

        public IResultSet getResultSet(IQueryInParam[] params) {
                double price = 3.5;
                if ((params != null) && (params.length >= 1))
                {
                        Object obj = params[0].getValue();
                        if ((obj != null) && (obj instanceof Integer)) price =
((Integer)obj).intValue();
                }
                String dataType[] = {"string", "String", "double"};
                String fieldName[] = {"Destination",  "Time", "Price"};
                String records[][] = {{"Mayfair",  "13:43", price+""},
                                {"Bond Street", "13:37", price+""},
                                {"Rickmansworth ", "13:12", price+""},
                                {"Picadilly",  "13:24", price+""}};
                return new DbData(dataType, fieldName, records);
        }
}
```

フルソースコード

パラメータ化されたクラスファイルをデータソースとして使用する場合、通常、パラメータダイアログボックスは選択肢のないテキストボックスです。ただし、IQueryParamaValuesProvider インタフェースを実装することによって、選択肢の選択肢を指定することができます（ドロップダウンボックスから使用可能)。

```
public class CustomParamClassFile implements IParameterizedDataSource {

        public IQueryInParam[] getParameters() {
                mySimpleQueryMultiValueInParam[] params = new
mySimpleQueryMultiValueInParam[1];
                params[0] = new mySimpleQueryMultiValueInParam("region", "Select Region(s):",
true, "Customers", "Region", Types.VARCHAR, "East", null);
                return params;
        }

        public IResultSet getResultSet(IQueryInParam[] params) {
                QueryResultSet data = null;
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
		ResultSet rs = null;

		String paramValue = "'East'";
		if ((params != null) && (params.length >= 1)) {
			Vector selectedValues = null;
			if (params[0] instanceof IQueryMultiValueInParam)
				selectedValues = ((IQueryMultiValueInParam)params[0]).getValues();

			for (int i = 0; i < selectedValues.size(); i++) {
				if ((selectedValues.get(i) != null) && (selectedValues.get(i) instanceof String)) {
					if (i == 0) paramValue = "'" + (String)selectedValues.get(i) + "'";
					else paramValue += ",'" + (String)selectedValues.get(i) + "'";
				}
			}
		}
		String myQuery = "select cu.region, c.categoryname, count(o.orderid), sum(od.quantity), sum(p.unitprice * od.quantity) from customers cu, categories c, products p, orders o, order_details od where cu.customerid = o.customerid and c.categoryid = p.categoryid and p.productid = od.productid and o.orderid = od.orderid and cu.region in (" + paramValue + ") group by cu.region, c.categoryname";

		try {
			Class.forName("org.hsqldb.jdbcDriver");

			String url = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
			Connection conn = DriverManager.getConnection(url, "sa", "");
			Statement stmt = conn.createStatement();

			rs = stmt.executeQuery(myQuery);
			data = new QueryResultSet(rs);
			//conn.close();
		} catch (Exception ex) {
			ex.printStackTrace();
		}
		return data;
	}
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
public class mySimpleQueryMultiValueInParam extends SimpleQueryMultiValueInParam implements IQueryParamValuesProvider {

	public String paramName, promptName, tableName, colName;
	boolean mapToColumn;
	int sqlType;
	Object defaultValue;
	Vector values;
	public mySimpleQueryMultiValueInParam(String paramName, String promptName, boolean mapToColumn, String tableName, String colName, int sqlType, Object defaultValue, Vector values) {
		super(paramName, promptName, mapToColumn, tableName, colName, sqlType, defaultValue, values);
	}

	public Vector getSelectionChoices() {
		System.out.println("getSelectionChoices called");
		try {
			Class.forName("org.hsqldb.jdbcDriver");

			String url = "jdbc:hsqldb:help/examples/DataSources/database/woodview";
			Connection conn = DriverManager.getConnection(url, "sa", "");
			Statement stmt = conn.createStatement();
			String query = "SELECT DISTINCT " + getColumnName() + " FROM " + getTableName();
			ResultSet rs = stmt.executeQuery(query);
			Vector v = new Vector();

			while (rs.next()) {
				switch (getSqlType()) {
				case Types.INTEGER:

					v.add(new Integer(rs.getInt(1)));
					break;

				case Types.VARCHAR:

					v.add(rs.getString(1));
					break;
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                                        }
                                }
                                stmt.close();
                                conn.close();

                                return v;
                        } catch (Exception ex) {
                                ex.printStackTrace();
                        }
                        return null;
                }

        }

}
```

フルソースコード

5.11.6 スプレッドシートモデルのデータ

グラフは、モデルビューコントローラ（MVC）アーキテクチャのスプレッドシートモデルへのビューとして機能することができました。自動的に、スプレッドシートのデータを読み込んでプロットします。グラフはスプレッドシートモデルのリスナーとして登録され、スプレッドシート data の変更が通知されると、自動的に更新されます。ISpreadSheetModel インタフェースを実装するクラスのインスタンスである、スプレッドシート（Java）オブジェクトがユーザによって提供されます。イベントクラス **SpreadSheetModelEvent** は、モデルのリスナーにデータの変更を通知するために使用されます。次の例は、スプレッドシートモデルをグラフに使用する方法を示しています。ISpreadSheetModel インタフェースを実装するユーティリティクラス SimpleSpreadSheet（quadbase.util パッケージで定義されている）を使用します。

このメソッドは、プロットされるデータを含むライブ Java プログラムスプレッドシートオブジェクトにのみ適用されるため、スプレッドシート形式のデータベースまたはデータファイルからデータを読み込むメソッドを補完します。

```
String[] columnVals = {"quantity", "high"};
String[] rowVals = {"coffee", "Soft Drinks", "Fruit Juice", "Water", "beer"};
Double[][] vals = { {new Double(1), new Double(30)},

                        {new Double(3), new Double(33)},
                        {new Double(7), new Double(34)},
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                    {new Double(8), new Double(40)},
                    {new Double(8), new Double(40)} };

//quadbase.util.SimpleSpreadSheet を参照してください
sss = new SimpleSpreadSheet(rowVals, columnVals, vals);
```

ここでデータはスプレッドシート形式で与えられ、次の表のようになります。

	quantity	high
coffee	1	30
Soft Drinks	3	33
Fruit Juice	7	34
Water	8	40
beer	8	40

スプレッドシート形式のデータ

EspressReport は、次にデータを変換します（これは内部的に行われます)。その結果、データは次のようになります。

coffee	**quantity**	1
coffee	**high**	30
Soft Drinks	**quantity**	3
Soft Drinks	**high**	33
Fruit Juice	**quantity**	7
Fruit Juice	**high**	24
Water	**quantity**	8
Water	**high**	40
beer	**quantity**	8
beer	**high**	40

トランスポーズ後のデータ

次に、変換されたデータに関してのグラフのデータマッピングが行われます。

QbChart オブジェクトを作成するには、次のコンストラクタを使用します。

```
public QbChart(Applet applet, int dimension, int chartType, ISpreadSheetModel spreadsheet, IColumnMap colMap, String templateFile);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.11.7 Enterprise Java Beans (EJBs) のデータ

エンティティ Bean から直接データをクエリできるようにすることで、EJB データソースからグラフにデータを渡すことができます。EJB をデータソースとして追加するには、最初に EJB をアプリケーションサーバにデプロイし、適切なスタブクラスを含むクライアント JAR ファイルをクラスパスに追加する必要があります（または API を使用する場合は、EspressManager バッチファイルの**-classpath** 引数 EspressManager と組み合わせて）。

レポートとは異なり、接続情報を格納するために **EJBInfo** オブジェクトを作成する必要はありません。次のコンストラクタに直接名前を指定して、**QbChart** オブジェクトを作成します。

```
public QbChart(Applet applet,
                int dimension,
                int chartType,
                String jndiName,
                String homeName,
                String remoteName,
                String selectedMethodName,
                Object[] selectedMethodParamVal,
                IColumnMap cmap,
                String template);
```

フルソースコード

上記のコードは、アプリケーションとしてもアプレットとしても実行できます。

このクラスのコンストラクタは、次のとおりです。

```
public QbChart(Applet applet, int dimension, int chartType, String jndiName, String homeName, String remoteName, String selectedMethodName, Object[] selectedMethodParamVal, IColumnMap cmap, String template);
```

5.11.8 複数のデータソースのデータ

EspressReport は、複数のデータソースをまとめて併合する機能も提供します。DataSheet オブジェクトは、データファイル、データベース、**IResultSet** などのデータソースの任意の組み合わせから作成できます（**引数として渡されたデータ**を参照）。**QbChart** コンストラクタを使用して、**DataSheet** の配列からグラフオブジェクトを作成できます。

次のプログラム例は、上記の例のデータをどのように組み合わせるかを示しています。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
//データをマージするために使用される DataSheet オブジェクトの宣言
DataSheet dataSheet[] = new DataSheet[3]; //データベースの宣言（データベースセクションからのデータ）
DBInfo dbinfo = new DBInfo("jdbc:quadbase:/machine/schema",
"quadbase.jdbc.QbDriver", "myName","myPassword","select * from sales"); //引数として渡されたデータの宣言（引数セクションとして渡されたデータ）
String dataType[] = {"varchar", "decimal"};
String fieldName[] = {"People", "Sales"};
String records[][] = {{"Peter", "93"}, {"Peter", "124"},
                      {"John" , "110"}, {"John" , "130"},
                      {"Mary" , "103"}, {"Mary" , "129"}};
DbData data = new DbData(dataType, fieldName, records); //データファイルからデータシートを作成する（テキストファイルセクションのデータ）
//DataSheet(Applet アプレット String dataFile)
dataSheet[0] = new DataSheet(this, "help/examples/data/Columnar1.dat"); //データベースからデータシートを作成する（データベースセクションからのデータ）
//DataSheet(Applet アプレット、IDatabaseInfo dbInfo)
dataSheet[1] = new DataSheet(this, dbinfo); //IResultSet からデータシートを作成する（引数セクションとして渡されたデータ）
//DataSheet(Applet アプレット、IResultSet data)
dataSheet[2] = new DataSheet(this, data); //QbChart を作成する
QbChart chart = new QbChart

                (this,                  //アプレット
                QbChart.VIEW2D,          //2 次元
                QbChart.PIE              //円グラフ
                dataSheet,               //データシート
                colInfo,                //カルム情報
                null);                  //テンプレートなし
```

DataSheet オブジェクトを作成した後は、DataSheet API を使用して変更することができます（行の値の追加/削除/更新）。**IResultSet** からデータをマージすると、データを更新できません。

5.11.9 スプレッドシート形式のデータ

上記の方法を用いて得られたデータは、スプレッドシートとして解釈することができます。スプレッドシートには、テーブルのようなグリッド構造があります。スプレッドシートの左端の列と最初の行には、レベル（または見出し）が含まれています。各セルは、その行ラベル、列ラベル、およびセル値（各行が異なるデータポイントを表す通常の表表記とは対照的に）を含む個別の出たポイントを表します。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次の例を考えてみます。

Date	Nasdaq	Dow	SP500
"12/04/2000"	2304	10503	1240
"12/05/2000"	2344	10486	1239
"12/06/2000"	2344	10458	1224
------	------	------	------

データベースのデータが、図のように 4 つの列として配置されているとします。スプレッドシート形式でデータを扱う場合、3 つのインデックス（Nasdaq、Dow、および SP500）をカテゴリ軸として Date を 3 行としてプロットすることができます。

この問題に対処するために、いくつかの **QbChart** コンストラクタが用意されています。3 つの主なものは以下のとおりです。

```
QbChart(java.applet.Applet applet, int dimension, int chartType, IResultSet data, boolean

        doTransposeData, IColumnMap cmap, java.lang.String template);

QbChart(java.applet.Applet applet, int dimension, int chartType, int fileType,

        java.lang.String filename, boolean doTransposeData, IColumnMap cmap, java.lang.String
        template);

QbChart(java.applet.Applet applet, int dimension, int chartType, IDatabaseInfo dbinfo, boolean

        doTransposeData, IColumnMap cmap, java.lang.String template);
```

データがスプレッドシート形式であることを指定するには、**doTransposeData** フラグを true に設定する必要があります。

上記のコンストラクタは、2 番目の列から最後の列までの全ての列を 3 列の表に変換します。したがって、2 列目以降のデータ型は数値でなければなりません。それを行わないと、変換は成功しません。

完全な変換と同様に、**QbChart** には選択的変換を可能にするいくつかのコンストラクタがあります。3 つの主なものを以下に示します。

```
QbChart(java.applet.Applet applet, int dimension, int chartType, IResultSet data, boolean

        doTransposeData, int[] transposeCol, IColumnMap cmap, java.lang.String template);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
QbChart(java.applet.Applet applet, int dimension, int chartType, int fileType,

        java.lang.String filename, boolean doTransposeData, int[] transposeCol, IColumnMap cmap,
        java.lang.String template);

QbChart(java.applet.Applet applet, int dimension, int chartType, IDatabaseInfo dbinfo, boolean

        doTransposeData, int[] transposeCol, IColumnMap cmap, java.lang.String template);
```

変換する列を指定するには、**transposeCol** の列インデックスを含む配列を渡し、**doTransposeData** フラグを true に設定する必要があります。

5.11.10 データ変換

EspressReport では、さまざまなデータソースからデータを取得できます。また、データを目的のグラフタイプに渡す前に、データを transpose する（つまり、列名がデータの一部になるようにデータを変換する）こともできます。

データが変換されると、元のデータ列（変換で使用される）がデータセットから削除され、2 つの新しい列が最後に追加されます。これらの 2 つの新しい列には、変換列名と元の列の値が含まれます。たとえば、元のデータセットに 5 つの列があり、3 つが変換用に選択されている場合、新しいデータセットは 5 から 3（変換列の数）+2（追加された新しい列)、または 4 列です。

データ列を変換する時は、注意すべき 2 つの点があります。

- データ列はすべて同じデータ列でなければなりません
- グラフの列マッピングに渡される列インデックスは、元のデータセットではなく新しいデータセットを参照します

以下のセクションでは、データの変換方法について説明します。

5.11.10.1 非選択変換

この章では、最初の列（列 0）を除くすべての列が変換されます。例えば、以下のデータがあるとします。

Product	January	February	March
Chairs	$3872.35	$3962.21	$4218.57
Tables	$6534.98	$6018.43	$5928.71

変換されたデータは、次のように表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Product	ColumnLabel	Value
Chairs	January	$3872.35
Chairs	February	$3962.21
Chairs	March	$4218.57
Tables	January	$4218.57
Tables	February	$6018.43
Tables	March	$5928.71

API を使用してデータを非選択的に転置するには、次のような **doTransposeData** boolean パラメータを持つ **QbChart** コンストラクタを使用します。

```
QbChart(java.applet.Applet applet, int dimension, int chartType, int fileType, java.lang.String filename, boolean doTransposeData, IColumnMap cmap, java.lang.String template)
```

フルソースコード
エクスポート後の結果

IColumnMap で渡される列インデックスは新しいデータセットを参照することに注意してください。

5.11.10.2 選択変換

この章では、変換する列を選択します。同じデータ型を共有する列のみを変換することができます。選択的変換では、変換される最初の列も選択できます。たとえば、以下のデータがあるとします。

Category	Product	January	January	March
Chairs	Side Chairs	$3872.35	$3962.21	$4218.57
Chairs	Arm Chairs	$2654.84	$1924.83	$2543.24
Tables	Round Tables	$6534.98	$6018.43	$5928.71
Tables	Rectangular Tables	$10227.32	$9721.83	$11748.93

数値列を変換した後、変換データは次のように表示されます。

Category	Product	ColumnLabel	Value
Chairs	Side Chairs	January	$3872.35
Chairs	Side Chairs	February	$3962.21
Chairs	Side Chairs	March	$4218.57
Chairs	Arm Chairs	January	$2654.84
Chairs	Arm Chairs	February	$1924.83
Chairs	Arm Chairs	March	$2543.24

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Tables	Round Tables	January	$6534.98
Tables	Round Tables	February	$6018.43
Tables	Round Tables	March	$5928.71
Tables	Rectangular Tables	January	$10227.32
Tables	Rectangular Tables	February	$9721.83
Tables	Rectangular Tables	March	$11748.93

API を使用してデータを選択的に変換するには、次のコンストラクタのように、変換対象の列のインデックスをとる **doTransposeData** ブール型パラメータと整数配列を持つ **QbChart** コンストラクタを使用します。

```
QbChart(java.applet.Applet applet, int dimension, int chartType, int fileType, java.lang.String filename, boolean doTransposeData, int[] transposeCol, IColumnMap cmap, java.lang.String template)
```

フルソースコード
エクスポート後の結果

IColumnMap で渡される列インデックスは、新しいデータセットを参照することに注意してください。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.12 付録 C. チャートの作成

新しいグラフを作成するには、グラフの種類、グラフの次元、入力データソース情報、およびグラフテンプレートの列マッピングを指定する必要があります。この章では、さまざまなグラフの種類と、列のマップに使用されるメソッドを見ていきます。この章には、いくつかの完全に機能的な例もあります。特に記載がない限り、すべての例では、**<EspressReportInstall> /help/samples/DataSources/database** ディレクトリにある Woodview HSQL データベースを使用しています。例を実行するには、データベース HSQL JDBC ドライバ（**hsqldb.jar**）をクラスパスに追加する必要があります。ドライバは**<EspressReportInstall>/lib** ディレクトリにあります。

チャートは、2D または 3D にできます。2D または 3D を指定する定数は以下の通りです。

2D	**QbChart.VIEW2D**
3D	**QbChart.VIEW3D**

チャートには、次の種類があります。以下に、それらを選択する定数を示します。

縦棒グラフ	**QbChart.COL**
横棒グラフ	**QbChart.BAR**
折れ線グラフ	**QbChart.LINE**
エリアチャート	**QbChart.AREA**
円グラフ	**QbChart.PIE**
散布図	**QbChart.SCATTER**
スタック縦棒グラフ	**QbChart.STACKCOL**
スタック横棒グラフ	**QbChart.STACKBAR**
スタックエリアチャート	**QbChart.STACKAREA**
High-Low	**QbChart.HILOW**
HLCO チャート	**QbChart.HLCO**
100% Column	**QbChart.PERCENTCOL**
サーフェスチャート	**QbChart.SURFACE** (3D 限定)
バブルチャート	**QbChart.BUBBLE** (2D 限定)
オーバーレイチャート	**QbChart.OVERLAY** (2D 限定)
ボックスチャート	**QbChart.BOX** (2D 限定)
レーダーチャート	**QbChart.RADAR** (2D 限定)
ダイヤルチャート	**QbChart.DIAL** (2D 限定)
ガントチャート	**QbChart.GANTT** (2D 限定)
極座標グラフ	**QbChart.POLAR** (2D 限定)

各チャートの詳細については、チャートの種類ごとのデータマッピングを参照してください。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

各チャートタイプは、チャートオブジェクトを作成するために 2 つ（またはそれ以上）のデータ列をマッピングする必要があります。マッピングの詳細（およびここで使用されている用語の定義）については、チャートの種類ごとのデータマッピングを参照してください。

チャートテンプレートの列マッピングを定義するには、（**quadbase.ReportAPI** パッケージにある）**ColInfo** クラスが使用されます。

（データソースからの）列は、左から列「0」で始まる番号が付いています。**ColInfo** オブジェクトに渡される列の位置は、データテーブルに基づいています。**ColInfo** オブジェクトのパラメータは、デフォルトでは「-1」です。負の列の位置は、特定のパラメータが使用されていないことを示します。

Java でオブジェクトを作成することは、リソースを集中的に使用することです。多くのオブジェクトを作成しないことをお勧めします。リソースを節約する 1 つの方法として、**QbChart** オブジェクトを可能な限り再利用することです。たとえば、多くのユーザがウェブサイトで単純な柱状チャートを要求した場合、要求を受け取ったときに新しい **QbChart** オブジェクトを作成するのではなく、1 つの **QbChart** オブジェクトを、特定の要求ごとに ahrt のデータと属性を変更するだけで、オブジェクトを作成して再利用します。

5.12.1 縦棒/横棒/折れ線/エリア/円/オーバーレイチャート

縦棒グラフ/横棒グラフ/折れ線グラフ/エリアチャート/円グラフ/オーバーレイチャートは、**category**（カテゴリ）および **value**（値）の列が必要になります。最小要件はこの 2 つの列があるチャートです。また、必要に応じて、**series**（シリーズ）および **subvalue**（2 軸の値）を追加できます。**Subvalue** は 2 軸を参照します。2 軸にマッピングされる列が含まれます。

5.12.1.1 列マッピング

例えば、以下に示すデータがあるとします：

Order #	Product	Units Ordered	Units Shipped
12	Chair	15	12
12	Table	34	26
14	Chair	8	8
14	Table	23	14

オリジナルデータ

次のコードはカテゴリとして列 1（Product）、シリーズとして列 0（Order #）、値として列 3（Units Shipped）、サブ値として列 2（Units Ordered）を定義しています。

```
ColInfo colInfo = new ColInfo();
colInfo.category = 1;
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
colInfo.series = 0;
colInfo.value = 3;
colInfo.subvalue = 2;
```

5.12.1.2 チャートの作成

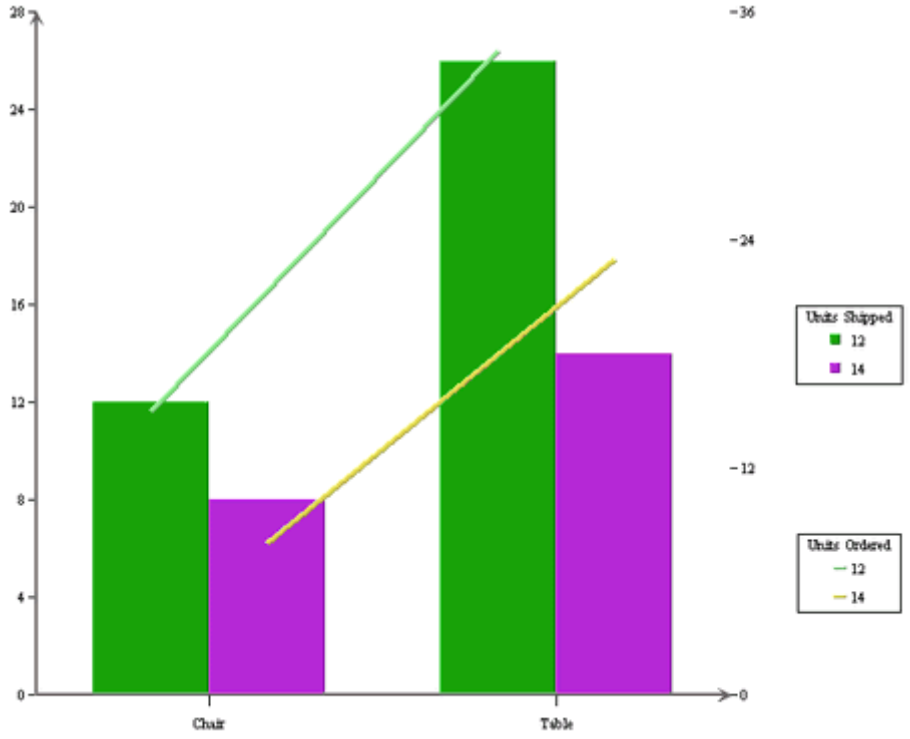
列グラフの作成は比較的簡単です。ColInfo 配列を設定する方法と、前のセクションでデータを取得する方法についてはすでに説明しました。次のコードは、前述の列グラフを作成する方法を示しています。

```
Component doDataFromArguments(Applet applet) {



    //EspressManager に接続しないでください
    QbChart.setEspressManagerUsed(false);

    //列マッピング
    ColInfo colInfo = new ColInfo();
    colInfo.category = 1;
    colInfo.series = 0;
    colInfo.value = 3;
    colInfo.subvalue = 2;

    String dataType[] = {"integer", "varchar", "decimal", "decimal"};
    String fieldName[] = {"Order #", "Product", "Units Ordered", "Units Shipped"};
    String records[][] = {{"12", "Chair","15", "12"}, {"12", "Table","34", "26"},
                          {"14" , "Chair","8", "8"}, {"14" , "Table","23", "14"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                        (applet,              //アプレット
                        QbChart.VIEW2D,       //2 次元
                        QbChart.COL,          //縦棒グラフ
                        data,                 //データ
                        colInfo,              //列情報
                        null);                //指定されたテンプレートがありません
    return chart;

}
```

フルソースコード

コードをアプレットまたはアプリケーションとして実行すると、下の図のような列グラフが表示されます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



作成されたチャートは、フォーマットはデフォルトになっています。

5.12.2 レーダーチャート

レーダーチャートの **CollInfo** パラメータは、**subvalue** (小数値)以外の列/横棒グラフ/線グラフ/範囲グラフのパラメータと同じです。2 軸を定義することはできません。

5.12.2.1 列マッピング

例えば、以下に示すデータがあるとします：

Order #	Product	Units Ordered
12	Chair	15
12	Table	34
12	Cabinet	21
12	Dresser	24
14	Chair	23
14	Table	23
14	Cabinet	16
14	Dresser	19

オリジナルデータ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次のコードでは、カテゴリに列 1 の Product、シリーズに列 0 の Order #、値に列 2 の Units Ordered を定義します。

```
ColInfo colInfo = new ColInfo();
colInfo.category = 1;
colInfo.series = 0;
colInfo.value = 2;
```

5.12.2.2 チャートの作成

次に示すコードは、前述した散布図の作成のデモンストレーションです。

```
Component doDataFromArguments(Applet applet) {
```

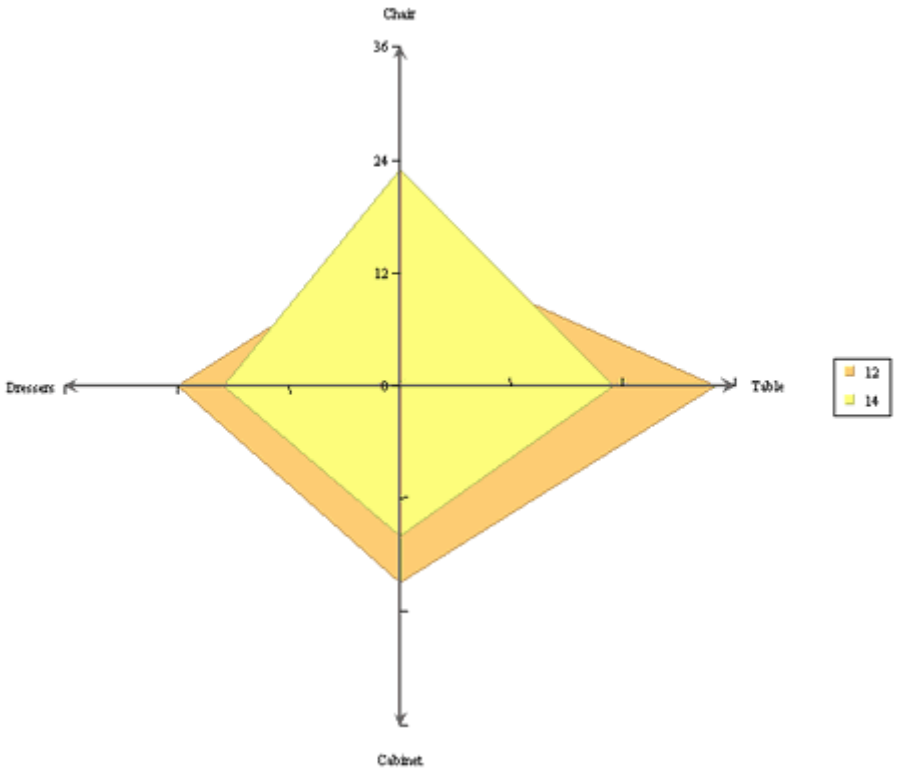
```
  //EspressManager には接続しないでください
  QbChart.setEspressManagerUsed(false);

  //列マッピング
  ColInfo colInfo = new ColInfo();
  colInfo.category = 1;
  colInfo.series = 0;
  colInfo.value = 2;

  String dataType[] = {"integer", "varchar", "decimal"};
  String fieldName[] = {"Order #", "Product", "Units Ordered"};
  String records[][] = {{"12", "Chair", "15"}, {"12", "Table", "34"},
                        {"12" , "Cabinet", "21"}, {"12" , "Dresser", "24"},
                       {"14" , "Chair", "23"}, {"14" , "Table", "23"},
                         {"14" , "Chair", "23"}, {"14" , "Table", "23"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                          (applet,              //アプレット
                          QbChart.VIEW2D,       //2 次元
                          QbChart.RADAR,        //レーダーチャート
                          data,                 //データ
                          colInfo,              //列情報
                          null);                //指定されたテンプレートがありません
    return chart;

}
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フルソースコード

コードをアプレットまたはアプリケーションとして実行すると、下の図のような 2 次元レーダーチャートが表示されます。



生成されたレーダーチャート

作成されたチャートはフォーマットされていないデフォルトのチャートです。

5.12.3 XY(Z)散布図

散布図は **xvalue** と **yvalue** の値となる列が必要になります。散布図のチャートを構築するための最低限の要件です。必要に応じて、シリーズと z 値（3 次元の散布図）軸を追加することもできます。

5.12.3.1 列マッピング

例えば、以下に示すデータがあるとします：

Season	High Temperature	Average Temperature	Low Temperature
Summer	110	101	96
Fall	103	85	78
Winter	85	75	67
Spring	93	88	81

オリジナルデータ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

次のコードでは、**xvalue** (x 値)として列 2 の Average Temperature、**yvalue** (y 値)として列 1 の High Temperature、**zvalue** (z 値)として列 3 の Low Temperature を使用しています。

```
ColInfo colInfo = new ColInfo();
colInfo.xvalue = 2;
colInfo.yvalue = 1;
colInfo.zvalue = 3;
```

5.12.3.2 チャートの作成

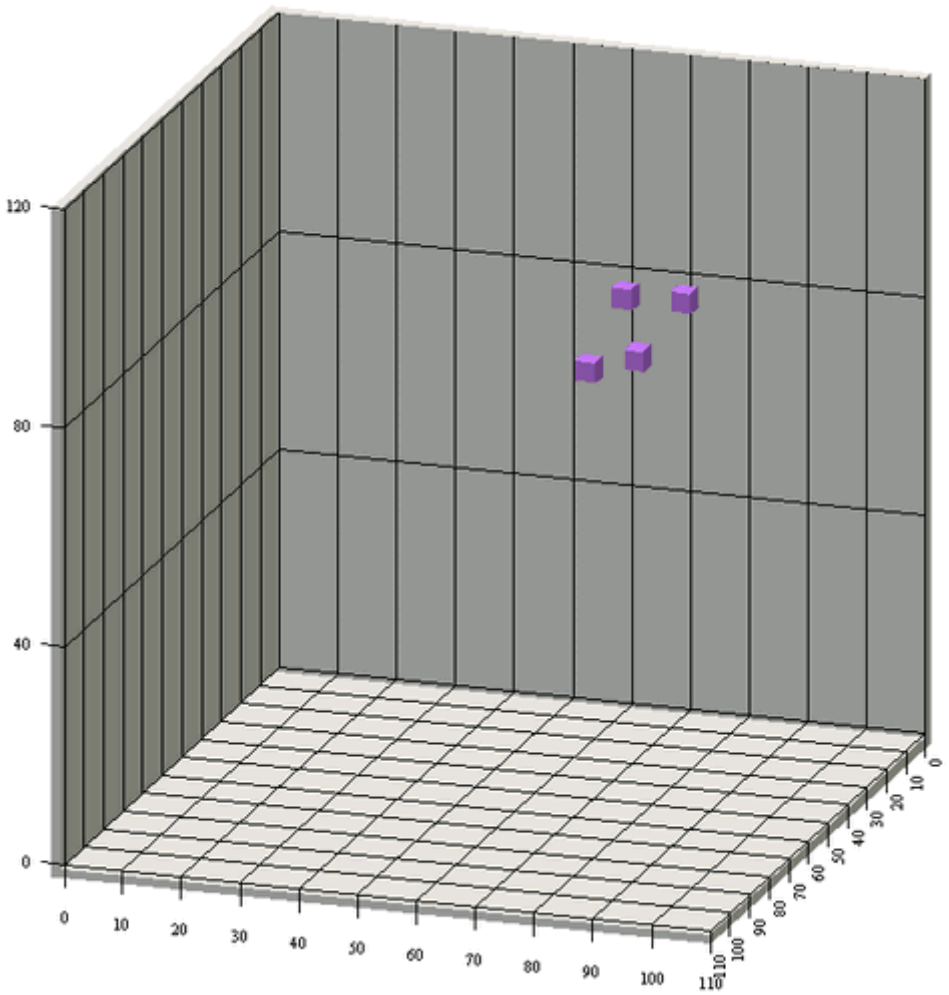
次に示すコードは、前述した散布図の作成のデモンストレーションです。

```
Component doDataFromArguments(Applet apple){

  //EspressManager には接続しないでください。
  QbChart.setEspressManagerUsed(false);

  //列マッピング
  ColInfo colInfo = new ColInfo();
  colInfo.xvalue = 2;
  colInfo.yvalue = 1;
  colInfo.zvalue = 3;

  String dataType[] = {"varchar", "integer", "integer", "integer"};
  String fieldName[] = {"Season", "High Temperature", "Average Temperature", "Low Temperature"};
  String records[][] = {{"Summer", "110", "101", "96"}, {"Fall", "103", "85", "78"},
                        {"Winter", "85", "75", "67"}, {"Spring", "93", "88", "81"}};
  DbData data = new DbData(dataType, fieldName, records);
  QbChart chart = new QbChart
                          (applet,              //アプレット
                          QbChart.VIEW3D,       //3 次元
                          QbChart.SCATTER,      //散布図
                          data,                 //データ
                          colInfo,              //列情報
                          null);                //指定されたテンプレートはありません
  return chart;
}
```

フルソースコード

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

コードをアプレットまたはアプリケーションとして実行すると、下の図のような 3 次元散布図が表示されます。



生成された散布図

作成されたチャートは、フォーマットされていないデフォルトのチャートです。

5.12.4 スタック縦棒/割合縦棒/スタック棒/スタックエリアチャート

縦棒グラフ/折れ線グラフ/エリアチャートのパラメータに加えて、スタック縦棒/100% Column/スタック横棒/スタックエリアチャートにも **SumBy** パラメータを設定する必要があります。スタック縦棒/100% Column/スタック横棒/スタックエリアチャートを作成するための最小要件は、カテゴリ、値、および **sumby** パラメータです。必要に応じて、シリーズと小数値を追加することもできます。

5.12.4.1 列マッピング

例えば、以下に示すデータがあるとします:

Day	Drink	Total	Average

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Day	Drink	Total	Average
Monday	Water	101	5.4
Monday	Coffee	85	2.4
Tuesday	Water	143	6.7
Tuesday	Coffee	92	2.5
Wednesday	Water	186	7.6
Wednesday	Coffee	121	4.2
Thursday	Water	173	6.3
Thursday	Coffee	75	1.1
Friday	Water	88	3.6
Friday	Coffee	193	5.9
Saturday	Water	152	7.3
Saturday	Coffee	57	1.6
Sunday	Water	194	8.8
Sunday	Coffee	25	0.6

オリジナルデータ

次のコードでは、列を Column 0（"Day"）、値を Column 2（"Total"）、sumby を Column 1（"Drink"）、subvalue を Column 3（"Average"）に設定します：

```
ColInfo colInfo = new ColInfo();
colInfo.category = 0;
colInfo.value = 2;
colInfo.subvalue = 3;
colInfo.sumBy = 1;
```

5.12.4.2 チャートの作成

次のコードは、前述のスタックエリアチャートを作成する方法を示しています。

```
Component doDataFromArguments(Applet applet) {


    //EspressManager に接続しないでください
    QbChart.setEspressManagerUsed(false);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

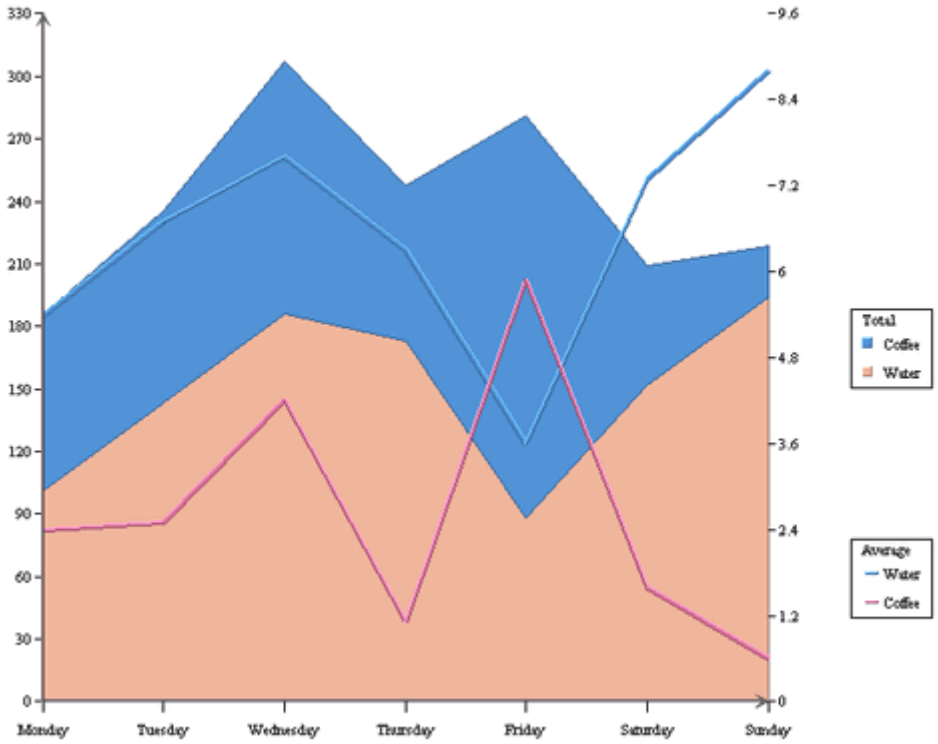
```
    //列マッピング
    ColInfo colInfo = new ColInfo();
    colInfo.category = 0;
    colInfo.value = 2;
    colInfo.subvalue = 3;
    colInfo.sumBy = 1;

    String dataType[] = {"varchar", "varchar", "integer", "double"};
    String fieldName[] = {"Day", "Drink", "Total","Average"};
    String records[][] = {{"Monday", "Water", "101","5.4"}, {"Monday", "Coffee", "85","2.4"},
                          {"Tuesday", "Water", "143","6.7"}, {"Tuesday", "Coffee", "92","2.5"},
                          {"Wednesday", "Water", "186","7.6"}, {"Wednesday", "Coffee",
"121","4.2"},
                          {"Thursday", "Water", "173","6.3"}, {"Thursday", "Coffee", "75","1.1"},
                          {"Friday", "Water", "88","3.6"}, {"Friday", "Coffee", "193","5.9"},
                          {"Saturday", "Water", "152","7.3"}, {"Saturday", "Coffee", "57","1.6"},
                          {"Sunday", "Water", "194","8.8"}, {"Sunday", "Coffee", "25","0.6"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                                (applet,              //アプレット
                                QbChart.VIEW2D,       //2 次元
                                QbChart.STACKAREA,    //スタックエリアチャート
                                data,                 //データ
                                colInfo,              //列情報
                                null);                //テンプレート指定無し
    return chart;

}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、以下に示す 2 次元のスタックエリアチャートが生成されます:

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



生成されたスタックエリアチャート

作成されたチャートは、フォーマットされていないデフォルトチャートです。

5.12.5 ダイヤルチャート

ダイヤルチャートの **CollInfo** パラメータは、シリーズを除くレーダーチャートのパラメータと同じです。列を定義することはできません。

5.12.5.1 列マッピング

例えば、以下に示すデータがあるとします:

Product	Units Ordered
Chair	15
Table	34
Cabinet	21
Dresser	24

オリジナルデータ

次のコードでは、カテゴリを Column 0（”Product”)、列を Column 1（"Units Ordered"）に設定します:

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
ColInfo colInfo = new ColInfo();
colInfo.category = 0;
colInfo.value = 1;
```

5.12.5.2 チャートの作成

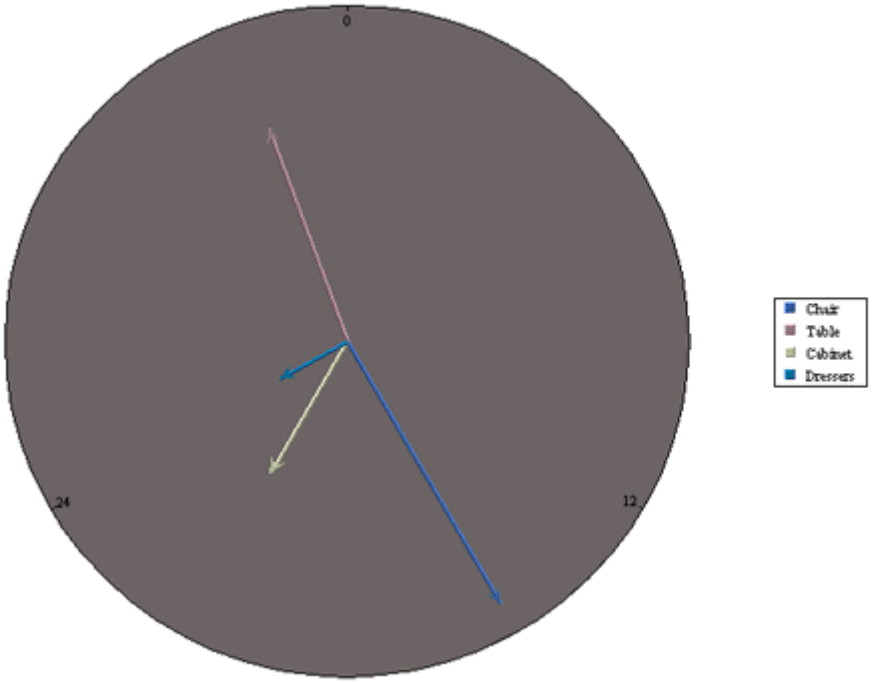
次のコードは、上記のダイヤルチャートを作成する方法を示しています。

```
Component doDataFromArguments(Applet applet) {


    //EspressManager に接続しないでください
    QbChart.setEspressManagerUsed(false);

    //列マッピング
    ColInfo colInfo = new ColInfo();
    colInfo.category = 0;
    colInfo.value = 1;

    String dataType[] = {"varchar", "integer"};
    String fieldName[] = {"Product", "Units Ordered"};
    String records[][] = {{"Chair", "15"}, {"Table", "34"},
                          {"Cabinet", "21"}, {"Dresser", "24"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                        (applet,            //アプレット
                        QbChart.VIEW2D,     //2 次元
                        QbChart.DIAL,       //ダイヤルチャート
                        data,               //データ
                        colInfo,            //列情報
                        null);              //テンプレート指定無し
    return chart;

}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、次のような 2 次元のダイヤルチャートが作成されます:

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



生成されたダイヤルチャート

作成されたチャートは、フォーマットされていないデフォルトチャートです。

5.12.6 ボックスチャート

ボックスチャートの **CollInfo** パラメータは、シリーズを除く縦棒/横棒/折れ線/エリア/円グラフ/オーバーレイチャートのパラメータと同じです。列を定義することはできません。

5.12.6.1 列マッピング

例えば、以下に示すデータがあるとします:

Subject	Student	Score
Math	A.S.	65
Math	T.E.	75
Math	V.Q.	83
Math	X.C.	87
Math	I.Z.	93
Science	A.S.	86
Science	T.E.	90
Science	V.Q.	73

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Subject	Student	Score
Science	X.C.	95
Science	I.Z.	84

オリジナルデータ

次のコードでは、カテゴリを Column 0（"Subject"）、値を Column 2（"Score"）に設定します。

```
ColInfo colInfo = new ColInfo();
colInfo.category = 0;
colInfo.value = 2;
```

5.12.6.2 チャートの作成

次のコードは、前述のボックスチャートを作成する方法を示しています。

```
Component doDataFromArguments(Applet applet) {


    //EspressManager に接続しないでください
    QbChart.setEspressManagerUsed(false);

    //列マッピング
    ColInfo colInfo = new ColInfo();
    colInfo.category = 0;
    colInfo.value = 2;

    String dataType[] = {"varchar", "varchar", "integer"};
    String fieldName[] = {"Subject", "Student", "Score"};
    String records[][] = {{"Math", "A.S.", "65"}, {"Math", "T.E.", "75"},
                          {"Math", "V.Q.", "83"}, {"Math", "X.C.", "87"},
                          {"Math", "I.Z.", "93"}, {"Science", "A.S.", "86"},
                          {"Science", "T.E.", "90"}, {"Science", "V.Q.", "73"},
                          {"Science", "X.C.", "95"}, {"Science", "I.Z.", "84"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                                (applet,              //アプレット
                                QbChart.VIEW2D,       //2 次元
                                QbChart.BOX,          //ボックスチャート
                                data,                 //データ
                                colInfo,              //列情報
```

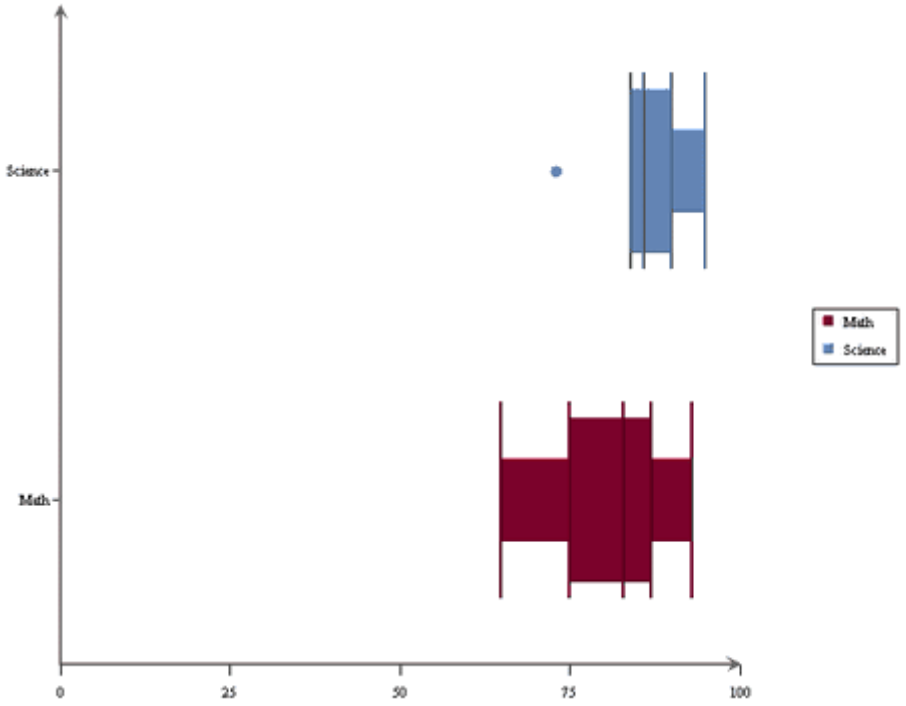
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                                    null);                    //テンプレート指定無し
        return chart;

}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、次のような 2 次元のボックスチャートが生成されます:



生成されたボックスチャート

作成されたチャートは、フォーマットされていないデフォルトチャートです。

5.12.7 バブルチャート

バブルチャートの **CollInfo** パラメータは、3 次元散布図のパラメータと同じです。バブルチャートを描画するには、**xvalue**、**yvalue**、**zvalue** の各パラメータが必要です。バブルチャートの場合、**zvalue** は Bubble Size を指します。

5.12.7.1 列マッピング

例えば、以下に示すデータがあるとします:

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Drink	High	Average	Low
Water	3	2	2
Soda	1	1	0
Coffee	5	2	1
Tea	3	1	1

オリジナルデータ

次のコードでは、列を Column 0（”Drink”）、xvalue を Column 1（"quote> High"）、yvalue を Column 3（"Low"）、zvalue を Column 2（"Average"）：

```
ColInfo colInfo = new ColInfo();
colInfo.xvalue = 1;
colInfo.yvalue = 3;
colInfo.zvalue = 2;
colInfo.series = 0;
```

5.12.7.2 チャートの作成

次のコードは、前述のバブルチャートを作成する方法を示しています:

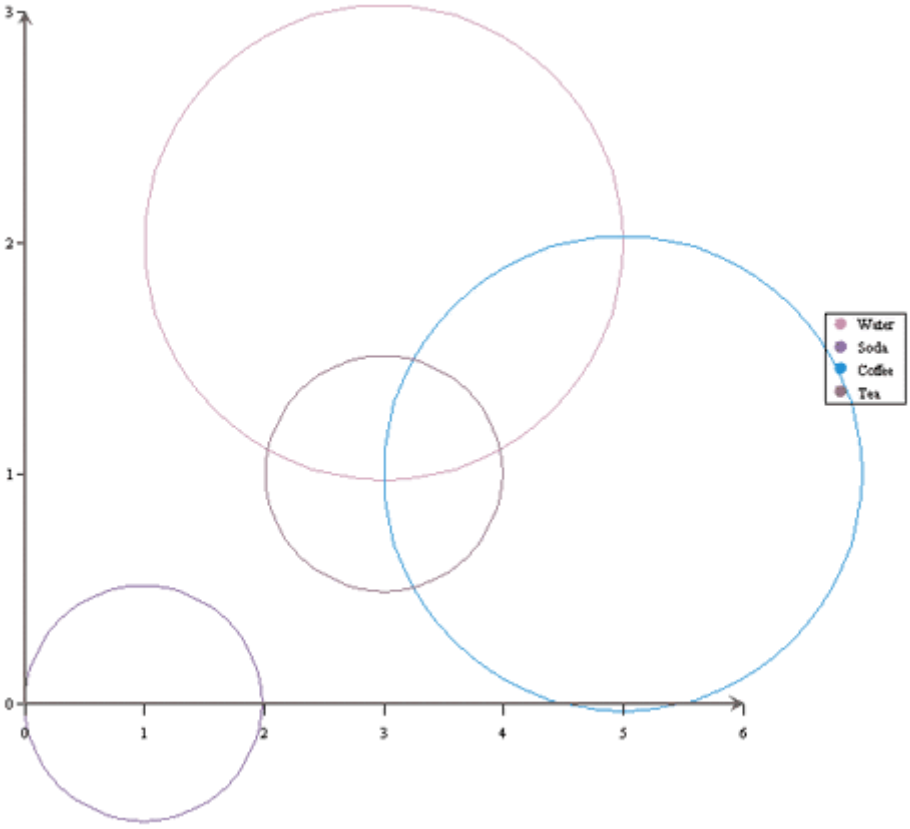
```
Component doDataFromArguments(Applet applet) {


    //EspressManager に接続しないでください
    QbChart.setEspressManagerUsed(false);

    //列マッピング
    ColInfo colInfo = new ColInfo();
    colInfo.xvalue = 1;
    colInfo.yvalue = 3;
    colInfo.zvalue = 2;
    colInfo.series = 0;

    String dataType[] = {"varchar", "integer", "integer", "integer"};
    String fieldName[] = {"Drink", "High", "Average", "Low"};
    String records[][] = {{"Water", "3", "2", "2"}, {"Soda", "1", "1", "0"},
                          {"Coffee", "5", "2", "1"}, {"Tea", "3", "1", "1"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                                (applet,                //アプレット
                                QbChart.VIEW2D,         //2 次元
                                QbChart.BUBBLE,         //バブルチャート
                                data,                   //データ
                                colInfo,                //列情報
                                null);                  //テンプレート指定無し
        return chart;

}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、以下に示す 2 次元のバブルチャートが生成されます:



生成されたバブルチャート

作成されたチャートは、フォーマットされていないデフォルトチャートです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.12.8 High-Low/HLCO チャート

High-Low/HLCO チャートの **CollInfo** パラメータは、他のチャートタイプのパラメータと異なります。ここで、チャートの値の部分は、さらに、**open** 値、**close** 値、**high** 値および **low** 値に細分される。高低チャートの最小要件は、カテゴリ、**high** 値および **low** 値パラメータ（HLCO の場合は、**close** 値および **open** 値のパラメータも必要です）です。また、series と subvalue のパラメータを追加することもできます。

5.12.8.1 列マッピング

例えば、以下に示すデータがあるとします:

Day	Company	High	Low	Close	Open	Volume
2001-01-01	ABC	1.33	1.18	1.22	1.23	43723
2001-01-01	DEF	9.24	8.74	9.16	8.89	18478
2001-01-01	GHI	2.20	1.82	2.14	1.97	46743
2001-01-02	ABC	1.87	0.79	1.63	1.12	33605
2001-01-02	DEF	9.48	8.12	8.93	8.66	16758
2001-01-02	GHI	2.47	2.32	2.44	2.34	60671
2001-01-03	ABC	2.47	0.22	0.44	0.63	45211
2001-01-03	DEF	9.94	9.92	9.93	9.93	10697
2001-01-03	GHI	2.48	2.40	2.46	2.44	45238
2001-01-04	ABC	1.8	1.38	1.79	1.44	50224
2001-01-04	DEF	9.49	8.87	8.94	8.93	11868
2001-01-04	GHI	2.06	1.45	1.96	1.46	62053
2001-01-05	ABC	1.23	0.58	0.79	0.79	37285
2001-01-05	DEF	9.94	8.61	8.08	8.94	10476
2001-01-05	GHI	2.8	1.58	2.45	1.96	47117

オリジナルデータ

次のコードでは、カテゴリを Column 0（"Day"）、列を Column 1（"Company"）、High を Column 2（"High"）、Low を Column 3（"Low"）、Close を Column 4 （"Close"）、Open を Column 5（"Open"）、subvalue を Column 6（Volume）に設定します：

```
CollInfo collInfo = new CollInfo();
collInfo.category = 0;
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
colInfo.series = 1;
colInfo.high = 2;
colInfo.low = 3;
colInfo.close = 4;
colInfo.open = 5;
colInfo.subvalue = 6;
```

5.12.8.2 チャートの作成

次のコードは、前述の HLCO チャートを作成する方法を示しています。

```
Component doDataFromArguments(Applet applet) {


    //EspressManager は接続しないでください
    QbChart.setEspressManagerUsed(false);

    //列マッピング
    ColInfo colInfo = new ColInfo();
    ColInfo colInfo = new ColInfo();
    colInfo.category = 0;
    colInfo.series = 1;
    colInfo.high = 2;
    colInfo.low = 3;
    colInfo.close = 4;
    colInfo.open = 5;
    colInfo.subvalue = 6;

    String dataType[] = {"date", "varchar", "double", "double", "double", "double", "integer"};
    String fieldName[] = {"Day", "Company", "High", "Low", "Close", "Open", "Volume"};
    String records[][] = {{"2001-01-01", "ABC", "1.33", "1.18", "1.22", "1.23", "43723"},
                          {"2001-01-01", "DEF", "9.24", "8.74", "9.16", "1.23", "18478"},
                          {"2001-01-01", "GHI", "2.20", "1.82", "2.14", "1.23", "46743"},
                          {"2001-01-02", "ABC", "1.87", "0.79", "1.63", "1.23", "33605"},
                          {"2001-01-02", "DEF", "9.48", "8.12", "8.93", "1.23", "16758"},
                          {"2001-01-02", "GHI", "2.47", "2.32", "2.44", "1.23", "60671"},
                          {"2001-01-03", "ABC", "1.12", "0.22", "0.44", "1.23", "45211"},
                          {"2001-01-03", "DEF", "9.94", "9.92", "9.93", "9.93", "10697"},
                          {"2001-01-03", "GHI", "2.48", "2.40", "2.46", "2.44", "45238"},
                          {"2001-01-04", "ABC", "1.80", "1.38", "1.79", "1.44", "50224"},
                          {"2001-01-04", "DEF", "9.49", "8.87", "8.94", "8.93", "11868"},
```

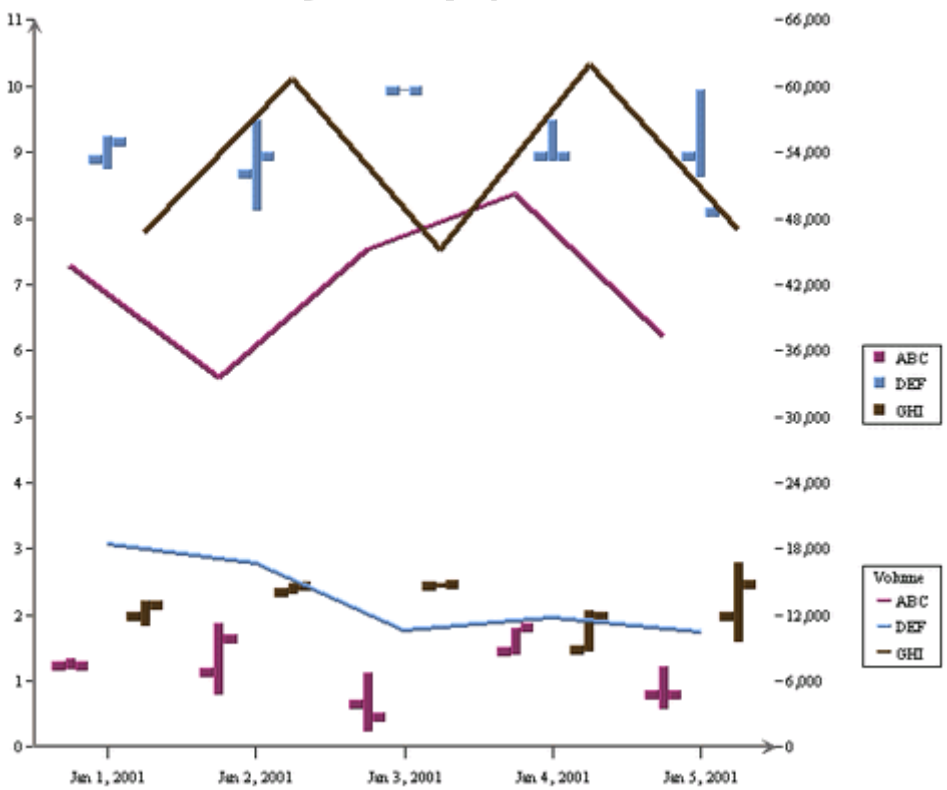
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                {"2001-01-04", "GHI", "2.06", "1.45", "1.96", "1.46", "62053"},
                {"2001-01-05", "ABC", "1.23", "0.58", "0.79", "0.79", "37285"},
                {"2001-01-05", "DEF", "9.94", "8.61", "8.08", "8.94", "10476"},
                {"2001-01-05", "GHI", "2.80", "1.58", "2.45", "1.96", "47117"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                        (applet,            //アプレット
                        QbChart.VIEW2D,     //2 次元
                        QbChart.HLCO,       //HLCO チャート
                        data,               //データ
                        colInfo,            //列情報
                        null);              //テンプレート指定無し
    return chart;

}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、次のような 2 次元 HLCO チャートが生成されます。



生成された HLCO チャート

作成されたチャートは、フォーマットされていないデフォルトチャートです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.12.9 サーフェスチャート

サーフェスチャートの ColInfo パラメータは、3 次元散布図のパラメータに似ています。サーフェスチャートを作成するには、**xvalue**、**yvalue**、**zvalue** パラメータが必要です。ただし、サーフェスチャートは **series** パラメータをサポートしていません。サーフェスチャートの場合、定義されている **ColInfo** オブジェクトは、データの内容にかかわらず常に同じです。

5.12.9.1 列マッピング

例えば、以下に示すデータがあるとします:

	0	10	20	30	40	50	60	70	80	90	100
0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	1	0	0	0	0	0
20	0	0	0	0	1	2	1	0	0	0	0
30	0	0	0	1	2	3	2	1	0	0	0
40	0	0	1	2	3	4	3	2	1	0	0
50	0	1	2	3	4	5	4	3	2	1	0
60	0	0	1	2	3	4	3	2	1	0	0
70	0	0	0	1	2	3	2	1	0	0	0
80	0	0	0	0	1	2	1	0	0	0	0
90	0	0	0	0	0	1	0	0	0	0	0
100	0	0	0	0	0	0	0	0	0	0	0

オリジナルデータ

次のコードは、サーフェスチャートのマッピングを設定します。このマッピングはサーフェスチャートでは一定であることに注意してください。

```
int map[] = {0, 2, 1};
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
ColInfo colInfo = new ColInfo(-1, map);
```

5.12.9.2 チャートの作成

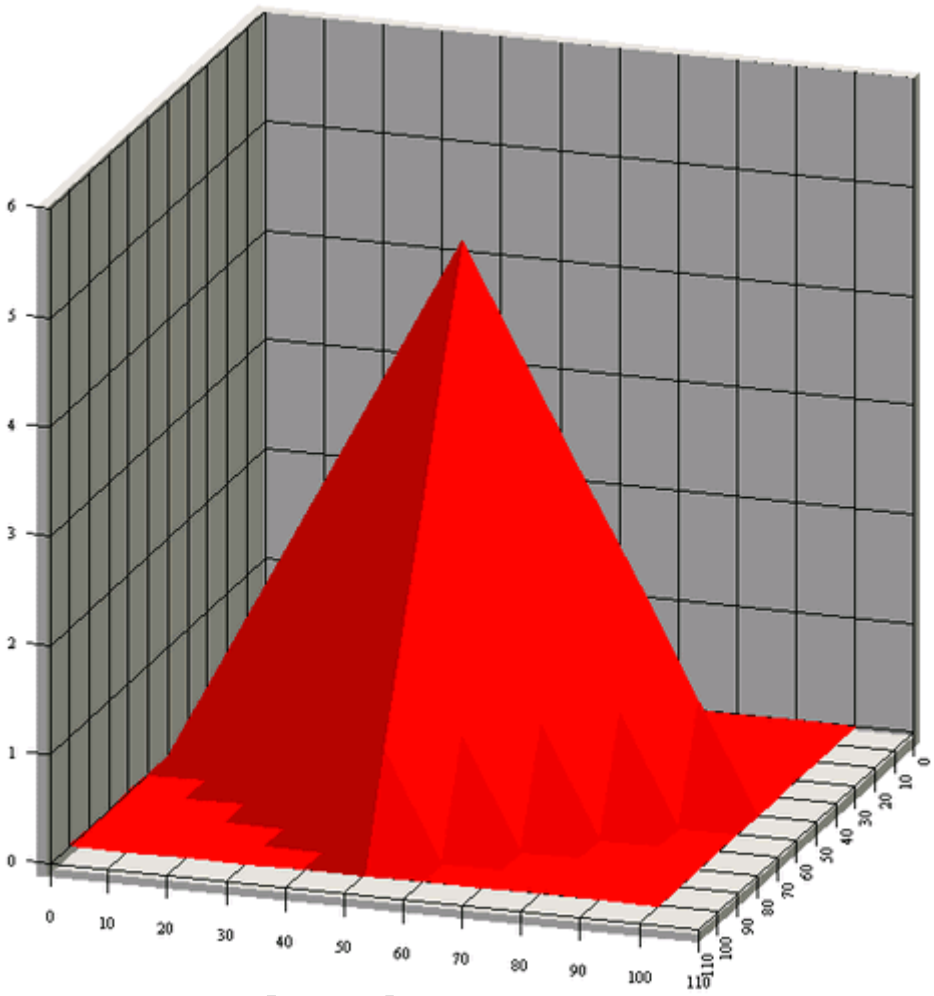
次のコードは、前述のサーフェスチャートを作成する方法を示しています:

```
Component doDataFromArguments(Applet applet) {

        QbChart.setEspressManagerUsed(false);

        int map[] = { 0, 2, 1 };
        ColInfo colInfo = new ColInfo(-1, map);

        String inputFileName = "surface.dat";

        QbChart chart = null;

        try {

                chart = new QbChart(applet, //Applet
                                QbChart.VIEW3D, //3 次元
                                QbChart.SURFACE, //サーフェスチャート
                                QbChart.DATAFILE, //テキストファイルの種類
                                inputFileName, //データ
                                false, //データを変換しないでください
                                colInfo, //列情報
                                null); //テンプレート指定無し

        } catch (Exception ex)
        {
                ex.printStackTrace();
        }

        return chart;

}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、次のような 3 次元サーフェスチャートが生成されます:

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



生成されたサーフェスチャート

作成されたチャートは、フォーマットされていないデフォルトチャートです。

5.12.10 ガントチャート

ガントチャートの **CollInfo** パラメータは、High-Low チャートのパラメータに似ています。ここでは、ガントチャートを作成するには、カテゴリ **High**（開始日）と **Low**（終了日）が必要です。必要に応じて、**series** パラメータを追加することもできます。

5.12.10.1 列マッピング

例えば、以下に示すデータがあるとします:

Project	Task	Starting Date	Ending Date
Project1	Task A	1998-01-15	1998-03-01
Project1	Task B	1998-02-25	1998-05-18
Project1	Task C	1998-06-11	1998-09-29
Project2	Task A	1998-03-15	1998-09-29

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

Project2	Task B	1998-04-25	1998-08-18
Project2	Task C	1998-08-11	1998-12-29

オリジナルデータ

次のコードでは、カテゴリを Column 1 ("Task")、High を Column 2 ("Starting Date")、Low を Column 3（"Ending Date")、列を Column 0（"Project"）に設定します:

```
ColInfo colInfo = new ColInfo();
colInfo.category = 1;
colInfo.high = 2;
colInfo.low = 3;
colInfo.series = 0;
```

5.12.10.2 チャートの作成

次のコードは、前述のガントチャートを作成する方法を示しています:

```
Component doDataFromArguments(Applet applet) {

    QbChart.setEspressManagerUsed(false);

    //列マッピング
    ColInfo colInfo = new ColInfo();
    colInfo.category = 1;
    colInfo.high = 2;
    colInfo.low = 3;
    colInfo.series = 0;

    String dataType[] = {"varchar", "varchar", "date", "date"};
    String fieldName[] = {"Project", "Task", "Starting Date","Ending Date"};
    String records[][] = {{"Project1", "Task A", "1998-01-15","1998-03-01"},
                          {"Project1", "Task B", "1998-02-25","1998-05-18"},
                          {"Project1", "Task C", "1998-06-11","1998-09-29"},
                          {"Project2", "Task A", "1998-03-15","1998-05-08"},
                          {"Project2", "Task B", "1998-04-25","1998-08-18"},
                          {"Project2", "Task C", "1998-08-11","1998-12-29"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                                (applet,                //アプレット
                                QbChart.VIEW2D,         //2 次元
                                QbChart.GANTT,          //ガントチャート
```

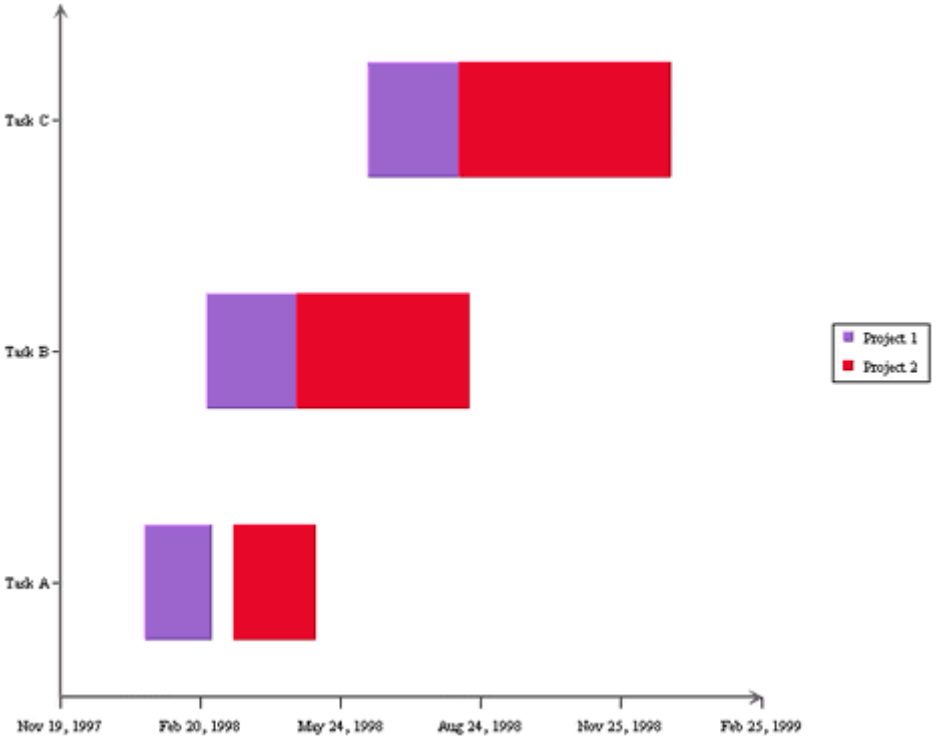
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                                data,                   //データ
                                colInfo,                //列情報
                                null);                  //テンプレート指定無し
        return chart;

}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、次のような 2 次元ガントチャートが生成されます。



生成されたガントチャート

作成されたチャートは、フォーマットされていないデフォルトチャートです。

5.12.11 極座標

極座標の **ColInfo** パラメータは、散布図のパラメータに似ています。極座標を作成するには、**radius** と **angle** パラメータが必要です。必要に応じて、**series** パラメータを追加することもできます。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.12.11.1 列マッピング

例えば、以下に示すデータがあるとします:

Radius	Angle	Series
0	2	A
90	4	A
180	6	A
270	8	A
360	10	A
45	3	B
135	5	B
225	7	B
315	9	B

オリジナルデータ

次のコードでは、半径を Column 0 ("Radius")、角度を Column 1 ("Angle")、列を Column 2 ("Series") に設定します:

```
ColInfo colInfo = new ColInfo(2, new int[]{0, 1});
```

5.12.11.2 チャートの作成

次のコードは、前述の極座標グラフを作成する方法を示しています:

```
Component doDataFromArguments(Applet applet) {

    QbChart.setEspressManagerUsed(false);

    //列マッピング
    ColInfo colInfo = new ColInfo(2, new int[]{0, 1});

    String dataType[] = {"integer", "integer", "varchar"};
    String fieldName[] = {"Radius", "Angle", "Series"};
    String records[][] = {{"0", "2", "A"}, {"90", "4", "A"}, {"180", "6", "A"},
                          {"270", "8", "A"}, {"360", "10", "A"}, {"45", "3", "B"},
                          {"135", "5", "B"}, {"225", "7", "B"}, {"315", "9", "B"}};
    DbData data = new DbData(dataType, fieldName, records);
    QbChart chart = new QbChart
                        (applet,                    //アプレット
                        QbChart.VIEW2D,             //2 次元
```

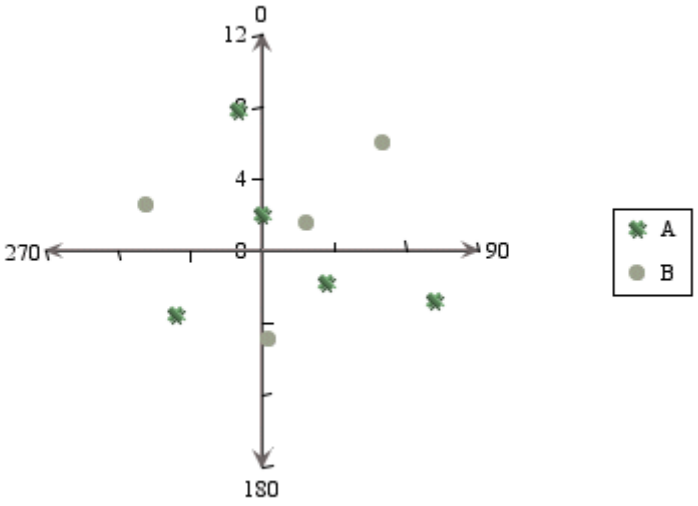
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
                        QbChart.POLAR,          //極座標
                        data,                   //データ
                        colInfo,                //列情報
                        null);                  //テンプレートの指定無し
    return chart;

}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、以下に示すような 2 次元の極座標が生成されます。



生成された極座標

作成されたチャートは、フォーマットされていないデフォルトチャートです。

5.12.12 日付・時間ベースのズームチャート

EspressReport では、チャートの日付/時刻ベースのズームが可能です。日付/時刻ベースのズームは、High Low、HLCO、散布図、サーフェス、ボックス、ダイヤル、レーダー、バブル、ガントチャートを除くほとんどすべてのタイプのチャートに適用できます。唯一の主要な要件は、カテゴリ軸に沿ったデータが日付、時刻、またはタイムスタンプのタイプであることです。

ズームを行うには、要素を設定してからチャートをリフレッシュします。属性は、**quadbase.util.IZoomInfo** インタフェースを使用して設定します。

次の例は、例またはアプリケーションとして実行でき、ズームを組み込んだチャートを示しています。ここでは、デフォルトのズームは一度に 5 日を表示し、最大のズームアウトは 1 ヶ月、最大のズームイ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ンは 1 日です。

```
//メモリ内の配列に渡されたデータ
DbData chartData = new DbData(dataTypes, fieldNames, data);

//列マッピングのセット
ColInfo colInfo = new ColInfo();
colInfo.category = 1;
colInfo.value = 0;

//チャートの作成
QbChart chart = new QbChart(

                    (Applet)null,               //アプレット
                    QbChart.VIEW2D,               //次元
                    QbChart.COL,                  //チャートタイプ
                    chartData,                   //データ
                    colInfo,                    //列マッピング
                    null);                      //テンプレート


//ズームインインタフェースのハンドルを取得する
IZoomInfo zoomInfo = chart.gethZoomInfo();
zoomInfo.setAggregateOperator(IZoomInfo.SUM);   //集計を指定する
zoomInfo.setMaxScale(1, IZoomInfo.YEAR);         //最大ズームアウトを指定する
zoomInfo.setMinScale(1, IZoomInfo.DAY);          //最大ズームインを指定する
zoomInfo.setScale(5, IZoomInfo.DAY);             //現在のズームを指定する
zoomInfo.setLinearScale(true);                   //リニアスケールをオンにする
zoomInfo.setZoomEnabled(true);                    //ズームをオンにする

try
{

    chart.refresh();                             //拡大表示でグラフを更新する

} catch (Exception ex)
{

    ex.printStackTrace();

}
```

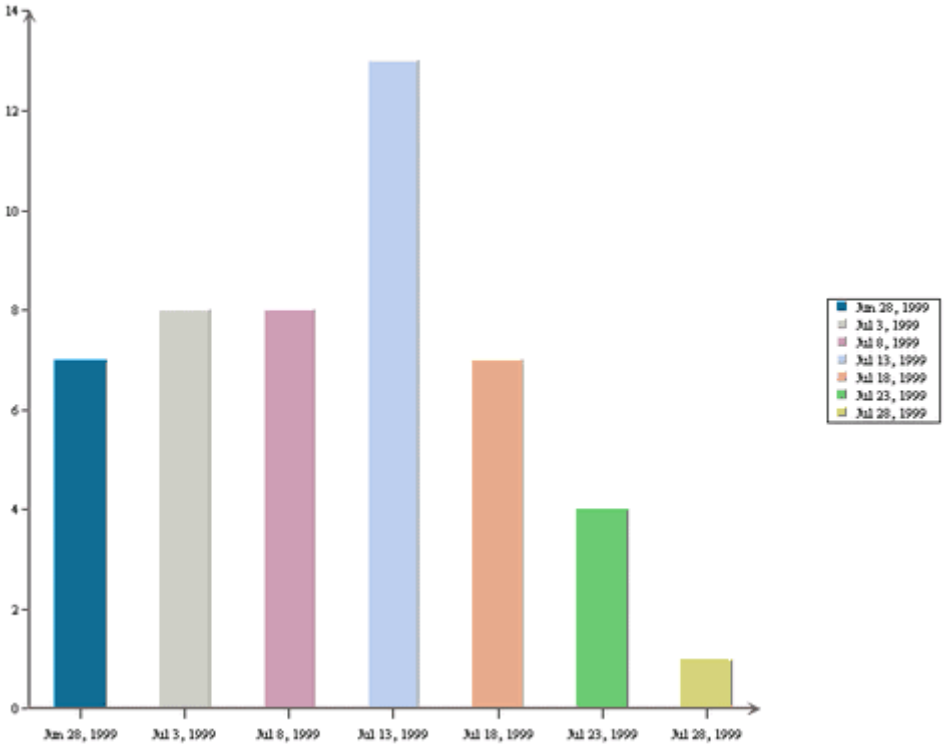
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
return chart;
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、次のようなトップレベルのチャートが生成されます。



生成されたグラフ

ポップアップメニューで選択した内容に応じて、このグラフを拡大または縮小することができます（グラフキャンバスを右クリックすると表示されます）。

これは、どのような種類の書式設定もせずに生成されたデフォルトチャートです。

5.12.13 パラメータ化されたグラフ

通常のクエリに加えて、1 つまたは複数のパラメータを持つクエリを渡すことができ、グラフを生成する前に、グラフにパラメータの値を入力するように求めることができます。スタンドアロンのチャートのみをパラメータ化できることに注意してください。

チャートのデータソースとしてパラメータ化されたクエリを使用するには、**IQueryFileInfo**（既に提供されている実装、**SimpleQueryFileInfo** を使用できます）を実装するクラスを作成し、そのクラスを使用してパラメータ情報を渡す必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

パラメータ化されたクエリを使用するチャートの例を次に示します。クエリが実行されるデータベースは WoodView HSQL なので、クラスパスにデータベース HSQL JDBC ドライバ（hsqldb.jar）を追加する必要があります。ドライバは<EspressReportInstall>/lib ディレクトリにあります。

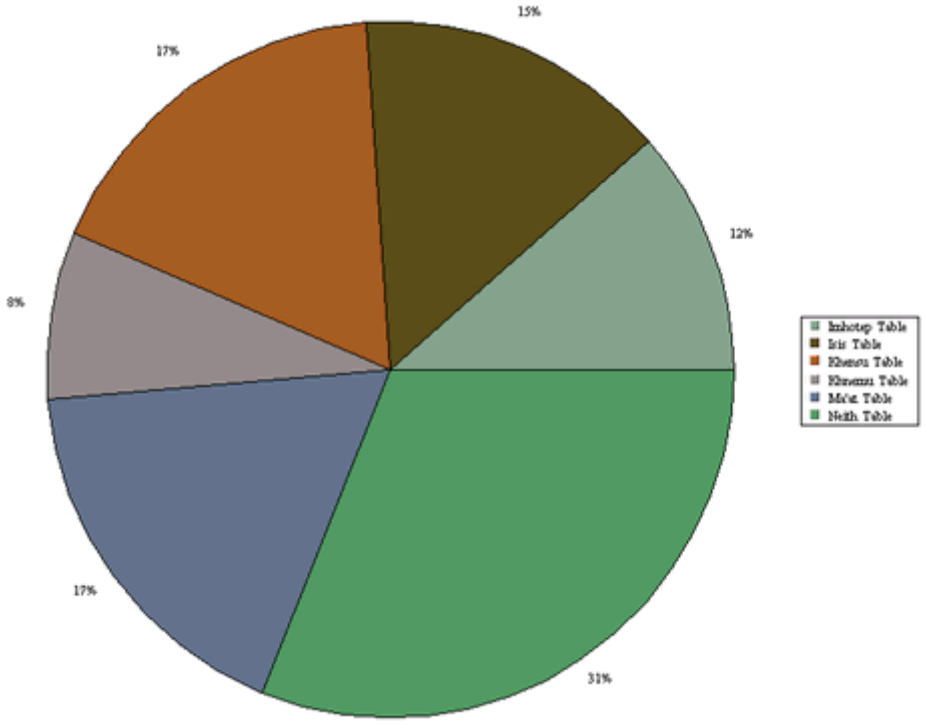
```
Component doParameterizedChart(Applet applet) {
	//EspressManager は使用しないでください
	QbChart.setEspressManagerUsed(false);

	//開始コード：パラメータ情報の追加
	//SimpleQueryInParam(name of Parameter, String to be displayed, MapToColumn?, tableName, ColumnName, SQL Type, DefaultValue, value)

	SimpleQueryInParam inParam = new SimpleQueryInParam("param", "Please select", true,
			"Categories", "CategoryName", Types.VARCHAR, "Arm Chairs", null);
	SimpleQueryInParam[] paramSet = { inParam };
	SimpleQueryFileInfo chartInfo = new SimpleQueryFileInfo(
			"jdbc:hsqldb:woodview",
			"org.hsqldb.jdbcDriver",
			"sa",
			"",
			"select Products.ProductName, Products.UnitsInStock from Categories, Products where Categories.CategoryID=Products.CategoryID and Categories.CategoryName=:param order by Categories.CategoryName, Products.ProductName;");

	chartInfo.setInParam(paramSet);

	//終了コード：パラメータ情報の追加//開始コード：列マッピングの設定
	ColInfo colInfo = new ColInfo();
	colInfo.category = 0;
	colInfo.value = 1;
	//終了コード：列マッピングの設定

	QbChart chart = new QbChart(applet, QbChart.VIEW2D, QbChart.PIE, chartInfo, colInfo, null);

	return chart;
}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、次のようなグラフが選択された

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

パラメータに応じて生成されます:



生成されたグラフ

これは、どのような種類の書式設定もせずに生成されたデフォルトチャートです。

パラメータ化されたグラフをレポートに追加する場合は、メインレポートの列をリンクして、グラフに定義されているパラメータの値を提供する必要があります。これを行うには、ReportChartObject で次のメソッドを使用します。

```
public void setParameterMap（String [] columnID）;
```

文字列配列には、チャートのパラメータの値を提供する列の ID が含まれています。

たとえば、ユーザが単一の値ではなく値の範囲に対してチェックしたい場合など、複数の値を持つようにパラメータを割り当てることもできます。 値の範囲は、通常、SQL クエリの"IN"句内で指定されます。 EspressReport は、"IN"句内のパラメータを複数値とみなすだけであることに注意してください。

グラフのデータソースとして多値のパラメータ化されたクエリを使用するには、IQueryFileInfo を実装するクラスを作成し、そのクラスを使用してパラメータ情報を渡す必要があります。
以下は、複数値のパラメータ化されたクエリを使用するチャートの例です。クエリが実行されるデータベースは WoodView HSQL なので、クラスパスにデータベース HSQL JDBC ドライバ（hsqldb.jar）を追加する必要があります。ドライバは<EspressReportInstall>/lib ディレクトリにあります。

```
Component doMultiValueParameter(Applet applet) {
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

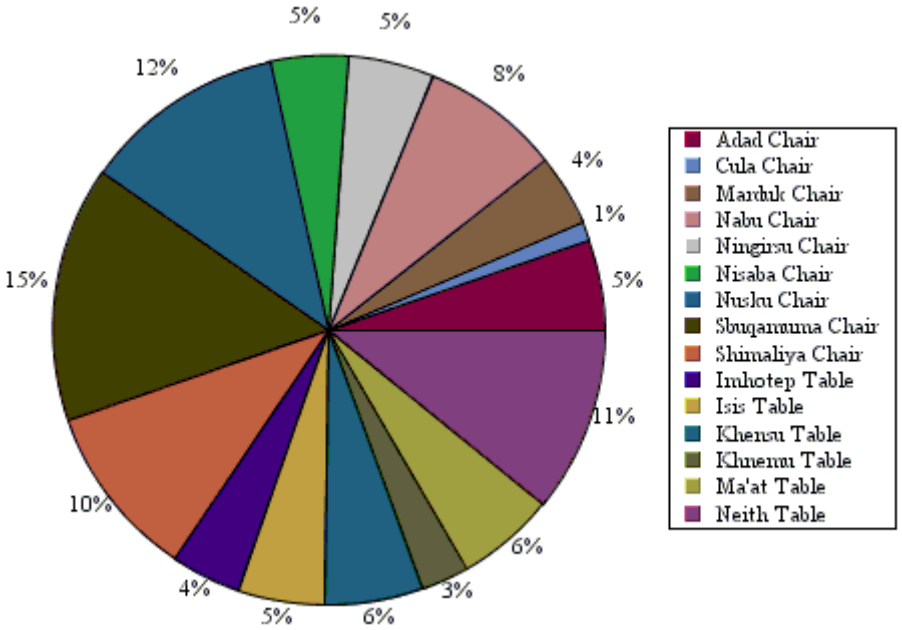
```
        //EspressManagerh は使用しないでください
        QbChart.setEspressManagerUsed(false);

        //開始コード：パラメータ情報の追加
        SimpleQueryMultiValueInParam inParam =
                        new SimpleQueryMultiValueInParam("param", "Please select", true,
                                        "Categories", "CategoryName", Types.VARCHAR, "Arm
Chairs", null);
        SimpleQueryMultiValueInParam[] paramSet = { inParam };
        SimpleQueryFileInfo chartInfo = new SimpleQueryFileInfo(
                        "jdbc:hsqldb:woodview",
                        "org.hsqldb.jdbcDriver",
                        "sa",
                        "",
                        "select Products.ProductName, Products.UnitsInStock from Categories,
Products where Categories.CategoryID=Products.CategoryID and Categories.CategoryName
in(:param) order by Categories.CategoryName, Products.ProductName;");
        chartInfo.setInParam(paramSet);

        //終了コード：パラメータ情報の追加

        //開始コード：列マッピングの設定

        ColInfo colInfo = new ColInfo();
        colInfo.category = 0;
        colInfo.value = 1;

        //終了コード：列マッピングの設定

        QbChart chart = new QbChart(applet, QbChart.VIEW2D, QbChart.PIE, chartInfo, colInfo,
null);

        return chart;
}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、以下に示すようなグラフ（選択されたパラメータに応じて）が生成されます:

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



生成されたグラフ

これは、どのような種類の書式設定もせずに生成されたデフォルトチャートです。

IN 句に複数のパラメータがある場合、それぞれが単一値とみなされることに注意してください。たとえば、クエリでは次のようになります。

```
Select * From Products
Where ProductID IN (:param1, :param2, :param3)
```

:param1, :param2, :param3 は単一値のパラメータです。

複数値のパラメータを初期化することは、単一値のパラメータを初期化することと同じです。唯一の違いは、値選択ダイアログです。単一値のパラメータはドロップダウンボックスに変換されますが、複数値のパラメータには複数選択のリストボックスが与えられます。

5.12.14 ドリルダウンチャート

EspressReport は異なるタイプのドリルダウン機能をサポートします。タイプは以下の通りです:

- パラメータドリルダウン
- データドリルダウン
- 動的ドリルダウン

パラメータ化されたチャートと同様に、ドリルダウンチャートはスタンドアロンチャートを生成する場

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

合にのみ使用できます。

さまざまなタイプのドリルダウンチャートの作成については、以下のセクションで説明します。

5.12.14.1 パラメータドリルダウンチャート

パラメータ化されたクエリによって、パラメータドリルダウンチャートを作成することができます。大量のデータを含むグラフを表示するのではなく、必要な最小限のデータを示す最上位のグラフを表示して、選択したデータをさらに詳しく調べることができます。パラメータドリルダウンチャートの詳細については、パラメータドリルダウンを参照してください

パラメータドリルダウンチャートを作成するには、さまざまなチャートオブジェクトを作成する必要があります（ルートチャートオブジェクト以外のすべてのチャートオブジェクトは、データソースとしてパラメータ化されたクエリを持つことに注意してください）。チャートの列/横棒/点/線/円を使用して、パラメータドリルダウンチャートの次のレベルをアタッチします。

以下に、パラメータドリルダウンチャートの例を示します。クエリが実行されるデータベースは WoodView HSQL なので、クラスパスにデータベース HSQL JDBC ドライバ（hsqldb.jar）を追加する必要があります。ドライバは<EspressReportInstall>/lib ディレクトリにあります。

```
Component doDrillDownChart(Applet applet)  {

    //EspressManager は使用しないでください
    QbChart.setEspressManagerUsed(false);

        //開始コード：チャート 1 の作成 - ルートチャート
        DBInfo rootInfo = new DBInfo("jdbc:hsqldb:woodview","org.hsqldb.jdbcDriver",
                                    "sa",
                                    "",
                                    "SELECT Employees.LastName, Count(Orders.OrderID) AS
Count_Of_OrderID FROM Employees, Order_Details, Orders WHERE (Orders.EmployeeID =
Employees.EmployeeID) AND (Orders.OrderID = Order_Details.OrderID) GROUP BY
Employees.LastName");

        ColInfo rootColInfo = new ColInfo(-1, 0, -1, 1);
        QbChart rootChart = new QbChart(applet, QbChart.VIEW2D, QbChart.PIE, rootInfo, false,
rootColInfo, null);

        SimpleQueryInParam inParam = new SimpleQueryInParam("LastName", "Please select",
true,
                                    "Customers", "Company", Types.VARCHAR,
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

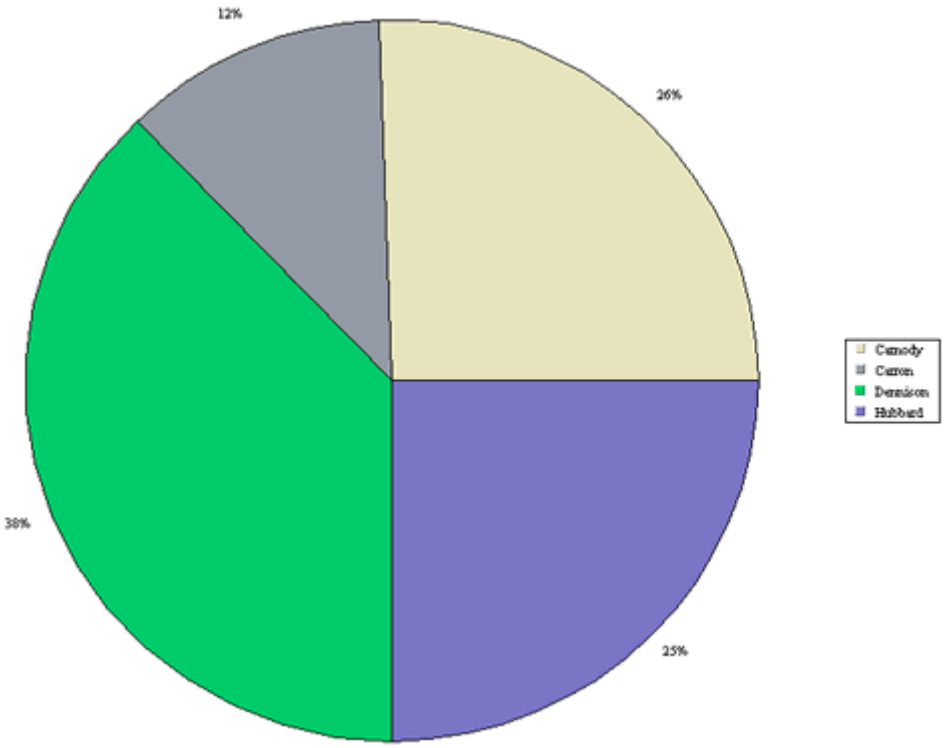
```
                                                    "All Unfinished Furniture", null);

        SimpleQueryInParam[] paramSet = {inParam};

        SimpleQueryFileInfo levelChartInfo = new SimpleQueryFileInfo("jdbc:hsqldb:woodview",
                                                    "org.hsqldb.jdbcDriver", "sa", "",
                                                    "SELECT Categories.CategoryName,
Employees.LastName, Sum(Order_Details.Quantity) AS Sum_Of_Quantity FROM Products,
Employees, Categories, Order_Details, Orders WHERE (Orders.OrderID = Order_Details.OrderID)
AND (Employees.EmployeeID = Orders.EmployeeID) AND (Products.CategoryID =
Categories.CategoryID) AND (Products.ProductID = Order_Details.ProductID) GROUP BY
Categories.CategoryName, Employees.LastName HAVING (Employees.LastName =:LastName)");



        levelChartInfo.setInParam(paramSet);

        ColInfo levelOneChartColInfo = new ColInfo(-1, 0, -1, 2);

        try {

                rootChart.createDrillDownChart("TestDrillDownChart", QbChart.VIEW2D,
                QbChart.COL, levelChartInfo, false, null,
                levelOneChartColInfo, null, new int[] {0});
                rootChart.updateDrillDownCharts();

        } catch (Exception ex) { ex.printStackTrace(); }

return rootChart;

}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、次のような最上位のグラフが生成されます:

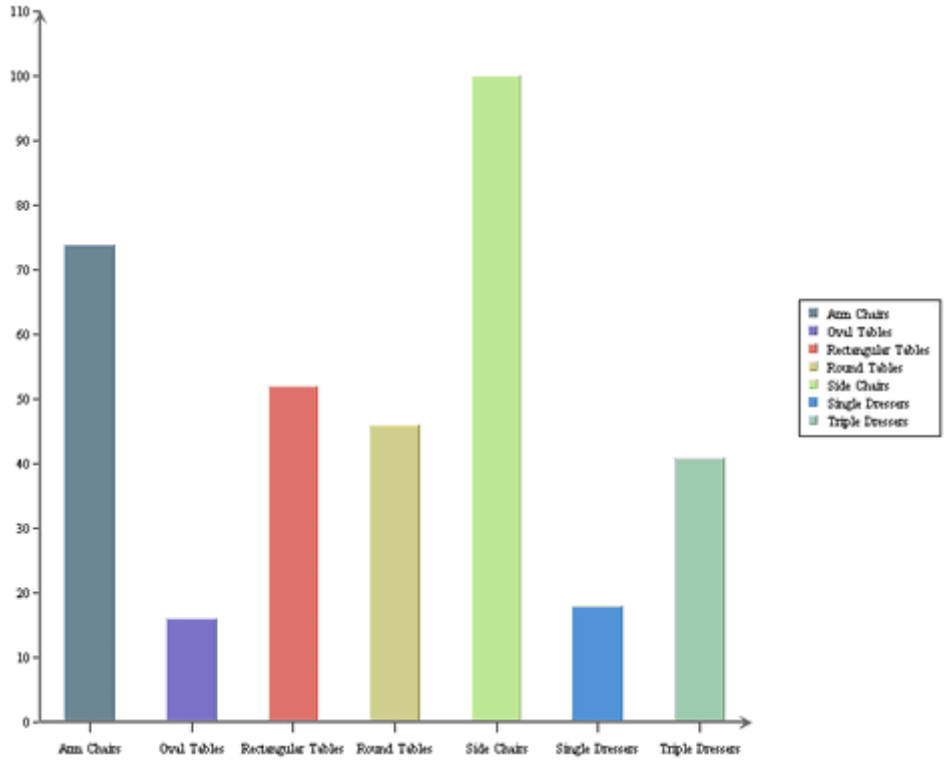
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



生成された最上位のグラフ

クリックされたリンクに応じて、以下に示すようなグラフが表示されます:



生成されたグラフ

これは、どのような種類の書式設定もせずに生成されたデフォルトチャートです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

EspressManager を使用せずにパラメータドリルダウンチャートを生成する場合は、.class ファイルの作業ディレクトリの下に drillTemplates というサブディレクトリが必要です。

5.12.14.2 データドリルダウンチャート

パラメータドリルダウンチャートを使用すると、さまざまなデータソースからチャートを生成できます。データドリルダウンでは、チャートのさまざまなレベルで同じデータソースが使用されます。最上位レベルには、データの要約が表示されます。データポイントをクリックすると、その特定のデータポイントに関する詳細な情報を示す別のチャートが表示されます。

縦棒グラフ、横棒グラフ、折れ線グラフ、円グラフ、エリアチャート、オーバーレイチャート、レーダーチャート、ダイヤルチャート、スタック縦棒グラフ、スタック横棒グラフのみがデータドリルダウン機能をサポートします。

データドリルダウンチャートを作成するには、最初にチャートオブジェクトを作成してから、ドリルダウンプロパティを指定する必要があります。

以下に、データドリルダウンチャートの例を示します。クエリが実行されるデータベースは WoodView HSQL なので、クラスパスにデータベース HSQL JDBC ドライバ（hsqldb.jar）を追加する必要があります。ドライバは<EspressReportInstall>/lib ディレクトリにあります。

```
Component doDataDrillDownChart(Applet applet) {

        //EspressManager は使用しないでください
        QbChart.setEspressManagerUsed(false);

        //開始コード：チャート 1 の作成 - ルートチャート
        DBInfo rootInfo = new DBInfo(
                        "jdbc:hsqldb:woodview",
                        "org.hsqldb.jdbcDriver",
                        "sa",
                        "",
                        "select Categories.CategoryName, Products.ProductName, Products.UnitsInStock from Categories, Products where Categories.CategoryID = Products.CategoryID order by Categories.CategoryName;");

        ColInfo rootColInfo = new ColInfo();
        rootColInfo.category = 0;
        rootColInfo.value = 2;
        QbChart chart = new QbChart(applet, QbChart.VIEW2D, QbChart.BAR, rootInfo, rootColInfo,
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

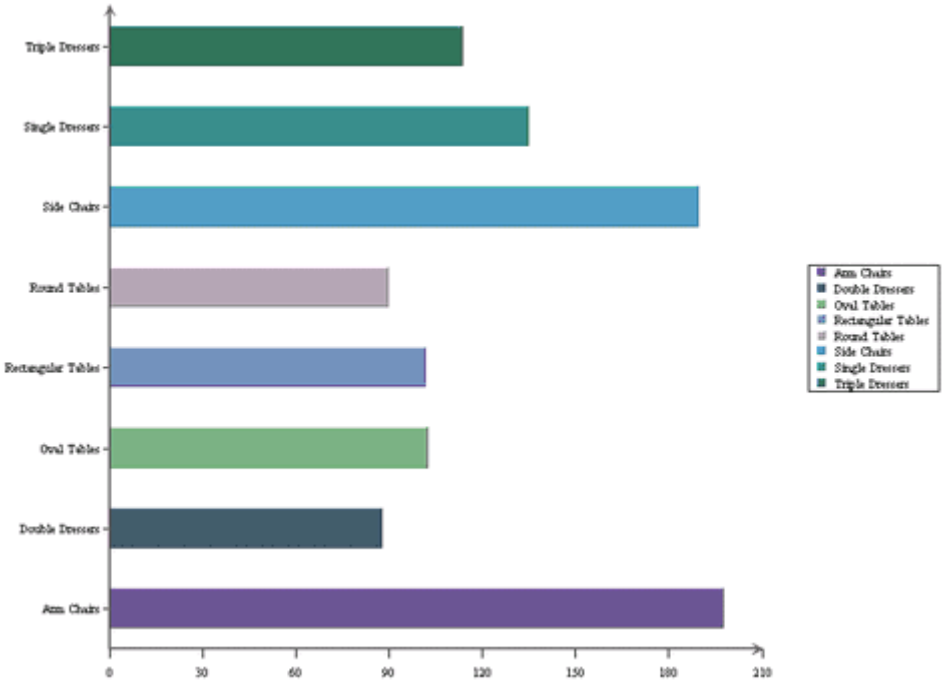
```
                        null);

        //終了コード：チャート 1 の作成 - ルートチャート

        try {

                //開始コード：チャート 2 の作成 – サブチャート

                IDrillDown chartDrillDown = chart.gethDrillDown();
                chartDrillDown.setAggregateOperator(IDrillDown.SUM);  //Set  the  Aggregation

                chartDrillDown.addDrillDown(QbChart.AREA,      1,      -1,      -1,      true);
//addDrillDown(chartType, category, series, sumby, is2DChart)
                chartDrillDown.setDrillName("DrillDownChart"); //Set the drill template name
        } catch (Exception ex)
        {
                ex.printStackTrace();
        }
        //終了コード：チャート 2 の作成 – サブチャート

        return chart;

}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、次のような最上位のグラフが生成されます。

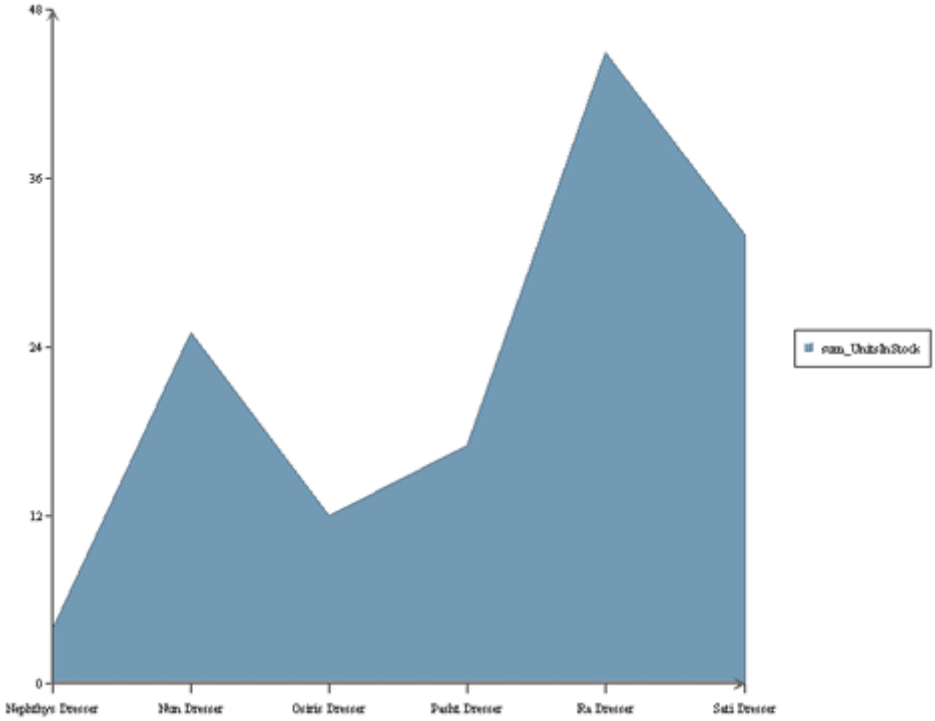
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



生成された最上位のグラフ

クリックされたリンクに応じて、以下に示すようなグラフが表示されます。



生成されたグラフ

これは、どのような種類の書式設定もせずに生成されたデフォルトチャートです。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.12.14.3 動的データドリルダウンチャート

データドリルダウンチャートでは、各ドリルダウンレベルの列間マッピングを事前に定義する必要があります。動的データドリルダウンを使用すると、チャートを表示しているときにドリルダウンチャートの列間マッピングを柔軟に選択できます。最上位のサマリー・グラフのみを構築し、ドリルダウン用に指定された集約のみを作成する必要があります。

縦棒グラフ、横棒グラフ、折れ線グラフ、円グラフ、エリアチャート、オーバーレイチャート、レーダーチャート、ダイヤルチャート、スタック縦棒グラフとスタック横棒グラフのみが動的データドリルダウン機能をサポートします。

動的データドリルダウンチャートを作成するには、まずチャートオブジェクトを作成してからドリルダウンプロパティを指定する必要があります。

以下に、データドリルダウンチャートの例を示します。クエリが実行されるデータベースは WoodView HSQL なので、クラスパスにデータベース HSQL JDBC ドライバ（hsqldb.jar）を追加する必要があります。ドライバは<**EspressReportInstall**>/**lib** ディレクトリにあります。

```
Component doDynamicDataDrillDownChart(Applet applet) {

        //EspressManager は使用しないでください
        QbChart.setEspressManagerUsed(false);

        //開始コード：チャート 1 の作成 - ルートチャート

        DBInfo rootInfo = new DBInfo(
                        "jdbc:hsqldb:woodview",
                        "org.hsqldb.jdbcDriver",
                        "sa",
                        "",
                        "select Categories.CategoryName, Products.ProductName, Products.UnitsInStock from Categories, Products where Categories.CategoryID = Products.CategoryID order by Categories.CategoryName;");

        ColInfo rootColInfo = new ColInfo();
        rootColInfo.category = 0;
        rootColInfo.value = 2;
        QbChart chart = new QbChart(applet, QbChart.VIEW2D, QbChart.BAR, rootInfo, rootColInfo,
                        null);

        //終了コード：チャート 1 の作成 - ルートチャート
```

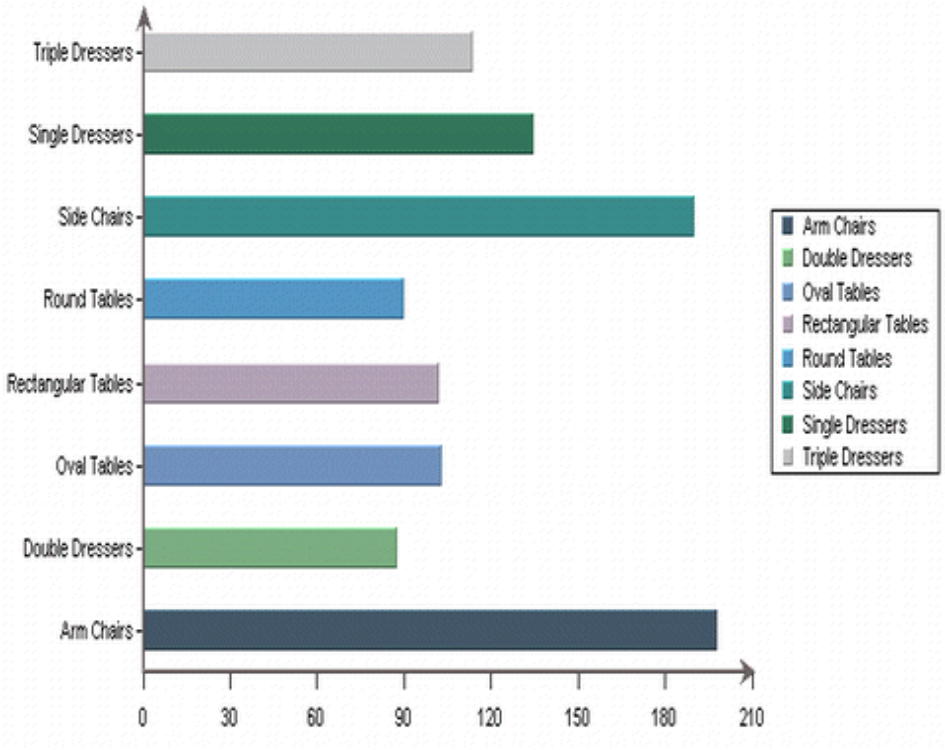
Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
        try {

                //開始コード：チャート 2 の作成 – サブチャート

                IDrillDown chartDrillDown = chart.gethDrillDown();
                chartDrillDown.setAggregateOperator(IDrillDown.SUM); //Set the Aggregation
                chartDrillDown.setDynamicDrillDown(true); //Turn on dynamic data drill-down
                chartDrillDown.setDrillName("DynamicDrillDownChart"); //Set the drill template
name

        } catch (Exception ex) {
                ex.printStackTrace();
        }

        //終了コード：チャート 2 の作成 – サブチャート

        return chart;
}
```

フルソースコード

上記のコードをアプレットまたはアプリケーションとして実行すると、次のような最上位のグラフが生成されます。

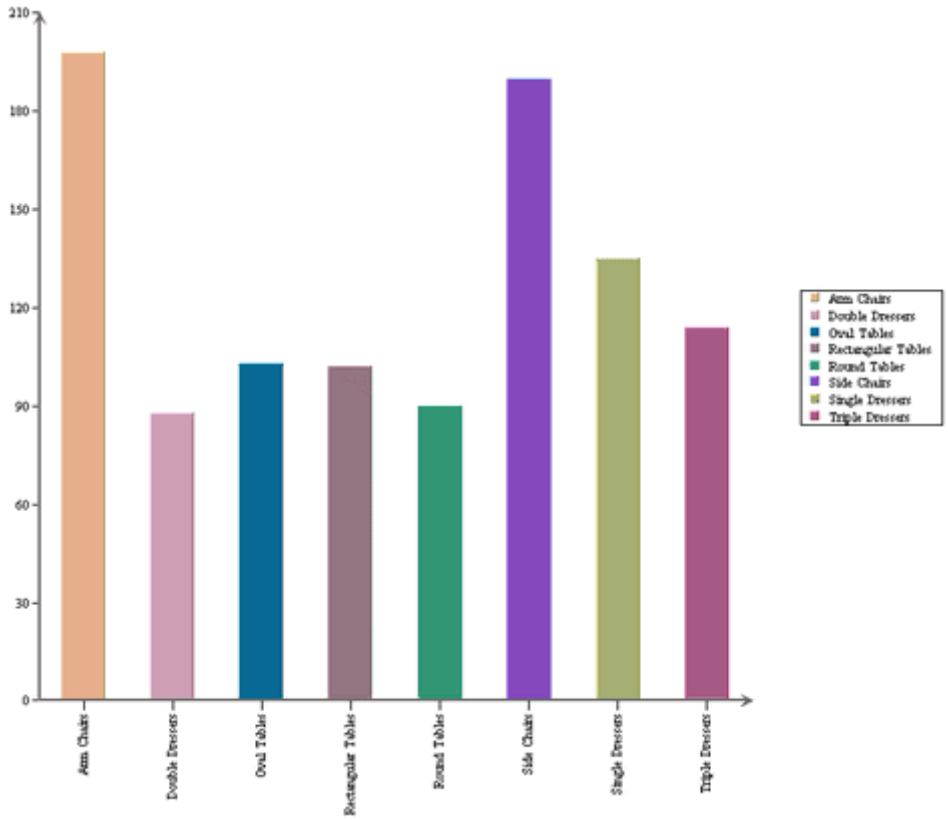


Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

生成されたグラフ

クリックされたリンクに応じて、以下に示すようなグラフが表示されます。



生成されたグラフ

これは、どのような種類の書式設定もせずに生成されたデフォルトチャートです。

Copyright(C) Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.12.15 レポートへのチャートの追加

グラフは、次の 3 つの方法のいずれかを使用してプログラムでプログラムに追加できます。

- Designer でチャートテンプレートを作成し、そのテンプレートをレポートコンポーネントに追加する。
- API を使用してグラフを作成し（そのデータソースとしてレポートデータを使用する）、そのグラフをレポートコンポーネントに追加する。
- API を使用して **QbChart** オブジェクトを作成し、そのオブジェクトをレポートコンポーネントに追加する。

5.12.15.1 チャートテンプレートの追加

チャートを追加するには、**ReportChartObject** オブジェクトを定義する必要があります。**ReportChartObject** オブジェクト内では、使用しているテンプレート（TPL）ファイルの場所を渡す必要があります。ディメンションと配置を指定することもできます。レポートを静的な形式（DHTML など）にエクスポートする場合は、エクスポートしたグラフのイメージタイプ（JPEG、GIF、PNG など）も指定できます。デフォルトの配置はセルの中心で、デフォルトの画像タイプは JPEG です。その後、レポート内の目的の場所にオブジェクトを追加できます。

TPL テンプレートファイルのみを使用することができ、チャートはレポートデータを使用します（テンプレートで独立したデータソースが使用されていても）

アプレットまたはアプリケーションとして実行できる次の例は、既存のチャートテンプレートをレポートに追加する方法を示しています。

```
ReportChartObject chartObject = new ReportChartObject();
String chartLocation0 = new String("AddingChartTemplate.tpl");
chartObject.setText(chartLocation0);
chartObject.setWidth(7.40);
chartObject.setHeight(4.90);
chartObject.setY(0.10);
chartObject.setImageType(QbReport.PNG);
report.getTable().getFooter().addData(chartObject);
```

フルソースコード
エクスポート結果

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.12.15.2 レポートデータを使用するグラフの追加

このメソッドを使用してグラフを追加するには、**ChartObject** クラスを使用してグラフオブジェクトを作成する必要があります。**ChartObject** オブジェクトには、チャートのプロパティ（チャートのマッピング、ディメンション、チャートのタイプなど）が含まれます。次に、ReportChartObject オブジェクトがインスタンス化/作成され、新しく作成された **ChartObject** オブジェクトが **setChart(IChart)**メソッドを使用して渡されます。最後に **ReportChartObject** がレポートの目的の場所に追加されます。

アプレットまたはアプリケーションとして実行できる次の例は、レポートデータをデータソースとして使用するグラフをレポートに追加する方法を示しています。

```
ColInfo chartColInfo = new ColInfo();
chartColInfo.category = 0;
chartColInfo.value = 1;

ChartObject chartObj = new ChartObject(report, //QbReport

        ChartObject.VIEW2D, //Dimension
        ChartObject.BAR, //Chart Type
        false, //Do Transpose
        null, //Transpose Columns
        chartColInfo, //Chart Mapping
        "AddingChart.tpl"); //Template

ReportChartObject chartObject = new ReportChartObject();
chartObject.setChart(chartObj);
chartObject.setWidth(7.40);
chartObject.setHeight(4.90);
chartObject.setY(0.10);
chartObject.setImageType(QbReport.PNG);
report.getTable().getFooter().addData(chartObject);
```

フルソースコード
エクスポート結果

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

5.12.15.3 独立したデータソースを使用するグラフの追加

このメソッドを使用してグラフを追加するには、QbChart クラスを使用してグラフオブジェクトを作成する必要があります。QbChart オブジェクトは、チャートプロパティ（すなわち、チャートマッピング、寸法およびチャートタイプ）を含む。その後、ReportChartObject オブジェクトがインスタンス化/作成され、新しく作成された QbChart オブジェクトが setChart（IChart）メソッドを使用して渡されます。最後に **ReportChartObject** がレポートの目的の場所に追加されます。

このシナリオは、グラフのデータがレポートから取得されていない点が前のセクションのシナリオと異なります。チャートのデータは、レポートのデータとは独立しています。

アプレットまたはアプリケーションとして実行できる次の例は、レポートデータをデータソースとして使用するグラフをレポートに追加する方法を示しています。

```
ColInfo chartColInfo = new ColInfo();
chartColInfo.category = 0;
chartColInfo.value = 1;

//Data Source for chart
String chartQuery = "SELECT ProductName, Sum(Order_Details.Quantity) "+
                    "FROM Products, Order_Details "+
                    "WHERE (Order_Details.ProductID = Products.ProductID) "+
                    "GROUP BY Products.ProductName "+
                    "ORDER BY Sum(Order_Details.Quantity) DESC";

DBInfo chartDatabaseInfo = new DBInfo("jdbc:hsqldb:database/woodview", "org.hsqldb.jdbcDriver",
"sa", "", chartQuery);

QbChart chartObj = new QbChart((Applet)null, QbChart.VIEW2D, QbChart.COL, chartDatabaseInfo,
false, chartColInfo, "AddingChartTemplate.tpl");

//Add ChartObject to report
ReportChartObject chartObject = new ReportChartObject();
chartObject.setChart(chartObj);
chartObject.setWidth(7.40);
chartObject.setHeight(4.90);
chartObject.setReportDataUsed(false);
chartObject.setY(0.10);
chartObject.setImageType(QbReport.PNG);
report.getTable().getFooter().addData(chartObject);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

フルソースコード
エクスポート結果

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

6 ローカライズ

6.1 EspressReport のローカライズ

EspressReport にはさまざまな機能が用意されており、あらゆるロケールや言語のチャートを生成できます。異なるインターナショナライゼーション機能では、特定の要件に応じて異なるシステム設定や、設定が必要になることがあります。

6.1.1 ローカライズの指定

EsspressReport では、チャートのタイムゾーンとロケールが異なります。それらは、作成されたマシン上のロケールに限定されません。ロケールは、チャートが実行される直前に API を通じて設定できます。

6.1.1.1 ロケール固有の書式設定

データ書式（日付と数値）の場合、EspressReport ではレポート要素のロケール固有の書式を設定できます。ロケール固有の書式設定では、レポートに使用されている特定のロケールのデータ要素を正しい形式で表示できます。ロケール固有のオプションは、Report Designer の **Data Formatting** ダイアログで使用できます（数式と列フィールドのデータ書式設定を参照）。

レポートとチャートのロケール、およびタイムゾーンは、API を使用して実行時に設定できます。これは、日付、時刻、およびデータ書式にのみ影響することに注意してください。

6.1.1.2 API を使用したタイムゾーンとロケールの設定

タイムゾーンまたはロケールを設定するには、QbReport/QbChart のメソッドを使用する（チャート全体に適用）か、**LocaleDataTimeFormat** および **LocaleNumericFormat** オブジェクトを使用します（グラフの特定のオブジェクトに適用）。以下のコードは、2 つの異なる方法で上記のことを行う方法を示しています。

```
reort.setLocale(Locale.UK);
report.settime zone(time zone.gettime zone(“GMT”));
```

フルソースコード
エクスポートされた結果

ここでは、書式がレポート全体に適用されます。また、次のように特定のデータ列にフォーマットを適用することもできます。

```
LocaleNumericFormat currencyFormat = LocaleNumericFormat.getCurrencyInstance();
currencyFormat.setLocale(Locale.UK);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

```
int columnIndex = 2; //column 2 is the "Unit Price" column
report.getTable().getColumn(columnIndex).setDataFormat(currencyFormat);

LocaleDateTimeFormat dateFormat = LocaleDateTimeFormat.getDateInstance();
dateFormat.settime zone(time zone.gettime zone("GMT"));
report.getPageFooter().getData(1).setDataFormat(dateFormat);
```

フルソースコード
エクスポートされた結果

6.1.2 言語とエンコーディング

EspressReport には、言語の違いによる制限を取り除く一連の機能が含まれています。使いやすいインタフェースを利用することで、EspressReport 内のすべてのテキスト、ボタン、メニューを別の言語に翻訳することができます。いくつかの基本的な設定によって、外国語の文字をインポート、表示、および入力することができます。また、その言語でレポートを保存およびエクスポートすることもできます。

6.1.2.1 EspressReport 言語翻訳

ローカライズは**.xml** ファイル、**Language.xml**（<EspressReport InstallDir>にあります）の使用によってサポートされています。**Language.xml** ファイルを操作し、英語の行を別の言語の行に置き換えるための GUI が提供されています。

Language.xml ファイルの構造は次のとおりです。

```
<Product name="REPORTDESIGNER" dir="quadbase/reportdesigner/designer">

        <File name="ReportMenubar.java">
                <CODE>File</CODE>
                <TEXT>File</TEXT>
                <CODE>New</CODE>
                <TEXT>New</TEXT>
                ... </File>

</Product>
```

このファイルには、Product（ディレクトリ） ＞ ファイル ＞ テキストのツリー構造があります。翻訳したい製品、ファイル、またはテキストを簡単に検索し、単純に**<Text>**と**</Text>**間のテキストを翻訳に置き換えることができます。EspressReport は、Java Swing UI および EspressReport Web アプリケーションの**<CODE>**を**<TEXT>**に置き換えます。翻訳 GUI にはすべての製品がリストされ、各製品には翻訳が必要なコードがリストされます。各製品の完全性を維持するために、複数の製品の下に同じ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

コード（たとえば、File）が表示される場合があります。ただし、コードの翻訳はすべての製品に複製されます。同じコードは、異なる製品に対して 2 つの異なる翻訳を持つことはできません。

Language.xml ファイルは、インストールディレクトリにあります。**Language.xml** のコピーを作成し、提供された GUI を使用して XML ファイルに直接触れることなく翻訳を行うことをお勧めします。UI を利用するには、EspressReport のルートディレクトリに移動し、次のコマンドを入力します。

```
java -classpath "./lib/ReportAPIWithChart.jar;.;" quadbase.internationalization.TranslateWizard
-file:<fileName> -enc:<encoding>
```

ここで、**<fileName>**は変換を格納する **xml** ファイルを表し、**<encoding>**は変換のエンコードを表します。正しい結果を得るには、適切なエンコーディングを指定する必要があります。ファイル名とエンコーディングが指定されていない場合、デフォルトのファイル名（**Language.xml**）とデフォルトのエンコーディング（**Cp1252**）が使用されます。

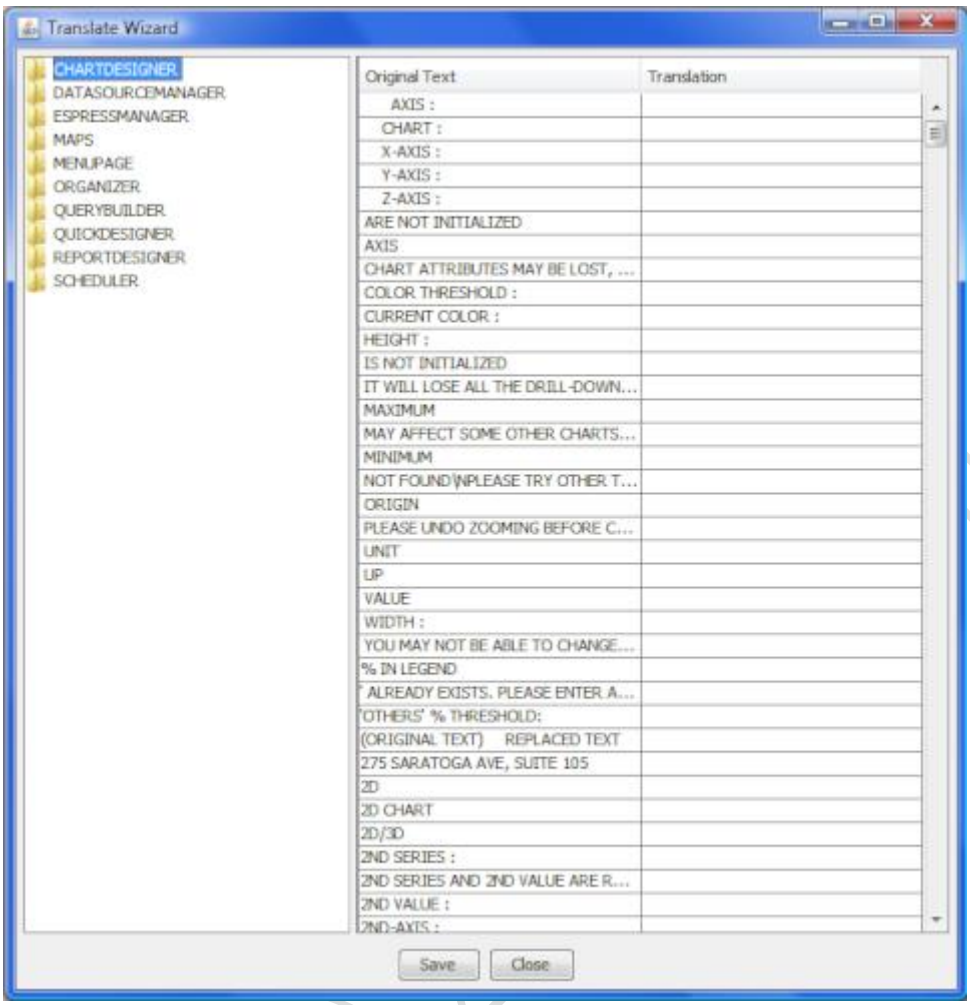
上記のコマンドの例を以下に示します。

```
java -classpath "./lib/ReportAPIWithChart.jar;.;" quadbase.internationalization.TranslateWizard
-file:Chinese.xml -enc:Big5
```

Big5 は、**Chinese.xml** ファイルを保存するために使用されるエンコーディングです。
変換ウィザードが開始されると、次のように表示されます。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

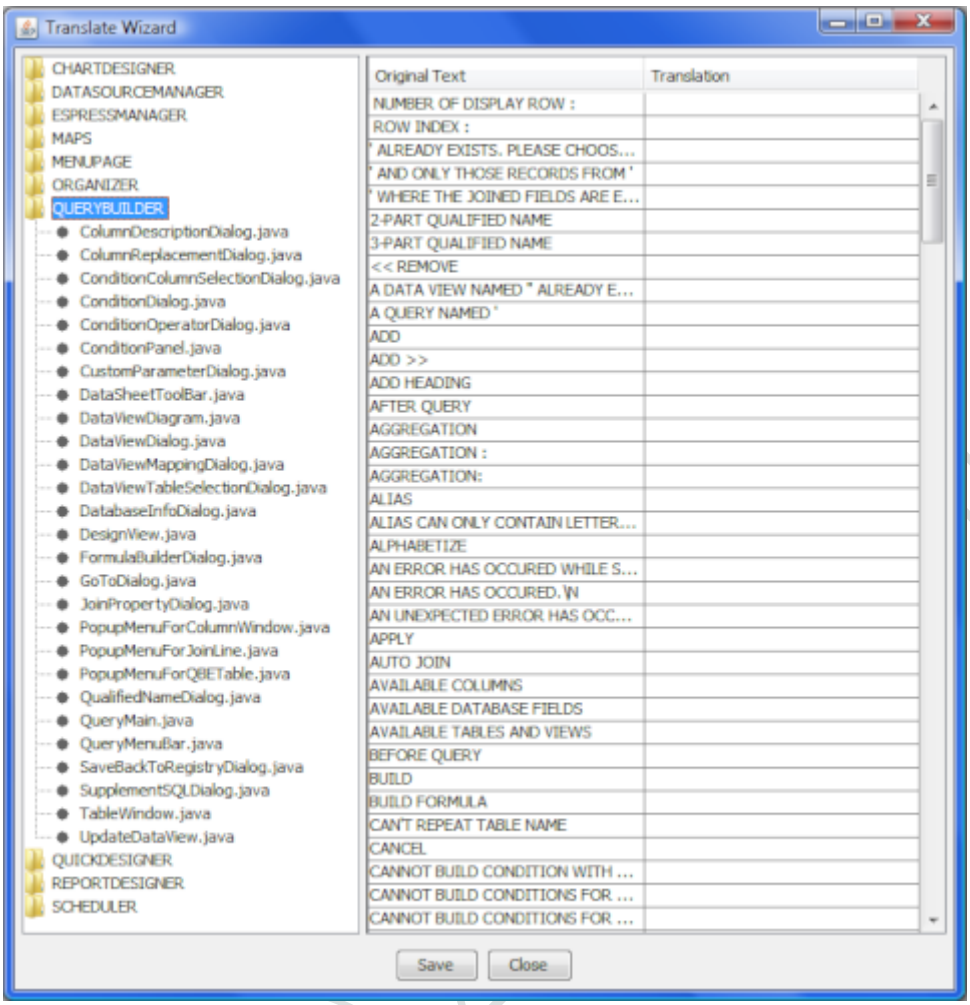


Translate ウィザード

適切な製品を選択すると、さまざまなコード（翻訳可能）が表示されます。製品は、翻訳可能なすべてのファイルを含む左側のツリーに表示されます。たとえば、下の画像には、QUERYBUILDER 製品のツリー内が表示されています。製品ノードを選択して製品全体を翻訳するか、翻訳するファイルを選択するだけです。

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.



翻訳の入力

英語のテキストが左側に表示され、右側に翻訳を入力できます。翻訳を入力するには、セルをクリックしてテキストを入力します。**Save** ボタンをクリックすると、翻訳内容がウィザードを開始したコマンドで指定されたファイルに保存されます。**PREVIOUS** ボタンをクリックすると、前の画面に移動できます。コードの翻訳が既に設定されている場合は、そのコードのすべてのインスタンスに翻訳が表示されます。

EspressReport よりも多くのオプションが Translate ウィザードに存在することに注意してください。たとえば、**ORGANIZER** 製品は EspressReport に存在しないため、この製品で行った変更は表示されません。

EspressReport で **Language.xml**（または変更したコピー）を使用するには、EspressManager、ReportDesigner および Scheduler を起動する Java アプリケーションまたは Java アプレットに**-file** および**-enc** 引数を追加します。

6.1.2.1.1 言語ファイルのアップグレード

ユーザが新しいバージョンにアップグレードするとき、**Language.xml** ファイルも同様にアップグレードする必要があります。言語アップグレードプログラムは、以前にカスタマイズされた **Language.xml** ファイルから翻訳をコピーし、新しいバージョンに追加エントリを追加します。言語アップグレードプ

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

ログラムを使用するには、コンソールウィンドウから EspressReport ディレクトリに移動し、次のコマンドを使用します。

```
java -classpath ".;.¥lib¥ReportAPIWithChart.jar"
quadbase.internationalization.UpgradeLanguageXMLFile -from:oldfile -to:newfile -enc:encoding
```

非 Windows 環境では、セミコロンをコロンとバックスラッシュをスラッシュに置き換える必要があります。

Oldfile は、以前のバージョンの EspressReport のカスタマイズされた **Language.xml** を参照し、newfile は、アップグレードされた EspressReport インストールに含まれる **Language.xml** を参照し、**enc** は、翻訳に使用される言語エンコーディングです。プログラムを実行すると、結果ファイルの名前は **Language.xml** になり、新しいエントリをさらに翻訳するために、Translate ウィザードを使用してファイルを開くことができます。

6.1.2.2 外国語の表示

外国語は、Designer やさまざまなビューアで簡単に表示できます。レポート/チャートに外国語を表示するには、その言語用のフォントをシステムにインストールする必要があります。次に、レポート/チャートで、外国語を含むオブジェクトのフォントを適切なシステムフォントに設定できます。

別のオプションは、JVM 内の **font.properties** ファイルを変更して、デフォルトの JVM フォントで外国語がサポートされるようにすることです。Sun JVM の場合、font.properties ファイルは **jre/lib** ディレクトリの下にあります。別の言語ファイルには、ロシア語の場合は **font.properties.ru**、中国語の場合は **font.properties.zh** などの名前が付けられます。JVM の言語設定を変更するには、現在の **font.properties** ファイルの名前を変更してバックアップし、**font.properties** に必要な言語ファイル JVM で言語設定を変更すると、EspressReport（ダイアログ、セリフ、モノスペースなど）のデフォルトのフォントが外国語で表示されます。

6.1.2.3 外国語の入力

EspressReport インタフェース（たとえば、ReportDesigner）に外部文字を入力するには、次のシステム設定の変更が必要です。

- システムの**デフォルトロケール**は、使用する言語のリージョンに設定する必要があります。
- システムの**入力ロケール**も、使用する言語の地域に設定する必要があります

Windows では、コントロールパネルの**時計、言語、および地域**から設定にアクセスできます。

さらに、外国語の表示で説明した手順に従って、JVM のフォント設定を目的の言語に調整する必要があります。

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

これらの設定が適用されると、レポート、チャートのラベル、クエリ、数式、パラメータに外国語を入力できます。

6.1.2.4 XML エンコーディング

EspressReport はデフォルトで、XML に書き込むときにエンコーディングに UTF-8 文字セットを使用します。これには、データレジストリファイル、XML レポートテンプレート、XML エクスポート、およびグローバルフォーマット情報とフォントマッピングの XML 表現が含まれます。ほとんどのユーザにとって、UTF-8 はすべての言語を完全にサポートしているため、文字セットのエンコードを変更する必要はありません。

このエンコーディングは、ReportDesigner と EspressManager にランタイムパラメータを追加することによって、他の言語でも変更できます。ReportDesigner の場合は、**ReportDesigner.bat/sh** ファイルを変更するか、ReportDesigner の起動に使用するアプレットページを変更します。

XML エンコーディングを変更するには、次の引数を
<EspressReport InstallDir>¥ReportDesigner.bat ファイルまたは**.sh** ファイルに追加します。

```
-xmlEncoding:Encoding.
```

たとえば、**-xmlEncoding:ISO-2022-JP** は、日本語文字セットのエンコーディングを設定します。

アプレットを使用して ReportDesigner を実行している場合、アプレットページに**<PARAM NAME="xml_encoding" VALUE="ISO-2022-JP">**というパラメータを追加する必要があります。

このパラメータは、サーバに対しても設定する必要があります。サーバ設定の詳細については、EspressManager の起動を参照してください。

6.1.2.5 外国語でエクスポート

ほとんどの EspressReport エクスポート形式は、UTF-8 文字エンコードで自動的に生成されます。エクスポートされたレポート/チャートを表示するには、クライアントに適切なシステムフォントがインストールされている必要があります。唯一の例外は PDF エクスポートです。PDF 形式は、クライアントのフォントに依存しません。したがって、文字を正しく表示するには、外国語フォントを PDF 文書に埋め込む必要があります。これを行うには、PDF フォントマッピング機能を使用する必要があります。フォントマッピングは、Designer と API で設定できます。

マシン上で設定されているロケールとは異なるロケールにレポート/チャートをエクスポートする場合は、エクスポートを実行する前に別のエンコードを指定することができます。これは、QbReport で次のメソッドを使用して行われます。

```
public void setExportEncoding(String encoding);
```


Copyright(C) 2018 Climb.Inc. All Rights Reserved.

encoding は、Java2 プラットフォームでサポートされている文字エンコーディングです。たとえば、英語（Windows Latin-1）の文字コードは **Cp1252** です。**java.io** および **java.lang API** で使用される標準名を使用する必要があることに注意してください。

QbReport の次のメソッドを使用して、DHTML エクスポートで使用される文字セットを指定することもできます。

```
public void setHTMLCharSet(String charset);
```

charset は有効な文字セットです。DHTML ドキュメントで使用される英語の文字セットは UTF-8 です。

フルソースコード
エクスポートされた結果

6.1.2.6 テキストデータファイルのエンコーディングの指定

デフォルトでは、EspressReport は任意のテキストデータファイルに対してシステムエンコーディングを使用します。入力テキストデータファイルのエンコーディングは、次のコードブロックを使用して API 経由で設定できます。

```
report.getInputData().setDataFile("sample.dat", false, "ASCII");
```

フルソースコード
エクスポートされた結果

上記のブロックは、**quadbase.reportdesigner.util.IInputData** にある次のメソッドを使用します。

```
public void setDataFile(String fileName, boolean isDataSorted, String encoding);
```

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.

7 更新履歴

版	修正日	修正者	内容
1.0	2018/3/20(火)	Climb	初版

Copyright(C) 2018 Climb.Inc.

Copyright(C) 2018 Climb.Inc. All Rights Reserved.