[EspressChart 7.0]
ユーザガイド

株式会社クライム

Climb Inc.
Growing to Meet Your Needs

作成日: 2017/11/24(金)
更新日: 2024/02/09(金)
バージョン: 1.1

目次

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、[EspressChart]の使用方法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。

- [EspressChart] Ver. 7.0

2 クイックスタート

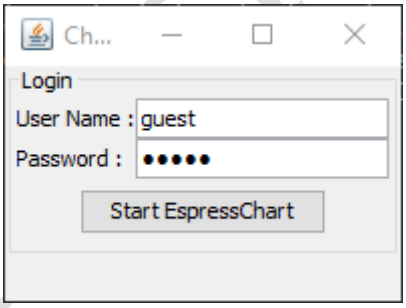
このガイドでは、データソースの設定と Chart Designer を使用したチャートの作成の基本について説明します。Windows マシンでローカルに EC を実行していることを前提としています。

Windows マシンで実行していない場合も、このガイドを引き続き使用できますが、サンプルの Access データベースを使用することはできません。代わりに、同じグラフを作成するために使用できるテキストファイルがあります。

2.1 Chart Designer を開始する

Chart Designer を起動するには、まず Espress Manager を起動する必要があります。Espress Manager を起動するには、インストールのルートディレクトリにある EspressManager.bat または EspressManager.sh ファイルを実行します。Windows インストールの場合、ソフトウェアのインストール時にショートカットを作成するように選択した場合は、**Start** メニューから **Espress Manager** を起動することもできます。

Espress Manager が実行すると、インストールのルートディレクトリにある Designer.bat または.sh ファイルを実行して、Chart Designer を起動できます。Windows インストールの場合、ソフトウェアのインストール時にショートカットを作成するように選択した場合は、**Start** メニューから **Chart Designer** を起動することもできます。Chart Designer を起動すると、ログインウィンドウが表示され、ユーザ名とパスワードの入力を求められます。ユーザ名として **guest** を入力し、パスワードフィールドを空白のままにします。



Start EspressChart をクリックすると、**Chart Designer** が新しいウィンドウとして開きます。

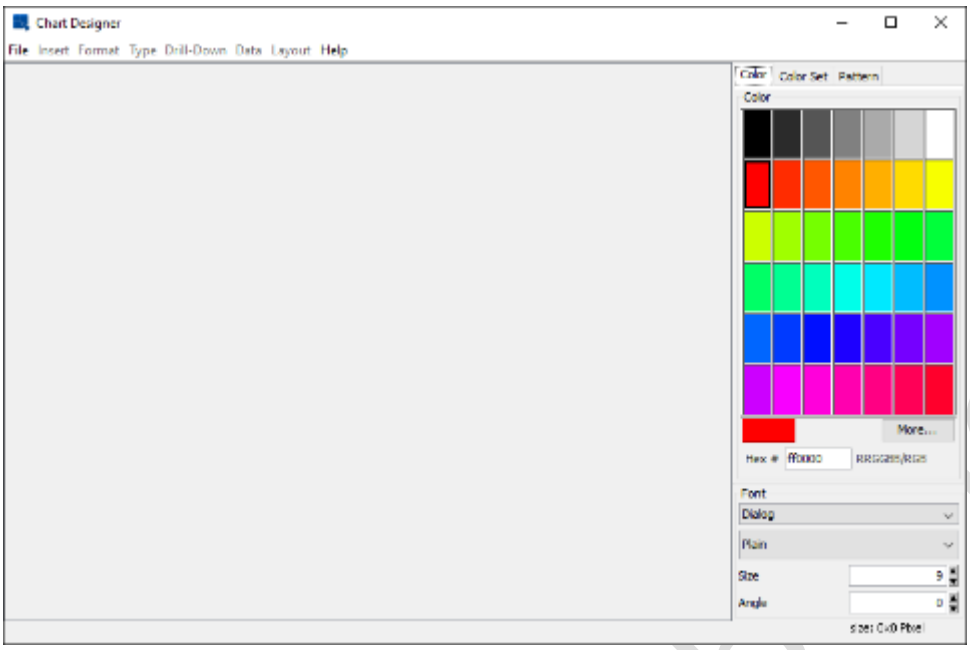
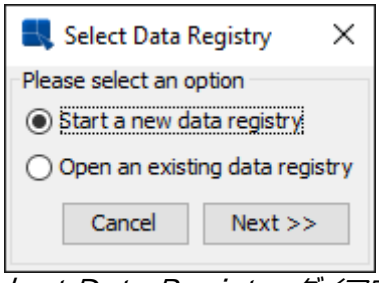


Chart Designer のメインウィンドウ

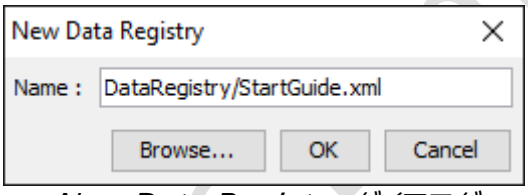
2.2 新しいチャートを開始してデータソースレジストリの設定する

新しいグラフを開始するには、**File** > **New** を選択します。新しいデータソースレジストリを作成するか、既存のレジストリを使用するかを尋ねるダイアログが表示されます。データレジストリは、テキストと XML ソースのデータベース接続、クエリ、およびファイルの場所を格納する XML ファイルです。**Start a new data registry** を選択し、**Next** をクリックします。

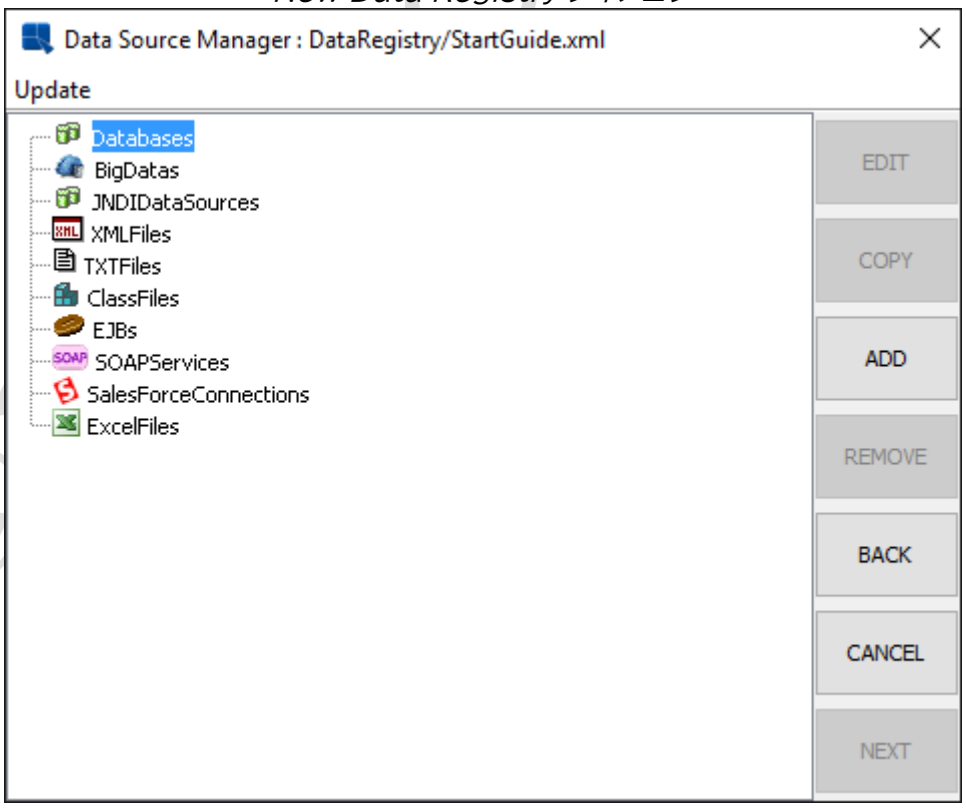


Select Data Registry ダイアログ

その後、データレジストリの名前を入力するよう求められます。必要な名前を付けて、同じ名前の XML ファイルを DataRegistry ディレクトリに作成します。名前を指定すると、Data Source Manager が開きます。これは新しいレジストリなので、ウィンドウ内のデータソースタイプの下にノードはありません。



New Data Registry ダイアログ

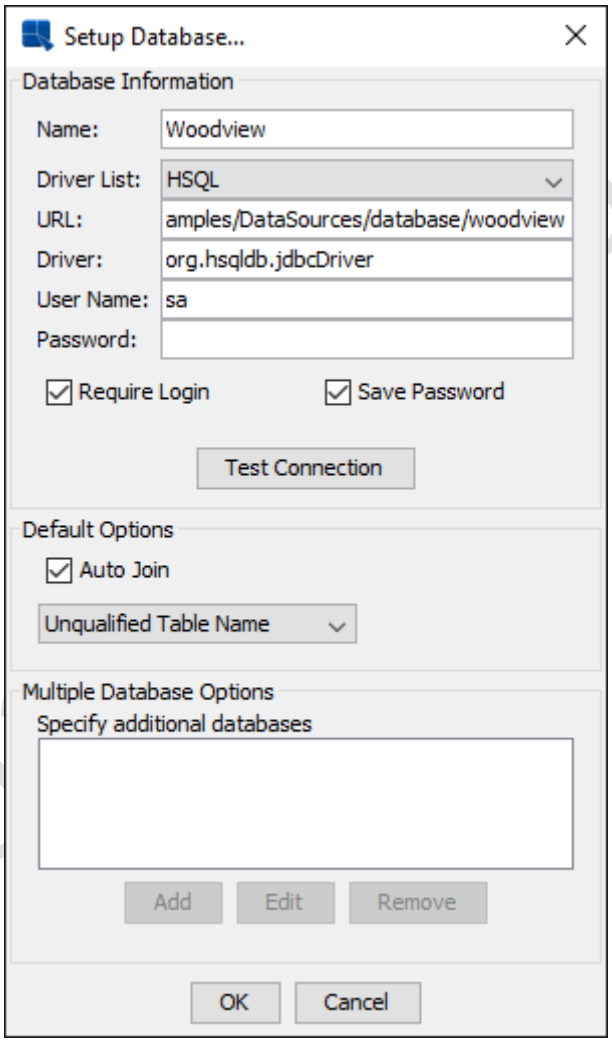


Data Source Manager

2.3 レジストリにデータソースを設定する

設定する最初のデータソースはデータベースです。EspressChart には 2 つのサンプルデータベースが付属しています。1 つは、純粋な Java アプリケーションである HSQL です。もう 1 つは MS Access データベースです。どちらも同じデータを含んでいます。

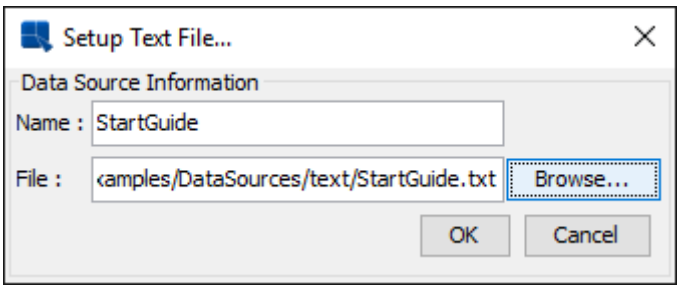
Chart Designer を起動し、データレジストリを再度開きます。**Data Source Manager** から、左側のフレームの **Database** をクリックし、**Add** をクリックします。ダイアログが表示され、新しいデータベースの接続情報を入力するよう求められます。データベースの名前の欄には **Woodview** と入力し、URL に **jdbc:hsqldb:help/examples/DataSources/database/woodview** と入力し、ドライバには **org.hsqldb.jdbcDriver** と入力します。**Require Login** と **Save Password** の両方のボックスをクリックします。次に、ユーザ名として **sa** を入力し、パスワードを空白のままにします。



Setup Database dialog

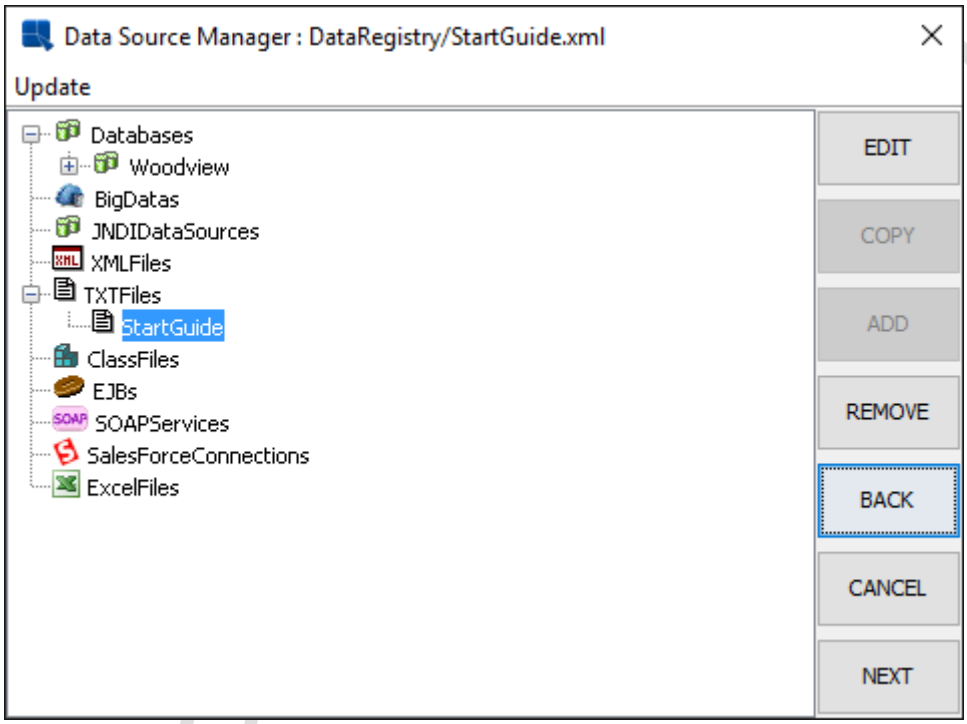
オートジョイントテーブル名のプロパティをそのままにして、**Test Connection** をクリックして、情報が正しく入力されていることを確認します。次に **OK** をクリックして、**Data Source Manager** ウィンドウを立ち上げます。そこには、**Woodview** のデータベースの下に新しいノードがあります。

テキストデータソースを追加するには、Data Source Manager の左側のフレームにある **TXTFiles** ノードをクリックし、**Add** をクリックします。ダイアログが表示され、データソースの表示名とテキストファイルの場所を指定するように求められます。任意の名前を入力し、**Browse** をクリックします。**help/examples/DataSources/text/**に移動し、**StartGuide.txt** を選択します。



Data Source Manager

OK をクリックすると、Data Source Manager が表示され、入力したばかりのテキストデータソースの **TXTFiles** に新しいノードが常駐されます。

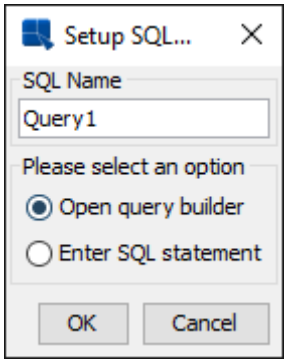


Data Source Manager

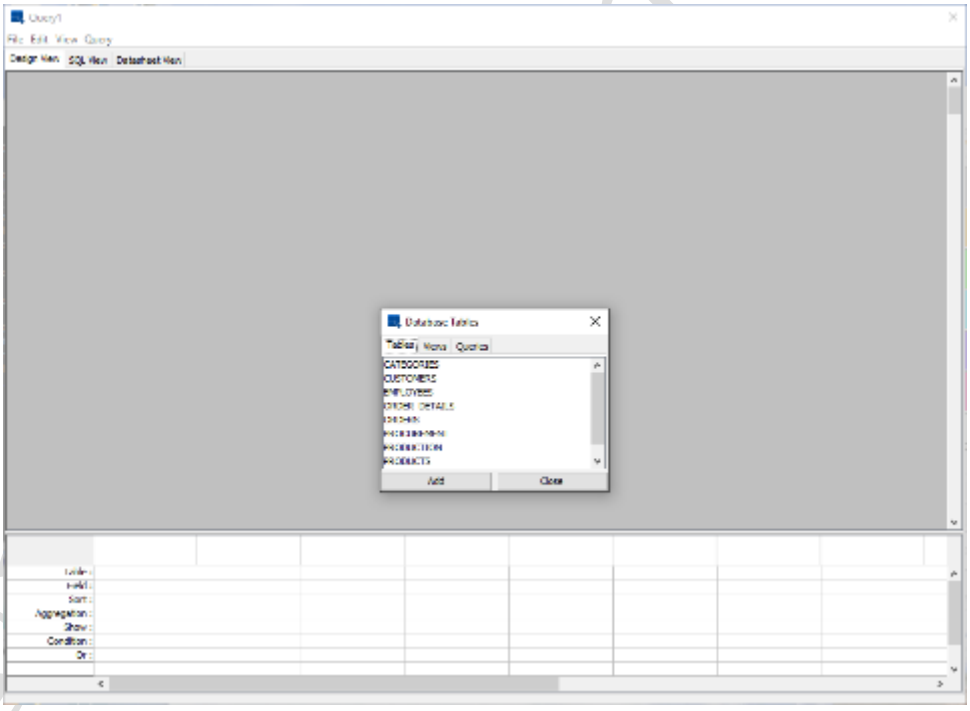
help/examples/DataSources/text/に **Sample.dat** という追加のサンプルテキストファイルがあります。この拡張サンプルファイルには、各データタイプのデータが含まれています。このファイルを使用して、さまざまなグラフの種類とオプションを調べることができます。

2.4 クエリを作成する

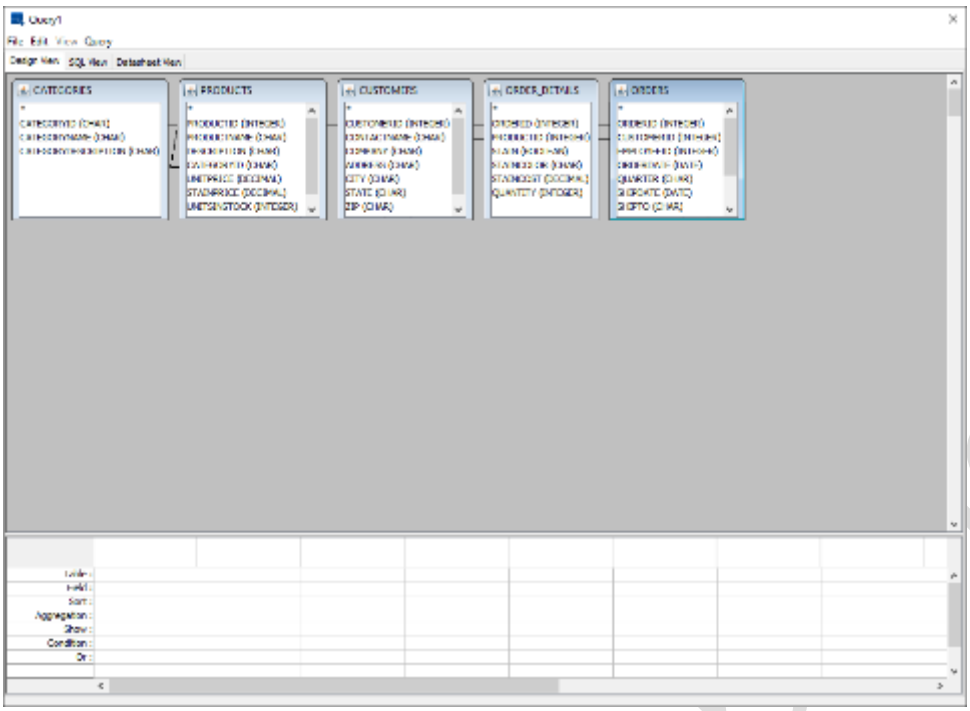
新しいクエリを作成するには、Data Source Manager の左側のフレームにある **Woodview** ノードをクリックして展開すると、2 つのサブノードが表示されます。1 つは**クエリ（Query）**と呼ばれ、もう 1 つは**データビュー（Data View）**と呼ばれます。**Queries** ノードを選択し、**Add** をクリックします。クエリの名前を指定し、クエリビルダを起動するか SQL 文を入力するかを選択するダイアログが表示されます。



任意の名前を入力し、**Open query builder** を選選択し、**OK** をクリックするとクエリビルダが起動します。メインのクエリビルダウィンドウの上部にある Woodview の全てのテーブルを含む別々のウィンドウが表示されます。

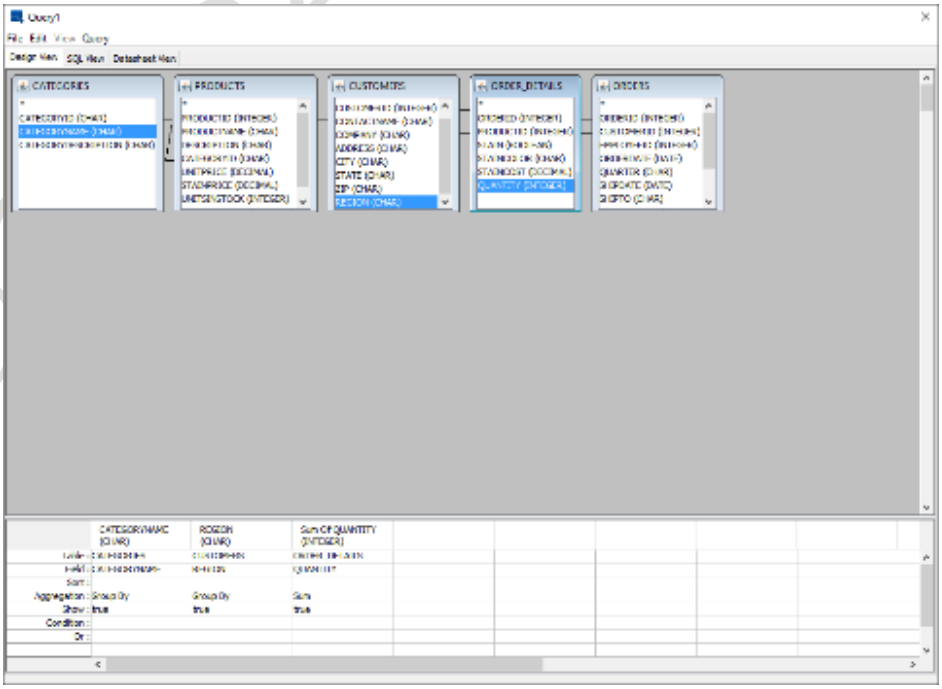


Tables タブから、**Categories**、**Customers**、**Order Details** および **Orders tables** をそれぞれクエリに追加し、**Add** をクリックします。テーブルが追加されると、クエリビルダウィンドウの上半分に表示されます。テーブルは、それらを結ぶ黒線で示されるように自動結合されます。テーブルを追加したら、**Tables** ウィンドウの **Close** をクリックします。

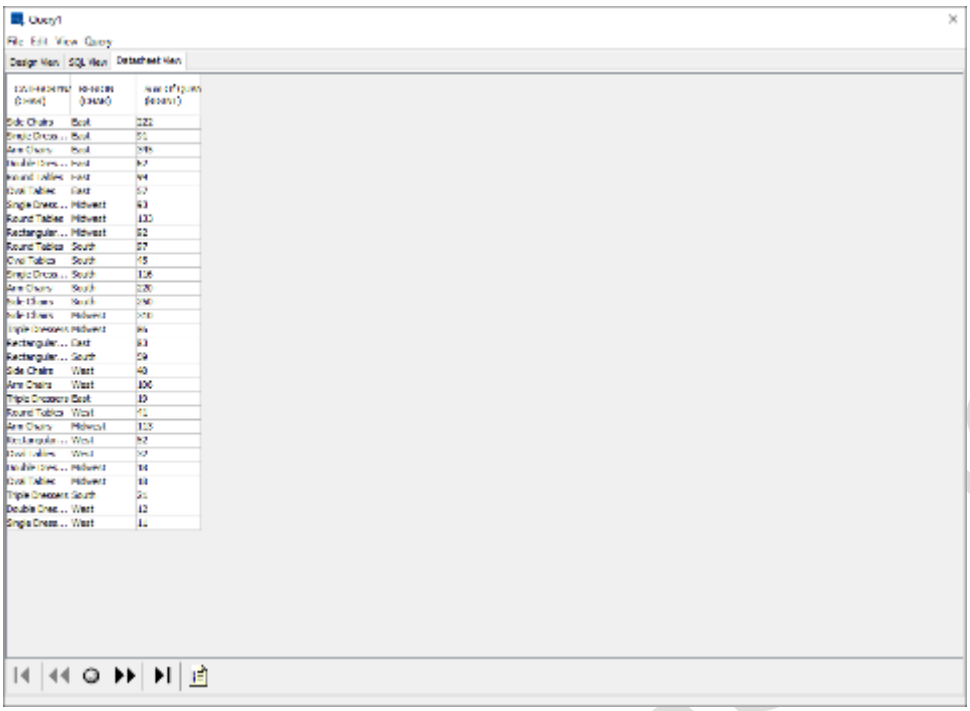


フィールドにクエリに追加するには、テーブルウィンドウ内のフィールドをダブルクリックするか、下半分の **Table** フィールドや **Field** フィールドをダブルクリックするか、ドロップダウンメニューから QBE (query by example)の部分をダブルクリックします。いずれかの方法を使用して、Categories テーブルの CategoryName、Customers テーブルの Region、および Order_Details テーブルの Quantity というクエリに、以下のフィールドを追加します。

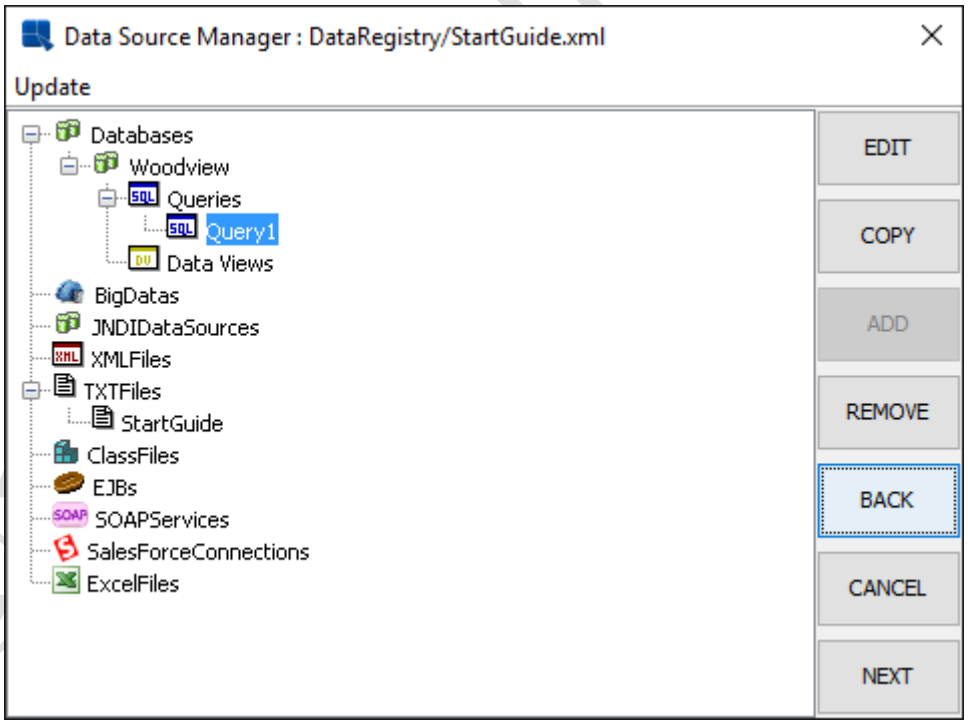
フィールドを追加したら、CategoryName 列の **Aggregation** フィールドをダブルクリックします。集約タイプとして **Group** を選択します。QBE フレーム内の他の場所をクリックすると、他の列はすべて集約としてグループ化されます。Aggregation 列の **Quantity** フィールドをダブルクリックし、aggregation を **Sum** に変更します。



フィールドを入力したら、**Datasheet View** タブをクリックして、クエリの結果をプレビューできます。クエリが実行され、製品カテゴリと販売地域別に分けた合計販売数量が表示されます。

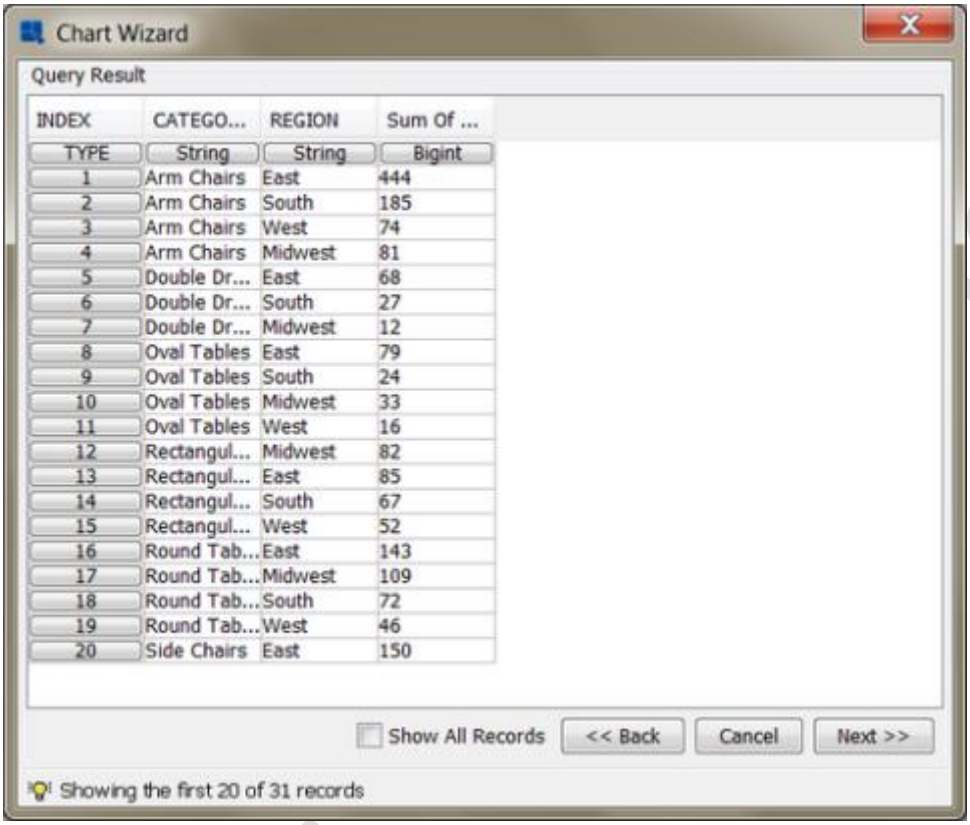


クエリの作成が完了したら、**File** > **Done** を選択します。クエリビルダが終了し、Data Source Manager が復旧します。新しく作成されたクエリの下にノードがあります。



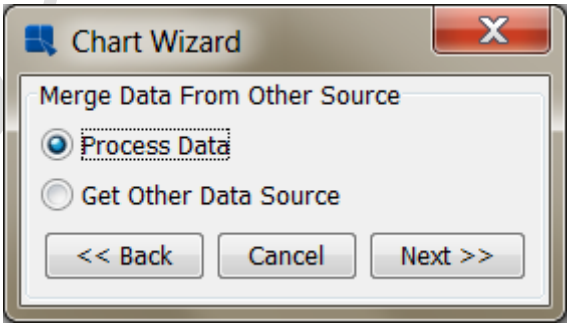
2.5 チャートを作成する

チャートを作成するには、まず使用するデータソースを選択する必要があります。作成したクエリまたは StartGuide テキストファイルのいずれかを選択し、**Next** をクリックします。新しいウィンドウが開き、データの最初の 20 レコードが表示されます。



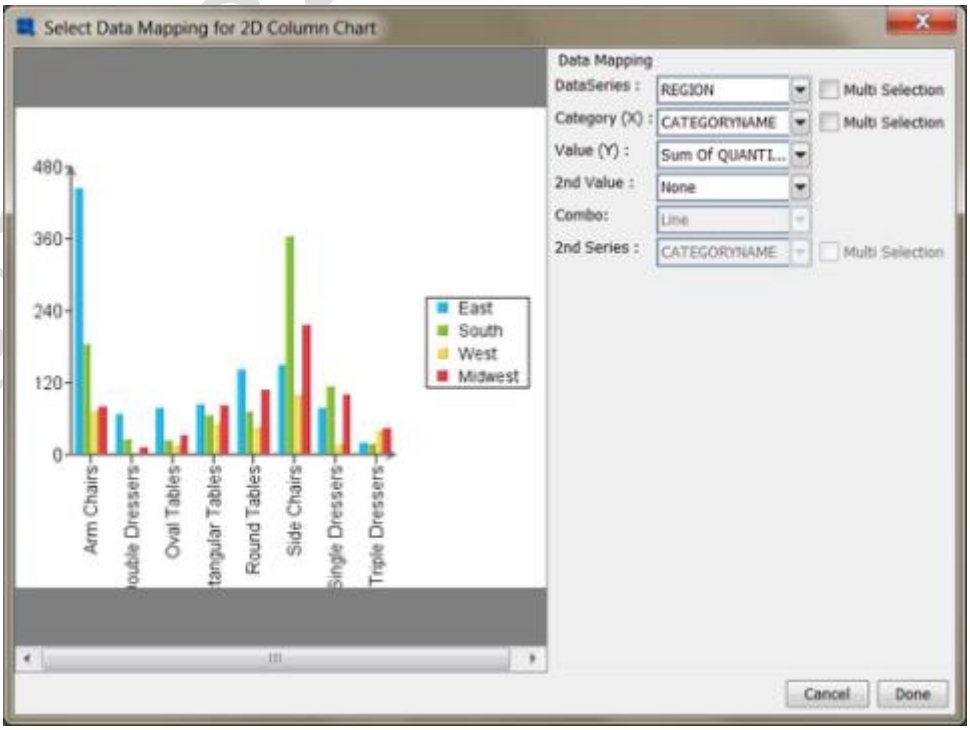
INDEX	CATEGO...	REGION	Sum Of ...
TYPE	String	String	Bigint
1	Arm Chairs	East	444
2	Arm Chairs	South	185
3	Arm Chairs	West	74
4	Arm Chairs	Midwest	81
5	Double Dr...	East	68
6	Double Dr...	South	27
7	Double Dr...	Midwest	12
8	Oval Tables	East	79
9	Oval Tables	South	24
10	Oval Tables	Midwest	33
11	Oval Tables	West	16
12	Rectangul...	Midwest	82
13	Rectangul...	East	85
14	Rectangul...	South	67
15	Rectangul...	West	52
16	Round Tab...	East	143
17	Round Tab...	Midwest	109
18	Round Tab...	South	72
19	Round Tab...	West	46
20	Side Chairs	East	150

このダイアログから、**Next** をもう一度クリックします。データの処理や別のデータソースの取得を求めるダイアログが表示されます。**Process Data** を選択し、**Next** をクリックします。



グラフウィザードの以下の画面では、使用するグラフの種類を選択できます。ラジオを使用して、2D と 3D のグラフタイプを切り替えることができます。2D の縦横グラフを使用するタイプとして選択し、**Next** をクリックします。

以下の画面では、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。デフォルトでは、左から右に数えて最初の列（CategoryName）はデータシリーズとして、第 2 列（Region）はカテゴリとして、第 3 列（Quantity）は値としてマッピングされます。マッピングを変更するには、データシリーズとして Region を、カテゴリとして CategoryName を選択します。値オプションは同じままにします。

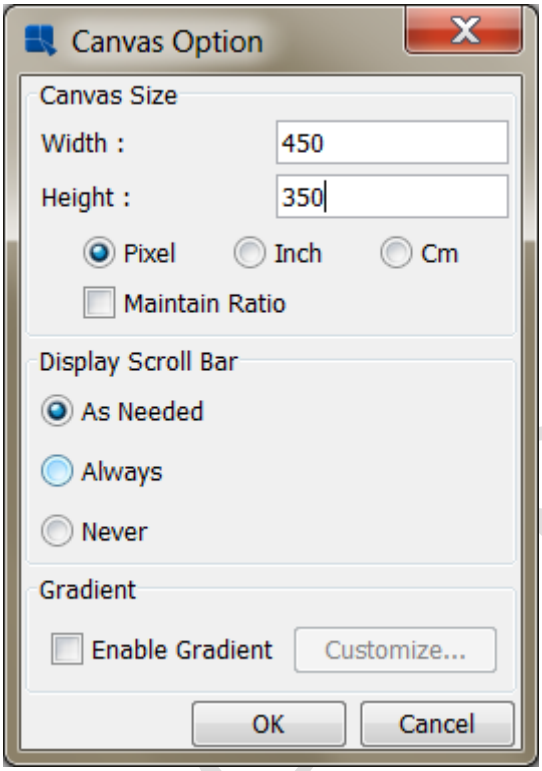


必要な変更を加えたら、**Apply** をクリックします。変更が反映されるように、マッピングウィンドウにグラフが再描画されます。マッピングオプションの設定が完了したら、**Done** をクリックすると、チャートをカスタ

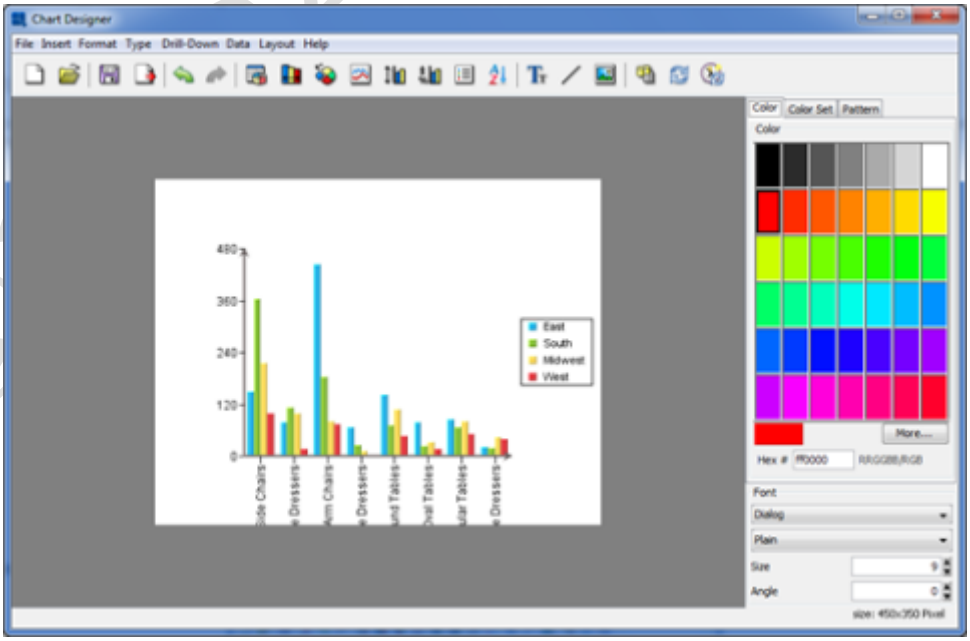
マイズできる **Chart Designer** ウィンドウが表示されます。

2.6 チャートのプロパティをカスタマイズする

チャートのデフォルトのキャンバスサイズは 600×500 ピクセルです。チャートキャンバスのサイズを調整するには、Format メニューから、**Canvas** を選択します。これにより、チャートキャンバスのサイズをピクセル、インチ、またはセンチメートルのいずれかで変更できるダイアログが表示されます。

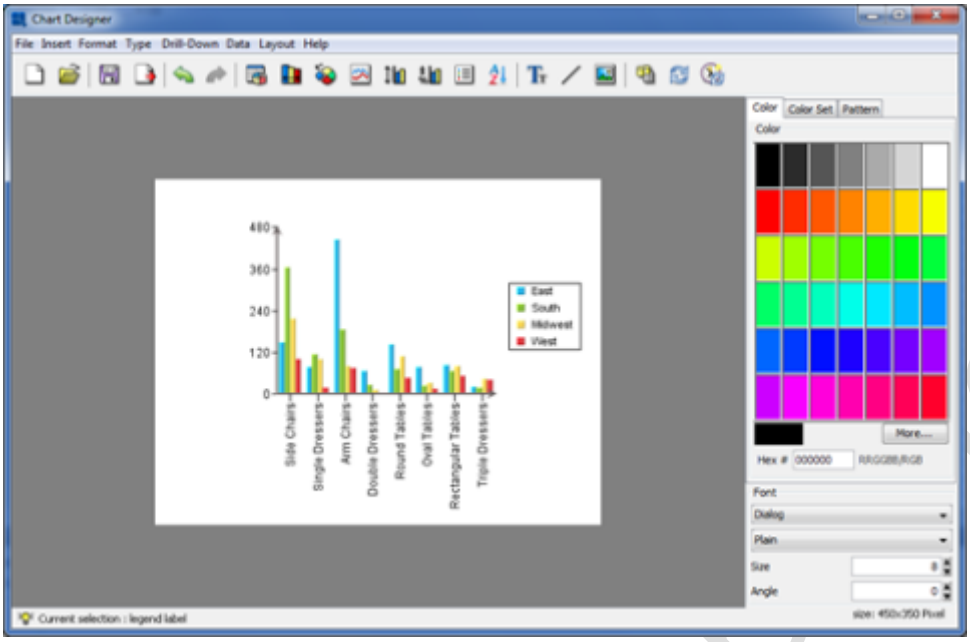


Maintain Ratio のチェックを外し、キャンバスの寸法 450×350 ピクセルに変更し、**OK** をクリックします。Chart Designer でキャンバスのサイズが変更されるようになります。

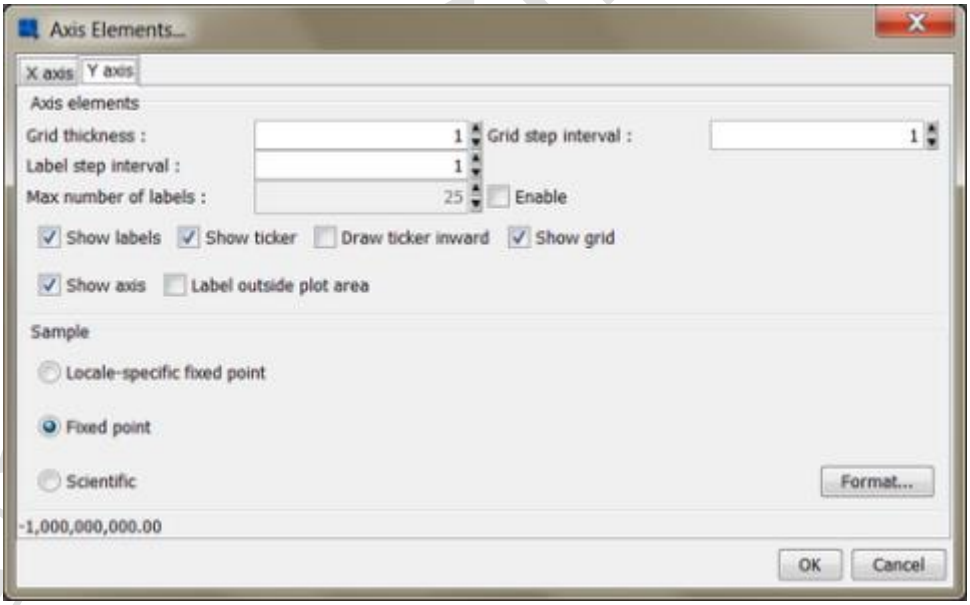


チャートキャンバスが縮小されたので、グラフプロットは小さく表示され、X 軸ラベルの一部は切り捨てられます。これは、チャートのサイズを変更することで調整できます。グラフプロットをクリックしてドラッグし、それを右クリックしてドラッグするとサイズの変更、凡例を移動させることができます。これらのオプシ

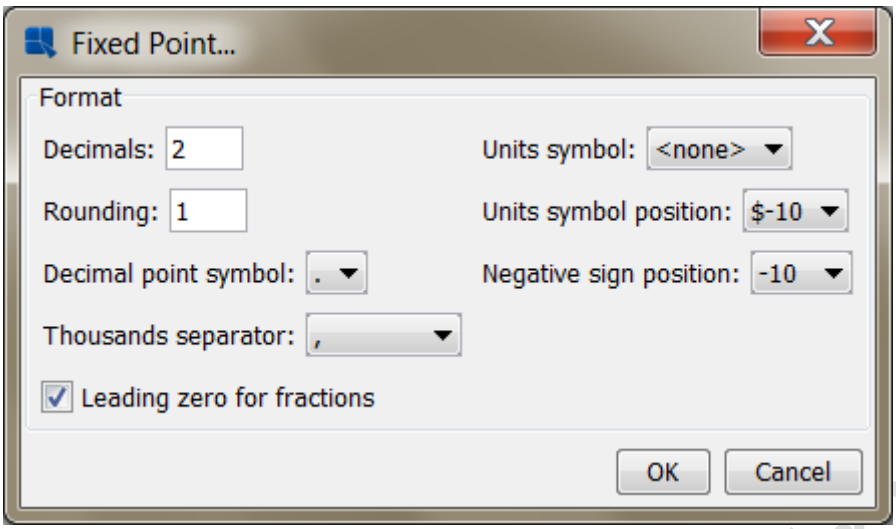
ョンを使用して、チャートプロットと凡例をキャンバス上に配置します。



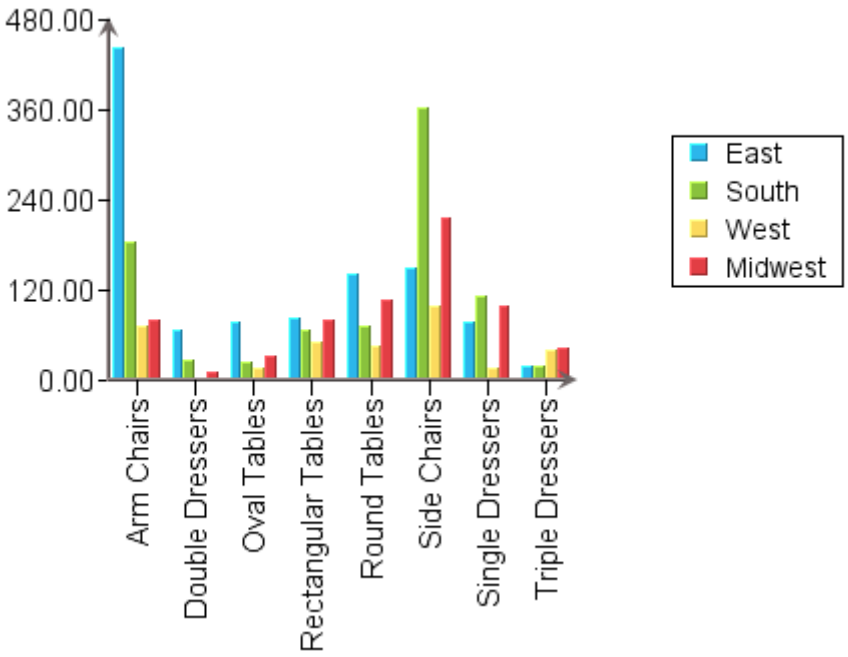
次に、軸ラベルの書式を変更することができます。これを行うには、**Axis Elements**  をクリックします。これにより、各チャート軸に異なるオプションを設定できるタブ付きのダイアログが表示されます。**Y Axis** タブをクリックすると、値軸のオプションが表示されます。



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** をクリックすると、数値のデータの書式オプションを設定できる追加ダイアログが表示されます。**Decimal**（小数点以下の桁数）を 2 として選択し、**OK** をクリックします。

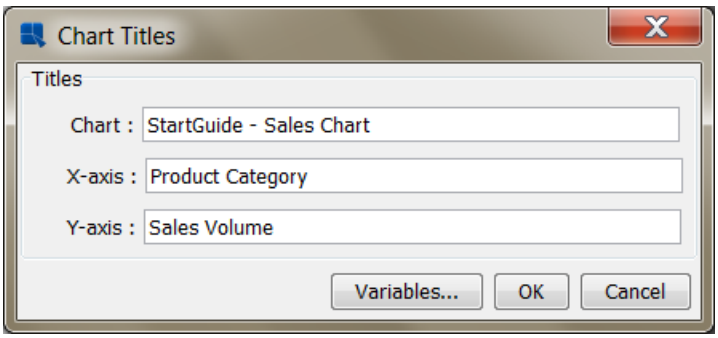


もう一度 OK をクリックして Axis Elements ダイアログを閉じると、指定した変更がグラフに反映されます。

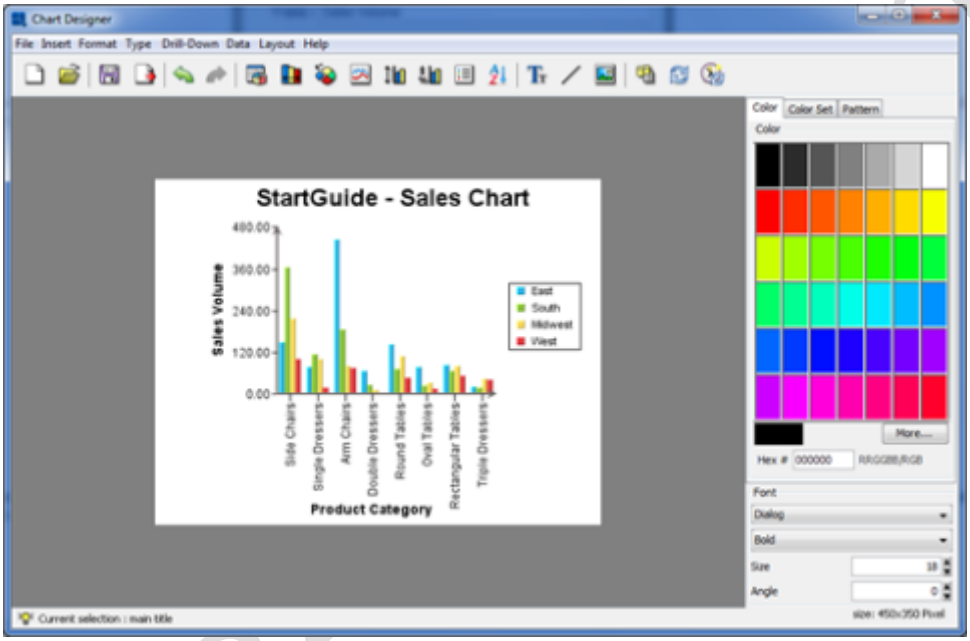


グラフの要素の色を変更するには、**Randomize Color** をクリックすると、データ要素の様々な色の組み合わせが繰り返されます。任意の色をクリックして選択することもでき（デザインウィンドウの左隅にあるヒントは現在選択されている要素を示します）、カラーパネルから新しい色を選択します。

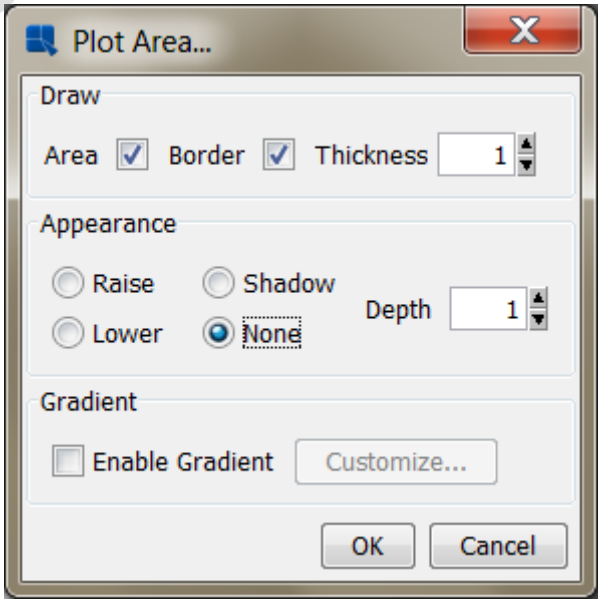
次にチャートにタイトルを追加することができます。これを行うには、**Insert** > **Titles** を選択して、チャートと各軸のタイトルを入力できるダイアログを表示させます。



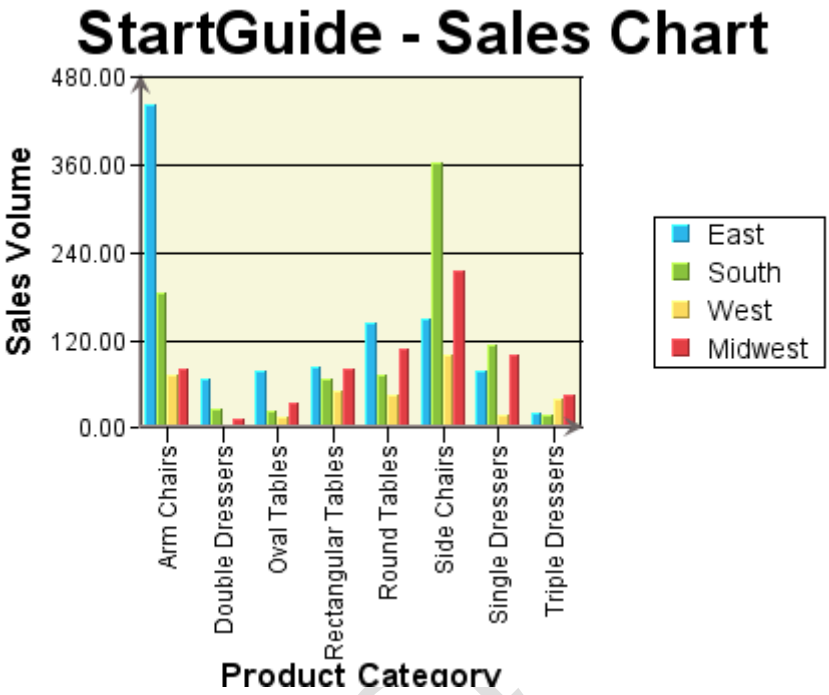
様々な要素に使用するタイトルを入力し、**OK** をクリックします。タイトルがチャートに追加されます。タイトルは自動的に配置されますが、チャートキャンバス上テキストをクリックしてドラッグすることで、タイトルを手動で調整できます。



最後に、プロットの背景と枠線を追加して、プロットエリア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** メニューから **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。

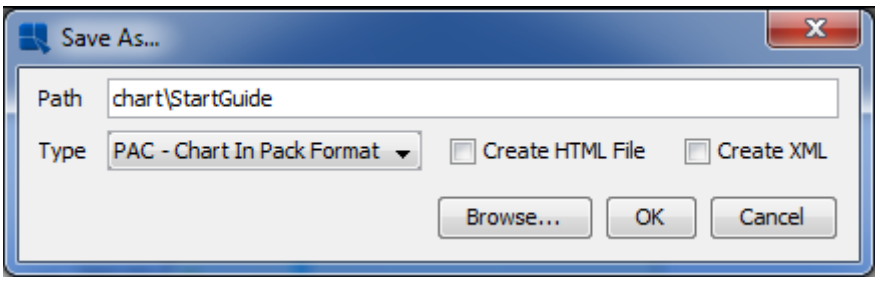


Draw の **Area**（プロットエリア）と **Border**（枠線）の両方を 1 の太さ（**Thickness**）で描画する場合に選択します。外観に **None** を指定します。オプションを指定したら、**OK** をクリックすると、グラフのプロットエリアが変更されます。プロットエリアの背景色をクリックして選択、カラーパネルで色を変更することで、プロットエリアの背景色を変更できます。



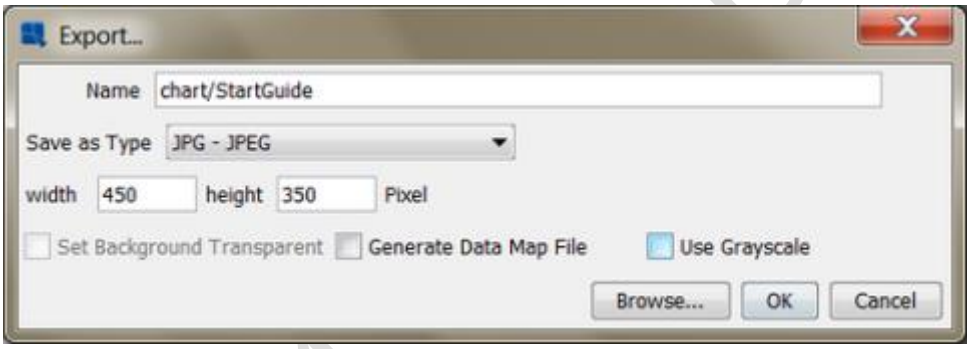
2.7 チャートを保存してエクスポートする

チャートのカスタマイズが終了したら、**File** > **Save As** を選択してチャート定義を保存できます。これにより、ファイル名と他のいくつかの保存オプションを指定するダイアログが表示されます。

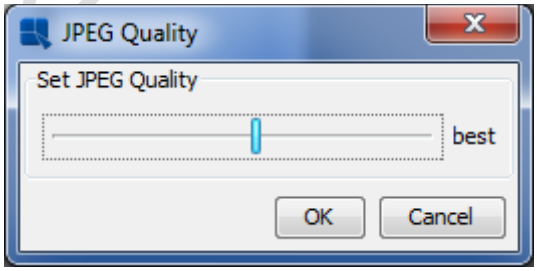


CHT 形式でグラフを保存し、グラフのファイル名を指定する場合に選択します。これにより、ルートディレクトリにデータが保存されたチャートが保存されます。必要に応じて別の場所を指定することができます。

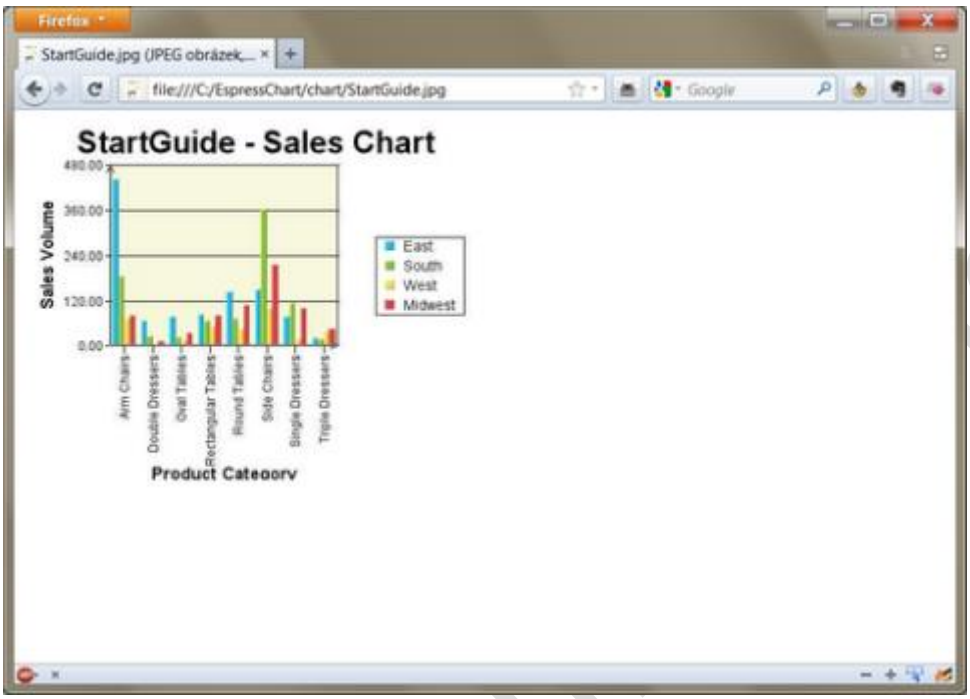
チャートを保存したら、ツールバーの **Export** をクリックしてグラフを書き出すことができます。これにより、チャートのエクスポートオプションを指定するダイアログが表示されます



生成されたイメージファイルの名前を指定し、エクスポート形式として JPEG を選択します。**OK** をクリックすると、生成されたイメージの画質を指定するように求められます。



最高品質（スライダーを一番右にする）を指定し、**OK** をクリックします。JPEG ファイルは、インストールのルートディレクトリに書きこまれます。生成されたファイルを Web ブラウザで開くと、画像をプレビューできます。



3 EspressChart 7.0 の概要

EspressChart は、動的チャートを作成して配置するための強力なツールセットです。EspressChart を使用すると、洗練された魅力的なチャートを簡単に作成し、事実上あらゆる環境、プラットフォーム、または設定でレンダリングすることができます。
EspressChart では、ユーザがデータソースを照会し、レンダリングすることができます。ユーザがデータソースを照会し、使いやすいポイント＆クリック環境でチャートを作成することが可能な様々なデータへの接続ツールを備えた強力なデザインインターフェイスが含まれています。その他にも、堅牢なオブジェクト指向の API も含まれているため、ユーザは動的なチャートや Chart Designer をアプリケーション、JSP、サーブレットに組み込むことができます。
EspressChart は 100%Pure Java™の認定を受けており、ネイティブライブラリを使用していないため、ほぼすべてのプラットフォームでチャートをレンダリングできます。また、JDBC データソース(Excel スプレッドシートを含む)に直接接続することも、テキスト、XML ファイル、または EJB からデータを描画することもできます。柔軟性を高めるために、ユーザはデータセットを引数または配列として直接グラフに渡すことができます。実行時には、GIF、JPEG、PNG、SVG、および SWF などの一般的な画像フォーマットでチャートをレンダリングすることや、アプレットでインタラクティブに表示することができます。

3.1 このガイドについて

EspressChart ユーザーズガイドは大まかに 3 つのセクションに分かれています。
最初のセクションでは、EspressChart の各種コンポーネントのインストール、アーキテクチャ、セットアップ、設定などの背景情報について説明します。
2 番目のセクションでは、チャート作成の基礎、Chart Designer、および EspressChart の各種機能について説明します。チャートを完全にプログラムで開発することを計画している場合でも、このセクションを参照して EspressChart の基本的な機能と用語を把握することをお勧めします。
3 番目のセクションでは、チャートのプログラミングとデプロイについて説明します。Chart Viewer アプレット、Chart API、サーブレット/JSP との統合、およびデプロイメントオプション/コンフィグレーションについて説明します。
このガイドの API セクションには、概要、使用モード、API で EspressChart の機能を使用する方法の説明が含まれています。インストールには、2 つの追加リソースも含まれています。

API の使い方
より広範囲な API チュートリアルは下記をご参照ください。具体的には、サンプルコードを使用して、API の基本的なチャート作成/操作関数の多くを行う方法について詳しく説明します。このガイドは、EspressChart インストールの次の場所にあります。
<EspressChartInstallDir> /help/api_HowTo/index.html

API JavaDoc
完全な API JavaDocs は、EspressChart インストールの次の場所に含まれています:
help/apidocs/index.html

サンプルチャートと API コード(サーブレットと JSP のサンプルを含む)については、下記ディレクトリをご参照ください。
<EspressChartInstallDir>/help/examples

3.2 EspressChart のコンポーネント

EspressChart は、4つの主要コンポーネントで構成されています。

Chart Designer
Chart Designer は、ユーザが視覚的なポイントとクリック環境でチャートを作成および編集できるインタラクティブなアプリケーションです。アプリケーションとしてローカルに実行することも、Web ブラウザ経由でアプレットとしてロードすることもできます。ほとんどのチャートプロパティは、コードを記述することなくデザイナで簡単に設定および変更できます。完成したチャートは、表示または印刷のために多数の静的フォーマットに直接エクスポートできます。グラフは、テンプレート(.tpl)ファイルとして保存することもできます。テンプレートにはチャート属性(色、サイズなど)が格納されますが、チャートデータは格納されません。テンプレートが読み込まれるたびに、ソースから最新のデータが取得され、プログラミングせずにグラフを更新することができます。テンプレートは、他のチャートやチャートオブジェクト(API)にも適用することができ、書式情報をチャート間で移動することができます。

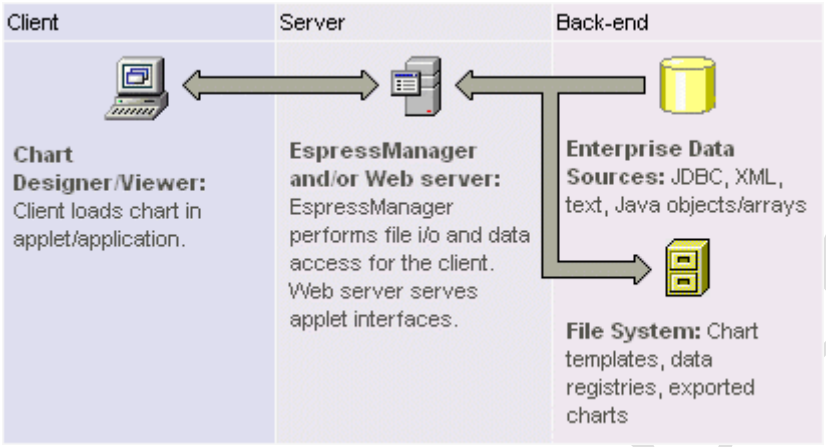
Chart Viewer
Chat Viewer は、チャートをリモートから見ることを可能にするアプレットです。これにより、表示したチャートを回転、サイズ変更、ズーム、パンすることができます。また、データポイントをクリックして、データポイントに関連付けられたデータを修正することや、別のチャートにリンクすることができます。

Chart API
Chart 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ーフェイスで、アプリケーション側(アプレット側)のサーバ側またはクライアント側のグラフを作成、カスタマイズ、展開することができます。これは、高度な 2D および 3D チャート作成ライブラリ関数のセットに Java パッケージを提供します。グラフはわずか 1 行のコードで構築できます。

EspressManager
EspressManager は EspressChart に「バックエンド」を提供します。データベースへのアクセスおよびクエリに必要な機能を提供し、Chart Designer を使用する場合、EspressManager はクエリユーティリティとデータベースをシームレスに接続します。EspressManager はデータバッファリング、ファイル I/O、ユーザ認証も提供します。Chart Designer を実行する時は、EspressManager が必要ですが、EspressManager なしで API や Chart Viewer を使用しチャートをデプロイすることもできます。

3.3 EspressChart アーキテクチャ

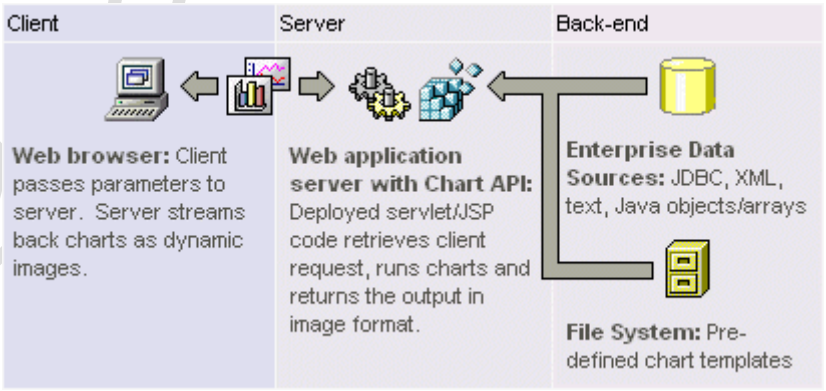
EspressChart には、設計時と実行時の両方で実行できるさまざまな構成があります。設計時に、データアクセスツールを含む Chart Designer の GUI インターフェイスは、クライアントマシン上のアプリケーションとして、またはクライアントサーバーアーキテクチャ内のアプレットとしてロードできます。次の図は、デザイン時にチャートデザイナで実行されている EspressChart を示しています。



EspressChart Designer のアーキテクチャ

Chart Designer が実行されている場合、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 はチャートデザイナと一緒に実行する必要があります。

ランタイムで、EspressManager を実行する必要はありません。EspressChart は、他のアプリケーション環境に組み込まれて実行されるように設計されており、API クラスとチャ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 EspressChart を示しています。



サーブレット環境での EspressChart

アプリケーションサーバで実行する場合、Chart API を使用してサーブレットと JSP 内のチャートを生成し、生成されたチャートをイメージとしてクライアントブラウザにストリーミングできます。 クライアントはチャートビュアーアプレットを Web ページに読み込んでチャートを表示することもできます。

EspressChart 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。Chart API をチャートビュアーと組み合わせて使用すると、アプリケーションインターフェイスでチャートを表示/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に含めることができるため、EspressManager は実行する必要はありません。上記の図は、すべての展開構成を示すものではありません。 EspressChart 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。

4 インストール/設定

EspressChart インストーラには、Windows 版、Unix 版、Mac OS X 版、Pure Java 版の 4 種類があります。EspressChart 自体は Pure Java ですが、各環境用のバージョンを使用すると、選択したプラットフォームに EspressChart を簡単にインストールできます。

4.1 インストーラの実行

Windows Version

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

Unix Version

Unix インストーラを起動するには、**installEC.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 Version

Mac インストーラを起動するには、**InstallEC.app** ファイルを解凍するために **InstallEC.zip** ファイルをダブルクリックします。**InstallEC.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 Version

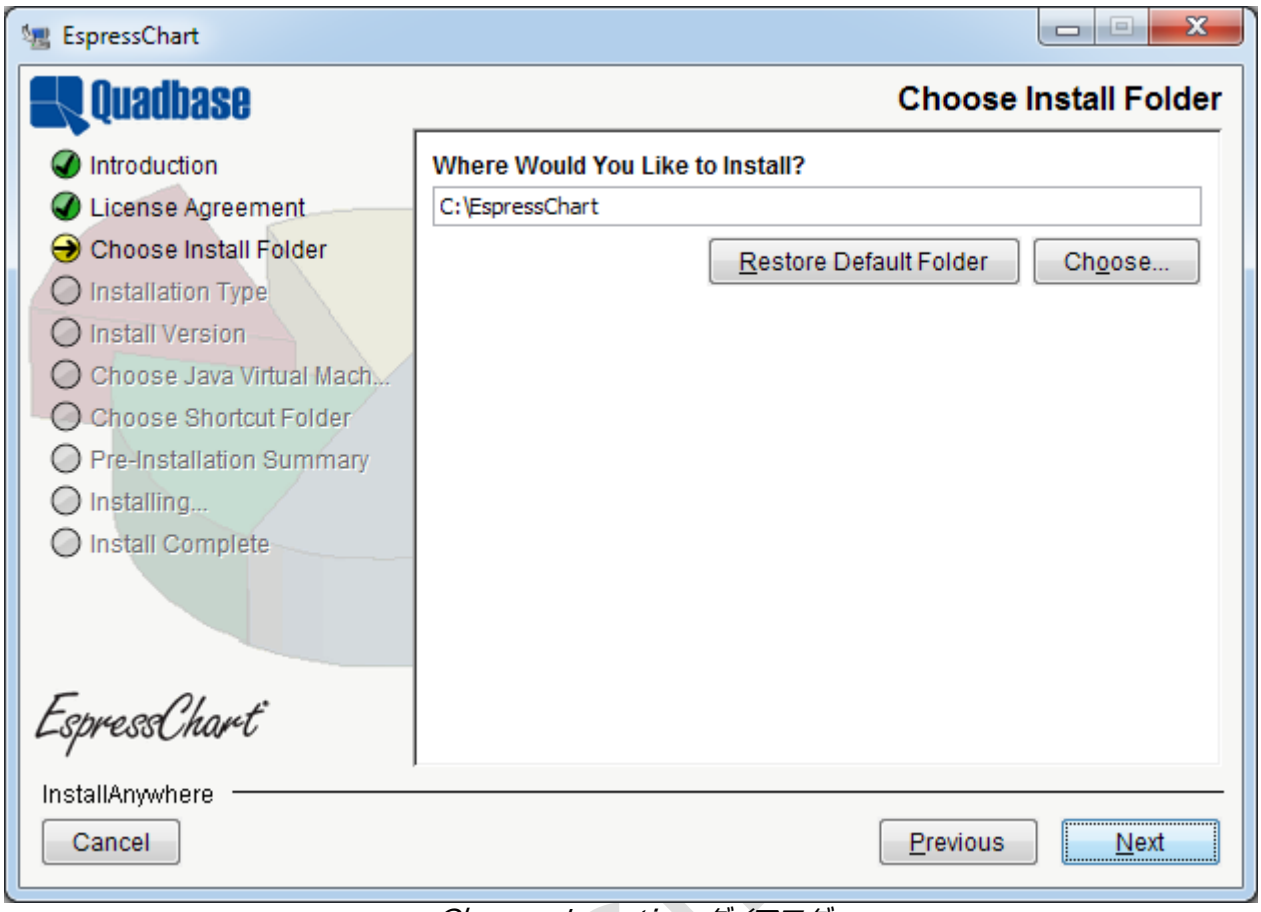
Pure Java インストーラを起動するには、Java 仮想マシン(Java 1.2 以上に相当)が、EspressChart がインストールされるマシンにインストールされている必要があります。JVM がパスに含まれていることを確認します(または **installEC.jar** ファイルを JVM と同じディレクトリに移動します)。コマンドプロンプトから、**installEC.jar** ファイルを置いたディレクトリにナビゲートし、次のコマンドを入力します。

```
java -jar installEC.jar
```

インストーラが起動します。

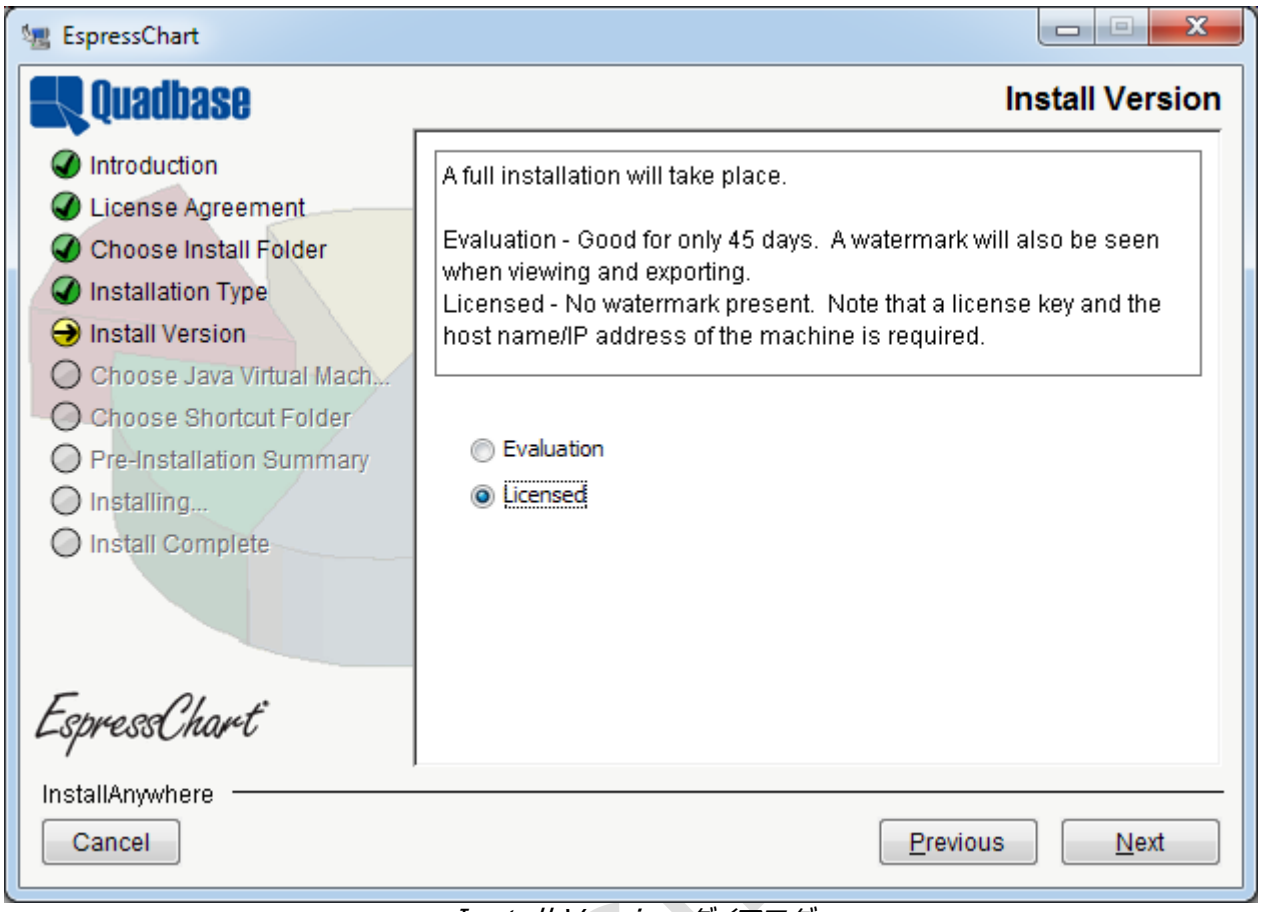
インストールプログラムが起動し、使用許諾契約書に同意すると、最初のオプションで、EspressChart をインストールするディレクトリを指定するよう求められます。

デフォルトの場所は**¥ EspressChart ¥**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することや、ディレクトリを参照することができます。



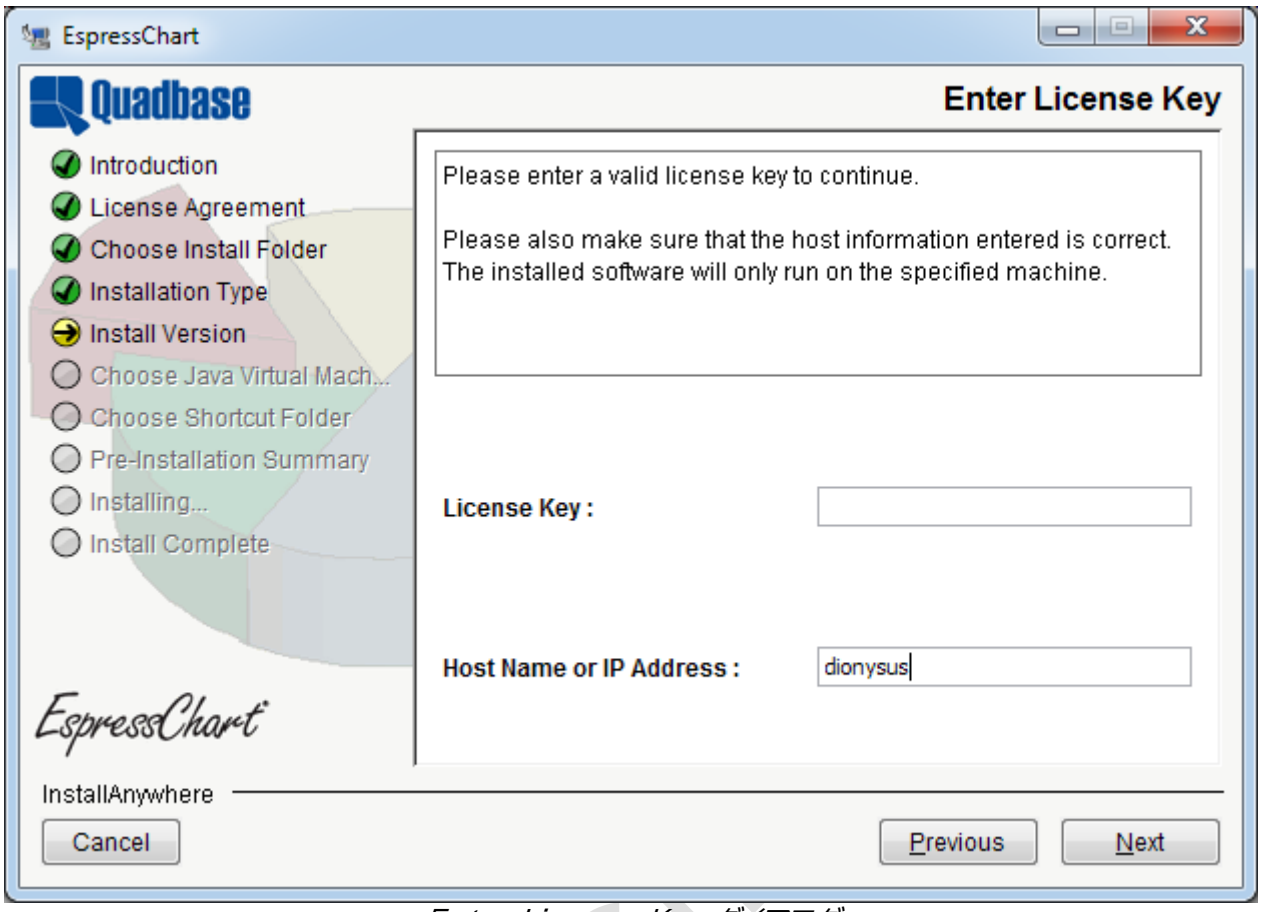
Choose Location ダイアログ

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尋ねられます。



Install Version ダイアログ

ライセンス版のインストールを選択すると、ライセンスキーを入力し、EspressChart 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。

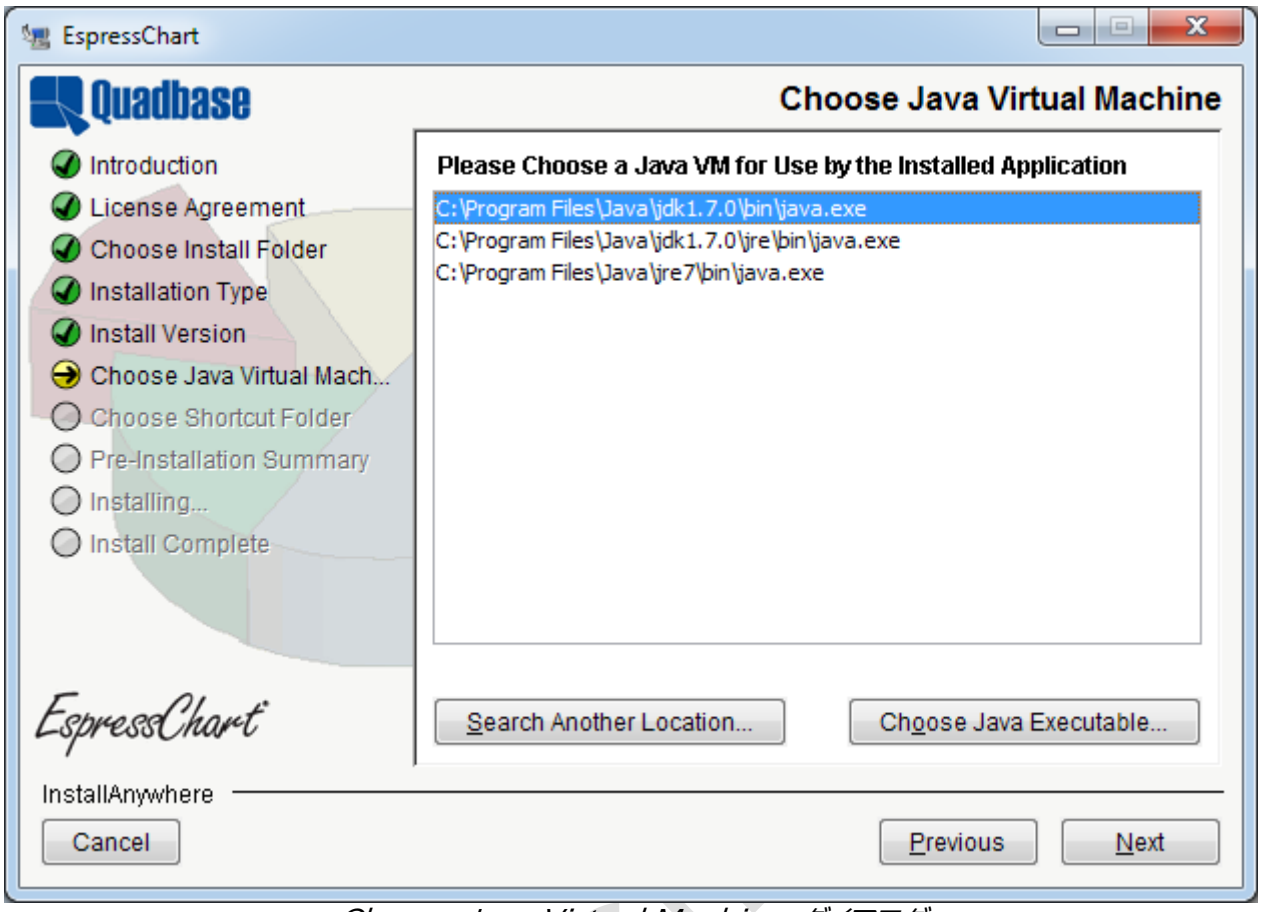


Enter License Key ダイアログ

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、EspressChart の正式版をインストールできません。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後に、http://www.quadbase.com/register/にてオンラインでライセンスキーを登録(登録手順)するか、こちらまでお問い合わせください。

インストールが完了すると、正式バージョンはこのダイアログで指定されたホスト名でのみ実行可能であり ため、入力したホスト名が正しいことを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、EspressChart の実行に使用する Java 仮想マシンのバージョンを指定できます。(Java プログラムなので、実行するには JVM が必要です。)



Choose Java Virtual Machine ダイアログ

このダイアログでは、使用する Java 仮想マシン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、**Select 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。EspressChart API を使用して Java コードを開発して実行するには、Java Development Kit および 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のみ表示されます。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 EspressChart というプログラムグループに追加されます。

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。

4.2 構成

EspressChart 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、環境に応じて下記の構成設定を変更する必要があります。

- EspressManager が使用するポート番号を変更する
- Web ルートを変更する
- Ctrl + J または Ctrl + P を使用してアプレットからチャートを印刷する
- Chart Designer ユーザの追加/削除/変更

これらの設定は 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:
<EspressChartInstallDir> /userdb/config.txt
設定ファイルの例を以下に示します。

```
[port]
22071

[webroot]
C:¥Inetpub¥wwwroot

[users]
guest
Name1 Password1
Name2 Password2
```

[port]セクションで、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。 EspressManager が IP アドレスをマシンに照会するのではなく、明示的な IP アドレスを指定することもできます。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]**セクションで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できます。

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

検索順序は、IP アドレス、domain_n、domain1 です。

[webroot] セクションには、Web サーバの Web ルートへのパスが含まれています。EspressManager は、Chart Viewer からグラフをエクスポートするときに使用します(Ctrl + J または Ctrl + P を使用して印刷する場合)。このオプションの詳細については、チャートビュアーを参照してください。

[user]セクションの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。

4.2.1 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。

4.3 EspressManager の起動

Chart Designer を起動する前に、EspressManager が起動し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 Chart Designer 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 EspressChart を使用することが可能です。割り当てられた IP 番号は 127.0.0.1 になります。これにより、ログファイル **<EspressChartInstallDir> /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 Chart Designer ユーザマシンで 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、EspressChart ルートディレクトリの **EspressManager.bat** または **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で自動的に起動します。

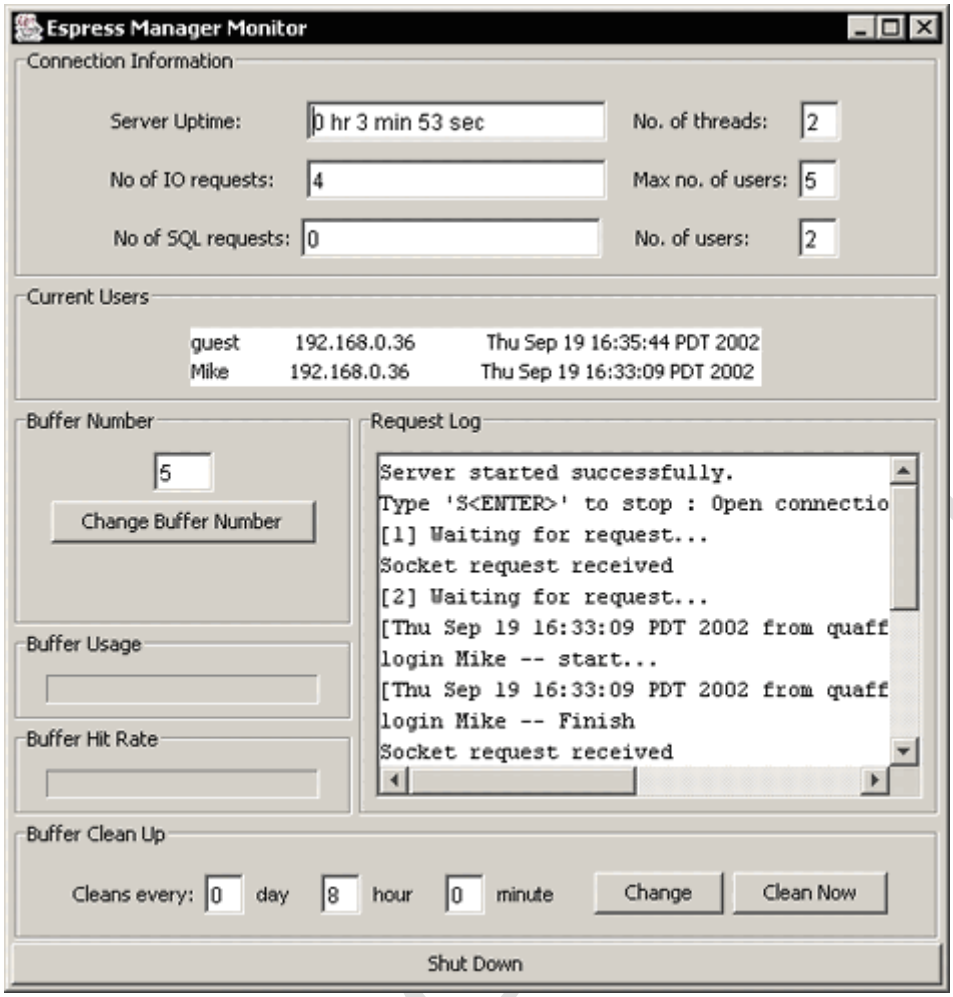
また、提供されているいくつかの引数を使用して独自の設定で EspressManager を設定することもできます。引数は"-"で始まり、その後にコマンドが続きます。各引数にはスペースは必要ありませんが、異なる引数を区切るには少なくとも 1 つのスペースが必要です。

これらの引数は、EspressManager の起動時にコマンドラインで入力することができます。また、**espressmanager.bat** または **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。

-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** または **-?**コマンドも同じくオンラインヘルプ情報を取得するために使用することができます。

-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/サーバー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 Monitor 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。



EspressManager Monitor

-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、シャットダウンする方法はプロセスを終了のみになります。

-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、チャートクエリ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 timeout 引数を過ぎると、EspressChart 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。

-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。つまり、省略された数字が高い数字であるという前提で値を変換します。
例えば:
-DBCleanAll:101010 は、10 日、10 時間、および 10 分ごとにバッファをクリーンにすることを意味します。
-DBCleanAll:1010 は、10 時間 10 分ごとにバッファをクリーンにすることを意味します。
-DBCleanAll:10 は 10 分ごとにバッファをクリーンに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、Chart Designer が API 経由で起動されたときに Chart Designer にセキュリティを適用するために使用されます。通常、Chart Designer が API 経由で呼び出されると、ユーザ認証はありません。これにより、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ます。これを防ぐために、ユーザはこの引数をオンにして(値は true / false)、Chart Designer が API から呼び出されたときに認証を強制します。この引数をオンにすると、**QbChartDesigner** 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)が提供されなければなりません。API から Chart Designer を呼び出す方法の詳細については、Chart API からの Chart Designer の呼び出し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、**QbChartDesigner** 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 Chart Designer を呼び出す方法の詳細については、Chart API からの Chart Designer の呼び出しを参照してください。

-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。グラフの表示/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。この機能により、チャートの実行中にシステムがメモリ不足になるのを防ぎます。

-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、完成したグラフ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、バッファのサイズが大きいほど、グラフが速く生成されます。

-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、個別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。

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。

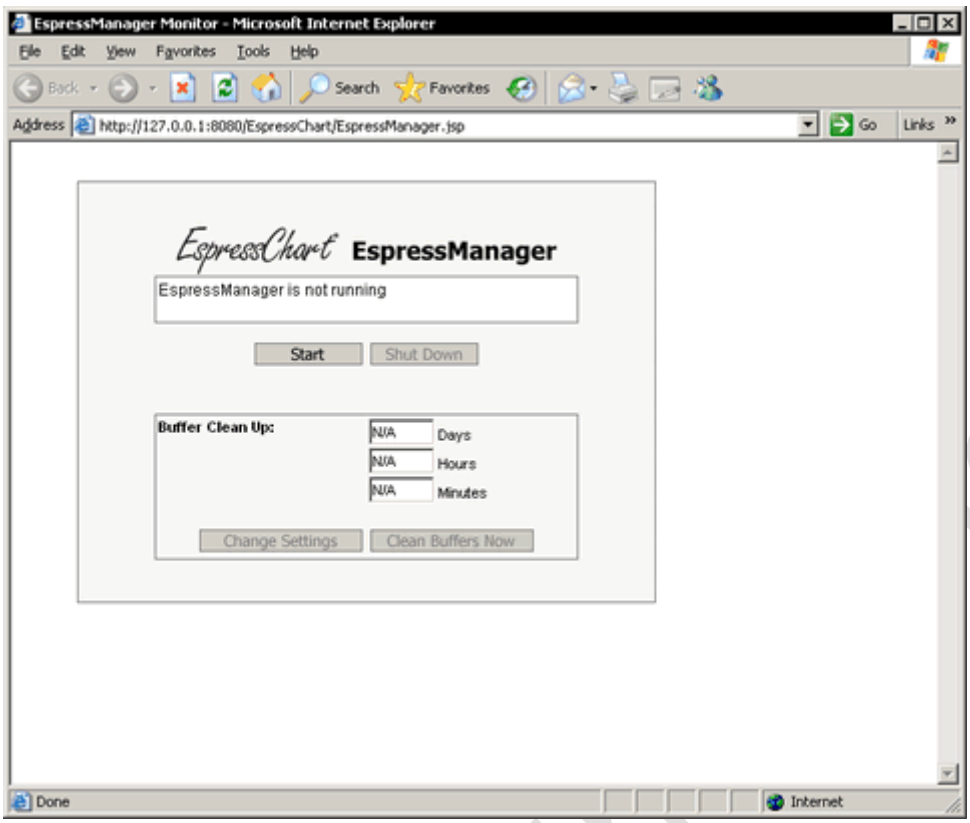
4.3.1 サーブレットとしての EspressManager の起動

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/サーブレットコンテナ（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**、および/**WebComponent**/ディレクトリをコピーします。
2. **EspressChart/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 EspressChart の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. EspressChart インストールディレクトリの/**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 には既に XML パーサがあります）。また、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るかもしれません。
5. アプリケーションサーバを再起動します。

これらのすべての手順が完了したら、ステップ 1 で説明したように、Web ブラウザのアプリケーションサーバで使用できる **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 Monitor

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、Chart Designer、スケジューラ、チャートビュアー、ページビュアー、およびチャート API を使用するアプリケーション/サーブレット/ JSP が含まれます。EspressChart 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。

4.3.1.1 Tomcat 4.x/5.x:サーブレットとしての EspressManager の配備

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。

以下の手順では、Tomcat のインストールディレクトリを **TomcatInstallDir**、EspressChart のインストールディレクトリを **EspressChartInstallDir** と表記します。

1. Tomcat のルートフォルダ（**<TomcatInstallDir>/webapps/ROOT**）の下に **EspressChart** という名前のディレクトリを作成します。**<EspressChartInstallDir>**から、**EspressManager.jsp**、**MenuError.jsp**、および **/Web_Component/** ディレクトリを、Tomcat ルートフォルダの下に新たに作成した**/EspressChart/**ディレクトリにコピーします。
2. **<EspressChartInstallDir>/classes** ディレクトリの内容を **<TomcatInstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリにコピーします。実行者が**<TomcatInstallDir>/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(これは Tomcat ではデフォルトでコメントアウトされており、**/servlet/context** を使用できません)。
3. EspressChart ディレクトリから QuadbaseDirectory.cfg ファイルを **<TomcatInstallDir>/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)。Web ブラウザを開き、Tomcat を実行している場合は **http://127.0.0.1:8080/servlet/whatIsMyWorkingDirectory** と入力します。サーブレットには作業ディレクトリのパスが表示されます)。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. **<EspressChartInstallDir>/lib** の下にある **EspressManager.jar** と **qblicense.jar** と **servlet-api.jar**(**<TomcatInstallDir>/common/lib** の下)を Tomcat の CLASSPATH に追加します。一般に、これは**<TomcatInstallDir>/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。

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/EspressChart/EspressManager.jsp
モニタがロードされ、**Start** ボタンをクリックして EspressManager を起動することができます。

4.3.1.2 Tomcat 8.x/9.x:サーブレットとしての EspressManager の配備

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。

以下の手順では、Tomcat のインストールディレクトリを **TomcatInstallDir**、EspressChart のインストールディレクトリを **EspressChartInstallDir** と表記します。

1. Tomcat のルートフォルダ(**<TomcatInstallDir>/webapps/ROOT**)の下に **EspressChart** という名前のディレクトリを作成します。EspressChart インストールディレクトリ(**<EspressChartInstallDir>**)から、**EspressManager.jsp**、**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 **/EspressChart/**ディレクトリにコピーします。
2. **<EspressChartInstallDir>/classes** ディレクトリの内容を **<TomcatInstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリにコピーします。
3. **<EspressChartInstallDir>/lib** の下にある **EspressManager.jar** と **qblicense.jar** と **servlet-api.jar**(**<TomcatInstallDir>/lib** の下)を Tomcat の CLASSPATH に追加します。一般的には、これは**<TomcatInstallDir>/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** の値を書き出します。

   setenv.bat のサンプル:

```
set
CLASSPATH=C:¥EspressReport70¥lib¥EspressManager.jar;C:¥EspressReport
70¥lib¥qblicense.jar;C:¥tomcat9¥lib¥servlet-api.jar
```

4. **<TomcatInstallDir>/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. **<TomcatInstallDir>/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。**<EspressChartInstallDir>/userdb** ディレクトリの **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/EspressChart/EspressManager.jsp

モニタがロードされ、**Start** ボタンをクリックして EspressManager を起動することができます。

4.4 Chart Designer の起動

Chart Designer は、スタンドアロンアプリケーションとして、または Web ブラウザを介してロードされるアプレットとして、2 つの方法で実行できます。どちらの場合でも、Chart Designer を起動するには、EspressManager を実行している必要があります。EspressManager を起動する方法については、前のセクションを参照してください。

アプリケーションとして実行
Chart Designer をアプリケーションとして起動するには、EspressChart インストールのルートディレクトリにある **Designer.bat** または **Designer.sh** ファイルを実行します。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のプログラムショートカットを使用できます。ユーザ名とパスワードの入力を求めるダイアログが表示されます。**config.txt** ファイルでユーザを設定した場合は、ユーザ名とパスワードを入力します。ユーザを設定していない場合は、パスワードなしでユーザ名 **guest** を入力します。**Start EspressChart** ボタンをクリックすると、アプリケーションが起動します。

Chart Designer ログインウィンドウ

ブラウザを使用して実行する
Chart 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てから、ブラウザに EspressChart の URL（**http://machinename/espresschart/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** ファイルが変更されてユーザが設定されている場合は、ユーザ名とパスワードを入力してログインします。**config.txt** ファイルが変更されていない場合は、パスワードなしのユーザ名 **guest** を入力し、**Start EspressChart** ボタンをクリックします。Chart Designer が起動します。

EspressChart は JDK 1.2 以上の JVM を必要とするため、ブラウザ経由で Chart 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。

4.4.1 サーブレットとして実行する EspressManager への接続

サーブレットとしての EspressManager の起動で説明したように、EspressManager がサーブレットとして実行されている場合、Chart Designer 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。

Stand-Alone
.bat または**.sh** ファイルを使用して Chart Designer を実行している場合は、そのファイルを変更して、EspressChart がサーブレットとして実行されていることを示し、EspressManager サーブレットの場所を指すようにする必要があります。これを行うには、次の引数を **.bat** または **.sh file -servlet:http://IP Address:Port/Context** 追加します。

例:
-servlet:http://127.0.0.1:8080/servlet

EspressManager が localhost 上の/**servlet**/コンテキスト、ポート 8080 で実行されていることを示します。

Browser
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 アプリケーションで Chart Designer を実行している場合は、次のものが必要です。
/lib/ ディレクトリ、**index.html**、**ChartDesigner.jsp** ファイル（3 つとも **<EspressChartInstallDir>** にあります）から **<TomcatInstallDir>/webapps/ROOT/EspressChart** ディレクトリにコピーします。

ChartDesigner.jsp で jnlp の内容を変更します。
href = ””のある **jnlp** 属性

```
<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 番目は、アプリケーションサーバが使用している URL（JSP エクスプレッションに保存：**<%codebase%>**）で、3 番目は 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。

<EspressChartInstallDir> の下の **/browserimages/** ディレクトリを **<TomcatInstallDir>/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**（**<TomcatInstallDir>/webapps/ROOT/WEB-INF** ディレクトリ下）で Chart Designer が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>
    <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>
  <url-pattern>/servlet/EspressManagerServlet</url-pattern>
</servlet-mapping>
```

Tomcat サーバを再起動します。

Chart Designer にアクセスするには、Web ブラウザを開き、次のアドレスにアクセスしてください：
http://127.0.0.1:8080/EspressChart/index.html
EspressChart.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 Chart Designer（Chrome / Microsoft Edge）を開きます。

4.5 後方互換性パッチ

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。後方互換性は可能な限り保持されますが、新しい動作が好まれることがあります。新しい動作はほとんどのユーザにとっては優れているはずですが、古いバージョンのグラフが既にある場合は、以前の動作と同じように古い動作を維持することができます。そのため、後方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。パッチが必要な場合、何をしているのかを知っている場合にのみ、それらを適用してください。よくわからない場合は、サポートに連絡してください。

パッチは<**EspressChartInstallDir**>/**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。

Chart Designer と Viewer にパッチを適用する場合は、**espressmanager.bat** と **designer.bat** ファイル（Windows を使用している場合）または **espressmanager.sh** ファイルと **designer.sh** ファイル（他の OS を使用している場合）を編集する必要があります。EspressChart のインストールディレクトリを開き、パッチパス JAR ファイル（例:/**lib/Patches/patch1.jar**）の相対パスを classpath パラメータに追加します。Chart Designer を HTML ページから実行している場合は、EspressChart 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 **applet** タグの **archive** 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 - 既定でグラフの軸のパディングをオフにする
- デフォルトの動作（パッチなし） - 軸パディングはデフォルトでオンになっています
- パッチ軸パディングの動作はデフォルトではオフです
- 新しい動作がバージョン 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する
- デフォルトの動作（パッチなし） - 注釈テキストに余白が残っていない
- パッチ - アノテーションテキストの動作に余裕がある
- 新しい動作がバージョン 5.0 で導入されました
- この機能はパブリック API や 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 の挿入→テキスト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
- パッチでの動作 - 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
- 新しい動作がバージョン 5.4 で導入されました
- この機能はパブリック API や 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。元の動作はバグです。**

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する（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る
- パッチ - 古い円グラフのラベル配置アルゴリズムの動作が使用されます
- 新しい動作がバージョン 6.0 で導入されました
- この機能はパブリック API や 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する
- デフォルト動作(パッチなし) - 整数は常に軸の自動スケールに使用されます
- パッチ軸の値のデータ型による動作は、軸の自動スケールに使用されます
- 新しい動作がバージョン 6.0 で導入されました
- この機能はパブリック API や 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する
- 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っている
- パッチの動作 - エラーチェックは無効になっています(データセットに設定されている値よりも最大値を低く設定することができます)
- 新しい動作がバージョン 6.2 で導入されました
- パッチは API 専用です
- この機能はパブリック API や UI で設定することはできません

patch7.jar - 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
- パッチの動作 - カテゴリ機能の単一色は、デフォルトではオフになっています
- 新しい動作がバージョン 6.3 で導入されました
- この機能は、API または Chart Designer の Chart Options ダイアログで **IDataPointSet.setSingleColorForCategories** メソッドを使用して設定することもできます

patch8.jar - 折れ線グラフの終わりから終わりへ単一点データを左軸に表示する

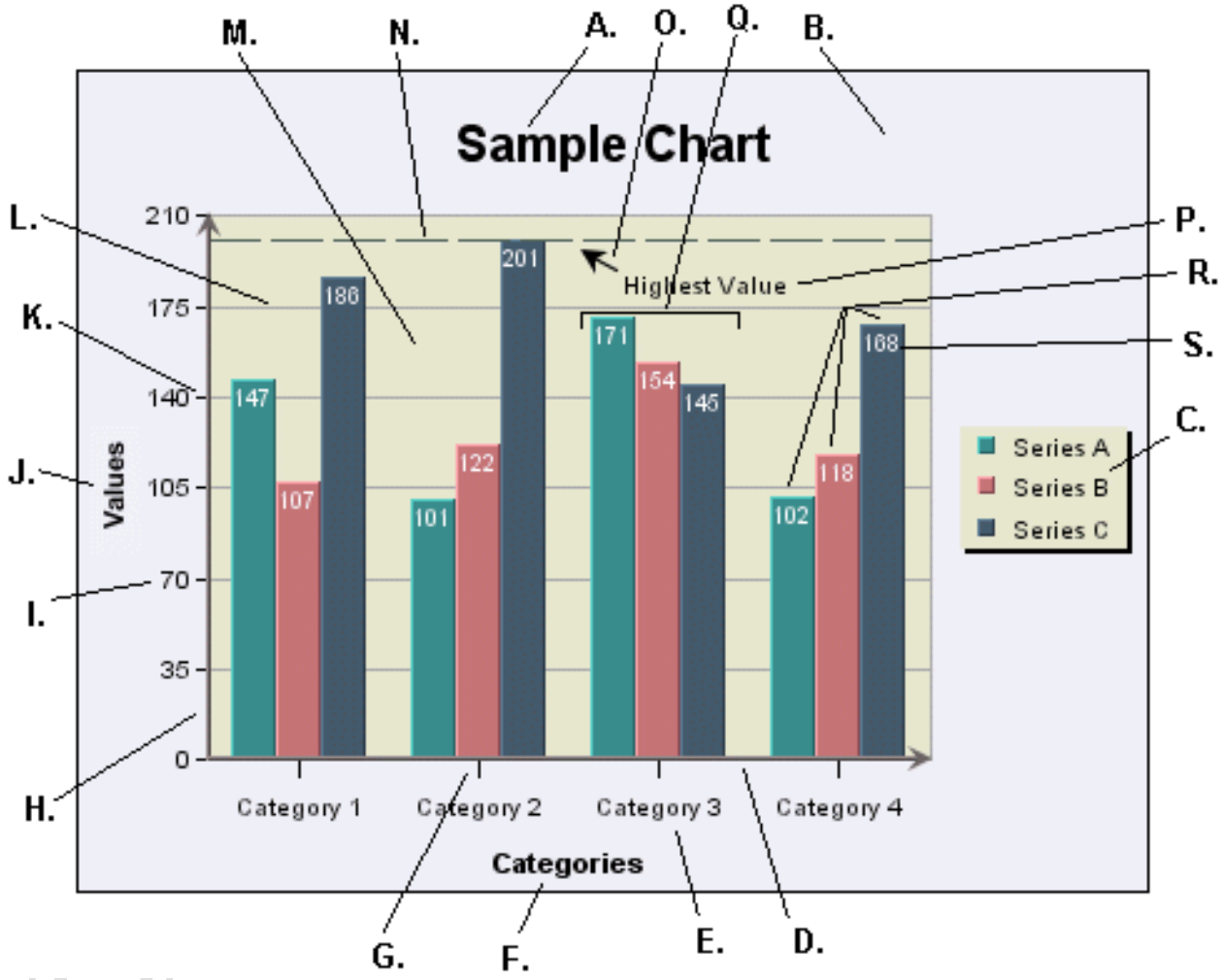
patch9.jar - 積み上げに十分なスペースがなくても積み上げラベルを表示する

5 Charting の基礎

この章では、チャートの各部分の名称から基本的なデータマッピング、保存/エクスポートのオプションまで、EspressChart の基本部分について説明します。この章で説明するすべての機能の詳細につきましては、後の章で詳しく説明します。

5.1 チャートについて

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解説しています。



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された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。

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、基本的な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。

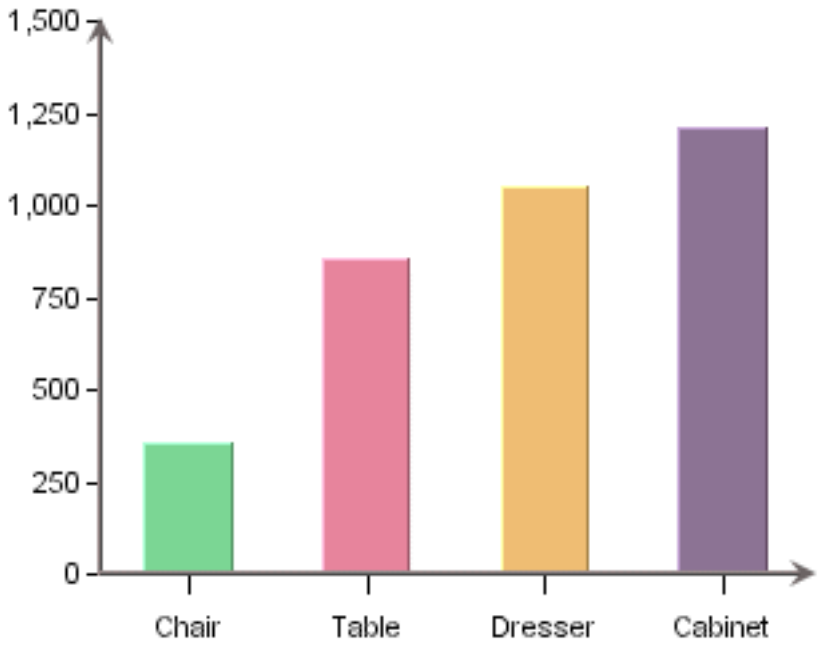
5.2 基本的なデータマッピング

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。

データソースの操作は、さまざまなソースからデータを取得する方法を説明しています。
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

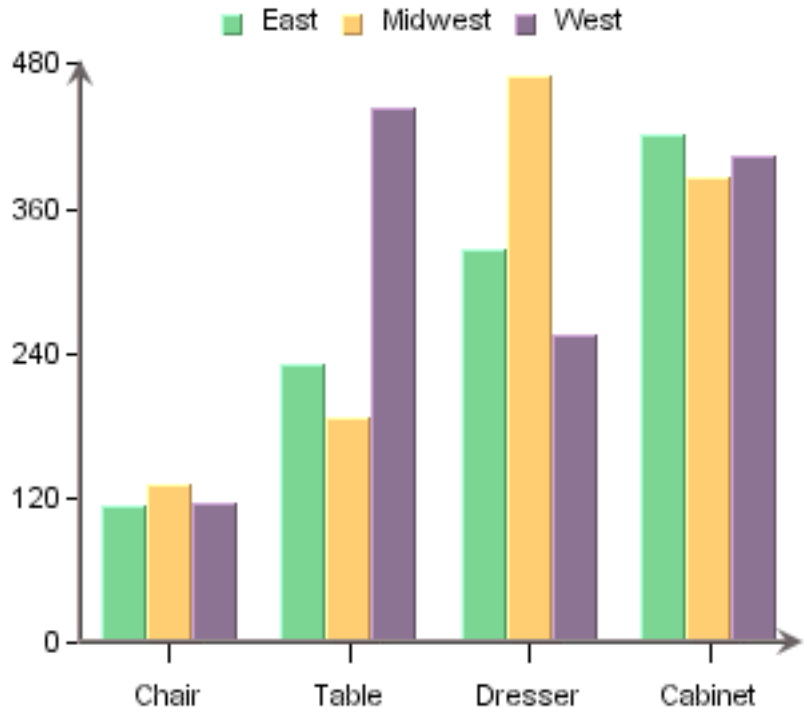
このデータをグラフにプロットするには、**Product** 列に各エントリの **Sales** 値をプロットします。したがって、**Product** はカテゴリであり、**Sales** は値です。チャートでは、**Product** 列を X(カテゴリ)軸に、Sales 列を Y(値)軸にマッピングします。結果のプロットは縦棒グラフで次のようになります。



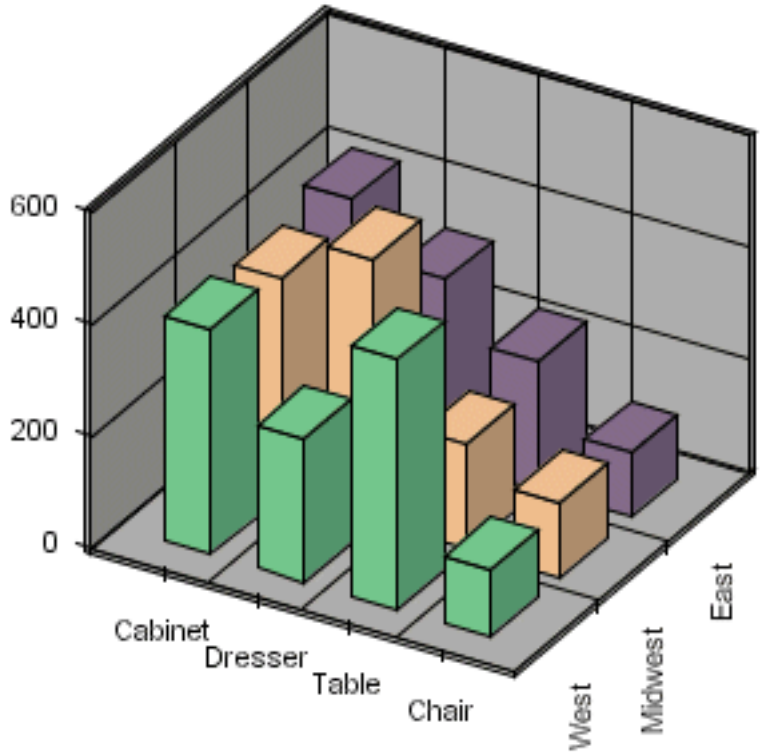
ここでは、カテゴリ列の各個別要素の値を示す列が描画されます。基本カテゴリ値の上に、追加情報をデータシリーズの形式で表示することができます。たとえば、データだけでなく Region 別の販売データも表示する要素が他にもあるとします。調整後の表は次のようになります。

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。



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、グラフタイプとデータマッピングを参照してください。

5.3 グラフの保存とエクスポート

チャート定義の保存とチャートのイメージファイルへのエクスポートには、いくつかのオプションがあります。チャートデザイナとチャート API の両方でグラフを保存してエクスポートする方法の詳細は、グラフの保存と書き出しと EspressChart Chart API を参照してください。

5.3.1 チャート定義の保存

EspressChart で作成したチャート定義をチャートやテンプレートファイルとして保存するには、主に 2 つの方法があります。

チャートファイル

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(型、寸法など)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。したがって、チャートファイルは移行可能です。グラフファイルを開くたびに、グラフを作成するために使用された元データが使用されます。チャートを開いた後、ソースからデータをリフレッシュするか、チャートのデータソース自体を変更することができます。

テンプレートファイル

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にはチャート定義のみが格納され、チャートには 10 レコードのデータしか格納されません。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。このため、テンプレートファイルはチャートファイルより移植性が低くなります。テンプレートファイルを使用して、グラフ属性をあるグラフから別のグラフに渡すこともできます。これを行うには、テンプレートをチャートに適用します。これにより、テンプレートの多くの属性が現在のグラフに引き継がれます。テンプレートの適用の詳細については、テンプレートの使用を参照してください。

チャート定義をバイナリ形式で保存するほか、チャートファイルやテンプレートファイルを XML 形式で保存することもできます。これらの XML チャート定義ファイルは、Chart Designer または Chart API により後から変更できます。

5.3.2 画像ファイルの生成

一般にチャートは、Web 経由でアプレットとして、またはイメージファイルを生成することによって、2 つの方法のいずれかで展開できます。アプレットは、テンプレートまたはチャートファイルのいずれかのチャート定義を直接ロードできます。アプレットの使用の詳細については、チャートビュアーのセクションを参照してください。画像ファイルの場合、EspressChart は次の形式でグラフをレンダリングできます。

GIF
EspressChart は、2 つの圧縮方法(RLE または LZW)のいずれかを使用して GIF 画像を生成できます。LZW メソッドは高速で、より小さなファイルを生成します。しかし、その使用は特許によって保護されています。LZW 圧縮を解除するには、Unisys からライセンスを取得する必要があります。デフォルトでは、RIF 圧縮を使用して GIF ファイルが生成されます。

JPEG
JPEG は他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、ファイルの品質と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。

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。この形式には 3 種類の圧縮オプションがあります。

SVG
SVG(スケーラブルベクターグラフィックス)は、XML ベースのテキスト形式でベクトルとして画像を保存する、比較的新しい画像フォーマットです。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、エクスポート後にグラフ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します。

BMP
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。

静的画像に加えて、チャートデータをテキストファイル、PDF、または XML ファイルとしてエクスポートすることもできます。

6 データソースの操作

グラフを設計するための最初のステップは、使用するデータを取得することです。Chart 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 からデータを描画したり、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。

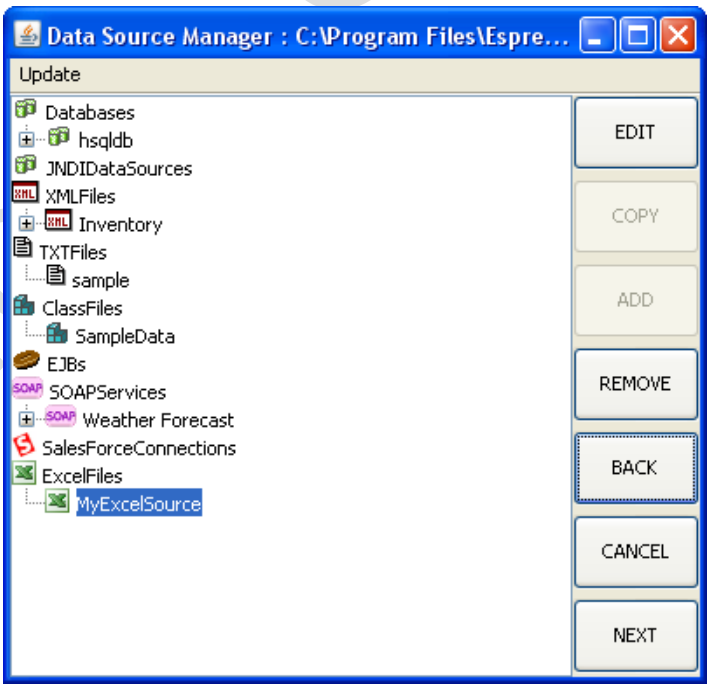
6.1 Data Source Manager

Data Source Manager 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。好きなだけ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チャートファイルに格納されているため、チャ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、チャ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。

6.1.1 Data Source Manager の使用

新しいグラフを開始（**File** ＞ **New** を選択するか、ツールバーの **New** をクリックして）すると、グラフウィザードが起動します。ウィザードは、チャ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。グラフ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。



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。グループ化されたデータベースには、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** の元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。**EJBs** の元にグループ化されたすべての EJB の元にグループ化されたすべての EJB 接続が指定され、**SOAP** の元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、Data Source Manager の機能を制御する各ボタンがあります。各ボタンは、以下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することや、XML クエリを変更することができます。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、クエリおよびデータビューでのみ使用できます。指定したクエリまたはデータビューのコピーを作成できます。

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。**TXTFiles** > **Add** と選択すると、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。

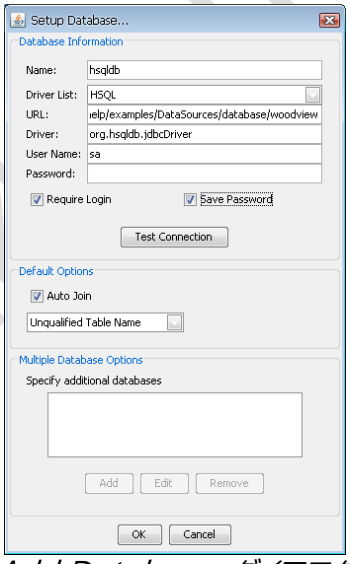
Cancel
ウィザードのプロセスがキャンセルされます。

Next:
これにより、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。

6.2 データベースからのデータ取得

EspressChart は、任意の 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、**EspressManager.bat** または **EspressManager.sh** ファイルを変更して、EspressManager がドライバのクラスを取得する必要があります。**.bat** または**.sh** ファイルの**-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 ドライバは含まれています。他のデータベース JDBC jar ファイルは、ライセンスや複数のドライバの存在などから含まれていませんが、jar ファイルを明示的に追加することでサポートされます。

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、**Database** ノード ＞ **Add** 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 **UserName** 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。



Add Database ダイアログ

EspressChart がデータベースへの接続を作成するには、以下の情報を提供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、以下のように分類されます。

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、またはデータベース接続メカニズムの名前、1 つまたは複数のドライバでサポートされている可能性があります。サブプロトコル名の顕著な例は、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、以下の URL を使用する必要があります。

```
jdbc:odbc:Northwind
```

この例では、サブプロトコルは **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>** - データベースを識別する方法。サブ名は、サブプロトコル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。サブネームの機能は、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
（次と同様、**jdbc:dbvendorname://machineName/SchemaName**）

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ために必要な JDBC ドライバです。インストール（または Oracle の J2SE）に含まれている JMV を使用している場合は、以下のドライバを指定し、ODBC データソースに接続します。

sun.jpdc.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。たとえば、Oracle データベースエンジンには **oracle.jdbc.driver.OracleDriver** が必要です。

User Name:
これは、データベースに使用されるユーザ名です。

Password:
上記のユーザのパスワードです。

接続情報を指定したら、**Test Connection** 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 **Default Options** 部分では、クエリビルダのインターフェイ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の結合を試みます。照会に使用される

テーブル名形式は、非修飾(テーブル名のみ)または 2 パートまたは 3 パート修飾のいずれかで指定できます。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(つまり、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 パート修飾のテーブル名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情報と同じドライバ(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。この問合せでは、3 パートでのテーブル名形式を使用して、追加のデータベースの列を参照してデータを取得できます。

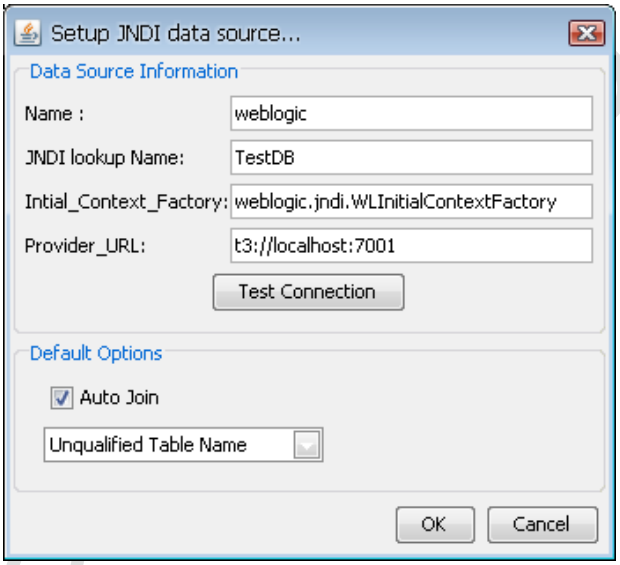
EspressChart のインストールには、2 つのサンプルデータベースが含まれています。1 つは HSQL(純粋な Java アプリケーションデータベース)データベースで、もう 1 つは MS Access データベースです。両方とも同じデータを含み、**help/examples/DataSources/database** ディレクトリにあります。これらのサンプルデータベースへの接続を設定する方法の詳細については、レジストリへのデータソースの設定を参照してください。

6.2.1 JNDI データベース

EspressChart 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。EspressChart では、JNDI データソースはデータベースの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、環境間でチャ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、チャートは変更なしで移行できます。

EspressChart 内の 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 EspressManager を実行する必要があります。サーブレットとしての EspressManager の実行の詳細については、サーブレットとしての 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、様々なベンダーが JNDI データソースを異なる方法で実装しているため、使用しているアプリケーションによって異なります。**Test Connection** をクリックすると、接続をテストできます。

6.2.2 クエリ

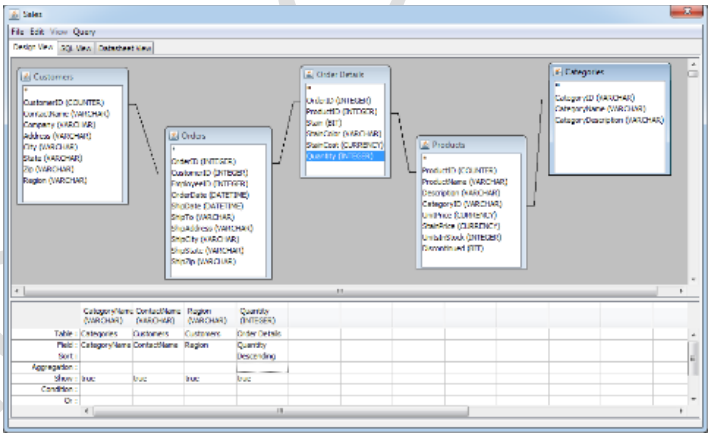
データベースを追加すると、データベースの新しいノードが Data Source Manager ウィンドウに表示されます。ノードを展開すると、さらに 2 つのノードが表示されます。1 つは**クエリ**（**Queries**）と呼ばれ、もう 1 つは**データビュー**（**Data Views**）です。これらは、データベースからデータを取得する 2 つの方法です。新しいクエリを作成するには、**Queries** ノードを選択して **Add** をクリックします。クエリ名を指定し、SQL 文を入力するクエリビルダを起動するかを選択するダイアログが表示されます。

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。

6.2.2.1 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、Data Source Manager 内に新しいクエリを追加し、**Open Query Builder** 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 Query Builder 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のかも示されます。メインウィンドウまたは QBE(列でクエリ)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、様々なメニューを切り替えることができます。**Design View** は、前述のメインデザイナウィンドウです。**SQL View** には、現在のクエリで生成された SQL 文が表示され、**Datasheet View** にはクエリ結果が表示されます。

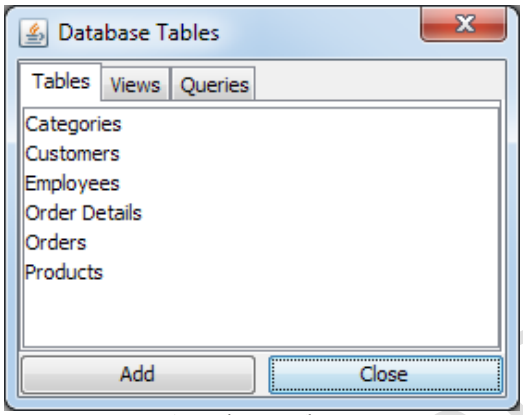
クエリの作成が完了したら、**File** の **Done** を選択して Data Source Manager に戻ります。

6.2.2.1.1 テーブル

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。**View** 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、**Queries** 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、**Queries** 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** 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 Query Builder 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。テーブルエイリアスを指定し、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** 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。それを再度開くには、**Queries** 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形式を使用してテーブルが表示されます。**Queries** メニューから **Table 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

6.2.2.1.2 結合

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。

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。

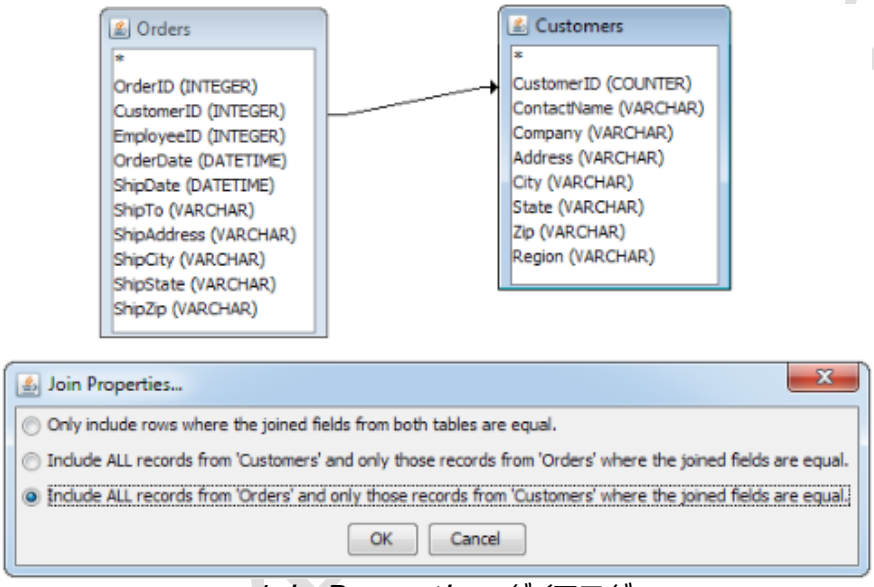
```
Select Orders.OrderID, Customers.CustomerName, Orders.Sales
From Customers, Orders
Where Customers.CustomerID = Orders.CustomerID
Order by Orders.OrderID;
```

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
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。

6.2.2.1.3 列

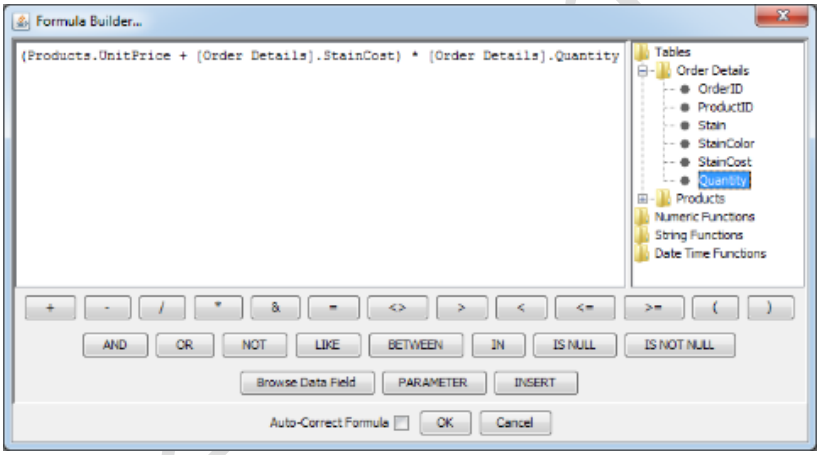
QBE ウィンドウには、照会のために選択された列フィールドに関する情報と選択の条件が含まれています。

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

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。

EspressChart 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、EspressChart 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、EspressChart 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。

6.2.2.1.4 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ーフェイスに追

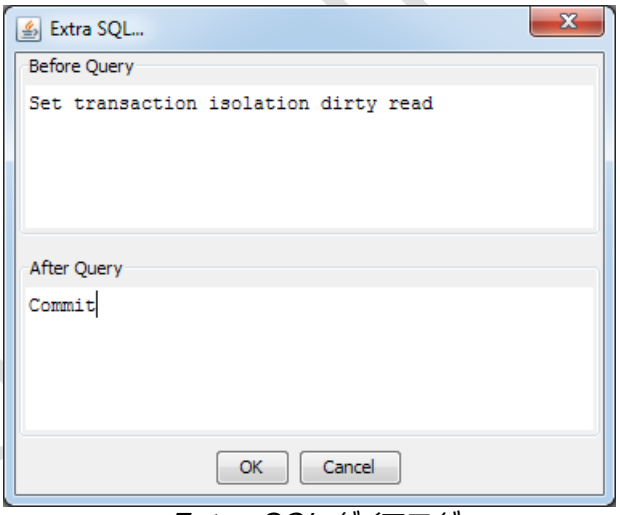
加することもできます。

EspressChart 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、EspressChart 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>
```

6.2.2.1.5 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。

6.2.2.1.6 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。

 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。

6.2.2.2 パラメータ化されたクエリ

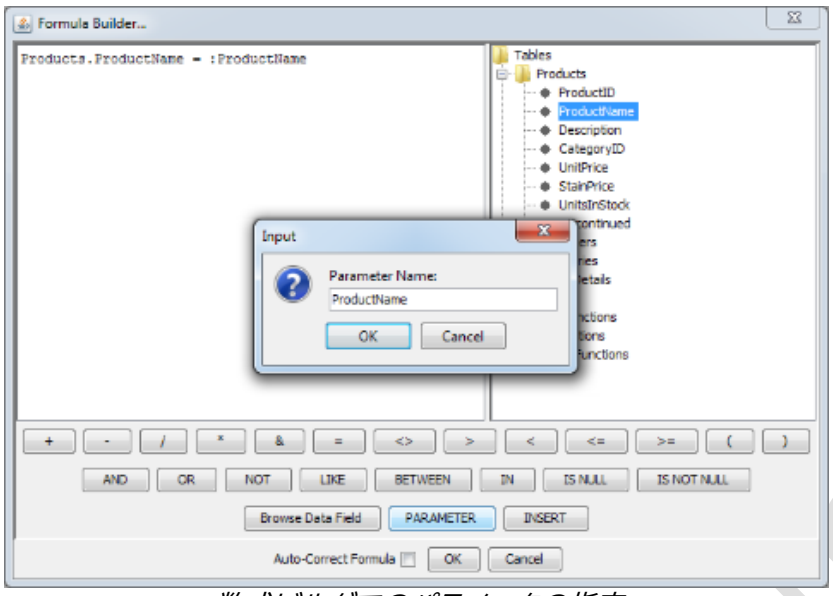
また、クエリビルダを使用してパラメータ化されたクエリを設計することもできます。この機能を使用すると、実行時にデータをフィルタリングできます。パラメータ化されたクエリは、チャートのパラメータドリルダウンにも使用されます。ドリルダウンチャートの詳細については、パラメータドリルダウンを参照してください。

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。



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。

6.2.2.2.1 複数値パラメータ

EspressChart 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。

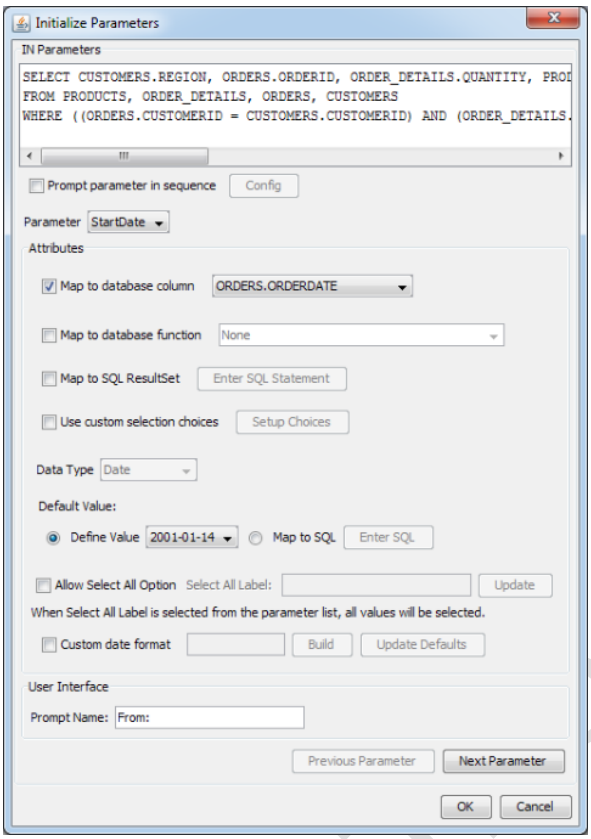
複数値のパラメータを作成するには、SQL 文の **IN** 句内にパラメータを配置します。たとえば、以下のクエリでは **State** という名前の複数値パラメータが作成されます。

```
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。

6.2.2.2.2 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。



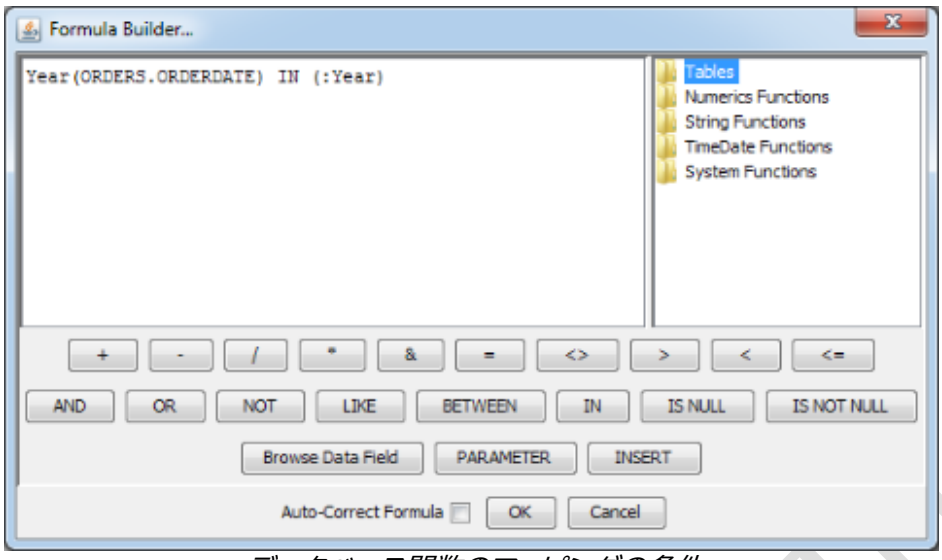
Initialize Parameters ダイアログ

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、チャートビュアーでチャ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。

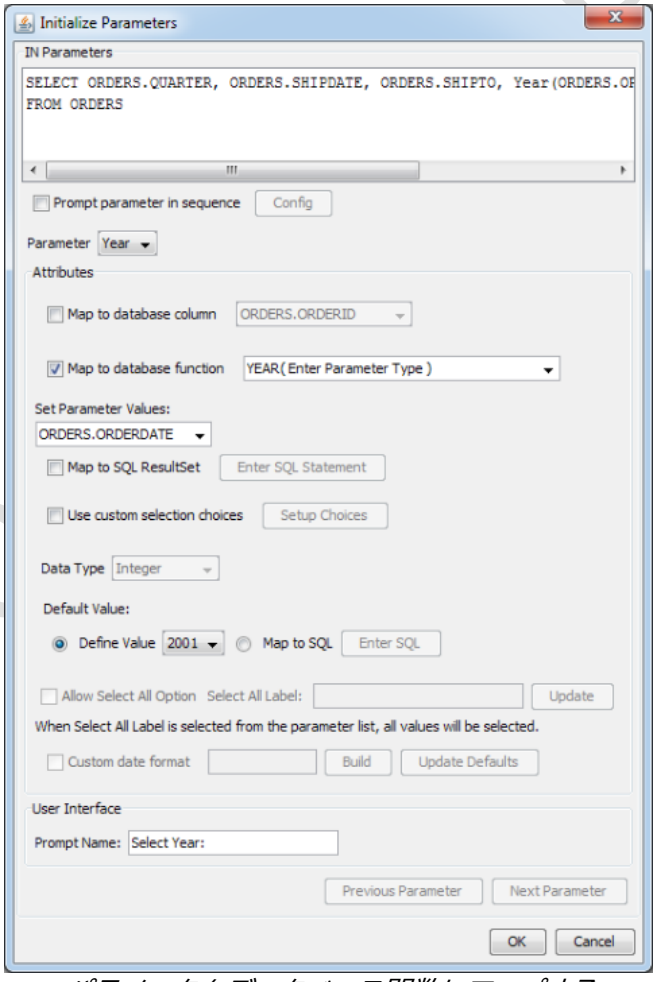
通常、このドロップダウンリストは、問合せから結合および条件を適用しながら、列に SELECT DISTINCT 文を実行することによって移入されます。制約なしでカラムからすべてのデータを取得したい場合(パラメータプロンプトのパフォーマンスを改善することがある)、EspressManager の起動時に **singleTableForDistinctParamValue** 引数を設定できます。EspressManager の設定オプションの詳細については、EspressManager の起動を参照してください。

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。



パラメータをデータベース関数にマップする

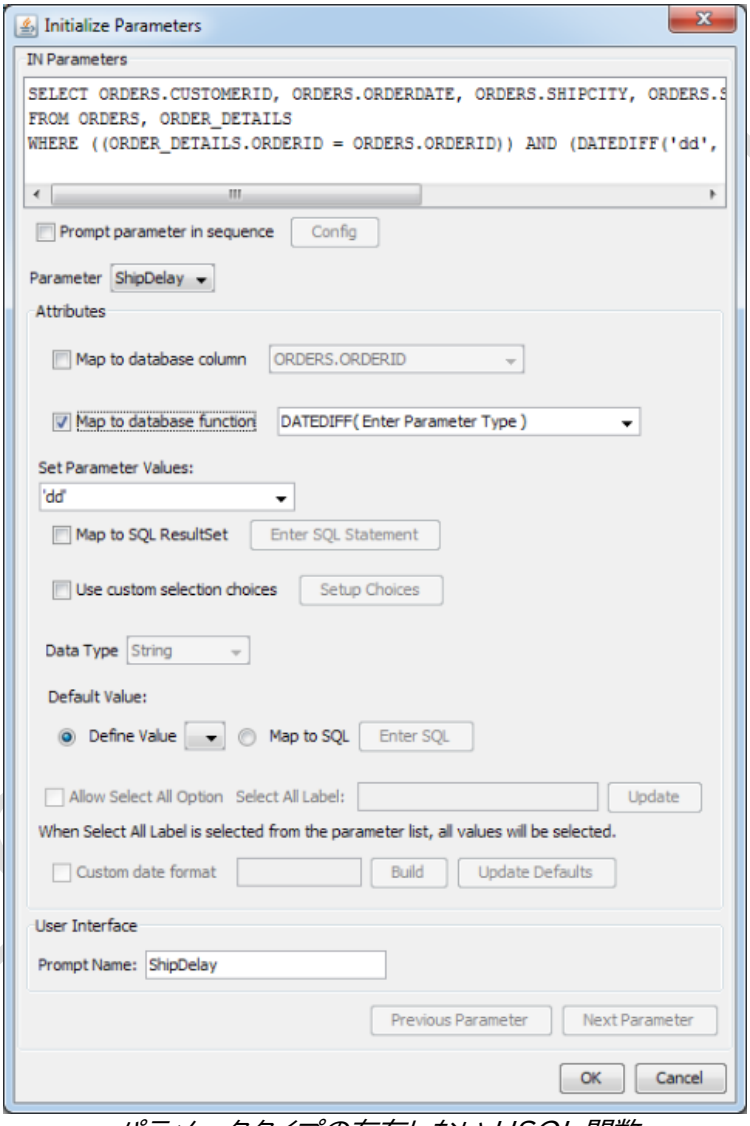
カスタム関数のリストは、/**userdb**/ディレクトリにある **DatabaseFunctions.xml** ファイルから抽出されます。新しいデータベースまたはカスタム関数を追加する場合は、.xml ファイルを変更します。新しい機能は、プログラムを再起動するとこのリストに表示されます。

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。

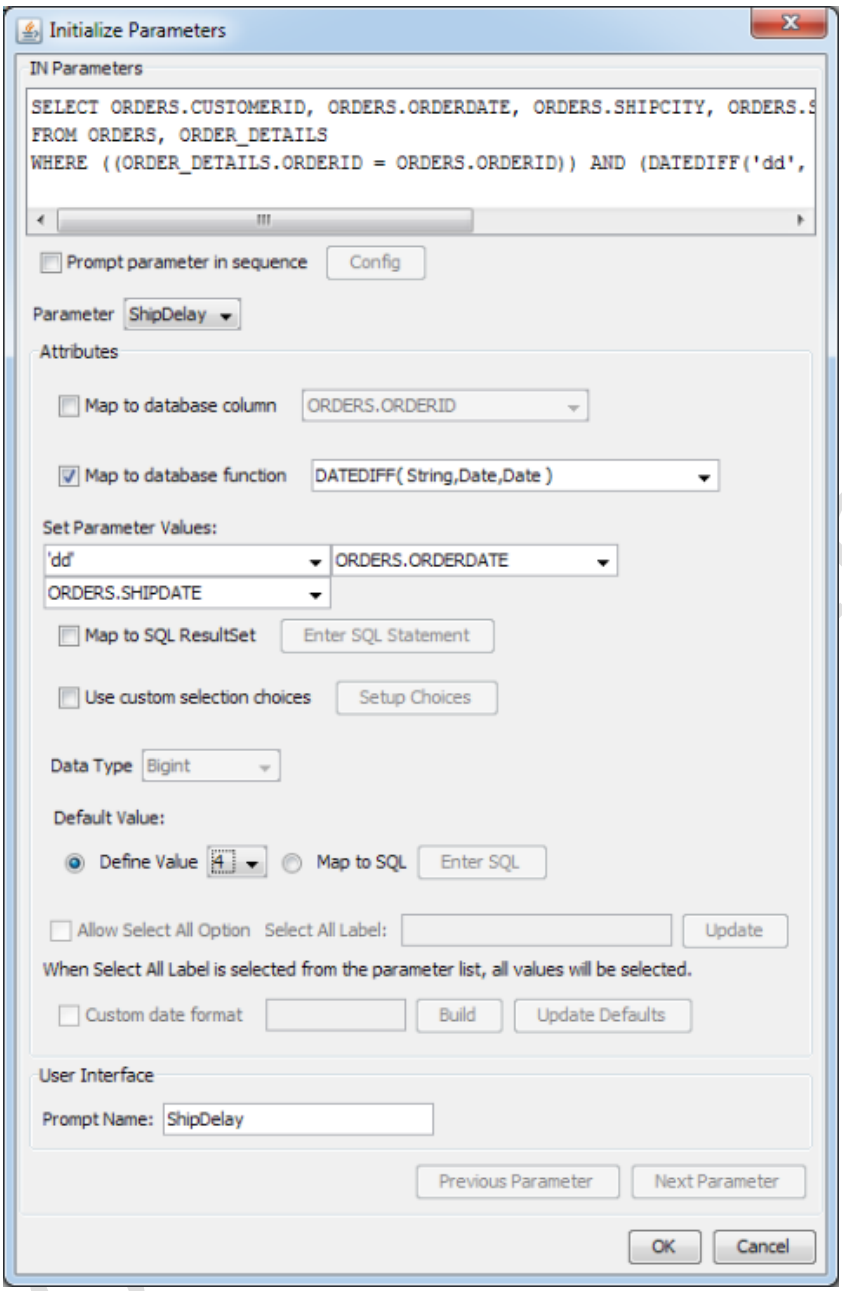
```
DATEDIFF( 'dd',ORDERS.ORDERDATE,ORDERS.SHIPDATE)> =:ShipDelay
```

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。



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。



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。

```
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)))
```

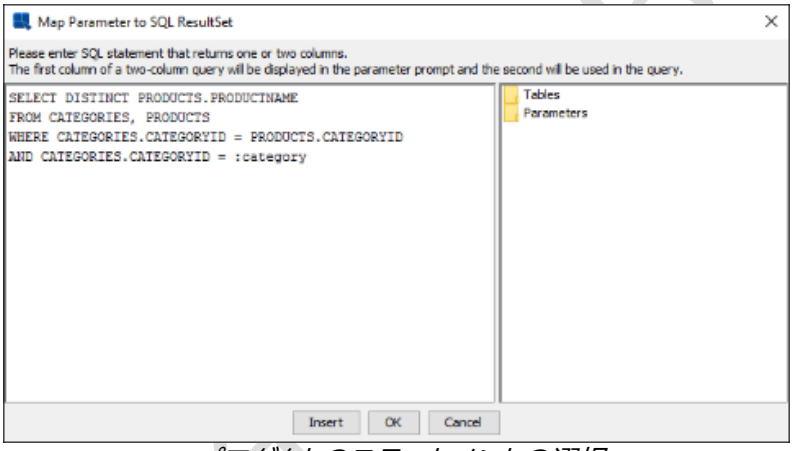
初期化パラメータの **config** プロンプトで、パラメータのプロンプトの順序を**カテゴリ（category）**に、次に**プロダクト（Product）**に設定します。

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
```

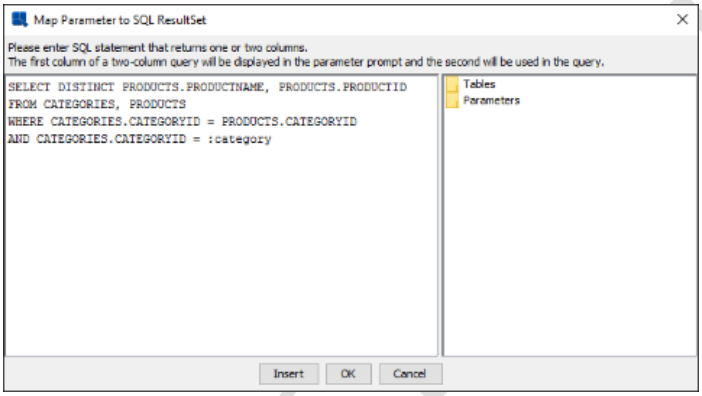


プロダクトのステートメントの選択

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。

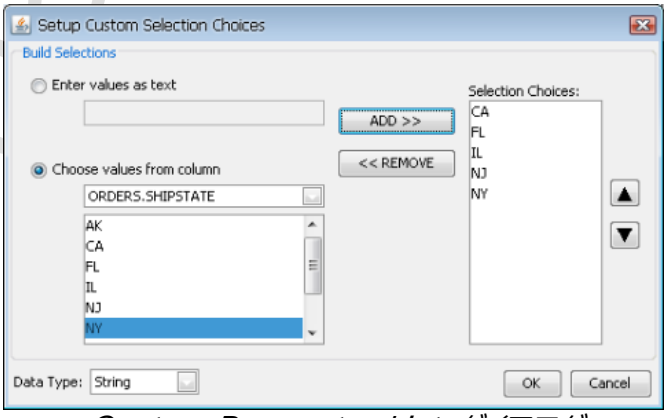
データベースにプロダクト ID が主キーとして含まれているテーブルがあるとします。エンドユーザがデータベースから製品を検索する場合、プロダクト ID は単なる暗黙のコードである可能性があるため、パラメータとして製品名を使用したい状況にあります（製品名が最初にクエリにリストされ、プロダクト ID が 2 番目にリストされます）。この機能を使用すると、ドロップダウンリストの値の製品名と実際の値フィルタ条件としての製品 ID を選択できます。



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

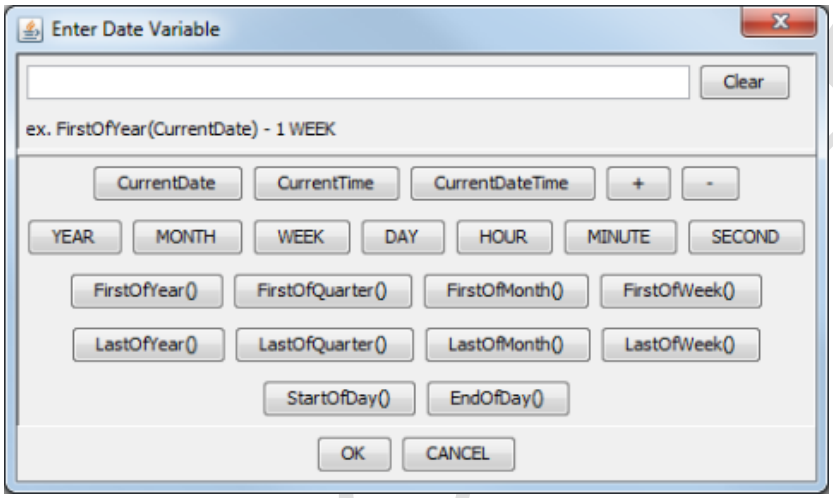
これにより、パラメータのデフォルト値を指定することができます。デフォルト値を指定する必要はありませんが、指定することをお勧めします。デフォルト値を指定しないと、データソースが存在しない場合でもグラフテンプレートを開いたり、操作したりすることはできません。

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(複数値のパラメータを参照)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、チャ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)**

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()
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。

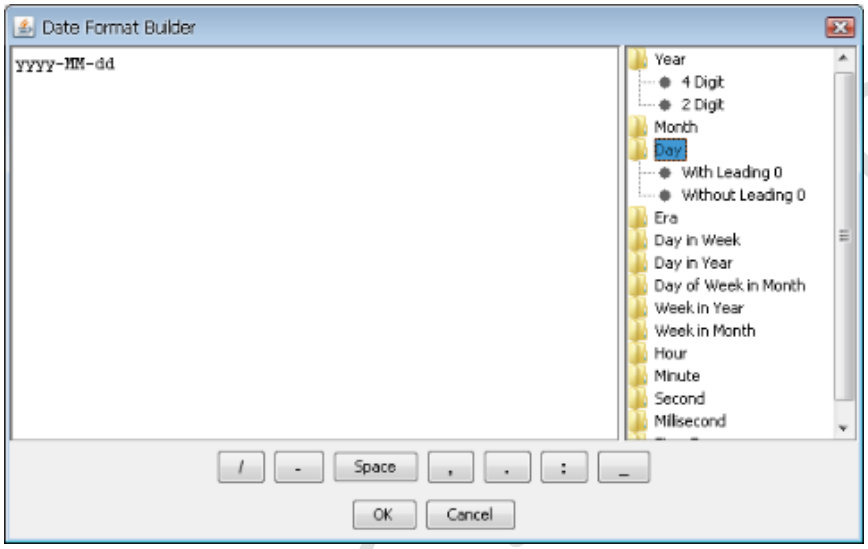
Data Type
これにより、パラメータ値のデータ型を指定することができます。パラメータを列にマップした場合、データ型は自動的に設定されます。

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。



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。

以下の表に、使用できる文字の組み合わせを示します。

文字	表示	出力(テキスト／数値)	例
G	era	text	AD
y	year	number	1996, 96
M	month in year	text or number (depends on length)	July, Jul, 07
d	day in month	number	10
h	hour am/pm (1-12)	number	1
H	hour 24 hr. (0-23)	number	18
m	minute in hour	number	30
s	second in minute	number	55
S	milisecond	number	978
E	day in week	text	Tuesday, Tue
D	day in year	number	189
F	day of week in month	number	2 (as in 2nd Wed. in July)
w	week in year	number	27
W	week in month	number	2
a	am/pm marker	text	AM, PM
k	hour 24 hr (1-24)	number	24

K	hour am/pm (0-11)	number	0
z	time zone	text	Pacific Standard Time, PST

これらの文字のほぼすべての組み合わせをまとめて、目的の形式で日付式を生成することができます。文字のグループの数は、要素が取る形式を決定します。テキスト要素の場合、グループ内の 4 つ以上の文字は、要素の完全な形式を使用します。4 文字未満の場合は、短い形式が使用されます(存在する場合)。たとえば、**EEEE** は月曜日を返し、**EE** は月を返します。文字 **M** または数字のいずれかで表示できる **M** の場合、グループ内の 4 つ以上は完全版を表し、3 は省略形を表示し、2 以下は数字書式を表示します。

数値要素の場合、文字数は要素が取る最小桁数です。短い数値は、先行 0 を実装します。たとえば、日付の日が 2 の場合、**dd** は 02 を返し、**d** は 2 を返します。

";"、 ":"、 "@"などの a-z や A-Z 以外の文字は、文字列式内のどこにでも挿入することができ、入力時に表示されます。単語と式を一重引用符で囲んで挿入することもできます(2 つの一重引用符を入力すると、アポストロフィをテキストとして挿入できます)。

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。

6.2.2.2.3 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。

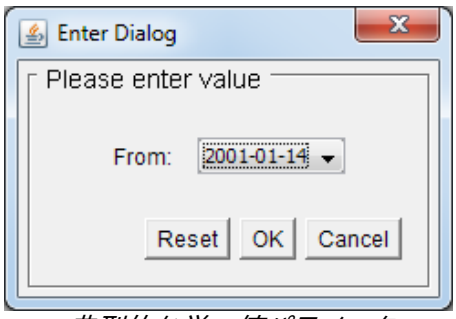
単一パラメータ
複数値選択ができないパラメータの場合でも、一度にすべてのパラメータ値を選択することができます。

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。

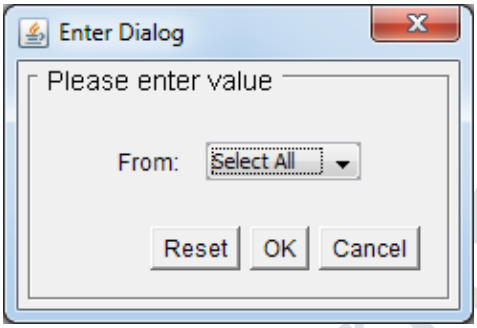
```
WHERE column = :Parameter
```

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。



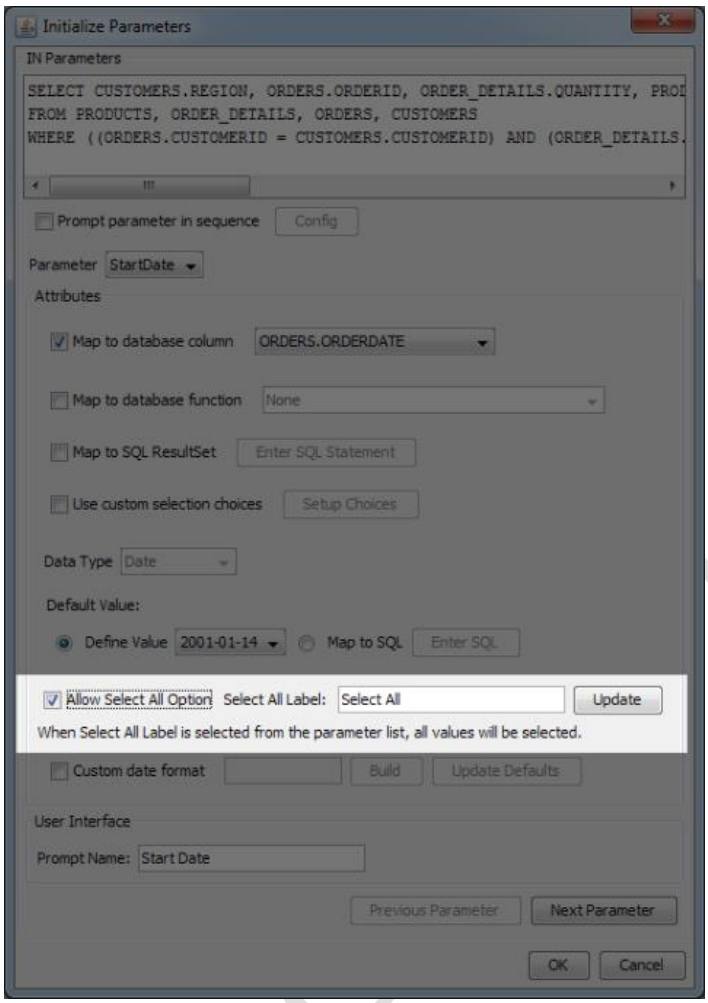
典型的な単一値パラメータ

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。



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。

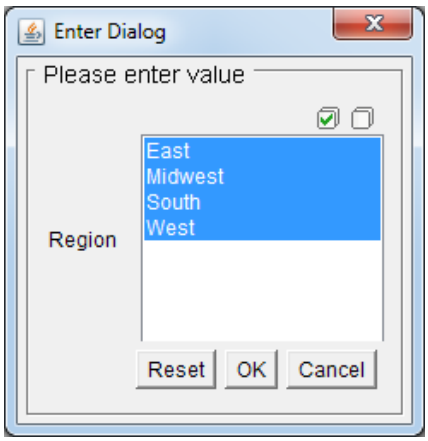
演算子	説明
<	未満
<=	以下
>	超過
>=	以上
=	等しい

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。

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ドレベルにない場合、選択肢たちがコンマで区切られた値のリストとして問い合わせに挿入されます。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
```

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）。

EspressChart 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

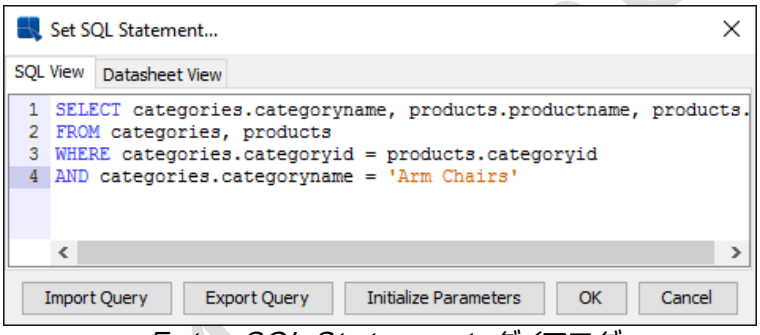
```
WHERE ((column = :Parameter) OR (:Parameter LIKE 'selectall'))
```

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. サポートにお問い合わせてください。

6.2.2.3 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

結果をプレビューするには、**Datasheet View** をクリックします。

6.2.2.3.1 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 は単一の問い合わせ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問い合わせを格納し、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。

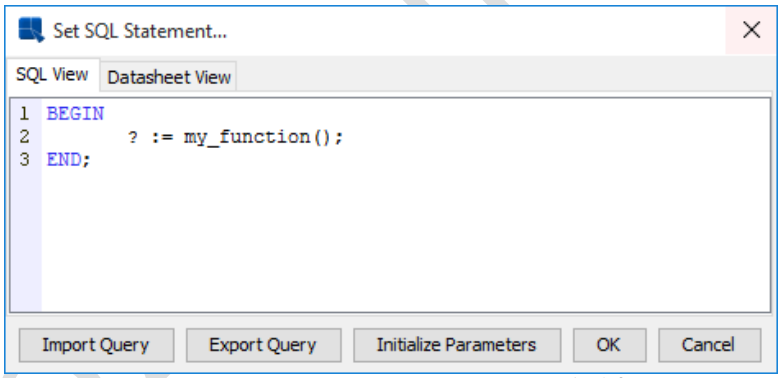
```
CREATE OR REPLACE PACKAGE types
AS

        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;

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 ChartDesgner 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、EspressChart 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。

EspressChart 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

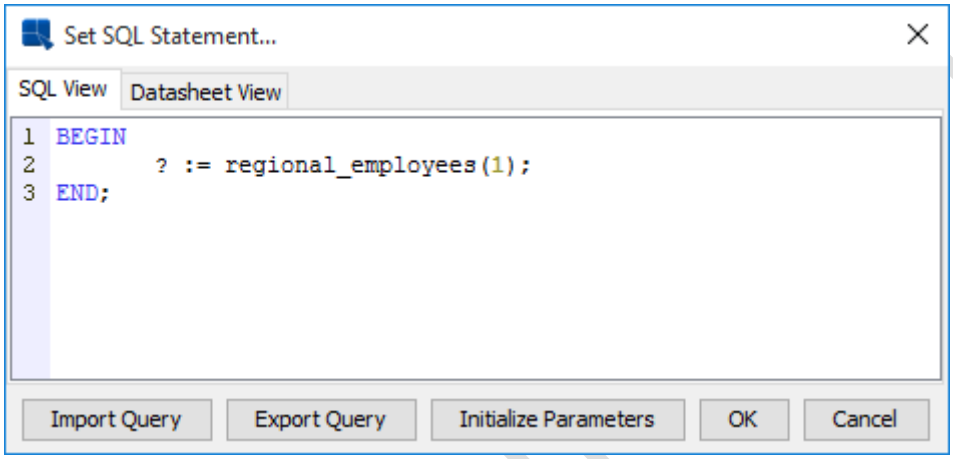
この表には、企業の様々な場所がツリー構造で一覧表示されています。従業員の数は、葉ノード(たとえば、ニューヨーク、ロンドンなど)に格納され、各ノードは、その直接の親に関する情報を含む。ある地域の従業員数と、その地域内の別々の枝に関する情報を表示するチャートを作成するとします。たとえば、ユーザが **ID = 2(アメリカ)**を入力した場合は、支店の所在地とともに米国内の従業員の総数をグラフに表示する必要があります。Oracle の **CONNECT BY** 句と **START WITH** 句を使用すると、2 つのシンプルな Oracle ストアドファンクションを使用して問題を解決できます。

```
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;

        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

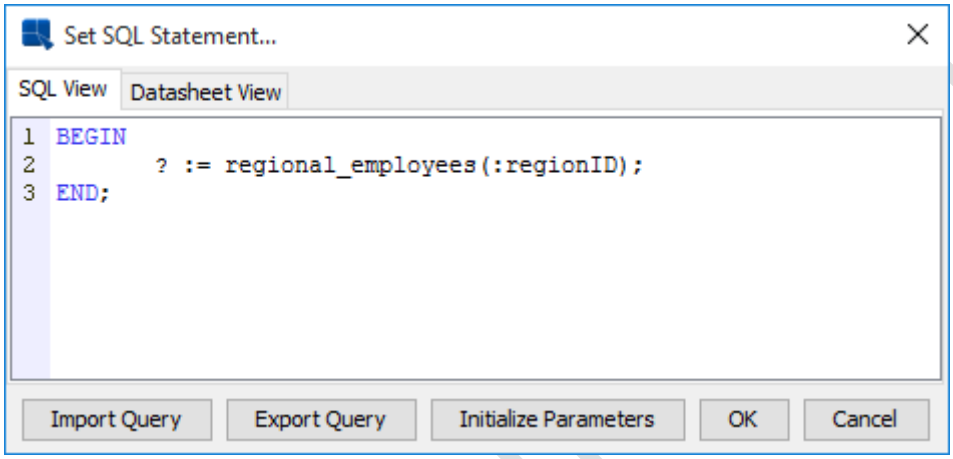
regional_employees の結果セット

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

パラメータ化されたグラフを作成するには、**:param_name** 構文を使用します。EspressChart 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

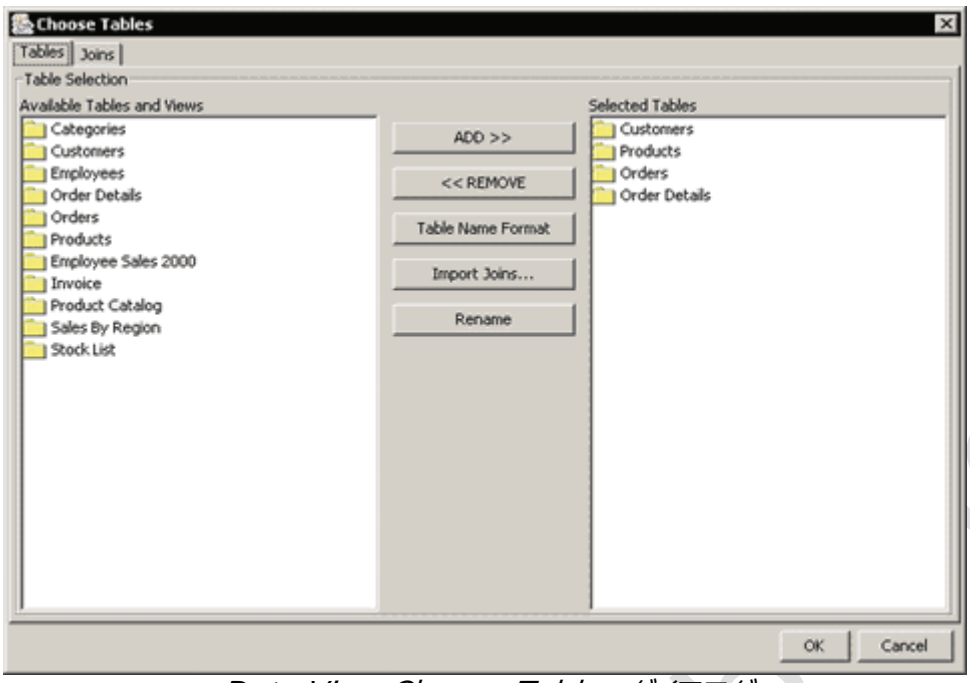
IN パラメータを使用する場合は、クエリを実行する前にパラメータを初期化する必要があります。パラメータを既存の列にマップするため、ストアドプロシージャを実行するための正しいデフォルトデータ型を設定することは特に重要です。パラメータの初期化の詳細については、クエリパラメータの初期化を参照してください。この例を試すために、**<EspressChartInstallDir>/help/examples/data/locationHierarchyExample.sql** には、**employee_table** と 2 つのストアドファンクションを作成する SQL コマンドが含まれています。

6.2.3 データビュー

EspressChart には、クエリインターフェイスに加えて、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。

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、グラフの設計を開始します。

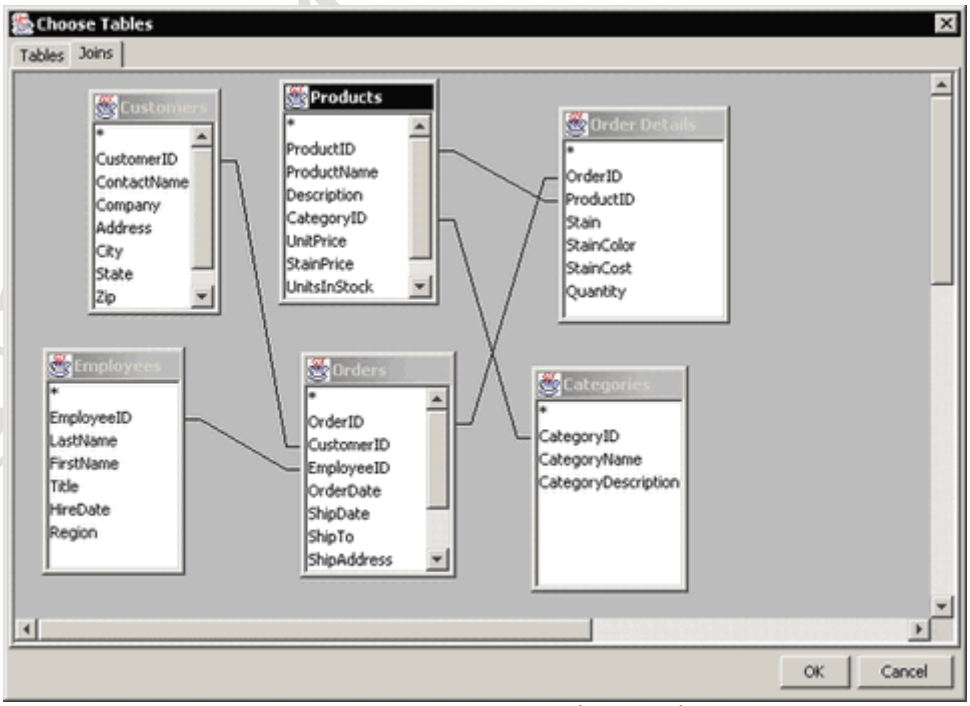
データビューを作成するには、Data Source Manager ウィンドウで **Data Views** ノードを選択し、**Add** ボタンをクリックします。新しいウィンドウが開き、データビューに使用するデータベーステーブルを選択できます。



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。

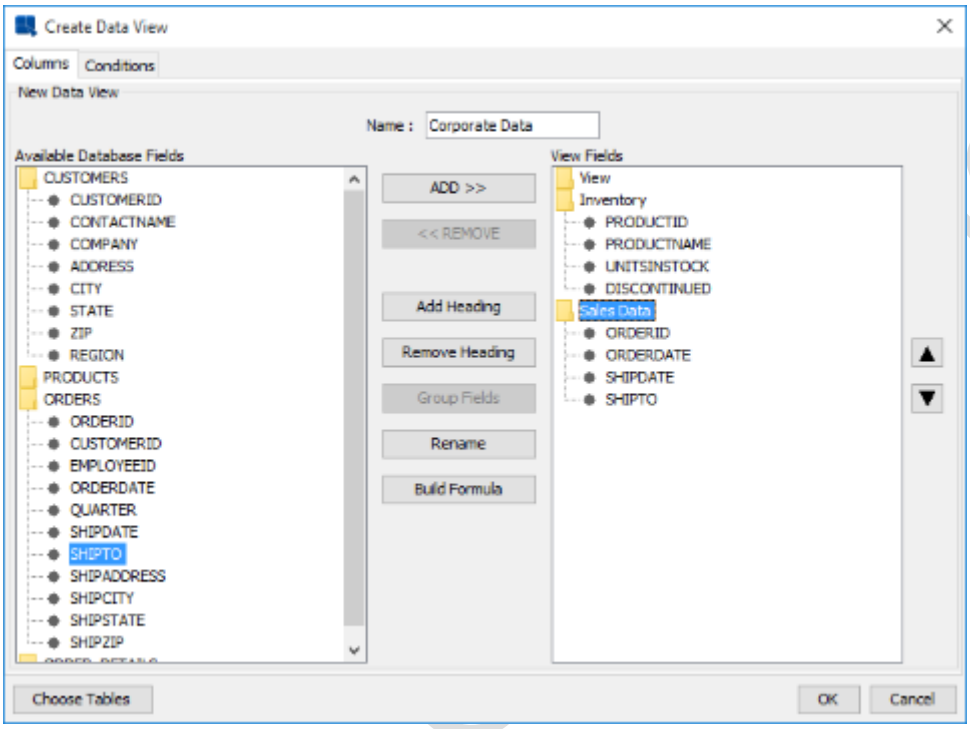


Data View Joins ダイアログ

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、結合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。



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。

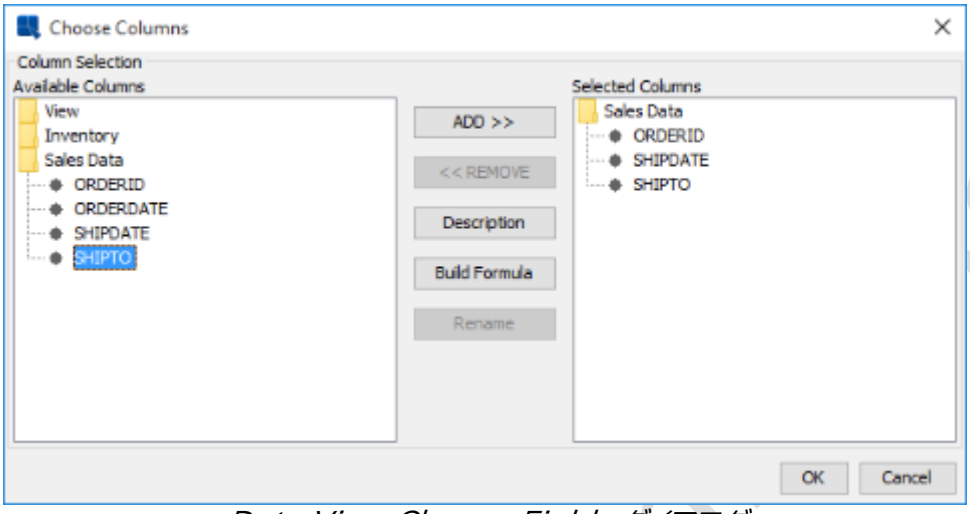
ヘッダーを追加することで、データビュー内のフィールドをグループ化することもできます。これにより、1 つの見出しの下にある異なるデータベーステーブルのデータをグループ化する、独自の仮想テーブルの組織構造を作成できます。見出しを作成するには、**Add Heading** ボタンをクリックします。見出しの名前を指定するよう求められます。新しい見出しは、右のウィンドウにフォルダとして表示されます。見出しの下にフィールドを追加するには、まず右側のウィンドウから追加するフィールドを選択し、**Group Fields** ボタンをクリックします。ドロップダウンメニューが表示され、フィールドを追加する見出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、列を参照してください。

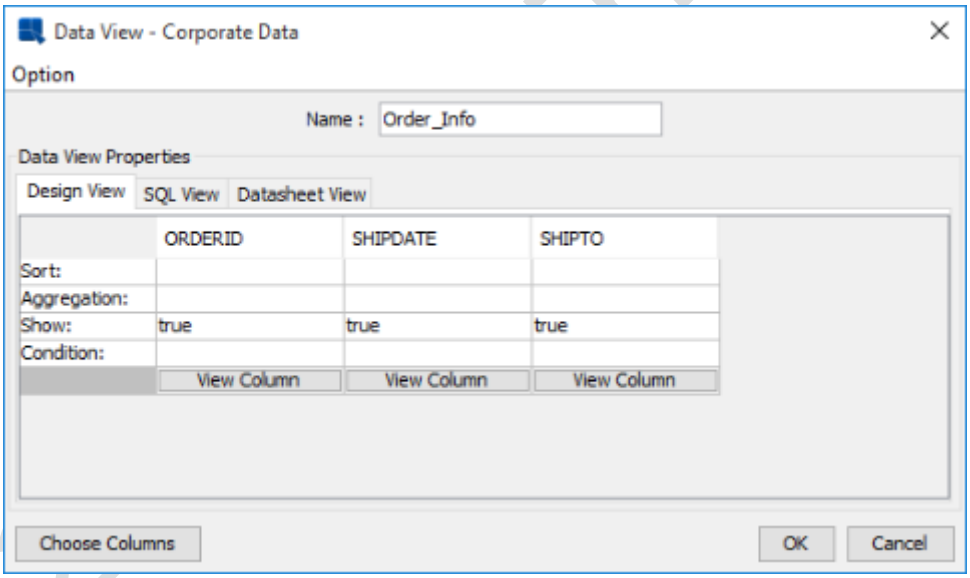
データビューの作成が終了したら、**OK** ボタンをクリックすると、データビューが Data Source Manager に追加されます。ユーザはこのビューを使用してアドホックなクエリを作成できるようになります。

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。

6.2.3.1 データビューのパラメータ

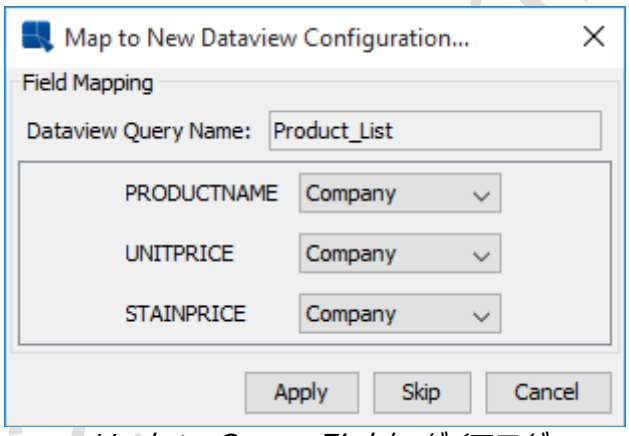
クエリビルダと同様に、ユーザはデータビューでクエリパラメータを指定できます。データビューにパラメータを追加するには、Data Source Manager でデータビューを選択し、**View** をクリックしてデータビューを実行します。データビューのフィールドを選択し、条件ウィンドウに入ったら、列の **Condition** フィールドを右クリックし、ポップアップメニューから **Build** を選択します。これにより、式ビルダが表示され、クエリビルダと同じ方法でパラメータを指定できます。詳細は、パラメータ化されたクエリを参照してください。

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。

6.2.3.2 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

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。

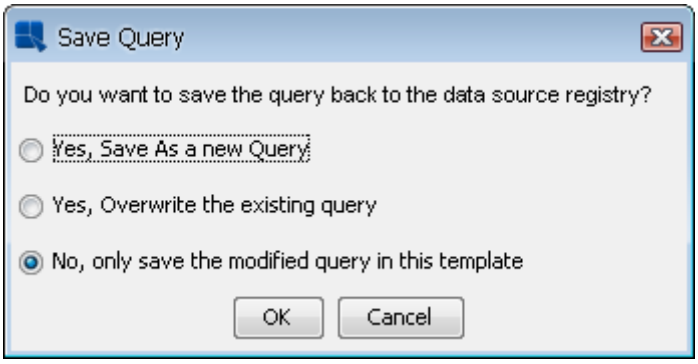
6.2.4 クエリの編集

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グラフを構築することを選択した場合は、Data Source Manager に戻らずに Chart Designer から直接クエリを変更できます

チャートのクエリを変更するには、Chart Designer のデータメニューからクエリの変更を選択します。クエリビルダでクエリを設計した場合は、クエリビルダーインターフェイスが再度開き、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、データビュー条件ウィンドウが再度開き、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。

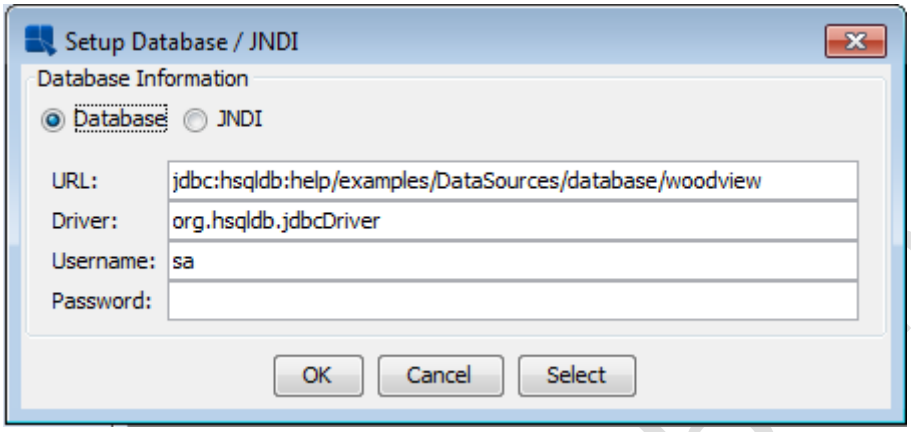


Saving Query Options

保存オプションを指定すると、変更されたクエリがグラフに適用されます。クエリを大幅に変更した場合は、グラフ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、グラフの種類とデータのマッピングを参照してください。

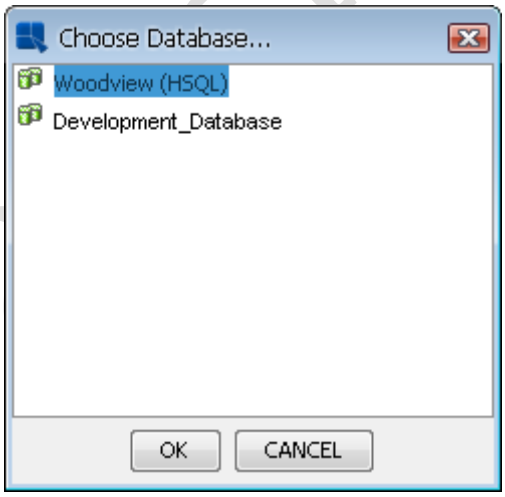
6.2.5 データベース接続の編集

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、データベースまたは JNDI データソースに接続するときに使用するチャートの別の設定を指定できるダイアログが表示されます。



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。



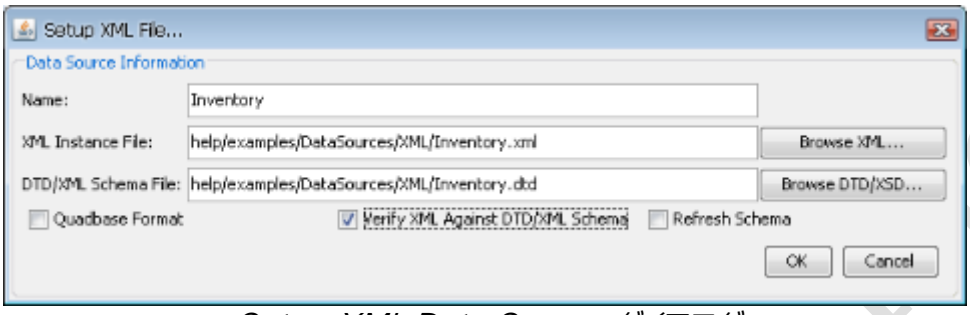
レジストリからデータベースを選択

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後、**OK** ボタンをクリックすると、変更が適用されます。

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。

6.3 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、EspressChart 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

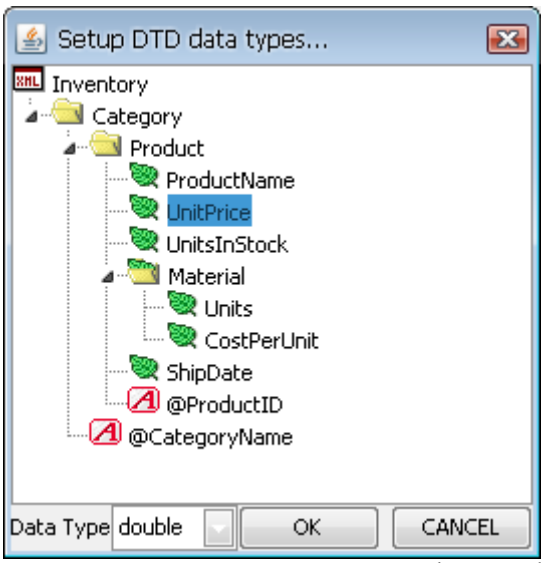
Name では、XML データソースの表示名を指定できます。**XML Instance File** では、データを取得する XML ファイルの場所を指定できます。このフィールドに XBRL ファイルを指定することもできます。ファイルの場所として適切な URL を追加することで、ここでも HTTP サーバから XML データを取得するデータソースを設定できます。**DTD/XML Schema File** では、XML ファイルの有効な DTD または XML スキーマファイルの場所を指定できます。

Quadbase Format チェックボックスを使用すると、XML ファイルが EspressChart 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。



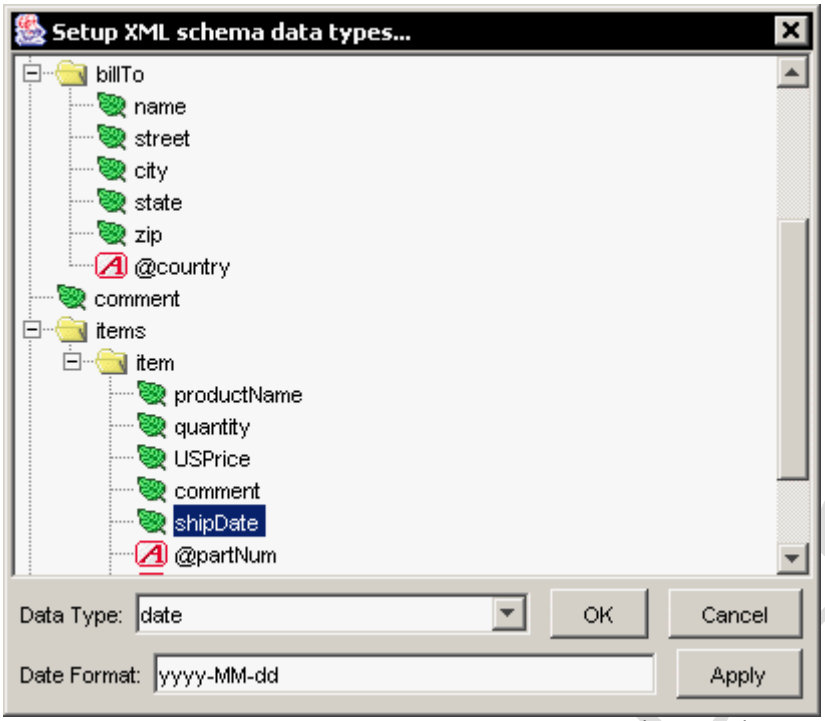
DTD Data Type Selection ダイアログ

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エレメント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。

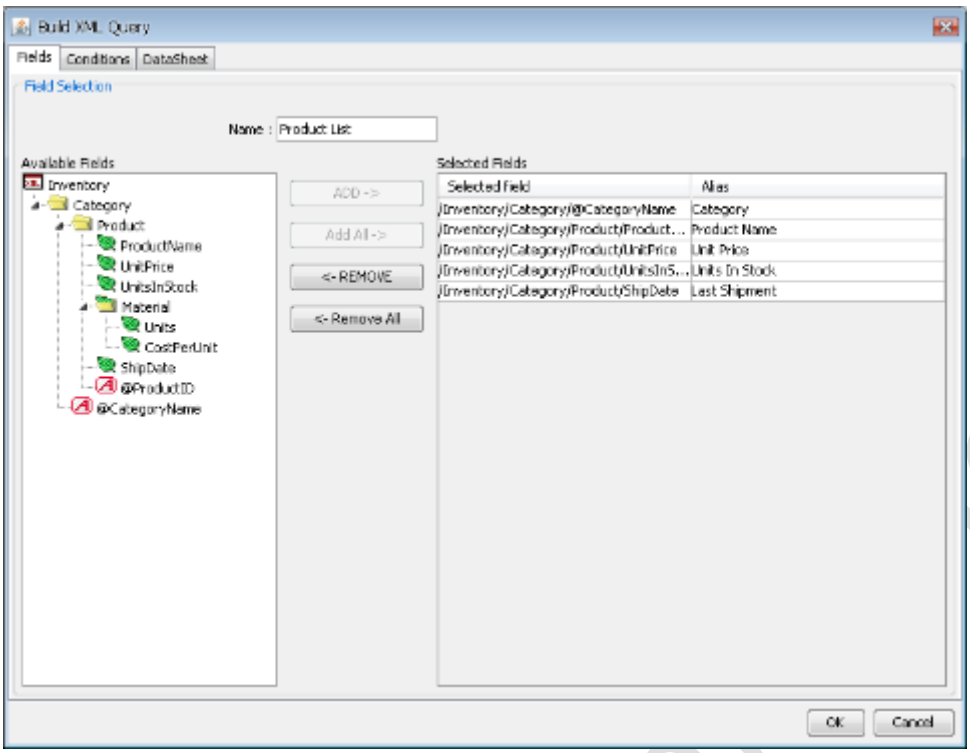


XML Schema Data Type Selection ダイアログ

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。

6.3.1 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 EspressChart 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 クエリビルダーインターフェイスが起動します。

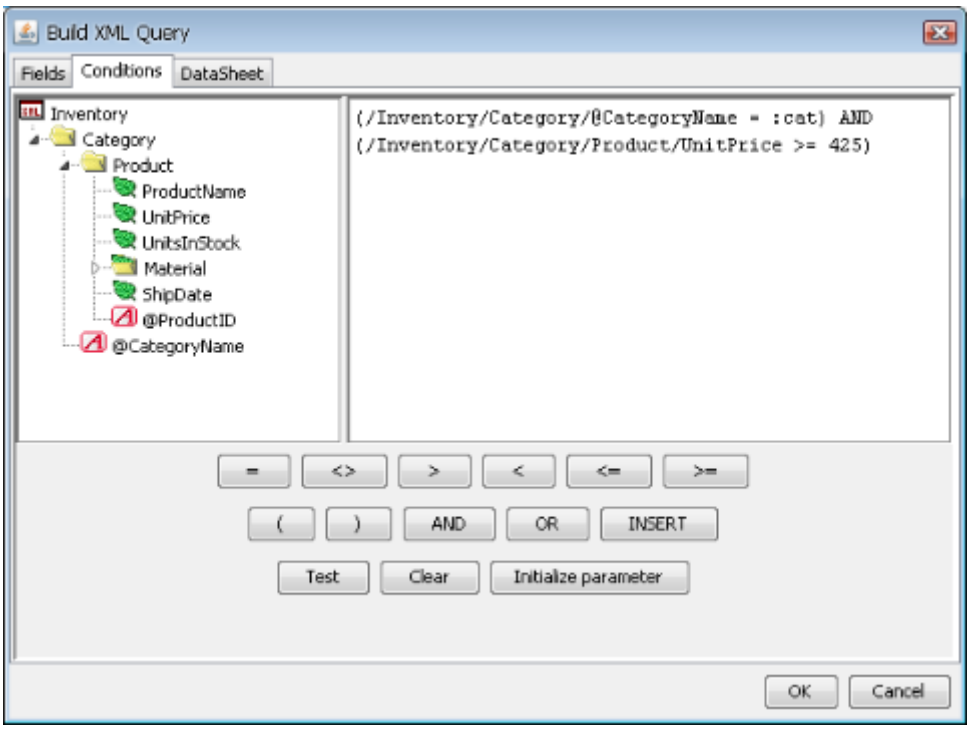


XML Query Field Selection タブ

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。



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。

EspressChart 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。

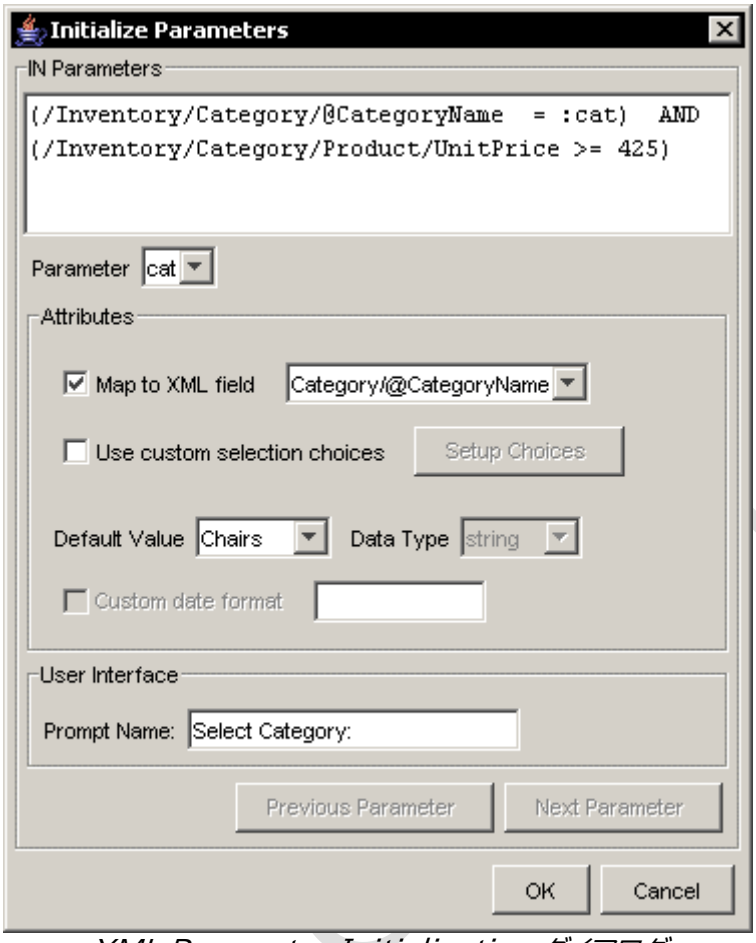
6.3.1.1 XML パラメータ

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問合せ条件の中に表示するために使用されます。
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

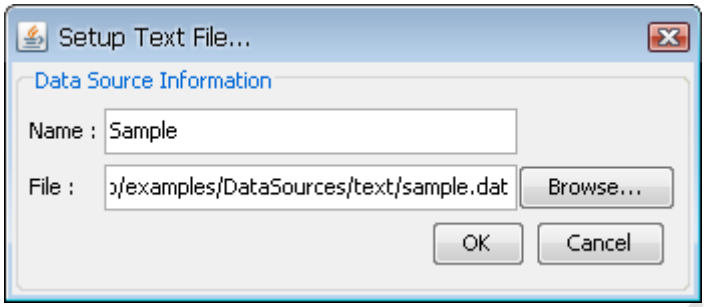
ードにマップできることです。



XML Parameter Initialization ダイアログ

6.4 テキストファイルからのデータ

EspressChart 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。



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。

6.4.1 テキストファイルのフォーマット要件

EspressChart 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)。

6.4.2 テキストファイルのデータ型と書式

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 EspressChart
Boolean, logical, bit	BIT	Boolean
tinyint	TINYINT	byte

smallint, short	SMALLINT	short
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

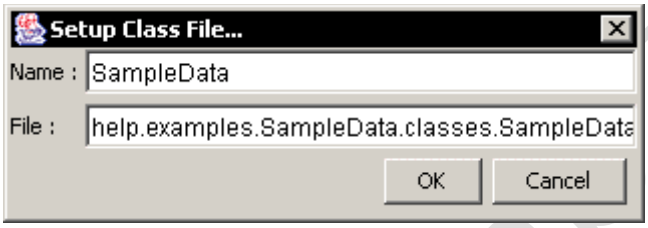
EspressChart インストールにはサンプルテキストファイルが含まれています。このファイルは、インストールの **help/examples/DataSources/text** ディレクトリにあり、**Sample.dat** と呼ばれています。

6.5 クラスファイルからのデータ

柔軟性を最大限に高めるために、EspressChart では、Chart Designer に引数として渡すためのインターフェイ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** インターフェイスと同様の 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、EspressChart 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、カスタム実装の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。



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。

6.5.1 パラメータ化されたクラスファイル

EspressChart 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 インターフェイス **IParameterizedDataSource** が追加されています。

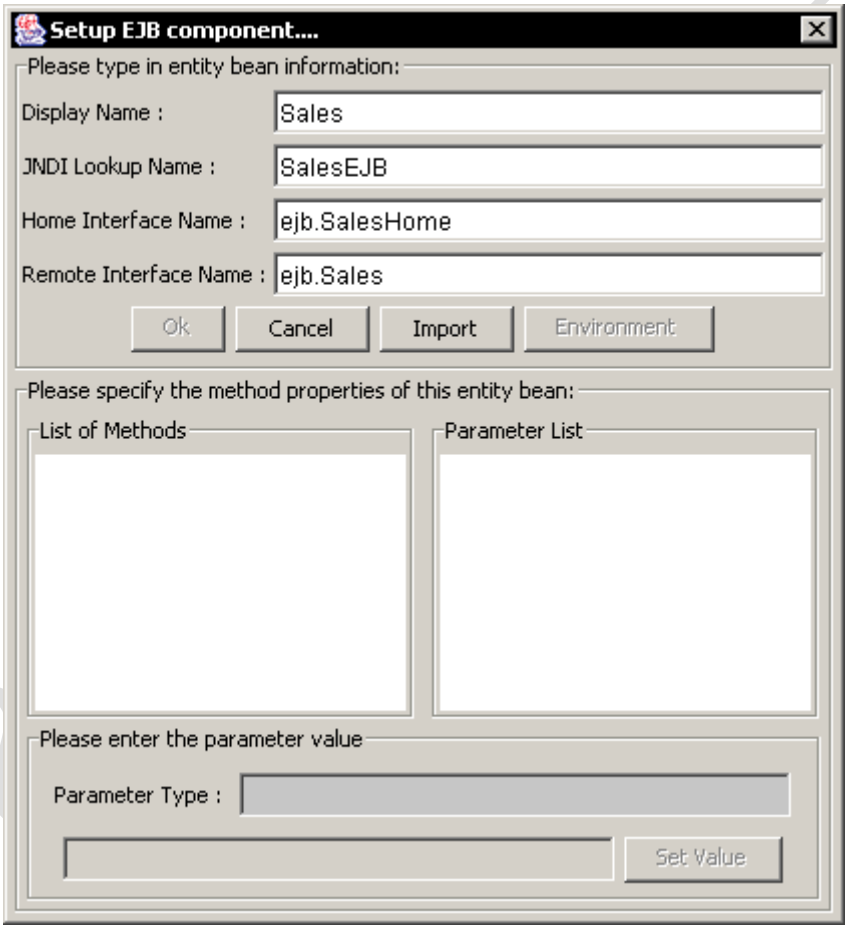
クラスファイルのコンテキスト内で定義されるパラメータは、クエリパラメータと同じ方法で動作します。パラメータ化されたクラスファイルのデータソースの設定の詳細については、カスタム実装で渡されたデータを参照してください。

6.6 EJB からのデータ

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ー(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、永続データインターフェイスは 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、EspressChart 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、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。

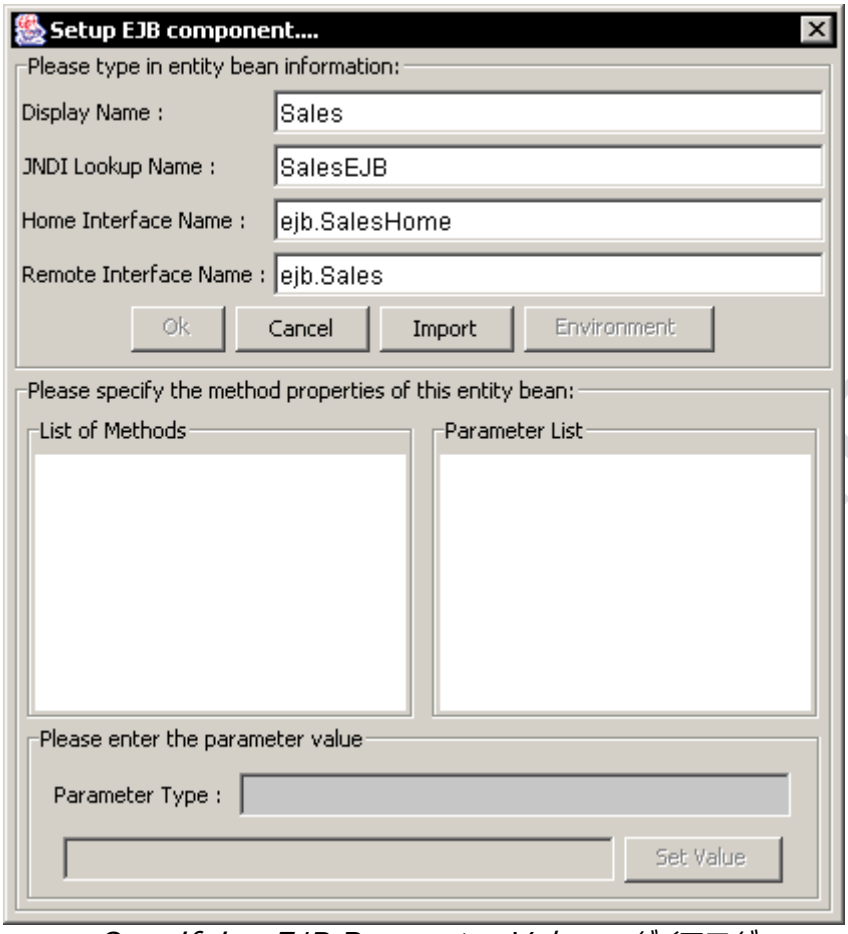


Add EJB ダイアログ

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、ホームインターフェイスとリモートインターフェイ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ーフェイスとローカルインターフェイスを指定します。すべての情報を指定したら、**Import** ボタンをクリックして、ホームインターフェイスを分析し、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。

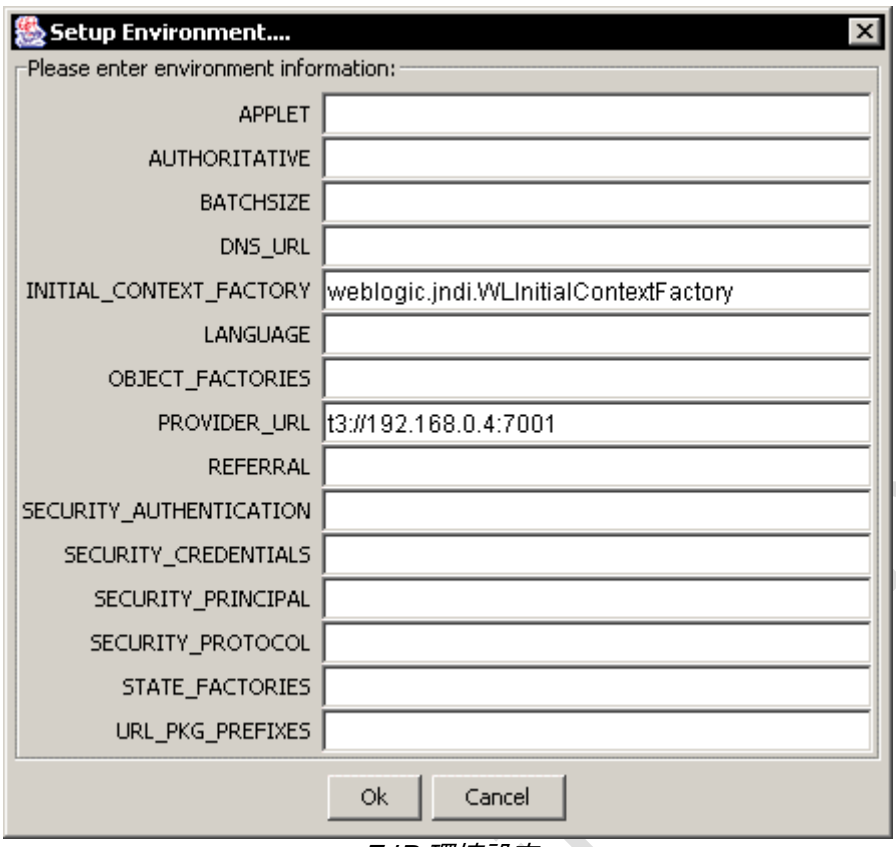
同じダイアログを使用して、ファインダーメソッドに存在するパラメータに基づいて取得されるデータをフィ

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。



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。



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ーフェイ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。

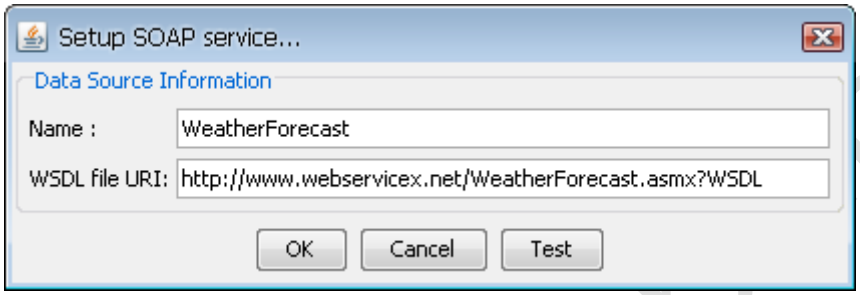
EJB ソースを選択して **Edit** ボタンをクリックすると、パラメータ値を変更できます。

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。

6.7 WSDL をサポートする SOAP のデータ

EspressChart 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。

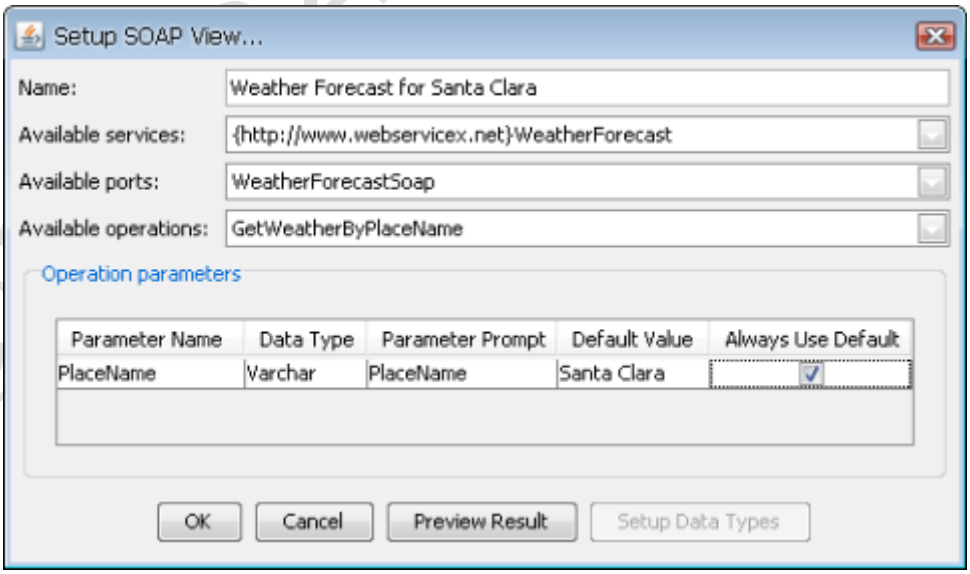
SOAP データソースを設定するには、Data Source Manager で **SOAPServices** ノードを選択し、**Add** ボタンをクリックします。新しい SOAP データソースのオプションを指定するダイアログが表示されます。



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、EspressChart 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。

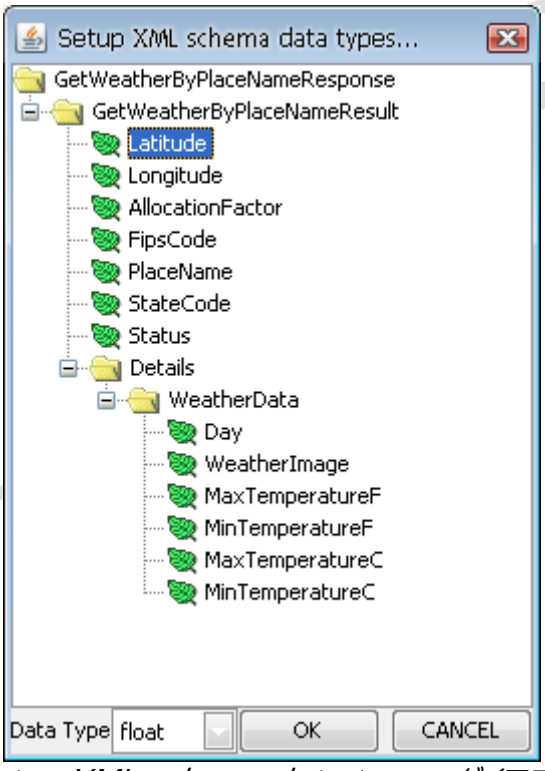


Setup SOAP View ダイアログ

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

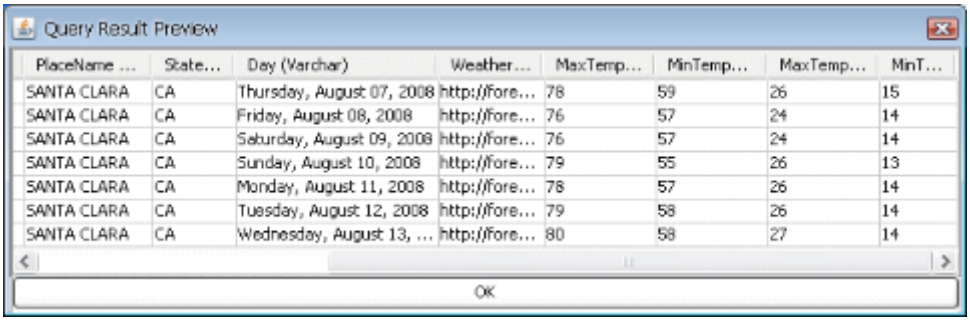
のポートのすべての操作が入力されます。任意のドロップダウンリストの上にマウスを移動すると、サービス/ポート/操作のドキュメント(WSDL ファイルにドキュメントがある場合)のヒントが表示されます。サービス、ポート、および操作を指定すると、操作のすべてのパラメータが読み取られます。いくつかのパラメータがある場合、それらはテーブルに表示されます。表の最初の 2 つの列は編集できません。それらは WSDL ファイルから読み込まれます。次の 2 列は編集可能で、パラメータプロンプトとデフォルト値を指定できます。最後の列には、デフォルトのパラメータ値を常に使用するかどうかを選択できるチェックボックスがあります。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。



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。

6.8 Salesforce のデータ

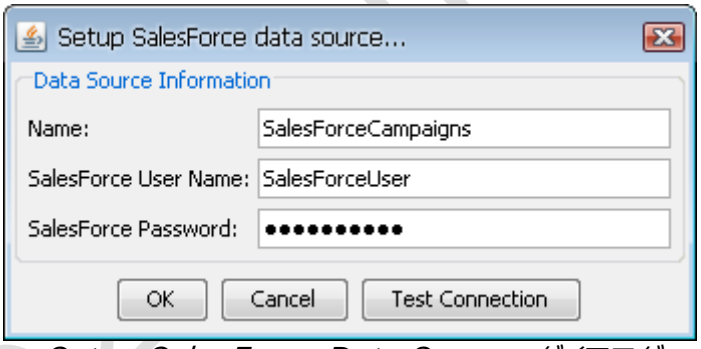
Salesforce データソースは、Salesforce データを EspressChart に表示する既存の Salesforce ユーザ向けに設計されています。Salesforce サーバーへの接続は、Salesforce Partner WSDL(バージョン 13.0)を使用して SOAP 経由で確立されます。ユーザは SOQL(Salesforce Object Query Language)クエリによって Salesforce サーバーと通信します。このデータソースで作業するには、ユーザ名とパスワードの有効な Salesforce アカウントが必要です。さらに、EspressChart 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

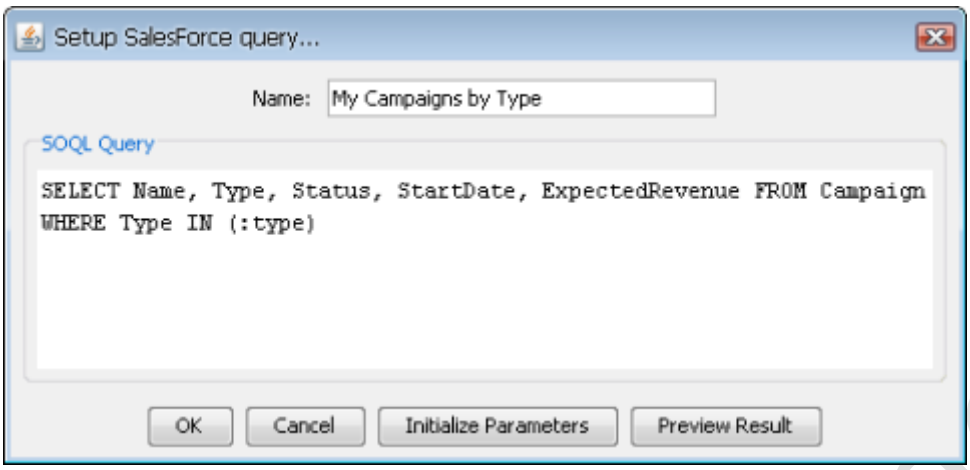
Salesforce ユーザのアカウントを有効にする

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。



Setup SalesForce Query ダイアログ

現在の EspressChart 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。親から子への問合せ(ネストされた SOQL 問合せを使用)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
```

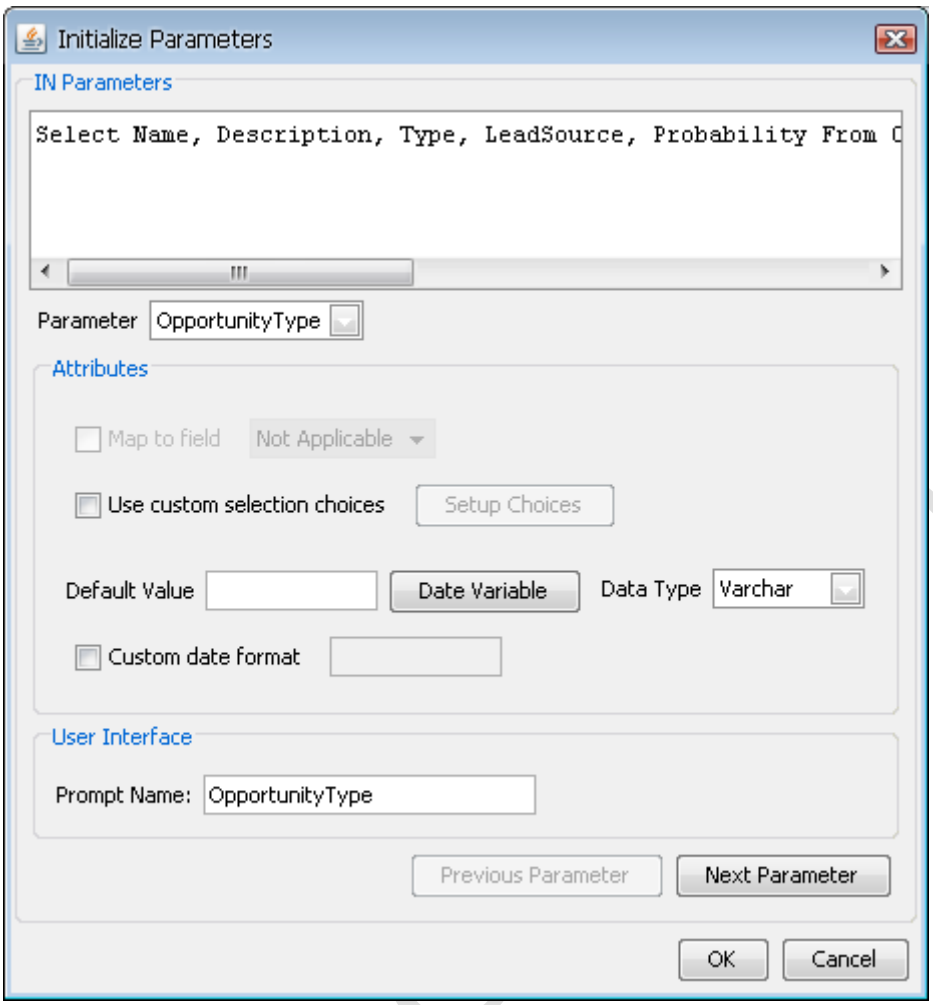
実行時にキャンペーン名を入力し、そのキャンペーンのデータのみを取得することができます。

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。



Initialize Parameters ダイアログ

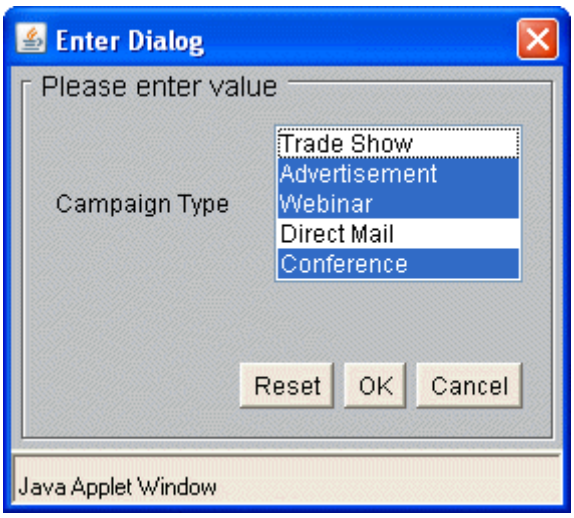
このダイアログでは、以下のオプションを指定できます。

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。

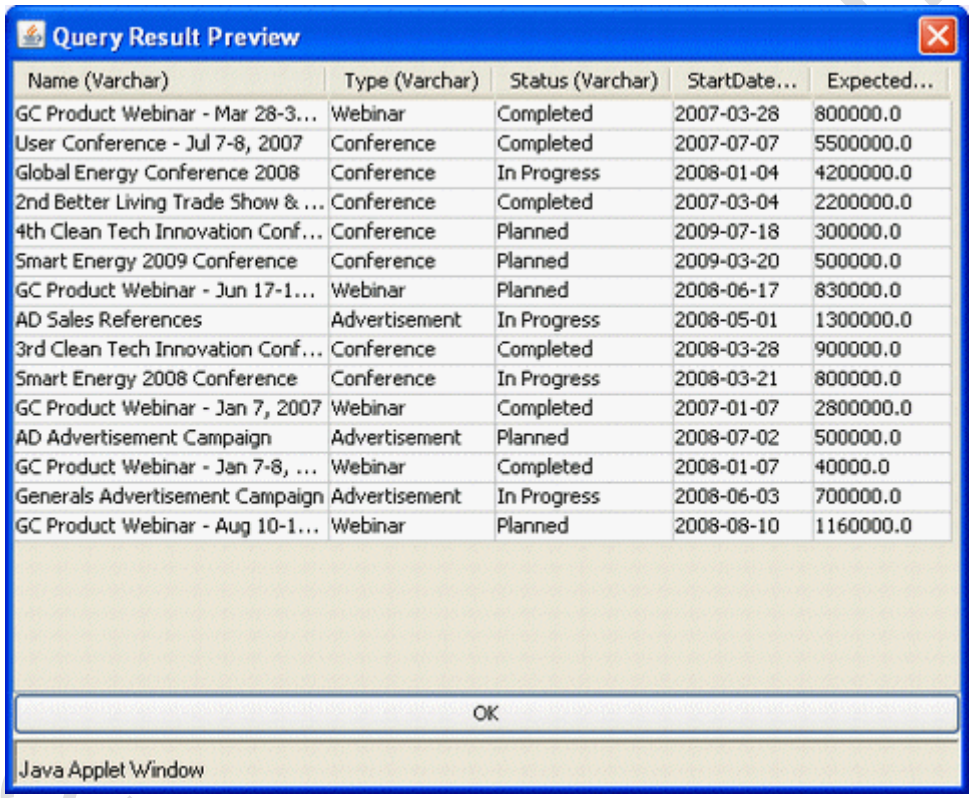
カスタム選択項目を使用する
個別の列の値をすべて含むドロップダウンメニューを表示するのではなく、パラメータ値のカスタムリストを作成できます。リストを設定するには、このオプションを選択し、**Setup Choices** ボタンをクリックします。これにより、新しいダイアログが開き、選択肢のリストを作成することができます。

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。

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



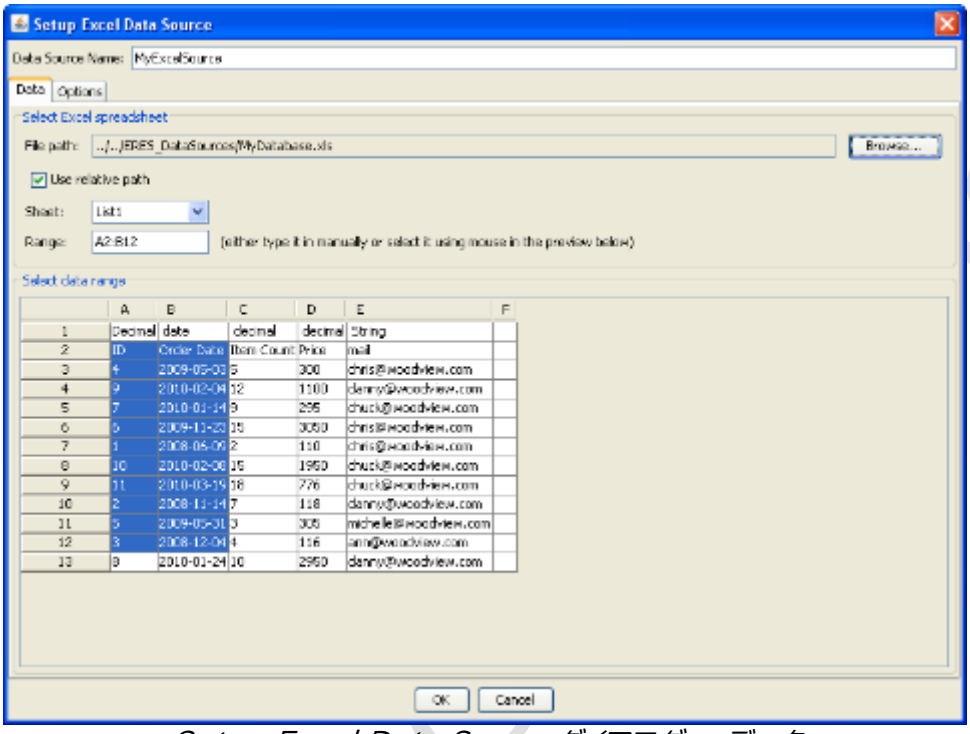
Query Result Preview ダイアログ

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。

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。

6.9 Excel ファイルのデータ

EspressChart 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 EspressChart 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 EspressChart 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。

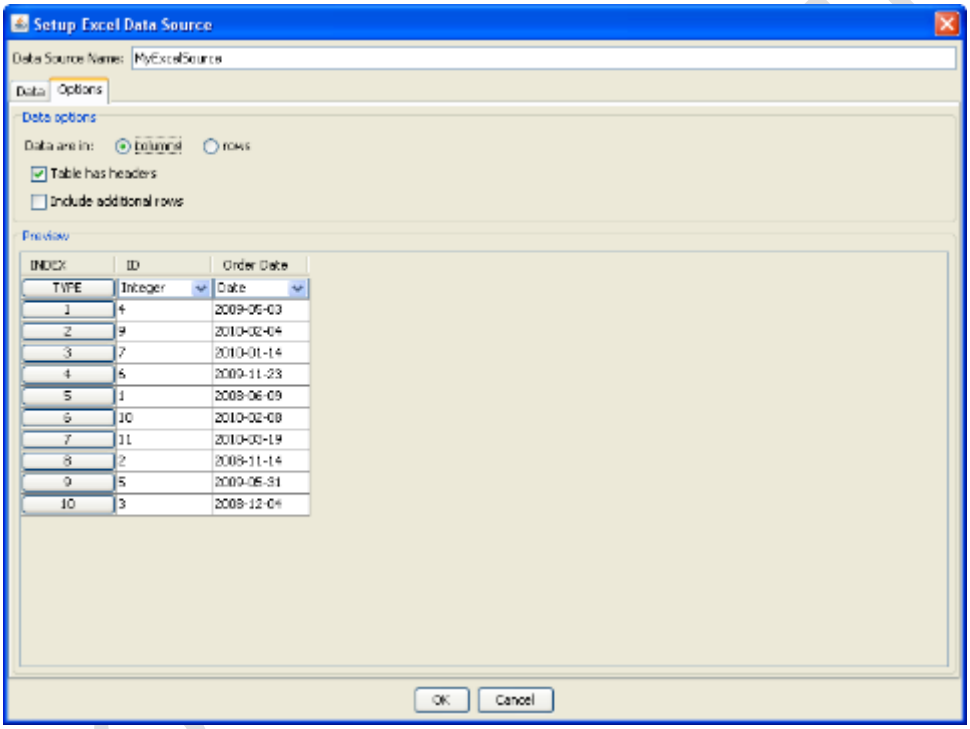
EspressChart は、* **.xls** ファイルと* **.xlsx** ファイルの両方を処理できます。* **.xlsx** ファイルは Microsoft Excel 2007 以降で使用され、Open XML に基づいています。

EspressChart 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。



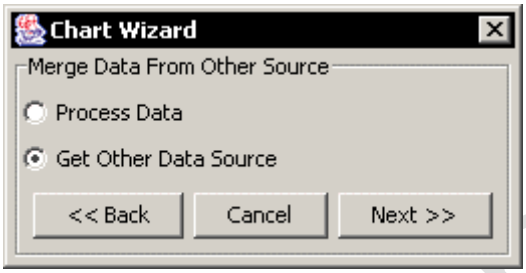
Setup Excel Data Source ダイアログ - オプション

6.10 グラフデータの使用

6.10.1 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。

EspressChart 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。

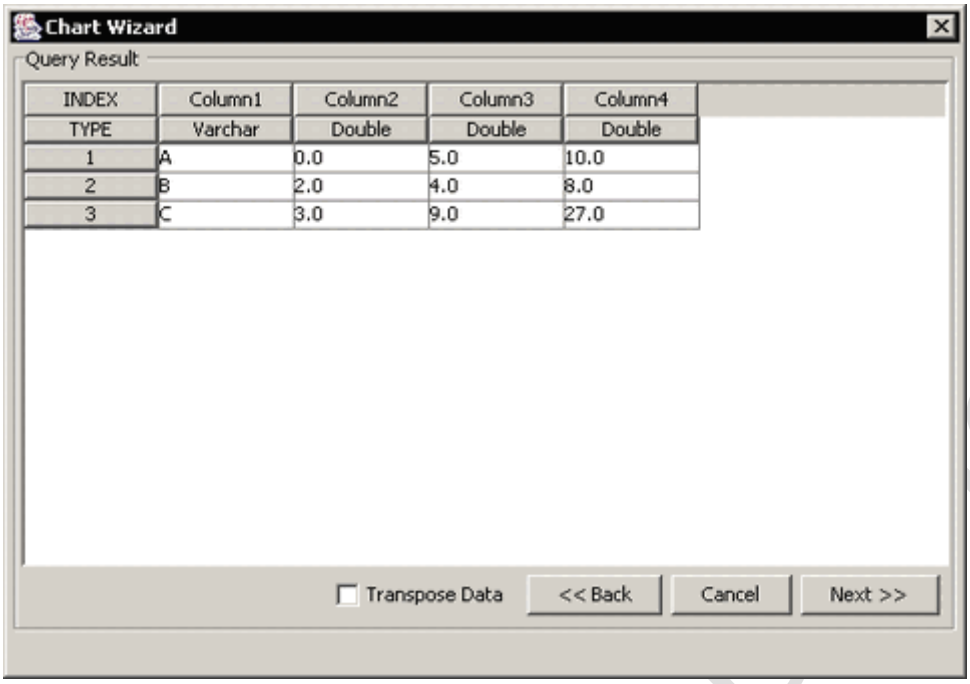


Additional Data Source ダイアログ

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。

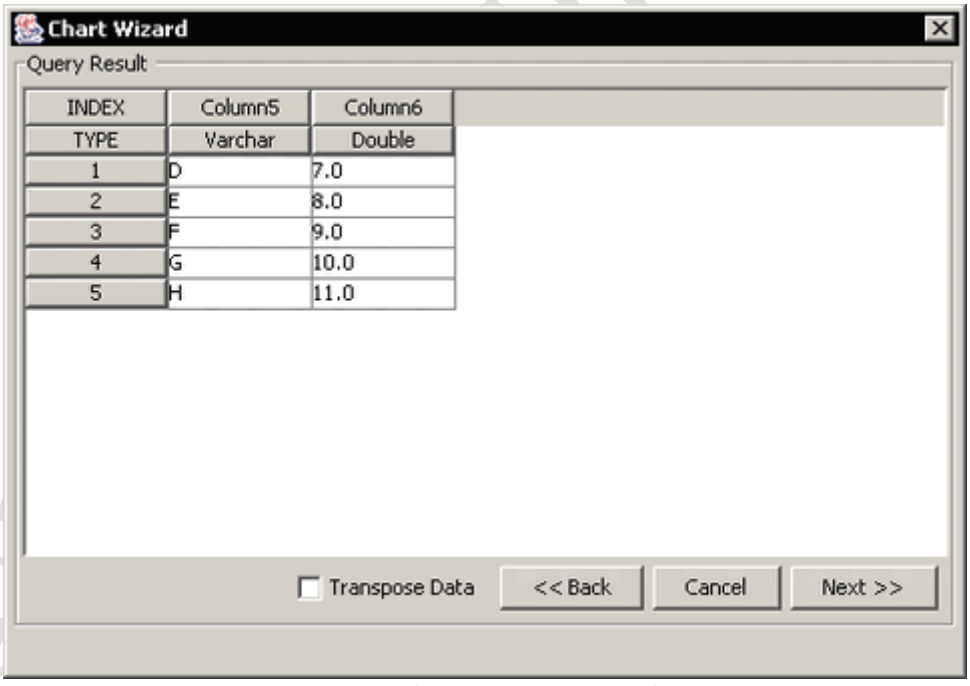
複数のデータソースがデータテーブル内で横方向に結合されています。これは、2 番目(または 3 番目、4 番目など)のデータソースの列が、最初のデータソースの列の右隣りの配置されることを意味します。1 つのデータソースの列に他の行よりも多くの行がある場合は、NULL 値が余分な行に配置されます。

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。



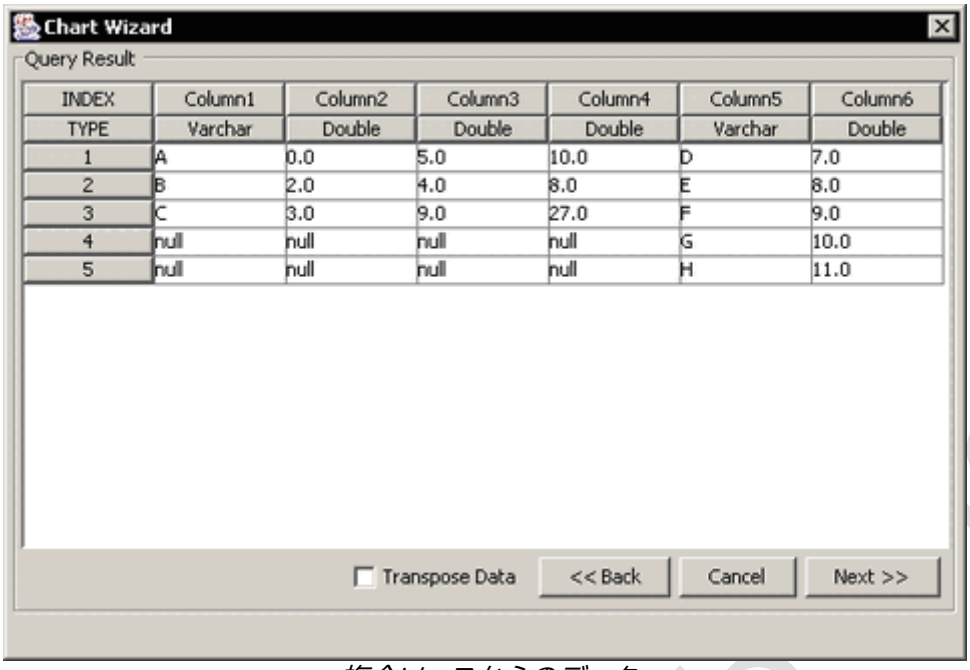
最初のデータソースからのデータ

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。



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。

6.10.2 データソースの変数

チャートデザイン中のどの時点でも、テンプレートのデータソースを変更することができます。チャート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、Chart 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かぎり、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、データメニューから **Modify Data Source** を選択するか、ツールバーの **Modify 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。

データソースを変更するときは、データをグラフに再マップする必要があります。レイアウトとデータマッピングの詳細については、グラフタイプとデータマッピングを参照してください。

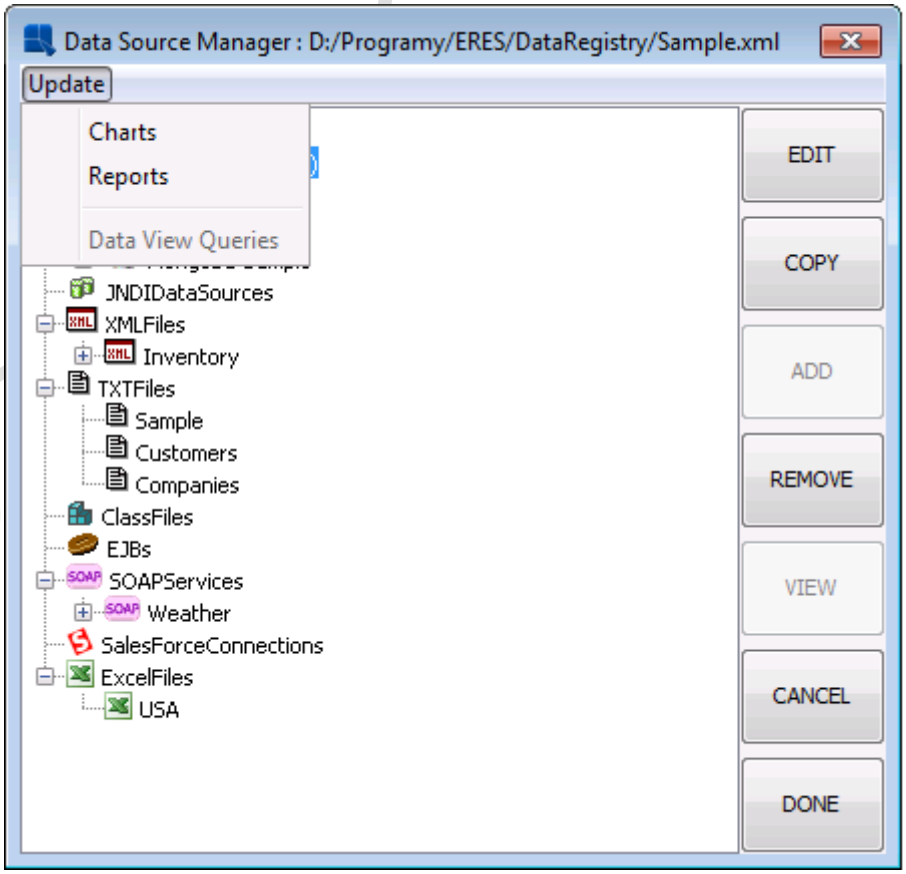
6.11 データソースの更新

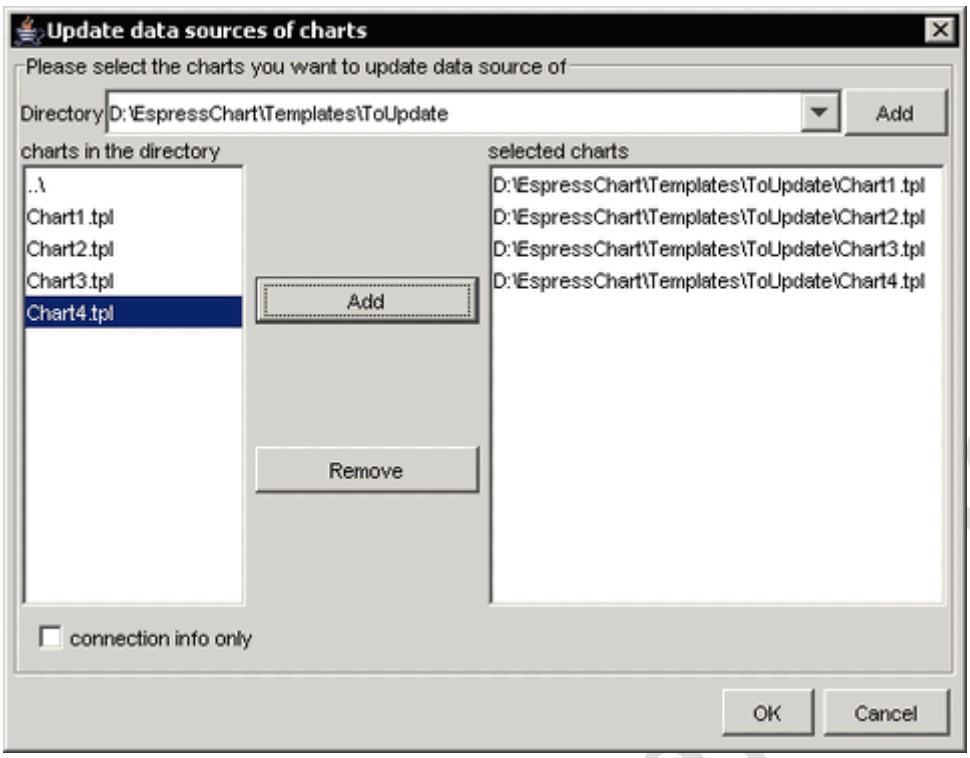
テンプレートのグループを移動したり、ある場所から別の場所に EspressChart を完全にインストールしたりするには、多くの状況があります。たとえば、アプリケーションを開発環境から実稼働環境に移行することができます。各環境では、データソースの場所と接続情報が異なる場合があります。このシナリオでは、前のセクションで説明したようにチャートテンプレートを 1 つずつ更新することは、多数のテンプレートの接続情報を変更することは現実的ではありません。代わりに、EspressChart を使用すると、データレジストリの情報に基づいてテンプレートのグループをすばやく更新できます。

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。





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。

6.12 CDataJDBC ドライバ

EspressChart に付属していないデータソースドライバに接続したい場合、CData JDBC ドライバは興味深い選択肢です。

CData の JDBC ドライバは、Excel や JSON のような非データベースのデータソースの機能を向上させるためにも使用できます。

CData JDBC ドライバはサードパーティの商用製品ですが、購入前に無料トライアル版をインストールして製品を試すことができます。

CData JDBC ドライバはサードパーティ製品です。ドライバを使用したい場合は、

https://www.cdata.com ができで購入する必要があります。

6.12.1 対応 CData ドライバ

現在、以下の CData JDBC ドライバをサポートしています:

- Salesforce
- BigQuery
- Excel
- JSON
- MongoDB
- Kintone

その他の CData JDBC ドライバも動作する可能性がありますが、サポートされていないドライバについては、完全な機能を保証することはできません。ここに記載されていないドライバへの接続が必要な場合は、こちらまでお問い合わせください。

6.12.1.1 Excel

ダウンロード: https://www.cdata.com/drivers/excel/jdbc
CData 公式ドキュメント(JDBC ドライバのみ):https://cdn.cdata.com/help/RXF/jdbc/

ドライバのダウンロード後、以下の章を参照してください:

- 6.12.2 章 - CData JDBC ドライバのインストール
- 6.12.3 章 - EspressChart での CData JDBC ドライバの配置
- 6.12.4 章 - DataSource Manager での CData JDBC ドライバの使用

CData Excel ドライバを使用すると、Excel ファイル上で SQL クエリをデータベースのように実行できます。これにより、QueryBuilder を使用したり、グラフィカルユーザーインターフェイスでデータビューを構築したり、SQL クエリを手動で(クエリビルダ内で)記述したりすることができます。

Excel ファイルをデータソースとして使用して、パラメータや複数値のパラメータを持つクエリを記述することもできます。

Excel には通常のデータベースのようにカラムのデータ型が設定されていないため、カラムのデータ型を決定するのが少し難しい場合があります。デフォルトでは、CData Excel ドライバの接続 URL に以下のパラメータが追加されています:
TypeDetectionScheme=RowScan;
RowScanDepth=10;

これにより、CData Excel ドライバは、選択された Excel ファイルを読み込む際に最初の 10 行をスキャンし、最初の 10 行のデータに基づいて各列のデータソースを自動的に検出します。

ただし、各列のデータ型を決定する方法には複数のオプションがあります。

詳細なオプションについては、CData JDBC Excel ドライバのドキュメントにある TypeDetectionScheme パラメータを参照してください。

TypeDetectionScheme の値は、Data Source Manager の Setup Database...ダイアログの URL フィールド(Excel ファイルのパスを入力したフィールド)で変更できます。

6.12.1.2 JSON

ダウンロード: https://www.cdata.com/drivers/json/jdbc/
CData 公式ドキュメント(JDBC ドライバのみ): https://cdn.cdata.com/help/DJF/jdbc/

ドライバのダウンロード後、以下の章を参照してください:
- 6.12.2 章 - CData JDBC ドライバのインストール
- 6.12.3 章 - EspressChart への CData JDBC ドライバのデプロイ
- 6.12.4 章 - DataSource Manager での CData JDBC ドライバの使用

CData json ドライバを使用すると、データベースのように json ファイル上で SQL クエリを実行できます。これにより、クエリビルダを使用したり、グラフィカルユーザーインターフェイスでデータビューを構築したり、SQL クエリを手動で(クエリビルダ内で)記述したりすることができます。

JSON ファイルをデータソースとして使用して、パラメータや複数値のパラメータを持つクエリを記述することもできます。

6.12.1.3 Salesforce

Salesforce に接続するには:
ダウンロード: https://www.cdata.com/drivers/salesforce/jdbc
CData 公式ドキュメント（JDBC ドライバのみ): https://cdn.cdata.com/help/RFF/jdbc

ドライバのダウンロード後、以下の章を参照してください:
- 6.12.2 章 - CData JDBC ドライバのインストール
- 6.12.3 章 - EspressChart への CData JDBC ドライバのデプロイ
- 6.12.4 章 - DataSource Manager での CData JDBC ドライバの使用

6.12.1.4 MongoDB

MongoDB に接続するには:
ダウンロード: https://www.cdata.com/drivers/mongodb/jdbc
https://cdatabuilds.s3.amazonaws.com/support/DGRJV 8598.exe から新しい MongoDB ドライバへのバグ修正が追加されました。
CData 公式ドキュメント（JDBC ドライバのみ): https://cdn.cdata.com/help/DGF/jdbc

ドライバのダウンロード後、以下の章を参照してください:
- 6.12.2 章 - CData JDBC ドライバのインストール
- 6.12.3 章 - EspressChart への CData JDBC ドライバのデプロイ
- 6.12.4 章 - DataSource Manager での CData JDBC ドライバの使用

6.12.1.5 BigQuery

BigQuery に接続するには:
ダウンロード: https://www.cdata.com/drivers/bigquery/jdbc
CData 公式ドキュメント(JDBC ドライバのみ): https://cdn.cdata.com/help/DBF/jdbc

ドライバのダウンロード後、以下の章を参照してください:
- 6.12.2 章 - CData JDBC ドライバのインストール
- 6.12.3 章 - EspressChart への CData JDBC ドライバのデプロイ
- 6.12.4 章 - DataSource Manager での CData JDBC ドライバの使用

6.12.1.6 Kintone

Kintone に接続するには:
ダウンロード:https://www.cdata.com/drivers/kintone/jdbc
CData 公式ドキュメント(JDBC ドライバのみ): https://cdn.cdata.com/help/DBF/jdbc

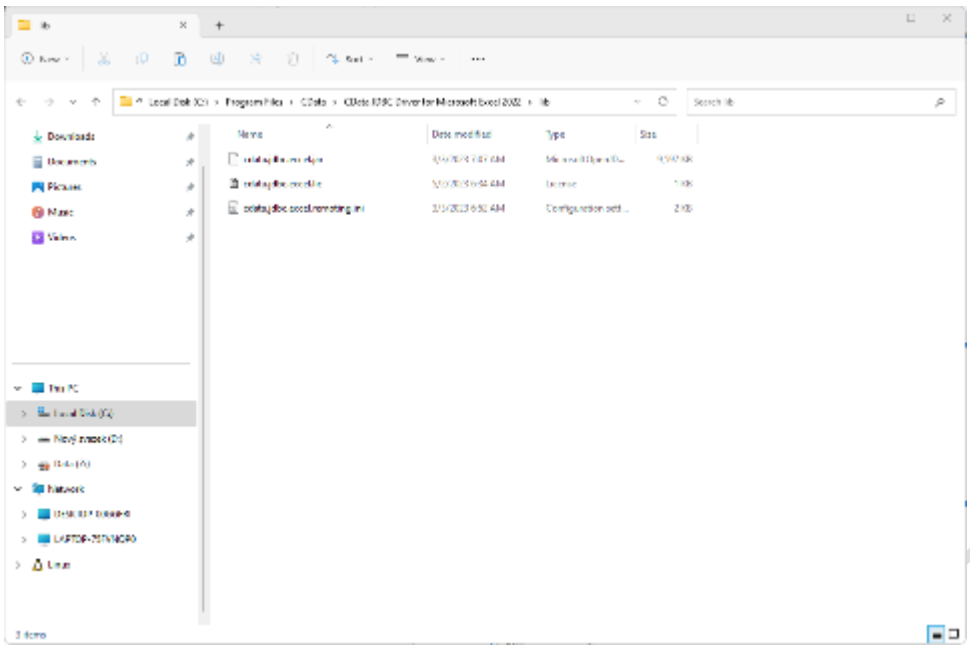
ドライバのダウンロード後、以下の章を参照してください:
- 6.12.2 章 - CData JDBC ドライバのインストール
- 6.12.3 章 - EspressChart への CData JDBC ドライバのデプロイ

- 6.12.4 章 - DataSource Manager での CData JDBC ドライバの使用

6.12.2 CData JDBC ドライバのインストール

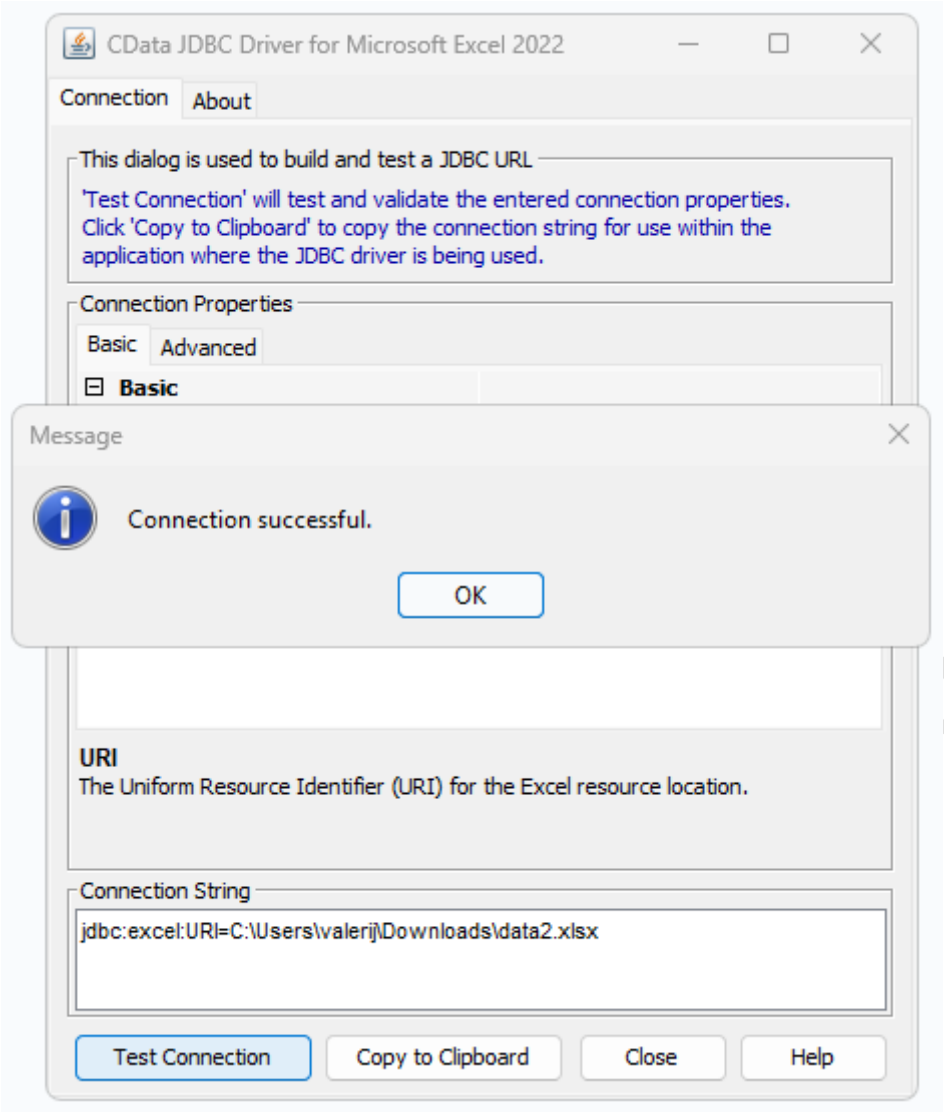
CData JDBC ドライバのインストール手順は、基本的にサポートされているすべてのデータソースで同じです。ここでは、CData Excel JDBC ドライバを例に、インストール手順を説明します。
まず、CData JDBC ドライバをインストールします。サポートされている各 CData JDBC ドライバへのリンクは、以下の章にあります。
インストーラをダウンロードした後、ダイアログに従ってセットアップを進めてください。

次のダイアログでは、有料版のインストールか、30 日間の無料トライアル版のインストールかを選択できます。この例では、トライアル版を使用します。
セットアップが完了すると、JDBC Driver が ”CData JDBC Driver for Microsoft Excel 2022lib” に作成されます。



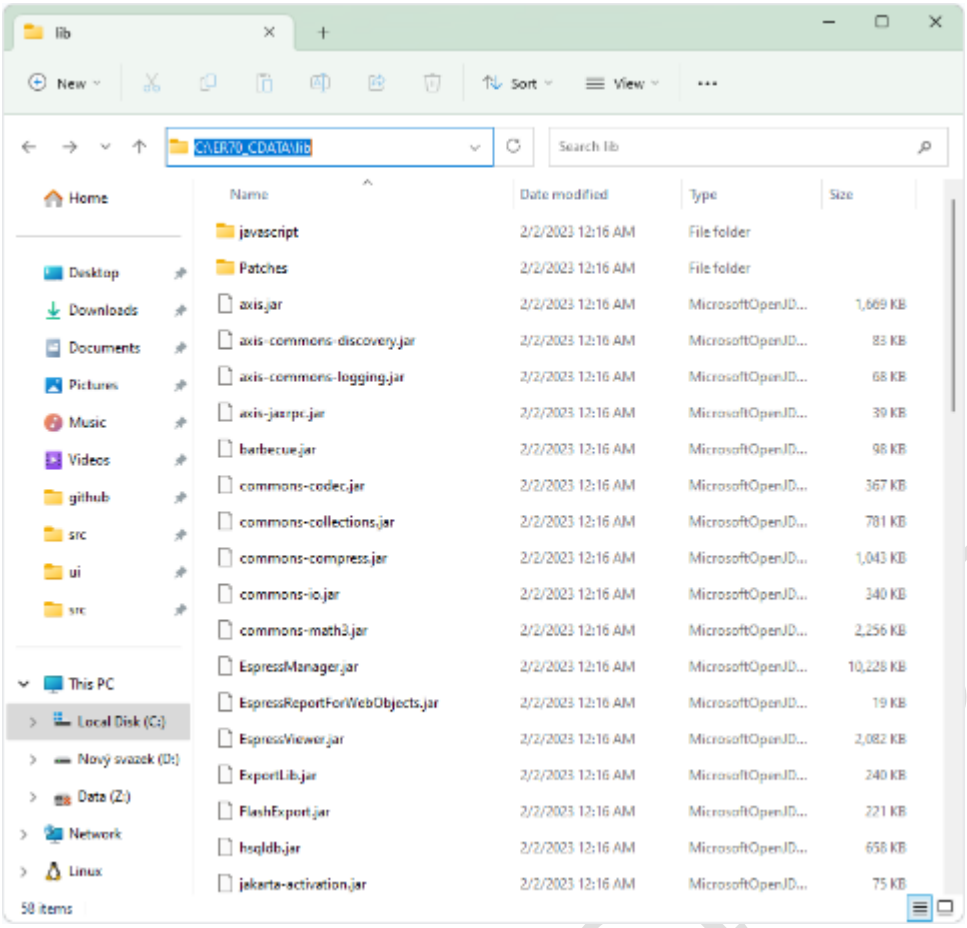
”cdata.jdbc.excel.jar ”を実行すると、接続テストツールが起動します。このツールを使用して、CData ドライバのテストやトラブルシューティングを行うことができます。

接続テストツールは、EspressChart データソースマネージャで使用できる接続 URL も表示します。



6.12.3 EspressChart への CData JDBC ドライバのデプロイ

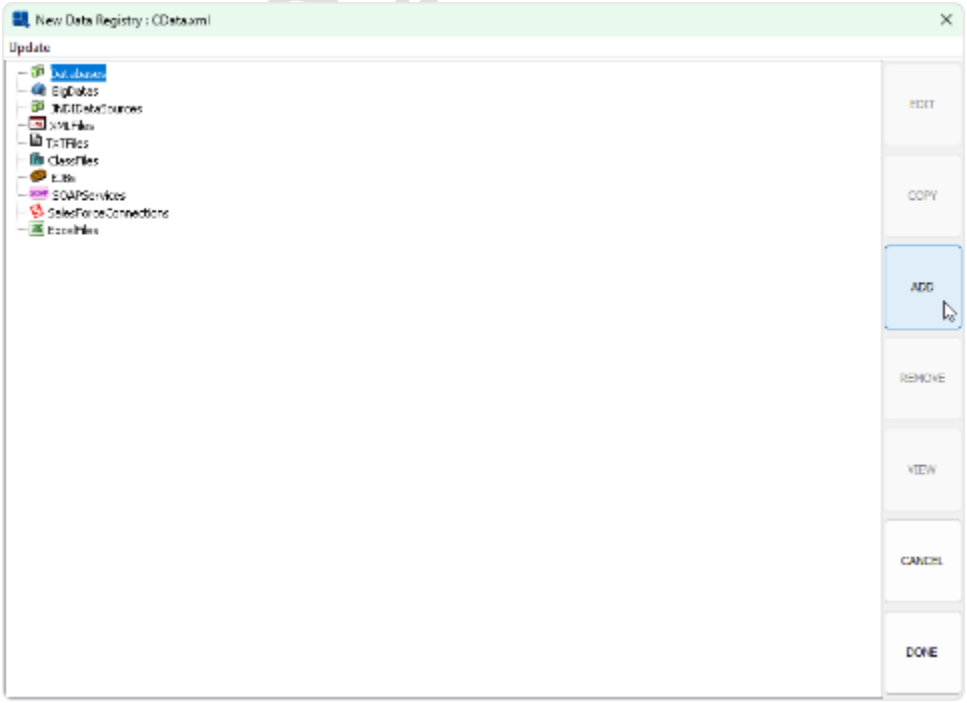
CData インストールディレクトリであるハードドライブ上の CData JDBC Driver for Microsoft Excel 2023lib を見つけます（例：C:¥Program Files¥CData¥CData JDBC Driver for Microsoft Excel 2023¥lib）。このディレクトリには、cdata.jdbc.excel.jar、cdata.jdbc.excel.lic、cdata.jdbc.excel.remoting の 3 つのファイルが含まれています。
この 3 つのファイルを EC/lib/にコピーします。



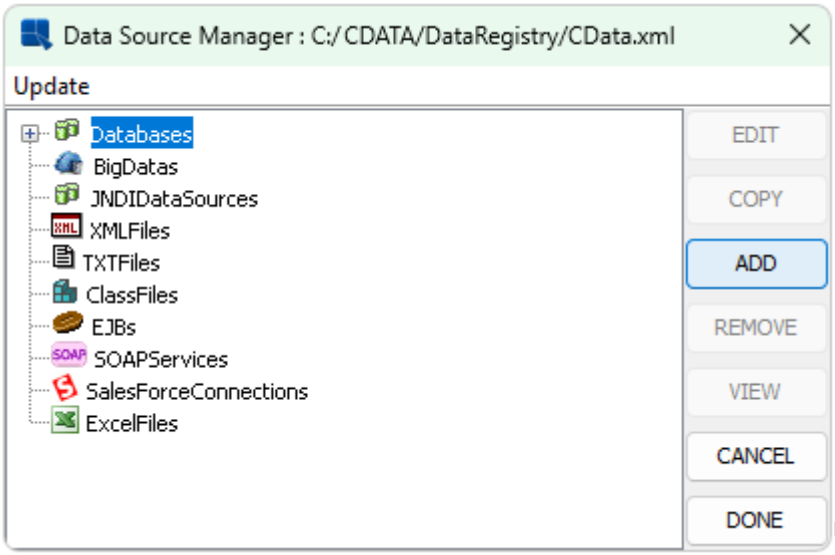
ファイルをコピーしたら、EspressManager を再起動してください(起動している場合)。

6.12.4 データソースマネージャでの CData JDBC ドライバの使用

データソースマネージャーを起動し、データレジストリを開く(既存のものでも新規のものでも可)。
ツリーリストで ”Databases”オプションを選択し、”ADD”ボタンを押してください。



“Driver List: “ドロップダウンメニューで、”CData Excel”（またはインストールする他の CData JDBC ドライバ）を選択します。



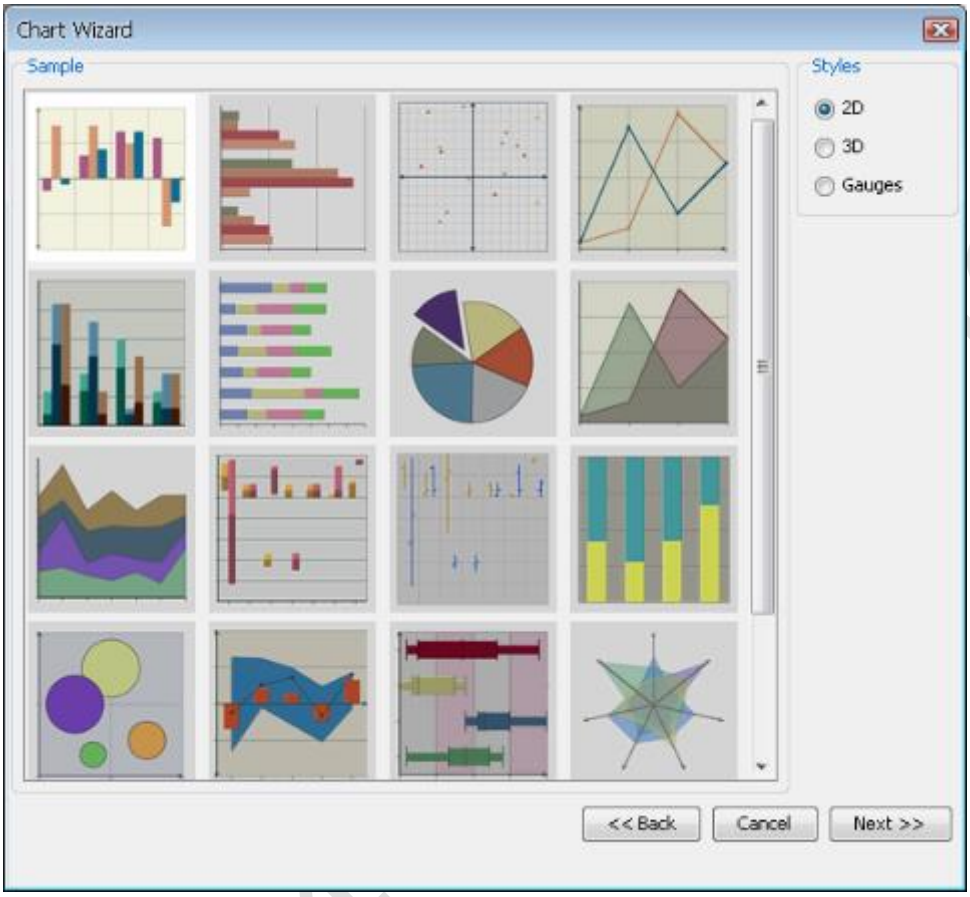
“URL:”テキストフィールドで、プレースホルダー（”EXCEL FILE LOCATION”など）を実際の値に置き換えます。

または、”URL:” テキストフィールドのテキストを、前の章で説明した CData Test Connection ツールで取得した接続文字列に置き換えることもできます。

OK をクリックしてください。これで完了です。Excel ファイルをデータベースと同じように使用することができます。

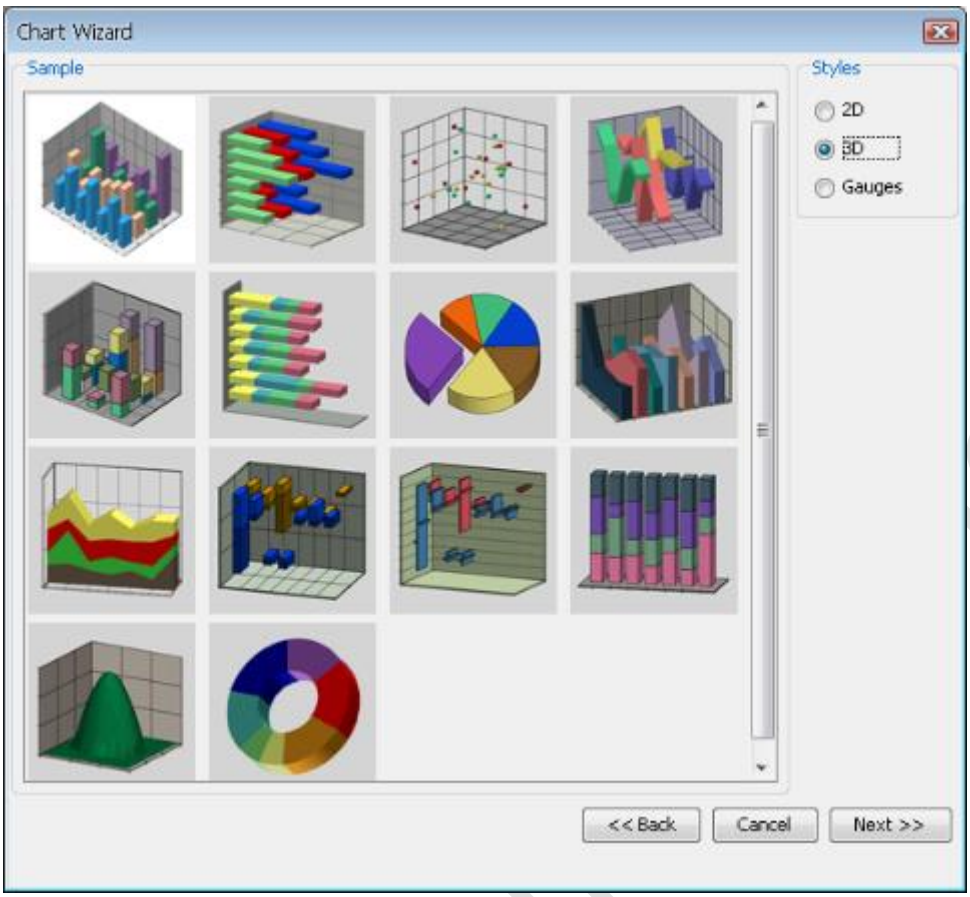
7 チャートタイプとデータのマッピング

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。



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：

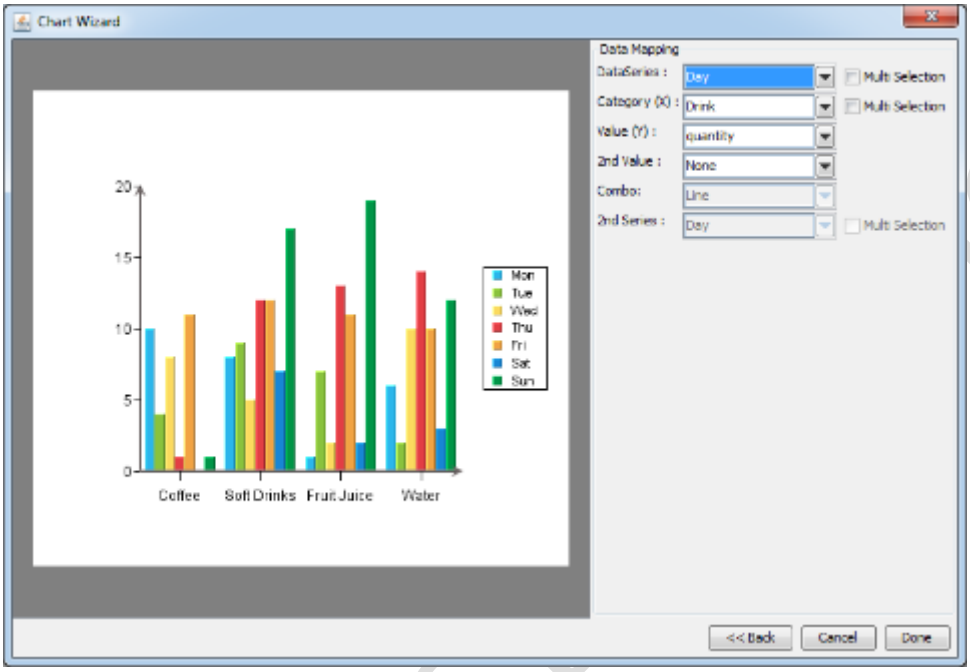
- 縦棒(Column)
- 横棒（Bar）
- 散布図（Scatter）
- 折れ線（Line）
- 積み上げ縦棒（Stack Column）
- 積み上げ横棒（Stack Bar）
- 円グラフ（Pie）
- エリアグラフ（Area）
- 積み上げエリア（Stack Area）
- High Low（High-Low）
- HLCO（HLCO）
- 100%縦棒（100% Column）
- ドーナツ（Doughnut）
- サーフェイス（Surface）（3D のみ）
- バブル（Bubble）（2D のみ）
- 重ね合わせ（Overlay）（2D のみ）
- ボックス（Box）（2D のみ）
- 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。

7.1 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、基本的な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。

7.1.1 データ転置

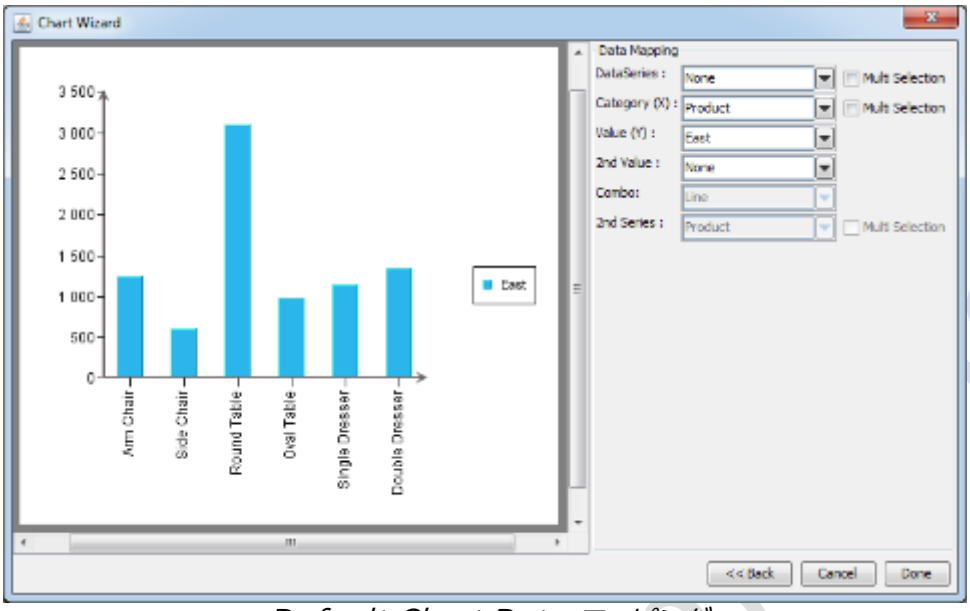
データマッピングダイアログからデータ転置を設定することもできます。Transposing を使用すると、データソースを変更せずに、複数のデータソース列を 1 つのグラフにプロットできます。

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

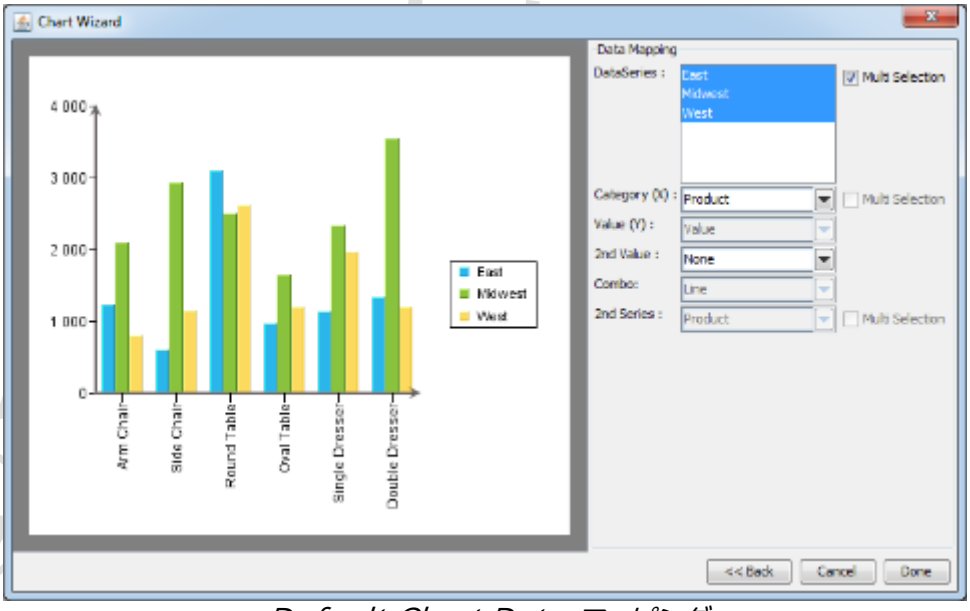
データソースの例

すべての地域のデータを 1 つのチャートにプロットしたい。データ転置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、データソースを転置する必要があります。



Default Chart Data マッピング

データを転置するには、**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。すべての地域列を選択するだけで、チャートは次のようになります：

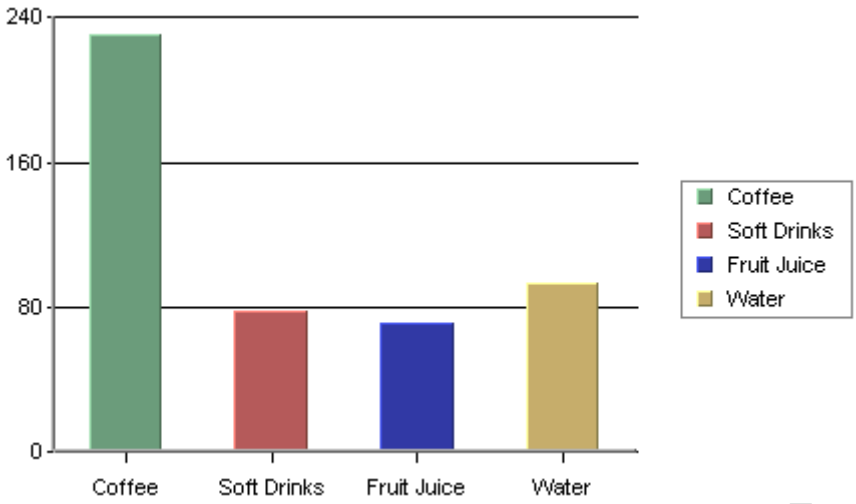


Default Chart Data マッピング

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)のみを転置することができます。1 つのチャートエレメントに対して **Multi Selection** チェックボックスを選択すると、他のチャートエレメントに対して **Multi Selection** チェックボックスが無効になります。

転置はドリルダウンチャートでは使用できません。

7.2 縦棒グラフ

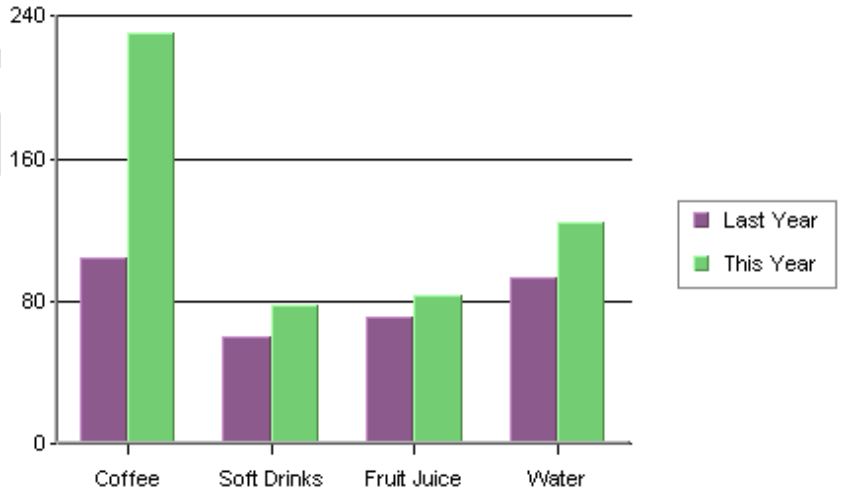


縦棒グラフ(データシリーズ無し)

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。

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。

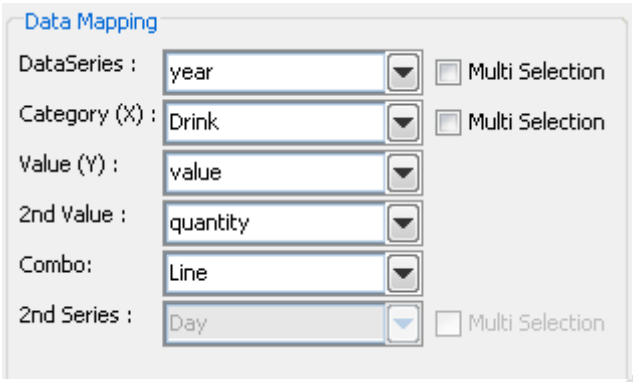
与えられた例では、カテゴリ変数として **Drink** を、値変数として **Value** を選択しました(ほとんどの例では、EspressChart インストールに含まれる **Sample.dat** ファイルが使用されています)。下のグラフのデータシリーズ変数として **Year** が選択されています。したがって、各名前には、データシリーズの列に存在する唯一の 2 つの値である **Last Year** と **This Year** の 2 つの縦棒が表示されます。



縦棒グラフ(データシリーズ有り)

7.2.1 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。基本的な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:
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。

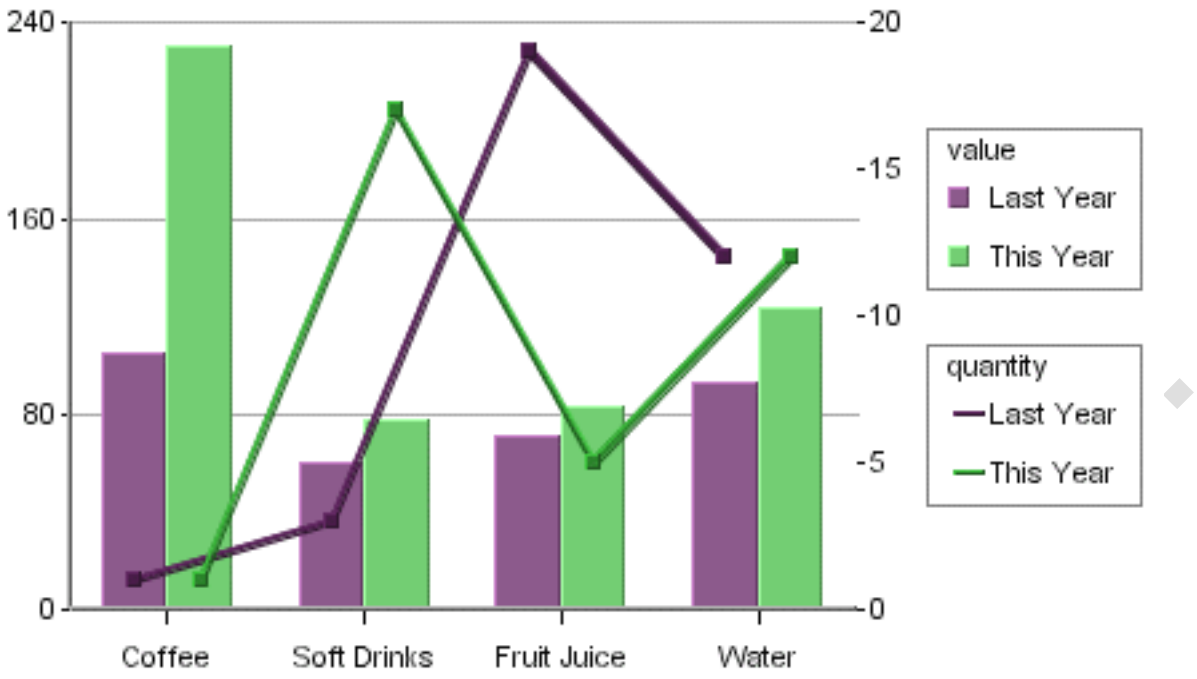
2nd value:
組み合わせチャートを作成するための 2 番目の値を追加します。

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重ね合わせチャートです。

データマッピングでは、データを転置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)。データ転置の詳細については、データ転置を参照してください。

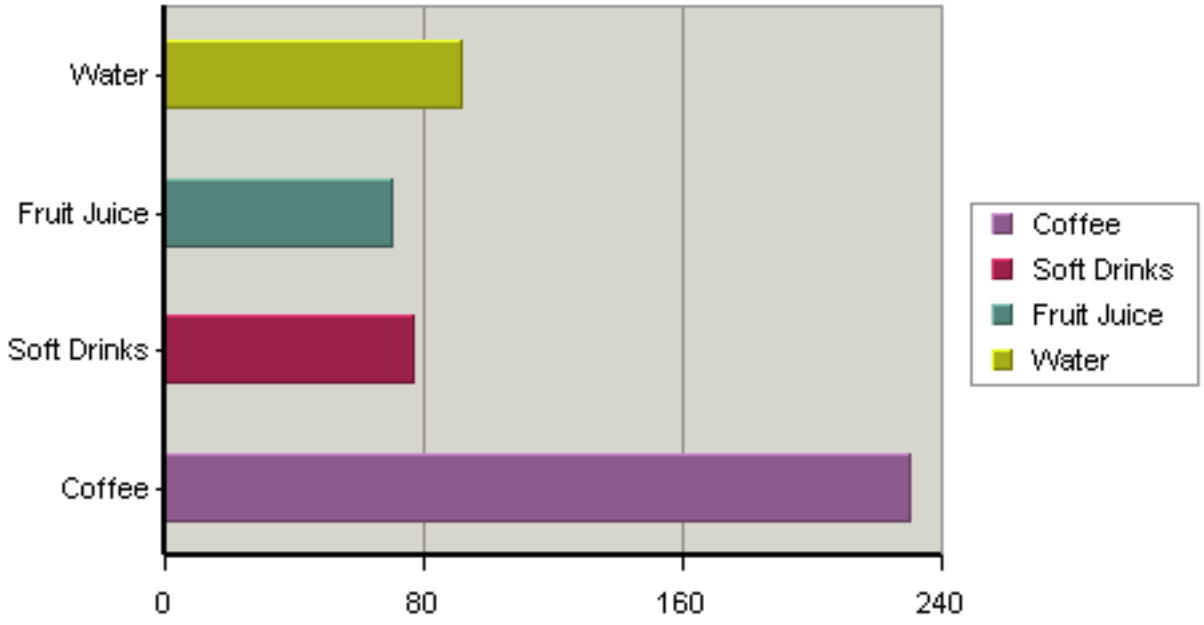
最後の 3 つのデータマッピングオプションを使用すると、グラフに第 2 値を追加できます。EspressChart は、円グラフチャート、レーダーチャート、バブルチャート、ダイヤルチャート、サーフェイスチャート、散布図以外のすべてのグラフタイプで第 2 値をサポートしています。下の例では、第 2 値が縦棒グラフに追加されています。



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、積み上げカラム、積み上げ面グラフ、High Low チャート、HLCO チャート、100% Column 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。重ね合わせチャートの詳細については、重ね合わせチャートを参照してください。

7.3 横棒グラフ

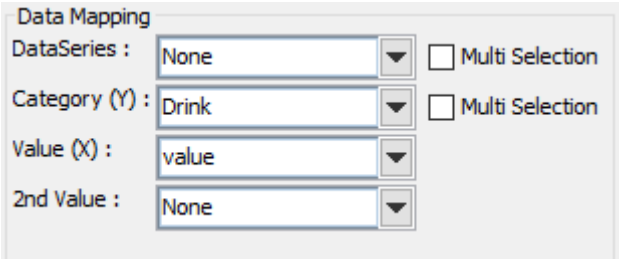


横棒グラフ

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。

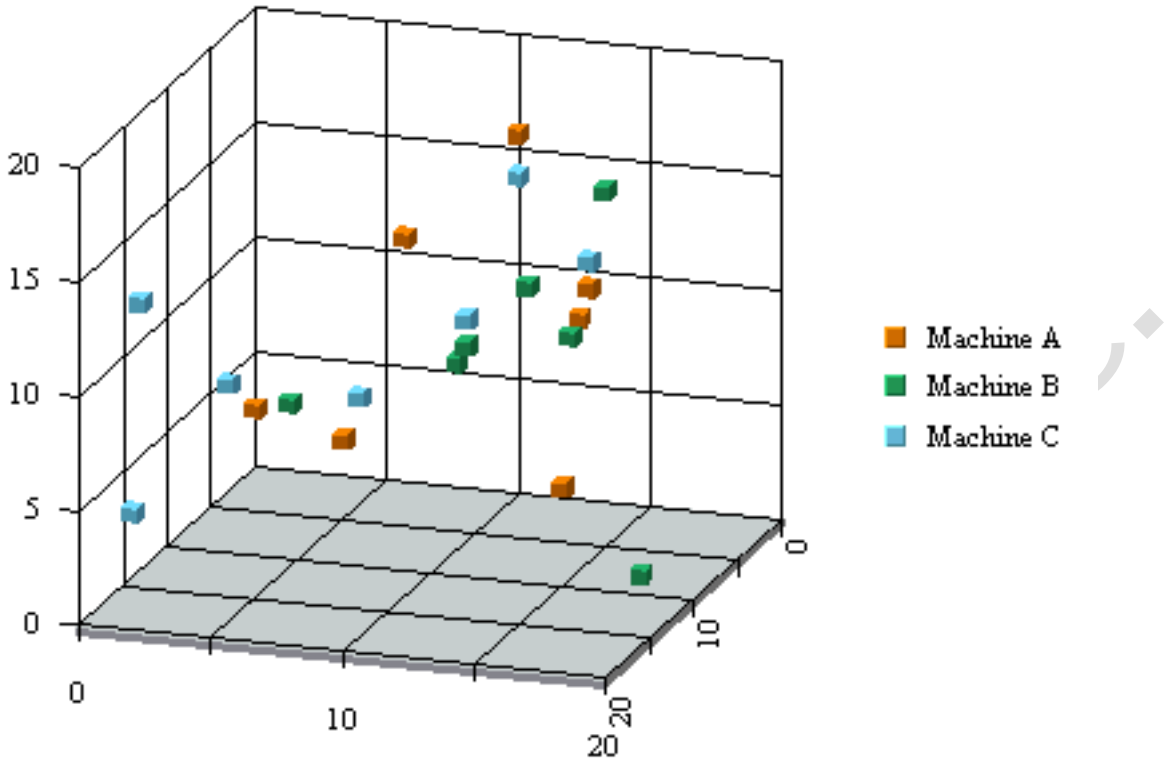
7.3.1 データマッピング



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。

7.4 XY(Z)散布図

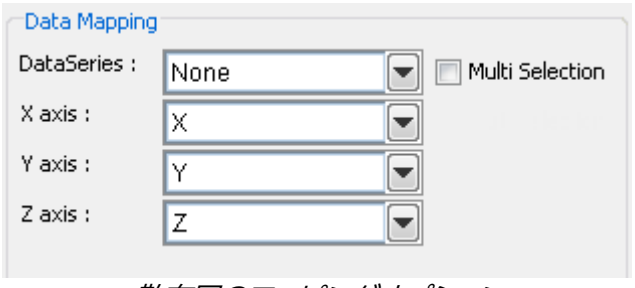


XY(Z)散布図

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。

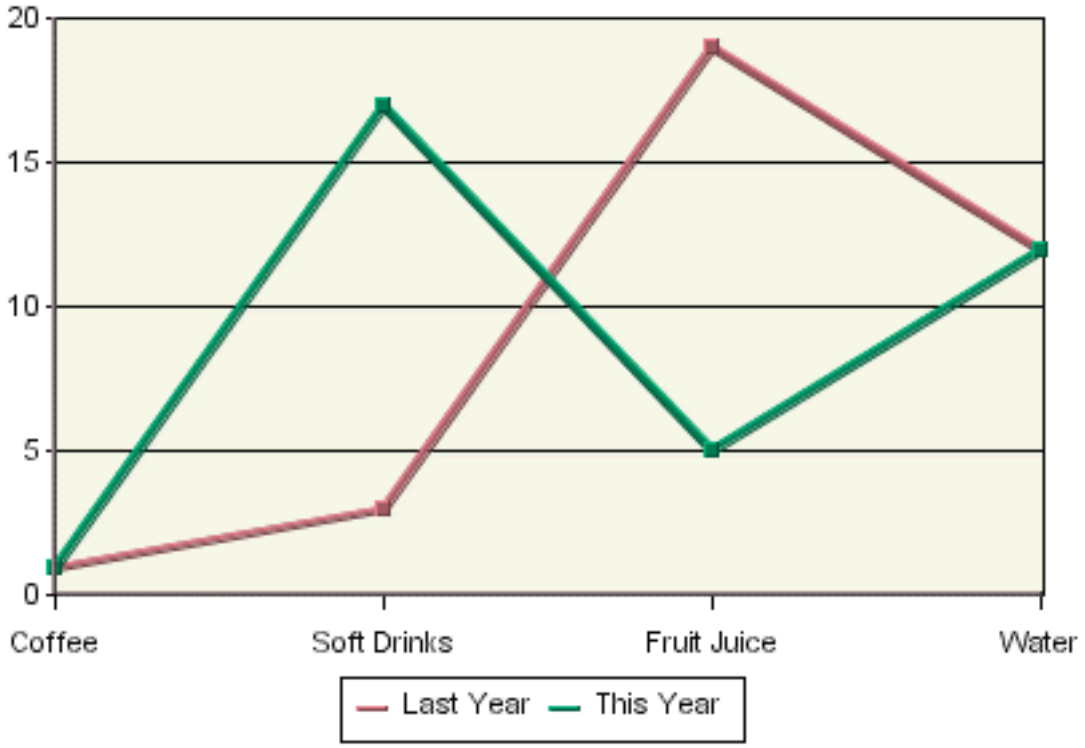
7.4.1 データマッピング



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。

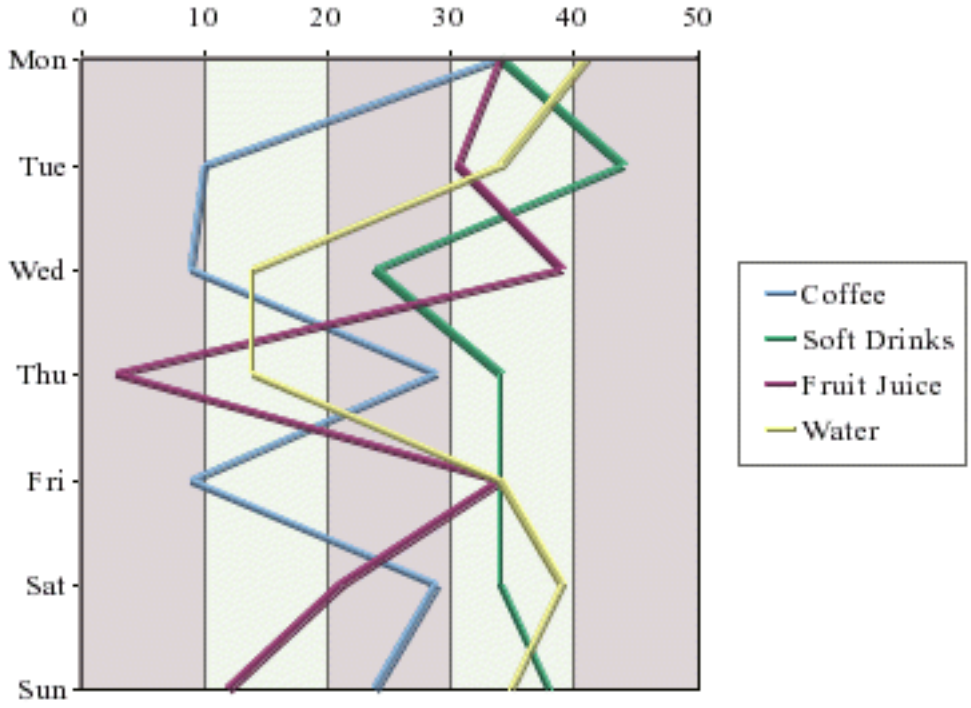
7.5 折れ線グラフ



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。線と点の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。

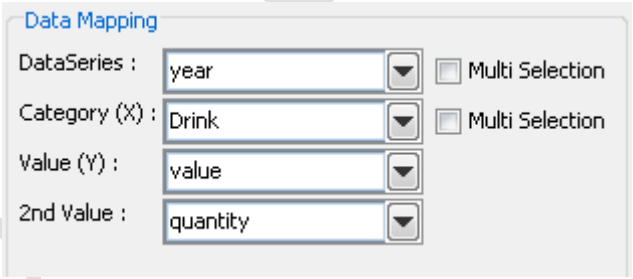
EspressChart では、デフォルトの水平設定に加えて、2D 折れ線グラフを垂直方向に表示することができます。チャートデザイナでこの機能を使用するには、折れ線グラフを作成し、**Format→Chart Options** を選択します。次に、**Vertical** を選択します。チャートは時計回りに 90 度回転します。



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。

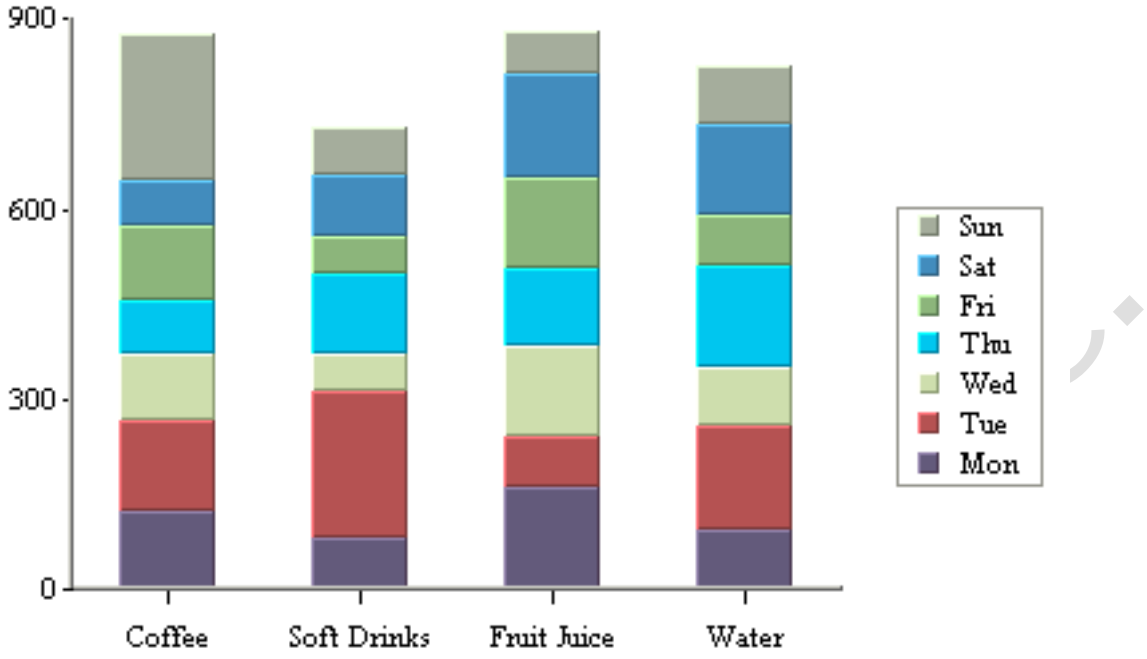
7.5.1 データマッピング



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、積み上げ縦棒など）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。

7.6 積み上げ縦棒グラフ



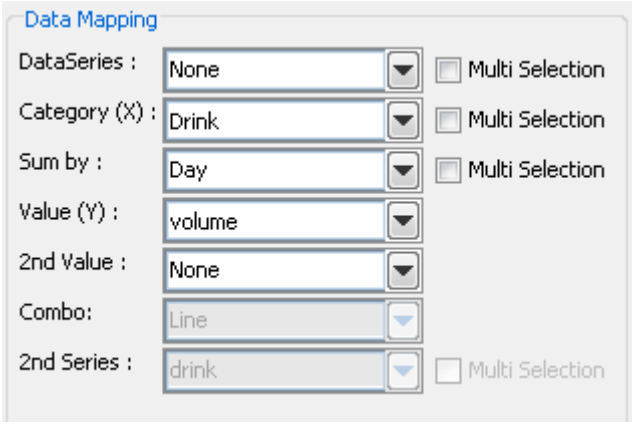
積み上げ棒グラフ

積み上げ縦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 積み上げ縦棒グラフの場合、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 積み上げ縦棒グラフでは、データシリーズが Z 軸に沿って表示されます。

7.6.1 データマッピング

積み上げ縦棒グラフでは、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。



積み上げ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、縦棒の個別の積み上げ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。

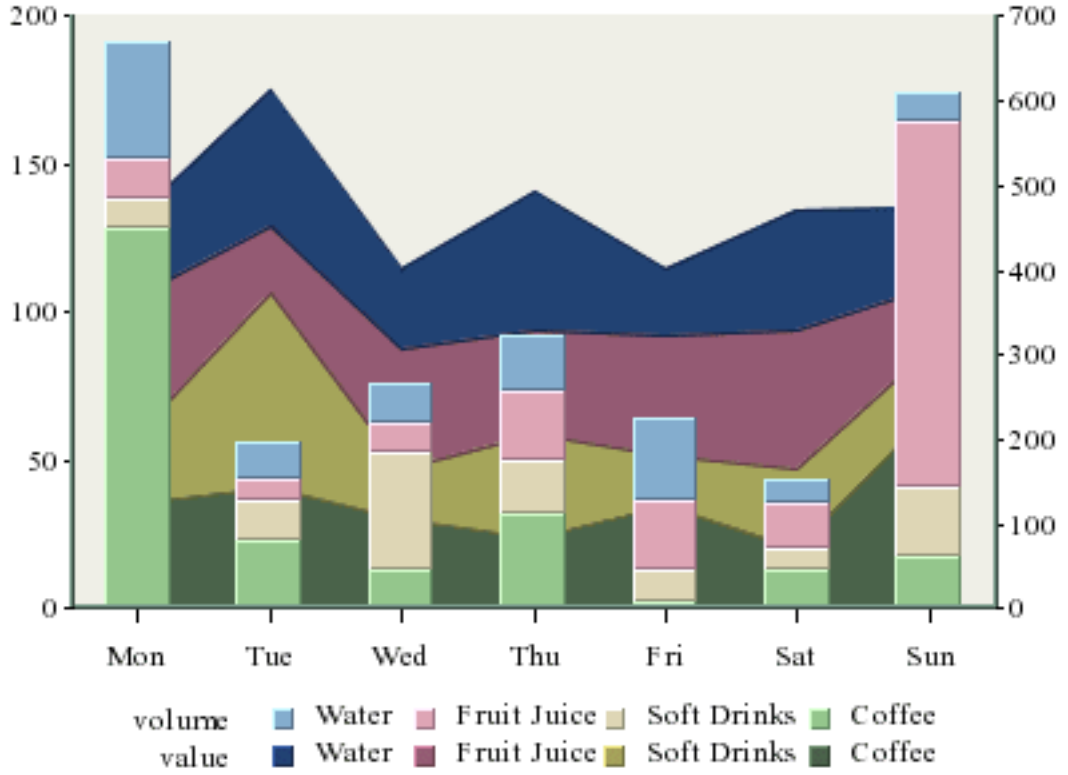
2nd value:
組み合わせチャートを作成するための 2 番目の値を追加します。

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、セカンリチャートが重ね合わせグラフ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。積み上げ縦棒グラフの場合、コンボオプションは折れ線、積み上げエリア、およびオーバレイです。

データマッピングでは、データを転置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）。データ転置の詳細については、データ転置を参照してください。

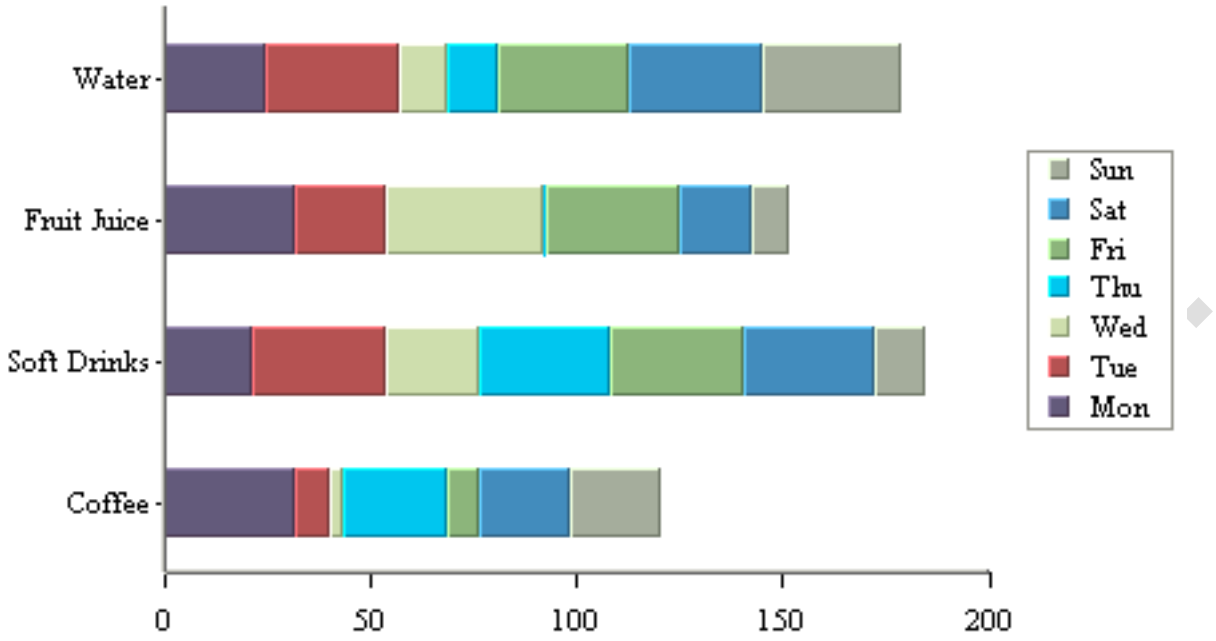
積み上げ縦棒グラフには、ユーザーが積み上げ面積グラフとして第 2 値の値をプロットするためのユニークな組み合わせオプションがあります。両方の値は同じカテゴリと合計値を共有します。このチャートは、データの異なる値に対するコンポーネントベースのカテゴリを比較できます。



積み上げ縦棒 - 積み上げエリアの組み合わせグラフ

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。

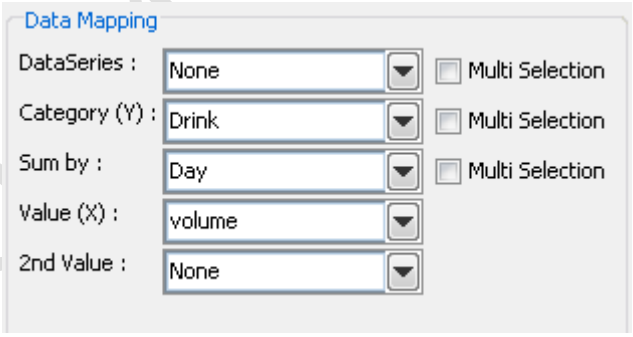
7.7 積み上げ横棒グラフ



積み上げ横棒グラフ

積み上げ縦棒グラフと同様に、積み上げ横棒グラフは、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。積み上げ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。

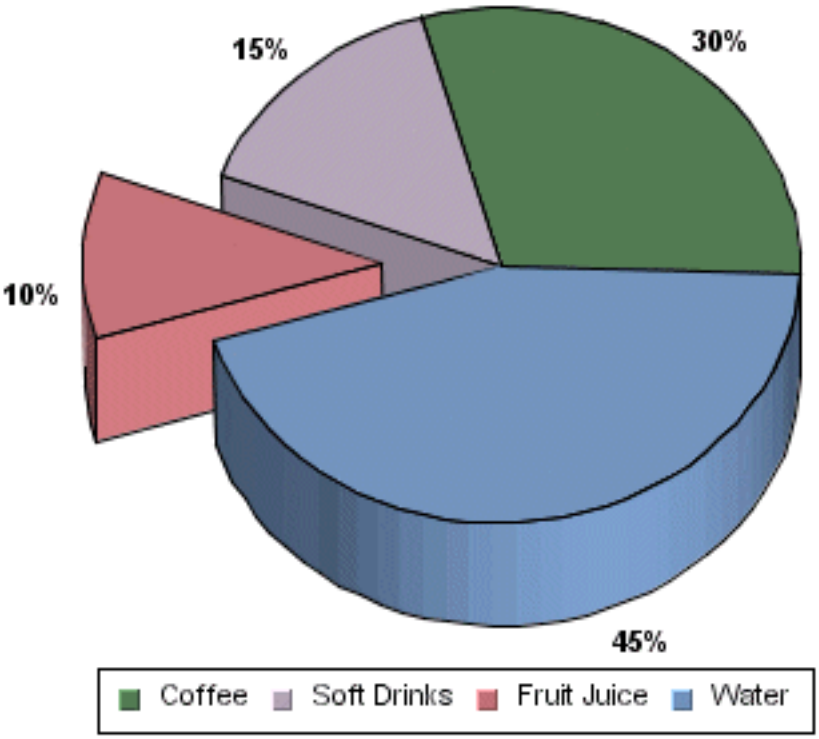
7.7.1 データマッピング



積み上げ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、積み上げ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積み上げ横棒グラフでは使用できません。これは、積み上げ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。

7.8 円グラフ

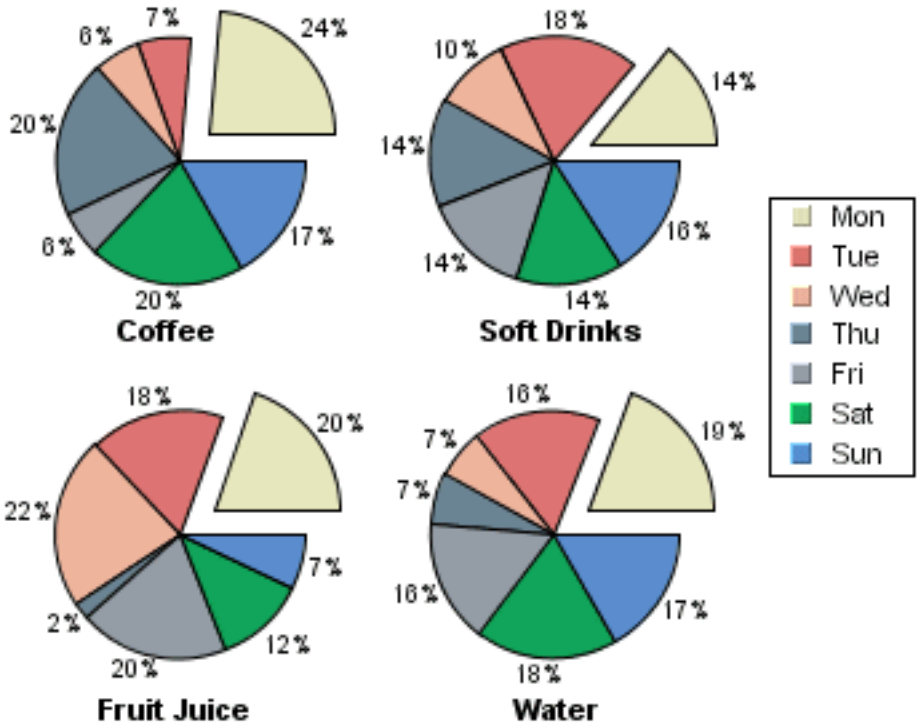


円グラフ

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、余分な情報は含まれません。

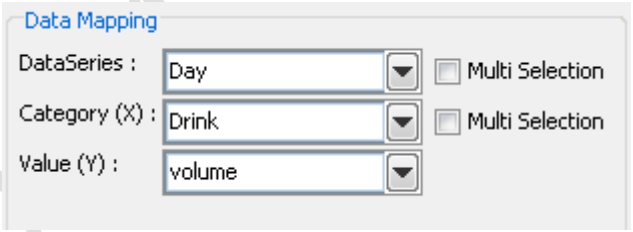
円グラフにはデータシリーズも指定できます。データマッピングでシリーズを指定すると、シリーズ要素で構成される各カテゴリ要素で別々の円が描画されます。データシリーズの円グラフは次のように表示されます。



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。

7.8.1 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。

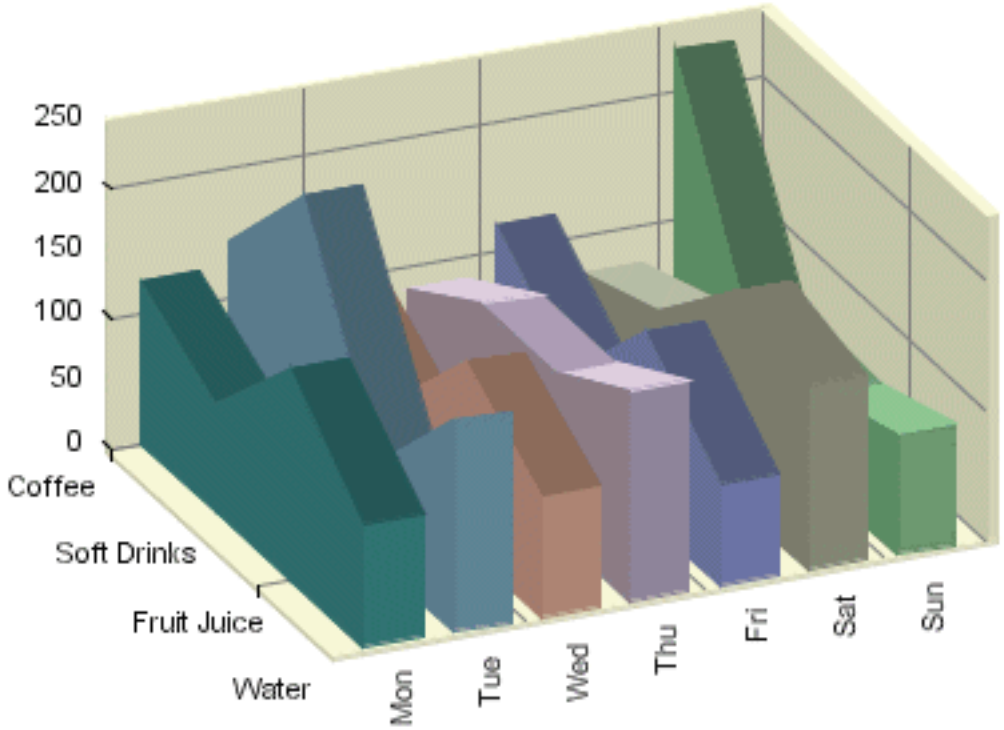
Value (Y):
各カテゴリの値を提供するデータ列を選択できます。

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、データを転置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

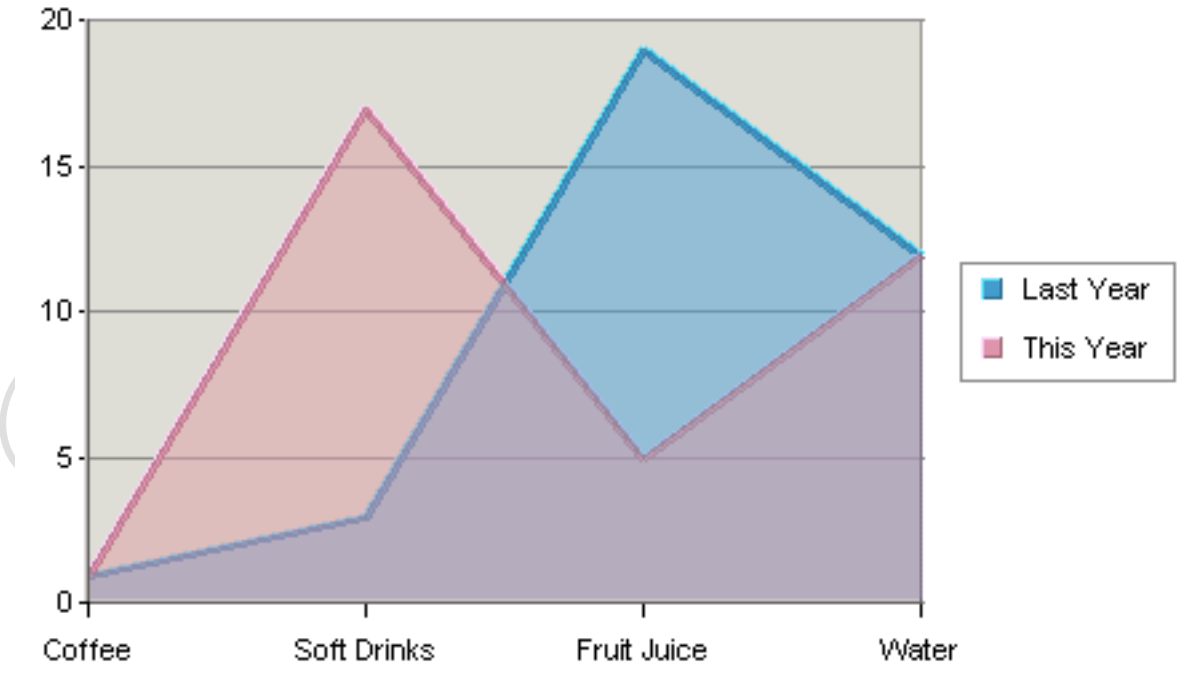
ータ転置の詳細については、データマッピングのデータ転置を参照してください。

7.9 エリアグラフ



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。

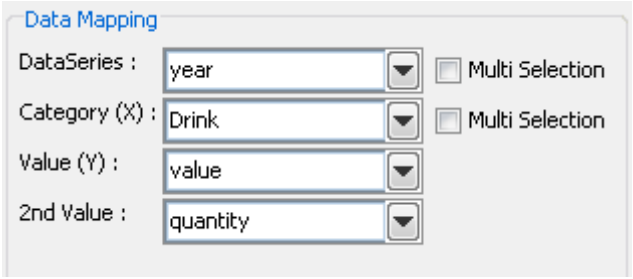


2D エリアグラフ

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(形式メニューを参照)。

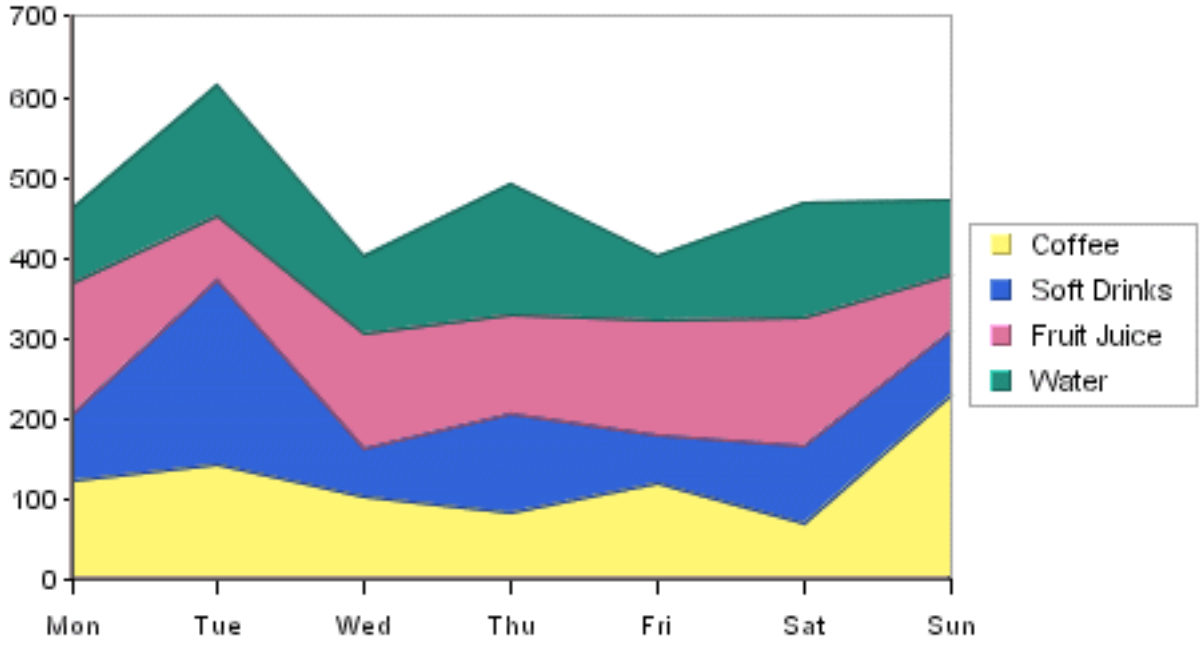
7.9.1 データマッピング



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、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。

7.10 積み上げエリアグラフ



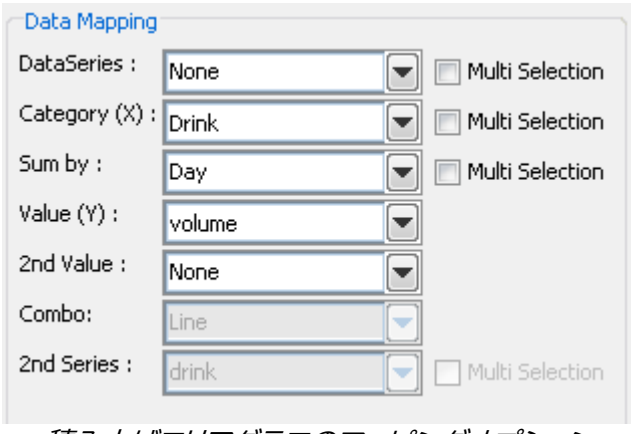
積み上げエリアグラフ

2D 積み上げエリアグラフは、2D 積み重ね縦棒チャートの派生物ですが、データシリーズはありません。このチャートは、次のように積み上げ縦棒グラフから作成できます。同じ色を持つ各積み上げコンポーネントの上端が、線で結合されています。積み重ね縦棒が削除され、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 積み上げエリアグラフは、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 積み重ね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 および 3D の積み上げエリアグラフは、それぞれ、それらのそれぞれの積み重ね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 の積み上げエリアグラフ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。

7.10.1 データマッピング

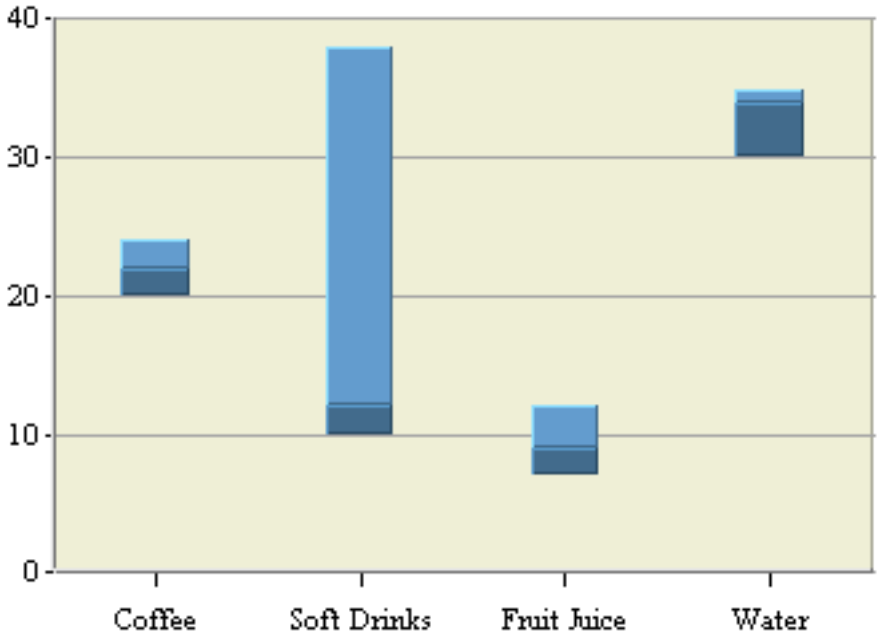


積み上げエリアグラフのマッピングオプション

積み上げエリアグラフのデータマッピングは、積み重ね縦棒チャート(積み上げ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 積み上げ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 **Line** と **Overlay** です。

7.11 HighLow チャート

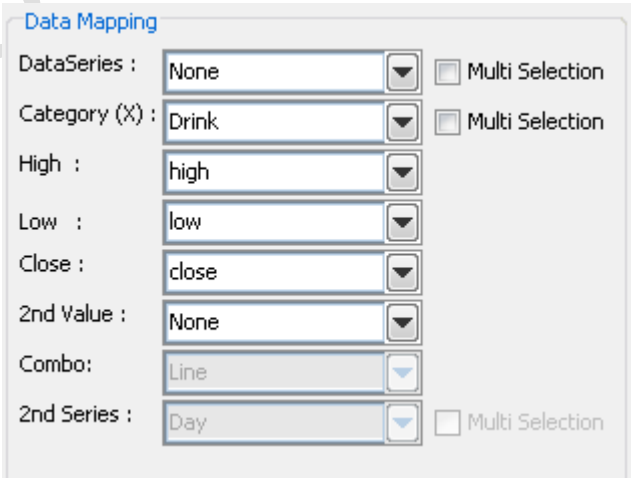


HighLow チャート

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、低値の少なくとも 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る。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。

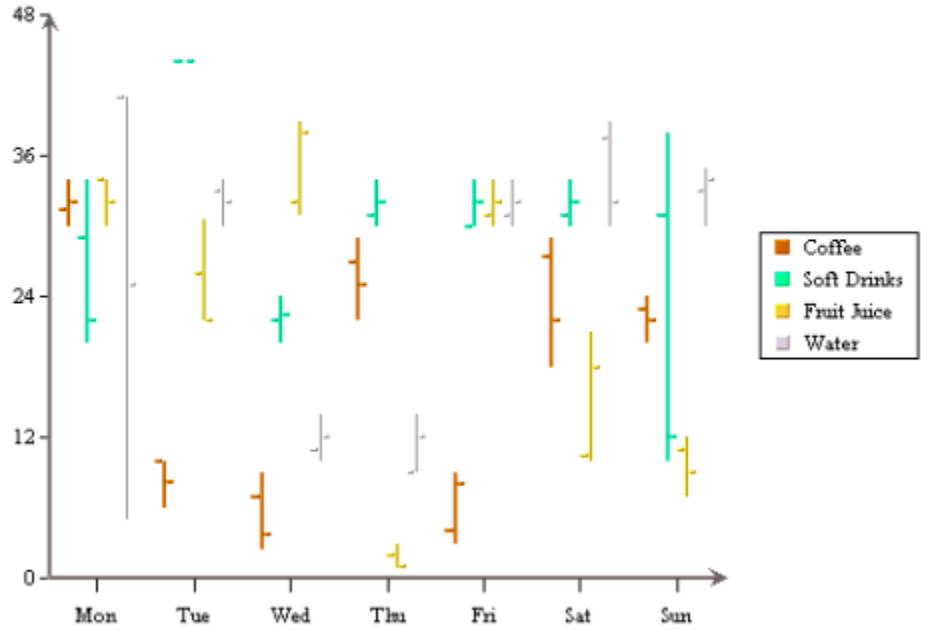
7.11.1 データマッピング



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、高い値と低い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、**Line**、**Column**、および **Overlay** です。

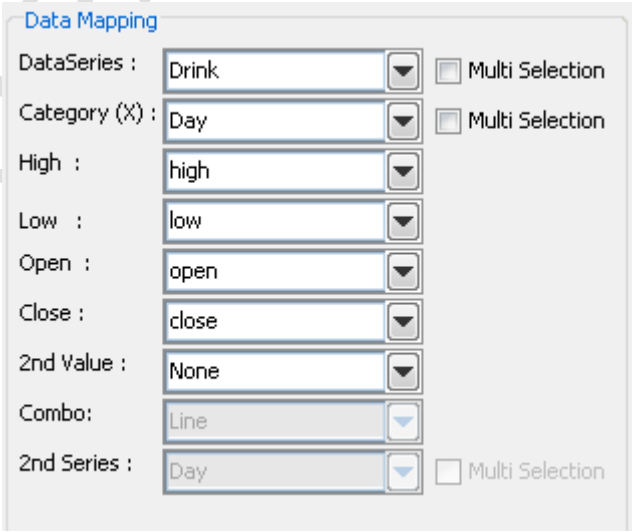
7.12 HLCO チャート



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燭台形式で表示することもできます。この機能については、デザインインターフェイスで説明しています。

7.12.1 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、**Line**、**Column**、**Overlay** があります。

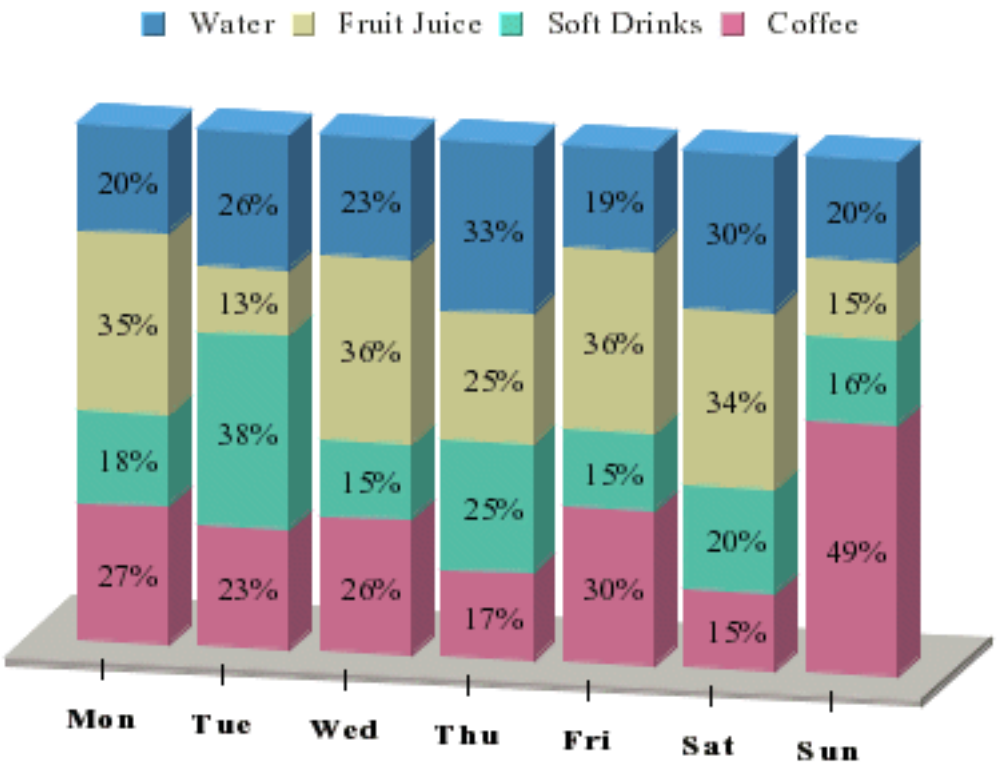
一般的な種類のチャートは、同じチャートで株価と取引量の両方をプロットしたものです。これは、

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

この例では、株価とボリュームデータを 3 か月間にわたってプロットしています。

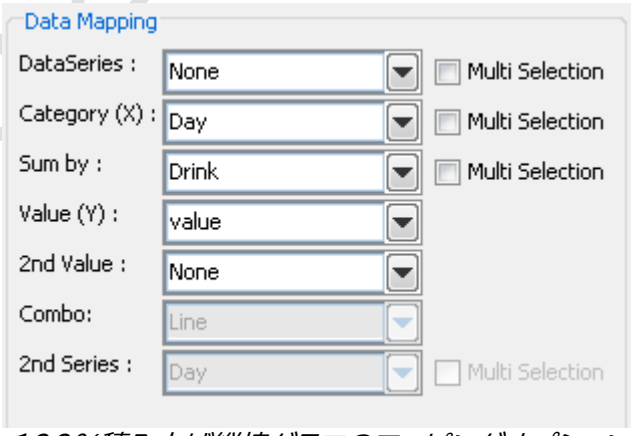
7.13 100%積み上げ縦棒グラフ



100%積み上げ縦棒グラフ

100%積み上げ縦棒グラフは、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。

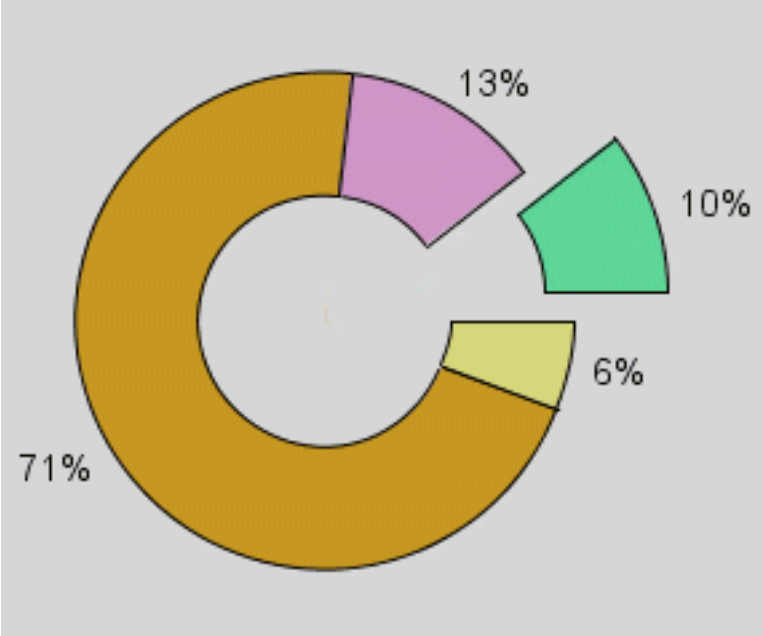
7.13.1 データマッピング



100%積み上げ縦棒グラフのマッピングオプション

パーセンテージカラムチャートのデータマッピングは、積み重ね縦棒チャート(データマッピングでの説明)と同じです。唯一の違いは、合計バイ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%積み上げ縦棒グラフのコンボオプションは、**Line** と **Overlay** です。

7.14 ドーナツグラフ

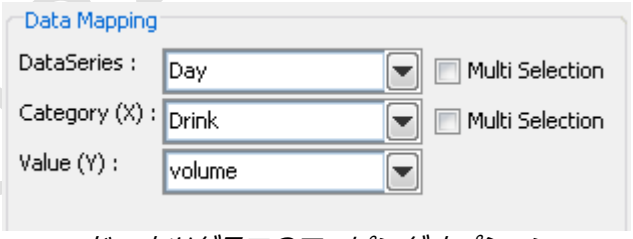


ドーナツグラフ

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。

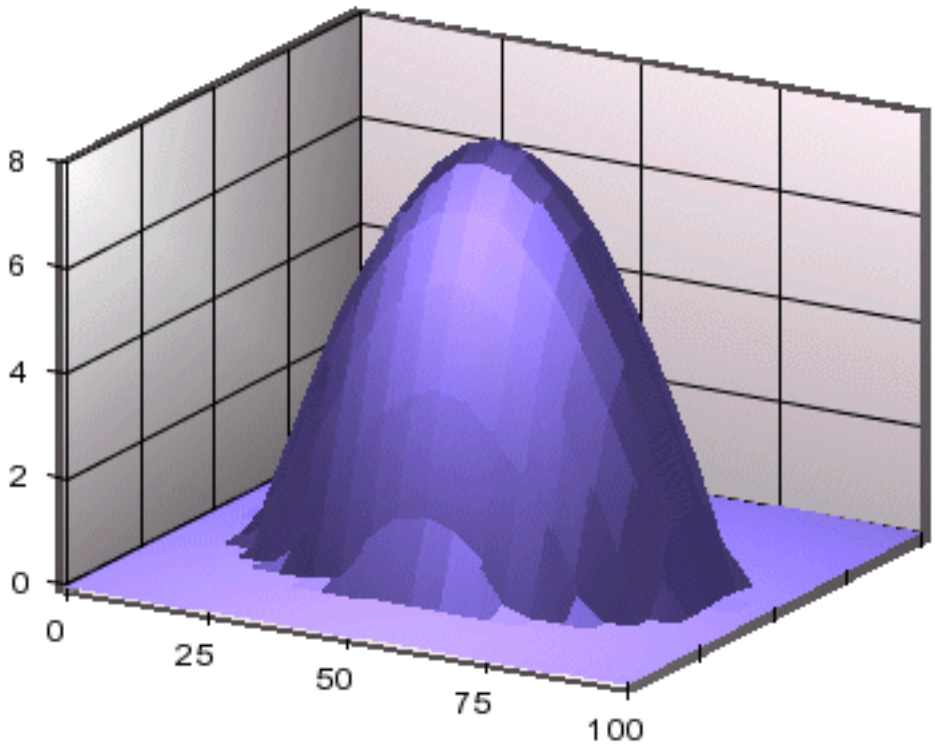
7.14.1 データマッピング



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)。

7.15 サーフェイスグラフ



サーフェイスグラフ

3D サーフェイスグラフは科学/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イスを形成します。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、サーフェイ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イスグラフをプロットすることはできません。

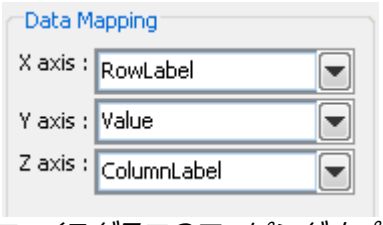
サーフェイスチャートデータのサンプル:

	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

サーフェイスグラフ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。

サーフェイスグラフは、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。サーフェイスグラフ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。

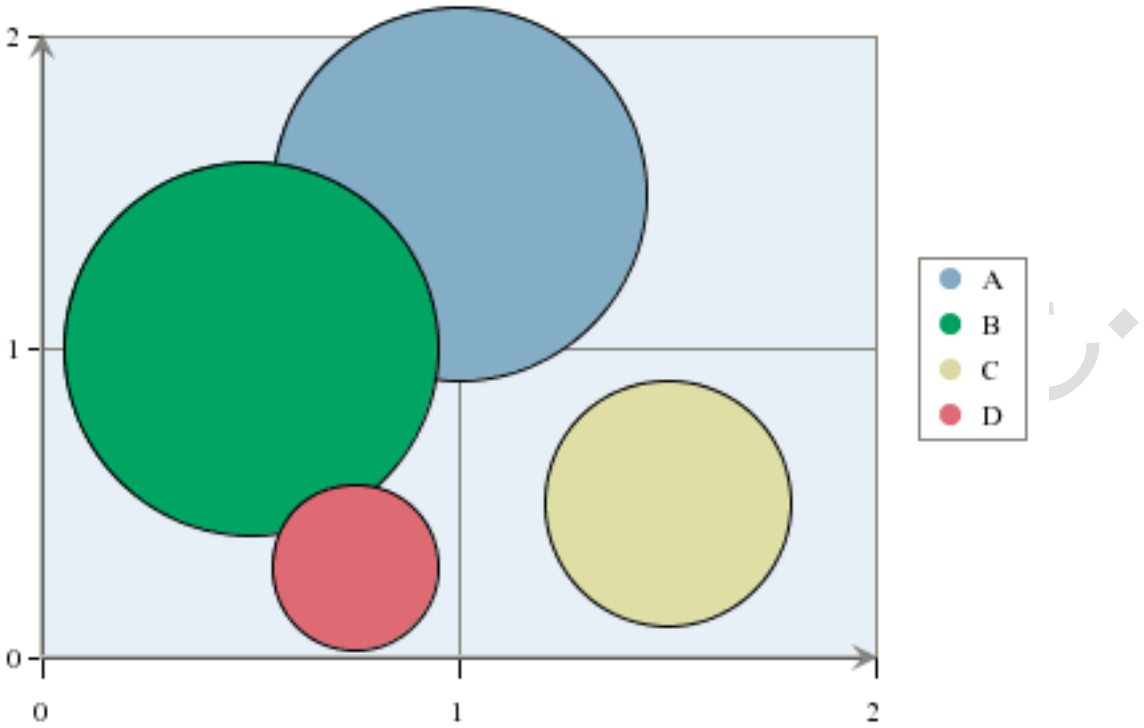
7.15.1 データマッピング



サーフェイスグラフのマッピングオプション

サーフェイスグラフ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、サーフェスグラフ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。

7.16 バブルチャート



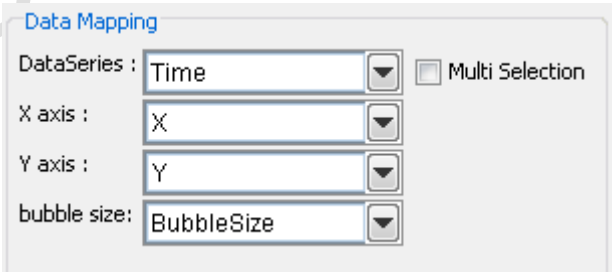
バブルチャート

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。

7.16.1 データマッピング

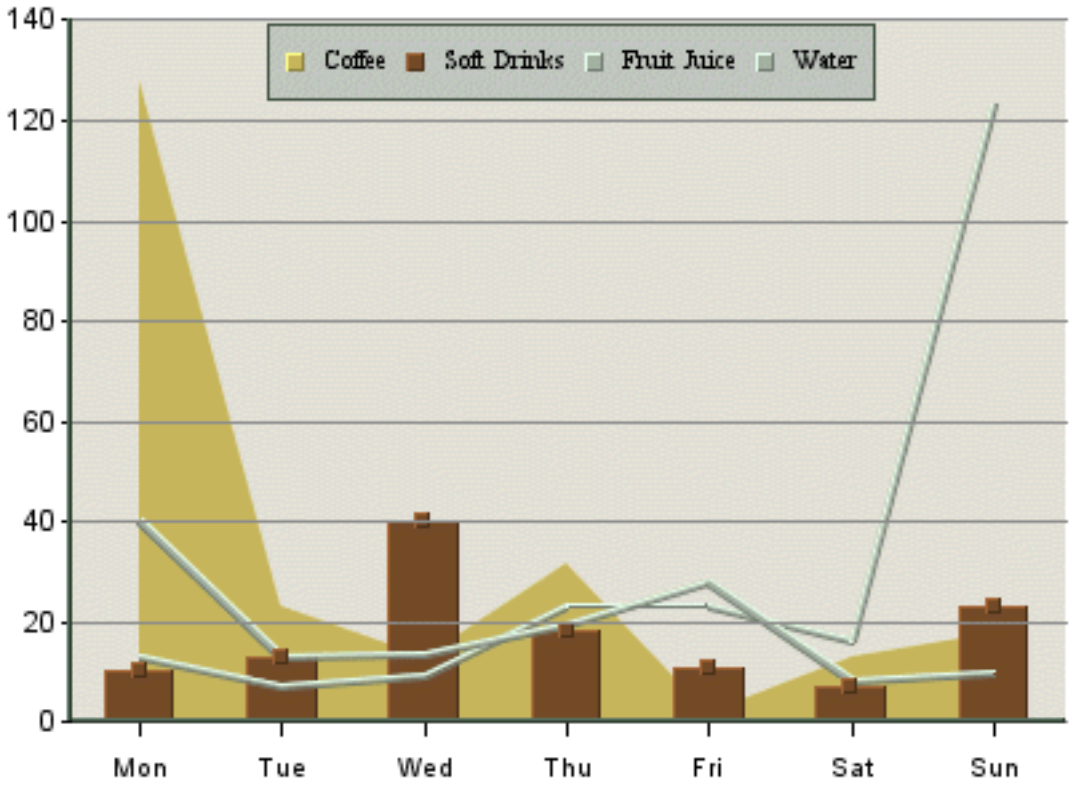
バブルチャートのデータマッピングは、3D 散布図に似ています。ただし、Z 軸の位置をプロットする代わりに、3 番目の値がバブルのサイズ(具体的には半径)を決定します。



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。

7.17 重ね合わせグラフ



重ね合わせグラフ

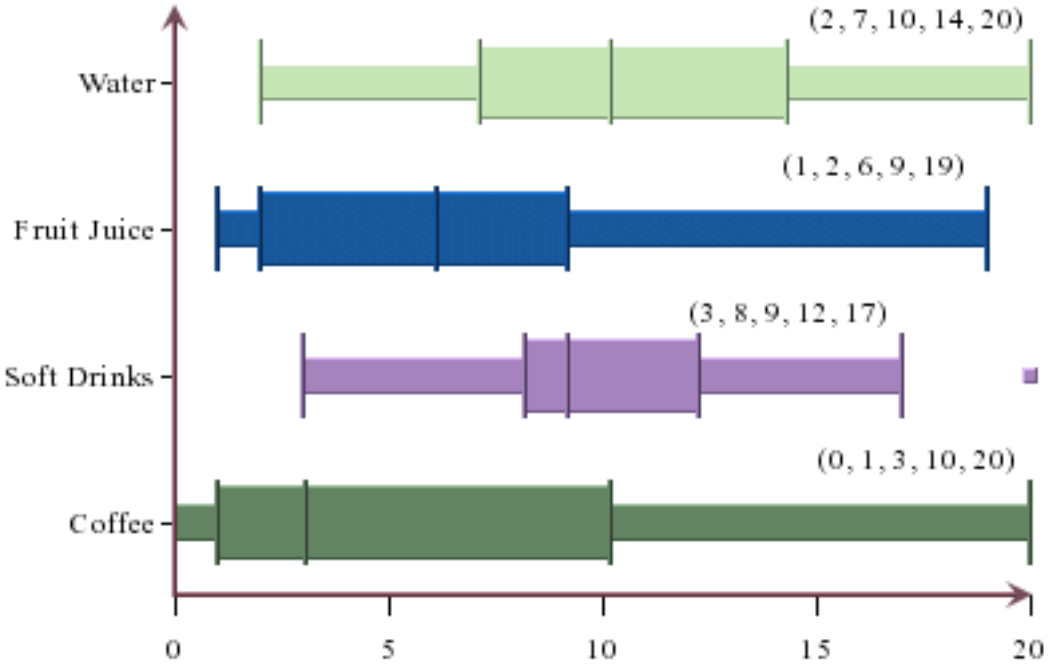
EspressChart は、重ね合わせグラフと呼ばれる特殊なチャートタイプをサポートしています。これにより、共通のカテゴリ軸を持つ 2 つ以上のチャートを重ね合わせることができます。そのため異なるチャートを使用して、データシリーズの各要素を表すことができます。これにより、チャート作成中の自由度が増し、より多くの情報を表現することができます。重ね合わせグラフでサポートされているチャートタイプは、カラムチャート、エリアグラフ、ラインチャートです。重ね合わせグラフには、2D チャートのみを含めることができます。

7.17.1 データマッピング

重ね合わせグラフのマッピングオプション

重ね合わせグラフのデータマッピングは、データチャートの各エレメントを別々のチャートとしてプロットすることを除けば、カラムチャート(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重ね合わせグラフには適用されませんのでご注意ください。重ね合わせグラフは第 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。重ね合わせグラフのオプションをプロットする方法については、重ね合わせグラフを参照してください。

7.18 ボックスチャート

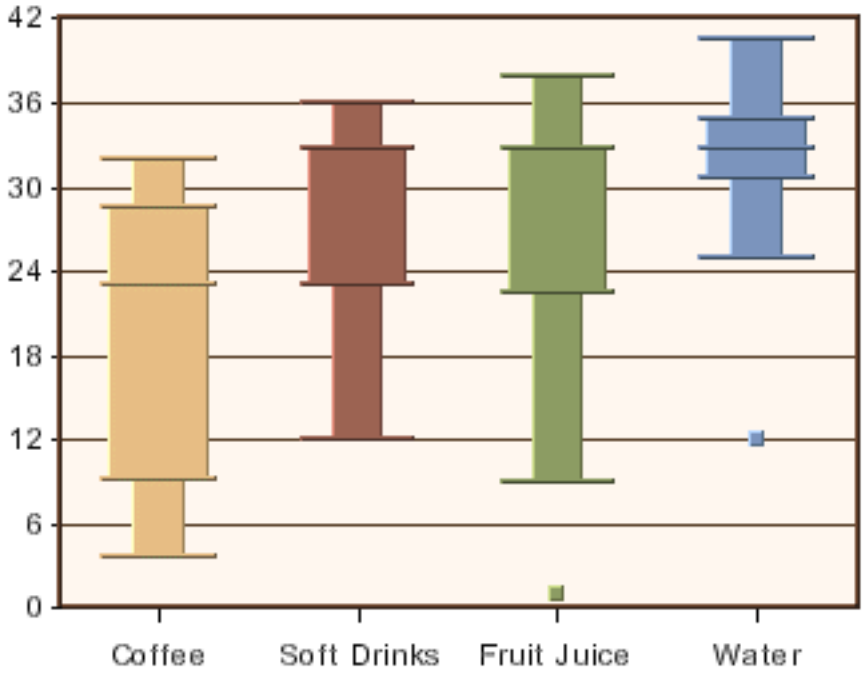


ボックスチャート

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ーを比較するために、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。

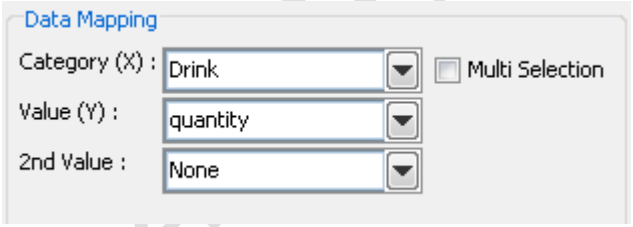
EspressChart では、デフォルトの水平設定に加えてボックスチャートを垂直に表示することができます。チャートデザイナでこの機能を使用するには、ボックスチャートを作成し、**Format** > **Chart Options** を選択し、**Vertical** を選択します。



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。

7.18.1 データマッピング



ボックスグラフ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。

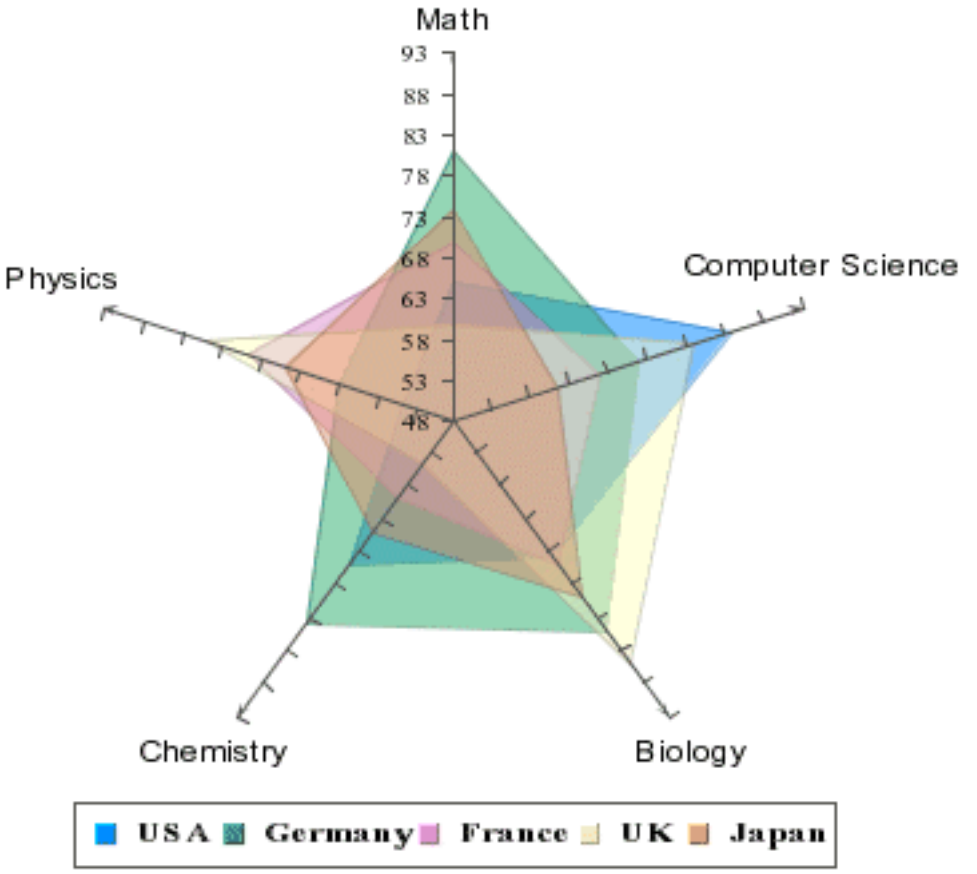
Value (Y):
各カテゴリの値を提供するデータ列を選択できます。これらの値は、最小データポイント、25 パーセンタイル、中央値、75 パーセンタイル、および最大データポイントを計算するために使用されます。

2nd value:
組み合わせグラフを作成するための 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。

データマッピングでは、データを転置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)。データ転置の詳細については、データ転置を参照してください。

7.19 レーダーチャート

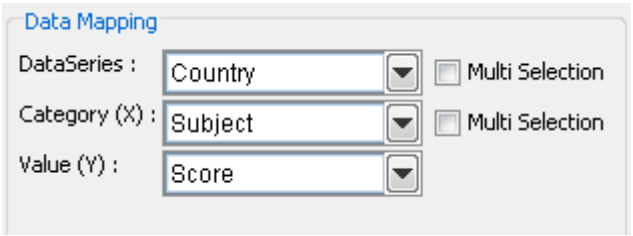


レーダーチャート

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。

このグラフは、2D でのみ利用できます。

7.19.1 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。

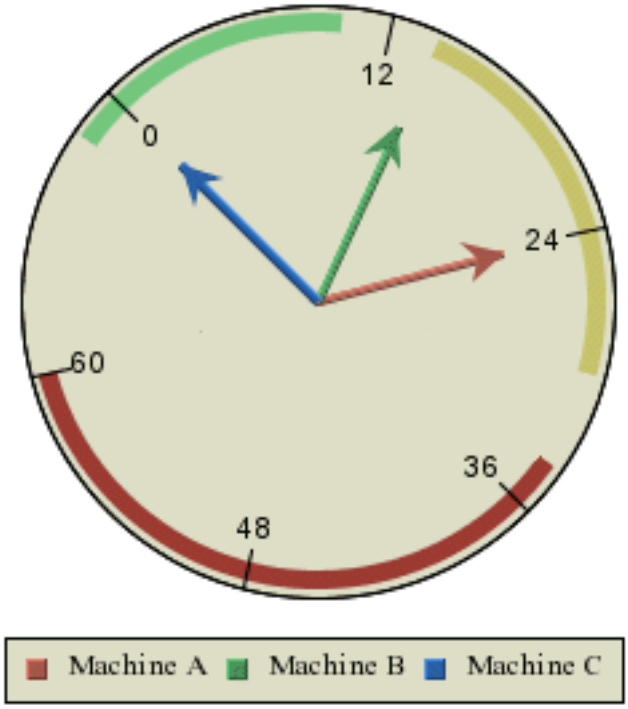
Category (X):
異なる値によってカテゴリが決定されるデータ列を選択できます。この列の固有の要素の数によって、レーダーチャート内の軸の数が決まります。

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、データを転置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)。データ転置の詳細については、データ転置を参照してください。

7.20 ダイヤルチャート

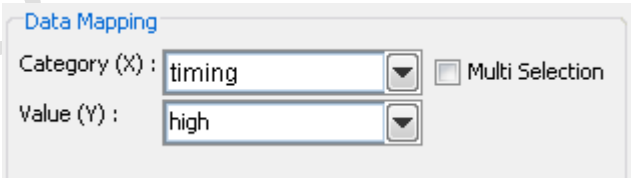


ダイヤルチャート

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ー操作をサポートしています。

このグラフタイプは、2D でのみ使用できます。

7.20.1 データマッピング

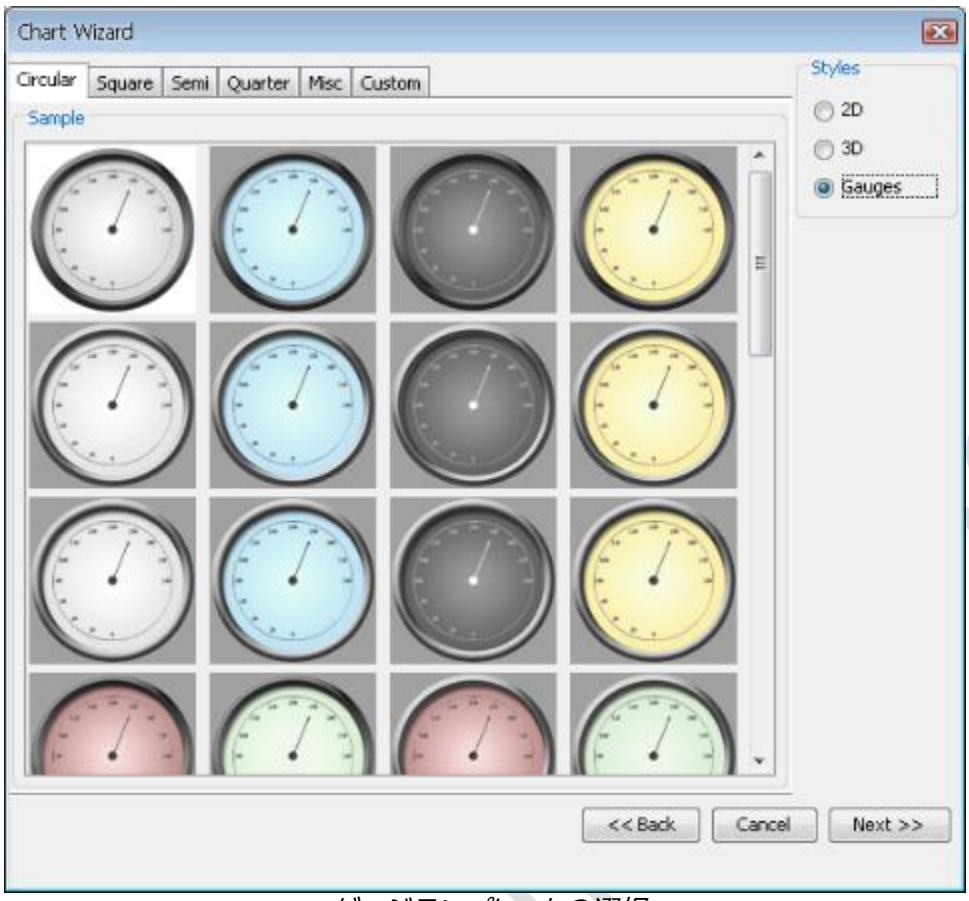


ダイヤルチャートのマッピングオプション

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。

7.20.2 ゲージ

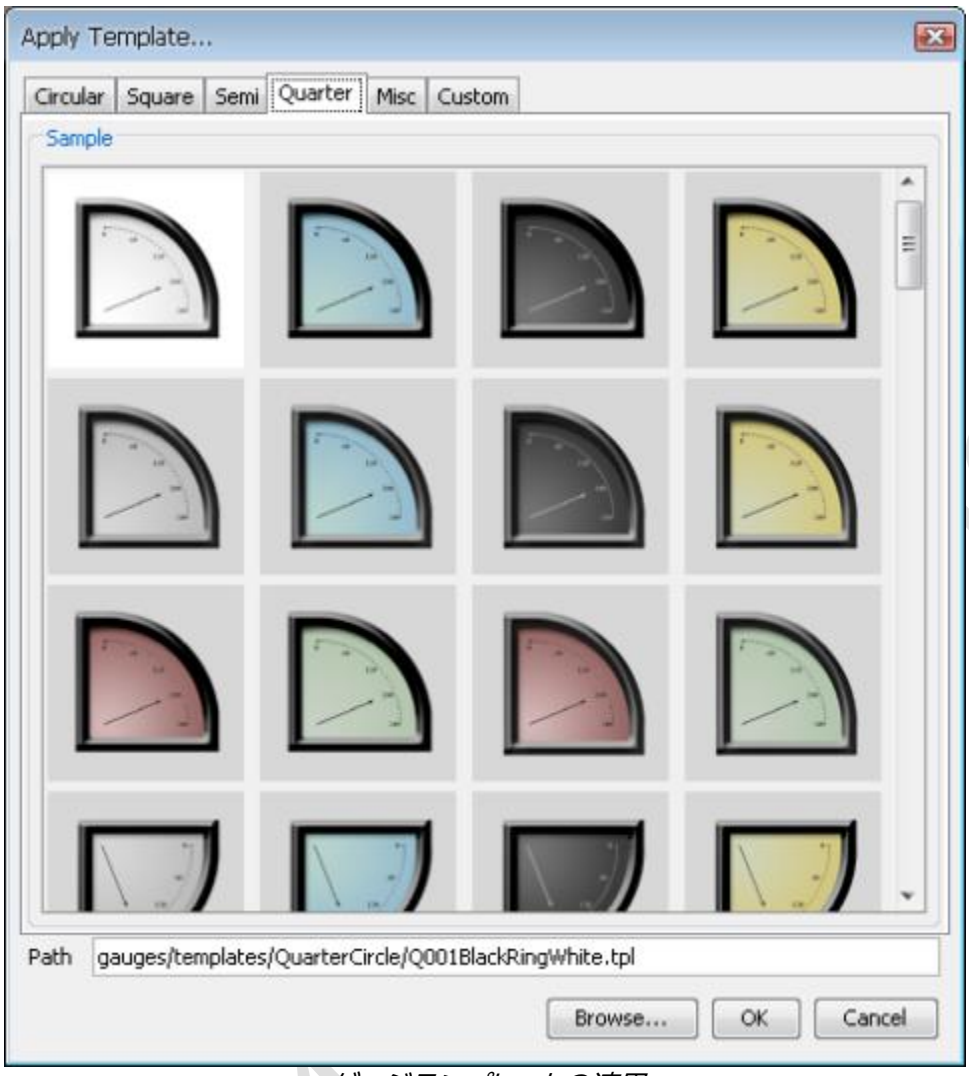
ゲージは、プロット背景画像またはプロットフォアグラウンド画像を含む特別なタイプのダイヤルチャートです。新しいゲージを作成するには、**Chart Type Selection** ダイアログで **Gauges** ラジオボタンを選択します。



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。

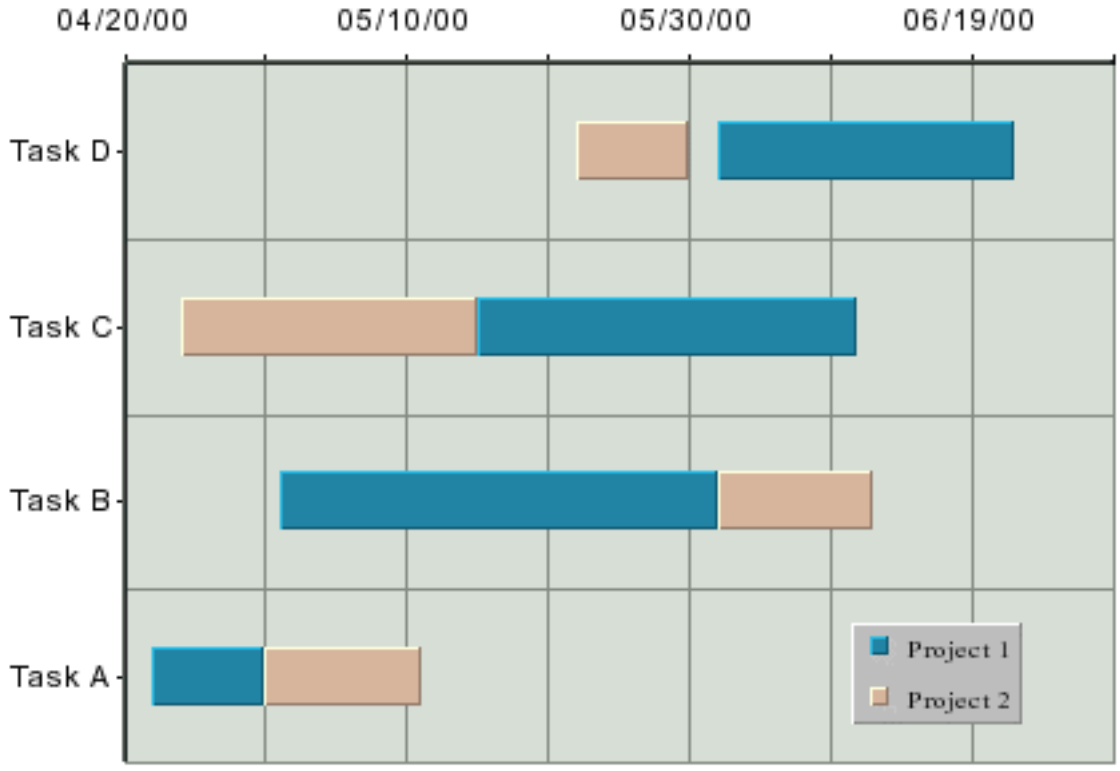
既存のダイヤルチャートにゲージテンプレートを適用することもできます。現在のチャートがダイヤルチャートのときに適用テンプレートを選択すると、新しいチャートを作成するときと同様にゲージタブが表示されます。



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。

7.21 ガントチャート



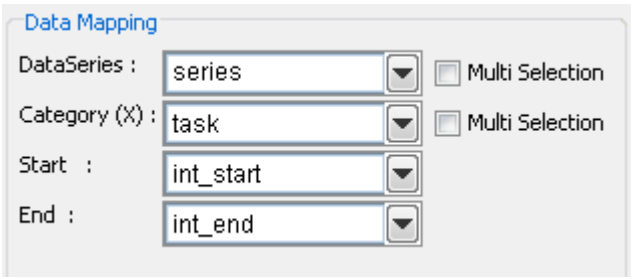
ガントチャート

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。

このグラフタイプは、2D フォームでのみ使用できます。

7.21.1 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

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。

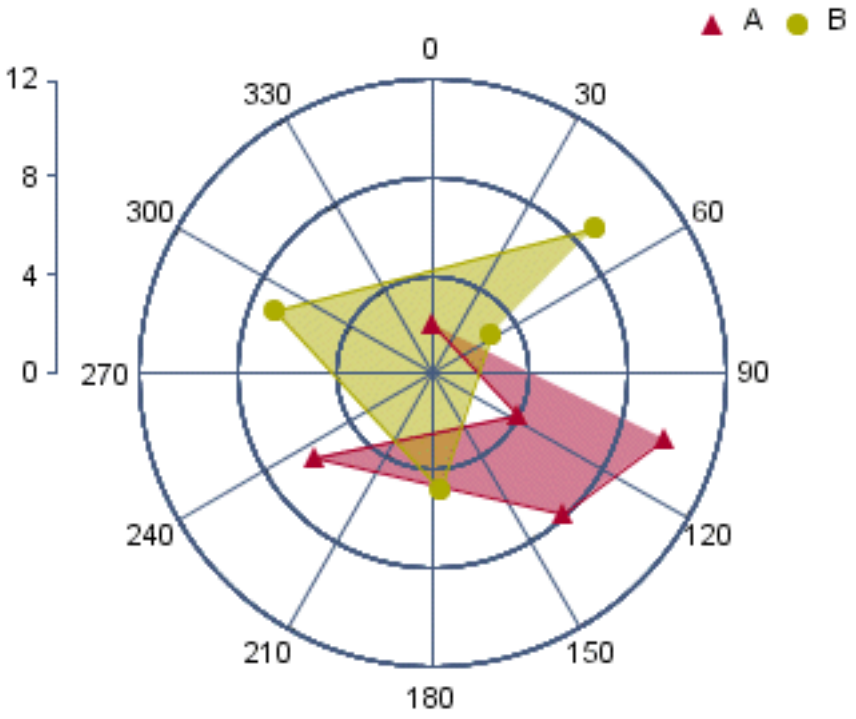
Category (X):
異なる値でカテゴリを決定するデータ列を選択できます。

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、データを転置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)。データ転位の詳細については、データ転置を参照してください。

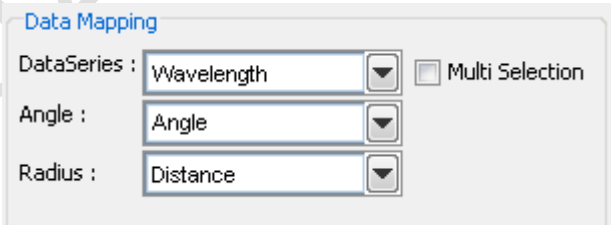
7.22 極座標グラフ



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。

7.22.1 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。

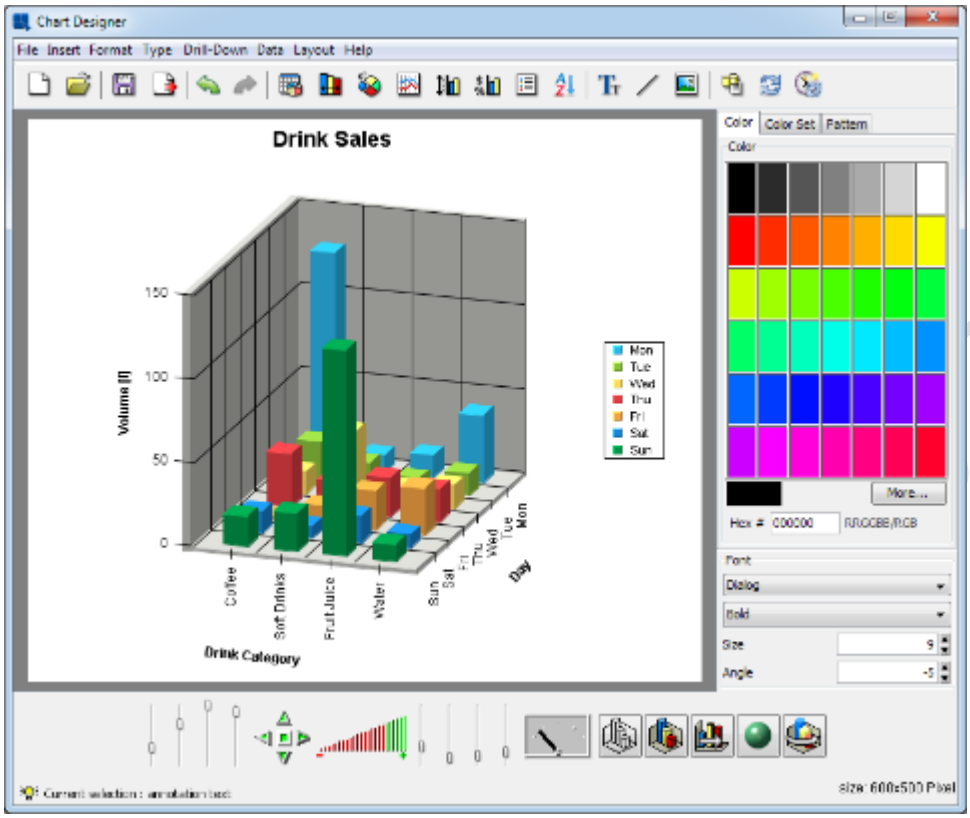
7.23 データマッピングまたはデータソースの変更

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 インターフェイスに移動します。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 が開きます。

8 デザインインターフェイス

ウィザードのすべての手順を完了すると、メインの Chart Designer インターフェイスが表示されます。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。



8.1 デザイナメニュー

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。

8.1.1 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、グラフの保存と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。

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。

8.1.2 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、テキストの追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。

8.1.3 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。

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。

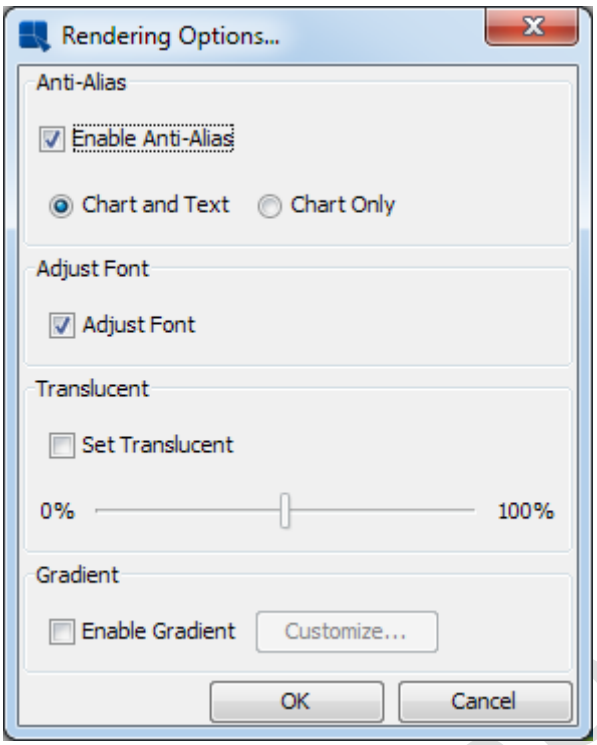
Histogram Options:
チャートをヒストグラムとして描画するかどうかを指定し、頻度カウントを指定するための追加オプションを表示することができます。

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。



Rendering Options ダイアログ

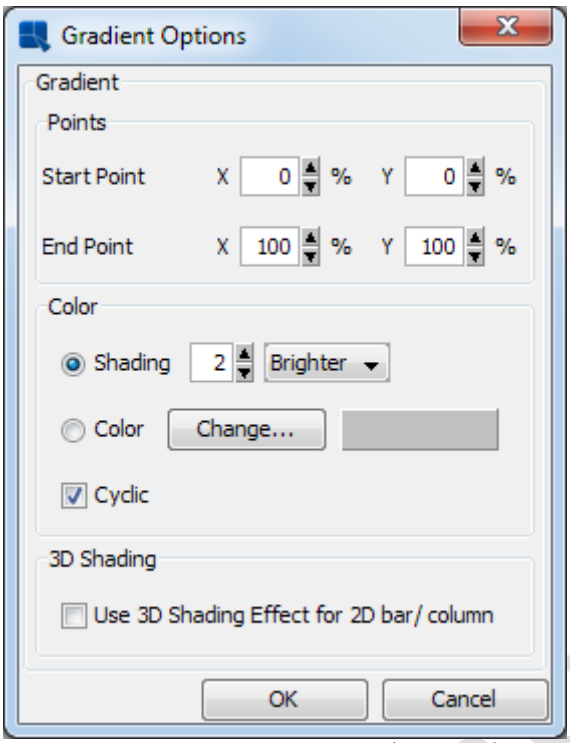
さまざまなレンダリングオプションを指定できます。

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。



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、積み上げ縦棒、積み上げ横棒、100%縦棒、重ね合わせ)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。

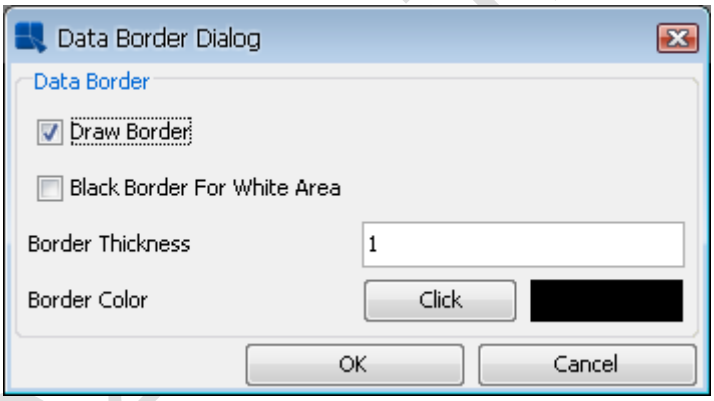
Chart Options:
使用しているチャートタイプの特定のオプションが表示されます。オプションはチャートの種類によって異なります。

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。

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。

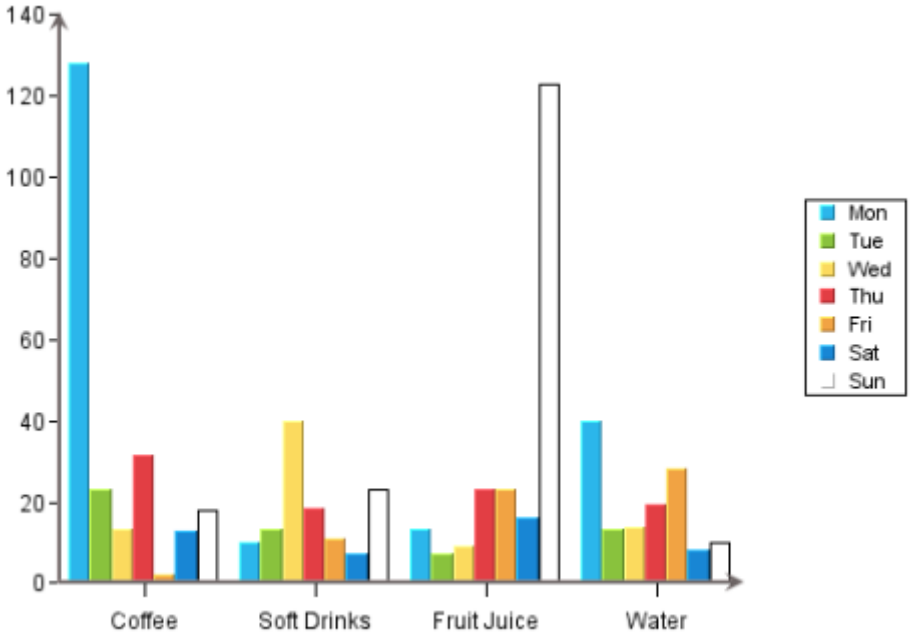
Data Border:
データ要素(縦棒、横棒など)に枠線を追加します。



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。



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、サーフェイスチャート、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。

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。

8.1.4 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。

8.1.5 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。

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。

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:
パラメータードリルダウン ナビゲーションウィンドウが表示され、パラメーター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。

8.1.6 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、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して、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。

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。

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。

8.1.7 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:
EspressChart のユーザガイドが開きます。

8.1.8 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ーフェイ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。

8.2 デザイナツールバー

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。

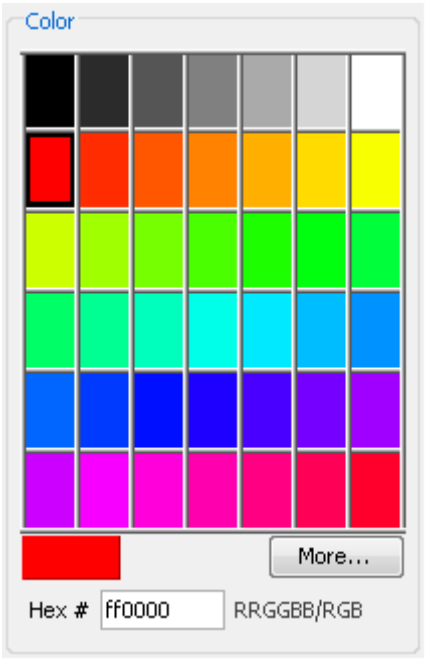
アイコン	説明
	新規チャートを開始します。
	既存のチャートを開きます。
	現在のチャートを保存します。
	現在のチャートをエクスポートします。
	最後の操作を取り消します。
	取り消した操作をもう一度行います。
	現在のチャートのデータマッピングを変更します。
	データ表示プロパティを変更します。
	チャート特有のオプションを変更します。
	ラインとポイントの属性を変更します。
	軸のスケールを変更します。
	軸要素/表示プロパティを変更します。
	凡例表示を変更します。
	データの表示順序を変更します。
	注釈のテキストを挿入します。
	フロートラインを挿入します。
	背景画像を追加または変更します。
	チャートデータソースを変更します。
	チャートデータを更新します。
	定期的なデータの更新をスケジュールします。

8.3 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。

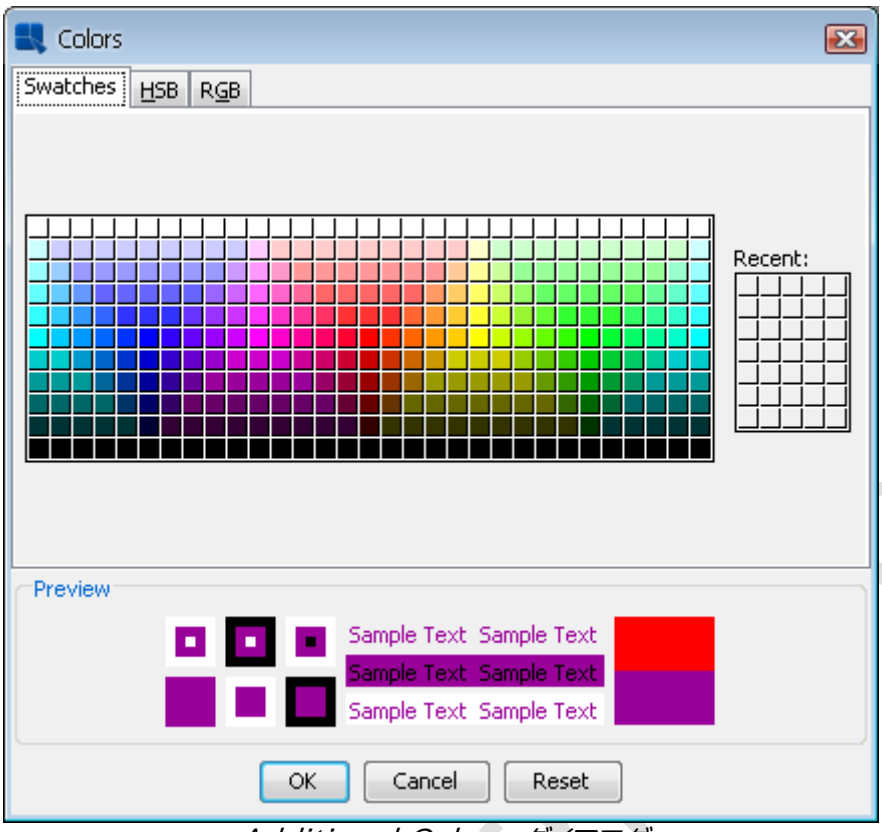
8.3.1 カラーパネル

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。



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。

8.3.2 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。

8.3.2.1 カテゴリに対する色設定の保存機能

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(この機能は積み上げ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。

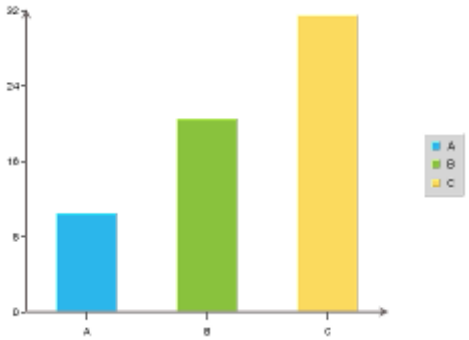


図 1 チャートの例

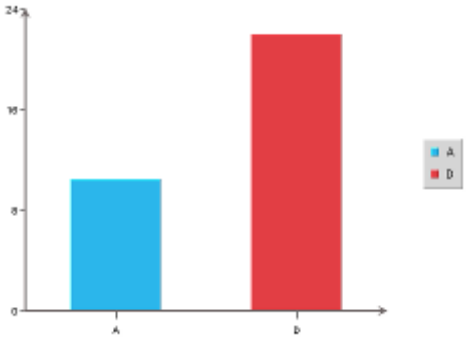


図 2 Save Colors for Categories 機能がオンのとき

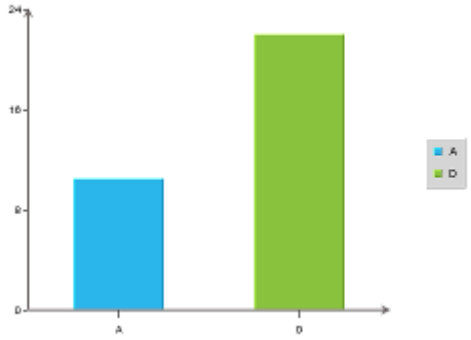


図 3 Save Colors for Categories 機能がオフのとき

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。

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。

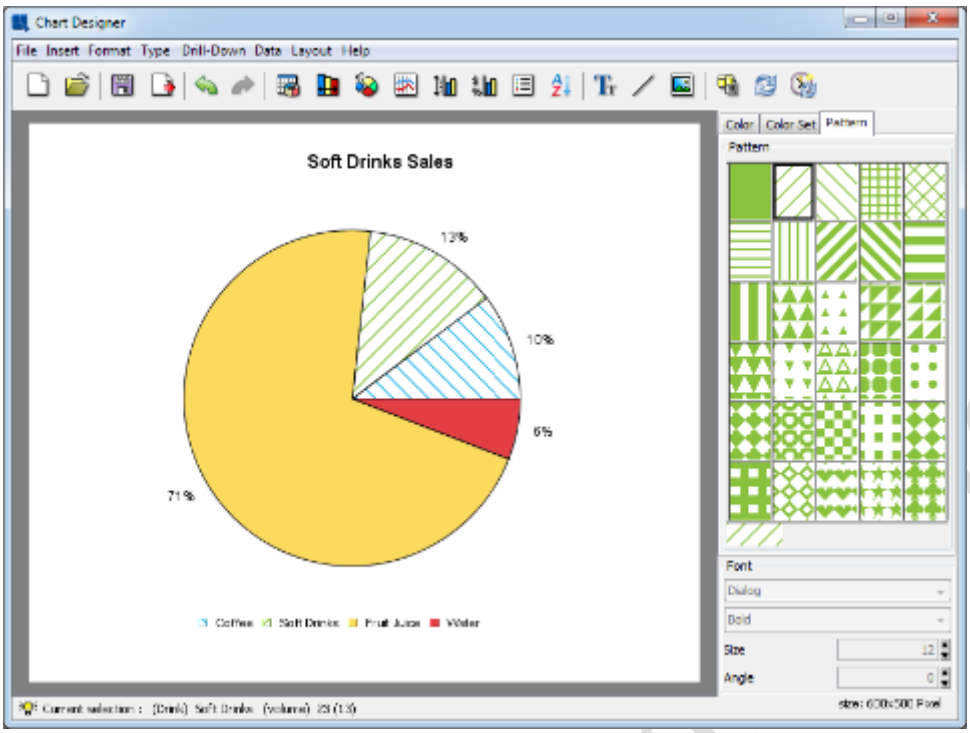
カテゴリ A ...青色(カテゴリ A は、カラーセットから最初に利用可能な色であるため、青色を有する)カテゴリ D ...緑色(カテゴリ D は、最初にカテゴリ A が割り当てられているため、カラーセットの 2 番目の色を持ちます)

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。

8.3.3 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。

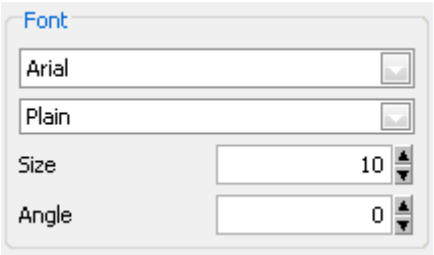
データポイントに対してパターンがすでに定義されている場合でも、ユーザーはカラーパネルタブを選択し、カラーパレットから別のカラーを選択することによって、カラーを変更することができます。パターンの色はすぐに新しい色に変更されます。パターンパレットに表示されるパターンも新しい色に変更されます。



パターン例

8.3.4 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

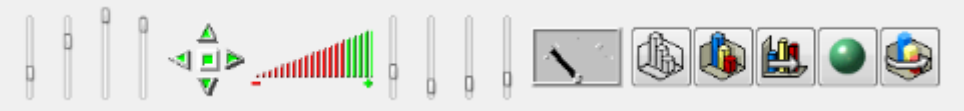
最初のドロップダウンボックスでは、使用するフォントを選択できます。2 番目のドロップダウンボックスでは、フォントスタイルをプレーン、ボールド、イタリック、太字のいずれかで選択できます。 3 番目のボックスでは、フォントサイズを選択できます。 最後のボックスでは、テキストの角度を指定できます。

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。

EspressChart 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。

8.4 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

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:
ユーザー定義の光の位置は、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)。

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。

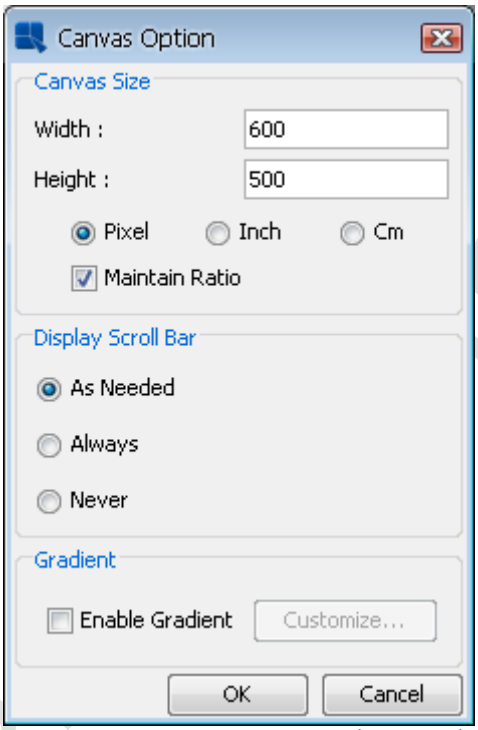
アイコン	名前	説明
	**Wire frame/solid mode On/Off**	スイッチがオンのときワイヤーフレームとして、スイッチがオフのときは立体オブジェクトとして 3D チャートを表示できます。
	**Border Drawing On/Off**	スイッチがオンのときにグラフの各端に黒い枠線を描画します。
	**InlineSeries On/Off**	シリーズの列を同じ XY 平面に描画します。このオプションは、データシリーズを含む縦棒グラフと棒グラフでのみ使用でき、チャートタイプが適用されない場合はナビゲーションパネルに表示されません。
	**Gouraud Shading 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イ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。

8.5 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。

8.5.1 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。



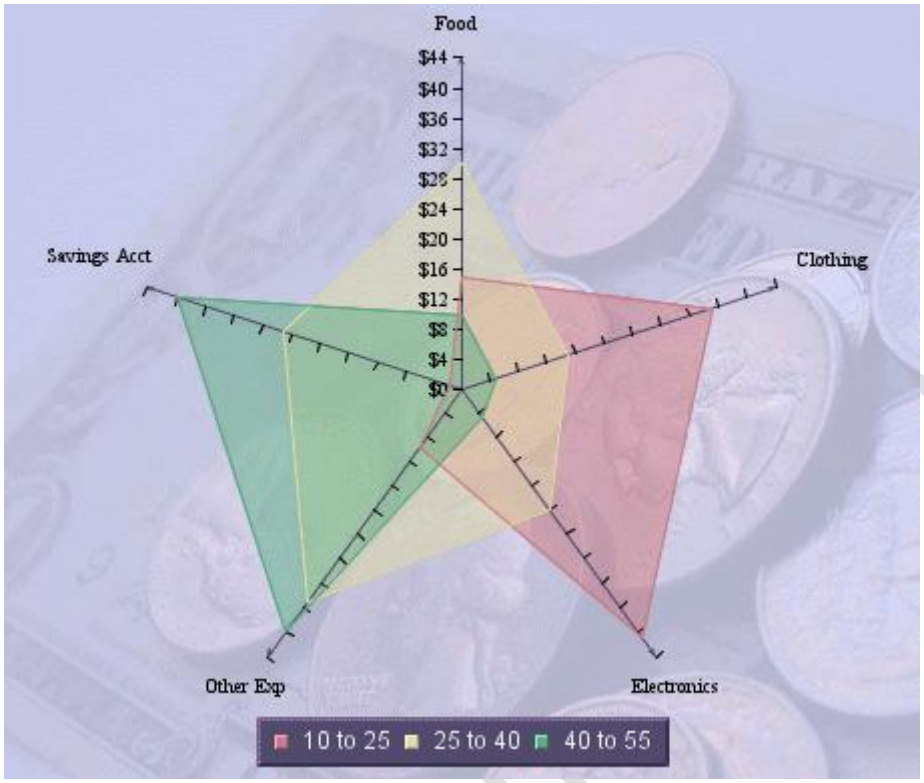
Canvas Formatting ダイアログ

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。

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。

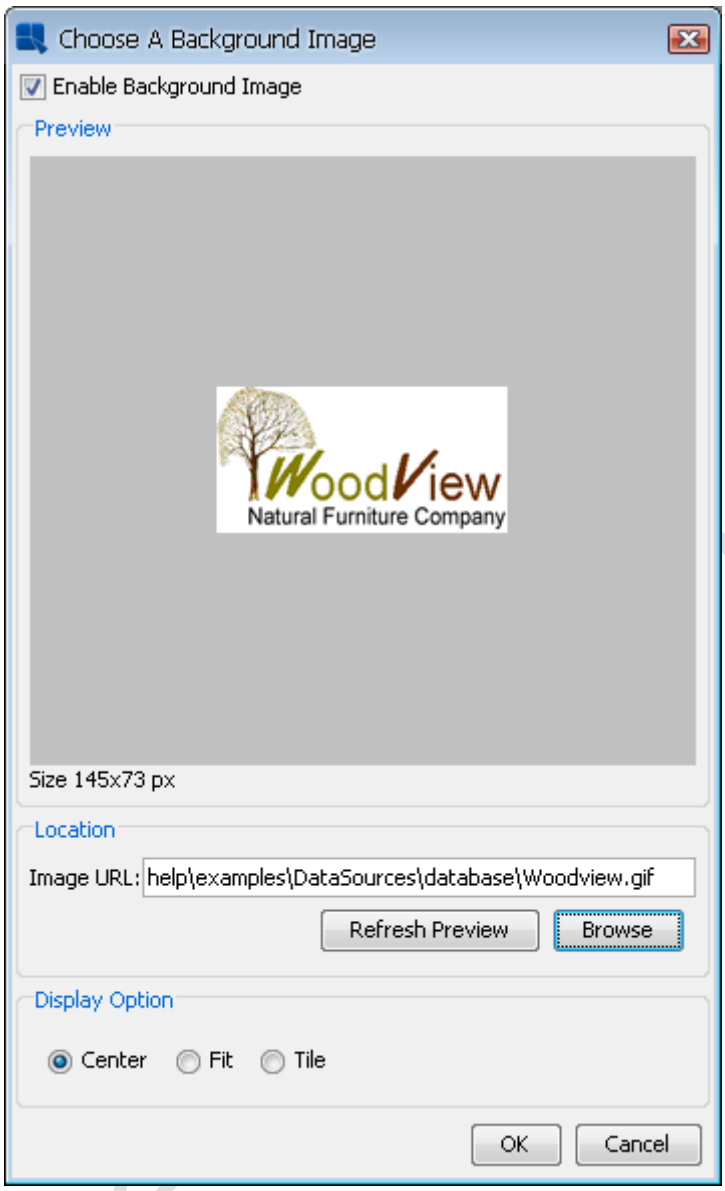
8.5.1.1 背景イメージ

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。



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、背景イメージを追加できます。これにより、グラフの背景イメージを指定できるダイアログが表示されます。



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。既存の背景イメージを削除し、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。

背景イメージを挿入するには、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。

背景イメージ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、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。

8.5.2 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。

プロットエリア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。チャートのサイズを変更するには、プロットエリア内を右クリックしてドラッグします。これにより、プロットが拡大または縮小されます。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調整することもできます。

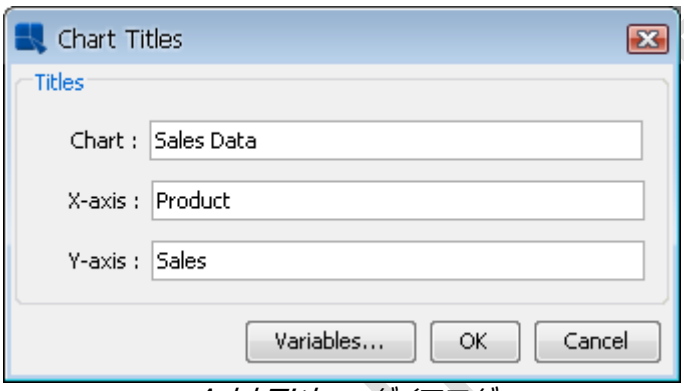
8.6 チャート要素の追加

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。

8.6.1 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。

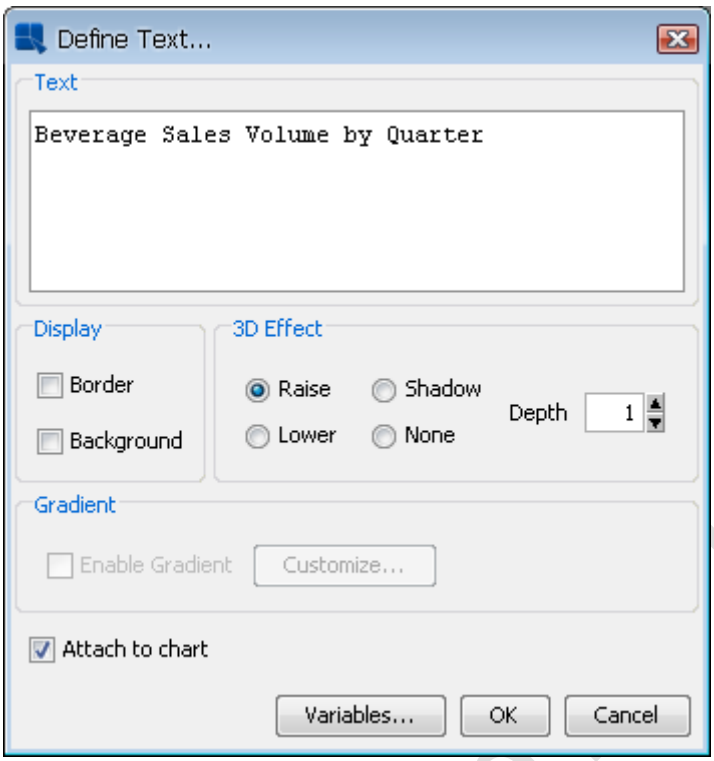
タイトルを追加
チャートにタイトルを追加するには、**Insert** > **Titles** を選択します。これにより、チャートのタイトルを入力するダイアログが表示されます。



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。



Add Text ダイアログ

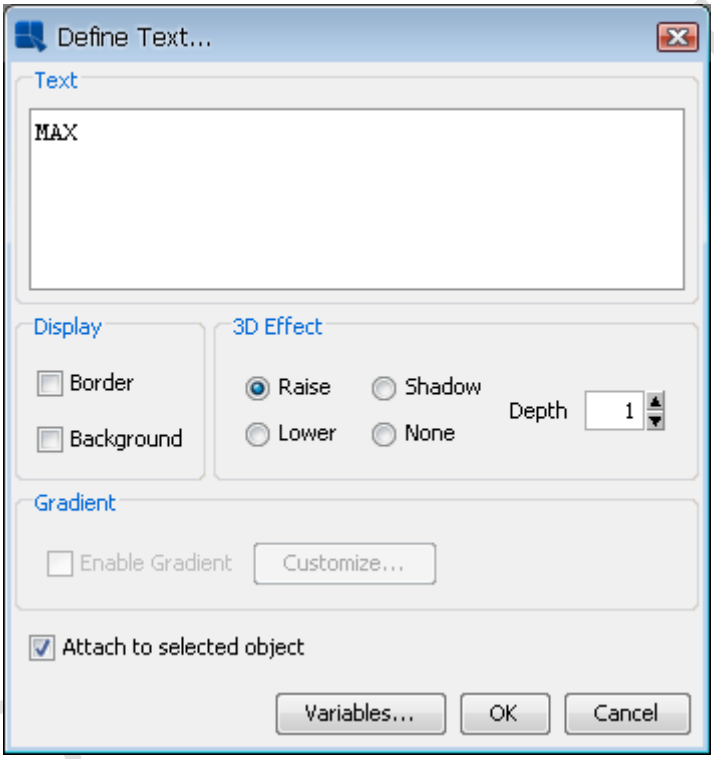
このダイアログでは、テキストを指定したり、表示オプションを設定したりできます。テキストの周りに、または背景の周りに枠線を描画するかどうかを指定できます。また、背景に適用する効果を指定することもできます。

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:
EspressChart 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。

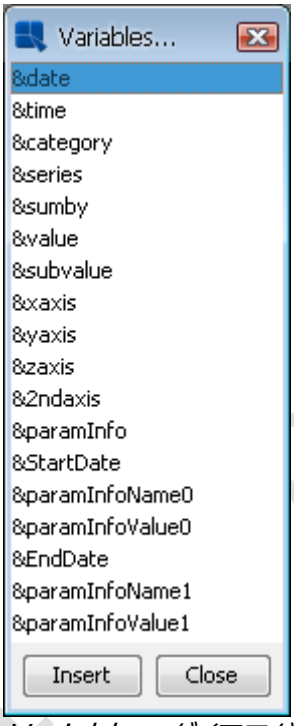


Add Text(Annotation) ダイアログ

8.6.1.1 TextVariables の設定

EspressChart 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。



Variables ダイアログ

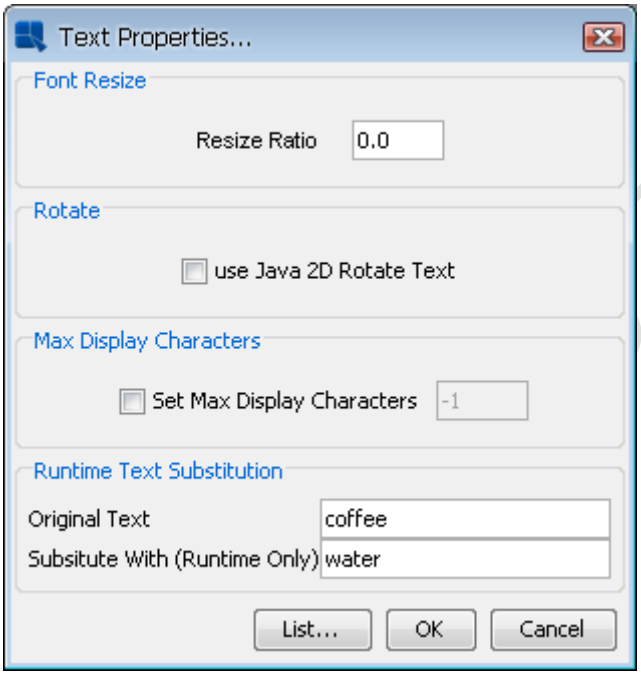
以下のテキスト変数がサポートされています。

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。

8.6.1.2 テキストの置換

EspressChart 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。

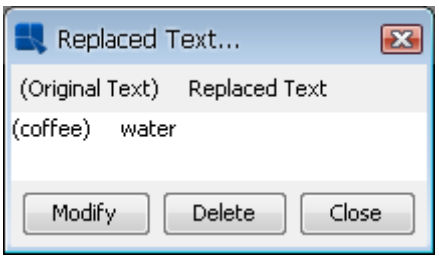
テキストの置換を使用するには、**Format > Text Properties** を選択します。置換されたテキストを指定するためのダイアログが表示されます。



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。

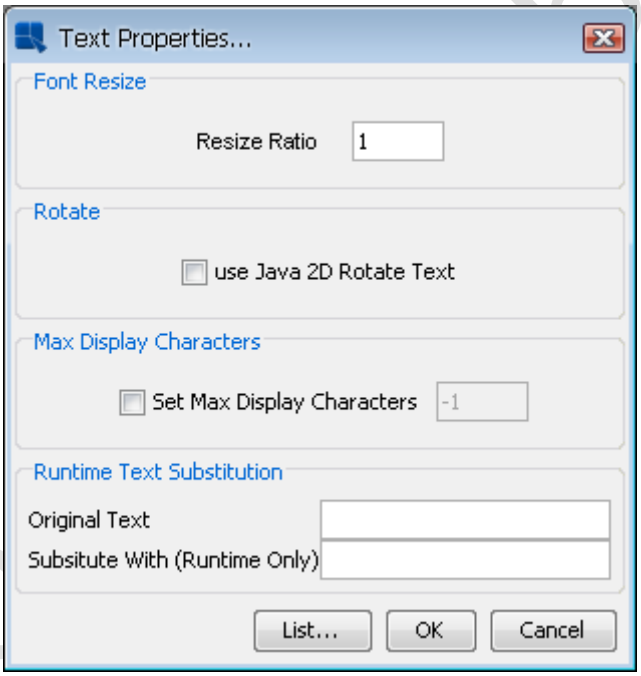


置換テキストリスト

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。

8.6.1.3 テキストサイズの自動調整

EspressChart 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。

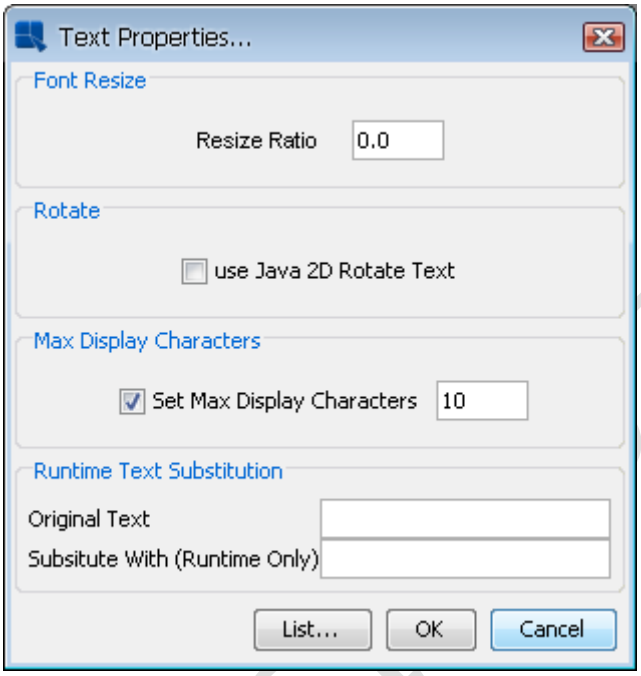


Text Properties ダイアログ

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。

8.6.1.4 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。ユーザー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

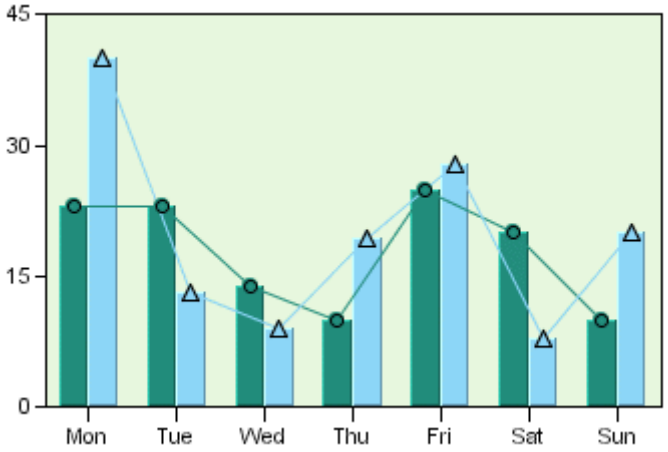
テキストのクロッピングを有効にするには、**Set Max Display Characters** オプションをオンにし、ダイアログ内の最大文字長を指定します。指定された文字数より長いテキストはクロッピングされます。

8.6.2 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。

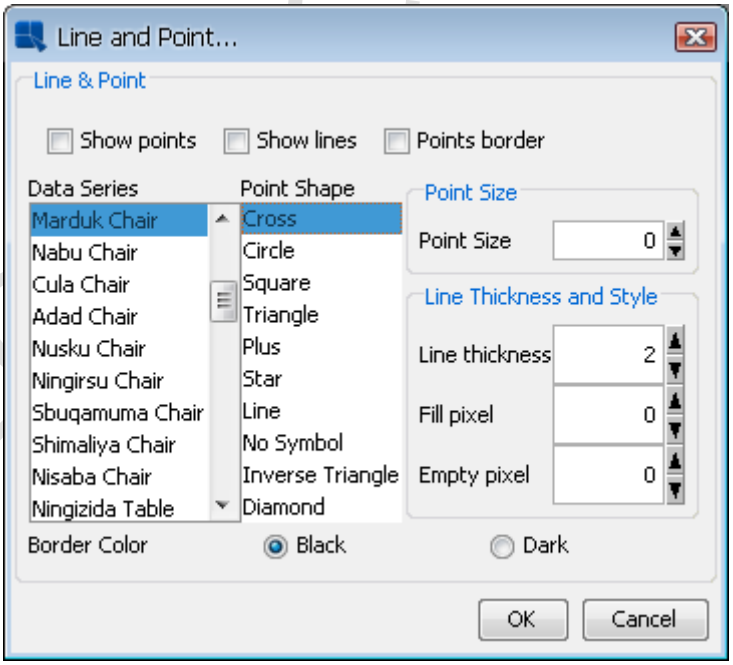
8.6.2.1 線と点の書式設定

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。



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。

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。

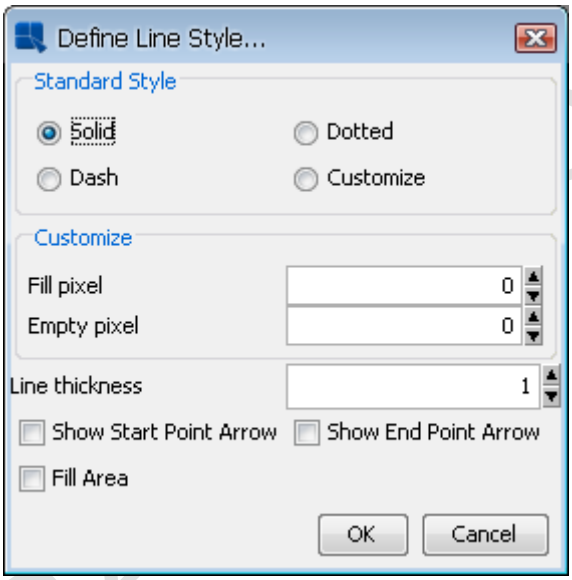
最後のオプションでは、データポイントのシンボルの枠線の色を黒または濃い色のシンボル色に変更できます。

8.6.2.2 グラフのフローティングライン

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。

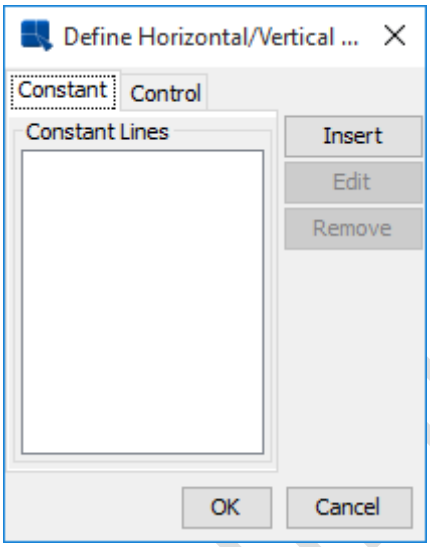


フローティングラインの Line and Point ダイアログ

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。

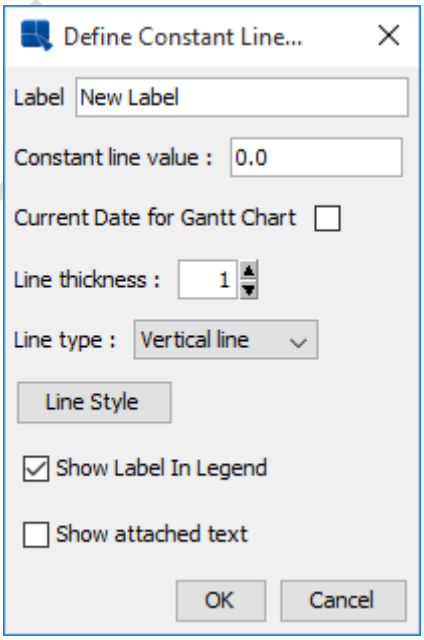
8.6.2.3 水平線または垂直線の設定

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。



Constant Line ダイアログ

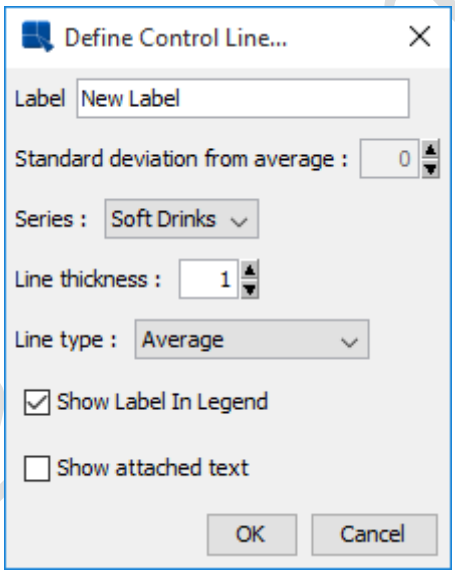
定数行の場合は、行に使用する数値だけでなく、行のラベルも指定する必要があります。カテゴリ軸の場合、データポイントは 0.5 から始まることに注意してください。線の太さと線が水平か垂直かを指定することもできます。最後の 2 つのオプションを使用すると、行のチャート凡例に項目を追加することや、その行に注釈を表示するかどうかを指定できます。

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、テキストの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。

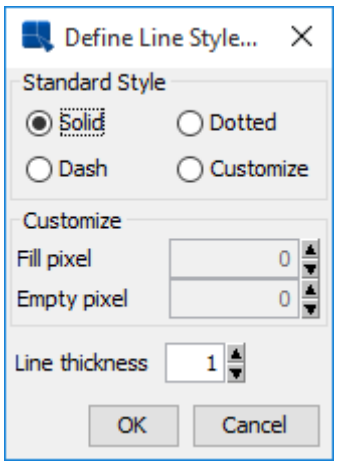


Control Line ダイアログ

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。

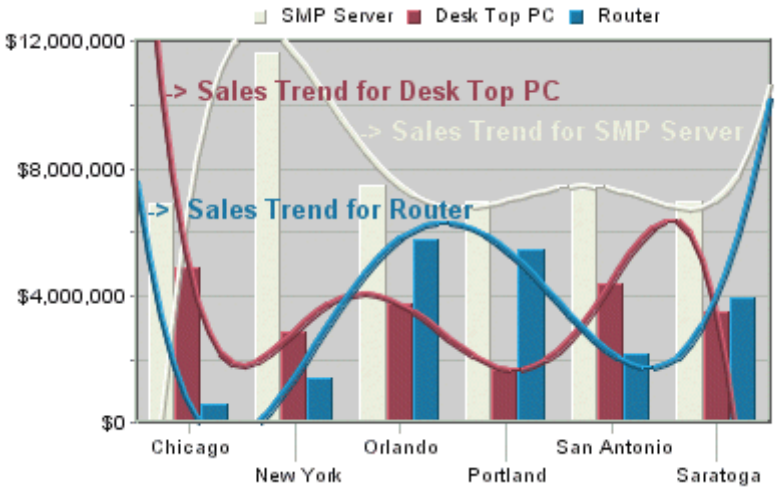


Line and Point ダイアログ

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。

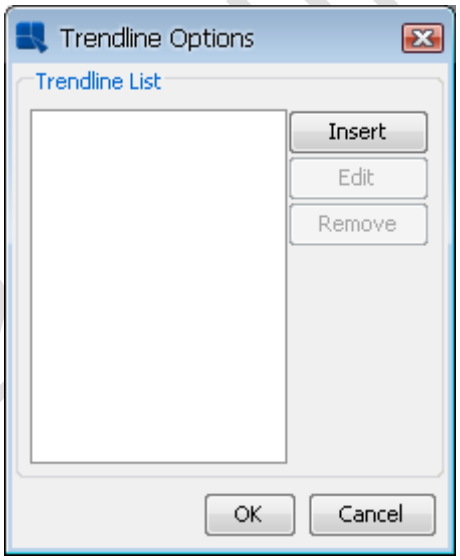
8.6.2.4 トレンドライン

EspressChart 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。



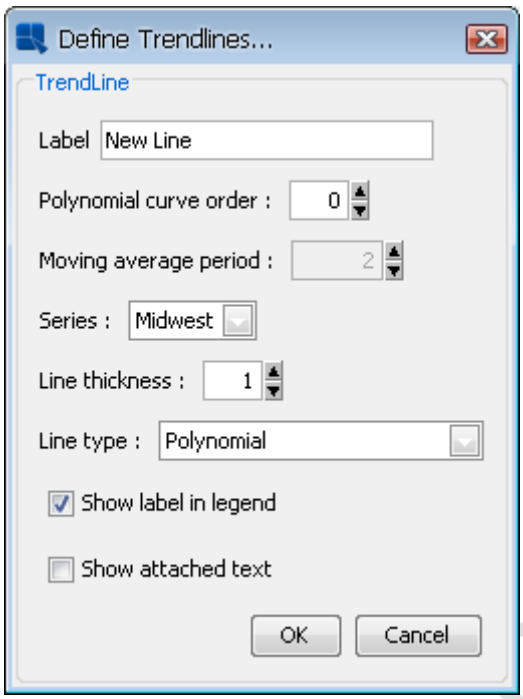
トレンドラインチャート

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。



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。



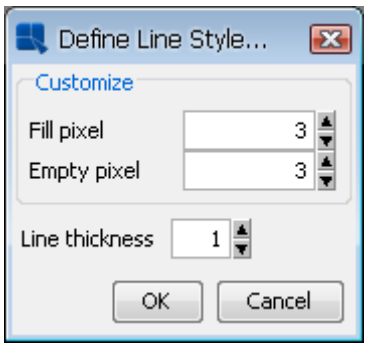
Define Trend Lines ダイアログ

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。

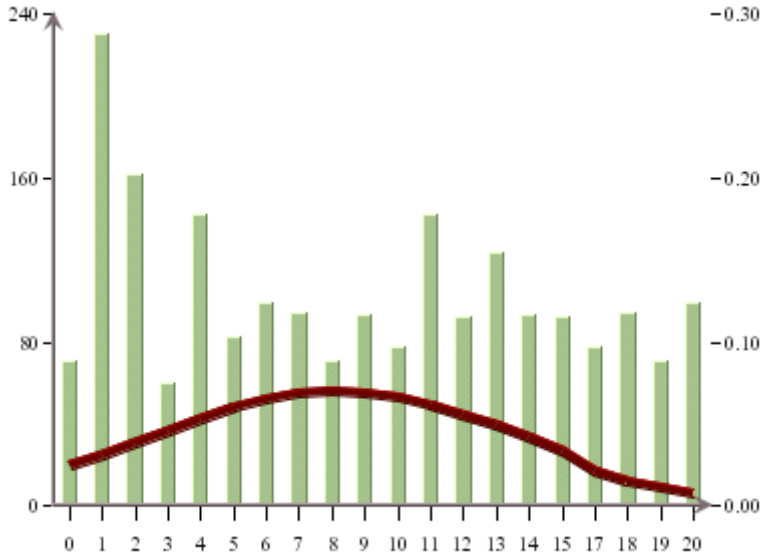


トレンドラインの Line and Point ダイアログ

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。

8.6.2.4.1 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。

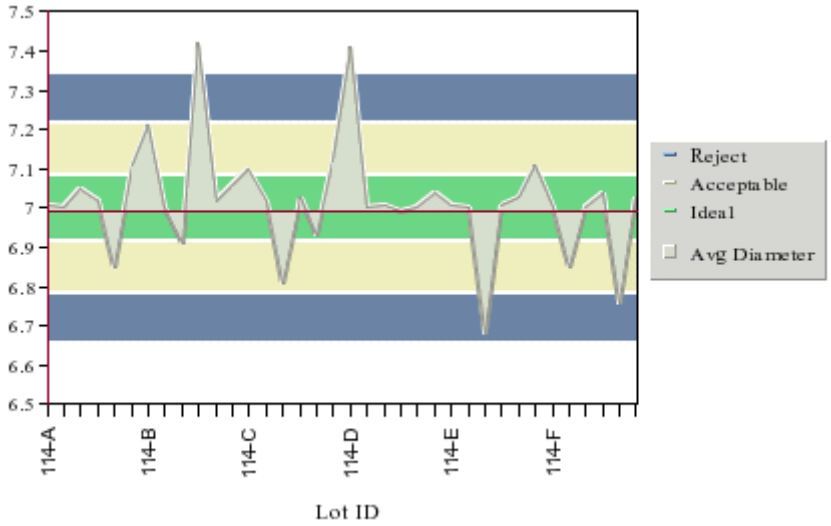


正規分布曲線のチャート

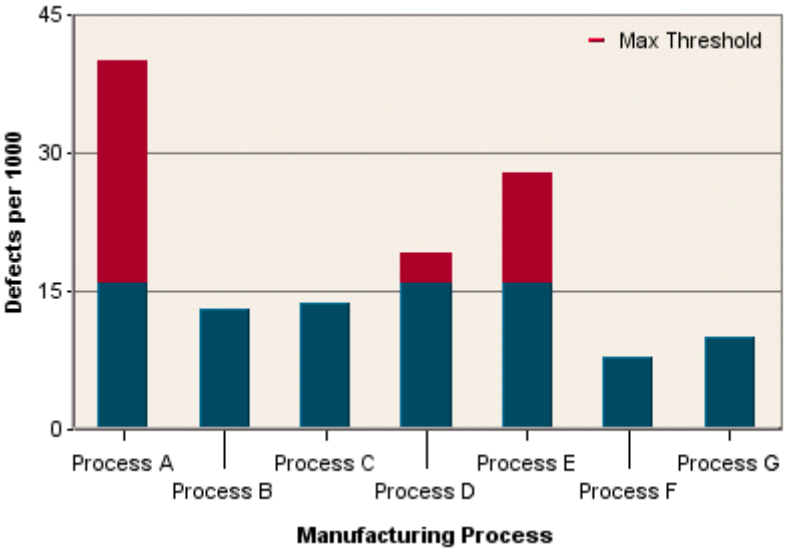
カーブのスケールは通常、値軸のスケールとは異なるため、カーブは 2 軸に表示されます。2 軸のスケールを変更することによって、スケールを変更できます。

8.6.3 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 2D 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エリア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。プロット上に背景エリア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ーに明確な視覚的参照が提供されます。

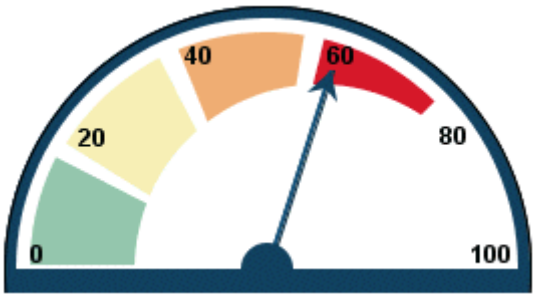


コントロールエリアを持つ 2D エリアチャート



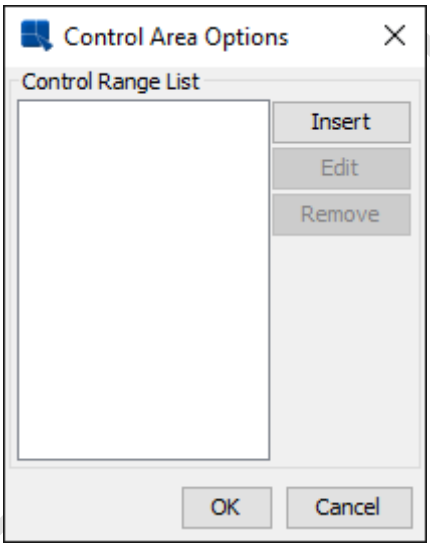
データポイント用のコントロールエリアを含む縦棒グラフ

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。



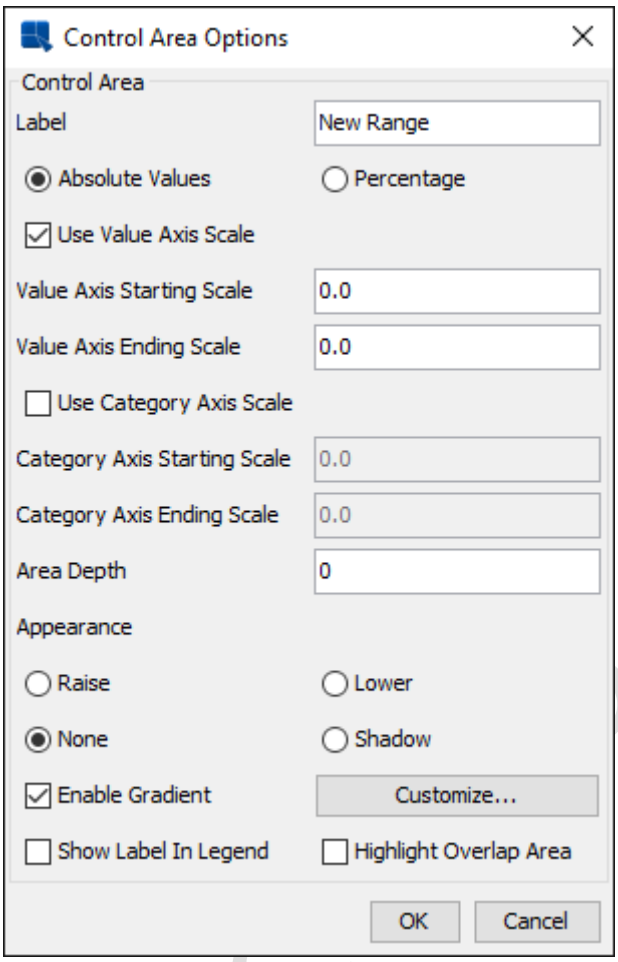
コントロールエリアを持つダイヤルチャート

チャートにコントロールエリアを追加するには、**Insert** > **Control 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。



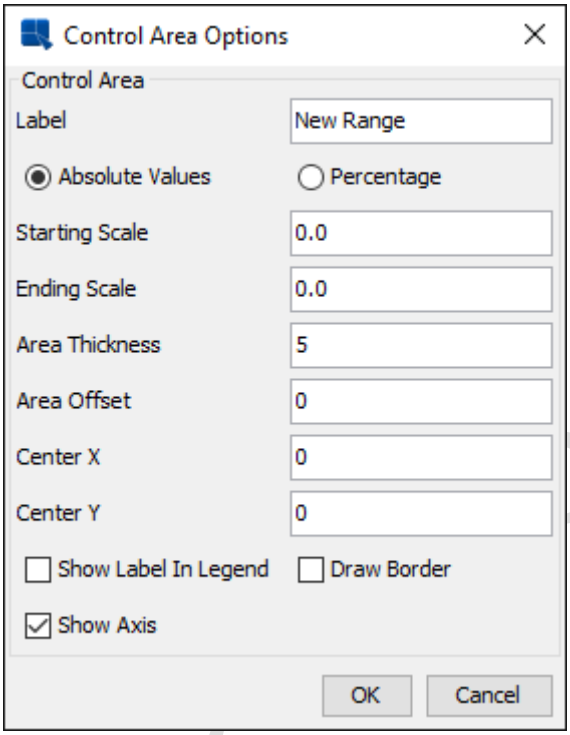
Contorol Area Configuration ダイアログ

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、コントロールエリアに 3D またはシャドウエフェクトを指定できます。エリア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。



ダイヤルチャートの Control Area ダイアログ

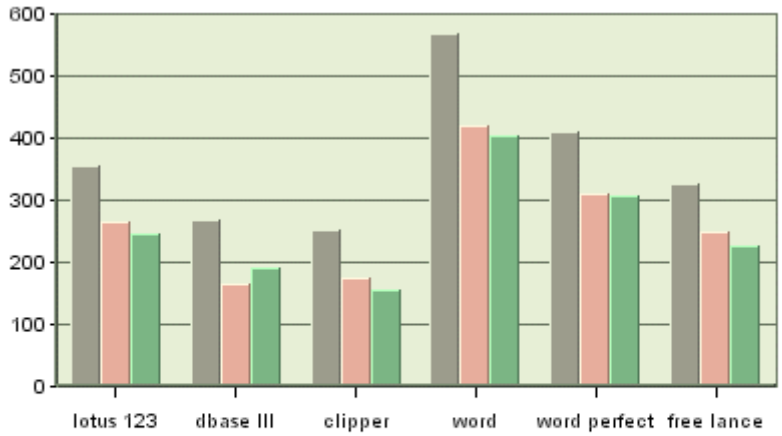
ダイヤルチャートのコントロールエリアダイアログは次のオプションを提供しています。

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、コントロールエリアの中心の 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枠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エリアの軸を表示または非表示にすることができます。

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。

8.6.4 テーブルの追加

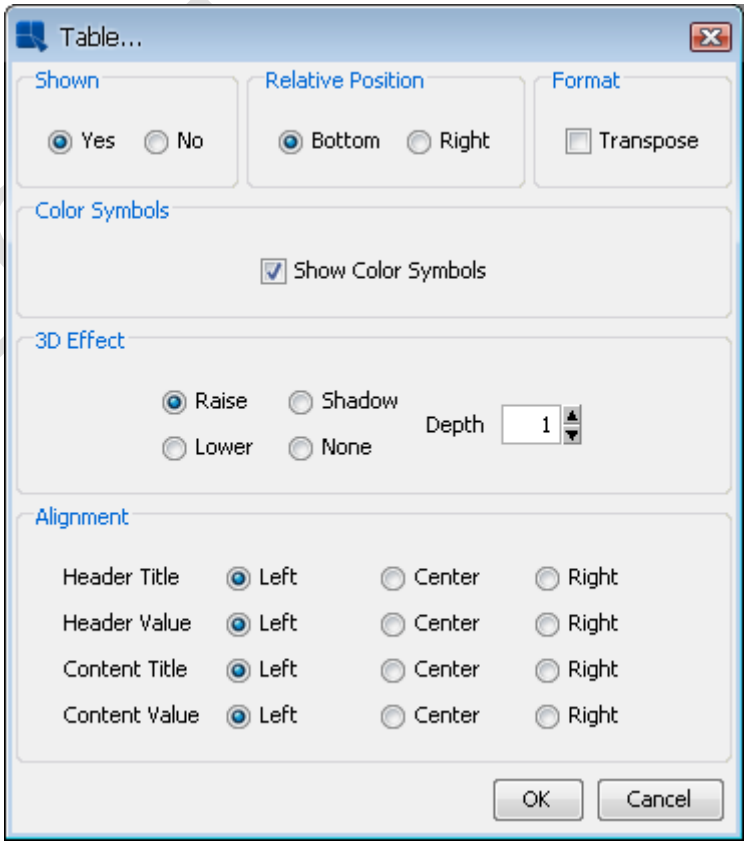
チャートの表示に加えて、チャートに表示されるデータポイントを示すテーブルを表示することもできます。表はチャートプロットの下または右に配置できます。



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。



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、データシリーズ要素は行として描画されます。

Show Color Symbols オプションでは、表のデータシリーズ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

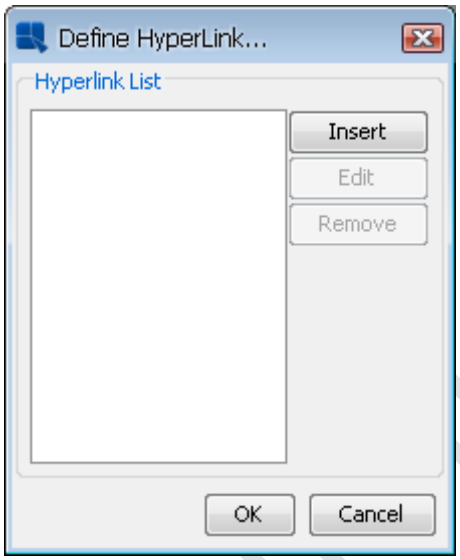
カラーボックス付きチャートテーブル

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。

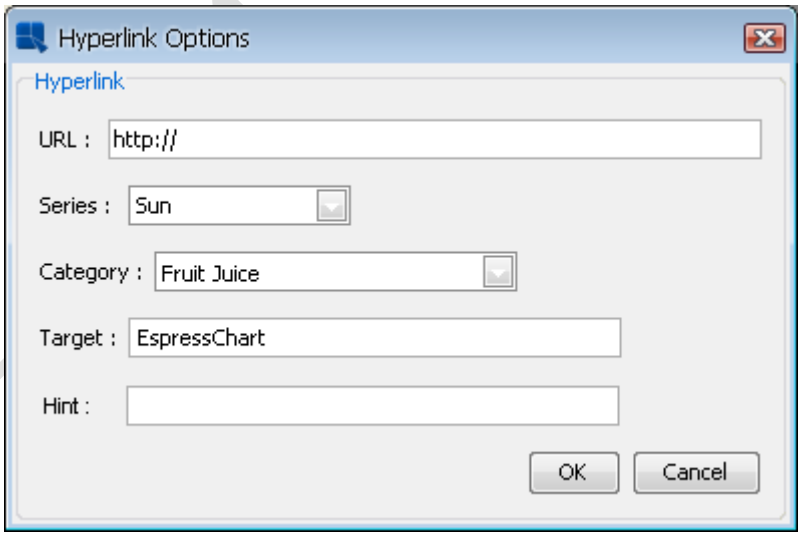
8.6.5 ハイパーリンクの追加

EspressChart 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エレメント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

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シリーズとカテゴリ要素の要素を選択できます。すべてのデータシリーズ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シリーズ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

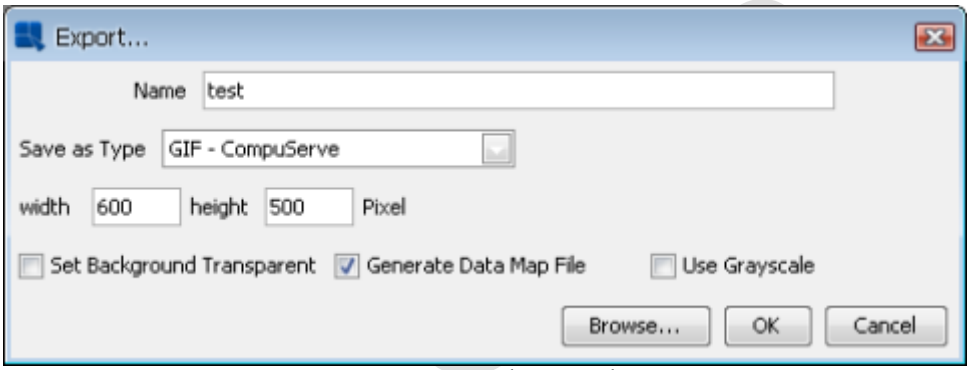
Target パラメータを指定できます。これは、新しい HTML ページを新しいブラウザウィンドウで開くか、同じブラウザウィンドウで開くか、または新しいページが現在のページと同じページの部分を占めるかどうかを判断するために使用できます。

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。

8.6.5.1 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。



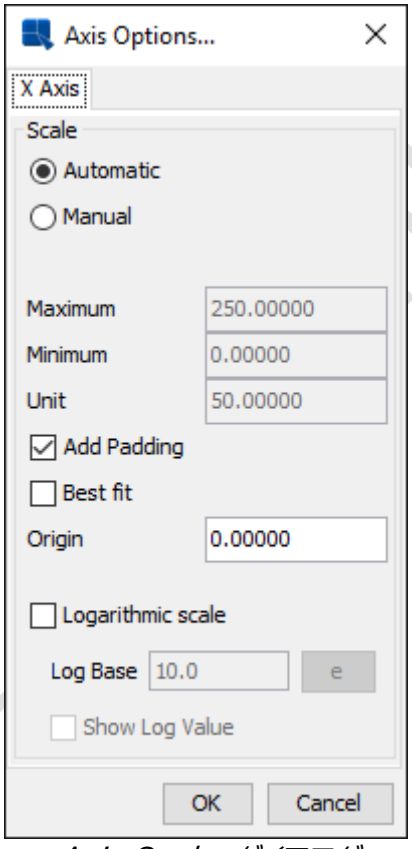
Export ダイアログ

8.7 グラフ軸の書式設定

EspressChart 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。

8.7.1 グラフ軸のスケール設定

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。



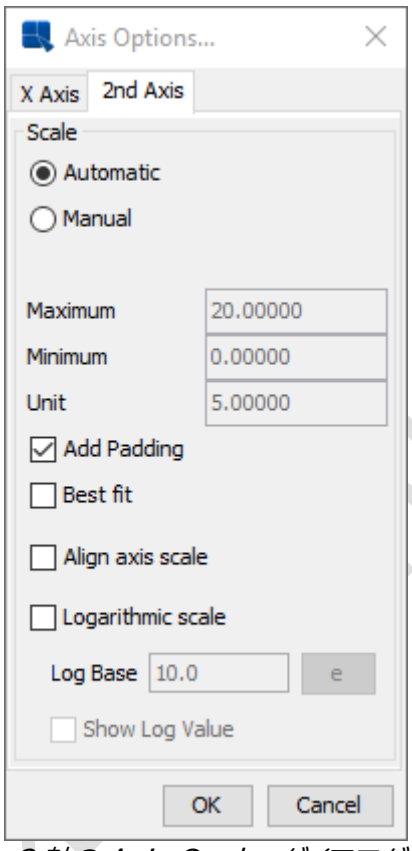
Axis Scale ダイアログ

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。

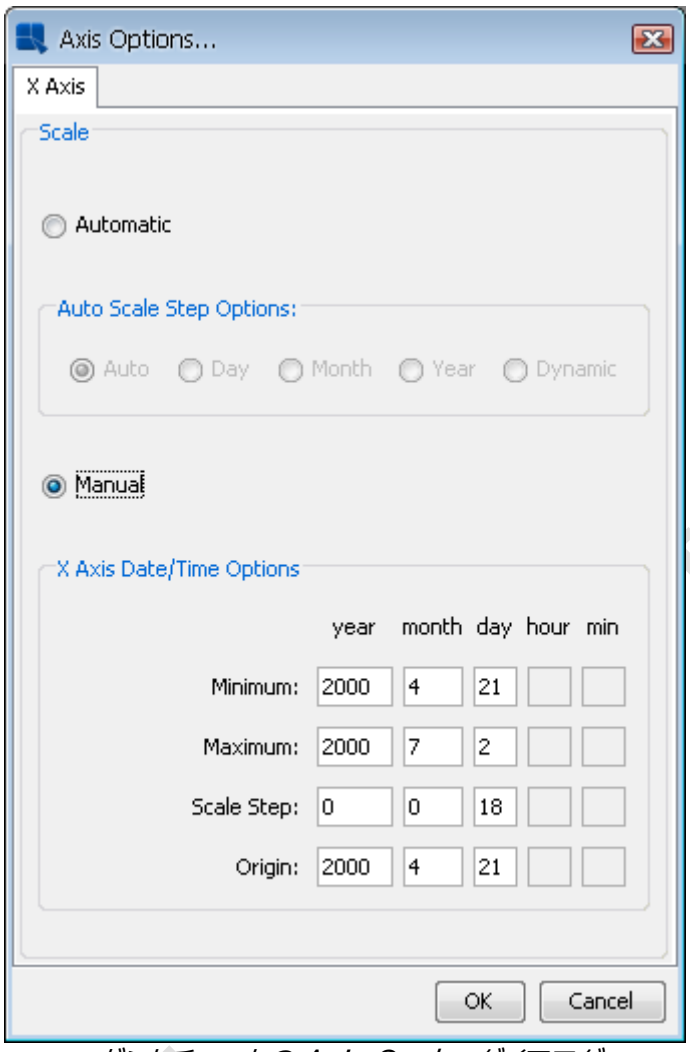
Log Base	ログ値のベースを指定することができます。
Show Log Value	軸ラベルにログ値を表示するかどうかを指定します。

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

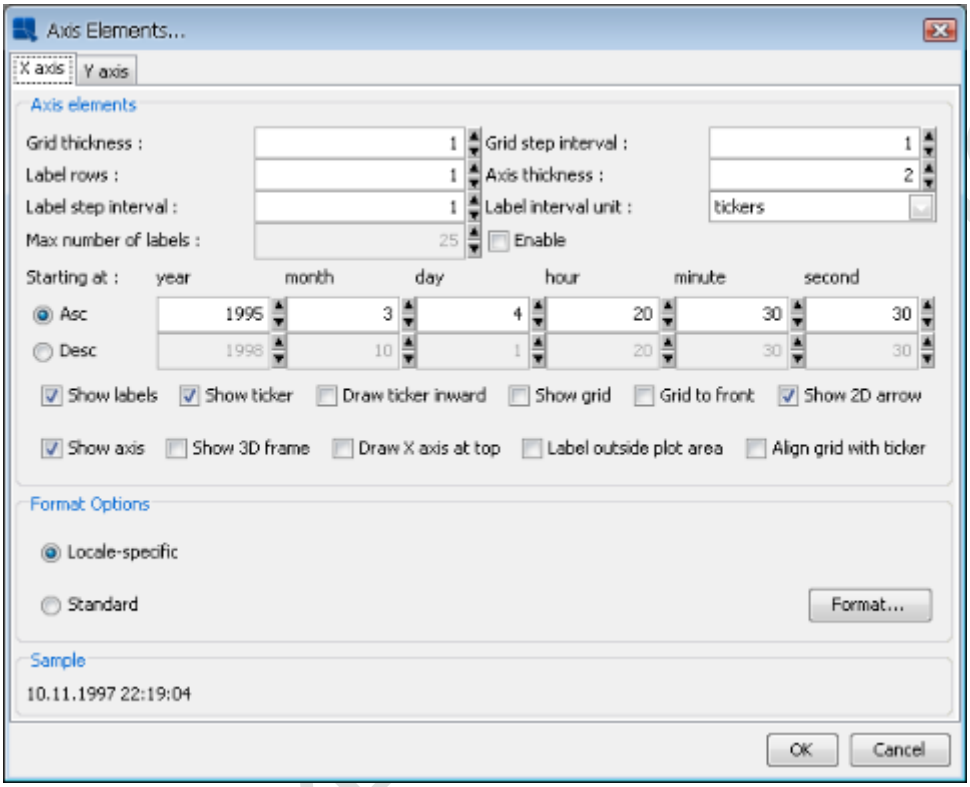
ガントチャートでは、軸スケールのダイアログが異なります。自動オプションを選択してスケールを自動的に設定するか、手動オプションを選択してスケールを手動で設定することもできます。自動軸スケールには、自動、曜日、月、年、および動的の、いくつかの自動スケールステップオプションがあります。自動オプションは常に最適なものを見つけます。動的オプションも最適ですが、自動オプションとは異なり、標準的なステップ間隔のみを使用します(たとえば、1 か月、1 年など...)。手動軸スケールを選択すると、Maximum、Minimum、Unit、Origin がそれぞれ Maximum Date、Minimum Date、Scale 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

8.7.2 グラフ軸の要素設定

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。

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。

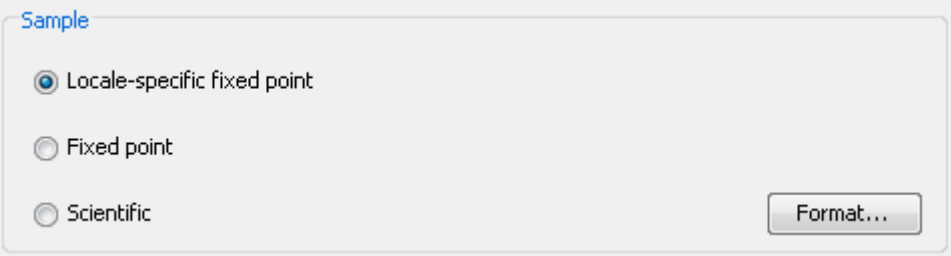
Ascending/Descending	時間ベースのデータを並べ替えることができます。データを昇順または降順に並べ替えるだけでなく、データの開始(または終了)ポイントを指定することもできます。
Show labels	各軸のラベルを削除または表示できます。
Show ticker	軸ティッカーを表示または削除できます。
Draw ticker inward	軸のテロップを外側(デフォルト)ではなくプロットエリアの内側に描画します。
Show sub/tickers	軸のスケールが 10 のログベースで対数に設定されている場合にのみ、値軸で使用できます。値軸上の点間に間隔ティッカー(非一様)が描画されます。
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、2D 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(2D チャートの場合)または壁面(3D 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、3D 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、プロットエリアの外側に配置されるラベルを設定します。 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。 2D 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。

8.7.2.1 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

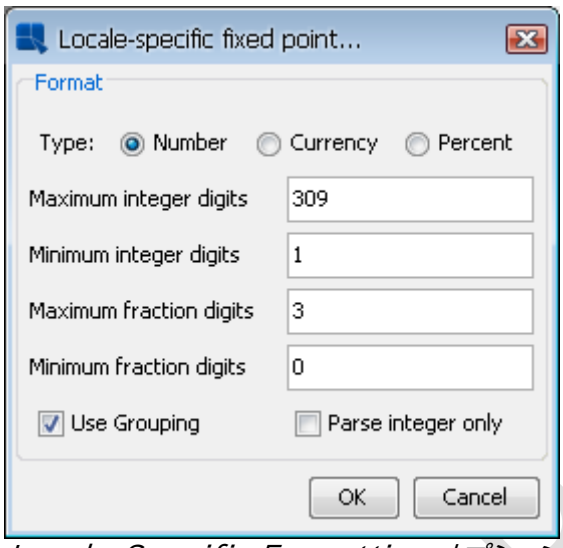
数値データの場合、表示フォーマットにはロケール固有の固定小数点、固定小数点、および科学という 3 つの主要なオプションがあります。**Format** ボタンをクリックすると、追加のオプションが表示されます。



数値データ形式のオプション

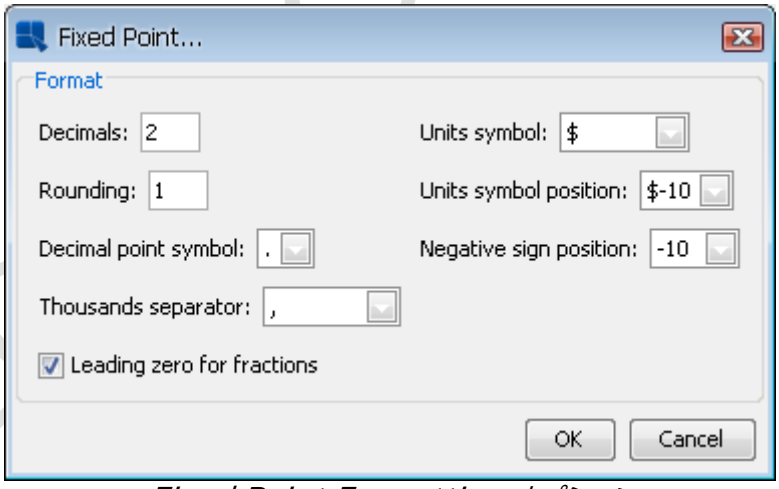
- **Locale-Specific Fixed Point:**

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。



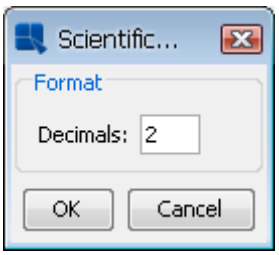
Locale-Specific Formatting オプション

- **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。



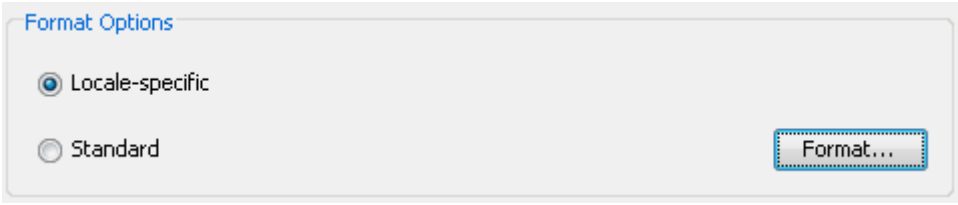
Fixed Point Formatting オプション

- **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。



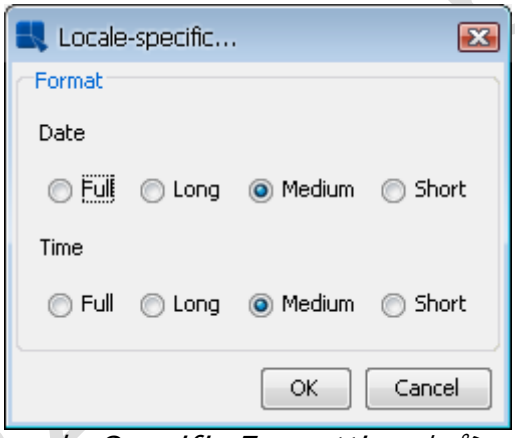
Scientific Formatting オプション

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。



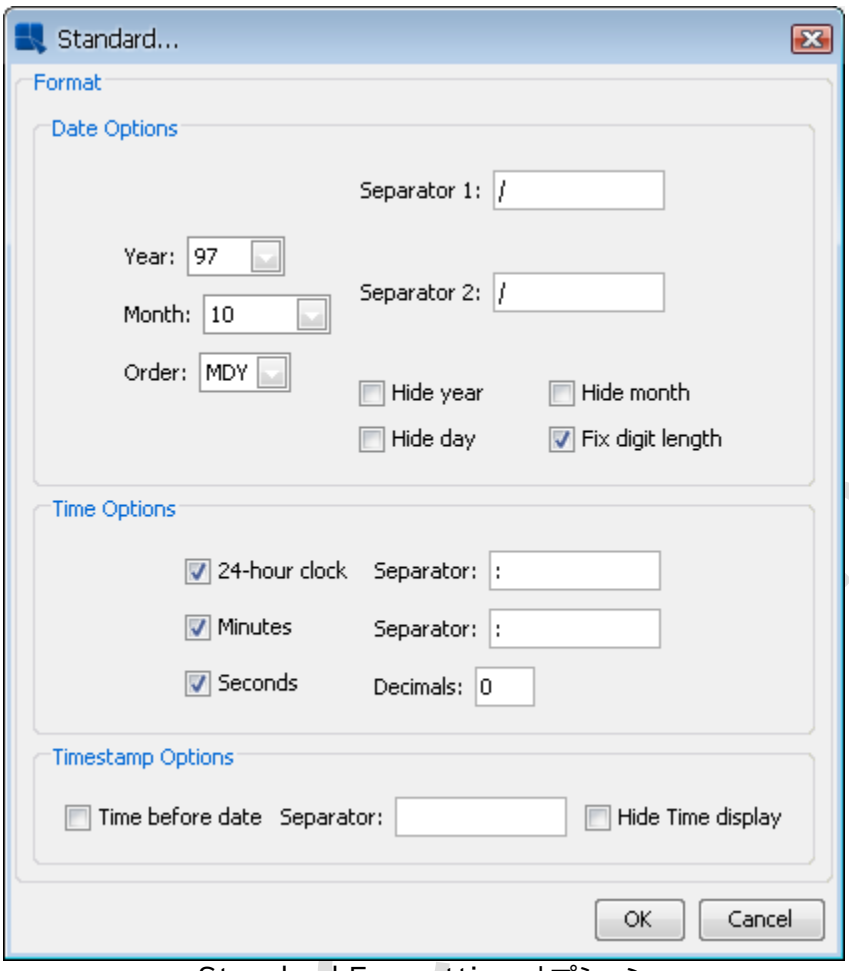
Date/Time Data Format オプション

- **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。



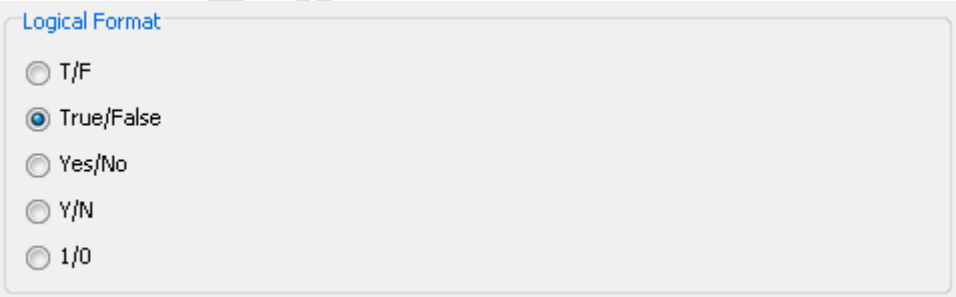
Locale-Specific Formatting オプション

- **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/または秒を表示し、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。



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

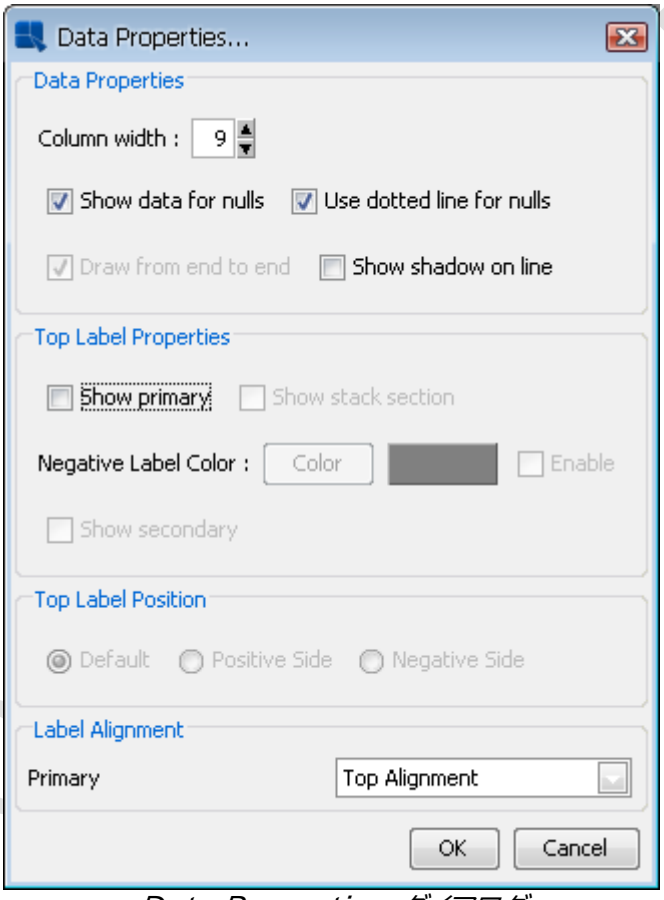
データの書式を変更すると、軸要素ダイアログで **OK** ボタンをクリックすると、その変更が有効になります。
文字列データの追加の書式設定オプションはありません。

8.8 プロット/データ要素の書式設定

EspressChart 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。

8.8.1 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、横棒/縦棒のサイズを制御することや、ヌル値の表示オプションを設定、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棒グラフの間のギャップに対する横棒/縦棒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の横棒、縦棒、積み重ね横棒、積み重ね縦棒、HighLow、HLCO、およびガントチャートにのみ関連します。3D グラフで棒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。

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。

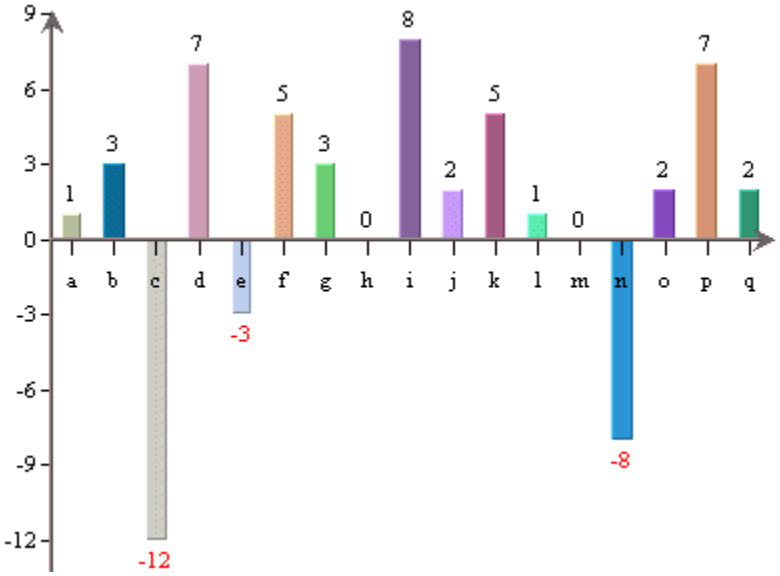
Draw from end to end:
チャート上の最初と最後のデータポイントにオフセットするのではなく、プロットエリア全体に 2D の折れ線グラフと面グラフを描画できます。

Show shadow on line:
2D 折れ線グラフ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:
積み重ね横棒、積み重ね縦棒、および積み重ねエリアチャートの各積み上げ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

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。追加のオプション積み上げチャートでは、積み上げセクションラベルのアラインメントを設定できます。

8.8.2 日付/時間ベースの拡大縮小

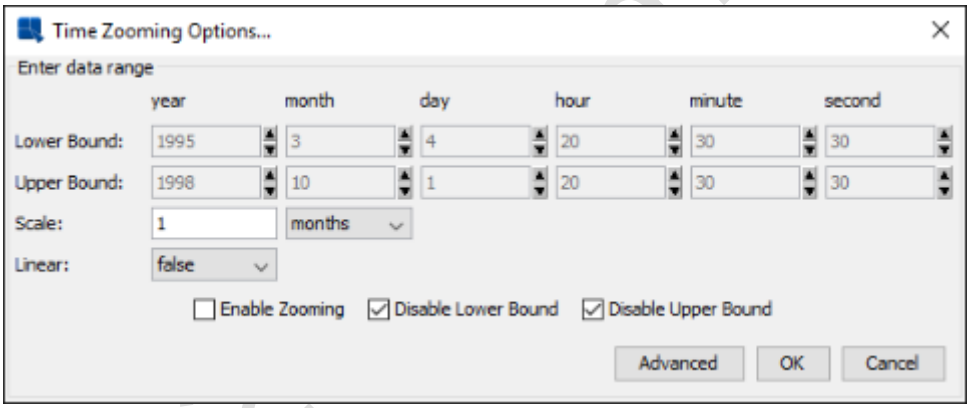
カテゴリ軸上の日付または時刻データを表示するチャートの場合、EspressChart はユーザーが日時ベースのズームを実行できるユニークな機能を提供します。この機能を使用すると、カテゴリ要素をユーザー定義の間隔にグループ化し、各グループのポイントを集約することができます。また、結果の上限と下限を指定してデータをフィルタリングすることもできます。

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、スキャッター、サーフェイス、ボックス、ダイヤル、ポーラー、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。

8.8.2.1 拡大縮小オプションの追加

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。



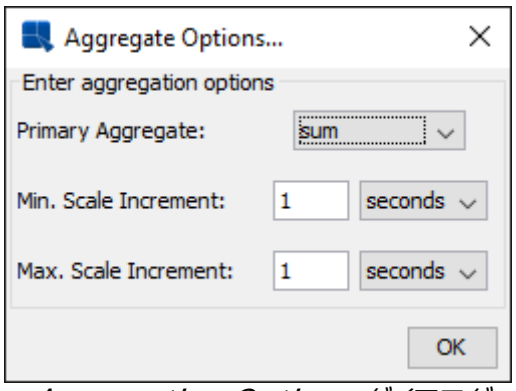
Zoom Options ダイアログ

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。

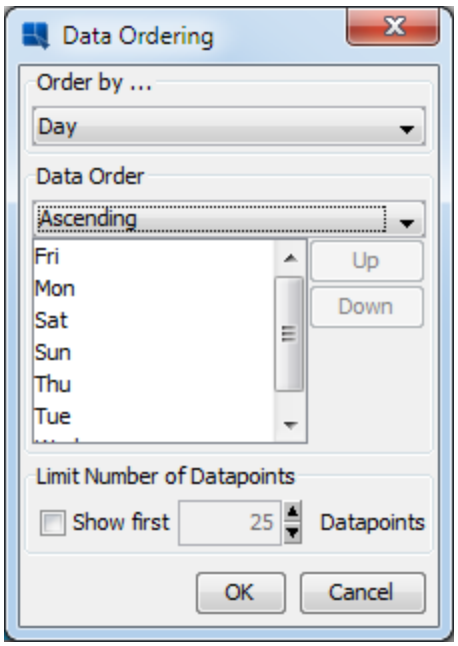
8.8.2.2 チャートビューの拡大縮小

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。チャートビュアーで時シリーズ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、チャートビュアー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。

8.8.3 発注データ

EspressChart 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。

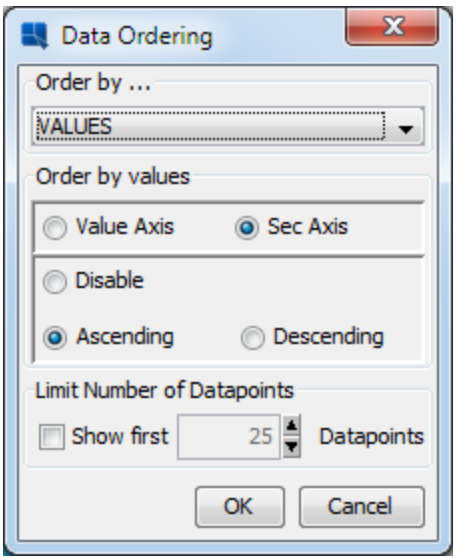
DataSource Order:
順序をオフにします。カテゴリ/シリーズの注文はデータソースのみに依存し、EspressChart では変更されません。

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。

Descending:
カテゴリ/シリーズ要素を降順で並べ替えます。

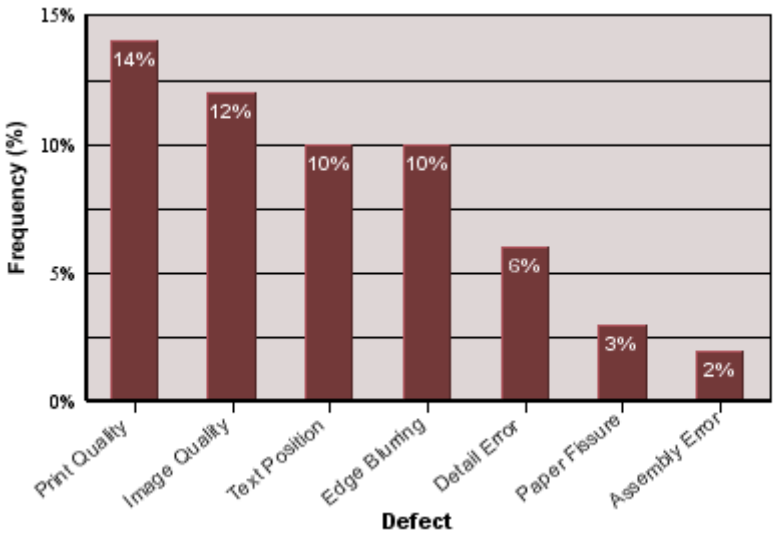
Customize:
このオプションでは、カテゴリ/シリーズの順序をカスタマイズできます。順序をカスタマイズするには、カテゴリ/シリーズ項目のリストから項目を選択し、**Up** または **Down** ボタンをクリックしてリスト内を上下に移動します(このボタンは **Customize** オプションを選択するまで非アクティブです)。

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。



パレート図

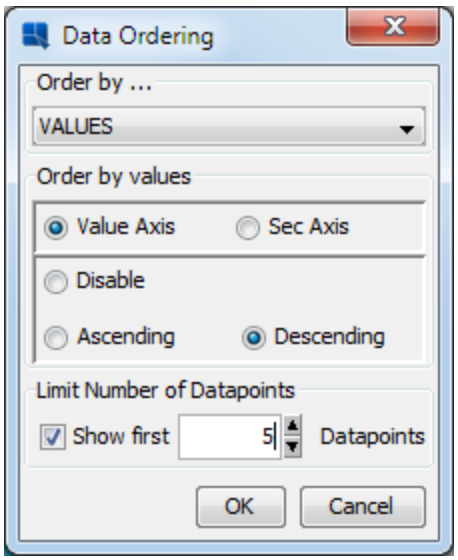
チャートが更新された場合、および/またはデータが変更された場合は、ソートセットが再度適用されます。

8.8.3.1 トップ/ボトム"N"値のチャート

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。

グラフにデータシリーズがない場合、**Ordering** ダイアログには **LimitPoint Of DataPoints** オプションが表示されます。



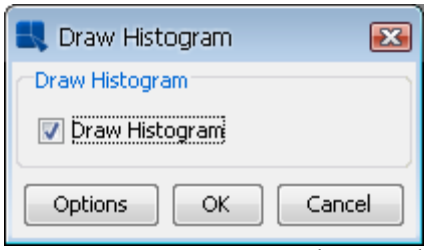
このオプションは、カテゴリまたは値、または昇順または降順の値でグラフをソートする場合にのみ有効にすることができます。 このようなグラフがある場合は、**Show first** オプションを選択してこの機能を有効にすることができます。 次に、グラフに表示される項目の最大数を指定できます。 データソースがこのオプションで指定した以上のアイテムを返した場合、余分なアイテムは、存在しないかのようにチャートに表示されません。

8.8.4 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。EspressChart 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。

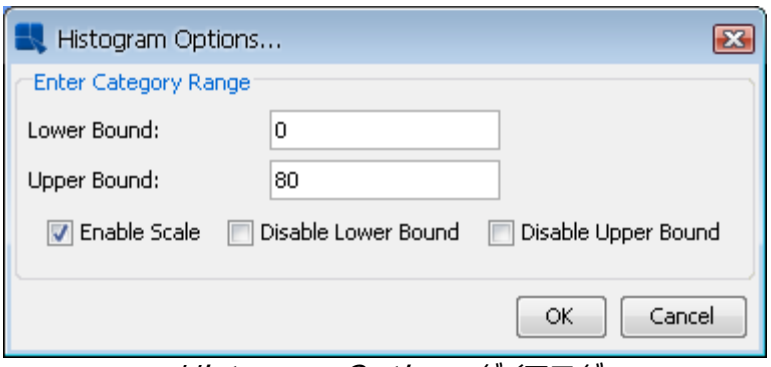
ヒストグラムは、各カテゴリ要素のデータ点またはインスタンスを数えることによって計算されます。数値カテゴリの場合は、さらに上限と下限を指定できます。また、頻度カウントの範囲を作成するためのスケールも指定できます。

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。



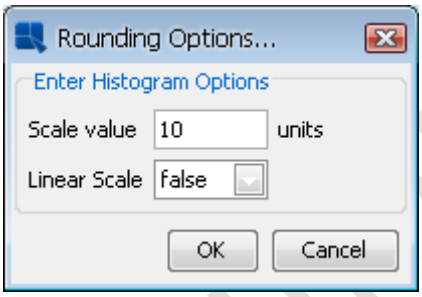
Select Histogram ダイアログ

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

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。

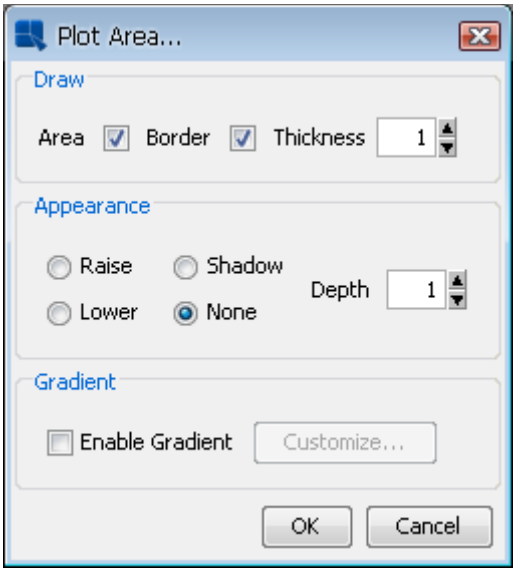


Histogram Scale オプション

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。

8.8.5 プロットエリアの書式設定

プロットエリアは、2D 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、プロットエリアの外観をカスタマイズできます。現在のチャートが 2D 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。



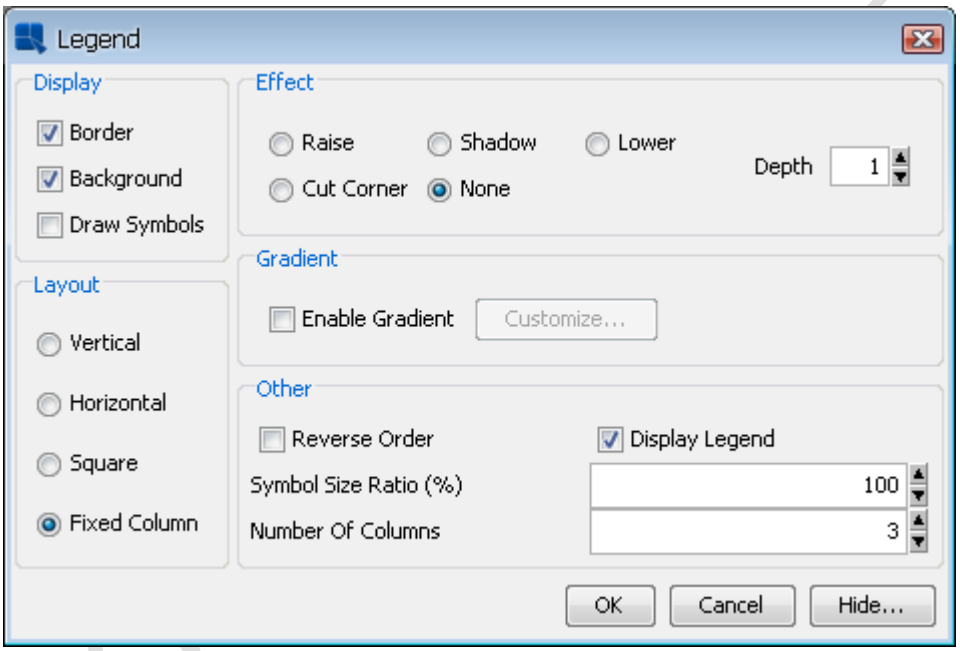
Plot Area ダイアログ

このダイアログでは、プロットエリアの周りに枠線を描くことも、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、プロットエリア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、7.1.3 - 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。

8.8.6 フォーマットデータの凡例

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、凡例の枠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、7.1.3 - 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。

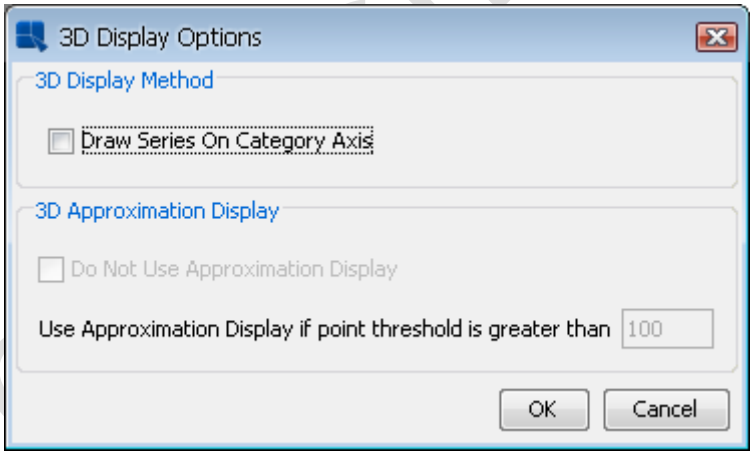
Other:
これにより、凡例を表示するかどうかを選択したり、凡例を逆順に描画したりすることができます。 凡例の固定列数を**列数**フィールドに設定できます。 このフィールドは、**レイアウト**セクションで**固定列**レイアウトオプションを選択した場合にのみアクティブになります。また、凡例のシンボルのサイズを変更することもできます。

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。

8.8.7 3D 表示のオプション

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 を消費することがあります。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。

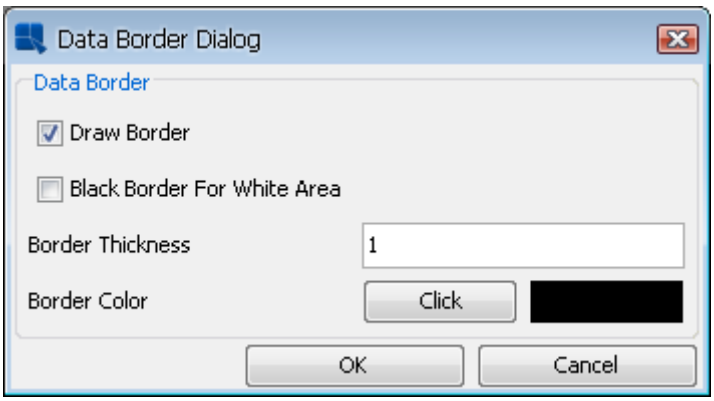


3D Display Options ダイアログ

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)。

8.8.8 枠線のオプション設定

縦棒グラフ、横棒グラフ、積み上げカラムグラフ、積み上げ棒グラフ、および HLCO チャートの場合、EspressChart を使用してカラムの周りに枠線を設定できます。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、枠線のオプションを設定できるダイアログが表示されます。



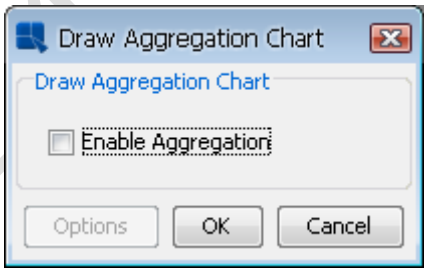
Data Border ダイアログ

Draw Border : データの枠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、枠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、枠線の太さと枠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。

8.8.9 集約オプション

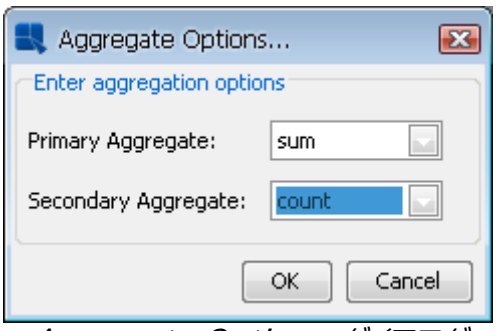
EspressChart では、特定のカテゴリ(シリーズや積み上げが存在する場合はそのシリーズや積み上げ)に複数のデータポイントが関連付けられている場合、データを集約できます。これにより、単一のデータポイント(多数のデータポイント)だけではなく、データを幅広く見ることができます。

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。



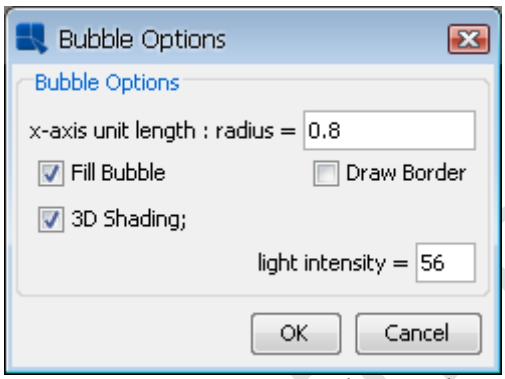
Aggregate Options ダイアログ

8.9 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。

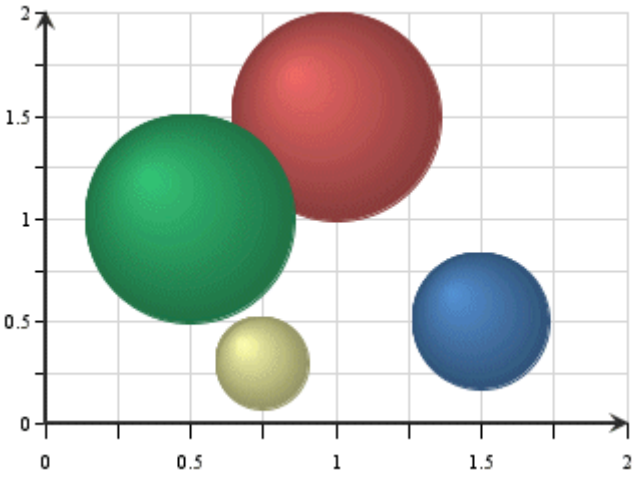
8.9.1 バブルチャート

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

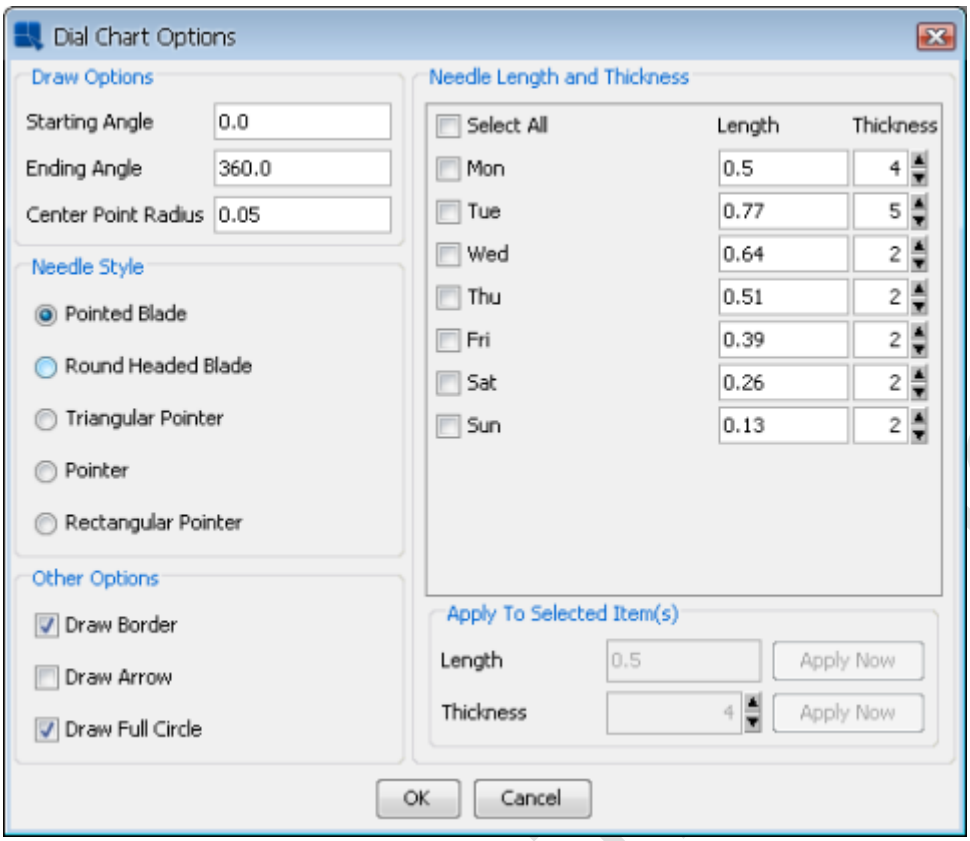
バブルチャートでは以下のオプションが利用可能です。
x-axis unit length : radius: バブルの半径に対する X 軸単位の長さの比を指定します。
Fill Bubble: バブルエリアを埋めるかどうか指定します。
Draw Border: バブルの枠線を描画するか指定します。
3D Shading: 3D シェーディングをバブルに追加します。シェーディングの光強度を調整することもできます。



バブルチャートの 3D シェーディング

8.9.2 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、枠線とダイヤルエリア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。

Ending 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、枠線とダイヤルエリア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:
ダイヤルの枠線を描画するか指定します。

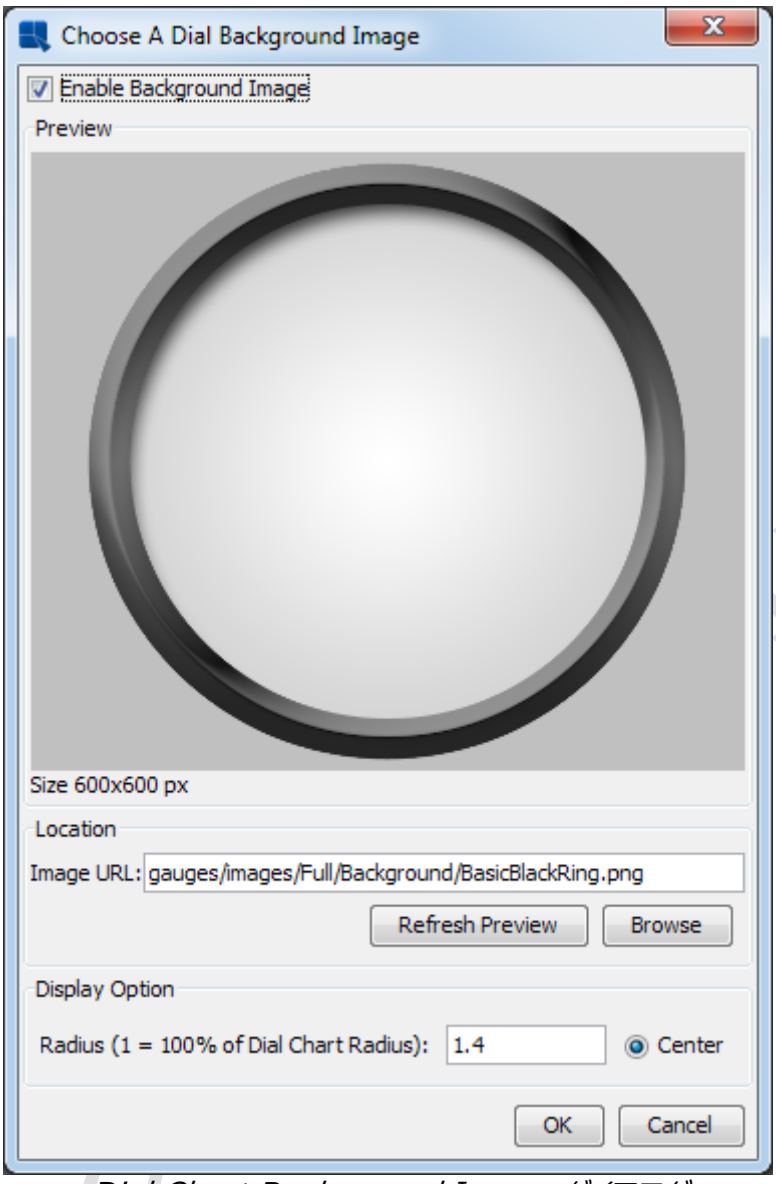
Draw Arrow:
ダイヤルの針の端に矢印を描くかどうか指定します。

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。

8.9.2.1 ゲージ画像

ダイヤルチャートには、ダイヤルプロットエリア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 Foreground...**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。

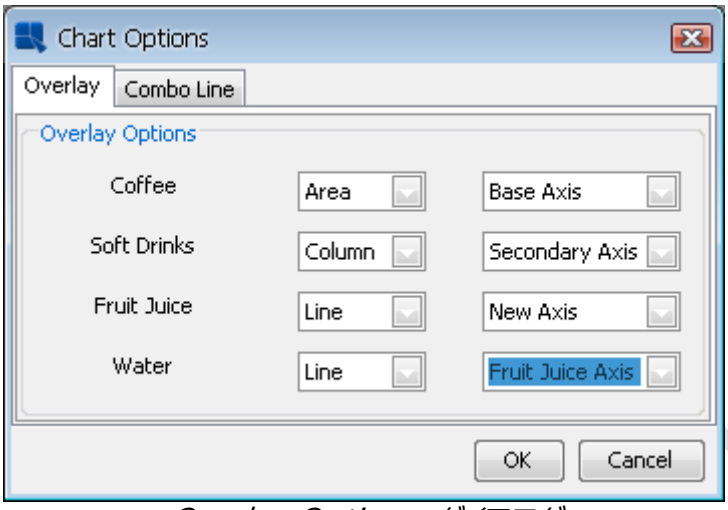


Dial Chart Background Image ダイアログ

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(背景イメージを参照してください)。Dial 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エリア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。

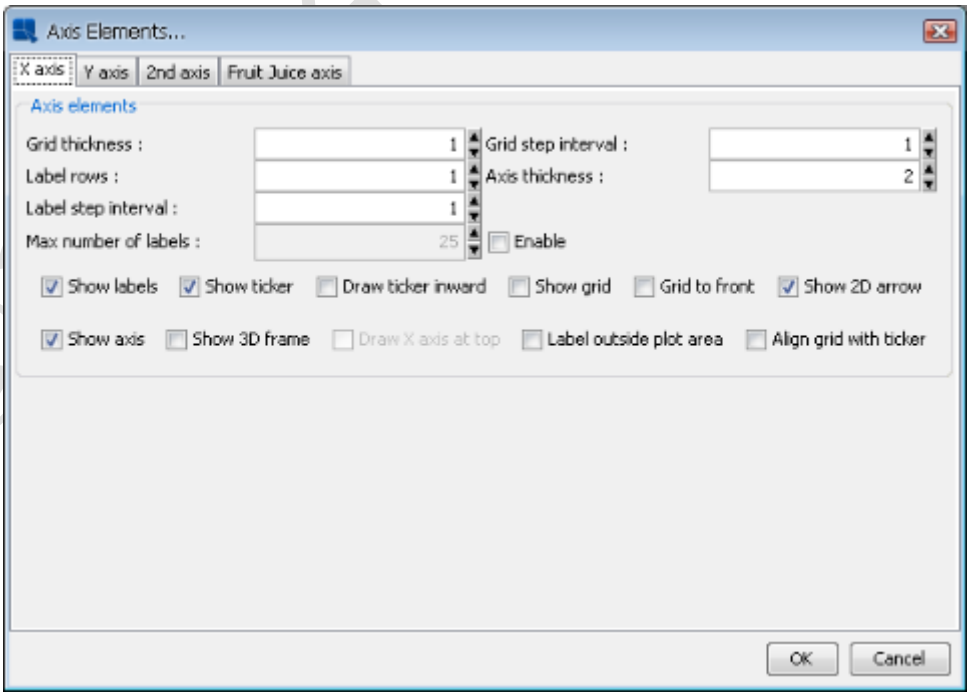
8.9.3 重ね合わせチャート

重ね合わせチャートでは、次のダイアログが表示されます。

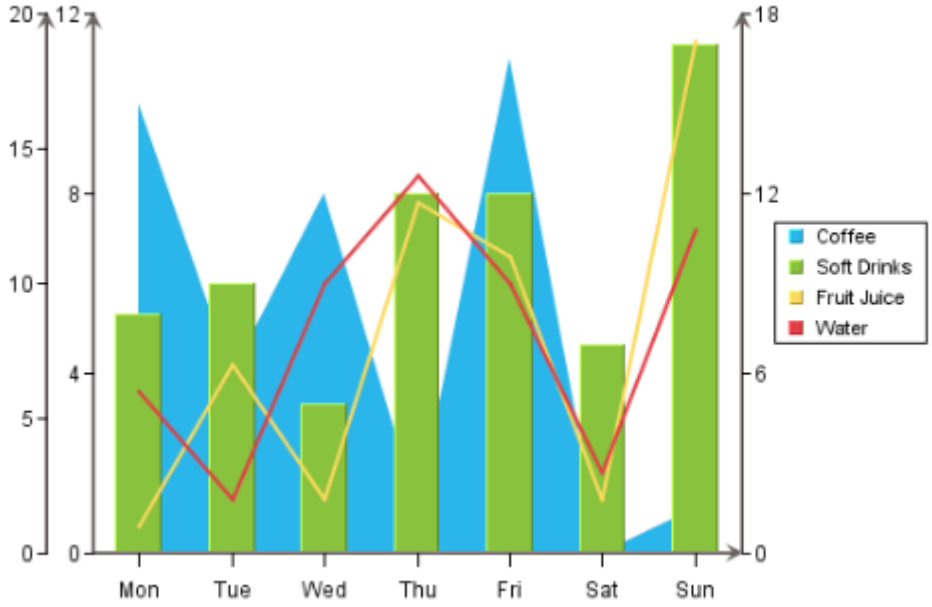


Overlay Options ダイアログ

このダイアログでは、データシリーズの各要素に使用するグラフの種類を指定できます。重ね合わせチャートのシリーズ要素に使用できるグラフの種類は、列、エリア、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、シリーズ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、他のデータシリーズ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。

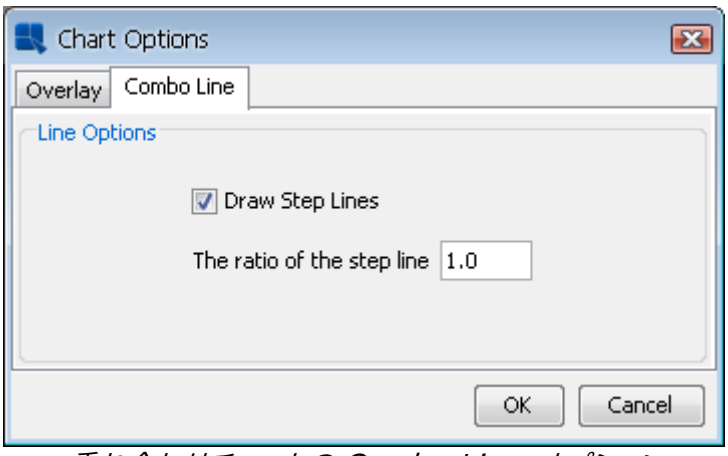


複数の値の軸の Axis Options ダイアログ



複数の値の軸を用いた重ね合わせチャート

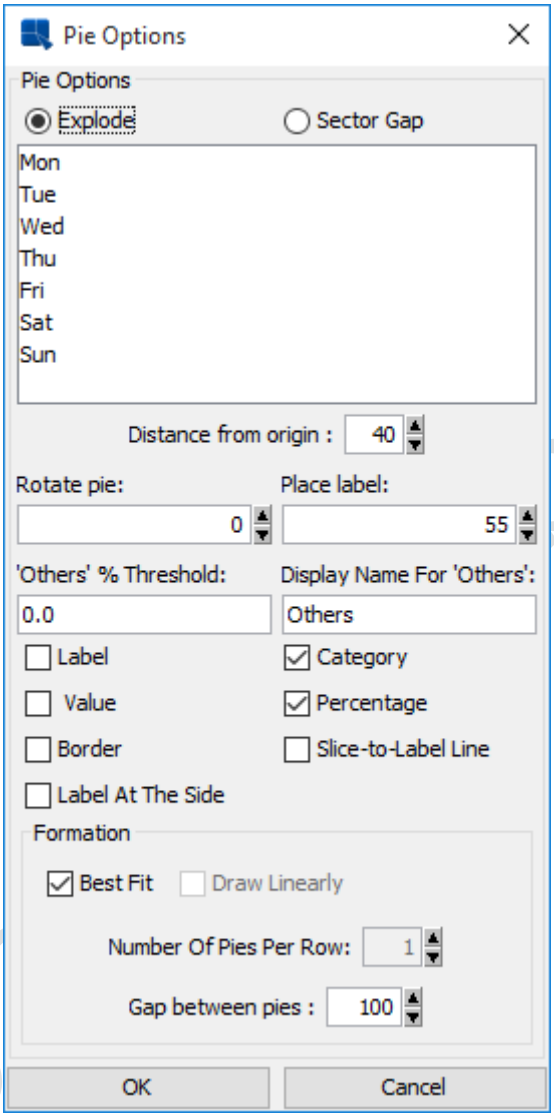
重ね合わせチャートで使用されるチャートタイプの 1 つがラインである場合、**Combo Line** タブを使用してラインをステップラインとして描画するかどうかを指定できます。ステップラインの詳細については、折れ線グラフを参照してください。



重ね合わせチャートの Combo Line オプション

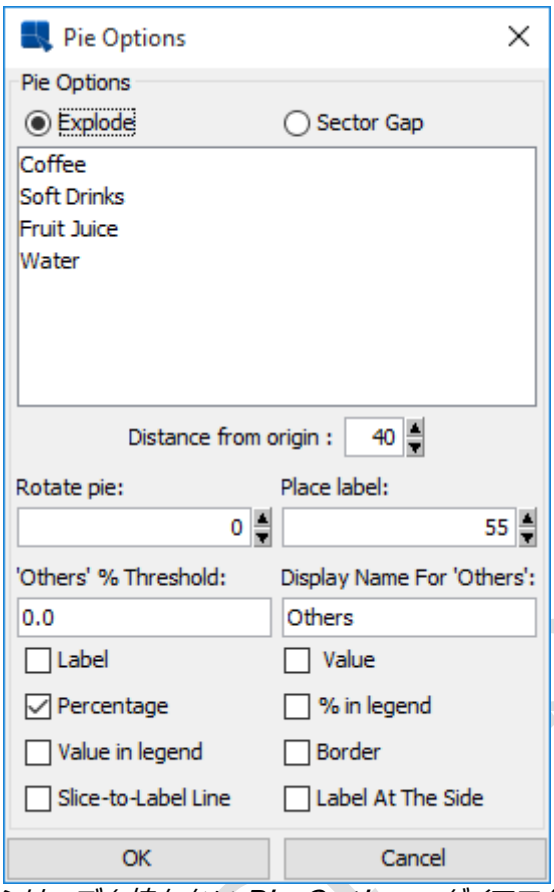
8.9.4 円グラフチャート

円グラフチャートの場合は、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。



シリーズを持たない Pie Options ダイアログ

円グラフ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、円の枠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。

“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/シリーズ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。

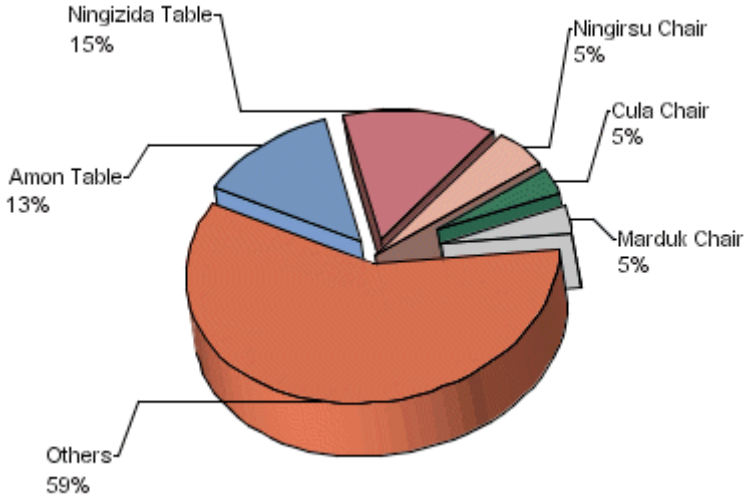
% in legend:
このオプションを使用すると、凡例の円の各スライスで表される割合を表示できます。円のスライスが薄いと、見やすくなります。このオプションは、円グラフにデータシリーズがない場合にのみ使用できます。

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、円グラフにデータシリーズ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、2D 円グラフチャートでのみ使用できます。3D 円グラフチャートの場合は、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、枠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。

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円グラフチャート

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、データシリーズのある円にのみ使用できます。

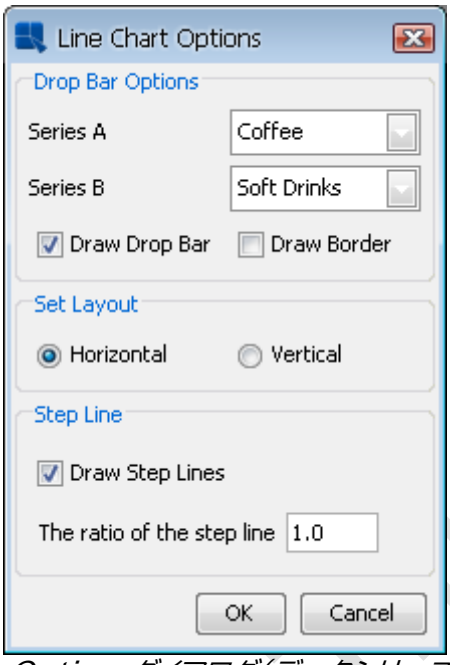
Draw Linearly:
複数の円を直線の水平線で並べます。これは、データシリーズのある円グラフにのみ使用できます。

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。

8.9.5 折れ線グラフ

2D 折れ線グラフの場合、グラフにデータシリーズ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。グラフにデータシリーズがある場合、次のダイアログが表示されます。



Line Option ダイアログ(データシリーズあり)

データシリーズを持つ折れ線グラフには、2 つのシリーズ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シリーズ要素を指定します。

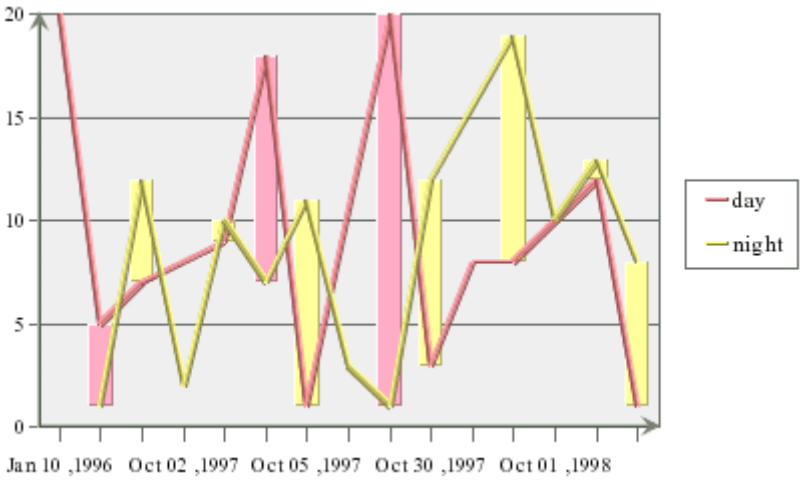
Series B:
ドロップバーの 2 番目以降のシリーズ要素を指定します。

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。

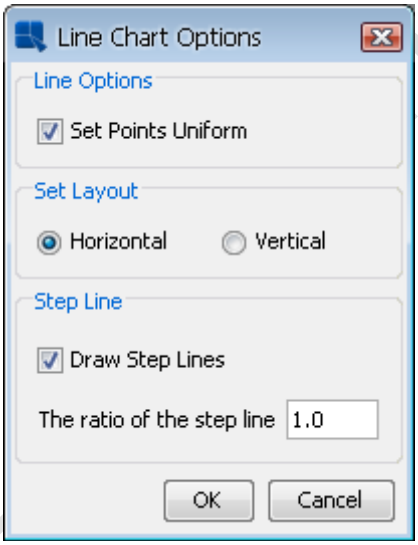
Set Layout:
縦横どちらの向きで折れ線グラフを描画するかどうかを指定します。

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。



ドロップバーありの折れ線グラフ

ドロップバーの色は、どのシリーズ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。折れ線グラフにシリーズがない場合、次のダイアログが表示されます。



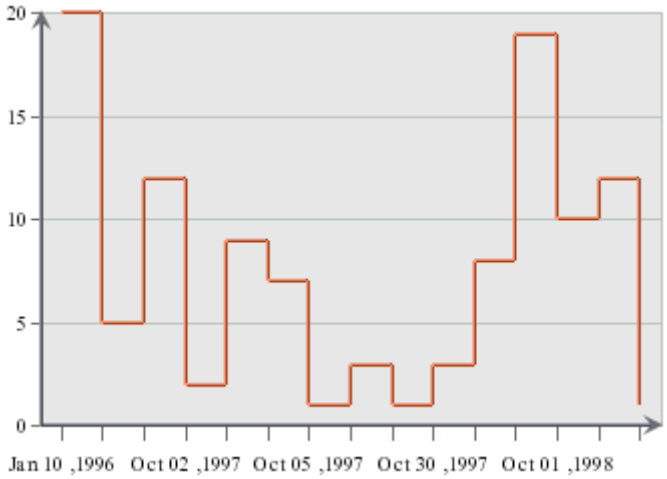
Line Options ダイアログ(データシリーズなし)

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。



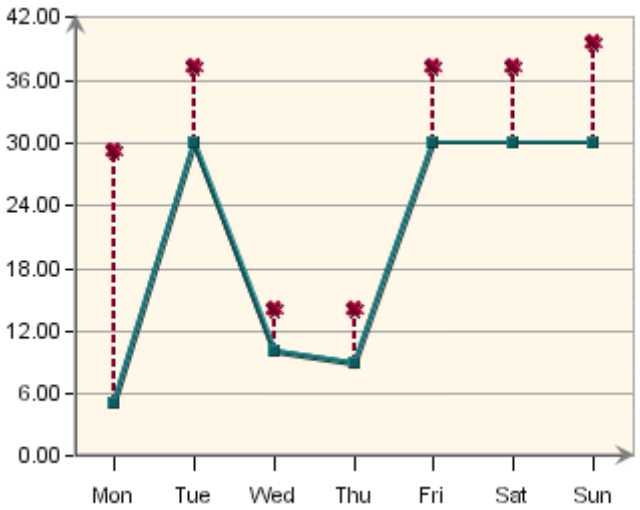
ステップラインとして描画した折れ線グラフ

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。

折れ線グラフの 3D 表示の際の追加オプションはありません。

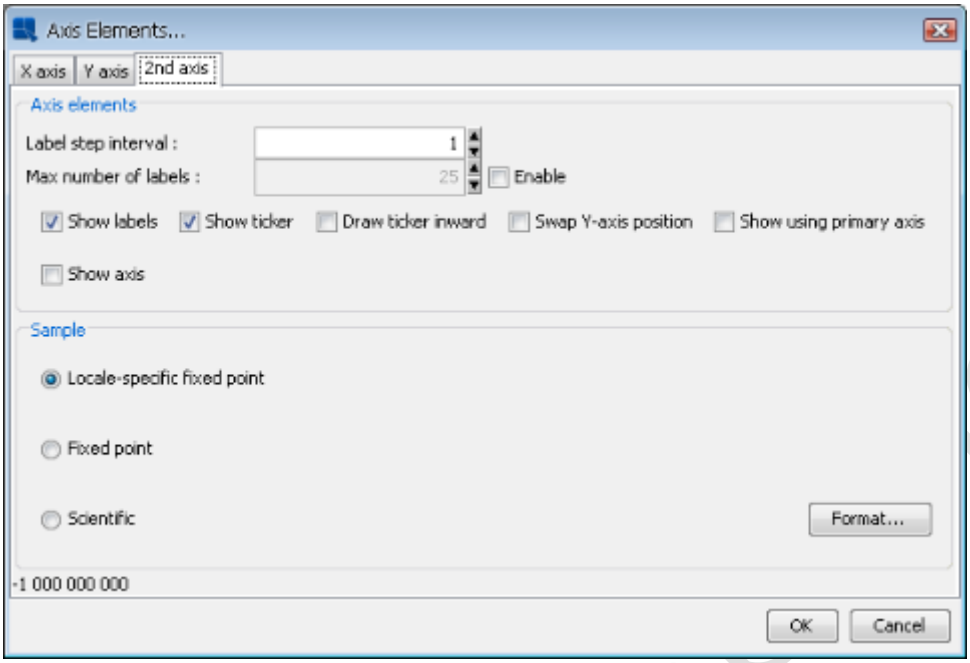
8.9.5.1 2 つの値を用いる折れ線グラフ

EspressChart には、同じ行に 2 つの値を表示できるようにする折れ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(線と点の組み合わせのように)、主軸の線と結合します。



2 つの値を用いる折れ線グラフ

2 値の折れ線グラフを作成するには、折れ線グラフ(第 1 値と第 2 値を持つ折れ線グラフ)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。

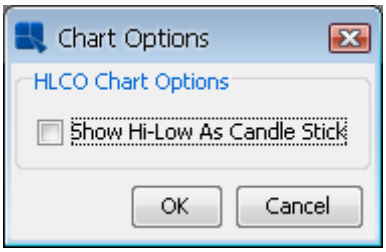


2 値の折れ線グラフの軸要素ダイアログ

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折れ線グラフとして描画されます。

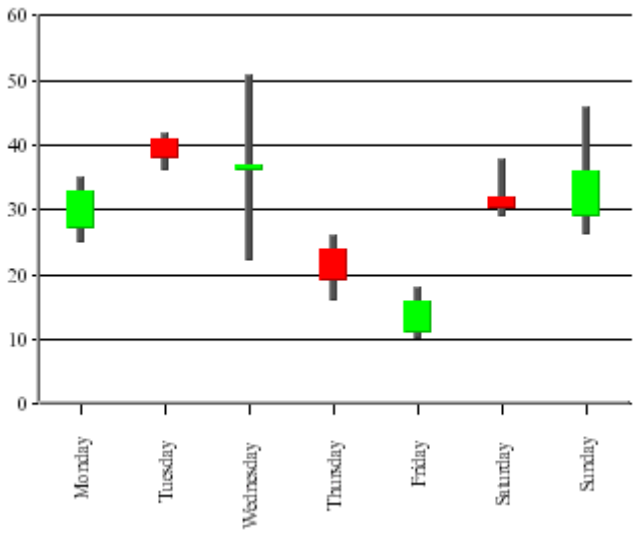
8.9.6 HLCO チャート

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

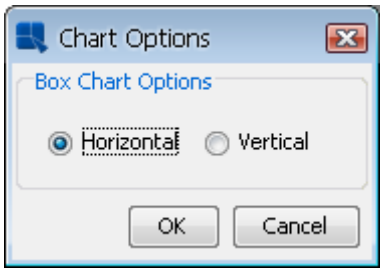
Show Hi-Low As Candle Stick オプションは、HLCO チャートをろうそく足に変えます。ろうそく足の HLCO チャートは、ローソク足に似ている単一のオブジェクトに始値、安値、高値、終値のデータを縦線で表示します。



ろうそく足 HLCO チャート

8.9.7 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。

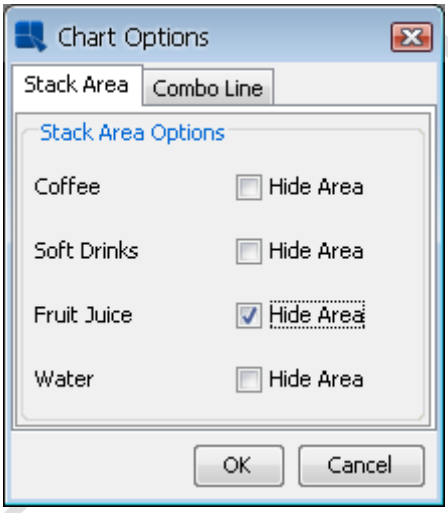


Box Options ダイアログ

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。

8.9.8 積み上げ面グラフ

積み上げ面グラフでは次のダイアログが表示されます。

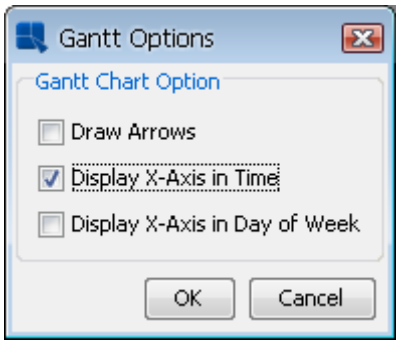


Stack Area Options ダイアログ

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、チャートがライン積み上げエリア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 積み上げ面グラフの追加オプションはありません。

8.9.9 ガントチャート

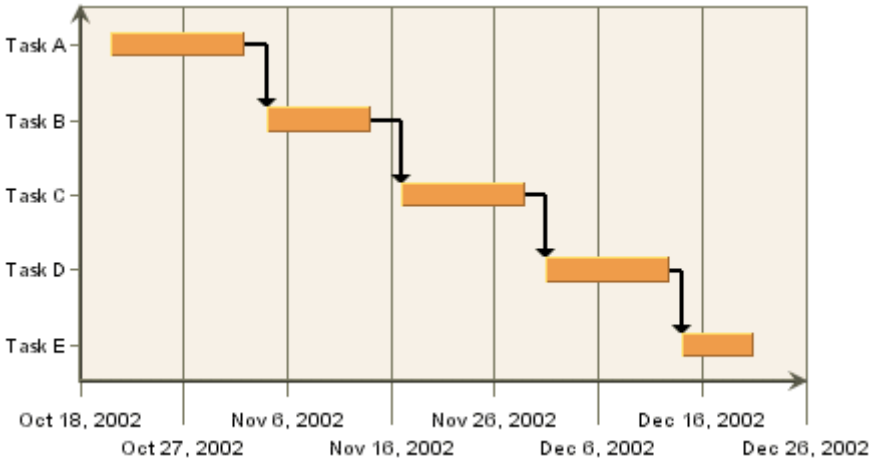
ガントチャートでは次のダイアログが表示されます。



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。



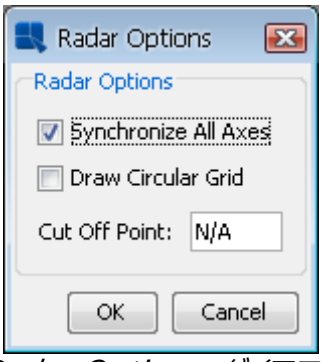
矢印のあるガントチャート

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。

8.9.10 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

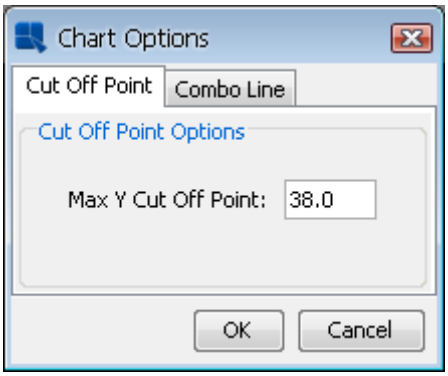
デフォルトでは、スケールはレーダーチャートのすべての軸で同じです。**Synchronize All Axes** のチェックを外すと、レーダーチャートの各軸を個別にスケーリングできます。各軸に自動スケーリングを使用するか、axis scale ダイアログを呼び出して手動でスケールを設定するかを選択できます。

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エリアは、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。

8.9.11 散布図

散布図では次のダイアログが表示されます。



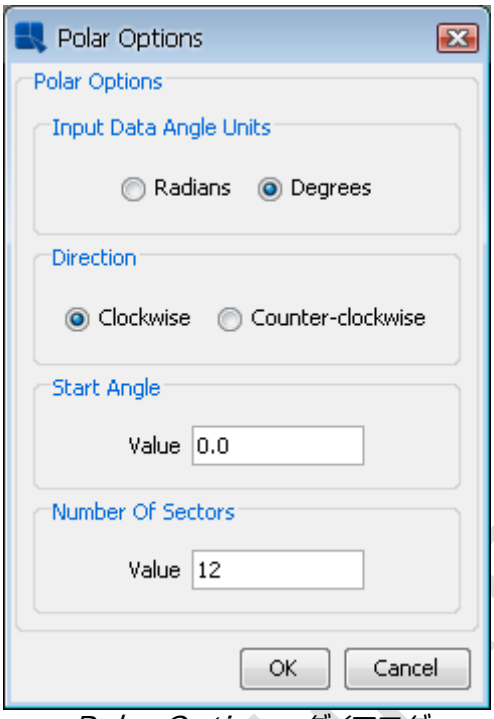
Scatter Options ダイアログ

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。

8.9.12 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。

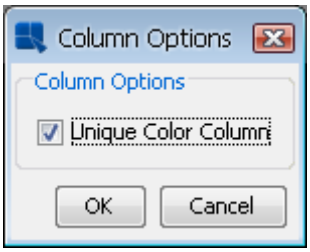
Direction:
プロットを時計回りか反時計回りに描画するか指定します。

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。

8.9.13 データシリーズを持つ縦棒グラフ

データシリーズを持つ縦棒グラフでは次のダイアログが表示されます。

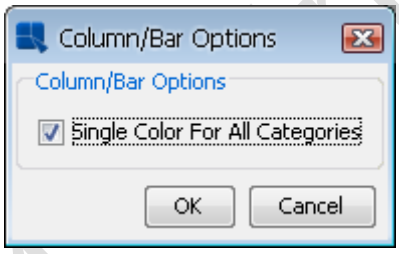


Column Options ダイアログ

通常、縦棒グラフにデータシリーズがある場合、各シリーズ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。シリーズに関係なくチャートの縦棒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、チャート内の各縦棒の色を個別に設定できます。

8.9.14 データシリーズのない縦棒グラフおよび横棒グラフ

データシリーズを持たない縦棒/横棒グラフでは次のダイアログが表示されます。

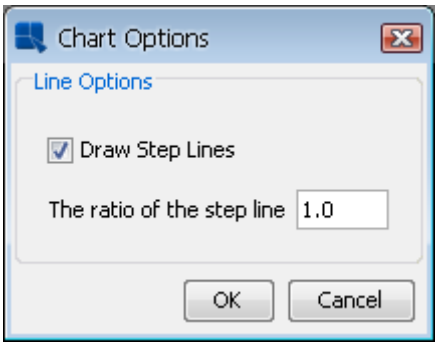


Column/Bar Options ダイアログ

通常、縦棒/横棒グラフにデータシリーズがない場合、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。

8.9.15 2D の組み合わせ折れ線グラフの作成

他の 2D 組み合わせ折れ線グラフでは次のダイアログが表示されます。

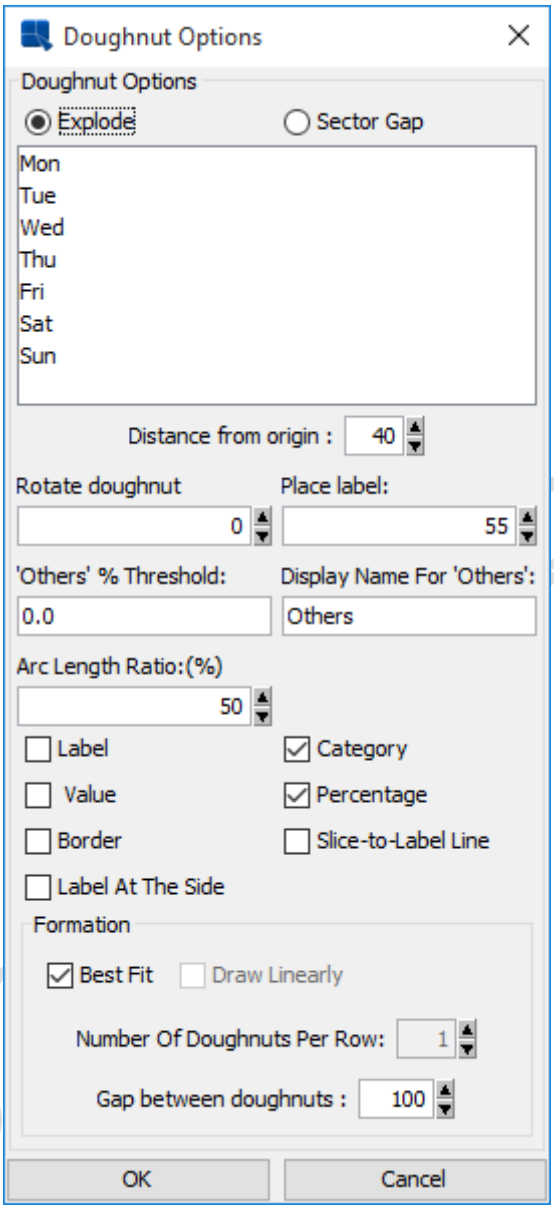


Line Combination Chart Options ダイアログ

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。

8.9.16 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、円グラフチャートのオプションとほぼ同じです。 詳細は円グラフチャート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。

8.10 MAC OS X のチャートデザイナ

Chart Designer を MAC OS X で実行すると、ほとんどの操作は Windows または Unix / Linux と同じですが、例外が 1 つあります。

右クリック:ポップアップメニューを呼び出し、チャートプロットエリアのサイズを変更するには、OS X では、実行中に右クリックするのではなく **Ctrl +クリック**を使用します。

9 ドリルダウン

グラフは、一連のデータの情報をすばやく分析して理解するための優れた手段を提供します。ただし、表示されるデータが膨大すぎるか複雑すぎるため、単一のグラフで簡単にレンダリングできないことがあります。このタイプのデータを表示する 1 つの方法は、集計されたデータのみを表示するトップレベルのチャートを作成することです。これにより、ユーザは、チャートの凡例や特定のデータポイントまたは対応するラベルをクリックして、元になるデータを見ることができます。これがドリルダウンの概念です。個別のチャートを個別に作成し、ハイパーリンクを使用して該当するチャートを適切なデータポイントまたは凡例の対応するラベルにリンクすることで、ドリルダウン効果を作成することができます。しかしながら、このアプローチは、データ点の数が少ない場合にのみうまくいきます。たとえば、20 のデータポイントを持つトップレベルチャートがあるとします。1 つのレベルのドリルダウンでは、データポイントごとに 1 つずつ、20 のチャートを作成する必要があります。サブレベルチャートに 15 のデータポイントがあるとします。別のレベルのドリルダウンを追加するには、20 x 15 のグラフまたは 300 の別々のグラフを作成する必要があります。この問題を回避するために、EspressChart にはいくつかの組み込みドリルダウンメカニズムが組み込まれており、ドリルダウンの各レベルに 1 つのチャ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。

9.1 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。

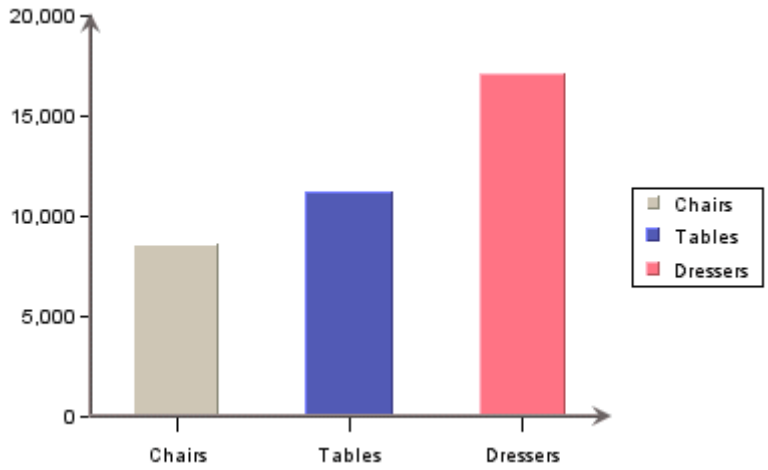
たとえば、チャートデータとして以下の表があるとします。

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。

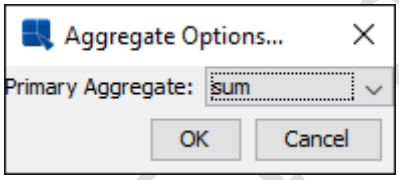
9.1.1 データドリルダウンの追加

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。



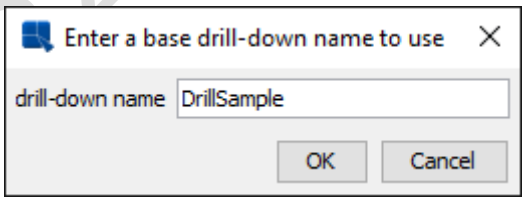
トップレベルチャート

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、最大、平均、合計、カウント、最初、最後、平方和、分散、標準偏差、および重複を除いたカウント(countdistinct)です。



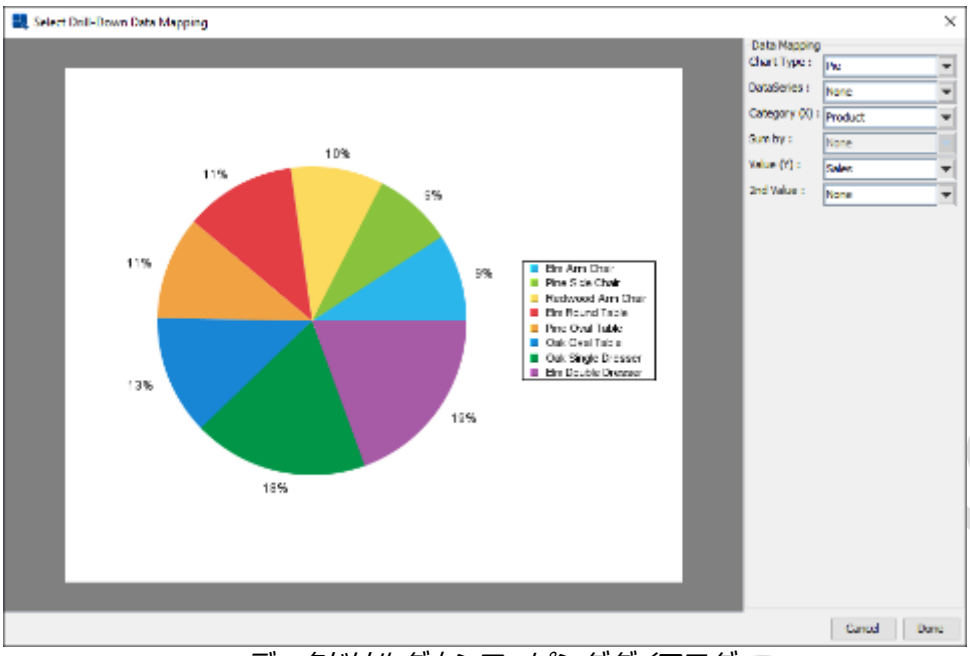
集計ダイアログ

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。



ドリルダウン名ダイアログ

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のマッピングを指定するよう求められます。



データドリルダウンマッピングダイアログ

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、縦棒、横棒、折れ線、積み上げ縦棒、積み上げ横棒、円、面積、ドーナツ、重ね合わせ、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。

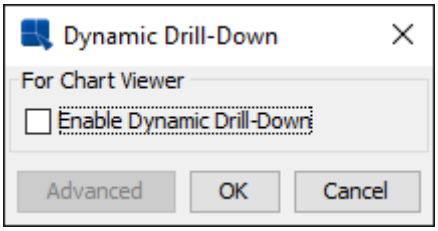
ドリルダウンのレベルを削除するには、そのレベルに移動して **Drill-Down > Remove** を選択します。**Drill-Down > Remove All** を選択することによって、ドリルダウンチャート全体を削除することもできます。**Drill-Down > Previous** への選択は、ドリルダウンチャートをより高いレベルにします。これは、右クリックしてポップアップメニューから **Back** を選択するのと同じです。**Next** を選択すると、ドリルダウンチャートが下位レベルになります。これは、チャート内の項目をダブルクリックするのと同じ機能を持ちます。

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。

9.2 動的データドリルダウン

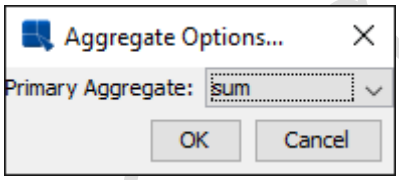
通常、ドリルダウンされるデータのマッピングオプションとドリルオプションは、設計時に固定されます。動的ドリルダウンは、マッピングを選択できる追加のオプションです。設計時に指定されるのは、最上位のグラフと集計だけです。

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。



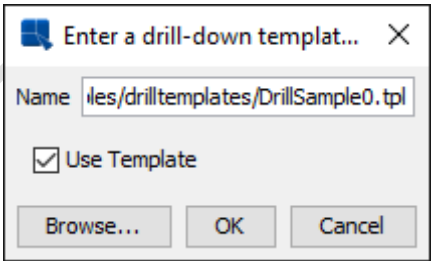
動的ドリルダウンダイアログを有効にする

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。



集計ダイアログ

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。



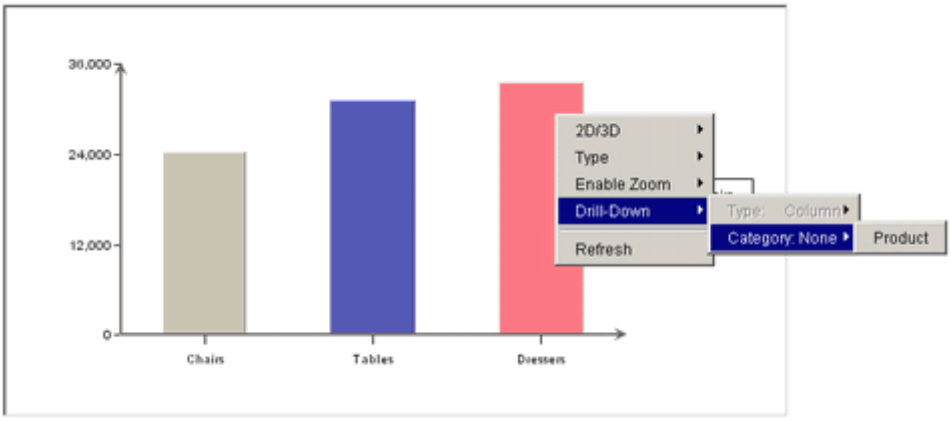
動的ドリルダウンテンプレートダイアログを選択

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。

ダイナミックドリルダウンチャートは、チャートビュアーアプレットでのみ表示できます。そこでは、デスクトップエリア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、チャートビュアー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。



チャートビュアーの動的データドリルダウン

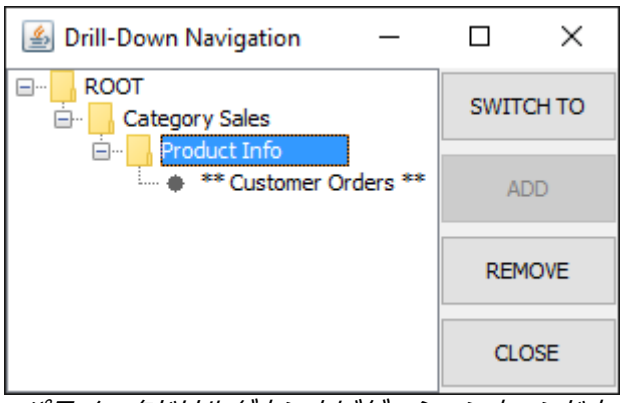
9.3 パラメータドリルダウン

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様々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、チェア縦棒をクリックすると、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。

9.3.1 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様々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、ドリルダウンの様々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。

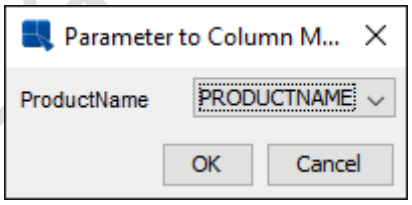


パラメータドリルダウンナビゲーションウィンドウ

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、またはシリーズの縦棒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。

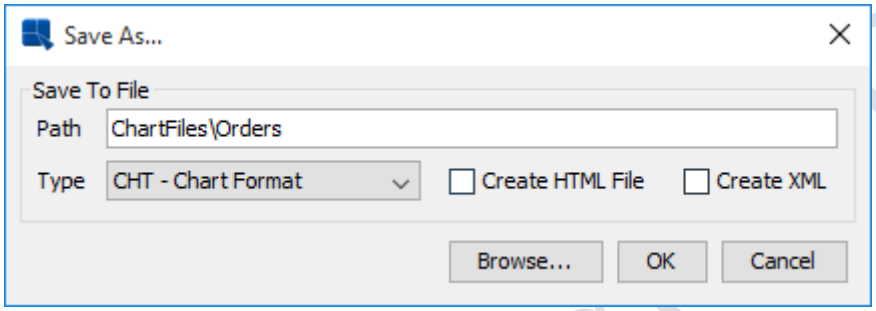
パラメータマッピングを正しく指定すると、ドリルダウンレベルの表示名を指定するよう求められます。名前を選択すると、サブレベルのグラフが Designer に表示され、カスタマイズすることができます。ナビゲーションウィンドウを使用してドリルダウンのレイヤーをナビゲートするか、データドリルダウンのようにデータポイントをダブルクリックしてレベルを下げて右クリックし、ポップアップメニューから **Back** を選択してレベルを上げることができます。

10 グラフの保存とエクスポート

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することや、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。

10.1 グラフを保存する

現在のチャートを保存するには、**File** > **Save** または **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ことや、**File** > **Save As** と、以下のダイアログが表示されます。



名前を付けて保存ダイアログ

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。

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、配置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。

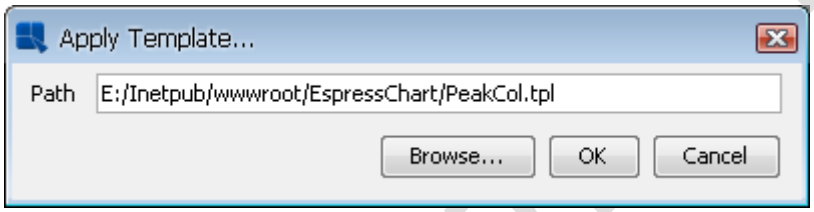
10.1.1 テンプレートの使用

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として 10 レコード分のデータを保存し、データソースが存在しないときに使用し、ファイルを開くことができます)。

テンプレートファイルは、主に 2 つの方法で使用でき、ファイルを開き、チャート描画することやエクスポートすることが可能です。この場合、最新の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デザインを作成できます。

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、チャートデザイナ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。



テンプレートの適用ダイアログ

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、フローティングライン、浮動小数点数、および軸目盛が引き継がれます。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 2D チャートである場合、3D チャートは 2D 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。

10.1.2 XML テンプレートの保存

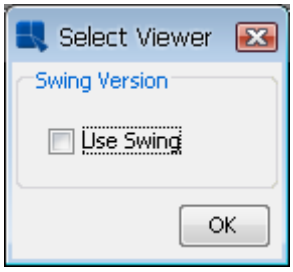
2 つのバイナリチャート形式に加えて、チャート定義を XML 形式で保存することもできます。このオプションを使用すると、Chart Designer または API の外部でグラフプロパティを変更するために使用できるテキストベースのグラフテンプレートを作成できます。

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。

10.1.3 ビューアーページの作成

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、チャートビュアー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、チャートビュアー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、EspressChart 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。



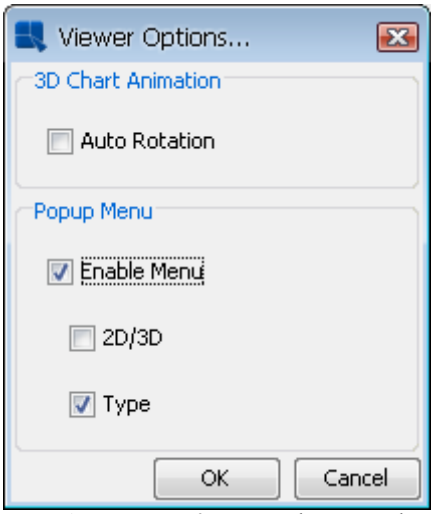
Select Viewer ダイアログ

EspresssChart は、AWT とスイングバージョンのビュアーをサポートしています。ビュアーを使用する場合は、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 ページを http プロトコルで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、EspressChart 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。

10.1.3.1 ビュアーオプション

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アー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、チャートがビュアーに表示されたときに表示されます。ビュアー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。



ビュアーオプションダイアログ

以下のオプションを使用できます。

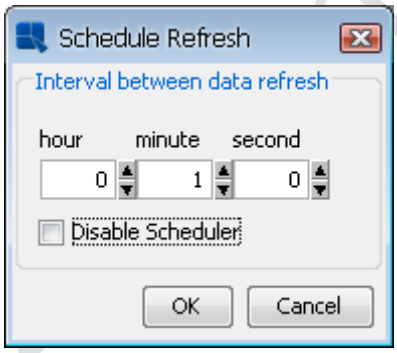
自動回転
これにより、3D チャートの自動回転が可能になります。アプレットがロードされると、チャートが回転し始めます。ナビゲーションは、ナビゲーションパネルの適切なボタンを使用して停止することができます。

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ー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。

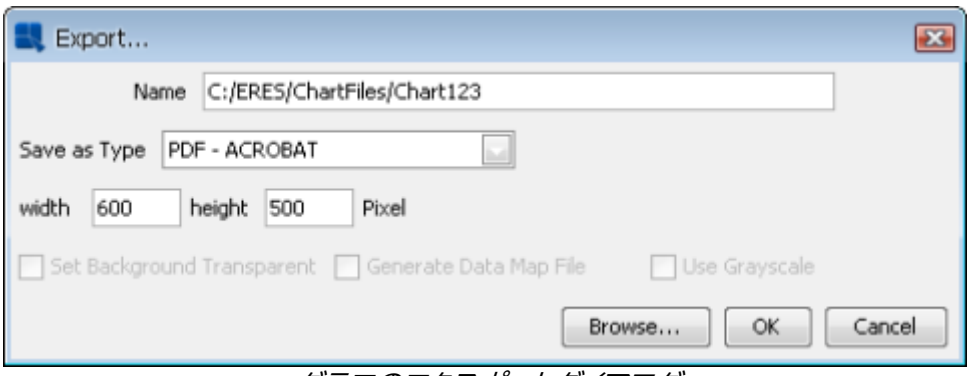


スケジュール更新ダイアログ

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。

10.2 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。



グラフのエクスポートダイアログ

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
EspressChart 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。

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。

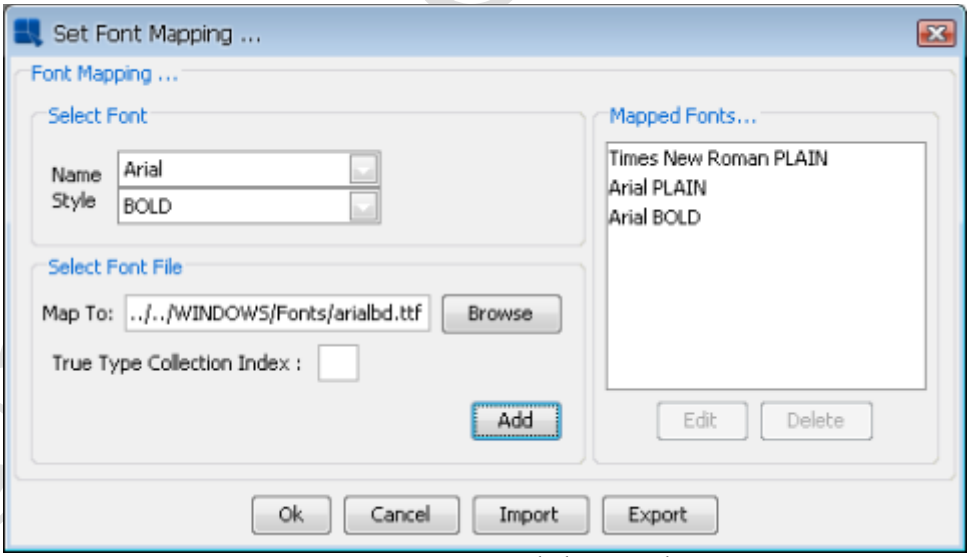
TXT
これにより、チャートのデータを含むテキストデータファイルが生成されます。

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。

10.2.1 PDF フォントマッピング

EspresssChart では、システム上の任意のフォントをチャート用に使用できます。ほとんどのイメージエクスポート(GIF/JPEG/PNG)フォントは、生成されたイメージに書き込まれます。ただし、PDF 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**(True Type フォント)、**.ttc**(True Type 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。



フォントマッピングダイアログ

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。

10.2.1.1 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。

11 チャートビュアー

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。チャートビュアーを使用すると、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に提供することができます。

チャートビュアー内で、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。3D 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 3D 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 Pure 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>
```

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)。

11.1 チャートビュアーのパラメータ

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アー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することや、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(2D および 3D 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

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、セカンダリ軸目盛を表示します(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

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

3D 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枠線を描く)、**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転置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

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、データを転置する

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し、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。

11.2 チャートビュアーのパラメータ

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、チャートビュアーのデータソースを指定できます。

11.2.1 データベースから読み取られたデータ

これは、チャートビュアーを使用して、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列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。HLOC または High Low チャートの場合、

series、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、表示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
- 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
- 2D Dial
- 2D Gantt
- 2D Polar

11.2.2 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>
```

11.2.3 引数から読み取ったデータ

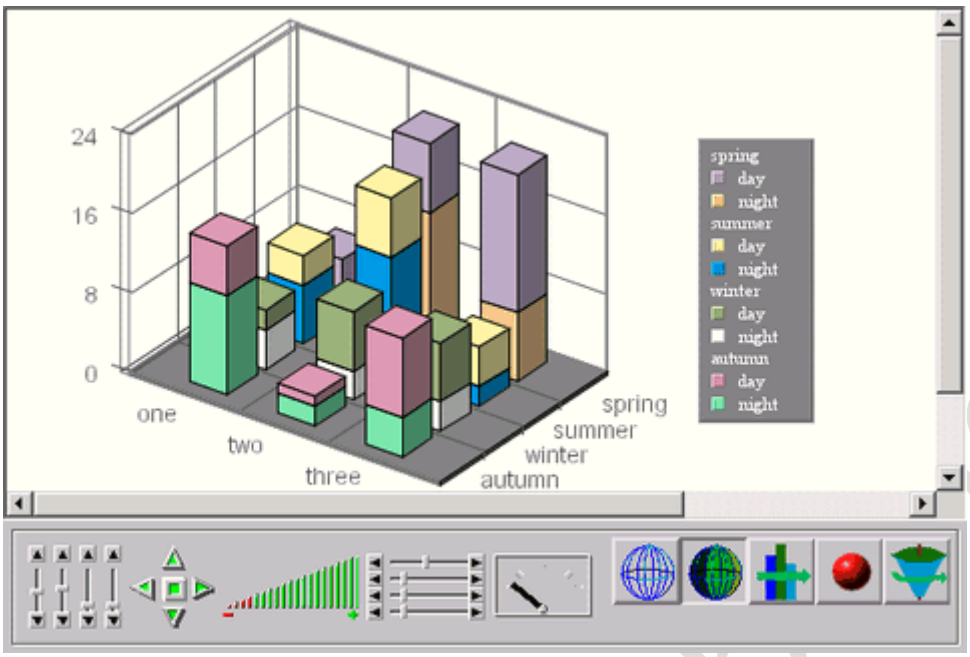
チャートビュアーをデータファイルまたはデータベースからではなく、HTML ファイルから直接読み込むことができます。

```
<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”I”で終わる点が異なります。

11.3 チャートビュアーの使用

チャートビュアー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。3D グラフの場合、チャートビュアー画面は、**drawing panel** と **navigation panel** の 2 つのセクションで構成されています。2D 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エリアに配置されています。



サンプルチャートビュアーの表示

以下は、チャートビュアー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

  棒、折れ線、ポイント、円グラフ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

  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
- プロットエリア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
- 左ボタンを**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

11.4 軸ルーラー

チャートビュアー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。

11.5 パラメータサーバ

パラメータサーバを使用すると、チャートビュアーは 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”**)。

11.6 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、時シリーズデータズームなどがあります。

11.6.1 グラフの寸法とタイプを変更する

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 3D 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。

11.6.2 軸ズーミング

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、2D チャートのみ使用できます。バブル、ダイアル、円グラフ、極座標、レーダー、散布図、積み上げエリア、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。

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。

11.6.3 ズーミング

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・時刻に基づくズームを参照してください。ズームを有効にすると、加減、上限などの様々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。

11.6.4 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。

11.6.5 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。

11.7 使用可能なスイングバージョン

チャートビューアーの JFS/Swing バージョンは、**EspressChart/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 ビューアーを使用するときは、次のクラスを呼び出します：**quadbase.chartciewer.Viewer.class**

12 EspressChart API

12.1 導入とセットアップ

EspressChart デザイナで、チャートテンプレートをデザインしたり作成したりするだけではなく、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イス(API)を提供し、ユーザはアプリケーションやアプレット内で 2D チャートや 3D チャートを作成してカスタマイズできます。これは、100%Pure Java で書かれているため、ほとんどまたは全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。

QbChart クラスは、チャートを作成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部分で、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、**EspressChart/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問合せ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。
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問合せ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。また、CLASSPATH に **qblicense.jar** も含める必要がある場合があります。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 EspressChart root directory>¥lib
(Solaris の場合) export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<path to EspressChart root directory>/lib

12.2 Chart API を使用するための推奨されるアプローチ

EspressChart 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。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そのテンプレートのデザインが適用されます。したがって、デザインのほとんどは、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。

EspressChart Chart API を使用する場合、EspressManager に接続するか、接続しないかを選択できます。Designer を使用する場合は接続が必要ですが、EspressChart 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はテンプレートを使用します(対応する章でダウンロードできます)。EspressManager には接続しません。

また、EspressChart Chart API を使用するアプレットを使用する場合は、ブラウザに少なくとも 1.5 の JVM プラグインが必要であることにも注意してください。

12.3 EspressManager とのインタラクション

EspressChart は、通常、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**クライアント**上で実行されている可能性があり、**データベースマシン**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、**データベースマシン**はファイアウォールや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**クライアント**から**データベースマシン**へアクセスすることが許可されていない可能性があります。EspressManager は**サーバ**上で実行でき、アプレット/アプリケーションは**サーバ**上の EspressManager に接続できます。JDBC を使用して**サーバ**から**データベースマシン**に接続し、データを取得できます。次に、データが**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(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 EspressChart 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、多少の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。

```
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。次の章にて Espress Manager への接続方法を解説します。

12.4 EspressManager への接続

12.4.1 EspressManager をアプリケーションとして実行

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)をリスニングします。

これらのメソッドは **QbChart** と **QbChartDesigner** に存在することに注意してください。

12.4.2 EspressManager をサーブレットとして実行

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 b);
public static void setServletRunner(String comm_url);
public static void setServletContext(String context);
```

たとえば、次のコードがあります:

```
QbChart.useServlet(true);
QbChart.setServletRunner("http://someMachine:somePortNumber");
QbChart.setServletContext("EspressChart/servlet");
```

http://someMachine:somePortNumber/EspressChart/servlet で動作する EspressManager に接続します。

これらのメソッドは QbChart と QbChartDesigner に存在することに注意してください。

12.5 API の使用

このセクションでは、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、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。

特に記載がない限り、すべての例では、**<EspressChartInstall>/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**<EspressChartInstall>/lib** ディレクトリにあります。

また、すべての API 例では、マニュアルのコアコードが表示されます。実行例をコンパイルするには、CLASSPATH に **EspressAPI.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。

12.5.1 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、チャートのルックアンドフィール）、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 ファイルを読み取り、グラフを再構成します:

```
Component doOpeningTemplate(Applet parent) {

        // EspressManager は使用しない
        QbChart.setEspressManagerUsed(false);

        //テンプレートを開く
        QbChart chart = new QbChart (parent, // container
                        "OpeningChartTemplate.cht"); // template

        //ビューアにグラフを表示

        return chart;

}
```

フルソースコード
エクスポートされた結果
この例のコンストラクタは次のとおりです:

```
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 文字列として、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。

12.5.1.1 パラメータ化されたグラフ

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、データが制限されます。

クエリパラメータは、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とき:

```
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。

12.5.1.1.1 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ボックス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。

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。

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 は使用しない
        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.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;

        //テンプレートを開く
        QbChart chart = new QbChart(parent, // container
                        "ObjectArray.cht", // template
                        queryParams); // Query Parameters

        //ビューでグラフを表示
        return chart;
}
```

フルソースコード
エクスポートされた結果

12.5.2 チャートテンプレートの適用

QbChart オブジェクトを最初から作成する場合(チャートの作成)、デフォルトの属性を使用してチャートが作成されます。そして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の種類を除く)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 **applyTemplateFile(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
        QbChart chart = new QbChart (parent, // container
                        QbChart.VIEW2D, // chart dimension
                        QbChart.BAR, // chart type
                        data, // data
                        columnMapping, // column mapping
                        templateLocation); // template

        // ビュアーにチャートを表示
        return chart;

}
```

フルソースコード
エクスポートされた結果

上記のコードは完全ではなく、ガイドとしてのみ存在します。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、**ColInfo** 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。

12.5.3 データソースの変更

Designer でチャ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 **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、テンプレートに必要なルックアンドフィールがすべてあるものの、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、グラフ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、**QbChart** オブジェクトは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。

12.5.3.1 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。使用するクエリと同様に、データベース接続情報を提供し、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 は使用しない
	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 {
		//データソースの切り替え
		chart.gethInputData().setDatabaseInfo(newDatabaseInfo);

		//チャートのリフレッシュ
		chart.refresh();

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

	}

	//ビュアーにグラフを表示
	return chart;

}
```

フルソースコード
エクスポートされた結果

上記のコードは完全ではなく、ガイドとしてのみ存在します。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。

12.5.3.1.1 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 接続情報を指定して、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 は使用しない
	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 {
                //データソースの切り替え
                chart.gethInputData().setDatabaseInfo(newDatabaseInfo);

                //チャートのリフレッシュ
                chart.refresh();

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

        }

        //ビュアーにグラフを表示
        return chart;

}
```

フルソースコード
エクスポートされた結果

上記のコードは完全ではなく、ガイドとしてのみ記載されています。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。

12.5.3.2 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う限り、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)。

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テキストファイルに切り替えます:

```
Component doChartSwitchToDataFile(Applet parent) {

        //EspressManager は使用しない
        QbChart.setEspressManagerUsed(false);

        //テンプレートを開く
        QbChart chart = new QbChart (parent, // container
                        "../templates/ChartSwitchToDataFile.cht"); // template

        try {
                //データソースの切り替え
```

```
            chart.gethInputData().setDataFile(....);

            //チャートのリフレッシュ
            chart.refresh();

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

        }

        //ビュアーにグラフを表示
        return chart;

}
```

フルソースコード
エクスポートされた結果

上記のコードは完全ではなく、ガイドとしてのみ記載されています。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。

12.5.3.3 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(詳細については XML データソースのデータを参照してください)。その後、**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 {
            //データソースの切り替え
            chart.gethInputData().setXMLFileQueryInfo(newData);

            //チャートのリフレッシュ
            chart.refresh();

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

```
        }

        //ビュアーにグラフを表示
        return chart;

}
```

フルソースコード
エクスポートされた結果

上記のコードは完全ではなく、ガイドとしてのみ記載されています。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。

12.5.3.4 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** インターフェイスを使用して **QbChart** 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 を使用しないでください
        QbChart.setEspressManagerUsed(false);

        //テンプレートを開きます
        QbChart chart = new QbChart (parent, // container
                        "ChartSwitchToCustomData.cht"); // template

        try {
                //データソースの切り替え e
                chart.gethInputData().setClassFile(..);

                //チャートのリフレッシュ
                chart.refresh();

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

        }

        //ビュアーにグラフを表示する
        return chart;

}
```

フルソースコード
エクスポートされた結果

上記のコードは完全ではなく、ガイドとしてのみ記載されています。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。

12.5.3.5 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(Applet parent) {

	//EspressManager は使用しない
	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(...);

	try {
		//データソースの切り替え
		chart.gethInputData().setData(newData);

		//チャートのリフレッシュ
		chart.refresh();

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

	}

	//ビュアーにグラフを表示
	return chart;

}
```

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。

12.5.4 グラフ属性の変更

EspressChart 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、インターフェイ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ーフェイスのハンドルを取得し、そのインターフェイス上のメソッドを呼び出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。ほとんどのインターフェイスは、quadbase.util および quadbase.ChartAPI パッケージに含まれています。

12.5.4.1 色、フォントの変更

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、適切なインターフェイスを取得し、その中のメソッドを呼び出すことによってプロパティを変更することによって、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 型のオブジェクトであるチャート
```

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
```

シリーズに基づいて、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(シリーズが定義されていない場合)。たとえば、チャートがシリーズ内に 5 つの要素を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、シリーズ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(シリーズ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、適切なインターフェイスを呼び出して、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、インターフェイスのメソッドを使用して設定できます。

12.5.4.2 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、明示的にパターンを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。定義済みパターン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、EspressChart は **QbPattern** プロパティを無視し、**Color** と同様に使用します。

次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ます。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(colors[2], QbChart.PATTERN_THICK_FORWARD_DIAGONAL);
dataColors[2] = dataPattern;
chart.gethDataPoints().setColors(dataColors);
```

フルソースコード
エクスポートされた結果

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** クラスおよび **quadbase.common.swing.color.PatternImage** クラスで定義されています。**QbChart** は **IMiscConstants** インターフェイ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 **Designer** で各パターンの見た目を確認することもできます。

12.5.4.3 カスタマイズされたパターンの設定

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れません。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。

次のコードは、アプレットまたはアプリケーションとして実行可能であり、カスタムパターンをデータポイントに適用することができます。

```
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
エクスポートされた結果

上記のコードでは、パターン ID を指定せずに **QbPattern** オブジェクトを作成しています。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。

12.5.4.4 サイズの変更

チャート作成の際に 2 つ考慮する必要があるサイズがあります。

Relative Size(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(0.7 x 300)ピクセルになり、幅は 240 ピクセル(0.8 x 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(0.7 x 600)になり、幅は 400 ピクセル(0.8 x 500)になります。相対サイズはキャンバスの高さと幅に依存します。

Absolute Size(絶対サイズ)

キャンバスサイズのピクセルに対して関係なく絶対的にサイズを指定します。

チャートのすべての要素(size パラメータが使用されている要素)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.85 (85%)に、グラフのプロットの幅を.65 (65%)に設定しています。

```
chart.gethChartPlot().setRelativeHeight(.85f);      // QbChart 型のオブジェクトのチャート
chart.gethChartPlot().setRelativeWidth(.65f);       // QbChart 型のオブジェクトのチャート
```

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 型のオブジェクトのチャート
```

同様に、他の要素のサイズは、インターフェイスのメソッドを使用して設定することができます。

12.5.4.5 日付、時刻ズームチャートの変更

Designer 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**インターフェイスを使用)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。

次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、日付/時刻ズームチャートを変更できます。

```
// 開始・終了期間とスケールを変更
IZoomInfo zoomInfo = chart.gethZoomInfo();

// 開始日
Calendar beginDate = new GregorianCalendar(2001, 0, 1);

// 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

エクスポートされた結果

上のコードでは、chart は **QbChart** 型のオブジェクトです。チャートのズームプロパティを変更した後は、変更したプロパティを有効にするためにチャートをリフレッシュする必要があります。

12.5.5 チャートのエクスポート

チャートに含まれる表は、JPEG、GIF、PNG 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、チャートのグラフは JPEG 形式でエクスポートされます。**ChartChartObject** オブジェクトの作成時に、イメージタイプを指定して **setImageType(int)**メソッドを使用して、チャートの書式を変更することができます。

EspressChart 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、チャートを独自の.cht または.tpl 形式にエクスポートすることもできます。.cht はすべてのチャート情報を格納しており、静的画像ファイルと同等の EspressChart と見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リフレッシュでき、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、EspressChart Viewer または EspressChart 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でさまざまなオプションでエクスポートするには、さまざまな方法があります。最も簡単な方法は次のとおりです。

ファイル形式	コンポーネント
Bitmap	QbCharrt.BMP
GIF	QbChart.GIF
JPEG	QbChart.JPEG
PNG	QbChart.PNG
PDF	QbChart.PDF
Flash	QbChart.FLASH
Scalable Vector Graphics	QbChart.SVG
Text Data	QbChart.TXTFORMAT
XML Data	QbChart.XMLFORMAT
Windows Meta File	QbChart.WMF

選択した形式に応じて、追加の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（0～99 の数字で表されます）の選択が可能になり、チャートオブジェクトを PNG 形式でエクスポートすることで、使用する圧縮を指定することができます。オプションは、次のメソッドを使用して指定できます。

```
public export(int format, String filename, int option)
```

次のコードは、アプレットまたはアプリケーションとして実行でき、チャートの作成およびエクスポートが可能です。

```
// 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
エクスポートされた結果

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)。

```
chart.exportDataFile("TextData", QbChart.COMMA, QbChart.TXTFORMAT);      // ここでは chart 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 true に設定すると、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();
```

このコードにより、.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る場合に便利です。

12.5.5.1 レコードファイルのエクスポート

EspressChart には、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あります。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を指定します。 データ内のレコード数が指定された数を超えると、指定された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、一定の条件を満たす必要があります。満たしていない場合使用できません。

- 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 ソースでは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>);
```

12.5.5.2 ストリーミングチャート

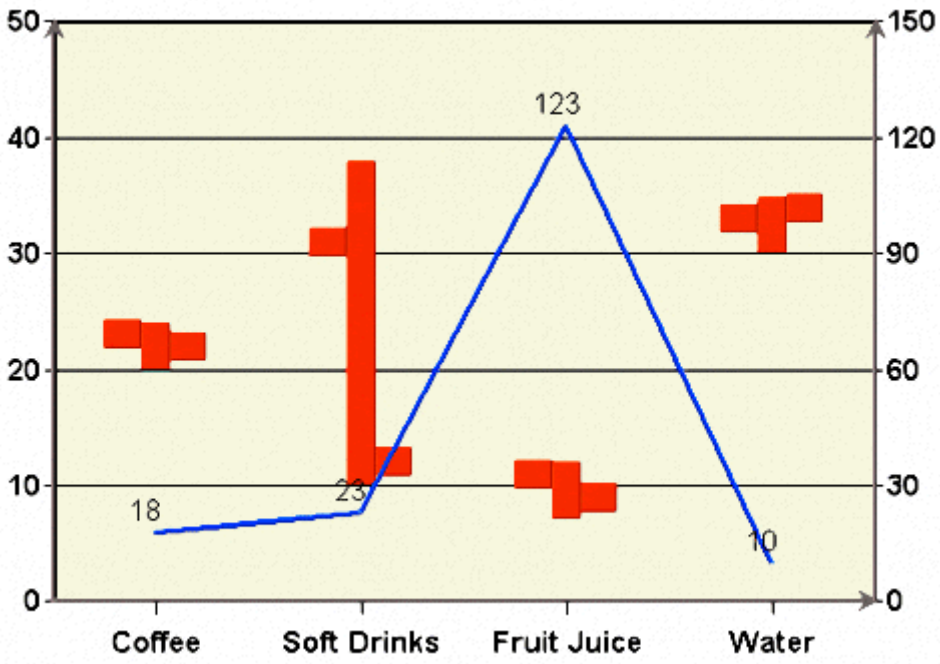
ローカルドライブへのエクスポートに加えて、チャートをバイトストリームとしてエクスポートして、Web ブラウザに直接ストリーミングすることもできます。サーバーサイドのコードは通常、イメージをバイナリ形式でのみエクスポートします。エクスポートされたイメージの周りにラッパー(つまり、DHTML / HTML コンテンツ)を作成する必要があります。

以下は、チャートを 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** 変数を絶対パスまたはアプリケーションサーバー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 は使用しない
	QbChart.setEspressManagerUsed(false);

	String chartTemplate = "StreamingChart.cht";

	// テンプレートを開く
	QbChart chart = new QbChart((Applet)null, chartTemplate);

	// チャートを PNG にエクスポートし、バイトストリーミング
	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();

}
```

フルソースコード

サンプルを実行するには、コードによってアクセスされる場所にチャートテンプレートをコピーしてください。相対パスを使用する場合(例と同様に)、カレント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。結果のチャートを以下に示します。



チャートストリーミング

12.5.6 Chart API から Chart Designer を呼び出す

Chart Designer は、アプリケーションまたはアプレットのいずれかで Chart 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、指定されたテンプレートファイル、指定したデータソースまたは他のパラメータを使用して Chart Designer を開くことができます。

Chart API から Chart 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。
- **EspressAPI.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 と 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 prper コンストラクタ内の(それらのディレクトリの)場所を渡します。

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 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、次のコードを追加する必要があります。

```
public login(String userName, String password);   // QbCharttDesigner クラス内にあるメソッド
```

EspressManager に **-QbDesignerPassword** を指定した場合、**setVisible()** または **getDesigner** メソッドを呼び出す前に、次のコードを追加する必要があります。

```
public login(String password);    // QbChartDesigner クラス内にあるメソッド
```

Chart Designer を Chart API 経由で呼び出すには、以下のような方法があります

12.5.6.1 グラフテンプレートを指定する

指定したチャートテンプレートファイルで Chart Designer を開くことができます。これにより、エンドユーザは Chart Designer GUI で独自のカスタムチャートを作成し、完成したチャ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。

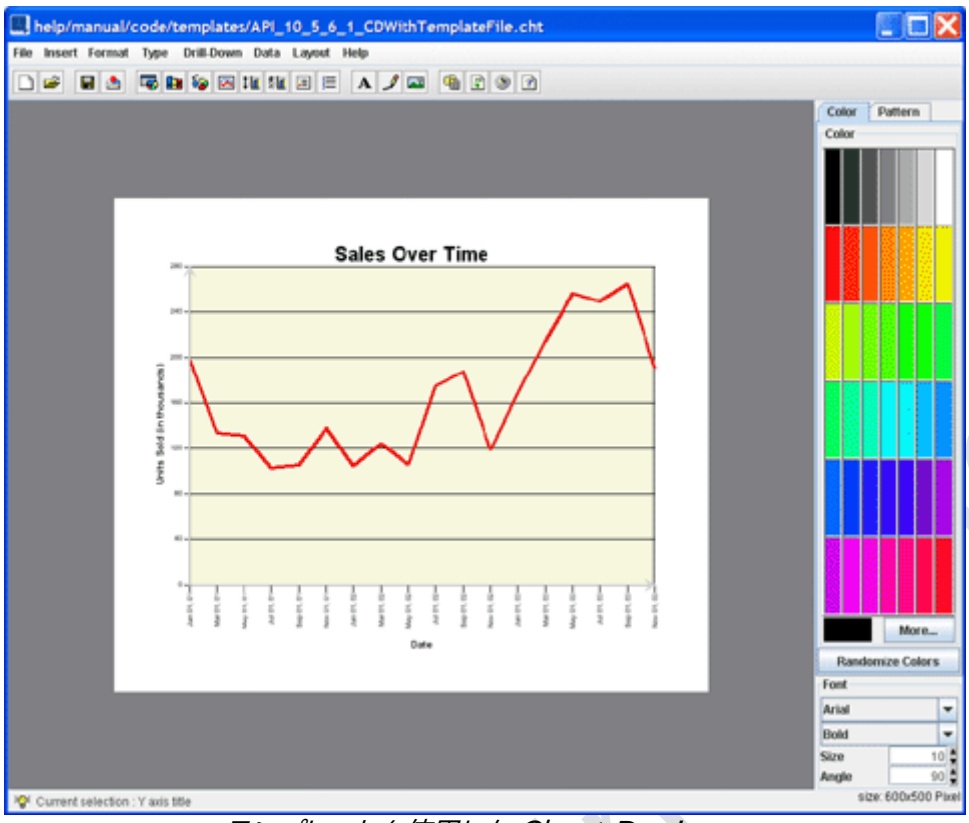
```
QbChartDesigner (Object parent, String chtFile, String[] imagesPath);
```

以下は、指定された.cht ファイルを使用して Chart Designer を呼び出す例です。

```
public void doChartDesignerApplet(Object parent)   throws Exception {

   // EspressManager に接続
   QbChartDesigner.setServerAddress("127.0.0.1");
   QbChartDesigner.setServerPortNumber(22071);

   // これらのフォルダは<InstallDir>ディレクトリに配置されています。アプリケーションに合わせて絶対パスまたは相対パスに変更します。
   String[] imagesPath = {"images", "backgroundImages"};

   // 新たな QbChartDesigner インスタンスを作成
   Designer = new QbChartDesigner(parent, "CDWithTemplateFile.cht", imagesPath);

   // デザイナを開始
   designer.setVisible(true);
}
```

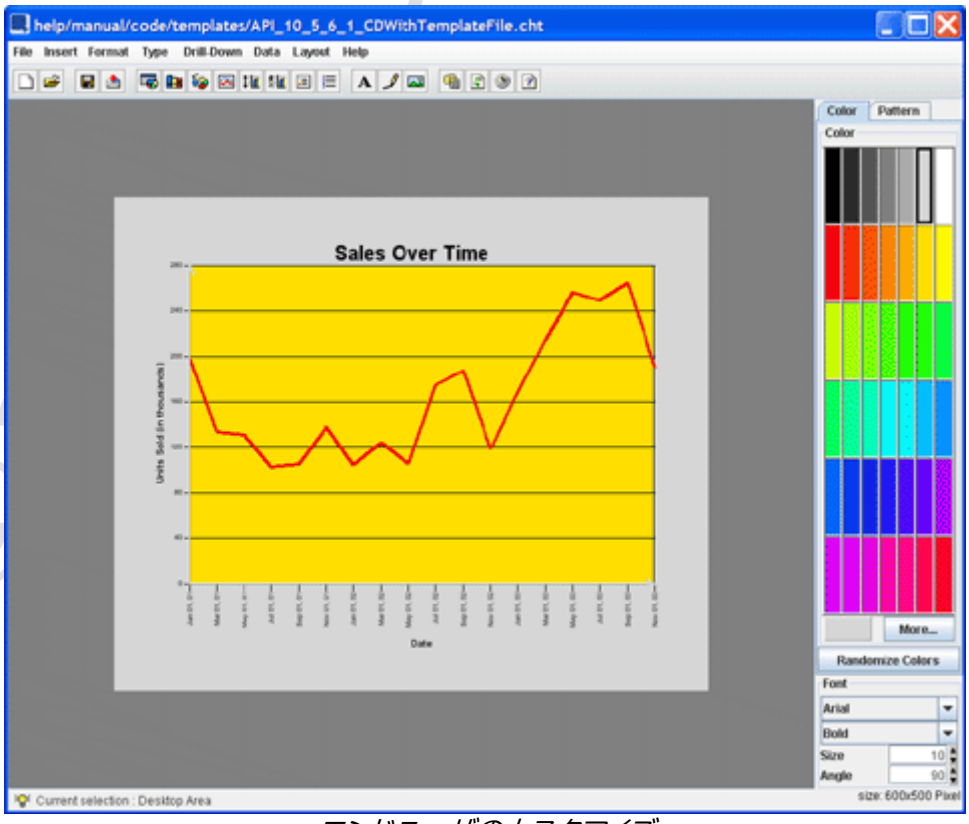
フルソースコード

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。
ユーザがこのコードを実行すると、指定したグラフテンプレートで Chart Designer が起動します。



テンプレートを使用した Chart Designer

ユーザはチャートをカスタマイズして結果を保存することができます。



エンドユーザのカスタマイズ

12.5.6.2 データレジストリを指定する

Chart Designer を開き、データソースマネージャ(データレジストリに指定された.xml ファイルを使用)

で始めることができます。これにより、エンドユーザは、データソースを選択し、Chart Designer の GUI で独自のカスタムチャートを作成し、完成したチャートを表示することができます。

次のコンストラクタが使用されます。

```
QbChartDesigner(Object parent, String nameOfXMLFile, boolean newChart, String[] imagesPath);
```

以下は、指定された.xml ファイルで Chart Designer を呼び出す例です。

```
public void doChartDesignerWithDataRegApplet(Object parent) throws Exception {
   // EspressManager に接続
   QbChartDesigner.setServerAddress("127.0.0.1");
   QbChartDesigner.setServerPortNumber(22071);

   // これらのフォルダは<InstallDir>ディレクトリに配置されています。アプリケーションに合わせて絶対パスまたは相対パスに変更します。
   String[] imagesPath = {"images", "backgroundImages"};

   // 新たな QbChartDesigner インスタンスを作成
   designer = new QbChartDesigner(parent, "DataRegistry/sample.xml", true, imagesPath);

   designer.setVisible(true);

}
```

フルソースコード

上記コードは、アプリケーションおよびアプレットとして実行可能です。

12.5.6.3 DBInfo オブジェクトの指定

Chart Designer を開き、Select Chart Type ウィザードウィンドウ(指定した **DBInfo** オブジェクトを使用)から開始できます。これにより、エンドユーザは Chart Designer GUI で独自のカスタムチャートを作成し、完成したチャ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。

```
QbChartDesigner(Object parent, DBInfo databaseInformation, QueryInParamSet inset boolean newChart, String[] imagesPath);
```

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 QbChartDesigner designer;
String url = ”jdbc:hsqldb:help/examples/DataSources/database/woodview”;
String driver = ”org.hsqldb.jdbcDriver”;
String username = ”sa”;
String password = ””;
String query = ”SELECT * FROM Products”;
```

```
public void doChartDesignerWithDBInfoApplet(Object parent) throws Exception {
   // EspressManager に接続
   QbChartDesigner.setServerAddress("127.0.0.1");
   QbChartDesigner.setServerPortNumber(22071);

   // これらのフォルダは<InstallDir>ディレクトリに配置されています。アプリケーションに合わせて絶対パスまたは相対パスに変更します。
   String[] imagesPath = {"images", "backgroundImages"};

   // 新たな QbChartDesigner インスタンスを作成
   DBInfo dbInfo = new DBInfo(url, driver, username, password, query);
   designer = new QbChartDesigner(parent, dbInfo, null, true, imagesPath);

   designer.setVisible(true);

}
```

フルソースコード

上記コードは、アプリケーションおよびアプレットとして実行可能です。

12.5.6.4 指定した DBInfo オブジェクトを使用してクエリビルダを開く

クエリビルダは、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、エンドユーザはクエリビルダ GUI で独自のカスタム SOL クエリを作成して保存することができます。

以下は、指定された DBInfo オブジェクトを使用してクエリビルダを呼び出す例です。

```
public void doQueryBuilderApplet(Object parent, String sqlName) {

        //EspressManager に接続
        QbChartDesigner.setServerAddress("127.0.0.1");
        QbChartDesigner.setServerPortNumber(22071);

        queryMain = new QbQueryBuilder(parent, this);
        if (sqlName != null) {
                //sqlName .qry ファイル名(拡張なし)
                // 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();
        } 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 QbChartDesigner(applet, dbInfo, queryMain.getInSet(), true, imagesPath);
	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)を返すことで、クエリビルドを開くこともできます。以下は、スキーマ情報を渡して、クエリビルダを呼び出す方法です。

```
public QBWithDBInfo2() {

	//EspressManager に接続する
	QbChartDesigner.setServerAddress("127.0.0.1");
	QbChartDesigner.setServerPortNumber(22071);

	try {
		Class.forName(driver);
		conn = DriverManager.getConnection(url, username, password);
		metaData = conn.getMetaData();

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

	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() {

	queryMain.getBuilder().setVisible(false);
	DBInfo dbInfo = new DBInfo(url, driver, username, password, queryMain.getSQL());
	designer = new QbChartDesigner(applet, dbInfo, queryMain.getInSet(), true, imagesPath);
	designer.setVisible(true);
```

```
}

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();

}

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として使用できます。

12.5.6.5 あらかじめ設定されたディレクトリを持つチャートデザイナを開く

チャートを読み込んで保存することもできます。この機能により、ユーザ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り、ユーザ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。

以下の例は、ブラウズするルートディレクトリを指定して Chart Designer をカスタマイズする方法を示しています。

```
public void doChartDesignerApplet(Object parent) throws Exception {
        QbChartDesigner.setServerAddress("127.0.0.1");
        QbChartDesigner.setServerPortNumber(22071);

        // これらのフォルダは<InstallDir>ディレクトリに配置されています。アプリケーションに合わせて絶対パスまたは相対パスに変更します。
        String[] imagesPath = {"images", "backgroundImages"};

        //重要:CDWithPreSetDirectories.cht ファイルを、<InstallDir>/ディレクトリに配置します。(例:C:/EspressChart/CDWithPreSetDirectories.cht)。
デザイナ=新しい QbChartDesigner(親、 "CDWithPreSetDirectories.cht"、imagesPath)
        designer = new QbChartDesigner(parent, "CDWithPreSetDirectories.cht", imagesPath);

        designer.setRootDirectoryForBrowse(".");

        designer.setVisible(true);
}
```

フルソースコード

次に、BrowseDirectories のメソッドを使用して、異なり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。

12.5.6.6 変更されたメニューバーとツールバーを使用してグラフデザイナを開く

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(Object parent) throws Exception {
        QbChartDesigner.setServerAddress("127.0.0.1");
        QbChartDesigner.setServerPortNumber(22071);

        // これらのフォルダは<InstallDir>ディレクトリに配置されています。アプリケーションに合わせて絶対パスまたは相対パスに変更します。
        String[] imagesPath = {"images", "backgroundImages"};

        designer = new QbChartDesigner(parent, "CDWithModifiedBars.cht", imagesPath);

        //新しいメニュー項目を追加する
        JMenuBar menuBar = designer.getChartMenuBar();
        JMenu fileMenu = menuBar.getMenu(0);
        newItem = new JMenuItem("Hello World");
        newItem.addActionListener(this);
        fileMenu.insert(newItem, 7);

        //ツールバーのボタンを削除
        JToolBar designBar = designer.getChartToolBar();
        designBar.remove(5);                                    // Remove Separator
```

```
        designBar.remove(4);                              // Remove Export
        designer.setSaveOnExitEnabled(false);   // Do not prompt to save the chart
if unsaved on exiting Designer
        designer.setVisible(true);
}

public void actionPerformed(ActionEvent e) {

        /***レポートを保存**********/
        designer.save("CDWithModifiedBars_Temp.cht");
        /****新しいテストフレームを作成*********/
        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として使用できます。

12.6 API のみの機能

Chart Designer のすべ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以下に説明します。一部の機能は、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、chart は QbChart 型のオブジェクトです。

12.6.1 ビジュアル

以下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。以下の各機能は、チャート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。

12.6.1.1 キャンバス・プロット

このセクションでは、API を使用してチャートプロットやキャンバスに行うことができる様々な視覚的な変更を示していて、チャートプロットやキャンバスの一般的な変更に対応しています。コンポーネント固有の変更については、それぞれのセクションで説明します。

12.6.1.1.1 プロットなしのデータのためのカスタマイズメッセージ

データがない時やプロットするデータが不十分な時に表示されるメッセージを変更できます。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");
```

詳細については、こちらを参照してください。

12.6.1.1.2 キャンバスに合わせてグラフを設定する

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。チャートをキャンバスに正しくフィットさせるには、ICanvas インターフェイスで **setFitOnCanvas(boolean b)**メソッドを使用します。

```
ICanvas canvas = chart.gethCanvas();
canvas.setFitOnCanvas(true);
```

詳細は、こちらを参照してください。

12.6.1.1.3 Chart Invisible を設定する

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** インターフェイ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、こちらを参照して下さい。

12.6.1.1.4 異なるグラフィックスレンダリングの適用

様々なレンダリング手法(アンチエイリアシング)をチャートに適用できます。これにより、あなた仕様にチャート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);
```

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、こちら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のみで使用可能です。

12.6.1.1.5 チャートプロット位置
チャートの位置を-0.5(キャンバスを帰順した x と y の位置)にすることができます。これにより、チャート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、こちらを参照してください。

12.6.1.1.6 同じプロットで複数の図を描画する
複数のチャート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チャートには、テキストとキャンバスの情報が削除されます。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。散布図チャート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きる以下のコードは、3 つのチャートを 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
                    ”AddOnChart1.cht”); // template

//他の 2 つのテンプレートを開く
QbChart chart2 = new QbChart(parent, ”AddOnChart2.cht”);
QbChart chart3 = new QbChart(parent, ”AddOnChart3.cht”);

primaryChart.setAddOnChart(new QbChart[] { chart2, chart3 });
```

フルソースコード

エクスポートされた結果

詳細は、こちらを参照してください。

12.6.1.2 ヒントボックス

このセクションでは、API を使用してチャートのヒントボックスに加えることの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ます。

12.6.1.2.1 ヒントボックスを変更する

ヒントボックスを修正することができます。つまり、ヒントボックスを表示したとき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以下のコードは、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
                    "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);

                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

エクスポートされた結果

詳細は、こちら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。

12.6.1.2.2 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
チャート内のチャートや他のコンポーネントと重ならないようにデータとハイパーリンクのヒントボックスにオフセットを設定できます。

API でこの機能を利用するには、チャート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、こちらを参照してください。

12.6.1.2.3 ヒントボックスの枠線
API を使用して、ヒントボックスの枠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、こちらを参照してください。

12.6.1.2.4 イメージマップのヒントボックスをカスタマイズする
チャートをマップファイルにエクスポートするときに、ヒントボックスのテキストをカ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、マウスを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以下のコードは、イメージマップのヒントボックスをカスタマイズ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, // container
                ”CustomizeImageMapHintBox.cht”); // template

chart.setImageMapDataHintBoxHandle(new customizeImageMap());

try {
        //チャートをイメージとマップファイルにエクスポートする
        chart.export(QbChart.PNG,                ”CustomizeImageMapHintBox”,
”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

エクスポートされたマップ

詳細は、こちらを参照してください。

12.6.1.3 凡例・注釈

このセクションでは、API を使用してチャートの凡例および、または任意の注釈に加えられる様々な視覚低変化を記述します。これにより凡例、注釈の外観ではだけではなく、内容の変更も含まれます。

12.6.1.3.1 記号により注釈

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チャート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、EspressChart 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レーションとして実行できる以下のコードは、記号を昼食と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, // container
                    "AnnotationWithSymbol.cht"); // template

//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(.65f, .7f);
anno.setRelativePosition(newPosition);
set.addAnnotation(anno);
```

フルソースコード
エクスポートされたイメージ

詳細は、こちらとこちらを参照してください。

12.6.1.3.2 凡例と注釈テキストの参照位置を設定する

凡例および注釈テキストの参照位置を左上隅または左下隅(デフォルト)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** メソッドを使用できます。

```
ICanvas canvas = chart.gethCanvas();
canvas.setReferenceAtTop(true);
```

詳細は、こちらを参照してください。

12.6.1.4 その他

このセクションでは、チャートの様々なコンポーネントに対して様々な視覚的な変更を行うことができます。これには、チャートとその要素の外観だけでなく、内容の変更も含まれます。

12.6.1.4.1 ティッカーラベルの交換

ティッカーラベルをユーザ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 **setTicketLabels** 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、こちらを参照してください。

12.6.1.4.2 カスタマイズ可能なデータトップラベル

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以下のコードは、シンボルをアノテーションと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, // container
                    "CustomizeDataTopLabel.cht"); // template

chart.gethDataPoints().setDataLabelInfo(new customizeDataTopLabel());

...

class customizeDataTopLabel implements IDataLabelInfo {

        static final long serialVersionUID = 1;

        public String getDataLabel(PickData pickData, String originalDataLabel) {
                return (pickData.toString());
        }
}
```

フルソースコード

エクスポートされたイメージ

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、こちらを参照してください。

12.6.1.4.3 軸を日付・時刻・タイムスタンプとして表示する
値の軸を持つ任意のチャートの数値軸ではなく、時間軸、日付の単位で数値軸を表示することができます。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** 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、こちらを参照してください。

12.6.1.4.4 選択的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ときは、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。EspressChart では、特定のテキストにアンチエイリアスを適用することや、アンチエイリアスを行わずに他の文字列(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、**setDisableJava2DSForStraightText** メソッドを使用します。

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setDisableJava2DForStraightText(true);
```

詳細は、こちらを参照してください。

12.6.1.4.5 水平線・垂直線・トレンドラインよりも上のデータポイントを描画する
水平、垂直、トレンドラインの上にデータポイントを描画することができます。つまり、データポイントがライン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、**ILkinePropertySet** 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、こちらを参照してください。

12.6.2 データ

以下に示すフィーチャは、チャートとそのコンポーネントにデータを処理します。以下の各機能は、表示されたデータを変更したり、チャート内のデータを取得したらします。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。

12.6.2.1 座標の取得
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、シリーズおよび存在する場合のみ Sumby 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)          //width=200、height=300 のピクセル
```

詳細は、こちらを参照してください。

12.6.2.2 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、EspressChart 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、データの制限を設定できます。

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setLimitAtAxisScale(true);
```

詳細は、こちらを参照してください。

12.6.2.3 Null データを0に設定する

ゼロデータ(関連する 0 値を有するデータポイント)として任意の null データを表すことができます。これを行うには、**IDataPointSet** で **setNullDataAsZero** のメソッドを使用できます。

```
IDataPointSet dataPoints = chart.gethDataPoints();
dataPoints.setNullDataAsZero(true);
```

詳細は、こちらを参照してください。

12.6.2.4 追加のトレンドラインオプション

EspressChart では、**ITrendLine** を呼び出し、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。

チャートが **setLinearScale** および **setRounded true** のヒストグラムチャートである場合にのみ、標準曲線の標準偏差トレンドラインを描画できます。

アプレットまたはアプリケーションとして実行できる以下のコードは、チャート内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

エクスポートされたイメージ

詳細は、こちらを参照してください。

12.6.3 チャート固有

以下に示すフィーチャは、特定のチャートタイプを処理します。以下の各機能は、チャートタイプによって、チャート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ときにも実行されます。

12.6.3.1 縦棒・横棒グラフ

このセクションでは、API を使用して縦棒、横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。チャートの外観だけでなく、コンテンツの変更も含まれます。

12.6.3.1.1 カラーセパレータ

定義されたカテゴリの値に基づいて縦棒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以下のコードは、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

エクスポートされたイメージ

詳細は、こちらを参照してください。

12.6.3.1.2 影を無効にする

縦棒、横棒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、こちらを参照してください。

12.6.3.2 円グラフ

このセクションでは、API を使用して円グラフに行うことのできるさまざまな変更点を示しています。これには、チャートの外観だけでなくコンテンツの変更も含まれます。

12.6.3.2.1 円グラフ・右回りの円スライスの描画

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することができます。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);
```

詳細は、こちらを参照してください。

12.6.3.2.2 0%スライスと 100%スライスの円ボーダ
円全体の 0%または 100%のスライスの円の枠線を削除するオプションがあります。これを行うには、**IPiePropertySet** のハンドルを取得し、**setRadealBorderDrawnForZero** メソッドを使用します。

```
IPiePropertySet pieProperties = chart.gethPieProperties();
pieProperties.setRadialBorderForZero(true);
```

詳細は、こちらを参照してください。

12.6.3.2.3 円グラフ凡例とカテゴリ、パーセント値文字文字列の区切り文字カスタマイズ
円グラフの凡例のカテゴリとパーセント値の文字列の間にユーザ定義のセパレータを配置できます。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、こちらを参照してください。

12.6.3.2.4 円ボーダカラーカスタマイズ可能
IPiePropertySet で **setBorderColor** メソッドを使用して、円の枠線の色を指定することができます。

```
IPiePropertySet pieProperties = chart.gethPieProperties();
pieProperties.setBorderColor(Color.red);
```

詳細は、こちらを参照してください。

12.6.3.3 折れ線グラフ
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示しています。チャートの外観だけでなく、コンテンツの変更も含まれます。

12.6.3.3.1 ラインエリア
水平線とデータ線の間に線のエリア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、水平ラインの上に囲まれたすべてのエリアは 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エリアと水平ラインの下のエリア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 折れ線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以下のコードは、行エリア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

エクスポートされたイメージ

詳細は、こちらを参照してください。

12.6.3.4 散布図
このセクションでは、API を使用して散布図に加えられるさまざまな変更点を示しています。これには、チャートの外観だけでなくコンテンツの変更も含まれます。

12.6.3.4.1 トップラベルにシリーズを表示
シリーズを散布頭脳のラベルに表示することができます。**IDataPointSet** に **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、こちらを参照してください。

12.6.3.4.2 図面の順序
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();
dataPoints.setConnectLinesInOriginalOrder(true);
```

詳細は、こちらを参照してください。

12.6.3.4.3 散布図キューブ幅
3D スキャッタチャートのキューブ幅を任意のサイズに変更できます。キューブの幅を変更するには、**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、こちらを参照してください。

12.6.3.5 重ね合わせチャート
このセクションでは、API を使用して重ね合わせ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。チャートの外観だけでなく、コンテンツの変更も含まれます。

12.6.3.5.1 複数の軸のタイトル
重ね合わせ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、個々の軸へ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、こちらを参照してください。

12.6.3.6 ダイヤルチャート

このセクションでは、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、チャートの外観だけでなくコンテンツの変更も含まれます。

12.6.3.6.1 コントリールエリアのスケールラベル

コントロールエリアの開始スケールと終了スケールをラベルとして表示できます。コントロールエリア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。

```
ControlRange cr1 = chart.gethControlRanges().elementAt(0);
cr1.setShowLabel(true);
```

詳細は、こちらを参照してください。

12.6.3.7 HLCO チャート

このセクションでは、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示しています。チャートの外観だけでなく、コンテンツの変更も含まれます。

12.6.3.7.1 キャンドルスティックの色を変える

API のみを使用して HLCO チャートの燭台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、こちらを参照してください。

12.6.3.7.2 キャンドルスティックの枝幅を変える

API のみを使用して、HLCO チャートの燭台の枝幅(燭台の上下の拡張部分)を変更することができます。**IDataPointSet** のハンドルを取得し、**setCandleStickWidth** メソッドを使用する必要があります。渡された数は、ローソクの幅とローソクの幅の比です。

```
IDataPointSet dataPoints = chart.gethDataPoints();
//幅を 0.5 に設定する
dataPoints.setCandleStickWidth((float)0.5);
```

詳細は、こちらを参照してください。

12.6.3.8 サーフェスチャート

このセクションでは、API を使用してサーフェスチャートの加えられるさまざまな変更点を示しています。これには、チャートの外観だけでなくコンテンツの変更も含まれます。

12.6.3.8.1 ヒートマップ
サーフェイス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従って異なる色でセクションを描画できます。

アプレットまたはアプリケーションとして実行でき、以下のコードは、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);
```

上記のコードは、3 つの色セクション、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

エクスポートされたイメージ

詳細については、こちらとこちらを参照してください。

12.6.4 パフォーマンス

以下に示すフィーチャは、チャートの生成とエクスポートのパフォーマンスを改善します。このセクションでは、チャートで使用できる追加機能を示し、チャートの生成に必要なメモリリソースと時間を削減します。

12.6.4.1 BufferedImage または Frame
エクスポート中に **java.awt.image.BufferedImage** または、**jaba.awt.Frame** を使用してチャート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ほどパフォーマンスが向上し、**java.awt.image.BufferedImage** を使用してチャート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、こちら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。

12.6.4.2 チャートの生成順序
チャートが生成される順序を選択し、API のみを使用してチャートの生成に要素を追加することもできます。たとえば、背景を描き、円を描き、円の中心にチャートを作成してチャートを作成することができます。これを行うには、**IChartGraphics** インターフェイスを実装するクラスを作成し、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** インターフェイスを使用すると、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以下のコードは、チャートとグラフィックス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
                ”ChartGeneration.cht”); //テンプレート

chart.setChartGraphics(new chartGenerationGraphics());

…

class chartGenerationGraphics implements IChartGraphics {

        static final long serialVersionUID = 1;

        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

エクスポートされたイメージ

詳細は、こちら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のみ使用できます。

12.6.5 ビュアー

下に示すフィーチャは、Java アプリケーションまたは Java アプレットのいずれかでチャートを表示するときにビュアーを処理します。このセクションの各機能は、Chart Viewer で使用できる追加の機能を示しています。

12.6.5.1 コールバックメカニズム

イベントを処理する上位レベルのコールバックメカニズムがあります。チャート内のデータオブジェクトがビュアー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シリーズ、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として実行できる以下のコードは、イベントをキャプチャ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
                ”CallBack.cht”); //  テンプレート

chart.addActionListener(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:
			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

エクスポートされたイメージ

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。

12.6.5.2 ツールのヒントテキストを有効・無効にする

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、こちらを参照してください。

12.6.5.3 キャンバスエリア

より多くのイベントプロパティ(ユーザ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ができます。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す。

```
ICanvas chartCanvas = chart.gethCanvas();
Component chartCanvasComponent = chartCanvas.getCanvasArea();
```

詳細は、こちらを参照してください。

12.7 チャートビュアーオプションの変更

時に、Chart Viewer を使用してチャートを表示しているときにユーザ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。

ユーザがチャートビュアーを使用してチャートを表示しているときに、チャートを右クリックすると、メニューがポップアップされます。このメニューを使用して、ユーザチャートタイプの変更、チャートディメンションの変更などのチャートオプションを選択できます。チャートやスタティックイメージ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ッ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、Chart Viewer 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、**IpopMenu** 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、こちらを参照してください。

12.8 Javadoc

API 全体の Javadoc は、EspressChart と共に提供されています。この API は、Chart API と Charting API の両方をカバーしています。これらは、Quadbase の web サイトにあります。

12.9 スイングバージョン

1.1 JFC/Swing 版の EspressChart charting API も利用可能です。詳細は、こちらを参照してください。

12.10 概要

EspressChart 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トライブラリを提供します。チャートデザイナ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 桁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。チャートのすべての属性は、EspressChart Designer で作成できるテンプレートファイルで設定できます。EspressChart Designer で作成できるテンプレートファイルで設定できます。EspressChart API は、Windows95、Windows NT/2000、Solaris、Linux、AIX と HP プラットホームで Nerscape の Communicator(4.06 以上)、Microsoft の Internet Explorer(4. X 以上)、Sun の Applerviewer(1.2 以上)でテストされています。

12.11 チャートデータの取得

この章の 2 つの付録(チャートデータの取得とチャートの作成)には、Designer を利用せずに、最初からチャ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、チャートを展開及び保守すること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チャートを作成することが必要になることもあります。

以下は、**QbChart** コンストラクタの例です。多数のチャート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 つの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、およびデータ列のグラフの各列へ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 でも問題ありません。チャート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。チャートの作成では、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するなどの、実践的な例も含まれています。

チャート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 つから取得することができます。

これらのソースは、以下の通りです。

- 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。

12.11.1 データベースからデータの取得

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 ドライバが利用可能であれば、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。以下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”,                          //ユーザーネーム
                    ”myPassword”,                      //パスワード
                    ”select * from sales”);            //SQL
```

DBInfo はインターフェイス **IDatabaseInfo** を実装しているため、以下の **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、データベース情報(ユーザ ID、パスワード、ロケーション、ドライバ、クエリなど)全体を EspressChart に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);

//以下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 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、チャートオブジェクトが作成されます。グラフを更新するには、データベースに再度接続し、結果セットオブジェクトをグラフに渡して、手動で更新する必要があります。

12.11.1.1 JNDI

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、異なる環境間でチャ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設定されている場合、チャ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 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 のために空です
//Tomcat コンテキスト内で実行します。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** を含む以下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);
```

12.11.2 データファイル（TXT,DAT,XML）からデータの取得

チャートを作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す。データファイルは、テキストファイルでも、XML 形式のファイルでも構いません。チャ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
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、チャート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、Chart API を使用してプログラムが読み取れるようになります。

XML 形式の場合、仕様は以下の通りです。

```
<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>
    </Row>
</EspressData>
```

XML データの詳細は、XML と 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単純です。以下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。EspressManager に接続している場合、相対パスは EspressManager の作業ディレクトリに関するものでなければなりません。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の URL(**http://localhost:8080/servlet/XMLDataGenerator** など)として指定できます。

12.11.3 XML データソースからデータの取得

上記に加えて、EspressChart では、データを検索して XML ファイルを紹介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。以下のコードは、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);
```

```
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後に繋がる階層の各 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、属性名に別のストリングも必要です。上記の例では、"ProductID"が"Product"の属性であるため、フィールド[0]はコンストラクタを使用して作成されています。

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、以下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 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 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);
```

12.11.4 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、インターフェイス IResultSet(quadbase.util パッケージで定義されている)によって可能になります。このインターフェイ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** インターフェイスと非常によく似ています。ユーザは、**IResultSet** の独自の実装を提供することも、EspressChart が提供する **IReslutSet** を使用することもできます。もっとも簡単な実装は、**DbData** クラスによって提供されます(**IResultSet** 実装を提供するほかのクラスは、QueryResultSet と StreamResultSet です)。グラ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.Strings)** - HTML ページからデータ値の引数を解析して **DbData** を構築する
- **DbData(java.lang.String [] fieldName、java.lang.Object [] [] records)** – 新しい **DbData** クラスを構築する
- **DbData (java.lang.String [] dataType 、 java.lang.String [] fieldName java.lang.String [] [] records)** - 新しい **DbData** クラスを構築する

この例では、以下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を示し、2 番目の引数はフィールド名を示します。3 番目の引数 records [] []は、レコード配列を提供します。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);
```

グラフを作成するには、以下の **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);
```

12.11.5 カスタムインプリメントで渡されたデータ

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、チャ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 インターフェイ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以下に示します。

```
public class CustomClassData extends Applet implements IDataSource {

    //引数としてデータを渡すための DbData の設定
    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)を作成し、メモリに格納しています。**getResultSet()**メソッドが呼び出されると、IResultSet インターフェイスを実装する **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。チャートを作成するには、以下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、チャートの生成に使用されます。これは、スタンドアロンのチャート構成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, 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は通常、パラメータダイアログボックスは選択肢の少ないテキストボックスです。ただし、IQueryParamValuesProvider インターフェイスを実装することによって、ドロップダウンボックスから選択肢の選択肢を指定することができます。

```
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;
		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;
	}
```

```
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;
				}
			}
			stmt.close();
			conn.close();

			return v;
		} catch (Exception ex) {
			ex.printStackTrace();
		}
		return null;
	}
```

```
        }

}
```

フルソースコード

12.11.6 スプレッドシートモデルからデータの取得

チャートは、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す。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。チャートはスプレッドシートモデルのリスナーとして登録され、スプレッドシートデータの変更が通知されると自動的に更新されます。ISpreadSheetModel インターフェイ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。以下の例は、スプレッドシートモデルをチャートに使用する方法を示しています。**ISpreadSheetModel** インターフェイ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ます)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)},
                    {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、以下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

EspressChart は次に内部でデータを転置します。その結果、データは以下のように変更されます。

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

次に転置されたデータに関して、チャート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、以下のコンストラクター—を使用します。

```
public QbChart(Applet applet, int dimension, int chartType, ISpreadSheetModel spreadsheet, IColumnMap colMap, String templateFile);
```

12.11.7 Enterprise Java Beans（EJBs）からデータの取得

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 データソースからチャ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** 引数を EspressManager と組み合わせてください)。

次のコンストラクタに接続情報を入力して **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);
```

12.11.8 SOAP データソースからのデータの取得

EspressChart では、SOAP サービスからデータを取得できます。SOAP データソースは、XML データソースに似ています。主な違いは、XML ファイルが SOAP メッセージで転送される点です。SOAP データソースを使用するには、XML ファイルを送信する SOAP サービスが必要です。このサービスは、どのプログラミング言語でも記述できます。

要求 SOAP メッセージにはパラメータがなく、この形式で SOAP 応答が必要です。

```
<soapenv:Envelope xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
xmlns:xsd=”http://www.w3.org/2001/XMLSchema”
xmlns:xsi=”http://www.w3.org/2001/XMLSchema-instance”>

        <soapenv:Body>
                <XMLTYPE>QUADBASE</XMLTYPE> </soapenv:Body>

</soapenv:Envelope>
```

メッセージは body 要素(XMLTYPE)が 1 つしかありません。XML ファイルのタイプを指定します。次の値が使用可能です。

QUADBASE	XML は Quadbase フォーマットです
DTD	XML は DTD スキーマを使用しています。
XSD	XML は XML スキーマを使用しています。

XML ファイルとスキーマは、SOAP メッセージの MIME 添付ファイルとして送信されます。QUADBASE XMLTYPE が使用されている場合、SOAP メッセージには 1 つの添付ファイル(Quadbase 形式の XML ファイル)が必要です。DTD または XSD XMLTYPE が使用されている場合、メッセージには 2 つの添付ファイルが必要です。最初のファイルは XML ファイルで、2 番目のファイルは DTD または XSD ファイルです。これらの添付ファイルの MIME タイプは **text/xml** です。

これは、SOAP データソースからチャートを作成するためのコンストラクタです。XML は Quadbase 形式でなければなりません:

```
QbChart(java.applet.Applet applet, int dimension, int chartType, java.lang.String
SOAPURL, java.lang.String serviceName, java.lang.String methodName, IColumnMap
cmap, java.lang.String template, boolean doTransposeData, boolean[]
transposeCol)
```

12.11.8.1 SOAP サービスの例

EspressChart は SOAP サービスのサンプルを提供します。ソースコードは、**<EC インストールディレクトリ> /help/examples/soap/SOAPService.java** にあります。この例を使用するには、ソースコードを編集し、すべての**<ECInstall>**を EspressChart のインストールディレクトリに置き換えてください。 次に、ファイルをコンパイルします（コンパイルするには、クラスパスに **<EC-installation-directory> / lib ディレクトリのすべての jar ファイル**を含める必要があります）。

次に、この SOAP サービスをデプロイする必要があります。以下の手順では、Apache Tomcat にデプロイする方法を示します。<EC>は EspressChart のインストールディレクトリを表します。

1. Apache Web サイトから Tomcat 5.0 以上をダウンロードし、インストールします。Tomcat がインストールされているディレクトリは<TOMCAT>と表します。
2. 次のファイルを**<EC>/lib** から**<TOMCAT>/common/lib** にコピーします。
   activation.jar、axis.jar、commons-discovery-0.2.jar、commons-logging-1.0.4.jar、jaxrpc.jar、log4j-1.2.8.jar、mail.jar、saaj.jar、wsdl4j-1.5.1.jar
3. 次のファイルを**<EC>/help/examples/soap/**から**<TOMCAT>/webapps/axis/WEB-INF/classes/quadbase/soap** にコピーします（ディレクトリがない場合は新規で作成してください）。
   deployDS.wsdd、SOAPService.class、undeployDS.wsdd
4. **<EC>/help/examples/soap/**にある **web.xml** ファイルを **<TOMCAT>/webapps/axis/WEB-INF/** にコピーします。
5. Tomcat を開始します。（**<TOMCAT>/bin/startup.bat** を使用します）
6. 次のコマンドを**<TOMCAT>/webapps/axis/WEB-INF/classes/quadbase/soap** で実行します。ステップ 2 以降の JAR ファイルをすべてコピーしていることを確認してください。SOAP サービスがデプロイされます。
   java org.apache.axis.client.AdminClient deployDS.wsdd
   Tomcat をデフォルトポート（8080）で実行していない場合、**-p** 引数を使用して、使用しているポートを指定してください。

”SOAPService-fetchData”は XML を Quadbase 形式で返し、 ”SOAPService-fetchData2”は XML と DTD ファイルを返し、 ”SOAPService-fetchData3”は XML と XML スキーマファイル（XSD）を返します。この例で必要な接続情報は次のとおりです。

サーバ URL は、3 つのサービスすべてについて
http://machine:port/axis/services/SOAPService です。マシンとポートを Tomcat が実行されているマシン名とポートで置き換えてください（デフォルトのポートは 8080 です）。Tomcat がローカルマシン上で実行されていない場合は、XML ファイル内の URL リンクを変更することを忘れないでください。

すべての例のサービス参照名とメソッド参照名を以下に示します。

- service lookup name:**SOAPService**
  method lookup name:**fetchData**
  XML が Quadbase 形式か確認します。
  このサービスは Quadbase 形式で XML を送信します。XML ファイルのデータソースと同じ方法でデータソースを使用できます。
- service lookup name:**SOAPService**
  method lookup name:**fetchData2**
  XML が Quadbase 形式か確認しません。
  これは DTD ファイルを使用します。このデータソースを使用して XML クエリを追加し、XML ファイルクエリと同じ方法でクエリを使用できます。
- service lookup name:**SOAPService**
  method lookup name:**fetchData3**
  XML が Quadbase 形式か確認しません。
  スキーマファイルを使用します。このデータソースを使用して XML クエリを追加し、XML ファイルクエリと同じ方法でクエリを使用できます。

12.11.9 複数のデータソースからの取得

EspressChart には、複数のデータソースをまとめて併合する機能もあります。**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。**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。

```
// 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]; // Declaration For Database (Data From a Database section)
DBInfo dbinfo = new DBInfo("jdbc:quadbase:/machine/schema", "quadbase.jdbc.QbDriver", "myName","myPassword","select * from sales");
//Declaration For Data Passed as an Argument (Data Passed as an Argument 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, "help/examples/data/Columnar1.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 as an Argument section)
// DataSheet(Applet applet, IResultSet data)
dataSheet[2] = new DataSheet(this, data); // create QbChart
QbChart chart = new QbChart

                    (this,                  // applet
                    QbChart.VIEW2D,         // Two-Dimensional
                    QbChart.PIE             // Pie Chart
                    dataSheet,              // DataSheet
                    colInfo,                // column information
                    null);                  // No template
```

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。

12.11.10 スプレッド形式のデータ

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、ラベル(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)を含む個別のデータポイントを表します。

次の例を考えてみましょう。

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。データをスプレッドシート形式で扱う場合は、3 つのインデックス(Nasdaq、Dow、および SP500)をカテゴリ軸、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。以下の 3 つは主に使用するものです。

```
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すべての列を 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。数値でない場合、転置は成功しません。

EspressChart では、選択列を転置することもできます。転置される列は、同じデータ型を共有する必要があります。転置後、元の列が削除され、新しい列が表データの最後に挿入されます。選択的転置は 1 回のみ行うことができます。特定の数値列を転置してから、他の数値列または文字列を再度転置することはできません。

非選択転置と同様に、QbChart には選択的転置を可能にするいくつかのコンストラクタがあります。以下の 3 つは主に使用するものです。

```
QbChart(java.applet.Applet applet, int dimension, int chartType, IResultSet data, boolean

        doTransposeData, int[] transposeCol, IColumnMap cmap, java.lang.String template);

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);
```

転置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。

12.11.11 データの転送

EspressChart 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、データを目的のチャートタイプに渡す前に、データを転記する(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。

データが転置されると、元のデータ列(転置で使用される)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 2 つの新しい列には、転置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 つが転置用に選択されている場合、新しいデータセットは 5 列から 3(転置列の数)+ 2(追加された新しい列)または 4 列になります。

データ列を転置するときは、2 つ注意点があります。

- データ列はすべて同じデータ型でなければなりません。
- チャート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、データの転置方法について説明します。

12.11.11.1 非選択的転置

この例では、最初の列(列 0)を除くすべての列が転置されます。たとえば、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

転置されると以下のようになります。

Product	ColumnLabel	Value
Chairs	January	$3872.35
Chairs	February	$3962.21
Chairs	March	$4218.57
Tables	January	$6534.98
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
エクスポートされた結果

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。

12.11.11.2 選択的転置

この例では、転置する列を選択します。同じデータ型を共有する列のみを転置することができます。選択的転置では、転置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	February	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

値の列を転置すると、次の表のようになります。

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
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転置するには、次のコンストラクタのように、転置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
エクスポートされた結果

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。

12.12 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、**<EspressChartInstall> /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

<EspressChartInstall> / **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
エリアグラフ	QbChart.AREA
円グラフチャート	QbChart.PIE
散布図	QbChart.SCATTER
積み上げ縦棒グラフ	QbChart.STACKCOL
積み上げ棒グラフ	QbChart.STACKBAR
積み上げ面グラフ	QbChart.STACKAREA
High-Low	QbChart.HILOW
HLCO チャート	QbChart.HLCO
100% Column	QbChart.PERCENTCOL
サーフェイス	QbChart.SURFACE（3D 限定）
バブルチャート	QbChart.BUBBLE（2D 限定）
重ね合わせチャート	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、チャートタイプとデータマッピングを参照してください。

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、チャートタイプと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.ChartAPI** 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。

12.12.1 縦棒、横棒、折れ線、エリア、円グラフ、重ね合わせチャート

縦棒/横棒/折れ線/エリア/円グラフ/重ね合わせチャートは、**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。

12.12.1.1 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 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;
colInfo.series = 0;
colInfo.value = 3;
colInfo.subvalue = 2;
```

12.12.1.2 チャートの作成

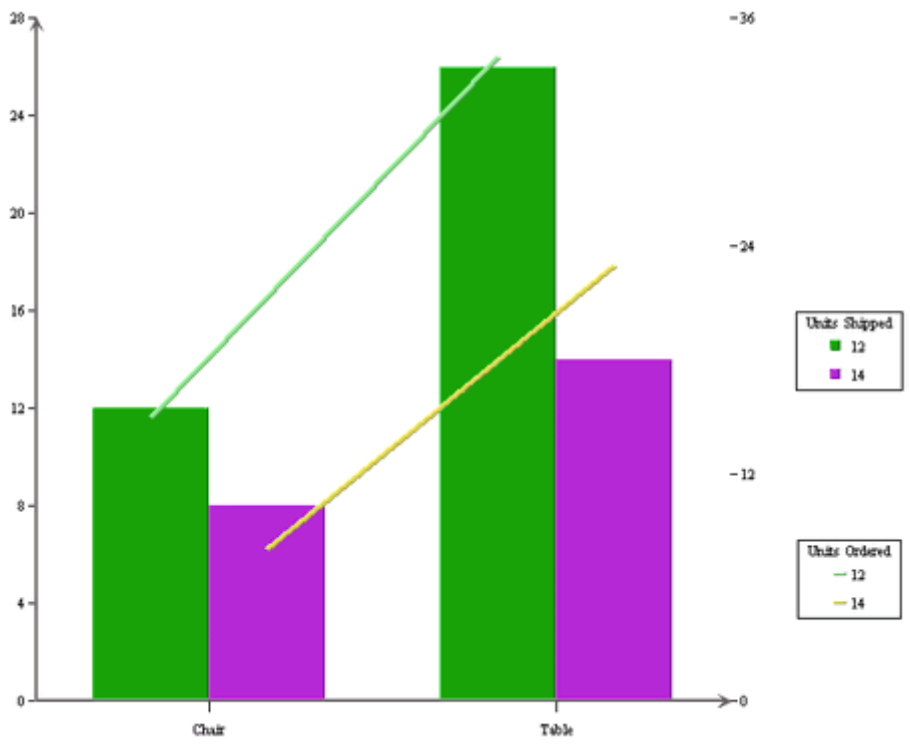
縦棒グラフの作成は比較的簡単です。ColInfo 配列を設定する方法と、前のセクションでデータを取得する方法についてはすでに説明しました。次のコードは、前述の縦棒グラフを作成する方法を示しています。

```
Component doDataFromArguments(Applet applet) {


    // EspressManager に接続しないでください
    QbChart.setEspressManagerUsed(false);

    // Column Mapping
    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,              // Applet
                              QbChart.VIEW2D,        // Two-Dimensional
                              QbChart.COL,          // Column Chart
                              data,                 // Data
```

```
                                colInfo,              // Column information
                                null);                // No specified template
    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、下の図のような縦棒グラフが表示されます。



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。

12.12.2 レーダーチャート

レーダーチャートの **ColInfo** パラメータは、**subvalue**（小数値）以外の縦棒/横棒グラフ/折れ線グラフ/エリア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。

12.12.2.1 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

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;
```

12.12.2.2 チャートの作成

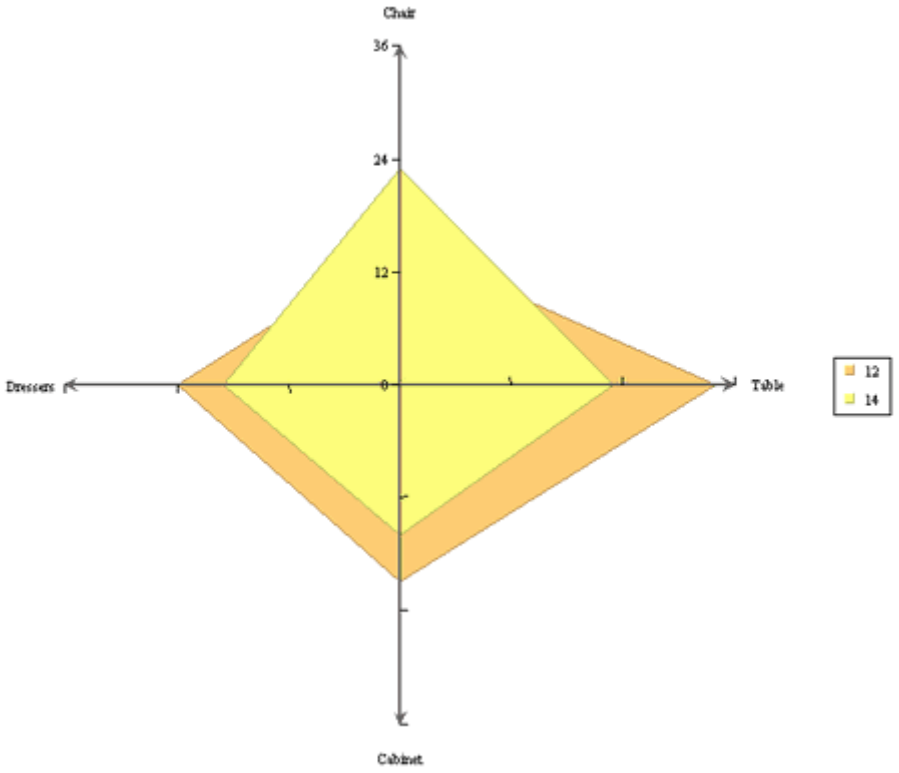
次に示すコードは、前述した散布図の作成のデモンストレーションです。

Component doDataFromArguments(Applet applet) {

```
  // EspressManager には接続しないでください
  QbChart.setEspressManagerUsed(false);

  // Column Mapping
  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,              // Applet
                          QbChart.VIEW2D,       // Two-Dimensional
                          QbChart.RADAR,        // Radar Chart
                          data,                 // Data
                          colInfo,              // Column information
                          null);                // No specified template
    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 2D 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。

12.12.3 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(3D の散布図)軸を追加することもできます。

12.12.3.1 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

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;
```

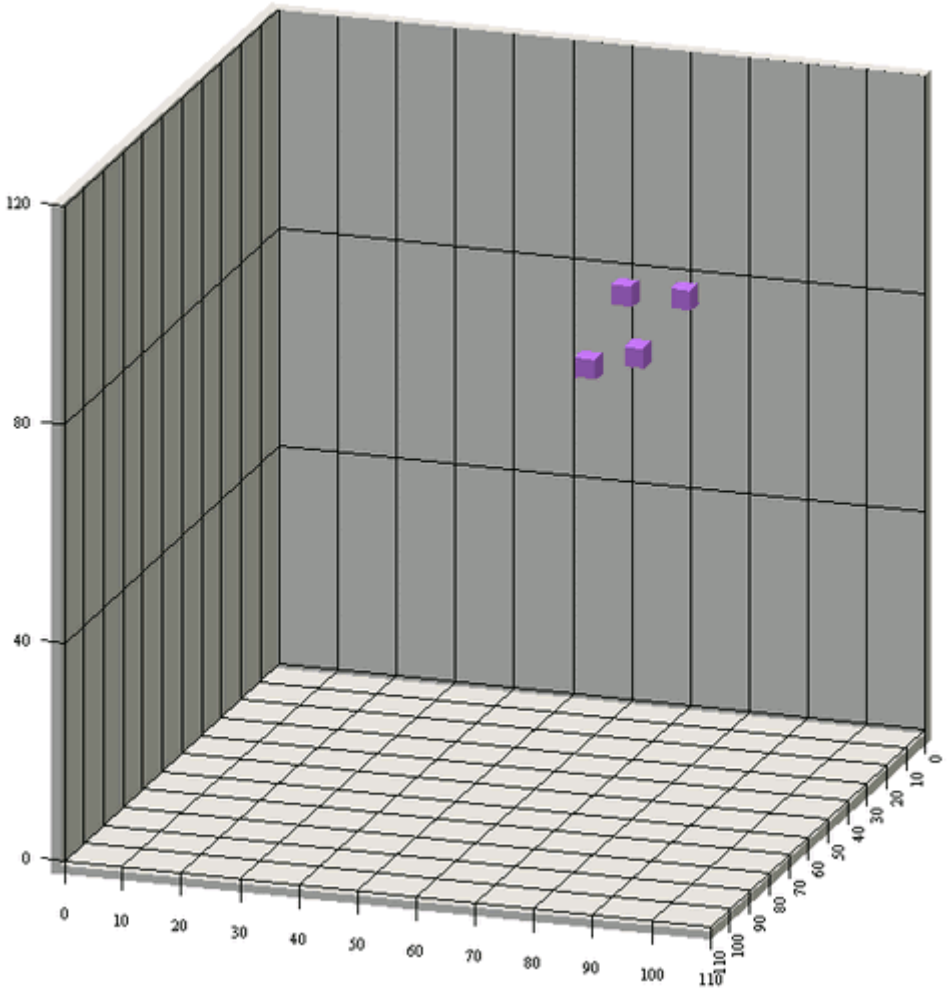
12.12.3.2 チャートの作成

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,              // Applet
                                QbChart.VIEW3D,       // Three-Dimensional
                                QbChart.SCATTER,      // Scatter Chart
                                data,                 // Data
                                colInfo,              // Column information
                                null);                // No specified template
  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 3D 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。

12.12.4 積み上げ縦棒・パーセンテージ縦棒・積み上げ横棒、および積み上げエリアのグラフ

縦棒/折れ線/エリアチャートのパラメータに加えて、積み上げ縦棒/パーセンテージ縦棒/積み重ね横棒/積み上げエリアにも **SumBy** パラメータを設定する必要があります。積み上げ列/パーセンテージ列/積み重ね横棒/積み上げ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。

12.12.4.1 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
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

Day	Drink	Total	Average
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;
```

12.12.4.2 チャートの作成

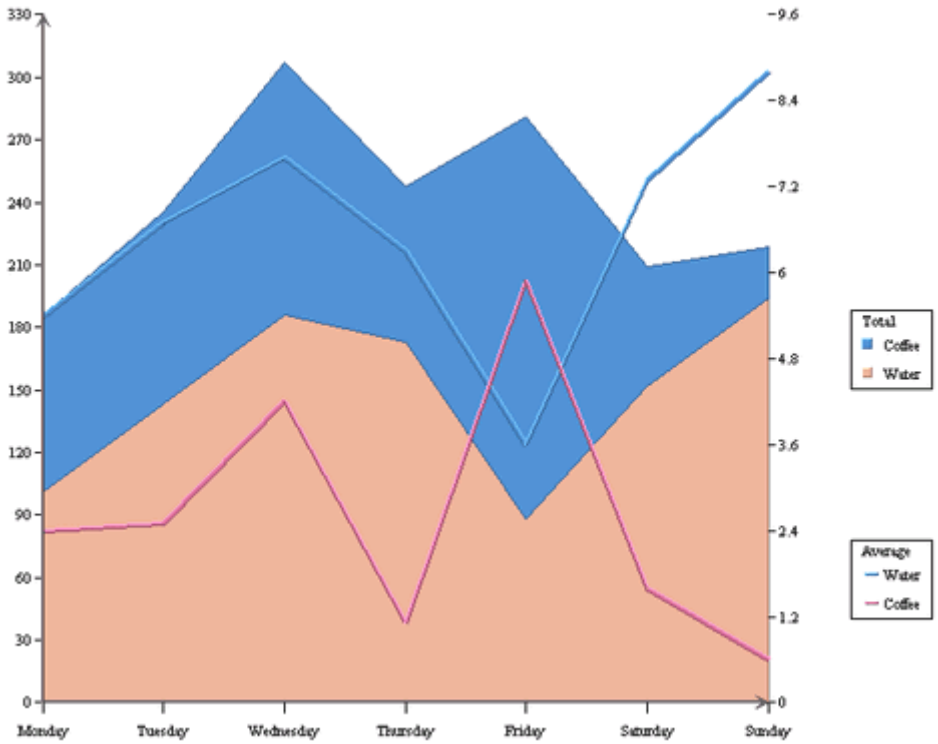
次のコードは、前述の積み重ねエリアチャートを作成する方法を示しています。

```
Component doDataFromArguments(Applet applet) {


    //EspressManager に接続しないでください
    QbChart.setEspressManagerUsed(false);

    //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,         //2D
                        QbChart.STACKAREA,      //積み上げエリアグラフ
                        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 2D の積み上げエリアグラフが生成されます:



生成された積み上げエリアグラフ

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。

12.12.5 ダイヤルチャート

ダイヤルチャートの ColInfo 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。

12.12.5.1 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:

```
ColInfo colInfo = new ColInfo();
colInfo.category = 0;
colInfo.value = 1;
```

12.12.5.2 チャートの作成

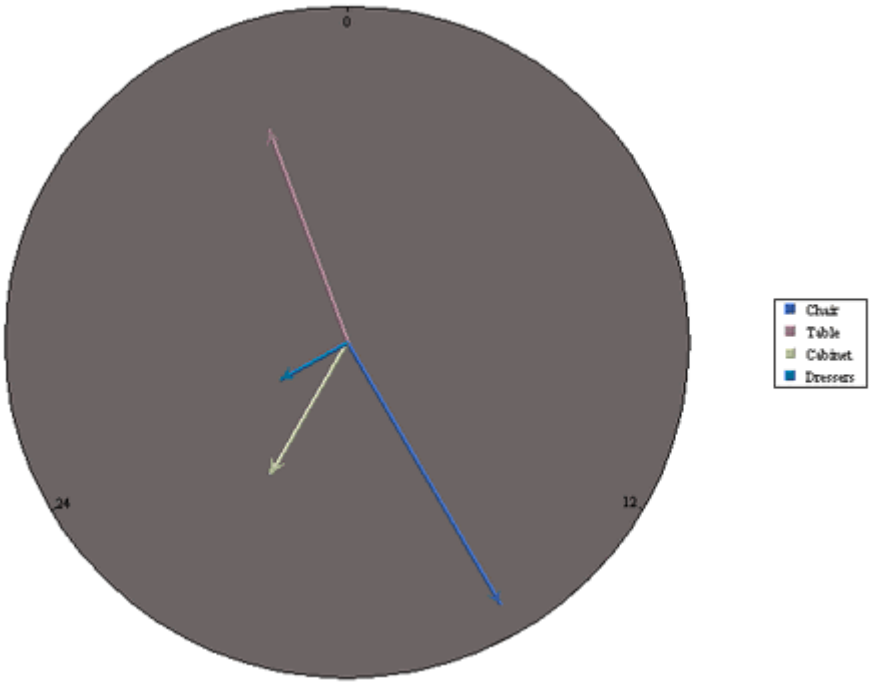
次のコードは、上記のダイヤルチャートを作成する方法を示しています。

```
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,         //2D
                                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 2D のダイヤルチャートが作成されます:



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。

12.12.6 ボックスチャート

Box チャートの ColInfo パラメータは、シリーズを除く縦棒/横棒/折れ線/エリア/円/重ね合わせ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。

12.12.6.1 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

Subject	Student	Score
Science	A.S.	86
Science	T.E.	90
Science	V.Q.	73
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;
```

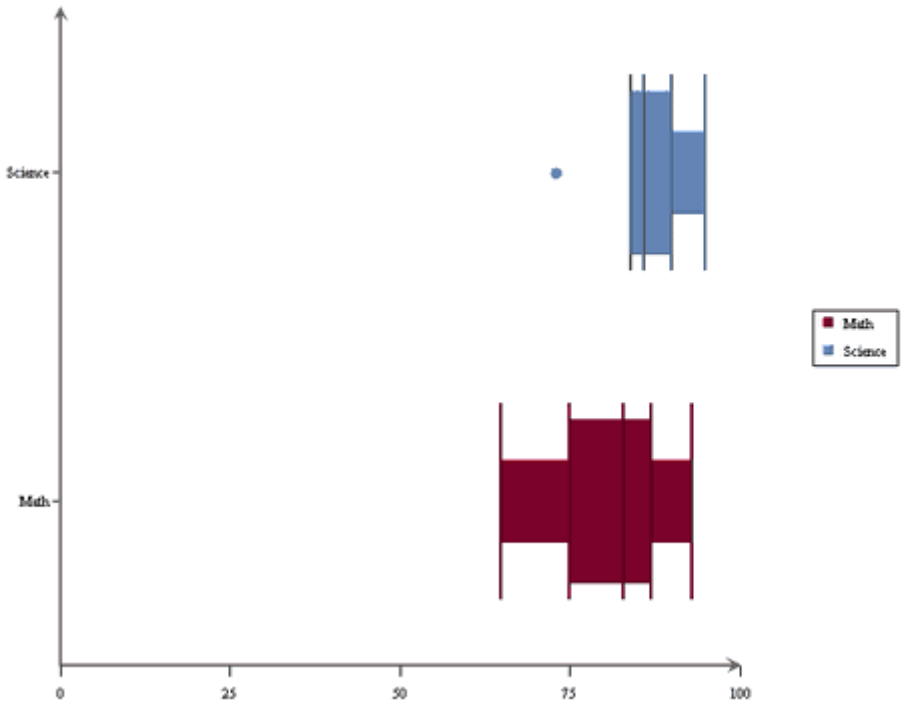
12.12.6.2 チャートの作成

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,          //2D
                              QbChart.BOX,             //ボックスチャート
                              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 2D 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。

12.12.7 バブルチャート

バブルチャートの ColInfo パラメータは、3D スキャッタチャート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。

12.12.7.1 列マッピング

例えば、以下に示すデータがあるとします:

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;
```

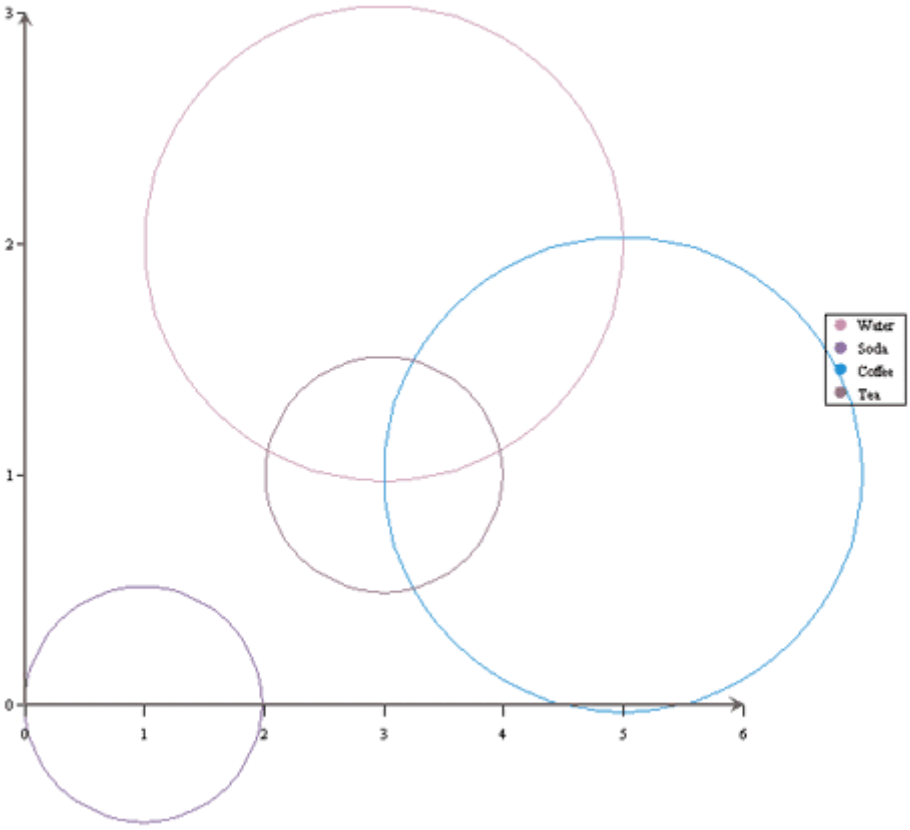
12.12.7.2 チャートの作成

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
                        (applet,              //アプレット
                        QbChart.VIEW2D,       //2D
                        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 2D 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。

12.12.8 高低および HLCO チャート

High-Low/HLCO チャートの **ColInfo** 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。

12.12.8.1 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-	DEF	9.48	8.12	8.93	8.66	16758

Day	Company	High	Low	Close	Open	Volume
01-02						
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:

```
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;
```

12.12.8.2 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。

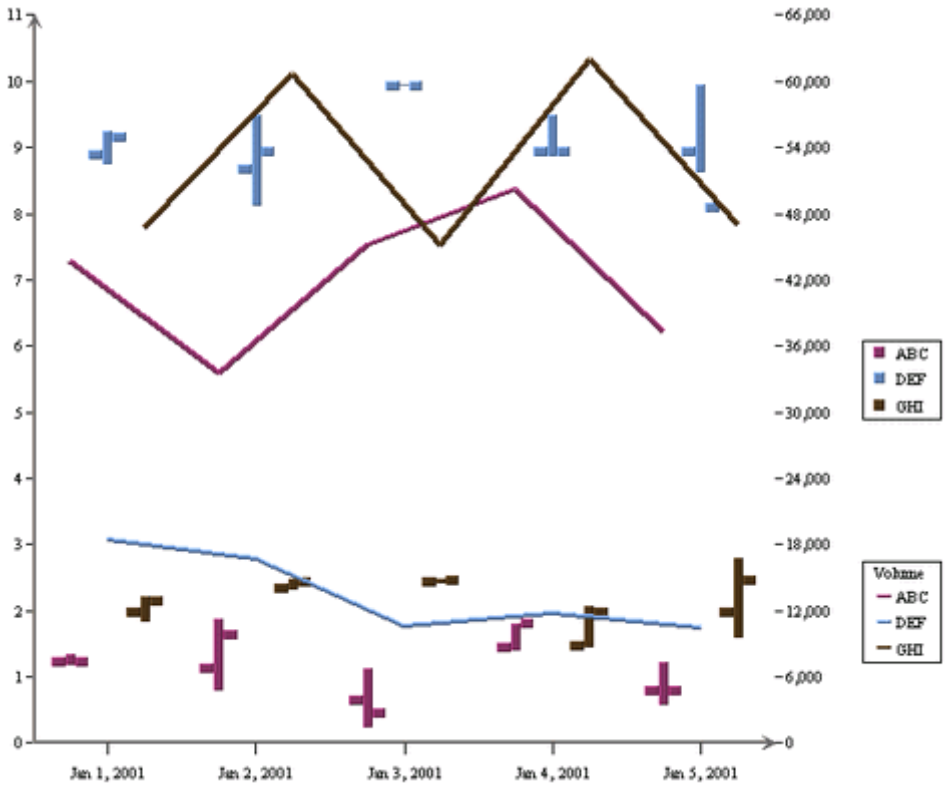
```
Component doDataFromArguments(Applet applet) {


    //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"},
                          {"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,       //2D
                        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 2DHLCO 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。

12.12.9 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、3D 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。

12.12.9.1 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};
ColInfo colInfo = new ColInfo(-1, map);
```

12.12.9.2 チャートの作成

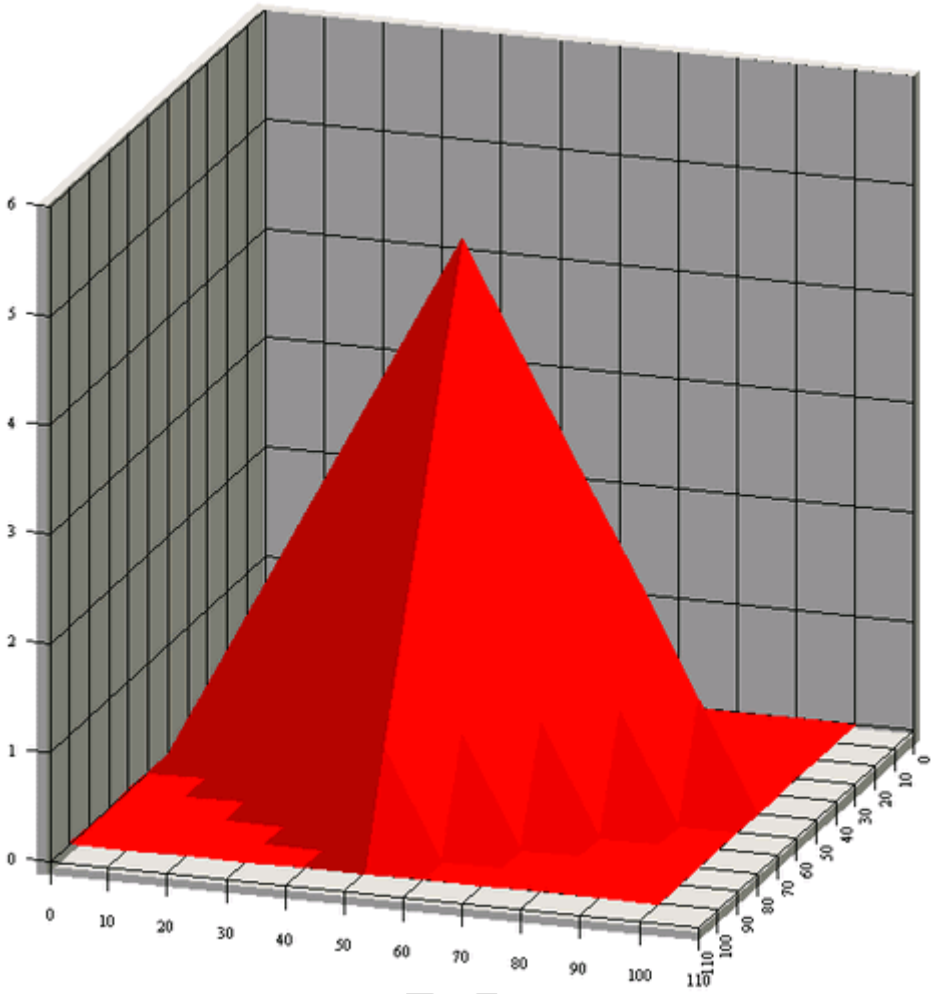
次のコードは、前述のサーフェスチャートを作成する方法を示しています:

```
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, //3D
                                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, //データを転置しないでください
                                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 3D サーフェスチャートが生成されます:



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。

12.12.10 ガントチャート

ガントチャートの **ColInfo** 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。

12.12.10.1 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
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;
```

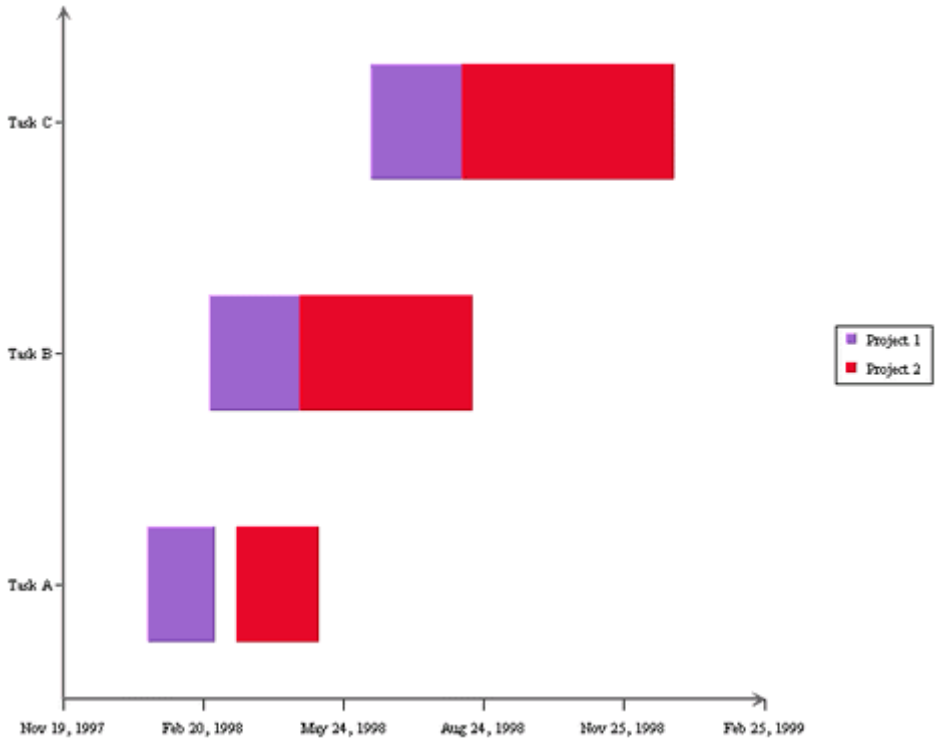
12.12.10.2 チャートの作成

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,       //2D
                        QbChart.GANTT,        //ガントチャート
                        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 2D 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。

12.12.11 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。

12.12.11.1 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});
```

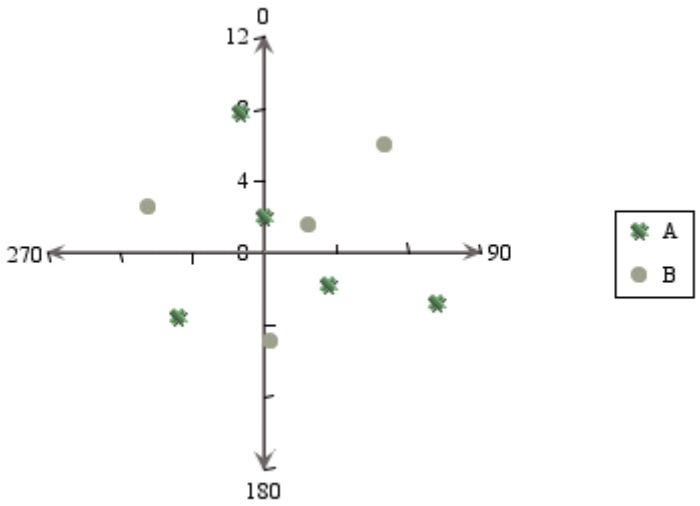
12.12.11.2 チャートの作成

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,         //2D
                        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 2D 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。

12.12.12 日付・時間ベースのズームチャート

EspressChart 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、サーフェイス、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、アトリビュート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 インターフェイスを使用して設定します。

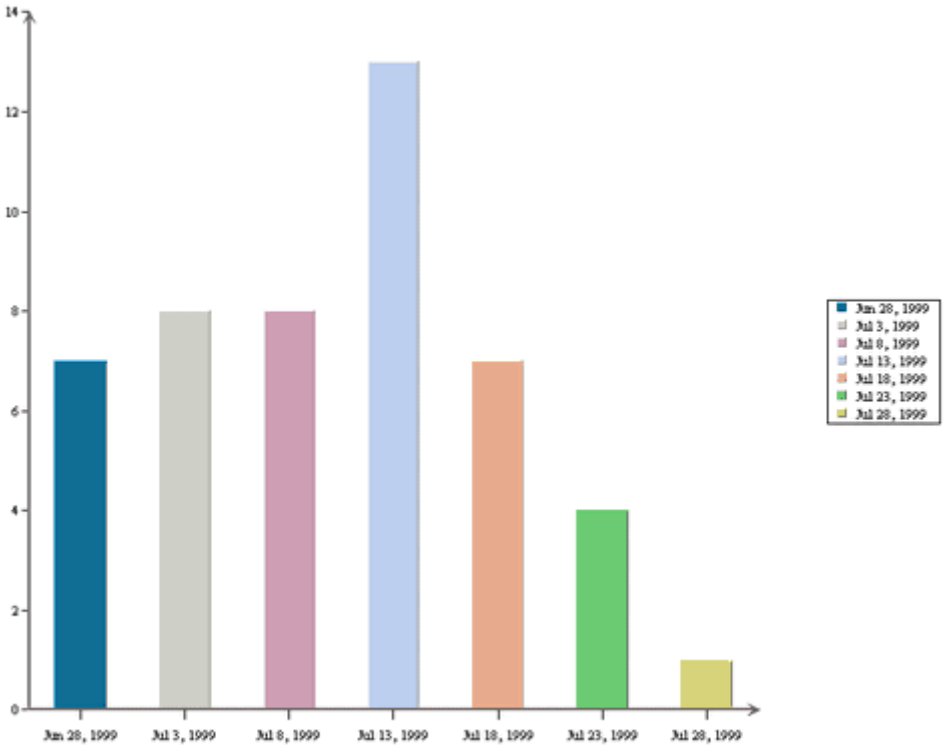
次の例は、例またはアプリケーションとして実行でき、ズームを組み込んだチャートを示しています。ここでは、デフォルトのズームは一度に 5 日を表示し、最大のズームアウトは 1 ヶ月、最大のズームインは 1 日です。

```
//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


//ズームインインターフェイ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();

}
```

```
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。

12.12.13 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。スタンドアロー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。

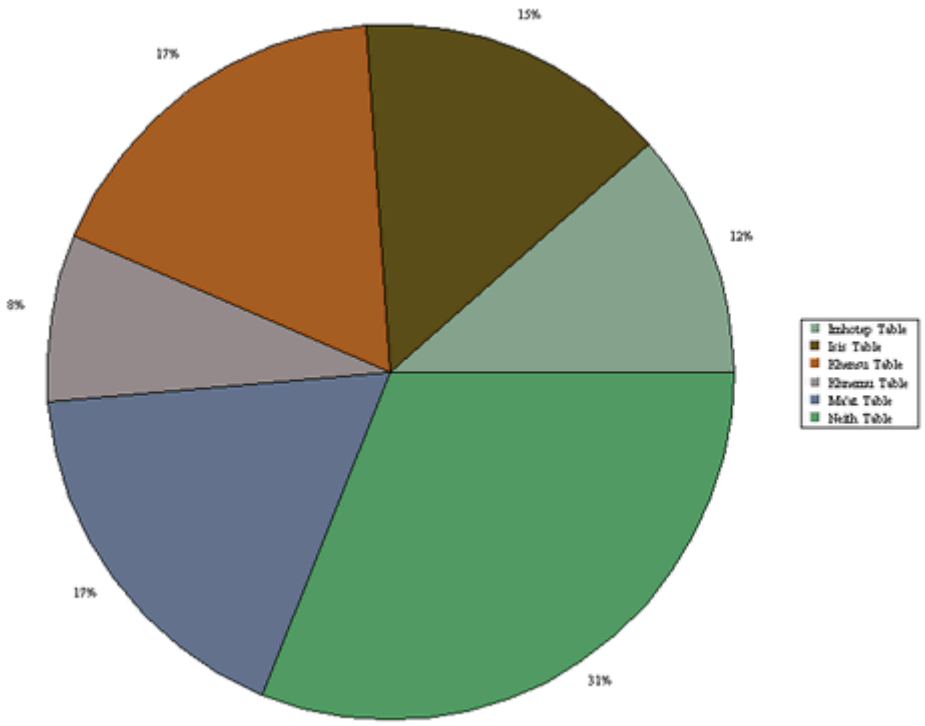
パラメータ化されたクエリを使用するチャートの例を次に示します。クエリが実行されるデータベースは WoodView HSQL なので、クラスパスにデータベース HSQL JDBC ドライバ(hsqldb.jar)を追加する必要があります。ドライバは<EspressChartInstall>/lib ディレクトリにあります。

```
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。

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。EspressChart では、”IN”句内のパラメータは複数値とみなされる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。

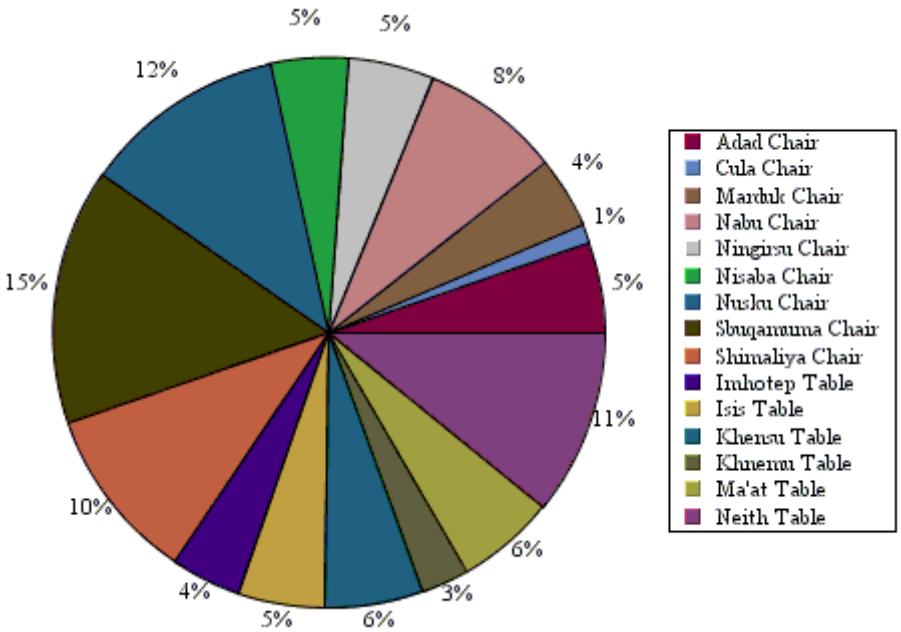
以下は、複数値のパラメータ化されたクエリを使用するチャートの例です。クエリが実行されるデータベースは WoodView HSQL なので、クラスパスにデータベース HSQL JDBC ドライバ(hsqldb.jar)を追加する必要があります。ドライバは<EspressChartInstall>/lib ディレクトリにあります。

```
Component doMultiValueParameter(Applet applet) {
        //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:



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。

12.12.14 ドリルダウンチャート

EspressChart 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。

12.12.14.1 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/折れ線/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。

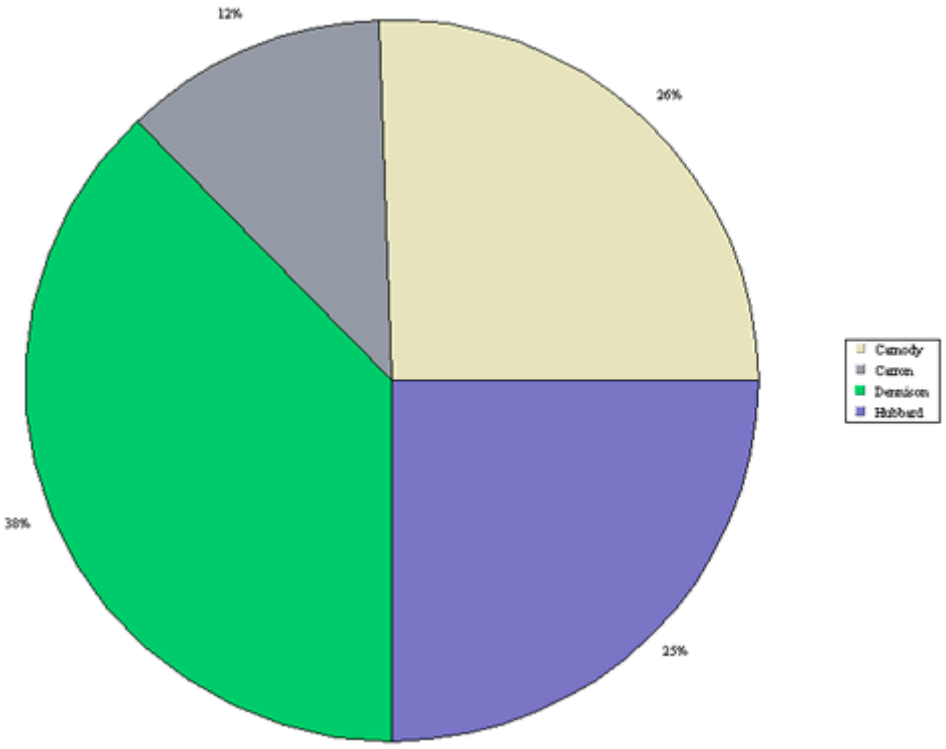
以下に、パラメータドリルダウンチャートの例を示します。クエリが実行されるデータベースは WoodView HSQL なので、クラスパスにデータベース HSQL JDBC ドライバ(hsqldb.jar)を追加する必要があります。ドライバは<EspressChartInstall>/lib ディレクトリにあります。

```
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,
                                                "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;

}
```

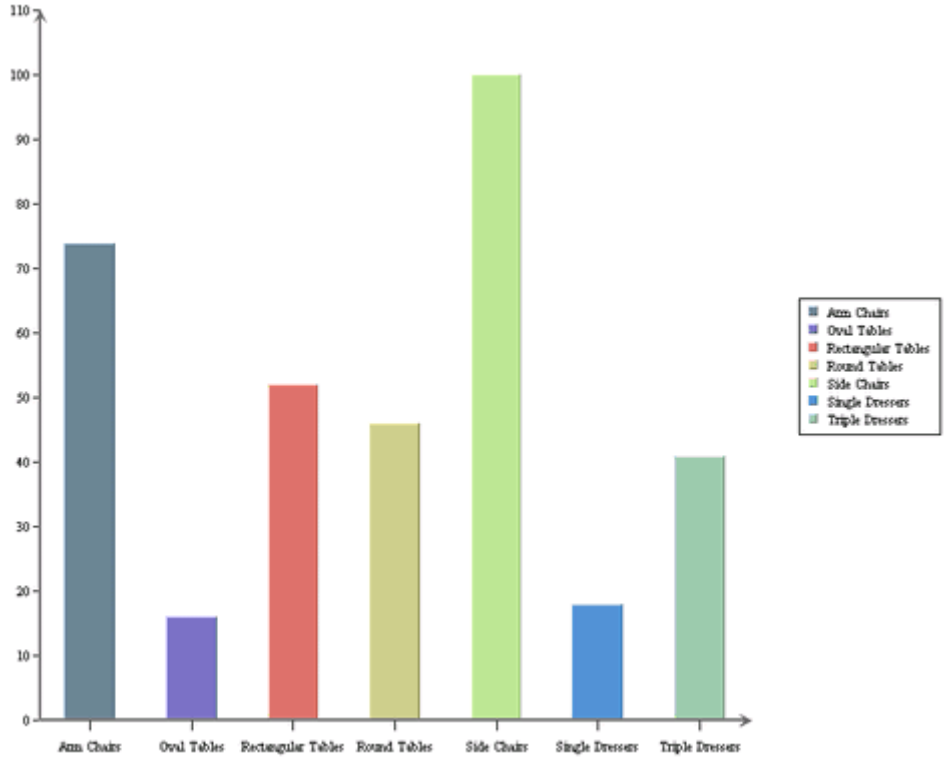
フルソースコード

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:



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。

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。

12.12.14.2 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。

列、横棒、折れ線、円、エリア、重ね合わせ、レーダー、ダイヤル、積み上げ縦棒と積み上げ横棒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。

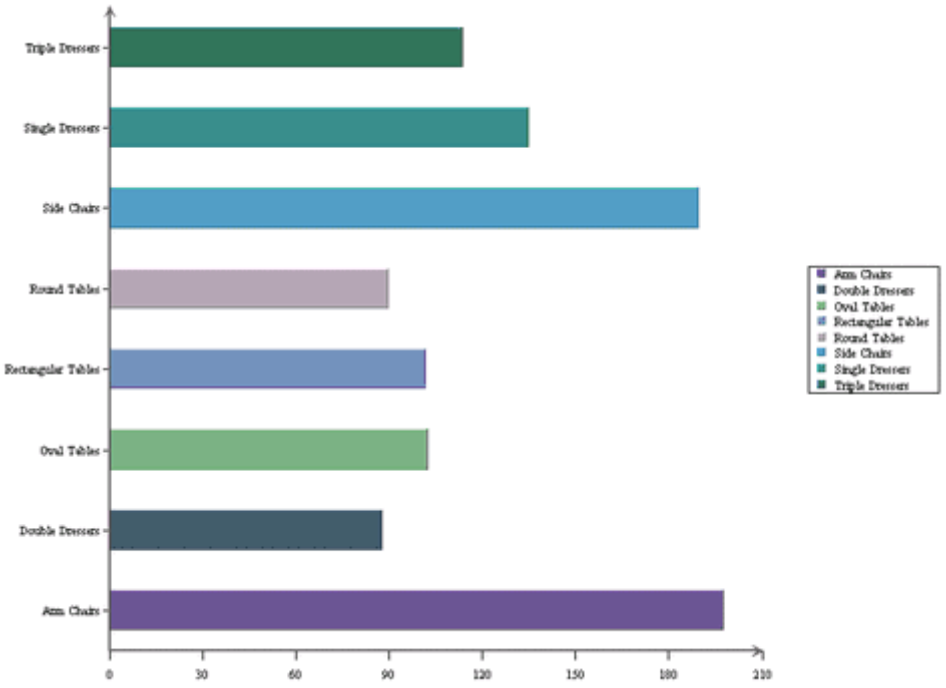
以下に、データドリルダウンチャートの例を示します。クエリが実行されるデータベースは WoodView HSQL なので、クラスパスにデータベース HSQL JDBC ドライバ(hsqldb.jar)を追加する必要があります。ドライバは<EspressChartInstall>/lib ディレクトリにあります。

```
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,
			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;

}
```

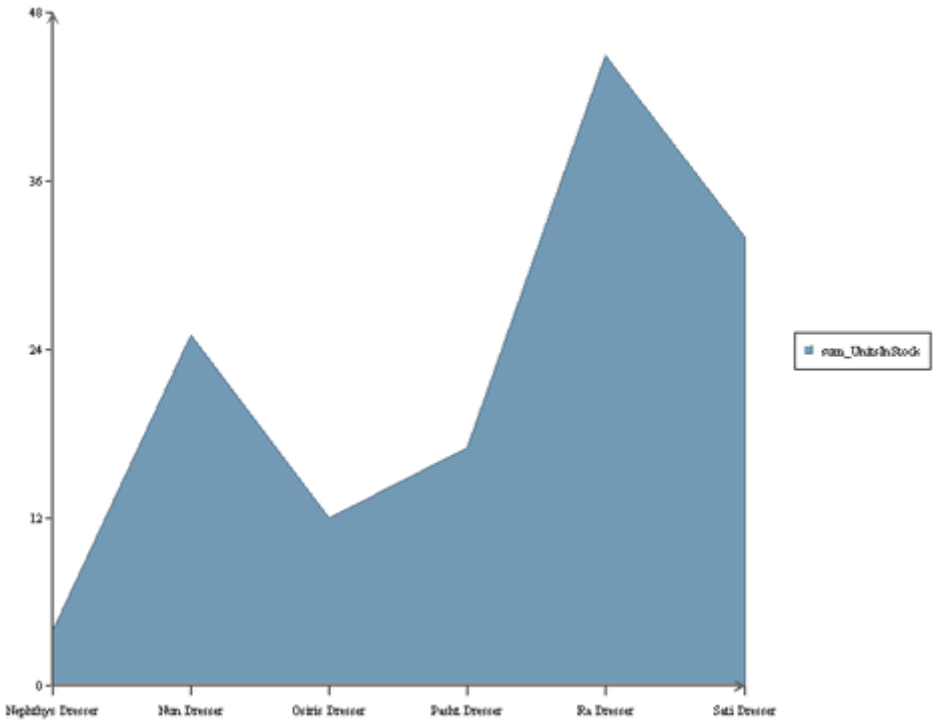
フルソースコード

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。



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。

12.12.14.3 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、横棒、折れ線、円、エリア、重ね合わせ、レーダー、ダイヤル、積み上げ縦棒と積み上げ横棒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。

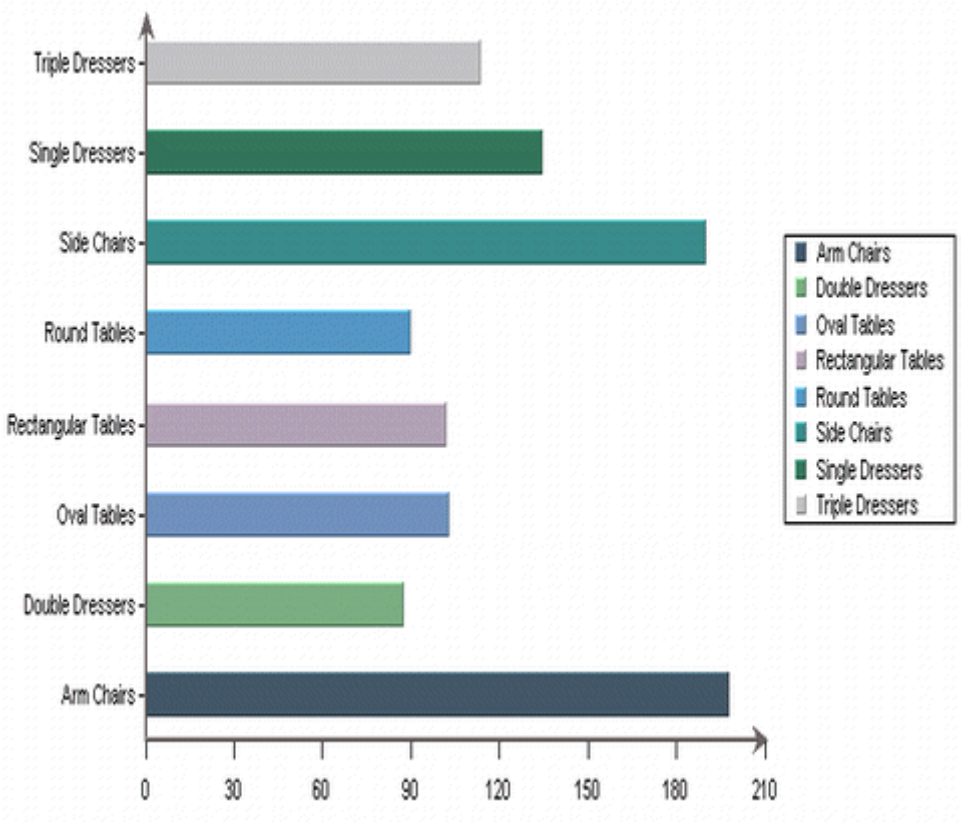
以下に、データドリルダウンチャートの例を示します。クエリが実行されるデータベースは WoodView HSQL なので、クラスパスにデータベース HSQL JDBC ドライバ(hsqldb.jar)を追加する必要があります。ドライバは<EspressChartInstall>/lib ディレクトリにあります。

```
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

        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;
}
```

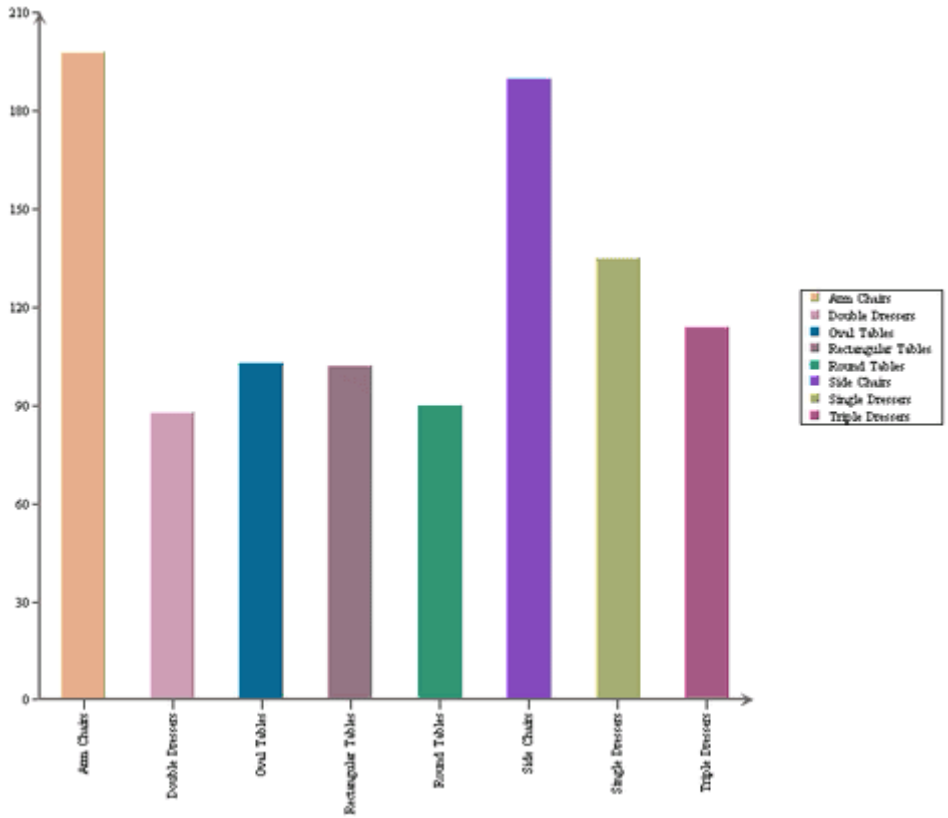
フルソースコード

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。



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。

13 Java サーブレットと Java サーバページ（JSP）

13.1 Java サーブレット

13.1.1 導入

Oracle Web サイトで説明されているように、Java サーブレットテクノロジは、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性のあるメカニズムを提供します。サーブレットコンテナは、Apache Web サーバ、Microsoft IIS などで利用できます。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、EspressChart に付属のサンプルサーブレット（EspressChart / help / examples / servlet / DatabaseJPEG）のセットアップ方法について説明します。各セクションでは、サンプルのデータベースサーブレットを具体的に扱っています。また、独自のサーブレットやその他のサーブレットの設定や実行のガイドとして使用することもできます。

データベース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。チャートの単純な表示（アプレットまたは静止画像のいずれか）から、チャートをブラウザに直接ストリーミングすることや、ドリルダウンチャートを静止画像として表示することまでさまざまです。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。

13.1.2 セットアップ

1. EspressManager が実行されていることを確認してください。
2. 使用しているサーブレットコンテナに応じて、以下のガイドラインに従ってください。次に、ブラウザから ExportChart2.html（EspressChart / help / examples / servlet / DataFile）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が必要になります。

これらの例はすべて、単純化のために単一スレッドモデルを使用しています。EspressChart API はマルチスレッド環境でも使用可能です。

以下は、EspressChart / help / examples / servlet の DataFile サーブレットの例を使用しています。

13.1.3 各サーブレットコンテナでのサーブレットの実行方法

13.1.3.1 Apache Tomcat

Tomcat のバージョンを確認:

Tomcat サーバ 5.5.20 または 6.0.10 がインストールされていることを確認してください。（このガイドは以前のバージョンにも適用されることがあります）。
WindowsOS 用のインストールファイルは 2 種類あります。

- **zip ファイル** **startup.bat** ファイルと **shutdown.bat** ファイルを使用して、サーバを起動および停止します。この動作は、bin ディレクトリのアーカイブを抽出した後に検出されます。
- **exe ファイル** Windows サービスを使用して **Apache Tomcat Monitor** をインストールします。

EspressManager を起動

<EspressChart のインストールディレクトリ>内の EspressManager.bat を実行して EspressManager を実行してください。

仮想ディレクトリマッピング

正常に動作させるため、Apache Tomcat の Web ルートの下に仮想ディレクトリとして EspressChart のインストールフォルダをマッピングします。
<Tomcatのインストールディレクトリ>¥conf配下にあるserver.xmlファイルをエディタで開き、次の文を<host></host>タグの間に追加してください。

```
<Context path="/EspressChart" docBase="c:¥EspressChart" debug="0" privileged="true"/>
```

<Tomcat のインストールディレクトリ>¥webapps¥root ディレクトリに EspressChart をインストールして、仮想ディレクトリのマッピングを回避することもできます。しかし、これはお勧めしません。

クラスパスの設定

EspressAPI.jar 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。
Servlet-api.jar ファイルは、<Tomcat のインストールディレクトリ>¥common¥lib ディレクトリに、EspressAPI.jar ファイルは EspressChart¥lib ディレクトリにあります。

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。
Tomcat 5.5 サーバ版の場合

```
set classpath =%classpath%; <Tomcat 5.5 インストールディレクトリ> ¥ common ¥ lib ¥ servlet-api.jar; C：¥ EspressChart ¥ lib ¥ EspressAPI.jar
```

Tomcat 6.0 サーバ版の場合

```
set classpath =%classpath%; <Tomcat 6.0 インストールディレクトリ> ¥ lib ¥ servlet-api.jar; C：¥ EspressChart ¥ lib ¥ EspressAPI.jar
```

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の簡略化のため推奨しています。

コントロールパネル ＞ システムとセキュリティ ＞ システム ＞ システムの詳細設定から、**詳細設定**のタブの**環境変数**をクリックすることで、クラスパス環境変数の設定をすることも可能です。

サンプルファイルのコンパイル（ExportServlet2.java）

コンパイルコマンドを入力します。
クラスパスの設定を行っていない場合

```
javac -classpath "<Tomcat 5.5 インストールディレクトリ>¥common¥lib¥servlet-api.jar; C:¥EspressChart¥lib¥EspressAPI.jar" ExportServlet2.java (for Tomcat 5.5 server version)
```

```
javac -classpath "<Tomcat 6.0 インストールディレクトリ dir>¥lib¥servlet-api.jar; C:¥EspressChart¥lib¥EspressAPI.jar" ExportServlet2.java (for Tomcat 6.0 server version)
```

クラスパスの設定を行った場合、非常にシンプルです。

```
javac ExportServlet2.java
```

Tomcat デプロイメントディレクトリにクラスファイルをコピーする

ExportServlet2.class ファイルを <Tomcat インストールディレクトリ>¥webapps¥root¥WEB-INF¥classes ディレクトリにコピーしてください。

<Tomcat のインストールディレクトリ> ¥ webapps ¥ ROOT ¥ WEB-INF ¥ classes ディレクトリはデフォルトでは存在しないため、作成してください。

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

このサンプルを実行するには、Web アプリケーションの lib ディレクトリに必要なライブラリを追加する必要があります。**EspressChart¥lib** ディレクトリにある **EspressAPI.jar** ファイルを **<Tomcat インストールディレクトリ> ¥ webapps ¥ ROOT ¥ WEB-INF ¥ lib** ディレクトリにコピーしてください。

<Tomcat のインストールディレクトリ> ¥ webapps ¥ ROOT ¥ WEB-INF ¥ classes ディレクトリはデフォルトでは存在しないため、作成してください。

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、アプリケーションの **web.xml** ファイルを変更する必要があります。**<Tomcat インストールディレクトリ> ¥ webapps ¥ ROOT ¥ WEB-INF ¥**ディレクトリに配置されています。

ExportServlet2 を登録するには、次のコードを**<web-app></web-app>**の間に追加してください:

```
<servlet>
        <servlet-name>ExportServlet2</servlet-name>
        <servlet-class>ExportServlet2</servlet-class>
        </servlet>

        <servlet-mapping>
        <servlet-name>ExportServlet2</servlet-name>
        <url-pattern>/servlet/ExportServlet2</url-pattern>
        </servlet-mapping>
```

また、webapp ディレクトリに配置されている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。サーブレットアプリケーションを登録する必要がなくなります。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 invoker サーブレットを使用することは推奨していません。また、サポートされていません。詳細は、Tomcat サーバのマニュアルを参照してください。

Tomcat サーバの開始

zip インストールファイルを使用した場合は、startup.bat(Linux の場合は startup.sh)ファイル(**<Tomcat のインストールディレクトリ> ¥ bin**)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。Tomcat サービスの起動に、Tomcat の **Monitor Tomcat** を使用することができます。

Tomcat サーバがすでに稼働している場合は、サンプルを実行するために再起動する必要があります。

サンプルの起動

最後に、このサンプルを実行します。このサンプルを実行するには、ブラウザの **ExportChart2.html** ファイル(**EspressChart ¥ help ¥ examples ¥ servlet ¥ DataFile** ディレクトリ配下)を実行します。データファイルへのパスを設定することも、EspressChart インストールに付属するデフォルトのパスを使用することもできます。**Get Chart Image** ボタンをクリックすると ExportServlet2 が呼び出され、チャートイメージがエクスポートされてページに印刷されます。

サンプルを起動する前に EspressManager が起動されていることを確認してください。

サーブレットは、サーブレットコンテナ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ます。クライアントを別のマシンに置くには、**ExportChart2.html** ファイルの文を上の文から下の文に変更します。

<form action = ”http://localhost:8080/servlet/ExportServlet2”method = ”post”>

<form action = ”http://<machine_name>:8080/servlet/ExportServlet2”method = ”post”>

13.1.3.2 JRun

- html ファイルのマシン名とポート番号を置換します。

```
<FORM ACTION=http://<machine name>:8100/servlet/ExportServlet method=POST>
```

<machine name>は、JRun サーバを実行しているマシンの名前です。

- JRun Default Server Administrator にログインします。
- **Java Settings > Classpath** の順にクリックします。
- **EspressAPI.jar** と **ExportLib.jar** のパスを別々の行に追加して、**Update** をクリックします。
- JRun デフォルトサーバを再起動します。
- サーブレットをコンパイルし、
  <jrun インストールディレクトリ>¥servers¥default¥default-app¥WEB-INF ¥lasses にコピーします。
- ブラウザで変更した **ExportChart2.html** を開き、パラメータを入力、ボタンをクリックすることでグラフを取得できます。

13.1.3.3 ColdFusion Server

- **ExportChart2.html** の名前を **ExportChart2.cfm** に変更し、cfm ファイル内のマシン名とポート番号を次のように変更します。

```
<FORM ACTION=http://<machine name>:8100/servlet/ExportServlet method=POST>
```

<machine name>は、JRun サーバを実行しているマシンの名前です。

- JRun Default Server　Administrator にログインします。
- **Java Settings > Classpath** の順にクリックします。
- **EspressAPI.jar** と **ExportLib.jar** のパスを別々の行に追加して、**Update** をクリックします。
- JRun デフォルトサーバを再起動します。
- サーブレットをコンパイルし、
  <jrun インストールディレクトリ>¥servers¥default¥default-app¥WEB-INF¥classes にコピーします。
- 変更した **ExportChart2.cfm** をブラウザで開き、パラメータを入力、ボタンをクリックすることでグラフを取得できます。

13.1.3.4 WebLogic 6.0

NT プラットフォームの場合、WebLogic 6.0 で EspressChart を実行するには、**bea¥wlserver6.0¥config¥examples¥applications¥examplesWebApp ディレクトリ**にインストールします。これにアクセスするには、**http://<Web アドレスまたはマシン名>:7001/examplesWebApp/EspressChart/index.html** と入力します。最初に EspressManager を起動し、Chart Designer にアクセスすることを忘れないでください。

サンプルのサーブレットを設定して実行するには、まず **wlserver6.0¥config¥examples ディレクトリ**に移動する必要があります。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。

1. **@rem Set user-defined variables** セクションに **ES_CHART = YOUR_WEBROOT ¥ espresschart** を追加します。**C:¥bea** の下に WebLogic 6.0 をインストールした場合、**YOUR_WEBROOT** は **C:¥bea¥wlserver6.0¥config¥examples¥applications** に等しくなります。**ES_CHART** 変数には、マシン内の EspressChart 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。

2. 同じファイルの **CLASSPATH** フィールドに **C：¥ bea;**、**% ES_CHART % ¥ lib ¥ EspressAPI.jar;**、**%ES_CHART%¥ lib ¥ ExportLib.jar;**を追加します。

次に、以下の手順に従います（ファイルは **espresschart ¥ help ¥ examples ¥ servlet ¥ Weblogic ディレクトリ**にあります）。

1. **ExportChart2.html** を **wlserver6.0 ¥ config ¥ examples ¥ applications ¥ examplesWebApp ディレクトリ**に置きます。

2. コード **archon：7001** のマシン名を **ExportChart2.html** ファイルの **yourMachineName:7001** に変更します。

3. **ExportServlet2.java** ファイルを **wlserver6.0¥samples¥examples¥servlets ディレクトリ**に置きます。

4. **wlserver6.0¥config¥examples¥applications¥examplesWebApp¥WEB-INF¥ ディレクトリ**の **examplesWebApp** ディレクトリにある **web.xml** ファイルに、次のコードを挿入します。**<servlet >**コードはファイルの**<servlet>**コードセクション、**<servlet-mapping>**コードは**<servlet-mapping>**セクションに挿入します。

```
<servlet>
	<servlet-name>ExportServlet2</servlet-name>
	<servlet-class>ExportServlet2</servlet-class>
		<init-param>
			<param-name>dataFileName</param-name>
			<param-value>help/examples/data/sample.dat</param-value>
		</init-param>
		<init-param>
			<param-name>category</param-name>
			<param-value>0</param-value>
		</init-param>
		<init-param>
			<param-name>series</param-name>
			<param-value>-1</param-value>
		</init-param>
		<init-param>
			<param-name>sumby</param-name>
			<param-value>-1</param-value>
		</init-param>
		<init-param>
			<param-name>value</param-name>
			<param-value>3</param-value>
		</init-param>
		<init-param>
			<param-name>chartType</param-name>
			<param-value>Column</param-value>
		</init-param>
		<init-param>
			<param-name>fileName</param-name>
			<param-value>c:¥temp</param-value>
		</init-param>
	</servlet>
<servlet-mapping>
	<servlet-name>ExportServlet2</servlet-name>
	<url-pattern>/ExportServlet2/*</url-pattern>
</servlet-mapping>
```

5. コマンドプロンプトウィンドウを開き、**wlserver6.0¥config¥examples ディレクトリ**に移動し、**setExamplesEnv** を実行します。
6. **wlserver6.0¥samples¥examples¥servlets** ディレクトリに移動し、以下に示すコマンドラインを使用して **ExportServlet2.java** をコンパイルします。

```
javac –d %EX_WEBAPP_CLASSES% ExportServlet2.java
```

7. 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**に移動して、**startExamplesServer** と入力し **Enter** を押して、**WebLogic Server** を起動します。
8. ブラウザを起動し、**http://yourmachineName:7001/examplesWebApp/ExportChart2.html** に移動し、サーブレットサンプルを確認します。

トラブルシューティングの際には、タイピングミスがないか確認してください。

13.1.3.4.1 Web Logic 9.2

次の手順は、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になります。手順では、システムに WebLogic 9.2 Server が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>**、EspressChart インストールの場所は**<EC_INSTALL_DIR>**として参照されます。

WebLogic 9.2 で EspressChart を実行するには、**<WL_INSTALL_DIR>¥samples¥server¥examples¥build¥examplesWebApp ディレクトリ**にインストールします。これにアクセスするには、**http://<Web アドレスまたはマシン名>:7001/examplesWebApp/EspressChart/index.html** と入力します。最初に EspressManager を起動し、Chart Designer にアクセスすることを忘れないでください。

サンプルサーブレット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**に移動する必要があります。次の手順に従って setExamplesEnv.cmd ファイルを変更します。

1. **ES_CHART=<EC_INSTALL_DIR>**を追加します。**C:¥bea** 配下に WebLogic 9.2 をインストールしている場合は、**<EC_INSTALL_DIR>**は、**C:¥bea¥weblogic92¥samples¥server¥examples¥build¥examples¥WebApp¥EspressChart** になります。**ES_CHART** 変数には、マシン内の EspressChart ホームディレクトリへのパスが含まれます。マシンの環境に合わせてパスを変更します。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なっていることを確認してください。
2. **%ES_CHART%¥lib¥EspressAPI.jar;**、**%ES_CHART%lib¥ExportLib.jar;**を同じファイルの **CLASSPATH** フィールドに追加します。

次に、以下の手順に従って操作します。ファイルは**<EC_INSTALL_DIR>¥help¥examples¥servlet¥Weblogic ディレクトリ**にあります。

1. **DialServletDemo.java** と **DialServletExp.java** を**<WL_INSTALL_DIR>¥samples¥server¥examples¥build¥examples¥WebApp¥WEB-INF¥classes ディレクトリ**に移動します。
2. **DialServletDemo.java** コードの **archon:7001** を **yourMachineName:7001** に変更します。
3. **DialServletExp.java** ファイルのコンストラクタを次のように変更します。
   変更前

```
chart = new QbChart(new Frame(), QbChart.VIEW2D, QbChart.DIAL, data, colInfo, "config/examples/applications/examplesWebApp/WEB-INF/classes/dial.tpl");
```

変更後

```
chart = new QbChart(new Frame(), QbChart.VIEW2D, QbChart.DIAL, data, colInfo, "samples/server/examples/examplesWebApp/WEB-INF/classes/dial.tpl");.
```

4. **<WL_INSTALL_DIR>¥samples¥server¥examples¥build¥examplesWebApp¥WEB-INF ディレクトリ内の web.xml** のコードの以下の文を追加します。**<servlet>**コードセクションおよび**<servlet-mapping>**コードセクションは、それぞれのセクションに挿入します。

```
<servlet>

        <servlet-name>DialServletExp</servlet-name>
        <servlet-class>DialServletExp</servlet-class>

</servlet>

<servlet>

        <servlet-name>DialServletDemo</servlet-name>
        <servlet-class>DialServletDemo</servlet-class>

</servlet>

<servlet-mapping>

        <servlet-name>DialServletExp</servlet-name>
        <url-pattern>/DialServletExp/*</url-pattern>

</servlet-mapping>

<servlet-mapping>

        <servlet-name>DialServletDemo</servlet-name>
        <url-pattern>/DialServletDemo/*</url-pattern>

</servlet-mapping>
```

5. **DialServletDemo.html** ファイルを **<WL_INSTALL_DIR>¥samples¥server¥examples¥build¥examplesWebApp** ディレクトリにコピーします。
6. **DialServletDemo.html** ファイルを編集し、すべてのマシン名を **archon:7001/servlet** から

yourMachineName:7001/exampleWebApp に変更します。

7. **Dial.tpl** ファイルを **<WL_INSTALL_DIR>¥samples¥server¥examples¥build¥examples¥WebApp¥WEB-INF¥classes ディレクトリ**にコピーします。
8. **<WL_INSTALL_DIR>¥samples¥server¥examples¥build¥examplesWebApp¥WEB-INF¥classes** ディレクトリに移動し、**DialServletExp.java** および **DialServletDemo.java** をコンパイルします。クラスパスに **EspressAPI.jar**、**ExportLib.jar** および **javax.servlet.jar** を含めてください。
   javax.servlet.jar は bea¥jrockit90_150_06¥mercuryprofiler¥lib ディレクトリにあります。
9. 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、**Enter** キーを押して WebLogic Server を起動します。
10. ブラウザを開き、**http://yourMachineName:7001/examplesWebApp/DialServletDemo.html** に飛び、サーブレットサンプルを表示します。

トラブルシューティングの際には、タイピングミスがないか確認してください。

13.1.3.5 WebSphere 3.5

1. WebSphere 管理コンソールを開き、ノード（通常はマシン名）**<MACHINE_NAME>** > **Default Servlet Engine** > **Default App** の順にクリックし、**Advanced** タブをクリックし、**EspressAPI.jar** と **ExportLib.jar** クラスパスに別々の行に追加し、**Apply** をクリックします。
2. コンパイル済みクラスファイル（**ExportServlet2.class**）を**<WebSphere インストールディレクトリ>¥AppServer¥servlets** ディレクトリに移動します
3. **FORM ACTION** タグを次のように変更します。

```
<FORM ACTION = http://<マシン名>/servlet/ExportServlet2 method = POST>
```

4. ノードの下にあるデフォルトサーバを再起動します。

13.1.3.5.1 WebSphere 6.1

以下の手順は、WebSphere 6.1 配下にサンプルサーブレットを設定して実行する方法になります。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。本項目では、WebLogic インストールの場所は**<WS_INSTALL_DIR>**、EspressChart インストールの場所は**<EC_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 **<EC_INSTALL_DIR>¥lib¥EspressAPI.jar**、**<EC_INSTALL_DIR>¥lib¥ExportLib.jar** および **<EC_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 **<EC_INSTALL_DIR>¥help¥examples¥servlet¥DataFile** ディレクトリにあります）。

1. **ExportServlet2.java** ファイルを **<WS_INSTALL_DIR>¥AppServer¥profiles¥AppSrv01¥installedApps¥<YourMachineName>Node01Cell¥ivtApp.ear¥ivt_app.war¥WEB-INF¥classes** ディレクトリに移動します。
2. **<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を挿入します。

コードの **<servlet>** は **<servlet>** セクションに、**<servlet-mapping>** は **<servlet-mapping>**セクションに追加します。

```
<servlet>
        <servlet-name>ExportServlet2</servlet-name>
        <servlet-class>ExportServlet2</servlet-class>
</servlet>

<servlet-mapping>
        <servlet-name>ExportServlet2</servlet-name>
        <url-pattern>ExportServlet2</url-pattern>
</servlet-mapping>
```

3. **ExportChart2.html** ファイルのマシン名とポートを **yourmachine:8080** から **yourMachineName:9080/ivt** に変更します。
4. **ExportChart2.html** を **<WS_INSTALL_DIR>¥AppServer¥profiles¥AppSrv01¥installedApps¥<YourMachineName>Node01Cell¥ivtApp.ear¥ivt_app.war** ディレクトリにコピーします。
5. ExportServlet2.java ファイルを編集し、**http://yourmachine:8080/EspressChart/**から**<EC_INSTALL_DIR>**に変更します。
6. **<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、**ExportServlet2.java** をコンパイルします。**EspressAPI.jar**、**ExportLib.jar** および **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。

7. WebSphere サーバを開始します。いくつか方法がありますが、最も簡単に行う方法として、**スタート > プログラム > IBM WebSphere > Application Server v6.1 > Profiles > AppSrv01 > First Steps** から **First Steps** ツールを起動することです。ファーストステップツールから管理コンソールを起動することもできます。
8. Web ブラウザを開き、**http://yourMachineName:9080/ivt/ExportChart2.html** に移動して、サーブレットサンプルを表示します。

トラブルシューティングの際には、タイピングミスがないか確認してください。

13.1.3.6 JBoss 4.0.5

次の手順は、JBoss 4.0.5 配下にサンプルサーブレット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。本項目では JBoss 4.0.5 インストールの場所は**<JB_INSTALL_DIR>**、EspressChart インストールの場所は**<EC_INSTALL_DIR>**として参照されます。
サンプルサーブレットを設定して実行するには、まず**<JB_INSTALL_DIR>¥bin ディレクトリ**に移動する必要があります。次の手順に従って、**run.bat** ファイルを変更します。

1. **set ES_CHART = <EC_INSTALL_DIR>**を追加します。**ES_CHART** 変数には、マシン内の EspressChart ホームディレクトリへのパスが含まれています。

2. 同じファイルの **JBOSS_CLASSPATH** フィールドに **%ES_CHART%¥lib¥EspressAPI.jar;**、**%ES_CHART%¥lib¥ExportLib.jar;**、**%ES_CHART%¥lib¥qblicense.jar;**を追加します。ダブルコーテーション

(“”)は入力しないでください。

次に、以下の手順に従います(ファイルは<EC_INSTALL_DIR>¥help¥examples¥servlet¥DataFile ディレクトリにあります)。

1. ExportServlet2.html を <JB_INSTALL_DIR>¥server¥default¥deploy¥jmx-console.war¥WEB-INF¥classes ディレクトリに置きます。

2. ExportServlet2.java ファイルのコード http://yourmachine:8080/EspressChart から <EC_INSTALL_DIR>に変更します。

3. <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、<servlet-mapping>コードは<servlet-mapping>セクションに挿入します。

```
<servlet>
        <servlet-name>ExportServlet2</servlet-name>
        <servlet-class>ExportServlet2</servlet-class>
</servlet>

<servlet-mapping>
        <servlet-name>ExportServlet2</servlet-name>
        <url-pattern>/ExportServlet2</url-pattern>
</servlet-mapping>
```

4. ExportChart2.html ファイルを編集し、URL を http://yourmachine:8080/servlet から http://yourMachineName:8080/jmx-console に変更します。
5. <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。クラスパスに EspressAPI.jar、ExportLib.jar および javax.servlet.jar を含めてください。
   javax.servlet.jar は<JB_INSTALL_DIR> ¥ server ¥ all ¥ lib ディレクトリにあります。

6. 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て、run.bat と入力し Enter を押して、JBoss server を起動します。
7. ブラウザを起動し、http://yourmachineName:8080/jmx-console/ExportChart2.html に移動し、サンプルサーブレットを表示します。

   トラブルシューティングの際には、タイピングミスがないか確認してください。

13.1.3.7 Orion 2.0.7

次の手順は、Orion 2.0.7 配下のサンプルサーブレット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。本項目では Orion のインストールの場所は<OR_INSTALL_DIR>、EspressChart インストールの場所は<EC_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<EC_INSTALL_DIR>¥help¥examples¥servlet¥DataFile ディレクトリにあります。

1. **ExportServlet2.java** ファイルを **<OR_INSTALL_DIR>¥default-web-app¥WEB-INF¥classes** ディレクトリに移動します。

2. **ExportServlet2.java** ファイルを編集し、**http://yourmachine:8080/EspressChart/**部分を**<EC_INSTALL_DIR>**に変更します。

3. **<OR_INSTALL_DIR>¥default-web-app¥WEB-INF** ディレクトリにある **web.xml** ファイルに以下のコードを挿入します。**<servlet>**コードは**<servlet>**セクションに、**<servlet-mapping>**コードは**<servlet-mapping>**セクションに挿入してください。

```
<servlet>
        <servlet-name>ExportServlet2</servlet-name>
        <servlet-class>ExportServlet2</servlet-class>
</servlet>

<servlet-mapping>
        <servlet-name>ExportServlet2</servlet-name>
        <url-pattern>ExportServlet2</url-pattern>
</servlet-mapping>
```

4. **ExportChart2.html** ファイルを編集し、URL 部分を **http://yourmachine:8080/servlet** から **http://yourMachineName:80** に変更します。
5. **ExportChart2.html** ファイルを**<OR_INSTALL_DIR>¥default-web-app** ディレクトリにコピーします。
6. **＜OR_INSTALL_DIR>¥default-web-app¥WEB-INF¥classes** ディレクトリに移動し、**ExportServlet2.java** をコンパイルします。クラスパスに、**EspressAPI.jar**、**ExportLib.jar**、**j2ee.jar** を含めてください。
   <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。

7. コマンドプロンプトウィンドウから、**<OR_INSTALL_DIR>**ディレクトリに移動し、**java –jar orion.jar** と入力、**Enter** キーを押して Orion サーバを起動します。
8. ウェブブラウザから **http://yourMachineName:80/ExportChart2.html** と入力し、サンプルサーブレットを表示します。

トラブルシューティングの際には、タイピングミスがないか確認してください。

13.1.3.8 JRun 4 (Update 6)

次の手順は、JRun 4(Update 6)配下にサンプルサーブレットを設定して実行する方法を示しています。この手順では、JRun 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。本項目では JRun のインストールの場所は**<JR_INSTALL_DIR>**、EspressChart インストールの場所は**<EC_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**<EC_INSTALL_DIR>¥help¥examples¥servlet¥DataFile** ディレクトリにあります。

1. **ExportServlet2.java** ファイルを**<JR_INSTALL_DIR>¥servers¥default¥default-ear¥default-war¥WEB-INF** ディレクトリに移動します。

2. **ExportServlet2.java** ファイルを編集し、**http://yourmachine:8080/EspressChart/**部分を**<EC_INSTALL_DIR>**に変更します。

3. **<JR_INSTALL_DIR>¥servers¥default¥default-ear¥default-war¥WEB-INF** ディレクトリにある **web.xml** ファイルに以下のコードを挿入します。**<servlet>**コードは**<servlet>**セクションに、**<servlet-mapping>**コードは**<servlet-mapping>**セクションに挿入してください。

```
<servlet>
        <servlet-name>ExportServlet2</servlet-name>
        <servlet-class>ExportServlet2</servlet-class>
</servlet>

<servlet-mapping>
        <servlet-name>ExportServlet2</servlet-name>
        <url-pattern>ExportServlet2</url-pattern>
</servlet-mapping>
```

4. **ExportChart2.html** ファイルを編集し、URL 部分を **http://yourmachine:8080/servlet** から **http://yourMachineName:8100** に変更します。
5. **ExportChart2.html** ファイルを **<JR_INSTALL_DIR>¥servers¥default¥default-ear¥default-war** ディレクトリにコピーします。
6. **<JR_INSTALL_DIR>¥servers¥default¥default-ear¥default-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。クラスパスに、**EspressAPI.jar**、**ExportLib.jar**、**j2ee.jar** を含めてください。
<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。

7. **EspressAPI.jar**、**ExportLib.jar**、**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。
8. **<JR_INSTALL_DIR>¥bin¥jrun.exe** を実行し、JRun 4 アプリケーションサーバを起動します。**JRun Launcher** ウィンドウが起動します。**Default** サーバが開始されます。
9. ウェブブラウザから **http://yourMachineName:8100/ExportChart2.html** と入力し、サンプルサーブレットを表示します。

トラブルシューティングの際には、タイピングミスがないか確認してください。

13.1.3.9 Oracle 10g (10.1.3.1.0)

次の手順は Oracle 10g (10.1.3.1.0)配下にサンプルサーブレットを設定して実行する方法を示しています。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。本項目では Oracle 10g のインストールの場所は **<ORA_INSTALL_DIR>**、EspressChart インストールの場所は**<EC_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 **<EC_INSTALL_DIR>¥help¥examples¥servlet¥StreamingChart** ディレクトリにあります。

1. **StreamChartServlet.java** ファイルを **<ORA_INSTALL_DIR>¥j2ee¥home¥default-web-app¥WEB-INF¥classes** ディレク

トリに移動します。

2. StreamChartServlet.java ファイルを編集し、http://yourmachine:8080/EspressChart/部分を<EC_INSTALL_DIR>に変更します。

3. <ORA_INSTALL_DIR>¥j2ee¥home¥default-web-app¥WEB-INF ディレクトリにある web.xml ファイルに以下のコードを挿入します。<servlet>コードは<servlet>セクションに、<servlet-mapping>コードは<servlet-mapping>セクションに挿入してください。

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

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

4. StreamChartServlet.html ファイルを編集し、URL 部分を http://yourmachine:8080/servlet から http://yourMachineName:8888/servlet に変更します。
5. StreamChartServlet.html ファイルを<ORA_INSTALL_DIR>¥j2ee¥home¥default-web-app ディレクトリにコピーします。
6. <ORA_INSTALL_DIR>¥j2ee¥home¥default-web-app¥WEB-INF¥classes ディレクトリに移動し、StreamChartServlet.java をコンパイルします。クラスパスに、EspressAPI.jar、ExportLib.jar、servlet.jar を含めてください。
servlet.jar ファイルは<ORA_INSTALL_DIR>¥j2ee¥home¥lib ディレクトリ配下にあります。

7. EspressAPI.jar、ExportLib.jar、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。
8. Oracle サーバを開始します。サーバの開始を最もすばやく行う方法は、スタート>プログラム>Oracle > Star Soa Suite と起動します。また、<ORA_INSTALL_DIR>¥j2ee¥home¥bin ディレクトリ配下の runstartupconsole.bat を実行することでも開始することができます。
9. ウェブブラウザから http://yourMachineName:8888/StreamChartServlet.html と入力し、サンプルサーブレットを表示します。

トラブルシューティングの際には、タイピングミスがないか確認してください。

13.1.3.10 Sun Java System Application PE (9.0, 8.2)

次の手順は、Sun Java System Application PE (9.0 または 8.2)配下にサンプルサーブレットを設定して実行する方法を示しています。この手順では、Sun Java System Application PE 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。本項目では Sun Java System Application PE のインストールの場所は<SAP_INSTALL_DIR>、EspressChart インストールの場所は<EC_INSTALL_DIR>として参照されます。

1. ユーザ用のディレクトリを作成します。このディレクトリのパスは<USER_DIR>として表記します。
2. <SAP_INSTALL_DIR>¥samples¥quickstart ディレクトリに移動し、hello.war ファイルを

<USER_DIR>にコピーします。その後 **hello.war** ファイルを解凍します。

war ファイルを展開するには、jar または unzip を使用します。jar を使用して解凍するには、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 **jar -xf hello.war** コマンドを実行します。

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。

1. **ExportServlet2.java** ファイルを**<USER_DIR>¥WEB-INF¥classes** ディレクトリに移動します。

2. **ExportServlet2.java** ファイルを編集し、**http://yourmachine:8080/EspressChart/**部分を**<EC_INSTALL_DIR>**に変更します。

3. **<USER_DIR>¥WEB-INF** ディレクトリにある **web.xml** ファイルに以下のコードを挿入します。

```
<servlet>
        <servlet-name>ExportServlet2</servlet-name>
        <servlet-class>ExportServlet2</servlet-class>
</servlet>

<servlet-mapping>
        <servlet-name>ExportServlet2</servlet-name>
        <url-pattern>ExportServlet2</url-pattern>
</servlet-mapping>
```

4. **ExportChart2.html** ファイルを編集し、URL 部分を **http://yourmachine:8080/servlet** から **http://yourMachineName:8080/hello/servlet** に変更します。
5. **ExportChart2.html** ファイルを**<USER_DIR>**ディレクトリにコピーします。
6. **<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。クラスパスに、**EspressAPI.jar**、**ExportLib.jar**、**j2ee.jar** を含めてください。
<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。

7. コマンドプロンプトウィンドウで**<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 *.
```

8. **hello.war** ファイルを**<SAP_INSTALL_DIR>¥bin¥asadmin start-domain domain1** ディレクトリに移動します。
9. **<SAP_INSTALL_DIR>¥bin¥asadmin start-domain domain1** を実行して Sun App サーバを開始します。Windows ユーザの場合、最も簡単な方法は、**スタート > プログラム > Sun Microsystems > Application Server PE 9 > Start Default Server** を実行することです。
10. ウェブブラウザから **http://yourMachineName:8080/hello/ExportChart2.html** と入力し、サンプルサーブレットを表示します。

トラブルシューティングの際には、タイピングミスがないか確認してください。

13.1.3.11 Sun Java System WebServer 7.0

次の手順は、Sun Java System WebServer 7.0 にサンプルサーブレットを設定して実行する方法を示しています。この手順では、Sun Java SystemWebServer 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。本項目では Sun Java System WebServer のインストールの場所は**<SWS_INSTALL_DIR>**、EspressChart インストールの場所は**<EC_INSTALL_DIR>**として参照されます。

1. ユーザ用のディレクトリを作成します。このディレクトリのパスは**<USER_DIR>**として表記します。
2. **<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>** にコピーします。その後 **webapps-simple.war** ファイルを解凍します。

war ファイルを展開するには、jar または unzip を使用します。jar を使用して解凍するには、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 **jar –xf webapps-simple.war** コマンドを実行します。

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。

1. **ExportServlet2.java** ファイルを**<USER_DIR>¥WEB-INF¥classes** ディレクトリに移動します。

2. **ExportServlet2.java** ファイルを編集し、**http://yourmachine:8080/EspressChart/**部分を**<EC_INSTALL_DIR>**に変更します。

3. **<USER_DIR>¥WEB-INF** ディレクトリにある **web.xml** ファイルに以下のコードを挿入します。**<servlet>** コードは **<servlet>** セクションに、**<servlet-mapping>** コードは **<servlet-mapping>**セクションに挿入してください。

```
<servlet>
        <servlet-name>ExportServlet2</servlet-name>
        <servlet-class>ExportServlet2</servlet-class>
</servlet>

<servlet-mapping>
        <servlet-name>ExportServlet2</servlet-name>
        <url-pattern>ExportServlet2</url-pattern>
</servlet-mapping>
```

4. **ExportChart2.html** ファイルを編集し、URL 部分を **http://yourmachine:8080/servlet** から **http://yourMachineName:81/webapps-simple/servlet** に変更します。
5. **ExportChart2.html** ファイルを**<USER_DIR>**ディレクトリにコピーします。
6. **<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。クラスパスに、**EspressAPI.jar**、**ExportLib.jar**、**j2ee.jar** を含めてください。
<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。

7. コマンドプロンプトウィンドウで<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 webapps-simple.war *.
```

8. **EspressAPI.jar**、**ExportLib.jar**、**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。
9. **<SWS_INSTALL_DIR>¥admin-server¥bin¥startserv.bat** を実行して管理サーバを開始します。その後、Web ブラウザを起動し、管理コンソールページ **http://yourMachineName:8989** に移動します。管理コンソールのページから、管理サーバにログインして、**Virtual Server** タスクの **Add web application** リンクをクリックします。
10. これにより、**Add Web Application** ウィンドウが表示されます。**Add Web Application** ウィンドウから **Browse** ボタンをクリックし、**<USER_DIR>¥webapps-simple.war** ファイルを参照します。アプリケーションのコンテキストルートを表し、サーバーホストに関連する 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。
11. 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。次に、**Save** ボタンをクリックします。その後、画面の右上に **Deployment Pending Warning** 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。
12. デプロイが成功した場合は、**Results** ウィンドウが表示されます。これは、使用可能なすべてのノードに構成が正常にデプロイされていることを示します。
13. ウェブブラウザから **http://yourMachineName:81/webapps-simple/ExportChart2.html** と入力し、サンプルサーブレットを表示します。

トラブルシューティングの際には、タイピングミスがないか確認してください。

13.1.3.12 Resin 3.1.0

次の手順は Resin 3.1.0 配下にサンプルサーブレットを設定して実行する方法を示しています。この手順では、Resin 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。本項目では Resin 3.1.0 インストールの場所は**<RES_INSTALL_DIR>**、EspressChart インストールの場所は**<EC_INSTALL_DIR>**として参照されます。

1. ユーザ用のディレクトリを作成します。作成したディレクトリのパスは**<USER_DIR>**と表記します。
2. **<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。**resin-doc.war** ファイルを解凍します。

war ファイルを展開するには、jar または unzip を使用します。jar を使用して解凍するには、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 **jar –xf webapps-simple.war** コマンドを実行します。

3. **<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>¥webapps**にコピーします。その後、**servlet-hello** ディレクトリを **test** にリネームします。

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。

1. **ExportServlet2.java** ファイルを **<RES_INSTALL_DIR>¥webapps¥test¥WEB-INF¥classes** ディレクトリに移動します。

2. **ExportServlet2.java** ファイルを編集し、**http://yourmachine:8080/EspressChart/**部分を**<EC_INSTALL_DIR>**に変更します。

3. **<RES_INSTALL_DIR>¥j2ee¥webapps¥test¥WEB-INF** ディレクトリにある **resin-web.xml** ファイルに以下のコードを挿入します。

```
<servlet-name="ExportServlet2" servlet-class="ExportServlet2"/>
<servlet-mapping url-pattern="/servlet/ExportServlet2" servlet-name="ExportServlet2"/>
```

4. **ExportChart2.html** ファイルを編集し、URL 部分を **http://yourmachine:8080/servlet** から **http://yourMachineName:8080/test/servlet** に変更します。
5. **ExportChart2.html** ファイルを**<RES_INSTALL_DIR>¥webapps¥test** ディレクトリにコピーします。
6. **< RES_INSTALL_DIR>¥webapps¥test¥WEB-INF¥classes** ディレクトリに移動し、**ExportChart2.java** をコンパイルします。クラスパスに、**EspressAPI.jar**、**ExportLib.jar**、**j2ee.jar** を含めてください。
   <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。

7. **EspressAPI.jar**、**ExportLib.jar**、**qblicense.jar** ファイルを**<RES_INSTALL_DIR>¥lib** ディレクトリにコピーします。
8. **<RES_INSTALL_DIR>**ディレクトリ配下の **httpd.exe** を実行し Resin サーバを開始します。
9. ウェブブラウザから **http://yourMachineName:8080/test/ExportChart2.html** と入力し、サンプルサーブレットを表示します。

トラブルシューティングの際には、タイピングミスがないか確認してください。

13.1.3.13 ColdFusion MX 7.02

次の手順は ColdFusion MX 7.02 にサンプルサーブレットを設定して実行する方法を示しています。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。本項目では ColdFusion のインストールの場所は **<CF_INSTALL_DIR>**、EspressChart インストールの場所は**<EC_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**<EC_INSTALL_DIR>¥help¥examples¥servlet¥DataFile** ディレクトリにあります。

1. **ExportServlet2.java** ファイルを **<CF_INSTALL_DIR>¥wwwroot¥WEB-INF¥classes** ディレクトリに移動します。

2. **ExportServlet2.java** ファイルを編集し、**http://yourmachine:8080/EspressChart/**部分を**<EC_INSTALL_DIR>**に変更します。

3. **<CF_INSTALL_DIR>¥wwwroot¥WEB-INF** ディレクトリにある **web.xml** ファイルに以下のコードを挿入します。**<servlet>**コードは**<servlet>**セクションに、**<servlet-mapping>**コードは**<servlet-mapping>**セクションにそれぞれ挿入してください。

```
<servlet>
        <servlet-name>ExportServlet2</servlet-name>
        <servlet-class>ExportServlet2</servlet-class>
</servlet>

<servlet-mapping>
        <servlet-name>ExportServlet2</servlet-name>
        <url-pattern>/servlet/ExportServlet2</url-pattern>
</servlet-mapping>
```

4. **ExportChart2.html** ファイルを編集し、URL 部分を **http://yourmachine:8080/servlet** から **http://yourMachineName:8500** に変更します。
5. **ExportChart2.html** ファイルを**<CF_INSTALL_DIR>¥wwwroot** ディレクトリにコピーします。
6. **< CF_INSTALL_DIR>¥wwwroot¥WEB-INF¥classes** ディレクトリに移動し、**ExportChart2.java** をコンパイルします。クラスパスに、**EspressAPI.jar**、**ExportLib.jar**、**j2ee.jar** を含めてください。
   <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。

7. **EspressAPI.jar** 、 **ExportLib.jar** 、 **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。
8. ColdFusion アプリケーションサーバを開始します。Windows プラットフォームの場合、ColdFusion アプリケーションサーバはサービスとして動作されるようインストールされています。したがって自動的に起動されます。サーバが動作していない場合は、Windows サービスから ColdFusion MX 7 アプリケーションサーバサービスを起動します。必要に応じて、アプリケーションサーバを再起動してください。
9. ウェブブラウザから **http://yourMachineName:8500/ExportChart2.html** と入力し、サンプルサーブレットを表示します。

トラブルシューティングの際には、タイピングミスがないか確認してください。

13.1.4 Java 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 **ExportChart2.html**（**EspressChart/help/examples/servlet/DataFile** ディレクトリにあります）を開きます。パラメータを指定し、**Get Chart Image** をクリックします。

最新の結果を取得する場合、メモリキャッシュおよびディスクキャッシュをクリアする必要があります。

13.2 Java サーバページ（JSP）

13.2.1 導入

Java サーバページ(JSP)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、動的コンテンツのコードは特別なタグ(通常は<%で始まり%>で終わる)内で HTML 内に囲まれます。

JSP は、通常の HTML ファイルと一緒に配置することができ、さまざまな方法で HTML ファイルのように動作します。ただし、JSP が実行されるたび、静的な HTML が印刷されている間は、ページは通常のサーブレットに変換されます。

JSP は基本的にサーブレットテクノロジの拡張版です。しかし、JSP の使用には大きな利点が 1 つあります。JSP テクノロジは、ユーザーインターフェイスとコンテンツ生成を分離して、デザイナが基本的な動的コンテンツを変更することなくページレイアウト全体を変更できるようになります。

EspressChart にはいくつかの JSP のサンプルがあります。ここでは、JSWDK で JSP のサンプルを実行する方法を説明します。ここでは、データファイルサーブレットの例から変更された例を示します(**EspressChart/help/examples/jsp/Chart** 配下にあります)。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。

13.2.2 各サーブレットコンテナでの実行方法

13.2.2.1 Apache Tomcat

1. **EspressAPI.jar、ExportLib.jar、servlet.jar** をクラスパスに含めてください。例えば、

```
set classpath = C:¥netscape¥suitespot¥docs¥espresschart¥lib¥EspressAPI.jar;
C:¥netscape¥suitespot¥docs¥espresschart¥lib¥ExportLib.jar;
C:¥tomcat¥common¥lib¥servlet.jar;
```

2. **webapps¥examples¥jsp¥**の下に **chart** のサブディレクトリを作成します。したがって、Tomcat が C:¥の下にインストールされている場合は、新しいディレクトリとして、C:¥tomcat¥webapps¥examples¥jsp¥chart が作成されます。
3. **ExportChart.jsp**、**ExportChartResponse.jsp** および **error.jsp** を **examples¥jsp¥chart** ディレクトリにコピーします。
4. **tomcat¥webapps** ディレクトリの下に **chart** という別のディレクトリを作成します。これで、**C:¥tomcat¥webapps¥chart** になりました。
5. **CreateChart.java** ファイルを **tomcat¥webapps¥chart** ディレクトリに配置します。続いて、http://yourMachineName/EspressChart を正しい URL に置き換えて、getFileLocation()の URL を変更します。例えば、マシン名が Mach1 で、Web サーバーディレクトリの下に EspressChart をインストールしたとする場合は、getFileLocation()の URL として **http://Mach1/EspressChart** を指定します。
6. **CreateChart.java** をコンパイルします。**CreateChart.class** が同じディレクトリに存在するはずです
7. Tomcat サーバを起動し、EspressManager を起動します。
8. Web ブラウザを開き、**http://yourMachineName:8080/examples/jsp/chart/ExportChart.jsp** に移動します(yourMachineName を実際のマシン名に置き換えてください)
9. 必要なオプションを指定してから **submit** をクリックすると、チャートが返されます。

13.2.2.2 WebSphere

1. **EspressAPI.jar** と **ExportLib.jar** をノードの下にある従属クラスパスに追加します(通常はマシン名で呼び出されます)
2. ツールバーの **Wizard** 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。
   a. Web アプリケーションの名前として **Quadbase** を入力します。**Enable File Servlet** オプションをオフにして、**Server Servlet by classname** にチェックを入れ、**Next** をクリックします。
   b. ノードとして **Default Servlet Engine** を選択し(**Default Servlet Engine** 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
   c. **Web Application Web Path** を/**Quadbase** に変更し、**Next** をクリックします。
   d. **EspressAPI.jar** と **ExportLib.jar** を別々の行に **Classpath** として追加し、**Finish** をクリックします。
3. デフォルトサーバを再起動します。
4. Web サーバのドキュメントルート配下に **Quadbase** という名前のディレクトリを作成し、そこに**.jsp** ファイルと **java** ファイルを移動します。
5. **CreateChart.java** を次のように変更します。
   a. パッケージを削除(またはコメントアウト)します。
   b. 次のエクスポートコマンドを入力します。

```
chart.export(QbChart.JPEG, "<Web サーバドキュメントルートの絶対パス>/EspressChart/temp", 500, 400);
```

6. .jsp ファイルを変更して、**Chart.CreateChart** を **CreateChart** に置き換えます。
7. Web サーバのドキュメントルートの下に **EspressChart** というディレクトリを作成します
8. ブラウザを起動し、**http://<マシン名>/Quadbase/ExportChart.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チャートを取得します

13.3 チャートをファイルに保存した時と直接ブラウザに送信した時の比較

サーブレットと JSP を使用して、チャートをブラウザに戻すことができます。チャートは、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チャートは、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、チャ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、チャ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。

13.3.1 チャートをファイルに保存

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、チャ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。

```
QbChart chart = new QbChart (.....);
....
....
....
int format = QbChart.CHT;     //インタラクティブなチャートに設定されています。
chart.export(format, "chart", 500, 500);
```

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.chartviewer.Viewer.class" width=650 height=650
archive="http://<machineName>/EspressChart/lib/EspressViewer.jar">
<PARAM name="filename" value="chart.cht">
</applet>
```

Chart Viewer は、ここで使用され、上記の場合に EspressManager が起動しています。

静的イメージの場合、静的イメージの位置を示す単純な**<img src>**タグは、グラフを表示するのに十分です。

このメソッドを使用すると、グラフを返すページをサーブレット/JSP から分離し、事前にサーブレット/JSP の外でデザイン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、生成されたチャートを追跡し、必要に応じて再度表示したり、チャートを定期的に再生成させたりすることができます。

例については、**help/examples/servlets/DatabaseJPEG** ディレクトリと、**help/examples/jsp/Chart** ディレクトリのファイルを参照してください。

13.3.2 チャートをブラウザに直接送信

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エリア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 {

            //まず、レスポンスの”cintent type”ヘッダを設定します
            response.setContentType(”image/html”);
            //ここで、response はサーブレットへの応答です
            OutputStream toClient = res.getOutputStream();
            QbChart chart = new QbChart (.....);
            ....
            ....
            ....
            chart.export(QbChart.JPEG, toClient, 500, 500);

        }

    }
```

HTML 部分では、サーブレットが囲まれた**<img src>**タグは、チャート画像を表示するのに十分です。

```
<HTML><BODY>

        <img src=”http://machine_name:8080/servlet/OutputStreamServlet>

</BODY></HTML>
```

また、マップファイルを出力ストリームとして出力、生成された HTML ファイルに埋め込むこともできます。マップファイルを出力ストリームとして生成するには、**QbChart** クラスの以下のメソッドを使用します。

```
export(int format, OutpuStream image, OutputStream mapFile, String fileName, int
w, int h, boolean generateMapFile, int option);
```

インタラクティブなチャートの場合、チャートは文字列としてエクスポートされます。

```
QbChart chart = new QbChart (.....);
String buffer = chart.exportChartToString();
....
....
....
```

HTML 側では、チャートビューアを使用してインタラクティブなチャ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.chartviewer.Viewer¥" width=600
height=600 " +

"archive=¥"http://<machineName>/EspressEnterprise/EspressChart/lib
                    /EspressViewer.jar¥">");
toClient.println("<PARAM name=¥"ChartData¥" value=¥"" +
chart.exportChartToString() +
                    "¥">");
toClient.println("</applet>");
```

例については、**help/examples/servlets/StreamingJPEG** および **help/examples/servlets/StreamingChart** ディレクトリにあるファイルを参照してください。

JSP では、Java Bean オブジェクト(**myStreamChart**)を作成する必要があります。このオブジェクトは、チャートを作成し、それを String オブジェクトとして返します。JSP ページを以下のように変更する必要があります。

```
<jsp:useBean id="myStreamChart" scope="page"
class="streamingChart.CreateChart" />
<jsp:setProperty name="myStreamChart" property="*"/>
<applet code="quadbase.chartviewer.Viewer" width=600 height=600
archive="http://<machineName>/EspressChart/lib/EspressViewer.jar">
<PARAM name="ChartData" value="<%= myStreamChart.export()%>">
</applet>
```

ここで、myStreamChart Java Beam には、以下のコードが含まれています。

```
    public string export() throws Exception {

        QbChart chart = new QbChart(....);

        return chart.exportChartToString();

    }
```

例については、**help/examples/jsp/streamingChart** のファイルを参照してください。

静的なイメージをブラウザに直接送信するために、JSP とサーブレットの組み合わせが使用されます。

14 デプロイメント

14.1 導入

EspressChart は、Chart Designer、Chart Viewer、EspressManager、および Chart API のいくつかのコンポーネントで構成されています。Chart Designer は、GUI 環境でチャートを作成するために使用されます。Chart Viewer はチャート(.cht または.tpl 形式で保存された)を表示するために使用されたアプレットです。Chart API は、チャートをプログラムで作成する使用されます。最後に、EspressManager はユーザ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(その他の追加の jar とともに)含めなければなりません。

Chart Designer は EspressManager と組み合わせて使用する必要がありますが、Chart Viewer と Chart API が EspressManager に接続せずに動作するようにオプションが用意されています。

EspressManager に接続するには、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以下の 2 つのメソッドを使用します。

```
static void setServerAddress(java.lang.String address);
static void setServerPortNumber(int port);
```

以下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 EsspressManager に接続します。

EsspressManager 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 EsspressManager に接続し、デフォルトのポート番号(22071)をリッスンします。

EspressManager がサーブレットとして実行されている場合は、以下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);
```

例:

```
Qbchart.useServlet(true);
Qbchart.setServletRunner(”http:¥¥someMachine:somePortNumber”);
Qbchart.setServletContext(”EspressChart¥servlet”);
```

http:¥¥someMachine:somePortNumber¥EspressChart¥servlet で動作する EspressManager に接続します。

これらのメソッドは、**QbChart** および **QbchartDesigner** に存在します。

Chart Viewer の場合(アプレット内)、次のパラメータを渡して接続情報を EspressManager に設定できます。

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、以下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様々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。

14.2 EsspressManager でデプロイ

このセクションでは、EspressManager と連携して動作するチャートコンポーネントのデプロイについて説明します。インタラクションの詳細については、EsspressManager とのインタラ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チャ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。

14.2.1 チャートデザイナ

前述のように、Chart Designer は EspressManager と組み合わせて使用する必要があります。Chart 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、CLASSPATH に **ChartDesigner.jar** を追加し、.class ファイルの作業ディレクトリの下に画像(**EspressChart¥images**)と backgroundImages(**EspressChart¥backgroundImages**)のコピーを置く必要があります。

API からチャートデザイナを呼び出すときにディレクトリを作業ディレクトリに対してコピーする代わりに、**images¥**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。パラメータ **imagesPath** を持つ **QbchartDesigner** コンストラクタを使用します。**QbchartDesigner** コンストラクタまたは **imagesPath** パラメータの詳細については、EspressChart Chart API 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、QbchartDesigner で以下の 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、QbchartDesigner で以下の 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、QbChartDesigner コンストラクタを呼び出す前に行う必

要があります。

14.2.2 チャートビューア

チャートビューアの配備は、チャートビューアのと同様です。チャートビューアは、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。

アプレット環境にチャートビューア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。

チャートビューを使用するには、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、チャートビューア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 が起動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、以下のパラメータをチャートビューアアプレットに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、以下のパラメータをチャートビューアアプレットに渡します。

comm_protocol (servlet), comm_url (machine name¥IP address and port number), servlet_context (servlet context)

14.2.3 チャート API

チャート 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。サーブレットまたは JSP 環境でチャート API を使用して、サーバ側のチャートを生成することもできます。

チャート API をデプロイするには、**EspressAPI.jar** と **ExportLib.jar** が CLASSPATH(アプリケーションとサーブレット/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、**QbChart** 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 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、**QbChart** の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、**QbChart** コンストラクタを呼び出す前に言う必要があります。

14.3 EsspressManager なしでデプロイ

このセクションでは、EspressChart から独立して機能するコンポーネントのデプロイについて説明します。インタラクションの詳細については、EspressChart API の概要の章の EspressManager とのインタラクションを参照してください。

EspressManager とは無関係にチャートを生成すると、EspressManager とチャートまたはチャートコンポーネントのやりとりが削除されるため、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チャートソースへの参照(**help¥examples¥data¥sample.dat** など)が HTML ファイル(アプレットの場合)の場合の作業ディレクトリからの相対パスであること(アプレットの場合)またはクラスファイル(アプリケーション内の)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 に以下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、データファイルとチャートテンプレートは、コードで指示されているように作業ディレクトリに対する相対位置に移動できます。

14.3.1 チャートビューア

チャートビューア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、必要に応じて追加の jar を含める必要があります。

チャートビューアは 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">
```

グラフの場所や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¥sale.dat** で、HTML ファイルが **D:¥EspressChart¥TestApplet** にある場合は、チャートが正常に表示されるように、**help¥examples¥data¥sample.dat** は、**D:¥EspressChart¥TestApplet** 内に存在する必要があります(つまり、**D:¥EspressChart¥TestApplet¥help¥examples¥data¥sample.dat** が存在する必要があります)。

14.3.2 チャート API

チャート API をデプロイするには、**EspressAPI.jar** と **ExportLib.jar** が CLASSPATH(アプリケーションとサーブレット/JPS 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。

チャート API は、以下のコード行を追加することで EspressManager に接続せずに使用することもできます。

QbChart.setEspressManagerUsed(false);

このメソッドは、QbChart インスタンス化の前に呼び出さなければなりません。

チャートビューアと同様に、グラフの場所やデータに対する相対参照は、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的なものです。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:¥EspressChart¥TestApplication** にある場合は、チャートが正常に表示されるように help¥examples¥data¥sample にします。**D:¥EspressChart¥TestApplication** 内に存在する必要があります。
(つまり、**D:¥EspressChart¥TestApplicationhelp¥examples¥datasample.dat** が存在する必要があります)

14.4 Windows 環境以外でのデプロイ

EspressChart はチャートを生成するための純粋な Java ツールです。そのため、色、フォント、その他の 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(グラフが生成されている)ライブラリに依存しています。したがって、AWT 情報を提供することが得きる環境および(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とされる。

Windows 環境では、すでに存在するような環境を設定するために特別な作業をする必要はありません。GUI インターフェイスは既に実行されており、グラフィックスカードは既に存在します。

これは通常、Windows 以外の環境(Unix や Linux など)では当てはまりません。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(たとえば、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す。

14.4.1 Xvfb(X Virtual Frame Buffer)

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。

14.4.2 JVM 1.4+ in Headless Mode

EspressChart 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 ランタイム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、チャートを jpg として生成する chartExport というアプリケーションがあり場合、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、以下のコマンドを使用します。

```
java -Djava.awt.headless=true exportChart
```

14.5 プラットフォーム固有の問題

14.5.1 AS/400

14.5.1.1 NAWT 下での実行

NAWT を使用する AS/400 環境で EspressChart を使用してアプリケーションを実行するには、以下の実行をする必要があります。

同じアプリケーションが X に接続せずに実行されます。

- **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
```

14.5.1.2 Headless モードで実行

Headless モードを使用する AS/400 環境で EspressChart 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、コードに以下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);
```

14.5.2 Linux/Unix

14.5.2.1 X 下での実行

X を使って Linux/Unix 環境で EspressChart を使っ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。

```
java.awt.headless=true
```

- 任意の **QbChart** コンストラクタを呼び出す前に、コードに以下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);
```

15 ローカライズ

15.1 ローカライズ EspressChart

EspressChart 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。

15.1.1 ロケールの指定

EsspressChart 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。

15.1.1.1 ロケール固有の書式設定
EsspressChart では、日付と数値のデータなど、チャ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、チャ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(軸ラベルの書式設定を参照)。

グラフのロケールとタイムゾーンは、API を使用して実行時に設定できます。これは、日付、時刻、およびデータの書式設定にのみ影響します。

15.1.1.2 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、QbChart のメソッドを使用する(チャート全体に適用)か、LicaleDataTimeFormat および LicaleNumericFormat 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。

```
chart.setLocale(Locale.UK);
chart.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、フォーマットはチャート全体に適用されます。または、以下のように手億艇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);
int columnIndex = 2; //column 2 is the "Sales" column
chart.gethDataPoints().setLabelFormat(columnIndex, currencyFormat);
```

フルソースコード

エクスポートされた結果

15.1.2 言語とエンコーディング

EspressChart には、言語の違いにより制限を取り除く一連の機能が含まれています。シンプルなインターフェイスを利用することで、EspressChart 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り、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、その言語でチャートを保存してエクスポートすることもできます。

15.1.2.1 EspressChart 言語翻訳

インターナショナライゼーションは、**.xml** ファイル、**Language.xml**(**<EspressChart インストールディレクトリ>**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 GUI が提供されています。

Language.xml ファイルの構造は、以下の通りです。

```
<Product name=”CHARTDESIGNER” dir=”quadbase/chartdesigner/designer”>

        <File name=”ChartMenubar.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) > File > Text 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。EsspressChart は、Java Swing UI および、または EspressChart 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、複数の製品のもとに同じコード(たとえば **File**)が表示される場合があります。ただし、コードの翻訳は全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** ファイルのコピーを作成し、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。ユーザインターフェイスを利用するには、EsspressChart のルートディレクトリに移動し、以下のコマンドを入力します。

```
Java -classpath ”./lib/EspressAPI.jar;.;” quadbase.internationalization.TranslateWizard -file:<fileName> -enc:<encoding>
```

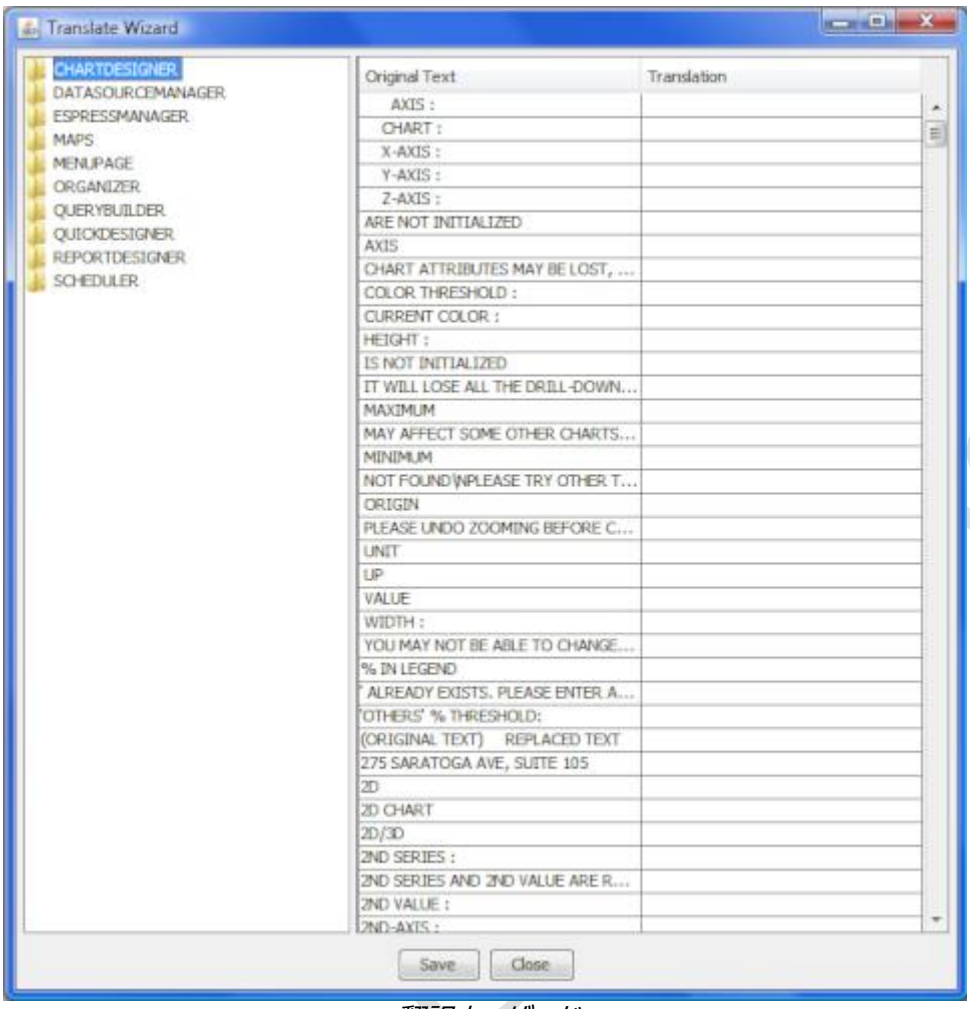
ここで、**<filename>**は変換を格納する xml ファイルを表し、**<encording>**は変換のエンコードを表します。正しい結果を得るには、適切なエンコーディングを指定する必要があります。ファイル名とエンコーディングが指定されていない場合、デフォルトのファイル名(**Language.xml**)とデフォルトのエンコーディング(**Cp1252**)が使用されます。

上記のコマンドの例を以下に示します。

```
java -classpath ”./lib/EspressAPI.jar;.;” quadbase.internationalization.TranslateWizard -file:Chinese.xml -enc:Big5
```

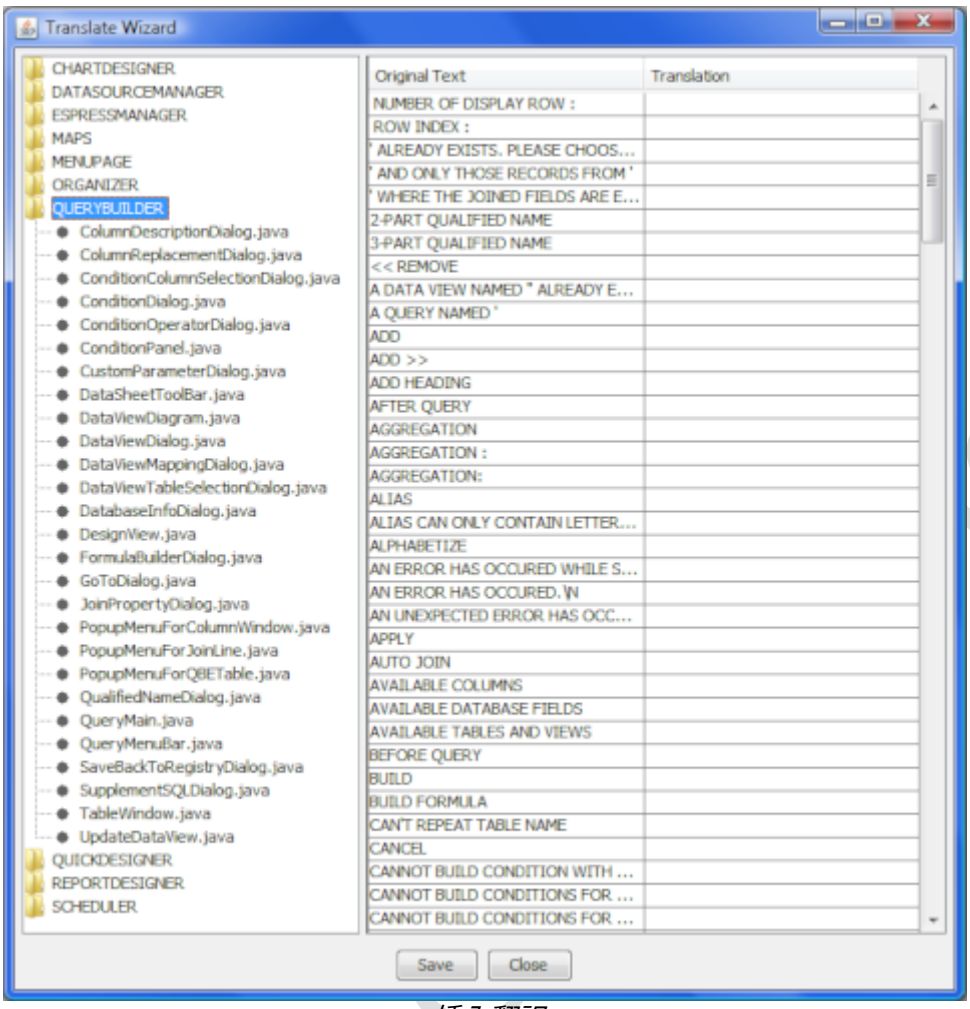
Big5 は、**Chinese.xml** ファイルを保存するために使用されるエンコーディングです。

変換ウィザードが開始されると、以下のように表示されます。



翻訳ウィザード

適切な製品を選択すると、翻訳可能なさまざまなコードが表示されます。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、拡張 QUERYBUILDER 製品が表示されます。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。



挿入翻訳

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、コードのすべてのインスタンスに対して表示されます。

EsspressChart よりも多くのオプションが Translate Wizard に存在することに注意してください。たとえば、REPORTDESIGNER 製品と ORGANIZER 製品は、EsspressChart には存在しないため、これらの製品で行われた変更は表示されません。

EspressChart で **Language.xml**(または変更したコピー)を使用するには、EsspressManager、ChartDesigner および、または 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。

15.1.2.1.1 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ログラムを使用するには、コンソールウィンドウから EsspressChart ディレクトリに移動し、以下のコマンドを使用します。

```
java -classpath ".;.¥lib¥EspressAPI.jar" quadbase.internationalization.UpgradeLanguageXMLFile -from:oldfile -to:newfile -enc:encoding
```

非ウィンド環境では、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 EspressChart 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 EspressChart インストールに含まれる **Language.xml** を参照し、**enc** は、翻訳で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 Wizard を使用してファイルを開くことができます。

15.1.2.2 外国語の表示

外事は、Designer と Viewer で簡単に表示できます。チャートに外国文字を表示するには、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、チャートでは、外部文字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 フォントで外部文字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、EspressChart(ダイアログ、セリフ、ものスペースなど)のデフォルトのフォントが外字で表示されます。

15.1.2.3 外国人文字の入力

ChartDesigner などの EspressChart インターフェイスに外部の文字を入力するには、システム設定を以下のように、変更する必要があります。

- システムの **default locale** 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 **input locale** 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、**Control Panel** の **Regional Options** 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されている手順に従って、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。

これらの設定が適用されると、チャートのラベル、クエリ、数式、およびパラメータに異種文字を入力できます。

15.1.2.4 XML エンコーディング

デフォルトでは、EspressChart は 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 チャートテンプレート、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 表現が含まれています。ほとんどのユーザにとって、UTF-8 は全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ディングを変更する必要はありません。

このエンコーディングは、ChartDesigner 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。ChartDesigner の場合は **ChartDesigner.bat/sh** ファイルを変更するか、ChartDesigner 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、**<EsspressChartInstallDir>¥ChartDesigner.bat** または**.sh** ファイルに以下の引数を追加します。**-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。

ChartDesigner をアプレット経由で実行している場合、アプレットページに **<PARAMNAME = "xml_Encording" 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。サーバ設定の詳細は、EspressManagerの起動を参照してください。

16 付録 A. パラメータサーバ

16.1 パラメータサーバの作成

Chart Viewer は、パラメータサーバと対話することによってプッシュテクノロジをサポートしています。プログラマは、**Quadbase.chartviewer.ParamServer** クラスを使用してパラメータサーバを作成し、チャートビューアがグラフ描画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 構文は以下のいずれかです。

```
<applet code = ”quadbase.chartviewer.Viewer.class” width=640 height=480>
<PARAM name=”filename” value=”example.cht”>
<PARAM name=”ParameterServer” value=”machine:portno”>
</applet>

または

<applet code = ”quadbase.chartviewer.Viewer.class” width=640 height=480>
<PARAM name=”filename” value=”example.tpl”>
<PARAM name=”ParameterServer” value=”:portno”>
</applet>
```

ブラウザが Chart Viewer クラスと Web サーバからチャートデータをロードすると、Chart Viewer は指定されたとおりに、パラメータサーバに接続しようとします。最初のケースでは、Chart Viewer は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。2 番目のケースでは、デフォルトマシンである、Web サーバマシンが使用されています。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。パラメータサーバは、Chart Viewer が保持するレコードを更新、および操作するために Chart 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.chartviewer.Paramserver** を使用して記述できます。

接続を開くと、サーバは Chart Viewer にデータを提供します。

サーバのコンストラクタは、以下のとおりです。

```
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、Chart Viewer でレコードを操作する 3 つの基本的なメソッドがあります。

```
int addRecord(Object record[]);
int deleteRecord(int recordNo);
int updateRecord(Object record[], int recordNo);
```

一連のレコードの更新後、最終コール。

```
void repaint();
```

以下の Java プログラムは、新しいチャートを生成するためにパラメータサーバクラスを使用する列を示しています。ParamServer 用の Chart 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.chartviewer.ParamServer;

// Sample Program to update the data in EspressChart Viewer
// using Parameter Server
public class pserver extends Thread {

    protected int port = 1997; // port no for chart 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); } }
        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 chart viewer
        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 chart 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) {}
        } }

}
```

Chart Viewer は、以下の形式の HTML ファイルを読み込んでいます。

```
<html>
<title>Sample Chart</title>
<applet code = "quadbase.chartviewer.Viewer.class" width=600 height=450>
<PARAM name="filename" value="col2d.cht">
<PARAM name="ParameterServer" value=":1997">
</applet>
</html>
```

ここで、Chart Viewer が読み込む Java クラスは、グラフファイル col2d.cht の名前であり、以下の行は Chart 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 Chart Viewer が表示するチャートを更新することができます。

16.1.1 変数

UPDATE_RECORD
public final static int UPDATE_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

REPAINT
public final static int REPAINT

OK
public final static int OK

ERROR
public final static int ERROR

16.1.2 コンストラクタ

パラメータサーバコンストラクタ

public ParamServer(DataInputStream in, DataOutputStream out) throws IOException

16.1.3 メソッド

updateRecord
public int updateRecord(String rec[], int recordNo) throws IOException

レコードを更新します。

パラメータ
rec:レコードが文字列配列として渡されます。クラス quadbase.chartAPI.DbData の文字列配列のフォーマットを参照
recordNo:更新されるレコード番号

Returns:成功すればOK、そのようなレコード番号やレコードタイプの不一致がなければERRORを戻す。

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:レコードはオブジェクト配列として渡す
recordNo:更新されるレコード番号

Returns:成功すれば、OK、レコードタイプの不一致なら ERROR を戻します。

addRecord
public int addRecord(Object rec[]) throws IOException

レコードを追加します。

パラメータ rec:レコードは文字列配列として渡す

Returns:成功すれば OK、レコードタイプの不一致なら ERROR を戻す

addRecord
public int addRecord(Object rec[]) throws IOException

レコードを追加します。

パラメータ rec:レコードはオブジェクト配列として渡される

Returns:成功すれば OK、レコードタイプの不一致なら ERROR を戻す

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
Returns:成功すれば OK、レコードタイプの不一致なら ERROR を戻す

repaint
public void repaint() throws IOException

グラフビューアにグラフの再描画を通知します。

checkRecord
public Boolean checkRecord(Object rec[])

指定されたレコードタイプが、Chart 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する補助機能です。

パラメータ rec:チェックされる入力レコード
Returns:一致するレコードがある場合は true、そうでない場合は false

getRecordNo
public int getRecordNo() throws IOException

使用レコード数を取得します。

Returns:記録の数

getDataType
public int[] getDataType() throws IOException

Returns: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い。

getRecordSize
public int getRecordSize()

レコードのフィールド数を取得します。

Returns:レコードのフィールド数

getRecord
public Object[] getRecord(int recordNo) throws IOException

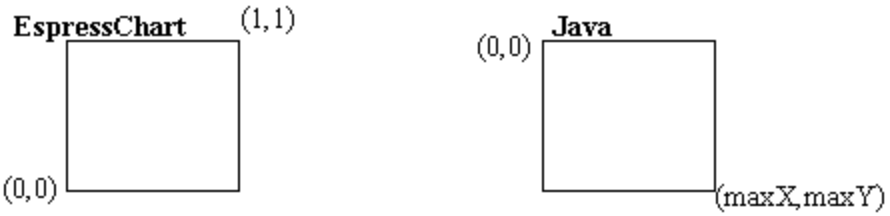
レコードを取得します。

パラメータ recordNo:取得するレコード番号
Returns:レコードオブジェクト配列

convertRecord
public Object[] convertRecord(String rec[])

文字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。
Returns:レコードのオブジェクト配列、データ型の不一致の場合は null

17 付録 B.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

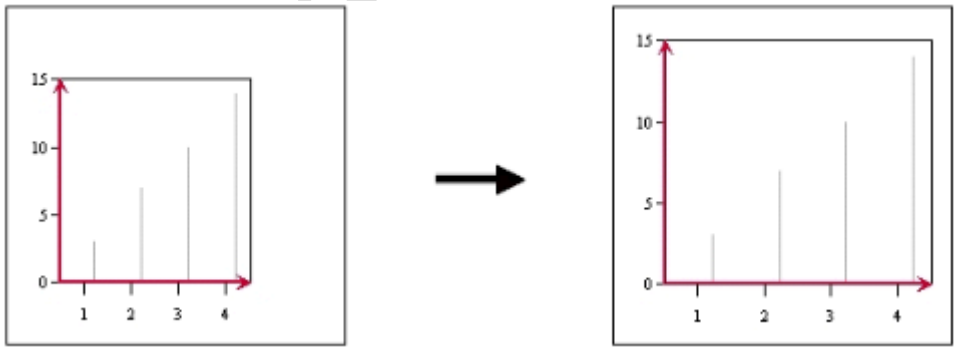
EspressChart は、コンポーネントレイアウトを相対座標で記述します。一方、Java はピクセル単位でレイアウト形式を使用します。下位互換性の問題により、EspressChart のレイアウト形式は変更されません。相対的な位置と絶対的な位置に関連する問題は、グラフィカルユーザのインターフェイスがありため、Chart Designer ではあまり関係がありません。ただし、レイアウトの書式が Chart API にどのようになっているかを知ることは有益です。原点(0.0)はキャンバスの左下隅にあり、最大(1.1)は右上隅にあります。

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。

17.1 チャートプロットエリアの変更



グラフプロットエリアの拡大

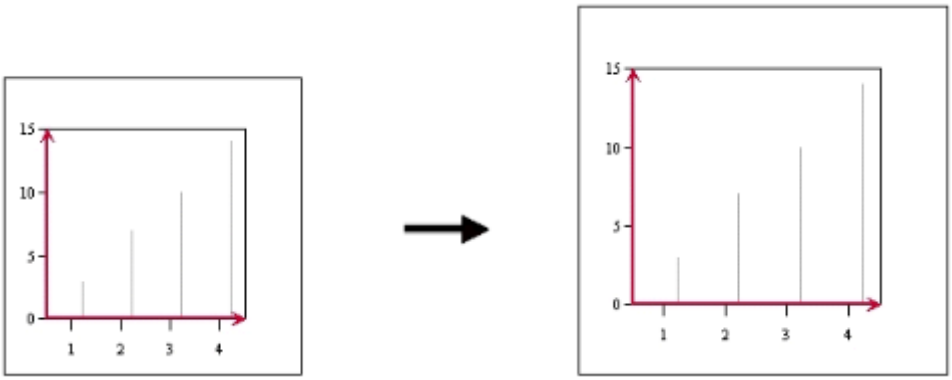
グラフプロットエリアのデフォルトの相対的なサイズは、x ディメンションで 0.6、y ディメンションで 0.6 です。API を使用してこれらの比率を調整する場合は、**setRelativeHeight()** および **setRelativeWidth()**メソッドを使用できます。
チャートのデフォルトの相対サイズは、0.6,0.6 です。

```
IPlot chartPlot = chart.gethChartPlot();
chartPlot.setRelativeHeight( (float) 0.75);
chartPlot.setRelativeWidth( (float) 0.8);
```

チャートデザイナでチャートプロットを調整するには、マウスカーソルがハート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、グラフプロットエリアが拡大され、左にドラッグすると、グラフプロットエリアが縮小されます。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。

17.2 キャンバスエリアの変更



キャンバスエリアの拡大

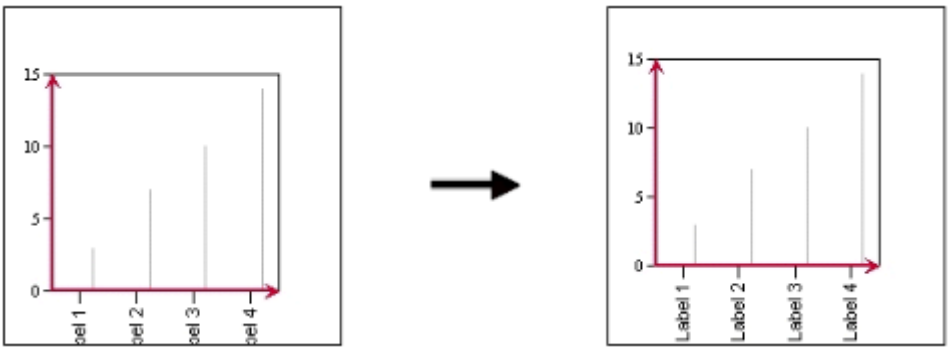
この例では、グラフのサイズをキャンバスに対して一定に保ちながら、キャンバスのサイズを調整する方法を示します。

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エリアを変更するには、**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。小さいキャンバスエリアが必要な場合は、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。

17.3 ページ区切り

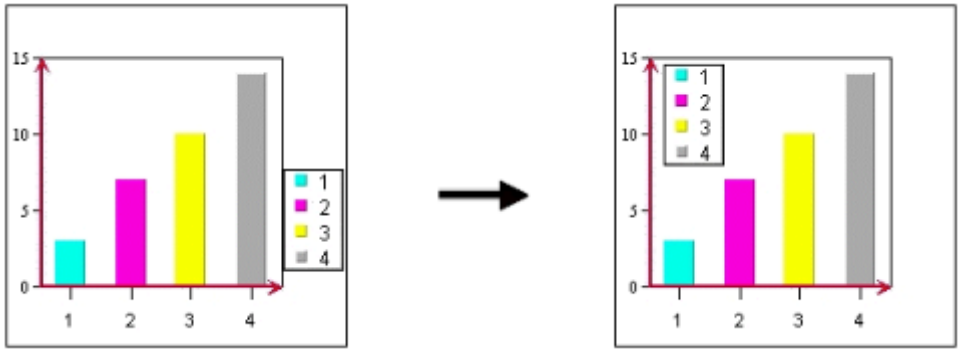


グラフ要素をキャンバスにフィットさせる

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。

```
// EspressChart はチャート全体に収まるようグラフの位置の変更、
//グラフのサイズを縮小します。
//デフォルト値は FALSE
chart.gethCanvas().setFitOnCanvas(true);
```

17.4 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。

17.5 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。EspressChart 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が、この場合は特定の選択されたデータポイントにのみラベルを付けることができます。

既存の機能を使用して、次のように実施できます。まず、任意のデータポイントの持つ水平線を描画します。次に、水平線に注釈テキストを追加し、水平線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(両方のシリーズの最大値)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)。

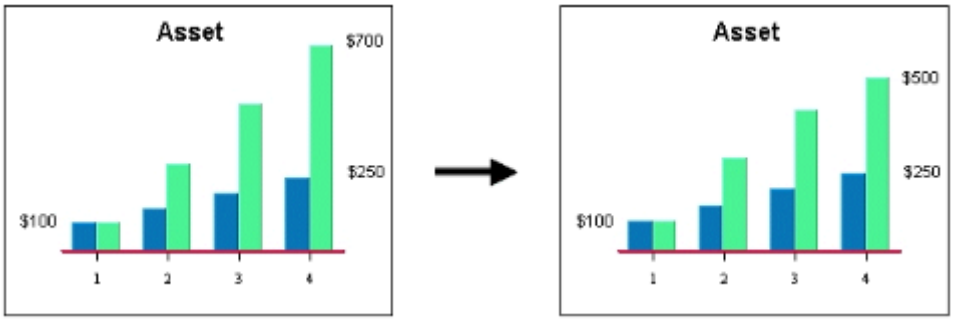
最後のデータポイントの 1 つが＄700 から＄500 に変更されると、ラベルの位置は自動的に更新されます。

日	値 1	値 2
1	100	100
2	150	300
3	200	500
4	250	700

オリジナルデータ

日	値 1	値 2
1	100	100
2	150	300
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

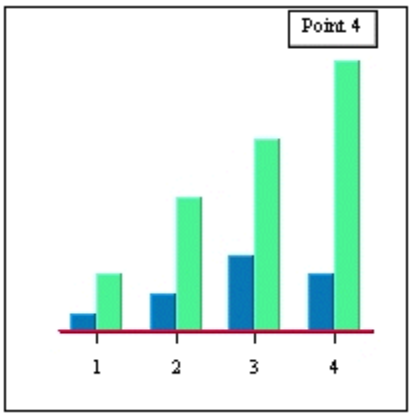
```
//example は、データベースクエリの最後の行の値ポイントの 1 つを検索します。
//最後の行の値ポイントに特別な意味があるわけではありません。
//プログラムで任意の値を指定できます。
//例:最大値、最小値、平均など

//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);
//初期値を取得する
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);

//最後の行の第 2 シリーズの最後の値
int value1 = Integer.parseInt(lastRow.getObject(3).toString());
//最後の行の第 1 シリーズの最後の値
int value2 = Integer.parseInt(lastButOneRow.getObject(3).toString());
```

```
String labelInit = "$" + valueInit;
String label1 = "$" + value1;
String label2 = "$" + value2;

//初期値の水平線を追加する
IDataLineSet hDataLines = chart.gethDataLines();
IHorzVertLine hvInit = hDataLines.newHorzVertLine(
IHorzVertLine.HORIZONTAL_LINE, labelInit);

hvInit.setLineValue(valueInit);    //値 0 の水平線
hvInit.setLineStyle( ((255*256) + 255) *256); //水平線を非表示にする
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 番目のシリーズ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);

//グラフプロットエリアの左側に、annolnit を追加する($100)
hvInit.addAnnotation(annoInit);
annoInit.setRelativePosition(new                    Point_2D(                    -
(float)chart.gethChartPlot().getRelativeWidth(), 0f));

//アノテーション 1 を行に追加し、表示する
hv1.addAnnotation(anno1);

//アノテーション 2 をグラフに追加し、表示する
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 などを参照します。唯一の例

外は、スキャッタチャートとバブルチャートです。

右側に行を挿入した後、行のスタイルを設定して行を非表示にすることができます。アノテーションが追加されます。



*チャートに **Point4** を追加する*

```
//4 番目のポイントの上に Point4 ラベルを追加
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);

//水平線に注釈を追加する
IAnnotationSet hAnnotation = chart.gethAnnotations();
IAnnotation annoTr1 = hAnnotation.newAnnotation("Point 4", tr1);

//annoTr1 をその行に追加すると、チャートに表示される
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。

18 更新履歴

版	修正日	修正者	内容
1.0	2017/11/24(金)	Y.A	初版
1.1	2024/02/09(金)	R.T	6.12 追加