[EspressReport 7.0]
クイックスタートガイド

株式会社クライム

Climb Inc.
Growing to Meet Your Needs

作成日: 2018/03/01(木)
更新日: 2018/03/01(木)
バージョン: 1.0

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、[EspressReport]のインストール手順について記載しております。

1.2 対象バージョン

本ドキュメントは、以下の製品のバージョンに対応しております。

- [EspressReport] Ver. 7.0

2 クイックスタート

2.1 概要

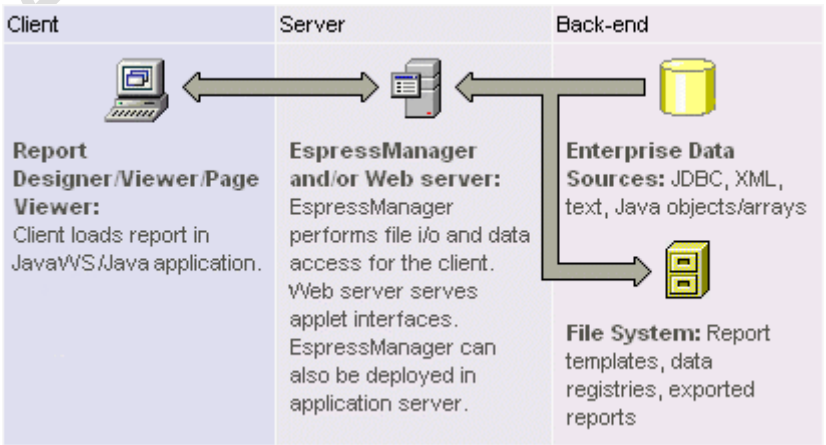
EspressReport は、インターネットやイントラネットアプリケーションで情報豊富なレポートを簡単に設計して展開できる強力な Java レポートツールです。EspressReport は事実上すべてのデータソースに接続し、DHTML スタイルシート、印刷品質の PDF、Microsoft Excel、またはリッチテキストの情報を簡単にフォーマットしてレンダリングできます。またテキスト、CSV、および XML 形式にデータを抽出することもできます。強力な Java API により、アプリケーション、サーブレット、JSP、および JavaWS にレポート機能を簡単に組み込むことができます。

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

2.1.1 EspressReport アーキテクチャ

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

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



設計時の EspressReport アーキテクチャ

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

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

インストールの詳細については、インストールをご参照ください。

2.2 主な特徴

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

Pure Java Architecture

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

フル機能の Report Designer

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

任意のソースからデータを描画

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

強力なクエリツール

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

広範なデータの可視化

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

柔軟な展開モデル

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

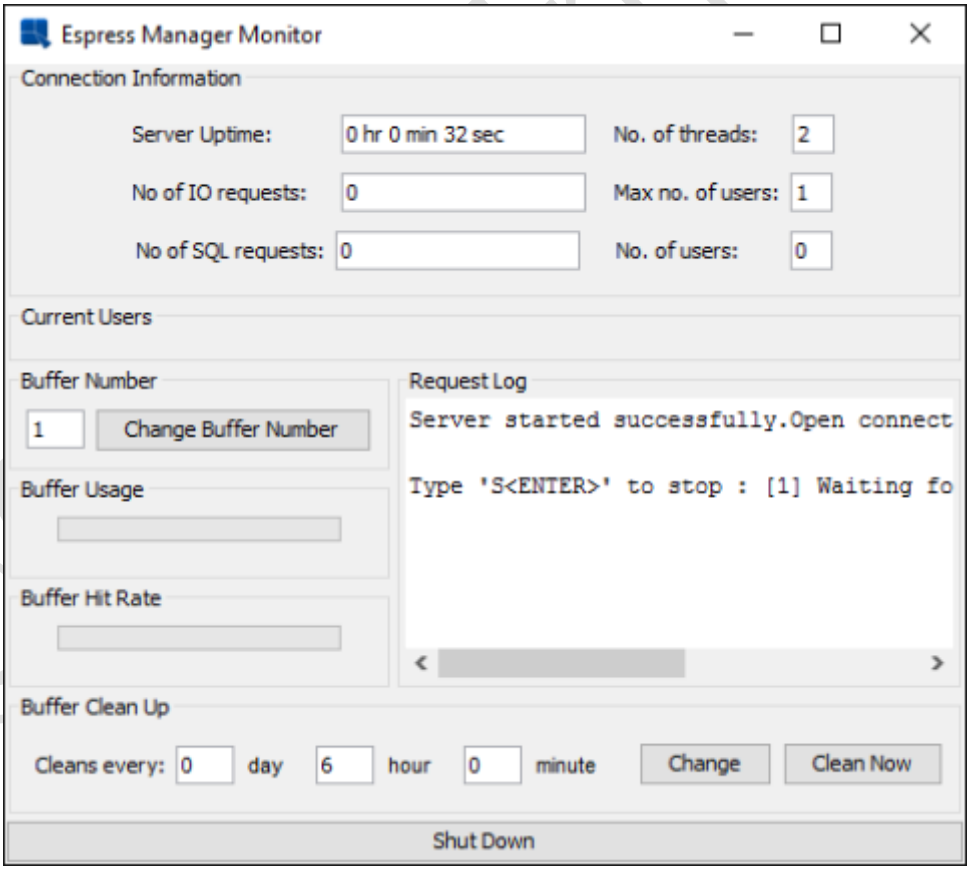
2.3 Designer のクイックスタート

このセクションでは、Report Designer の基本機能の一部を説明するための一連の短いチュートリアルが含まれています。

2.3.1 Report Designer の起動

この例では、Report Designer がインストールされているマシン上でローカルに Report Designer を実行することを前提としています。

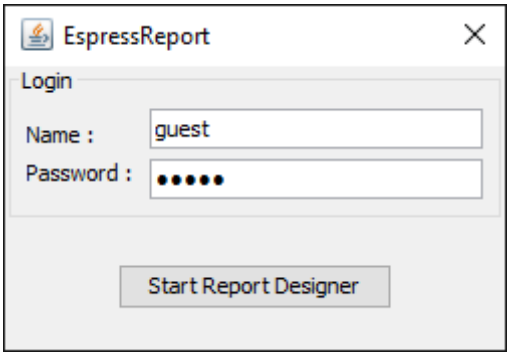
Report Designer を起動する前に、EspressManager が実行されている必要があります。EspressManager は、Report Designer のデータアクセスとファイル I/O を管理するバックエンドコンポーネントです（これにより、ローカルとリモートの両方で実行できます）。EspressManager を起動するには、インストールのルートディレクトリ（Unix インストールの場合は **EspressManager.sh**）で **EspressManager.bat** ファイルを実行します。デフォルトでは、EspressManager の起動時にモニタが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



EspressManager モニタ

EspressManager を実行すると、インストールのルートディレクトリにある **ReportDesigner.bat** ファイル(Unix インストールの場合は **ReportDesigner.sh**)を実行して、Report Designer を起動できます。デフォルトのユーザとしてログインするには、パスワードなしでユーザ名に **guest** と入力して使用しま

す(ユーザの設定の詳細は、Designer ガイドの設定を参照してください)。

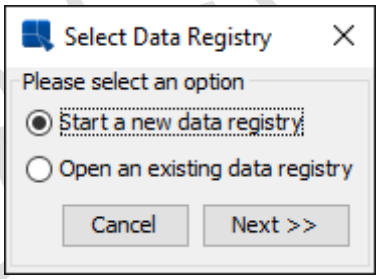


Designer ログインウィンドウ

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

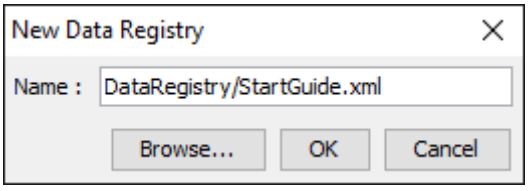
2.3.2 データソースを設定する

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

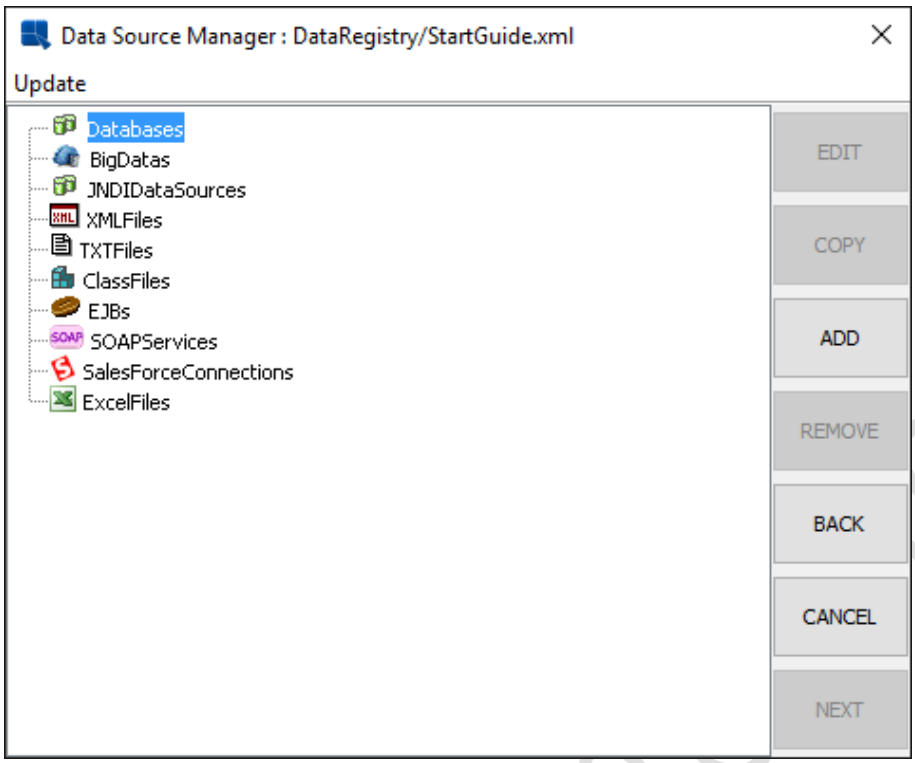


Select Data Registry ダイアログ

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



New Data Registry ダイアログ



データレジストリウィンドウ

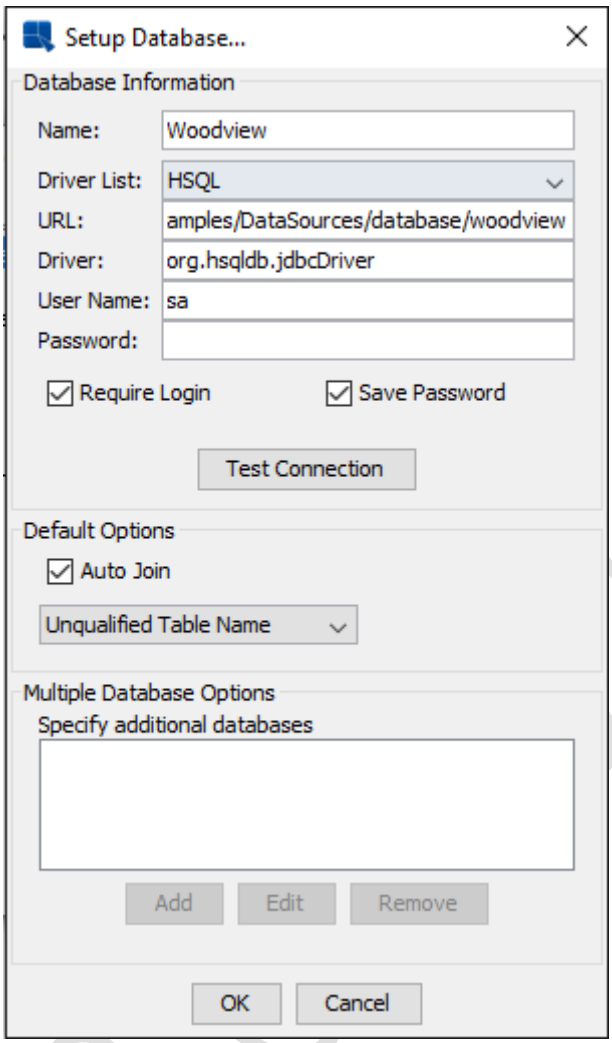
2.3.2.1 データベース接続のセットアップ

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

2.3.2.1.1 JDBC 接続のセットアップ

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

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



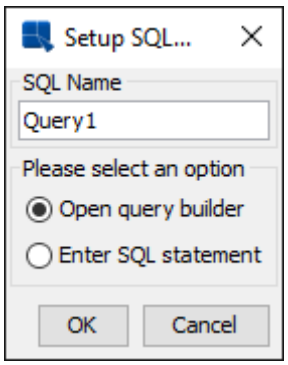
Add Database ダイアログ

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

2.3.2.2 クエリを作成する

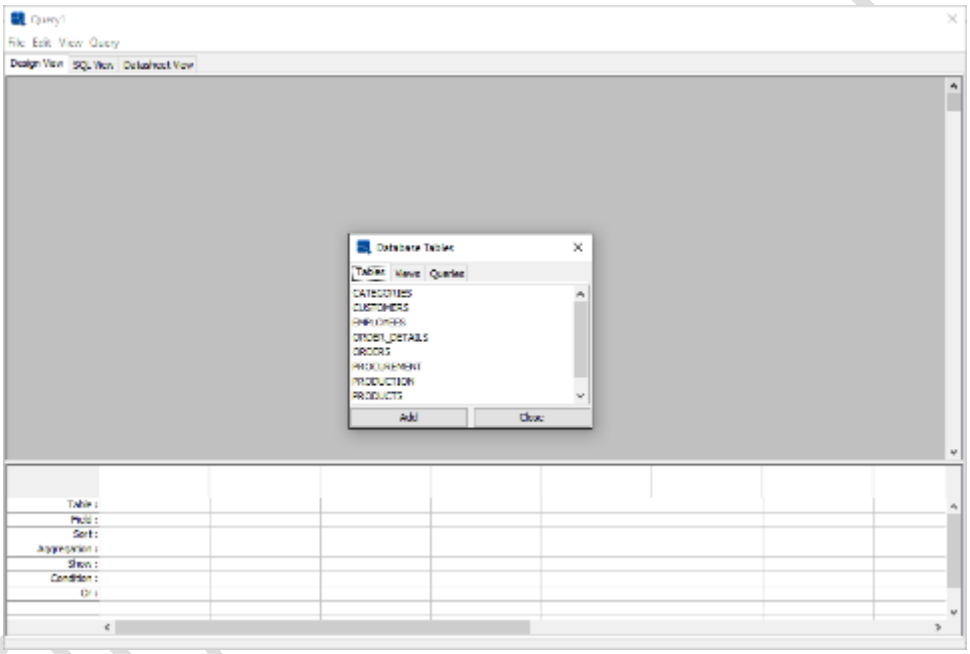
EspressReport は、データベースに照会してレポートデータを取り出すためのさまざまなインタフェースを提供します。SQL 文を実行したり、クエリビルダを使用したり、データビューを使用してエンドユーザをデータベース構造から隔離するクエリインタフェースを作成することができます。この例では、クエリビルダを使用してクエリを作成します。

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



Query Name ダイアログ

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



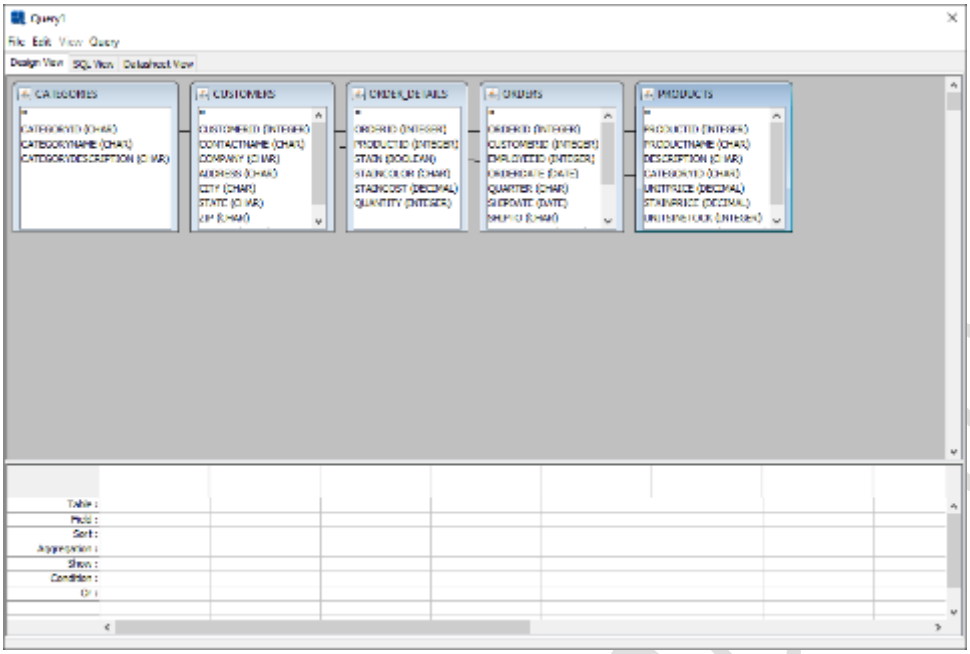
Query Builder ダイアログ

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

CATEGORIES
CUSTOMERS
ORDER_DETAILS
ORDERS
PRODUCTS

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

合線が表示されます。



テーブルを持つ Query Builder

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

ORDERID (INTEGER) from ORDERS

COMPANY (CHAR) from CUSTOMERS

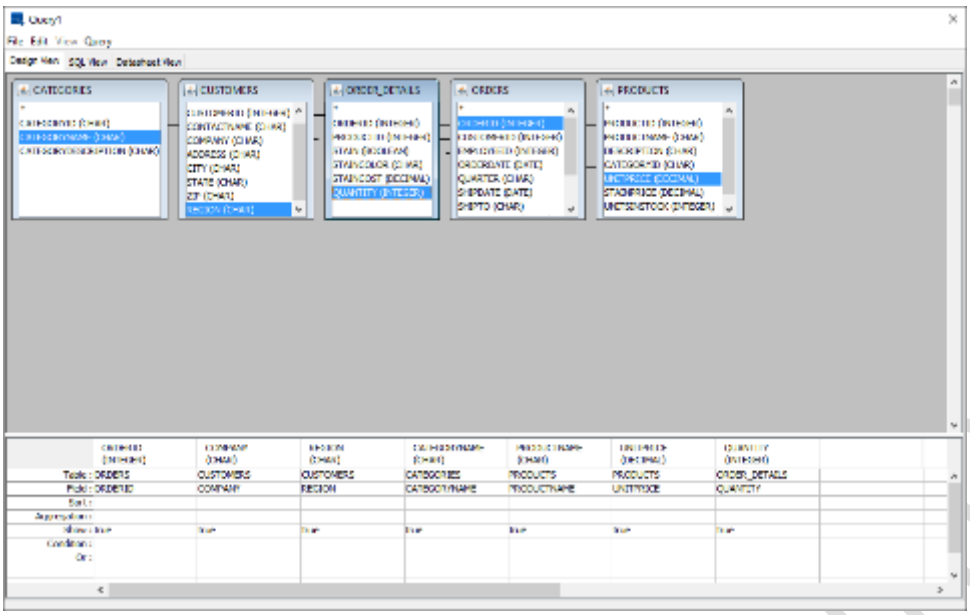
REGION (CHAR) from CUSTOMERS

CATEGORYNAME (CHAR) from CATEGORIES

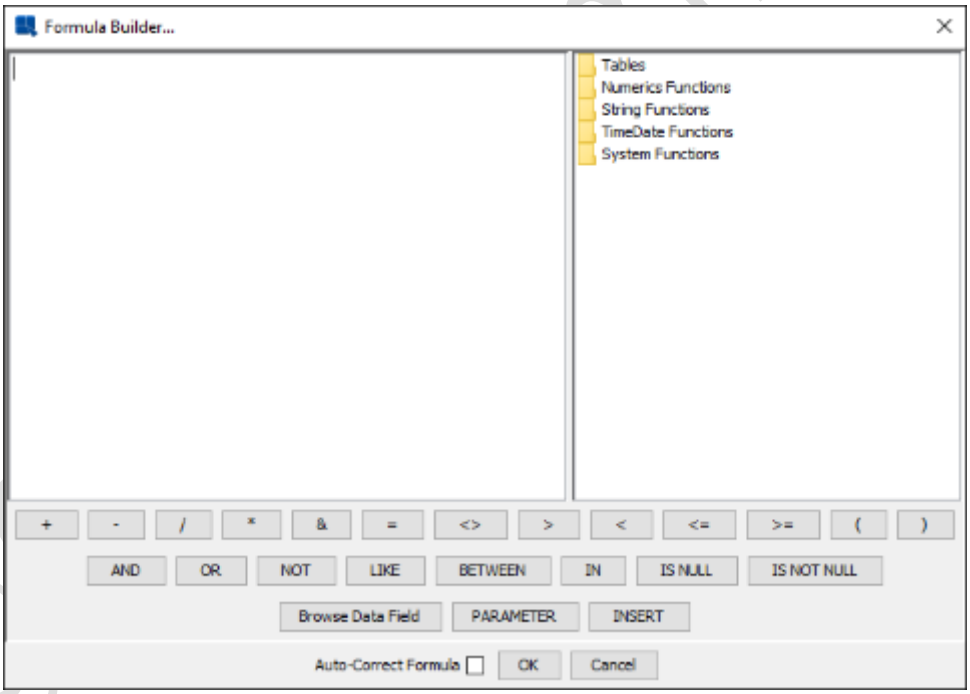
PRODUCTNAME (CHAR) from PRODUCTS

UNITPRICE (DECIMAL) from PRODUCTS

QUANTITY (INTEGER) from ORDER_DETAILS



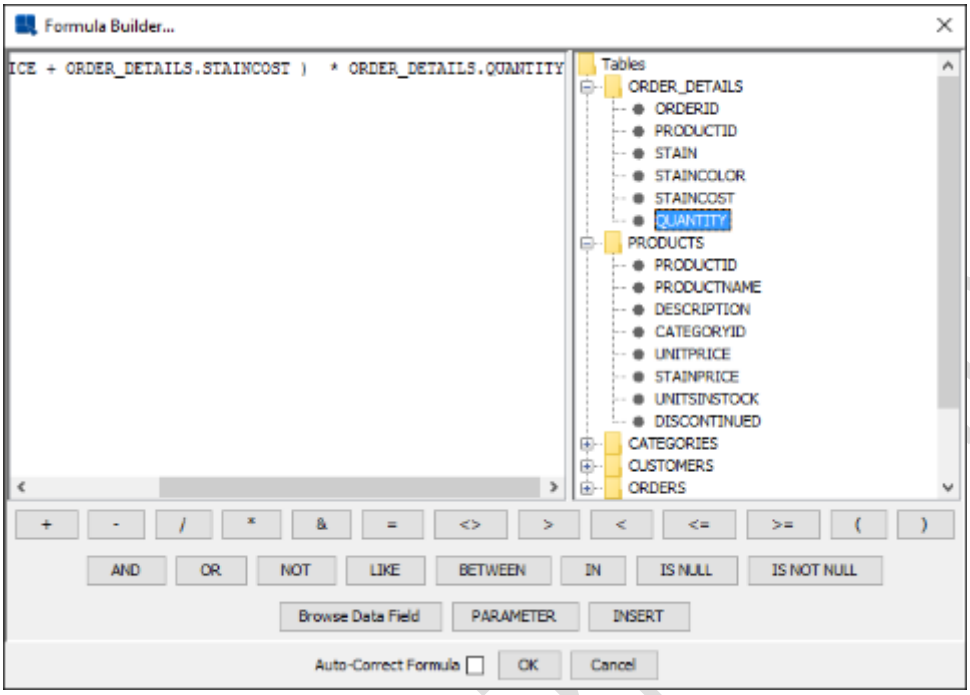
空白にする必要がある 8 番目の列で、**Field** フィールドを右クリックし、ポップアップメニューから **Build** を選択します。計算された列を作成できるようにする、数式ビルダインタフェースが開きます。



数式ビルダウィンドウ

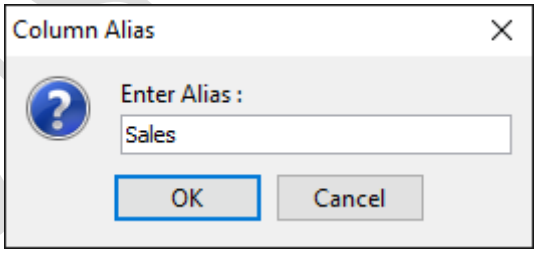
まず、**left parenthesis** ボタンをクリックして列を作成します。次に、**Tables** フォルダをダブルクリックして、クエリ用に選択した各テーブルに 5 つの node が展開します。テーブルフォルダを開くと、そのテーブルのすべての列フィールドが一覧表示されます。**PRODUCTS** フォルダを開き、**UNIRPEICE** を選択して **INSERT** ボタンをクリックします。次に、追加(+)ボタンをクリックします。次に、**ORDER_DETAILS** 表から **STAINCOST** を挿入します。次に、**right parenthesis** ボ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

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



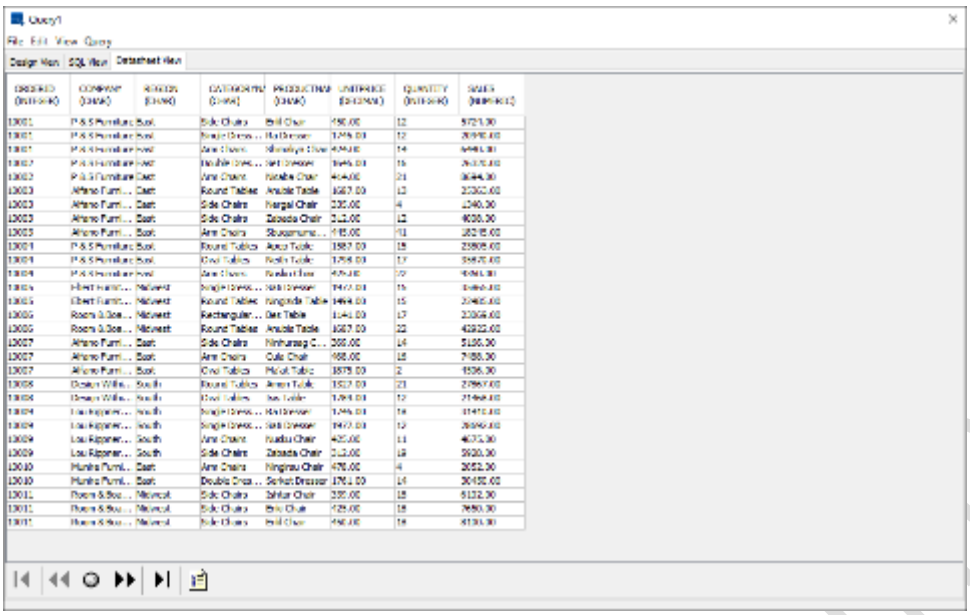
数式を含む数式ビルダウィンドウ

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



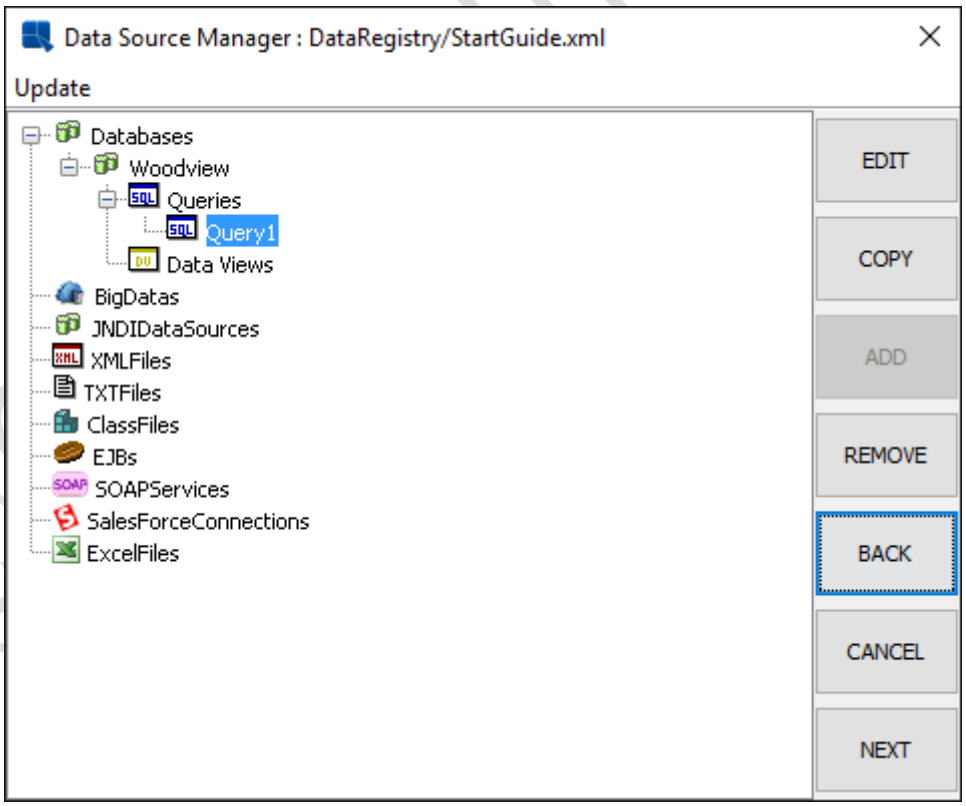
Column Alias ダイアログ

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



Query Builder のデータシートビュー

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



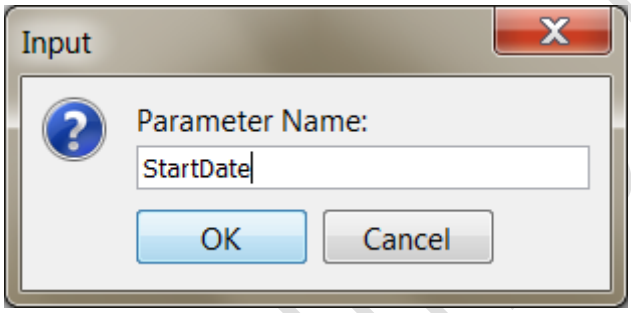
クエリ付きデータソースマネージャ

2.3.2.2.1 クエリパラメータの追加

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

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

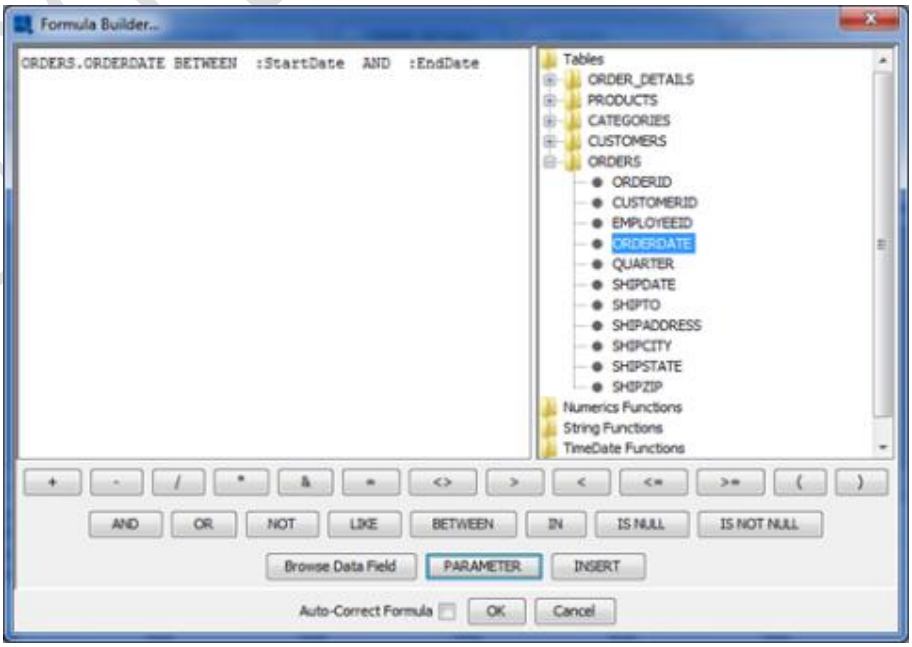
数式ビルダ内で、**Tables** フォルダをダブルクリックして展開します。**ORDERS** ノードを展開し、**ORDERDATE** フィールドをダブルクリックします。次に、**BETWEEN** ボタンをクリックし、**PARAMETER** ボタンをクリックします。これにより、クエリパラメータの名前を指定するダイアログが表示されます。



Parameter Name ダイアログ

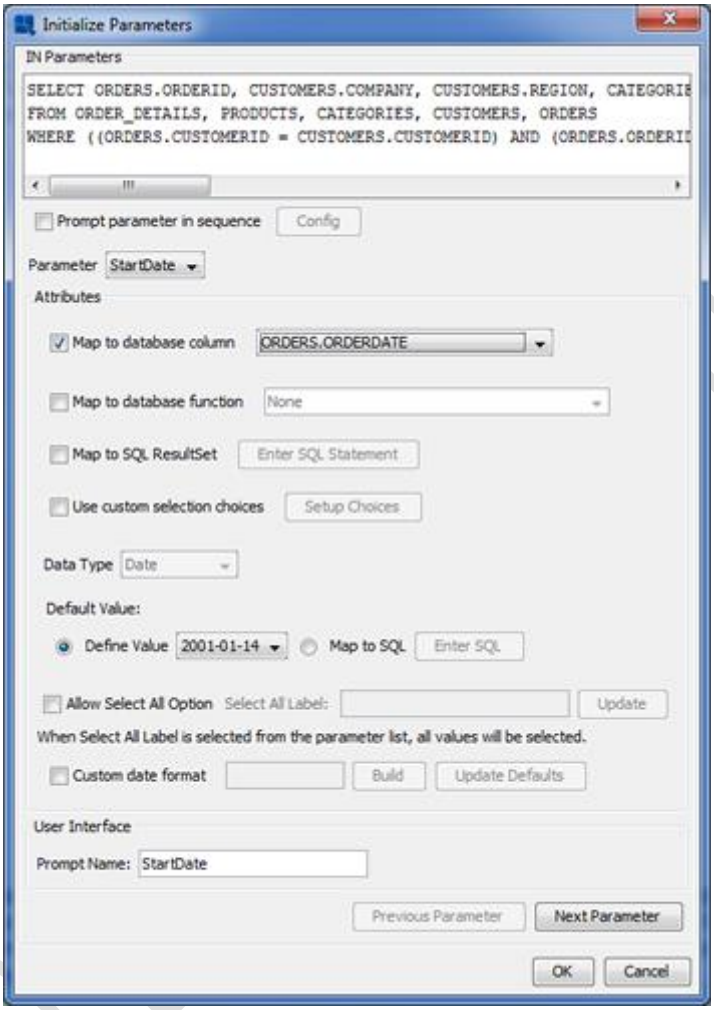
パラメータ名に **StartDate** と入力し、**OK** ボタンをクリックします。パラメータがクエリに追加されます。次に、**AND** ボタンを押してから、**PARAMETER** ボタンをもう一度クリックします。2 番目のパラメータ名として **EndDate** を入力します。完成した状態は、以下のようになります。

ORDERS.ORDERDATE BETWEEN :StartDate AND :EndDate



条件付き数式ビルダ

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



Initialize Parameter ダイアログ

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

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

Parameter Selection ダイアログ

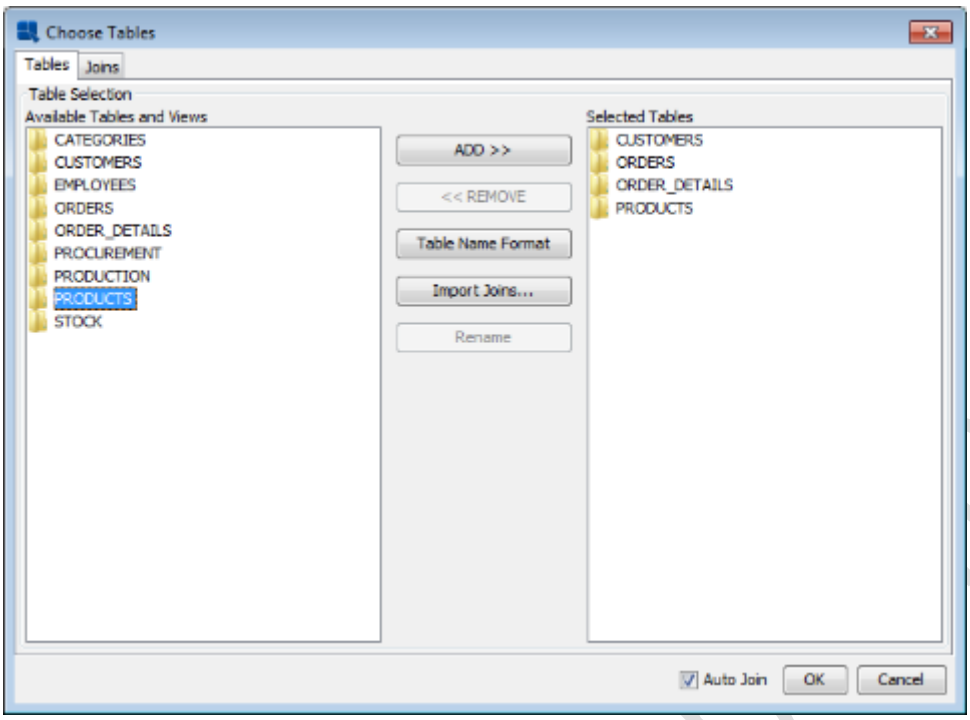
必要な開始日と終了日を選択し、**OK** ボタンをクリックします。フィルタされた結果がデータシートウィンドウに表示されます。ここで、File メニューの **Done** をクリックして、クエリに加えた変更を保存します。

2.3.2.3 データビューを作成する

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

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

CUSTOMERS
ORDERS
ORDER_DETAILS
PRODUCTS



Data View Table ダイアログ

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

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

CONTACTNAME
COMPANY
ADDRESS
CITY
STATE
ZIP

以下のように、他のテーブルのフィールドを追加します。

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

SHIPSTATE
SHIPZIP

ORDER DETAILS:
ORDERID
STAIN
STAINCOLOR
QUANTITY

PRODUCTS:
PRODUCTNAME
UNITPRICE
STAINPRICE

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

CONTACTNAME
COMPANY
ADDRESS
CITY
STATE
ZIP

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

SHIPDATE
SHIPTO
SHIPADDRESS
SHIPCITY
SHIPSTATE
SHIPZIP

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

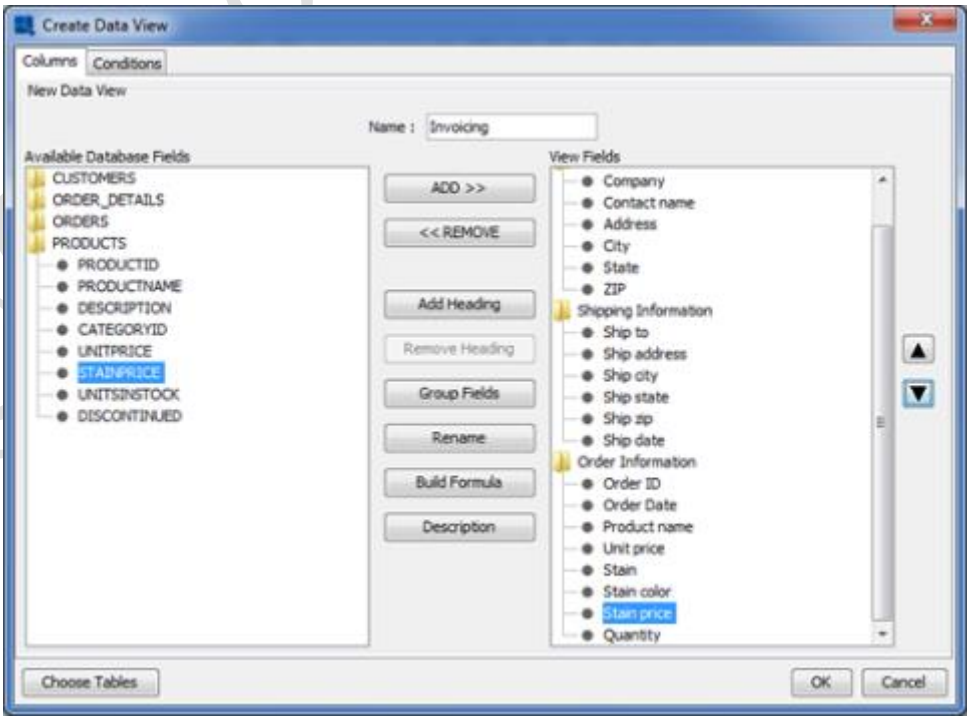
ORDERDATE
ORDERID

STAIN
STAINCOLOR
QUANTITY
PRODUCTNAME
UNITPRICE
STAINPRICE

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

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

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



データビューフィールドウィンドウ

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

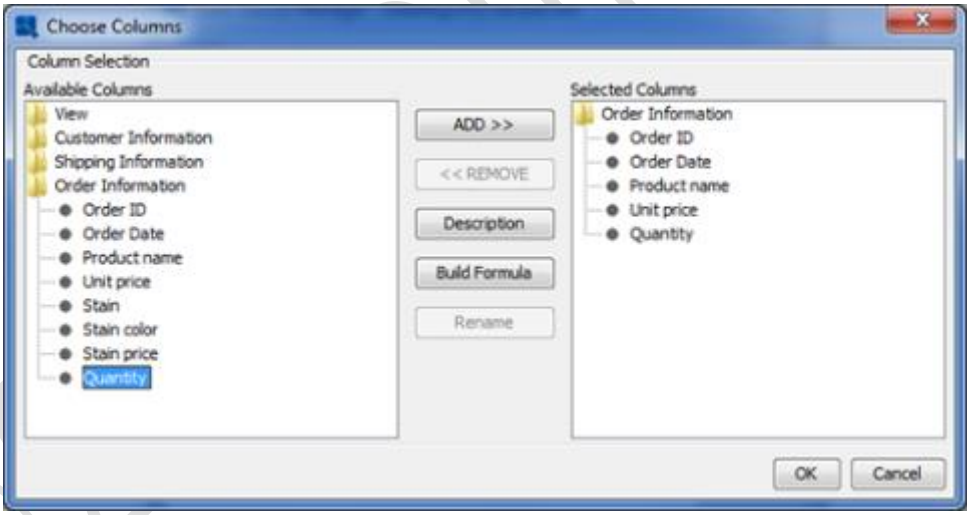
ャのデータビューの下に、新しいノードとして保存されます。

2.3.2.3.1 データビューのクエリ

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

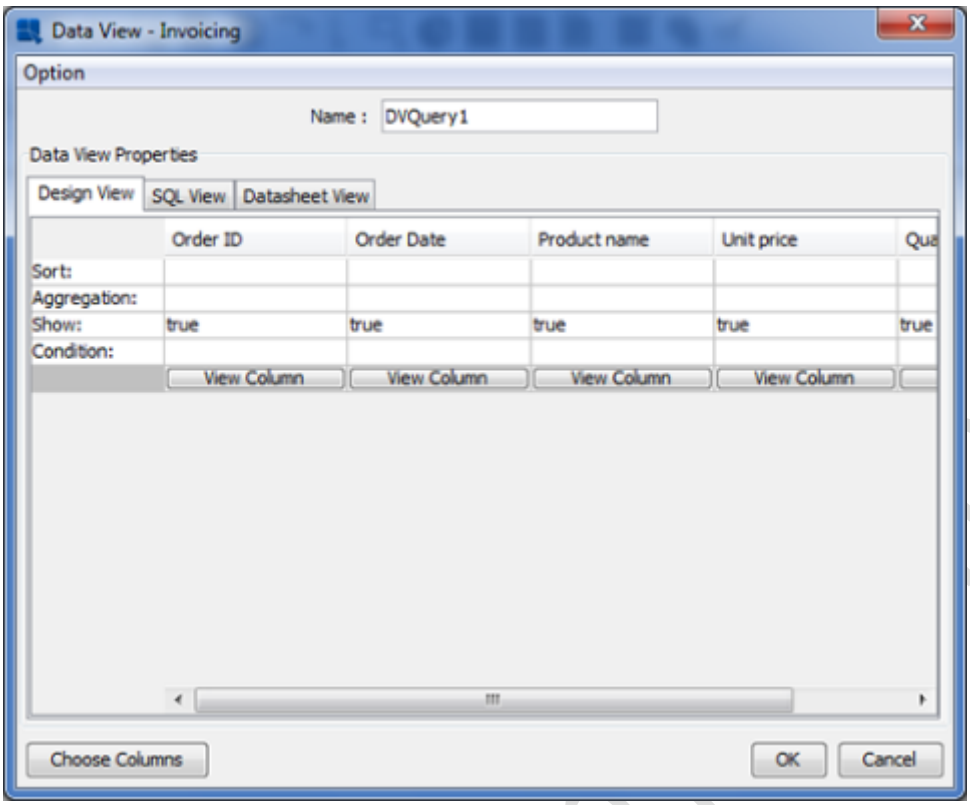
データソースマネージャで、**Invoicing** データビューを選択します。次に、**Next** ボタンをクリックします。これは、ビューからフィールドを選択するように促すダイアログを開きます。フィールドを選択するには、見出しをダブルクリックして展開します。それらを選択し、**Add** ボタンをクリックして、以下のフィールドをクエリに追加します：

Order ID
Order Date
Product name
Unit Price
Quantity



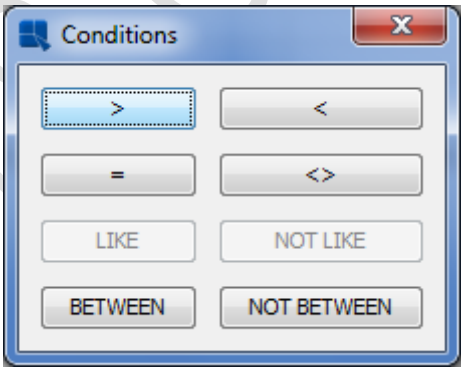
Data View Query Field Selection ダイアログ

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



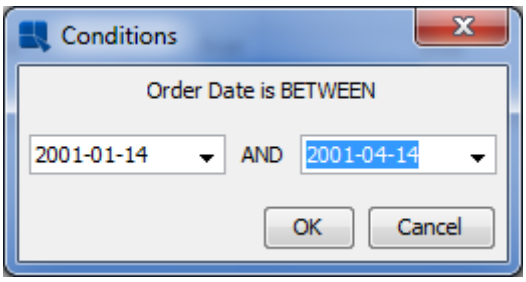
Data View Conditions ダイアログ

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



Specify Condition ダイアログ

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



Specify Condition Fields ダイアログ

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

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

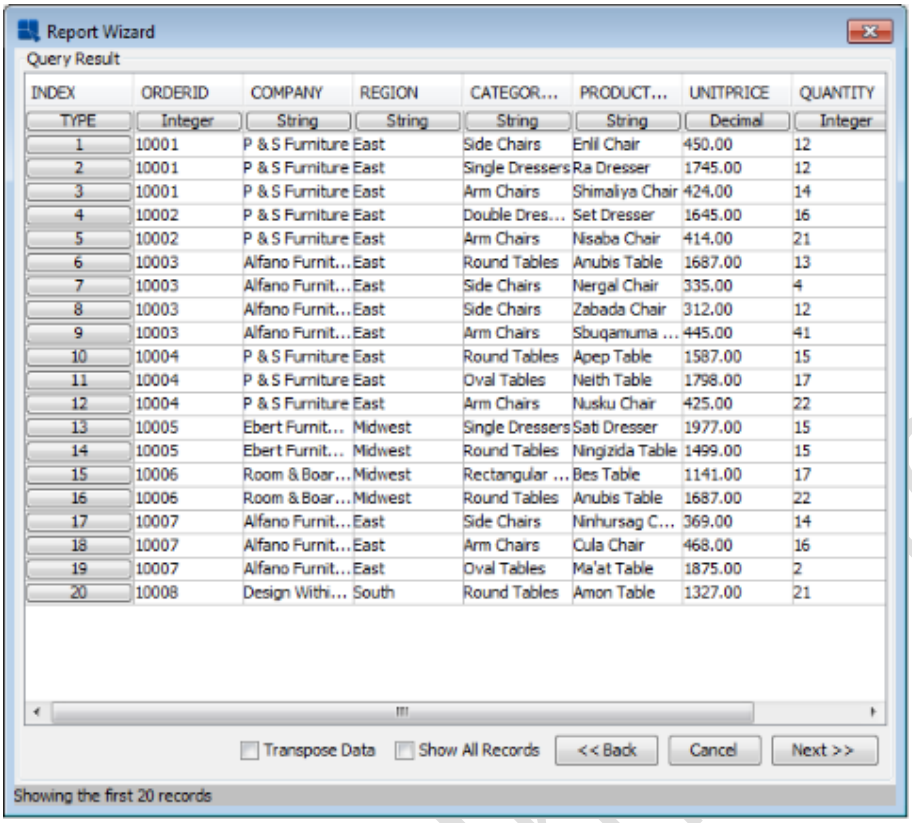
2.3.3 レポートマッピング

このセクションでは、データソースのデータをレポート構造にマップするさまざまな方法について説明します。このセクションでは、クエリを作成するで作成したクエリから結果セットを取り出し、それを使用して EspressReport がサポートするさまざまなレイアウトのレポートを生成します。

2.3.3.1 シンプルカラムナレイアウト

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

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



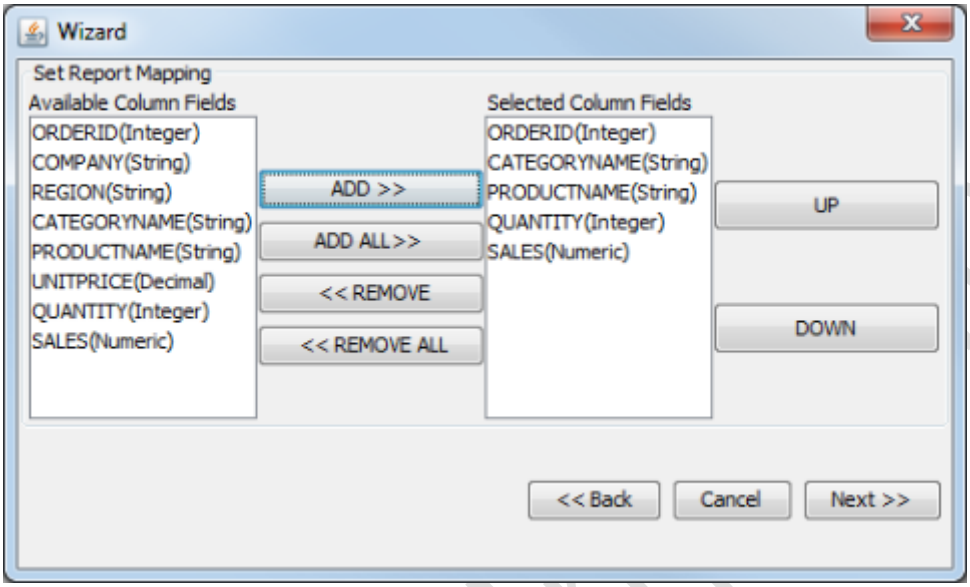
INDEX	ORDERID	COMPANY	REGION	CATEGOR...	PRODUCT...	UNITPRICE	QUANTITY
TYPE	Integer	String	String	String	String	Decimal	Integer
1	10001	P & S Furniture	East	Side Chairs	Enlil Chair	450.00	12
2	10001	P & S Furniture	East	Single Dressers	Ra Dresser	1745.00	12
3	10001	P & S Furniture	East	Arm Chairs	Shimaliya Chair	424.00	14
4	10002	P & S Furniture	East	Double Dres...	Set Dresser	1645.00	16
5	10002	P & S Furniture	East	Arm Chairs	Nisaba Chair	414.00	21
6	10003	Alfano Furnit...	East	Round Tables	Anubis Table	1687.00	13
7	10003	Alfano Furnit...	East	Side Chairs	Nergal Chair	335.00	4
8	10003	Alfano Furnit...	East	Side Chairs	Zabada Chair	312.00	12
9	10003	Alfano Furnit...	East	Arm Chairs	Sbuqamuma ...	445.00	41
10	10004	P & S Furniture	East	Round Tables	Apep Table	1587.00	15
11	10004	P & S Furniture	East	Oval Tables	Neith Table	1798.00	17
12	10004	P & S Furniture	East	Arm Chairs	Nusku Chair	425.00	22
13	10005	Ebert Furnit...	Midwest	Single Dressers	Sati Dresser	1977.00	15
14	10005	Ebert Furnit...	Midwest	Round Tables	Ningizida Table	1499.00	15
15	10006	Room & Boar...	Midwest	Rectangular ...	Bes Table	1141.00	17
16	10006	Room & Boar...	Midwest	Round Tables	Anubis Table	1687.00	22
17	10007	Alfano Furnit...	East	Side Chairs	Ninhursag C...	369.00	14
18	10007	Alfano Furnit...	East	Arm Chairs	Cula Chair	468.00	16
19	10007	Alfano Furnit...	East	Oval Tables	Ma'at Table	1875.00	2
20	10008	Design Withi...	South	Round Tables	Amon Table	1327.00	21

クエリ結果画面

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

Select Report Layout ダイアログ

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

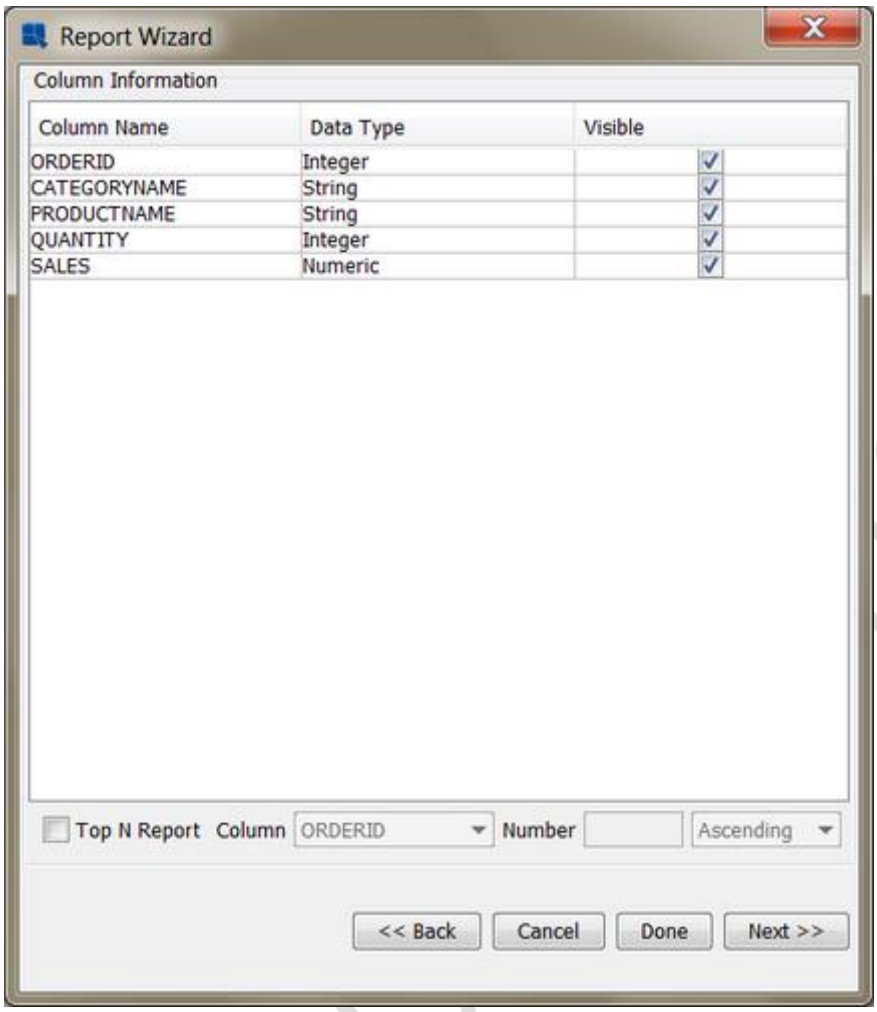


Column Selection/Ordering ダイアログ

このダイアログで、レポートの以下のフィールドを選択します。

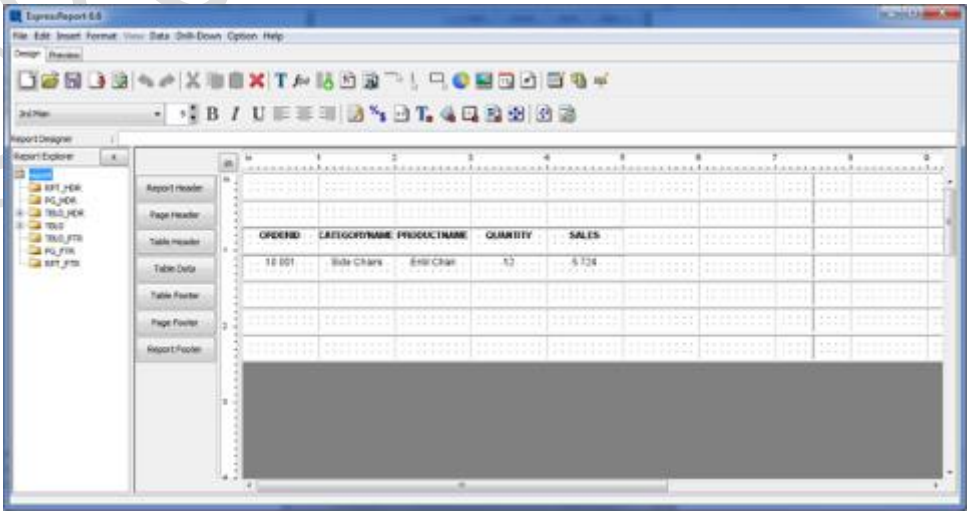
ORDERID
CATEGORYNAME
PRODUCTNAME
QUANTITY
SALES

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



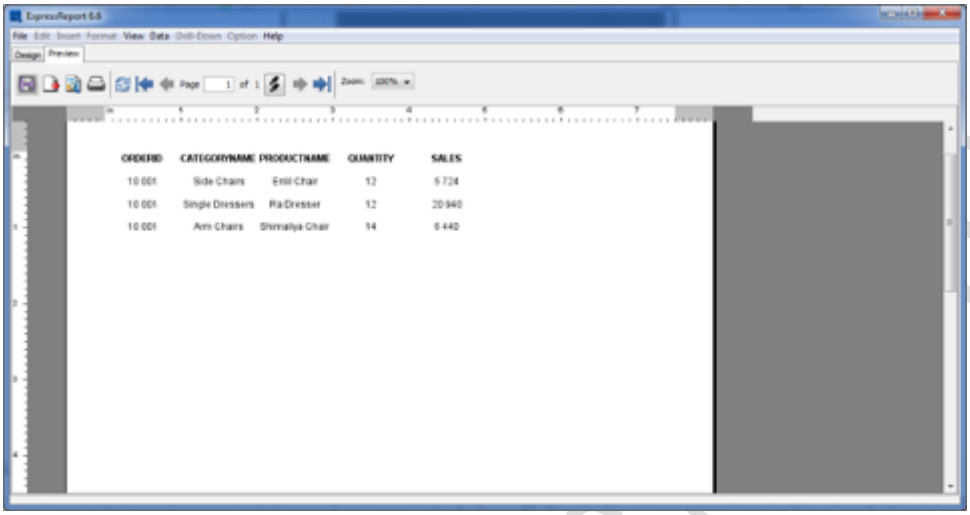
Data Mapping ダイアログ

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



デザインウィンドウでのシンプルカラムナレポート

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



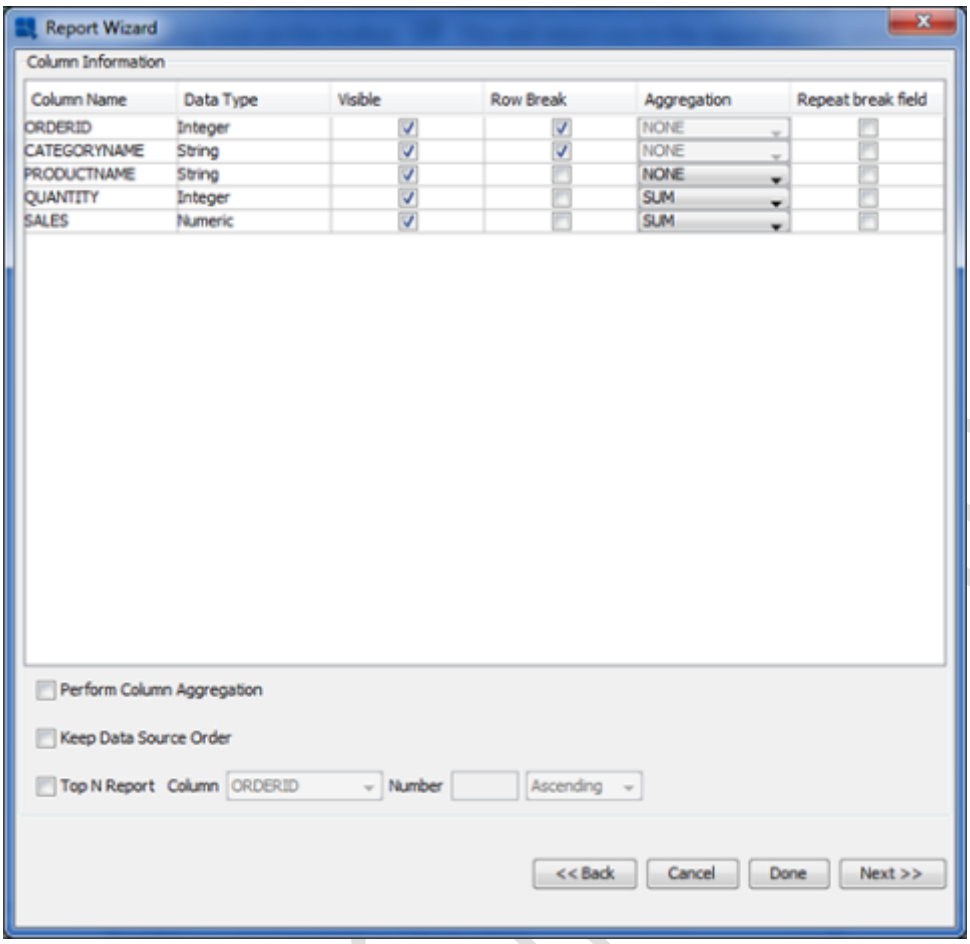
シンプルカラムナレポートのプレビュー

2.3.3.2 サマリブレークレイアウト

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

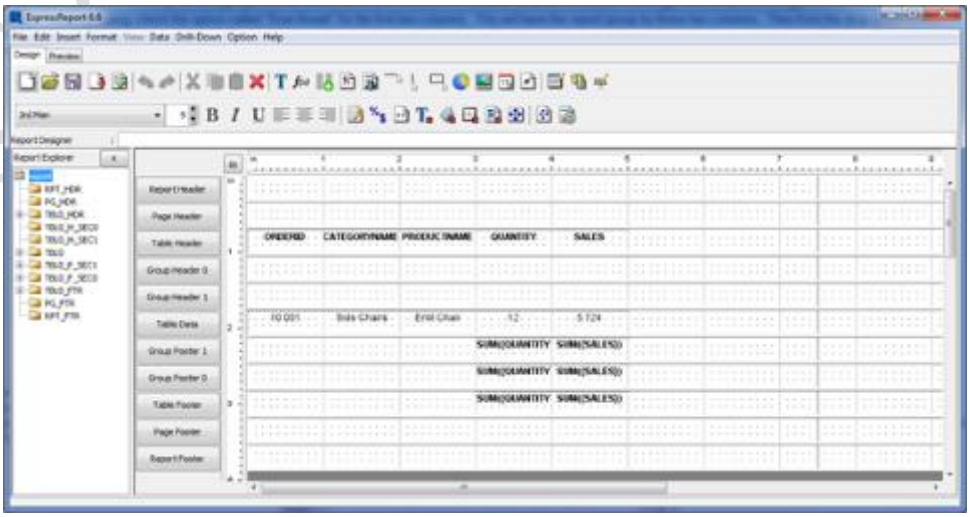
design タブに移動し、ツールバーの **Change Data Mapping** アイコン  を選択します。これにより、レポートウィザードが表示され、別のレポートレイアウトが選択されます。このダイアログから、**Select Report Format** ウィンドウに戻るまで **Back** ボタンを 2 回押します。

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



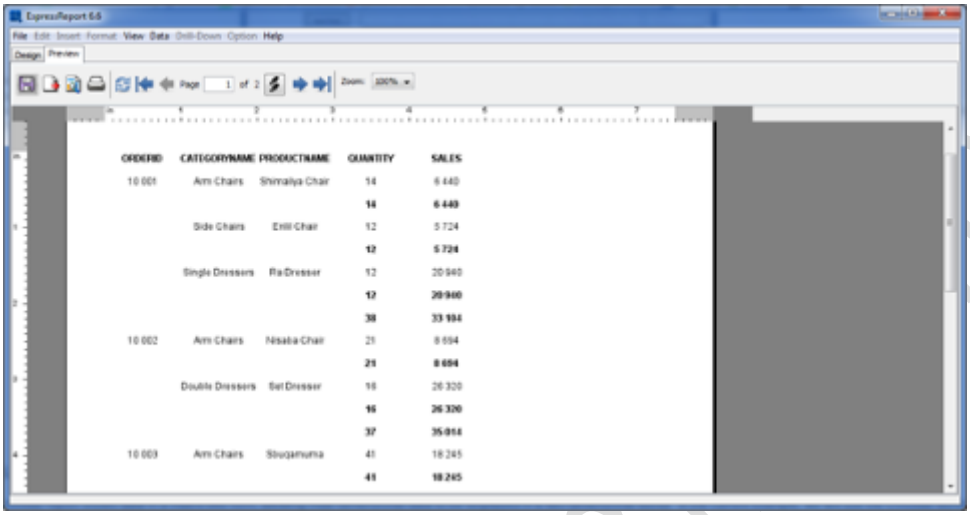
サマリブレークレイアウトのデータマッピング画面

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



デザインウィンドウでのサマリブレークレポートの中断

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



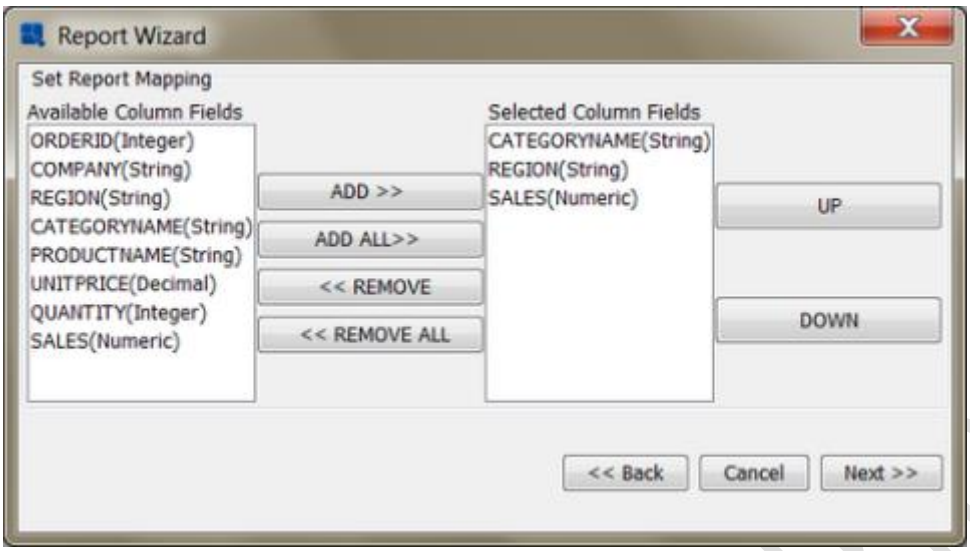
サマリブレークレポートのプレビューを中断する

2.3.3.3 クロス集計レイアウト

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

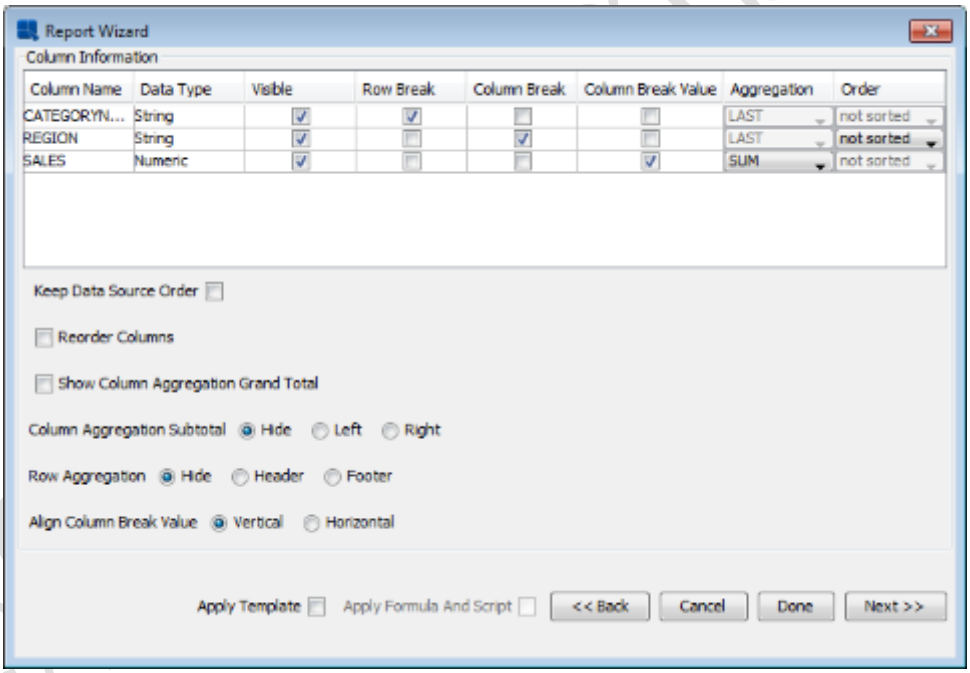
Design タブに移動し、**Change Data Mapping** アイコンを選択します。レポートウィザードが再度開いたら、レイアウト選択画面に戻るまで **Back** ボタンをクリックします。この画面で、**CrossTab** レイアウトを使用するように選択し、**Next** をクリックします。列選択/注文画面（ウィザードの次に）で、選択内容を以下のように変更します：

CATEGORYNAME
REGION
SALES



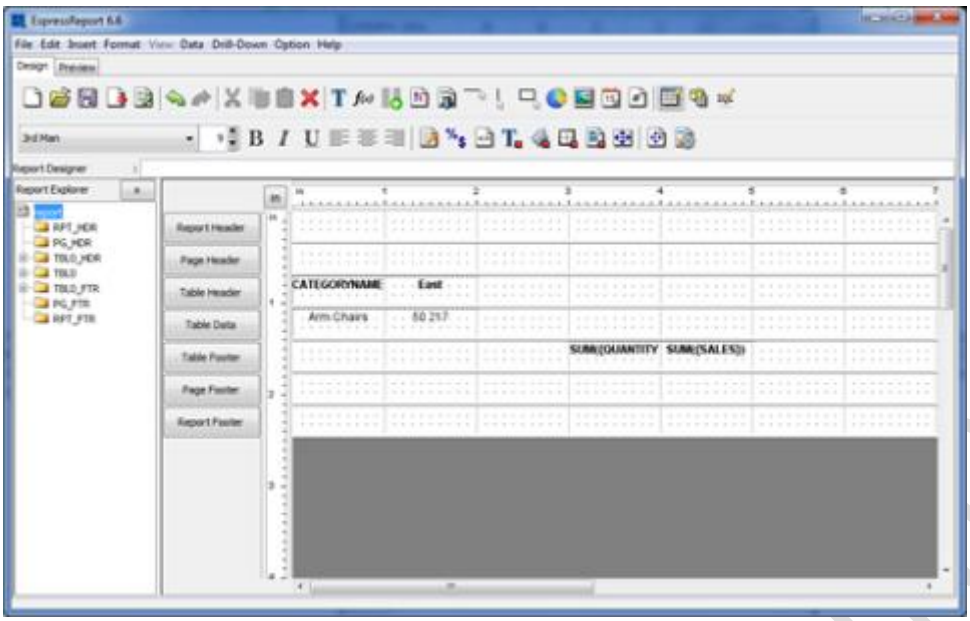
クロス集計レポートの列選択

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



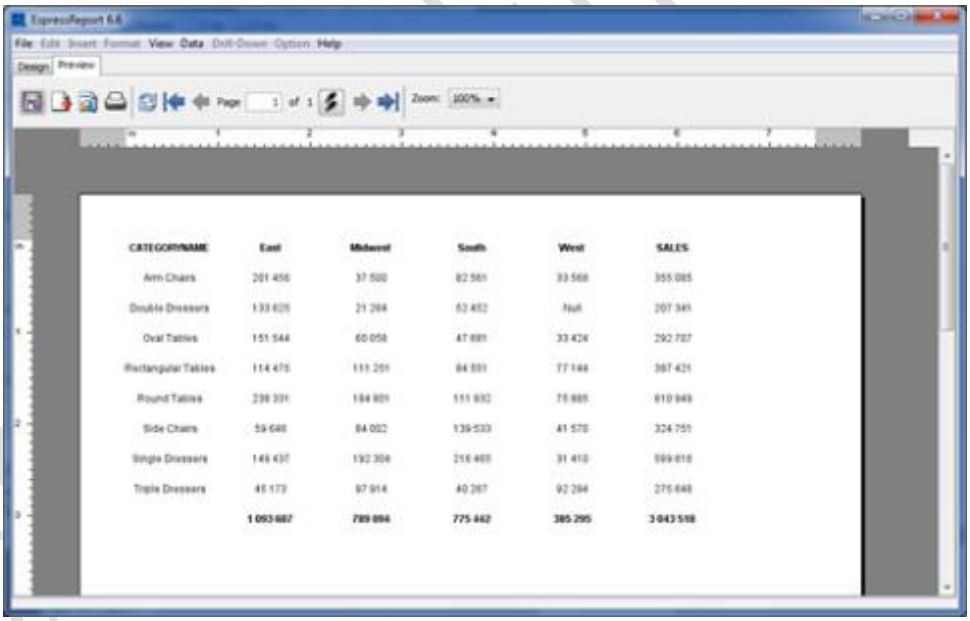
クロス集計レイアウトのデータマッピング画面

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



デザインウィンドウでのクロス集計レポート

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



クロス集計レポートのプレビュー

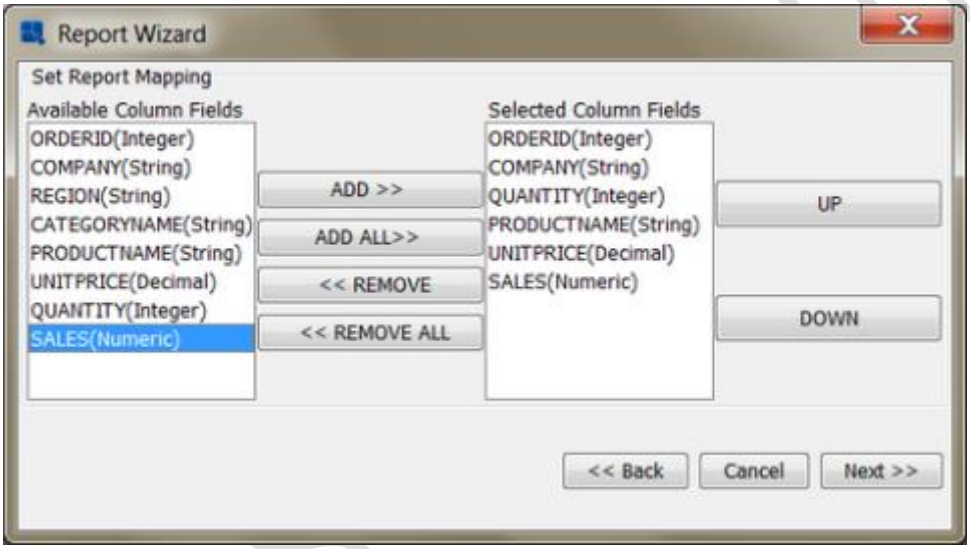
プレビューをもう一度試してみてください（デザインウィンドウに移動し、もう一度プレビューをクリックします）。今回は、小さな時間範囲を指定します。値がデータに存在しなくなったため、列の数がどのように減少するかに注目してください。日付範囲を再び拡大すると、列が再び表示されます。

2.3.3.4 マスタ&詳細レイアウト

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

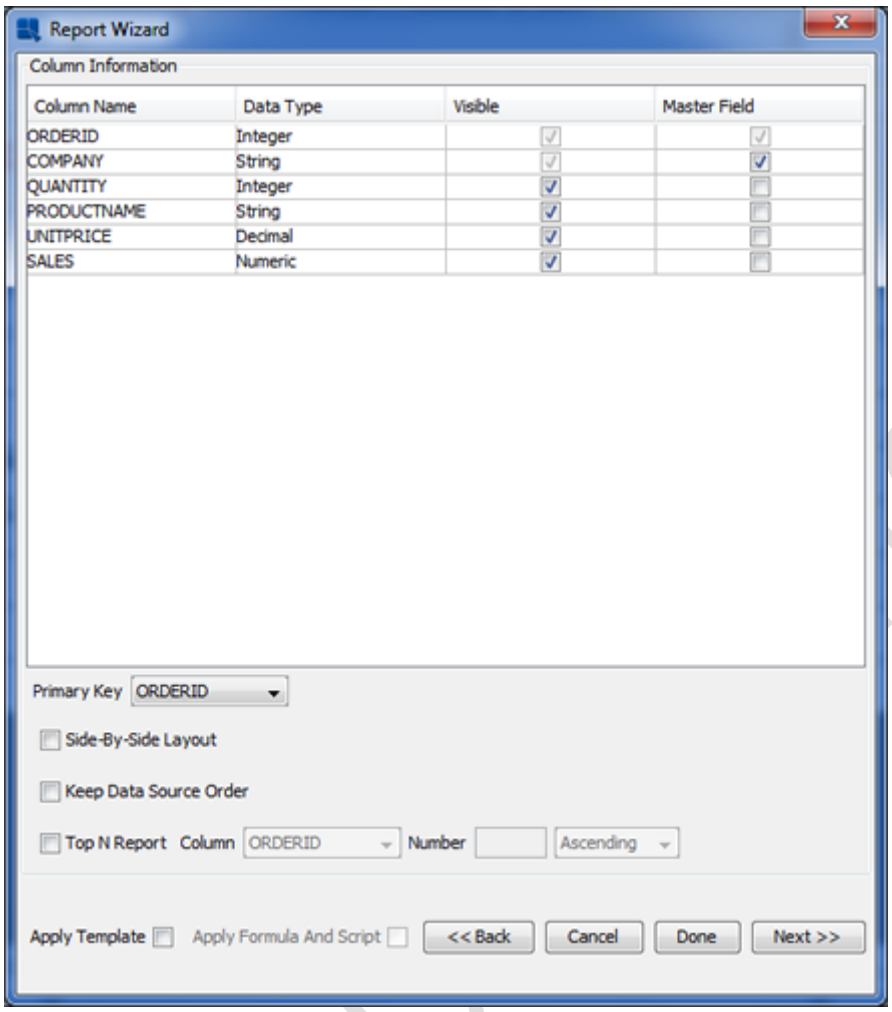
Design タブに戻り、ツールバーから **Change Data Mapping** を選択します。再度、レイアウト選択画面に戻ります。今回は、Master & Details レイアウトを選択し、**Next** ボタンをクリックして列選択画面に進みます。列選択画面で、以下の列を含むように選択を変更します：

ORDERID
COMPANY
QUANTITY
PRODUCTNAME
UNITPRICE
SALES



マスタ&詳細レイアウトの列選択

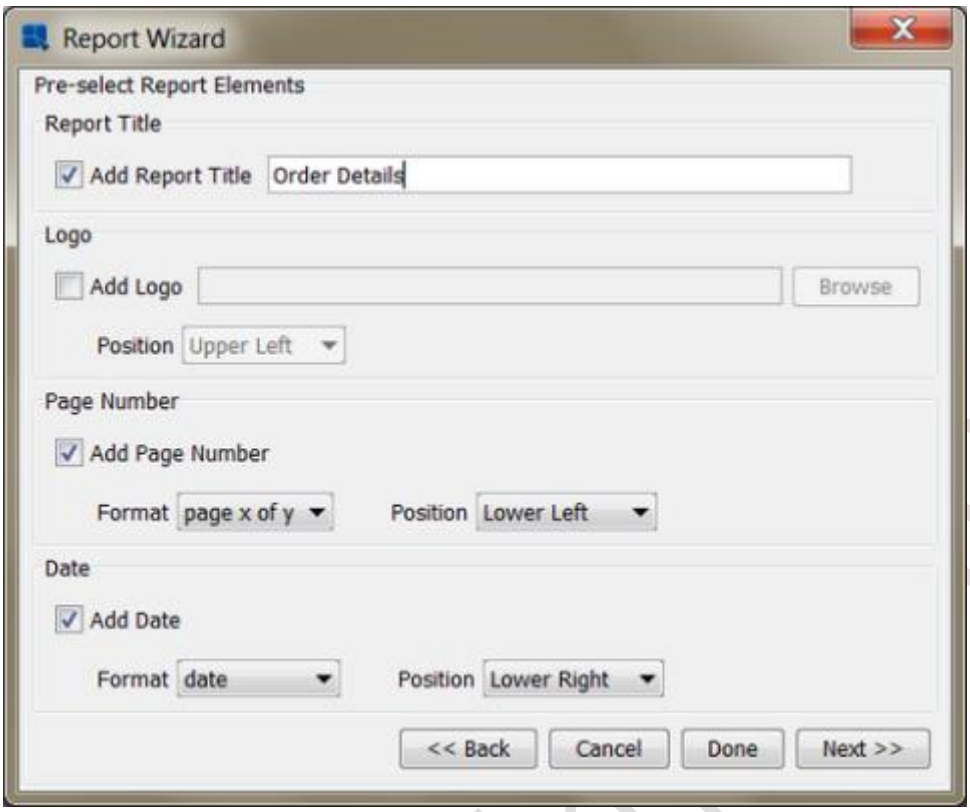
列を正しい順序で指定したら、**Next** ボタンをクリックして、マスタ&詳細レイアウトのデータマッピングダイアログを表示します。



マスタ&詳細レイアウトのデータマッピング画面

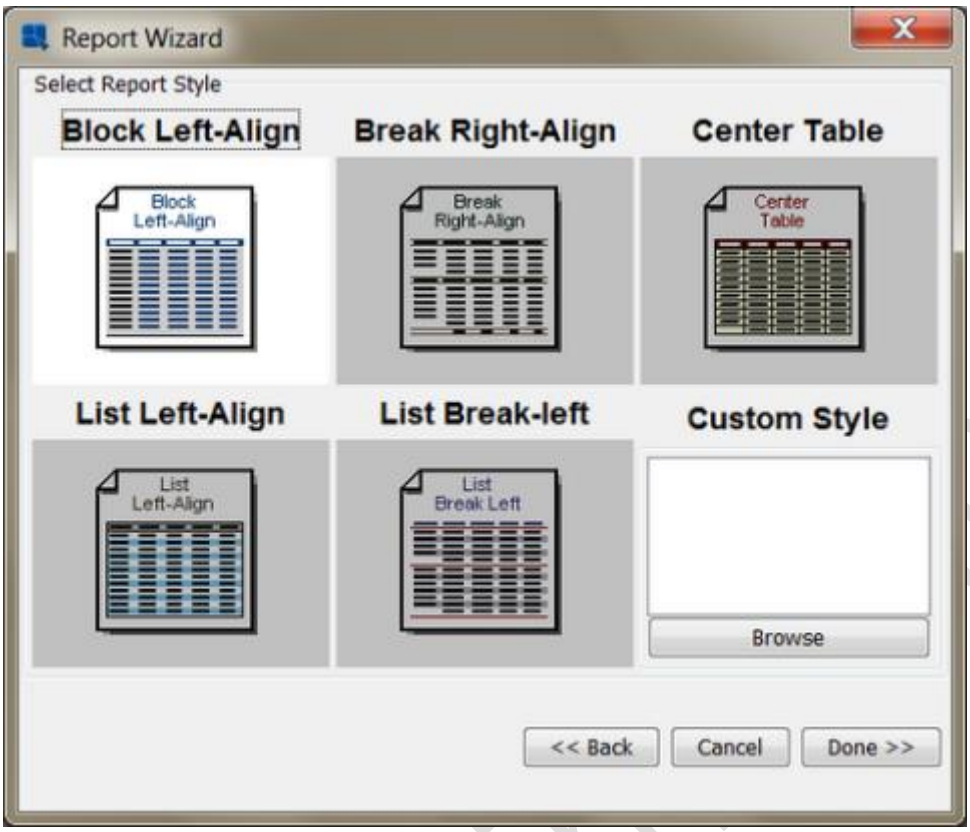
画面の左下部分のドロップダウンメニューから、**ORDERID** フィールドを **Primary Key** として選択します。これにより、データが **OrderID** フィールドでグループ化されます。次に、**Company** 列の **Master Field** オプションをチェックします。これにより、**Company** フィールドがレポートの Group Header セクションに配置されます。

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



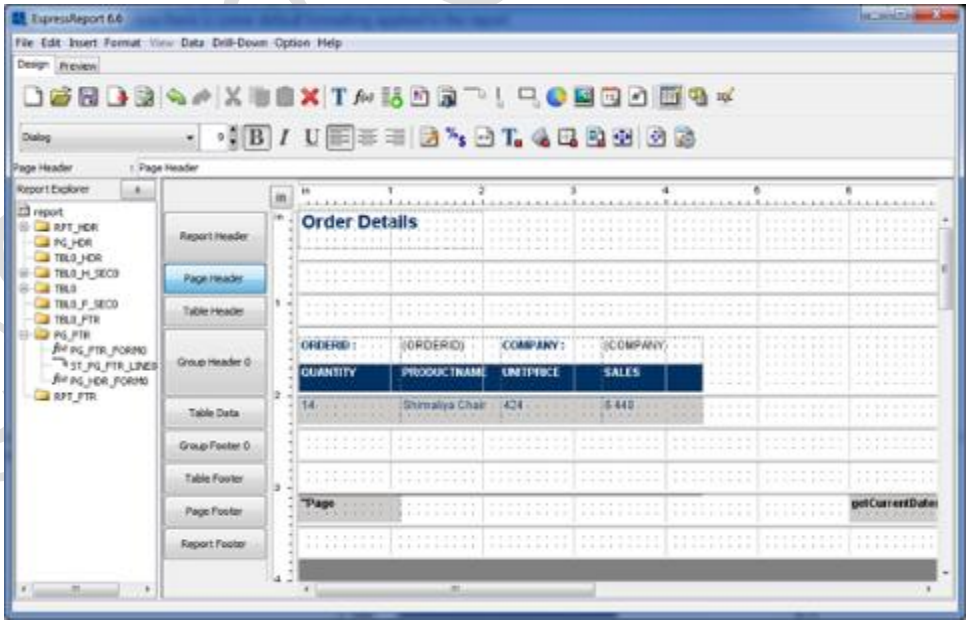
Add Report Elements ダイアログ

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



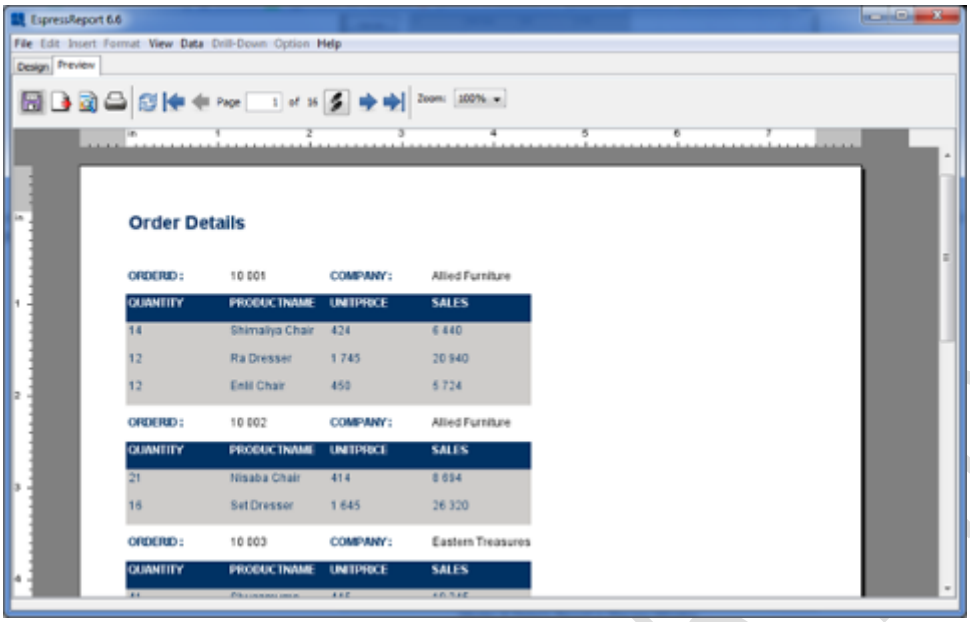
レポートスタイルの選択

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



デザインウィンドウのマスタ&詳細レポート（プリフォーマット）

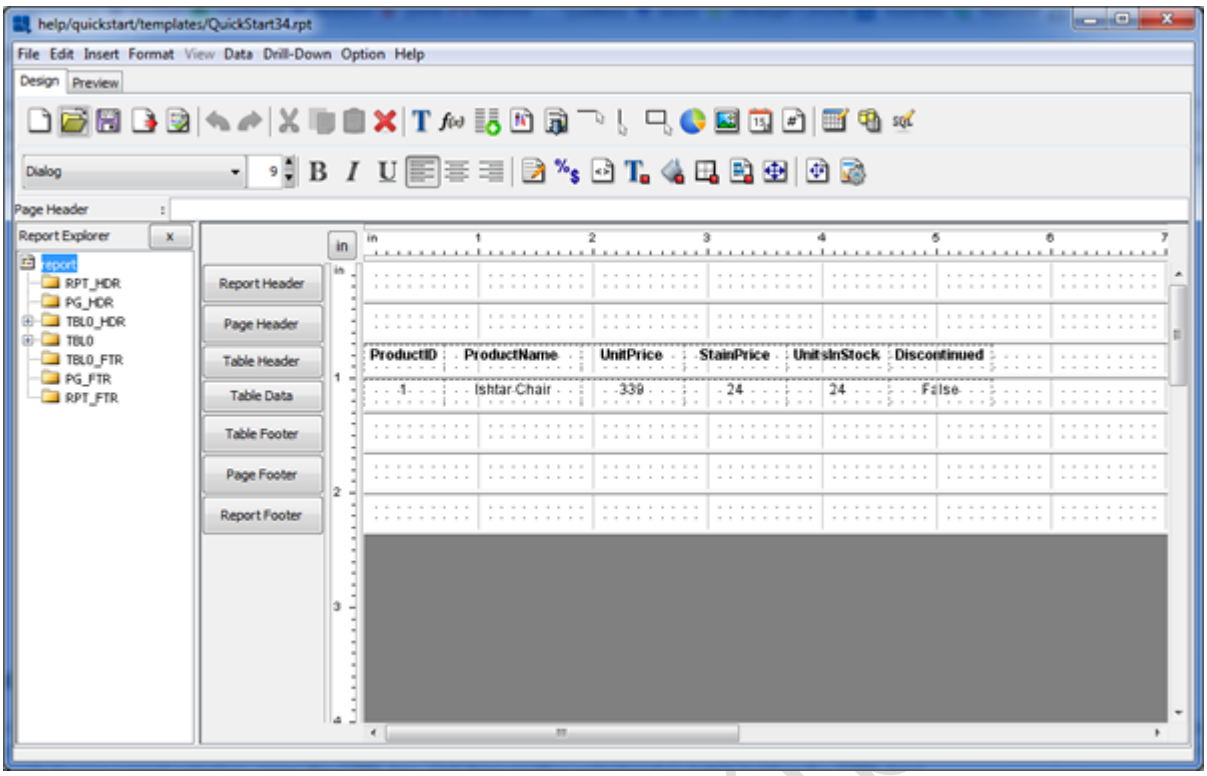
ご覧のとおり、**OrderID** フィールドと **Company** フィールドは、Group Header セクションに生成されています。レポートをプレビューすると、各注文のグループと、ページ番号と日付が配置されたセクション（ヘッダーまたはフッター）が表示されます。



プレビューウィンドウのマスタ&詳細レポート

2.3.4 基本的なレポート書式設定

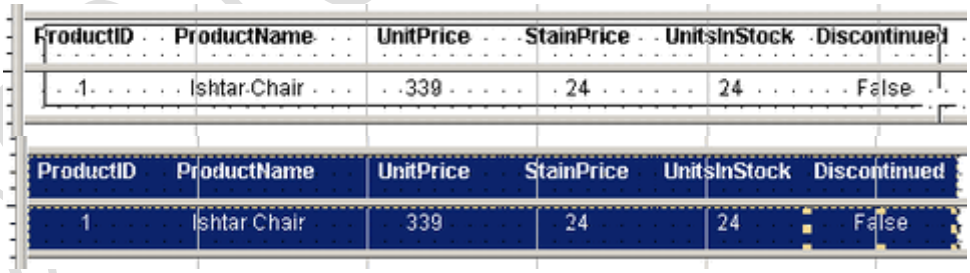
このセクションでは、書式なしのテンプレートを開き、EspressReport の基本的な書式設定機能を使用して洗練されたプレゼンテーションを作成します。Report Designer のデザインウィンドウで、**Open** ボタン  をクリックします。これにより、開くテンプレートのファイル名を指定するダイアログが表示されます。**Browse** ボタンをクリックし、**<InstallDir>/help/quickstart/templates** ディレクトリに移動します。一度そこにいたら、**QuickStart34.rpt** ファイルを選択して開きます。レポートウィンドウが開きます。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



デザインウィンドウでの QuickStart34.rpt

2.3.4.1 レポート要素の移動と整列

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

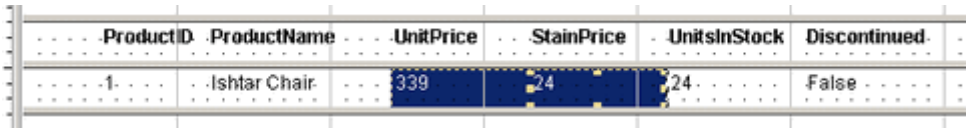


Report Designer の選択ボックスの描画

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

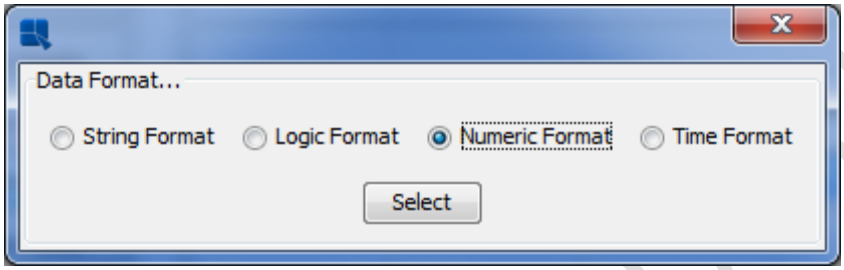
2.3.4.2 データフォーマット

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



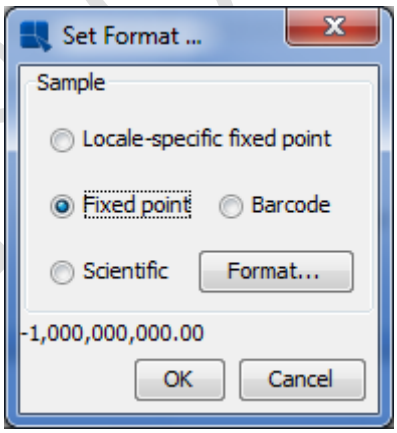
2 列の選択

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

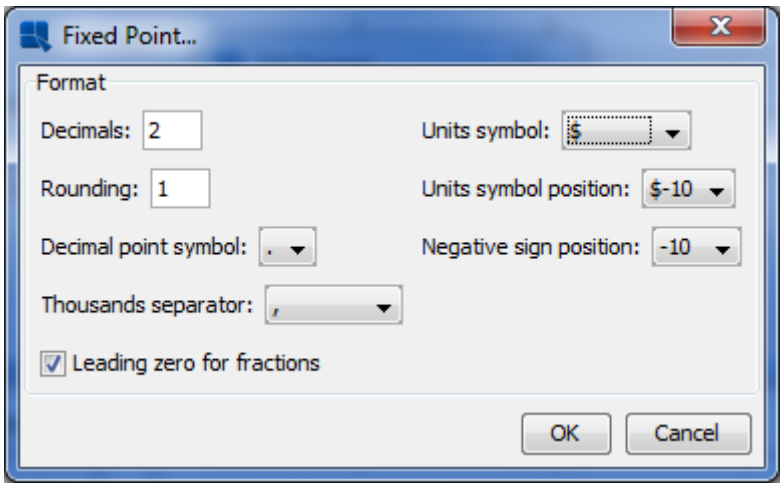


Data Type for Formatting ダイアログ

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



Set Format ダイアログ

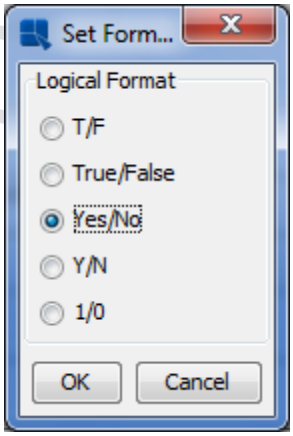


Numeric Format ダイアログ

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

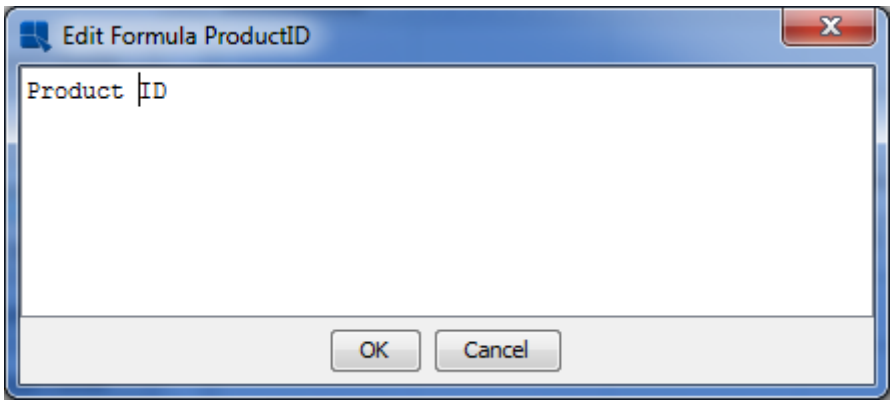
Currency Format の数値データ

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



Data Format for Boolean ダイアログ

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



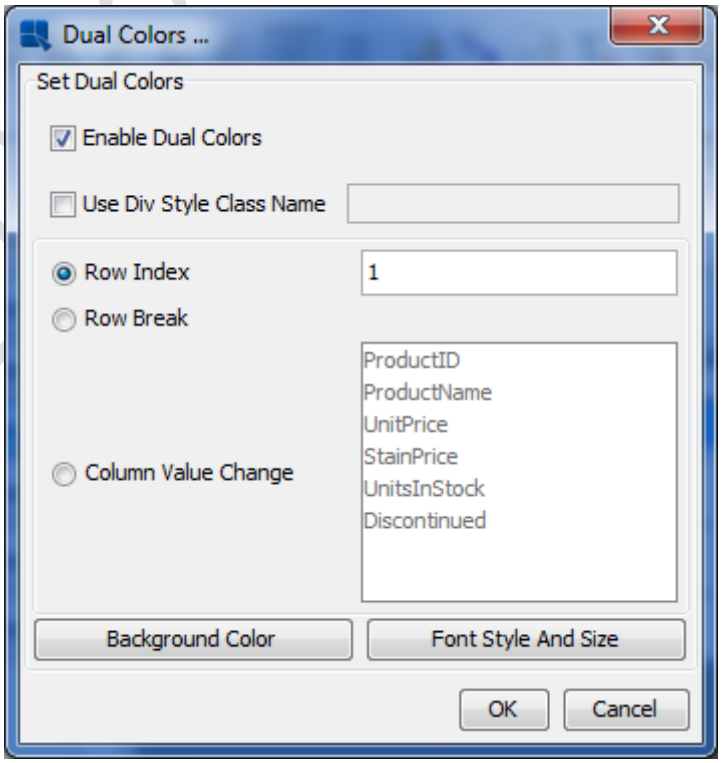
Edit Column Header ダイアログ

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

2.3.4.3 デュアルカラーを設定する

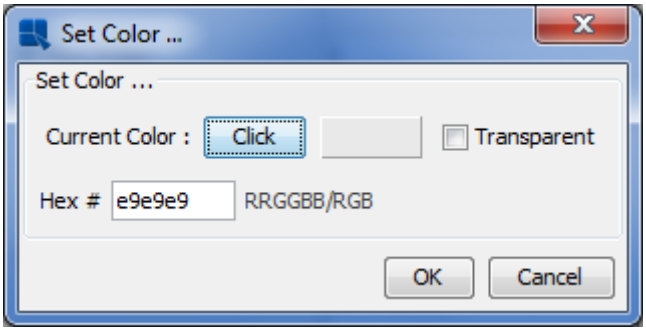
EspressReport のデュアルカラー機能により、ユーザーは背景色やフォントを変更することで、異なる行やグループのデータを区別することができます。デュアルカラーをオンにするには、グループ選択ツールを使用して、ヘッダーではなくレポートのすべての列を選択します。

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



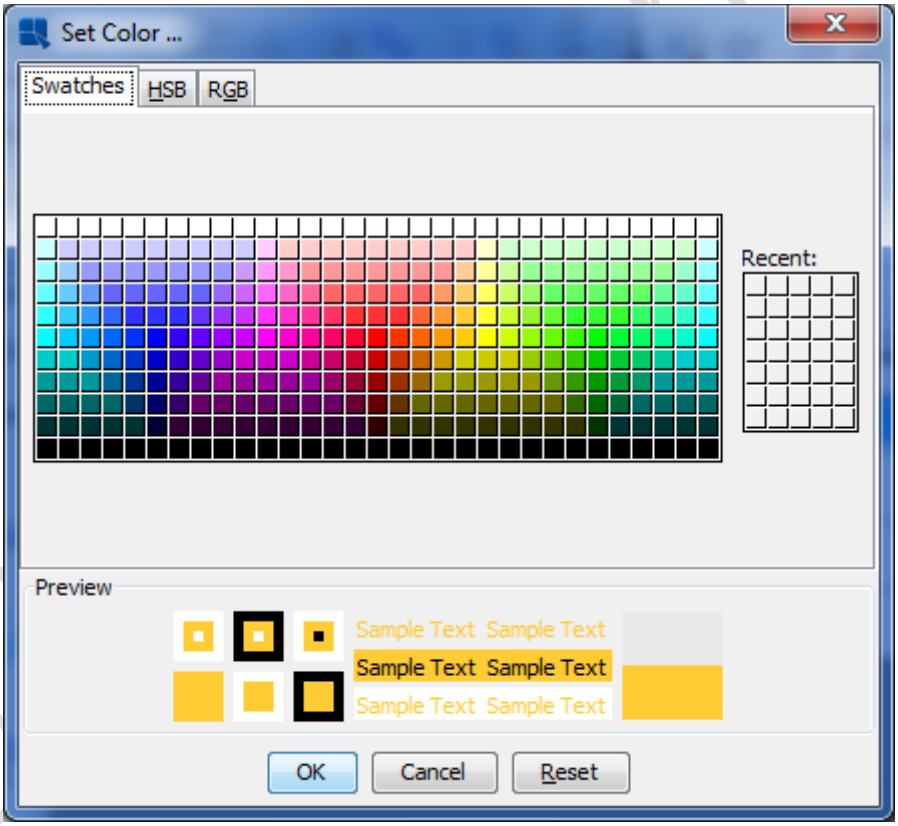
Dual Colors ダイアログ

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



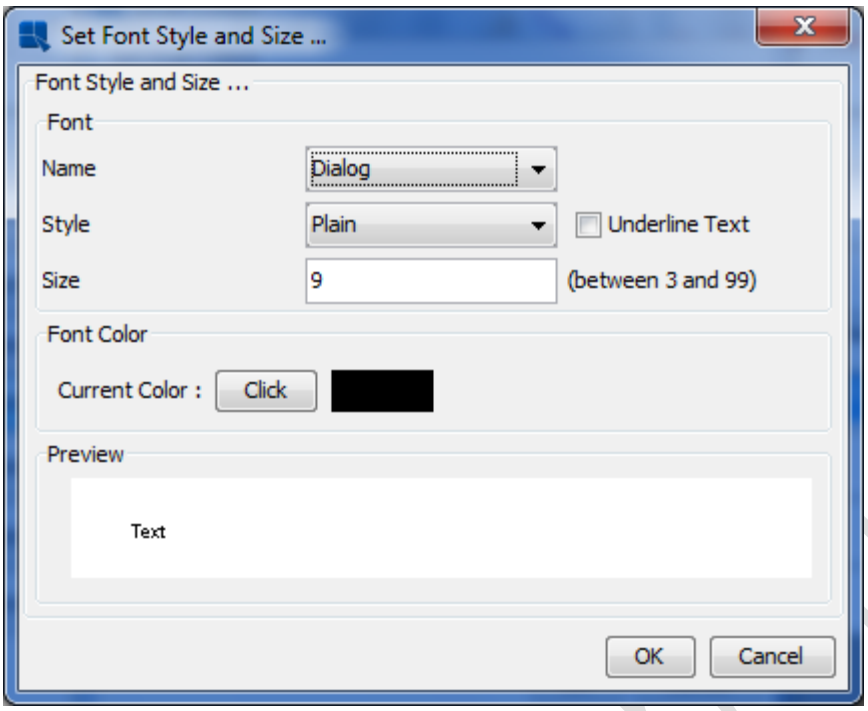
Background Transparency ダイアログ

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



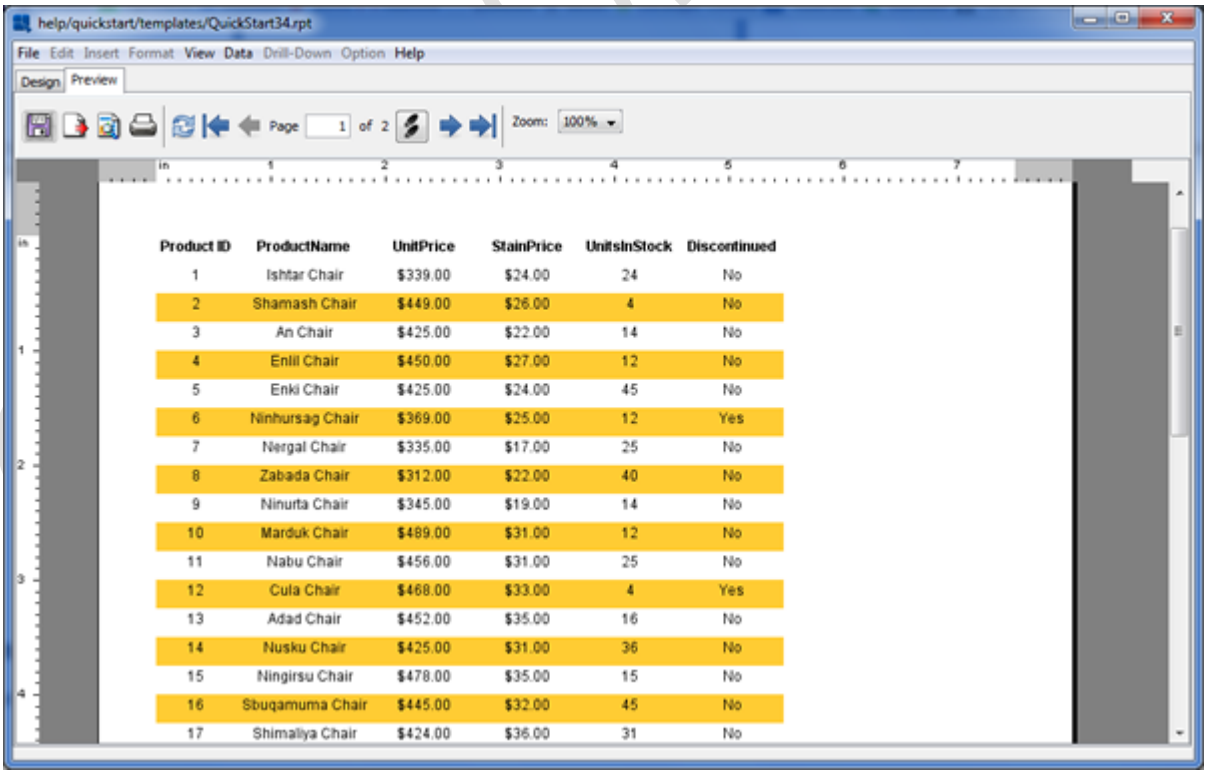
Choose Color ダイアログ

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



Font Style and Size ダイアログ

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



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

Dual Colors のプレビュー

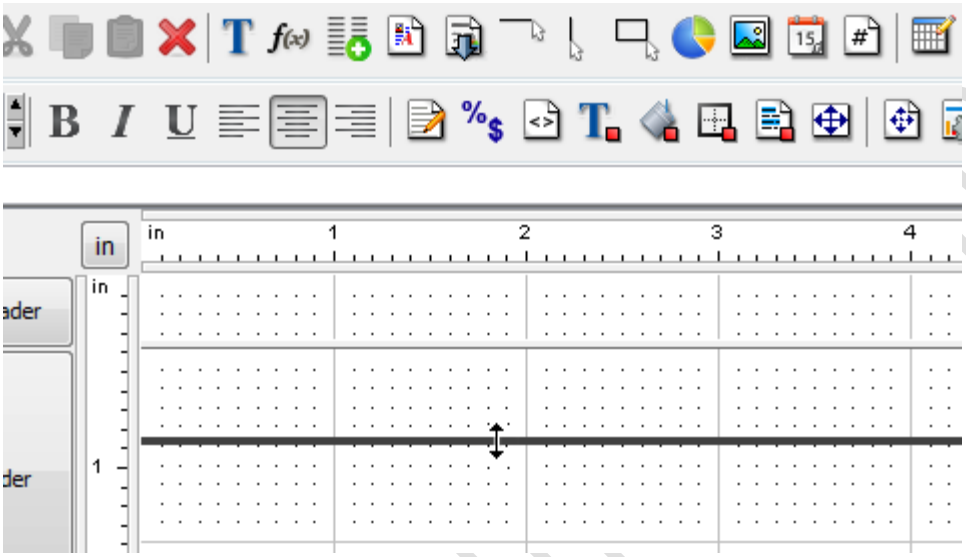
2.3.4.4 要素を挿入する

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

す

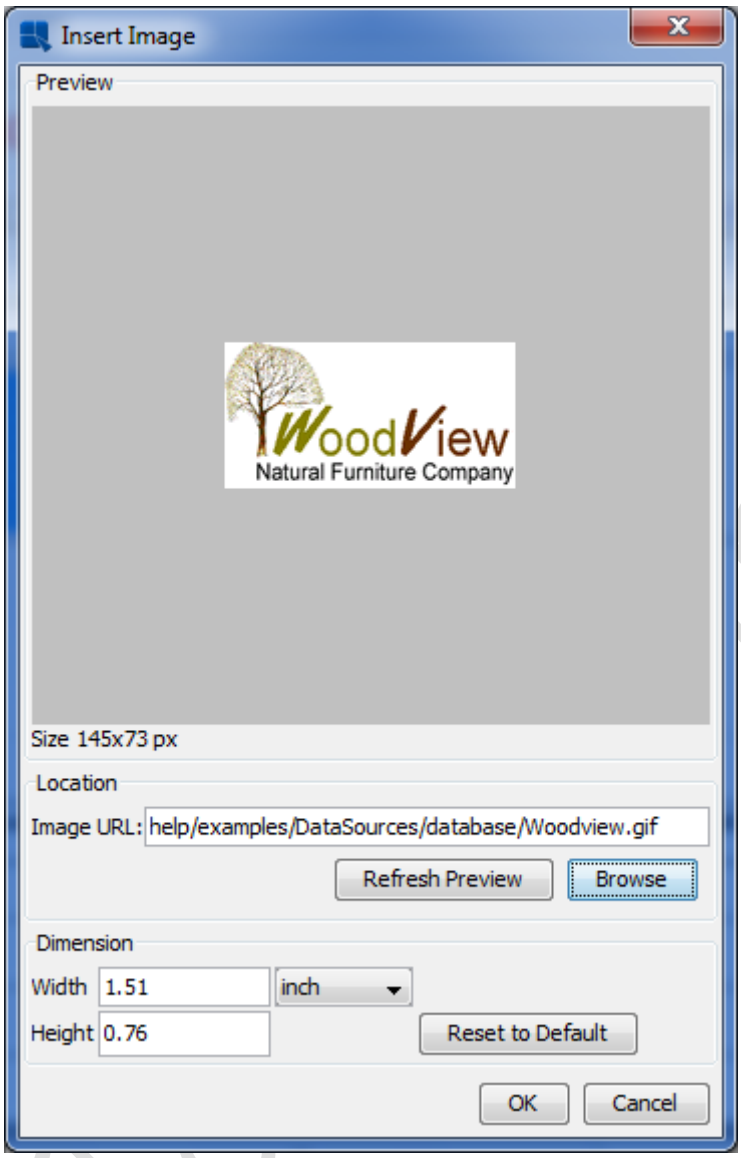
2.3.4.4.1 イメージを挿入する

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



レポートセクションのサイズ変更

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

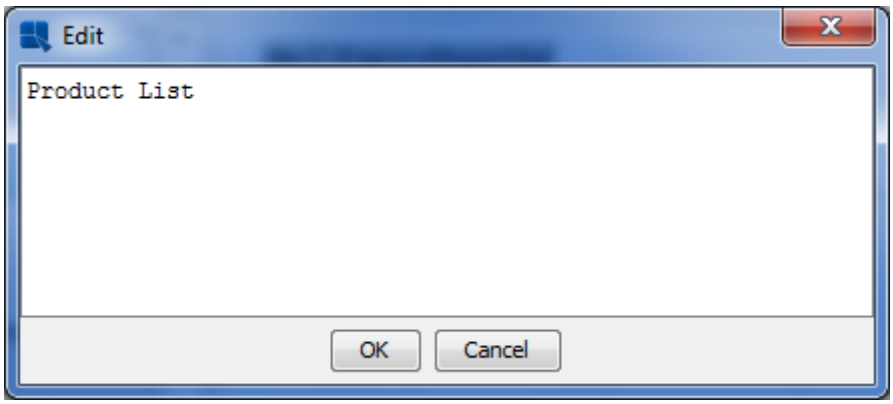


Insert Image ダイアログ

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

2.3.4.4.2 タイトルを挿入する

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



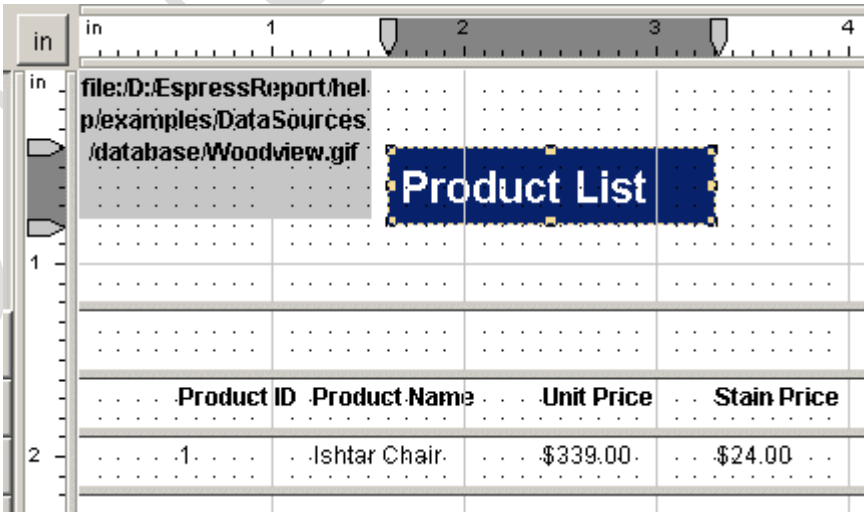
Insert Label ダイアログ

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

ツールバーのフォントオプション

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

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

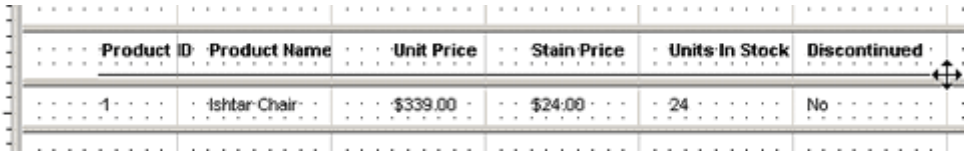


画像とタイトルを含むレポートヘッダー

2.3.4.4.3 ラインを挿入する

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

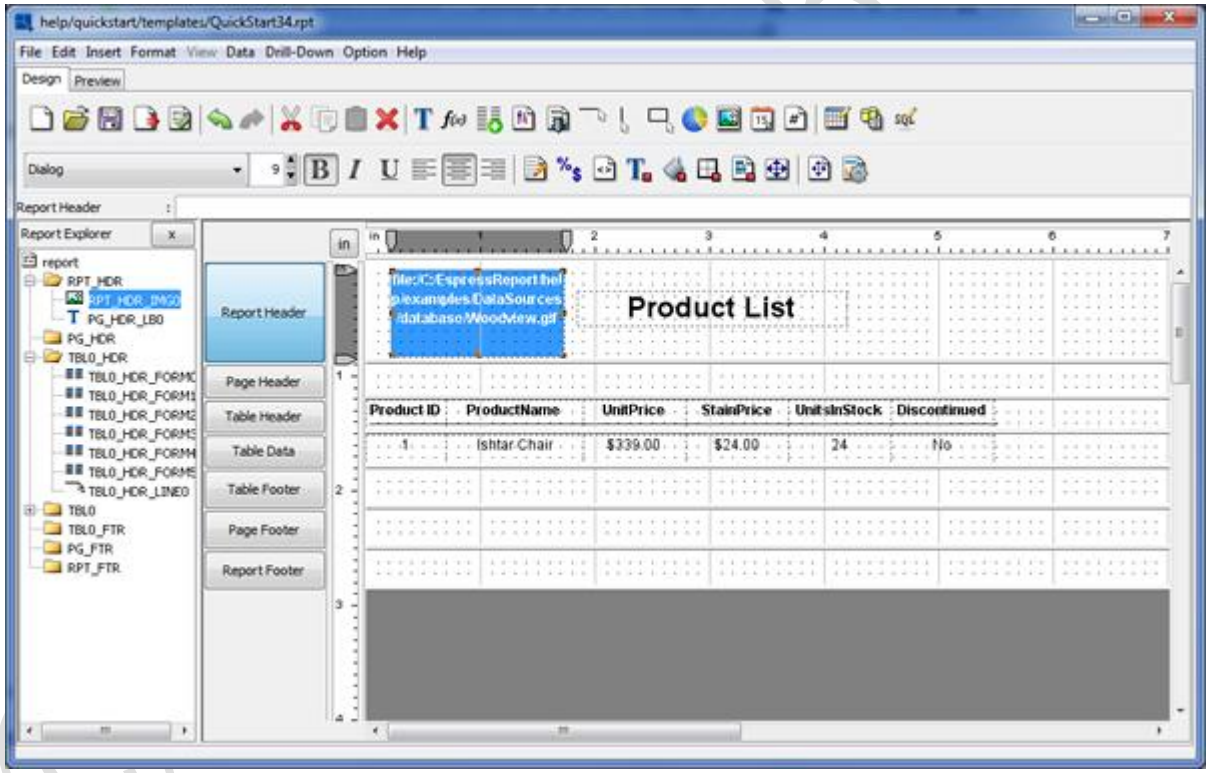
Horizontal Line ボタン を選択します。カーソルが十字に変わります。ProductID ヘッダーの下をクリックし、最後の列にドラッグして線を描画します。



Report Designer で線を描く

2.3.4.5 レポートエクスプローラでのレポート要素の表示

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



エクスプローラを開いた Designer

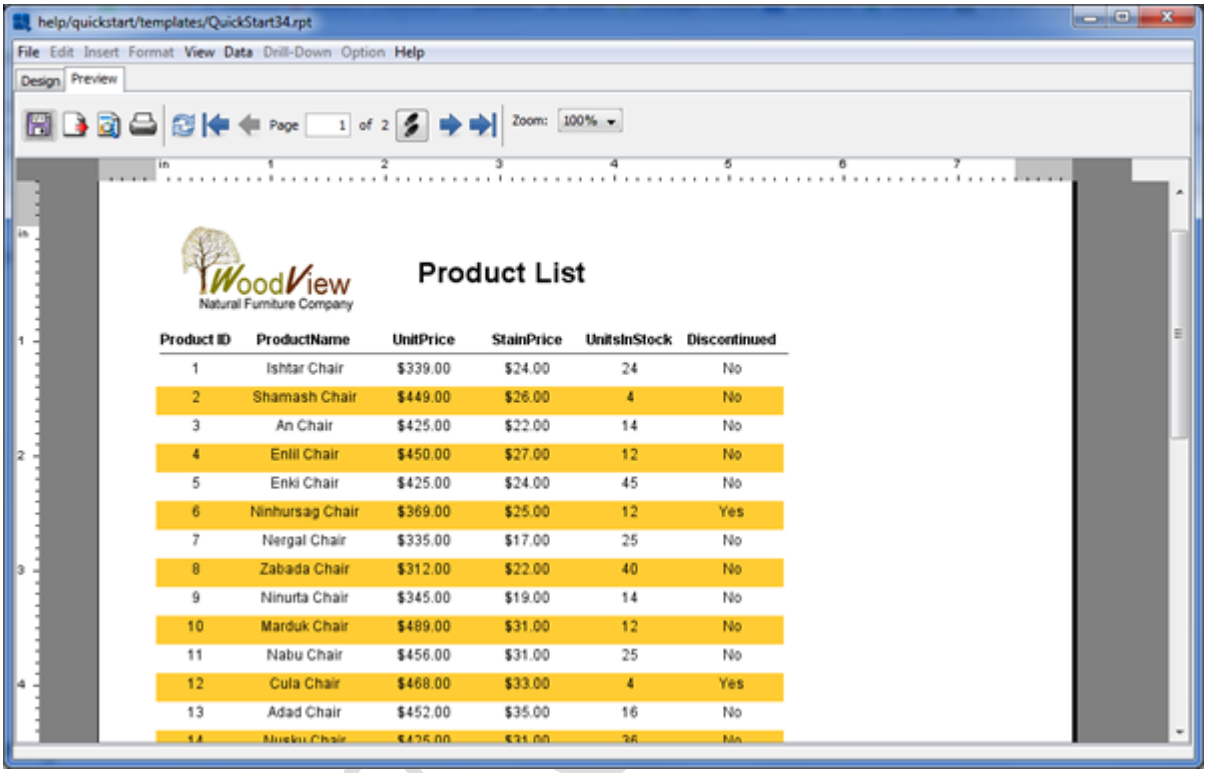
Option > **Report Explorer** を選択するか、**Report Explorer** パネルの右上隅にある **X** ボタンをクリックして、Report Explorer を閉じることができます。

2.3.4.6 セクションオプションを設定する

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

ポップアップメニューから **Repeat On Every Page** を選択します。これにより、テーブルヘッダーが各レポートページで毎回描画されます（これがデフォルトです）。

レポートの書式を設定したら、プレビューして結果を確認します。



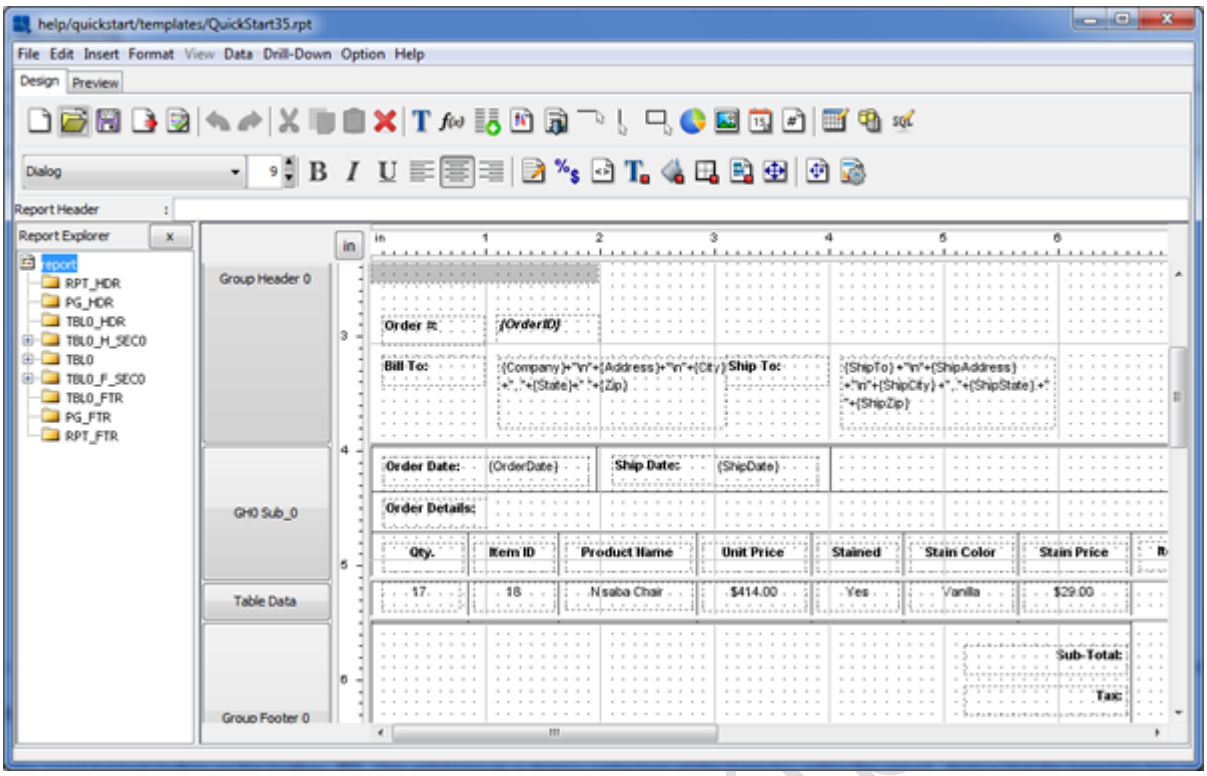
Product ID	ProductName	UnitPrice	StainPrice	UnitsInStock	Discontinued
1	Ishtar Chair	$339.00	$24.00	24	No
2	Shamash Chair	$449.00	$26.00	4	No
3	An Chair	$425.00	$22.00	14	No
4	Enlil Chair	$450.00	$27.00	12	No
5	Enki Chair	$425.00	$24.00	45	No
6	Ninhursag Chair	$369.00	$25.00	12	Yes
7	Nergal Chair	$335.00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

完成したレポート

2.3.5 数式とスクリプティング

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

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

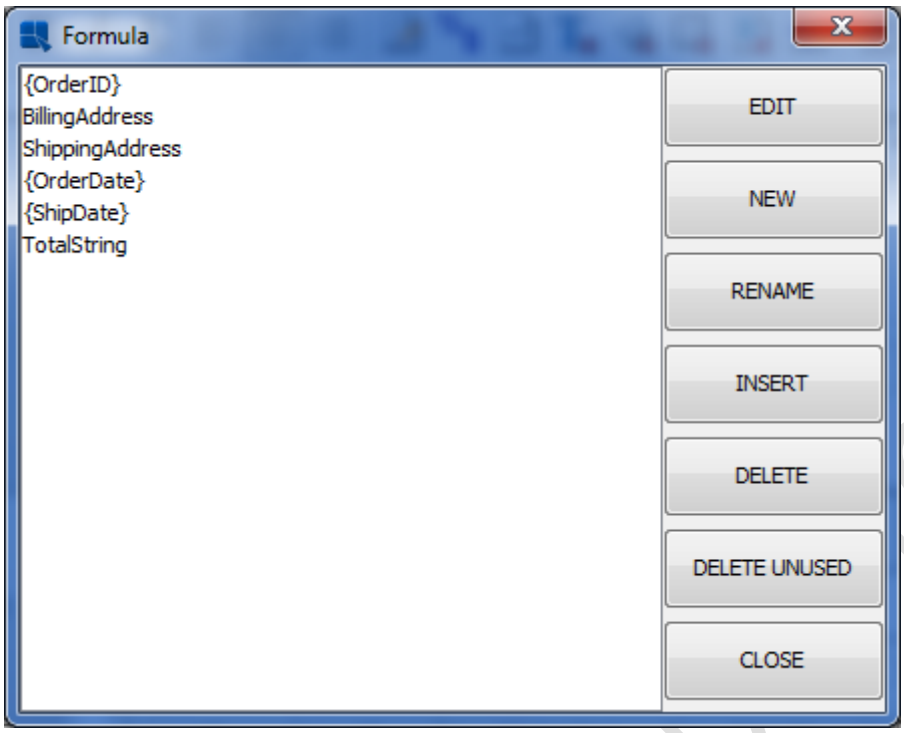


デザインウィンドウでの QuickStart35.rpt

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

2.3.5.1 数式を追加する

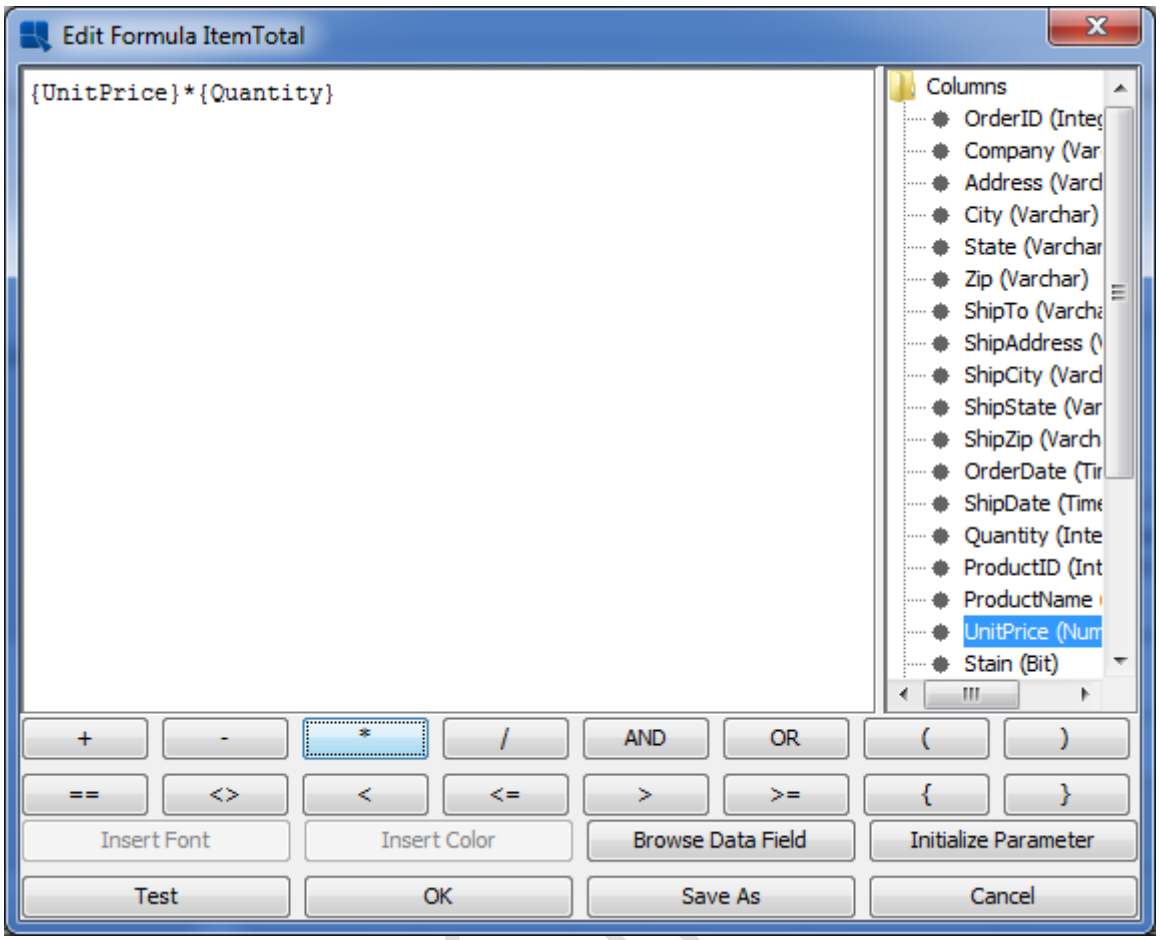
数式を挿入するには、ツールバーの **Insert Formula** ボタン $f(x)$ をクリックします。これにより、レポート内のすべての数式を含むダイアログが表示されます。テンプレートにはいくつかの既存の数式があることに注意してください。**NEW** ボタンをクリックして新しい数式を作成し、プロンプトに **ItemTotal** という名前を入力します。



Report Formula List

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

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



Formula Builder ウィンドウ

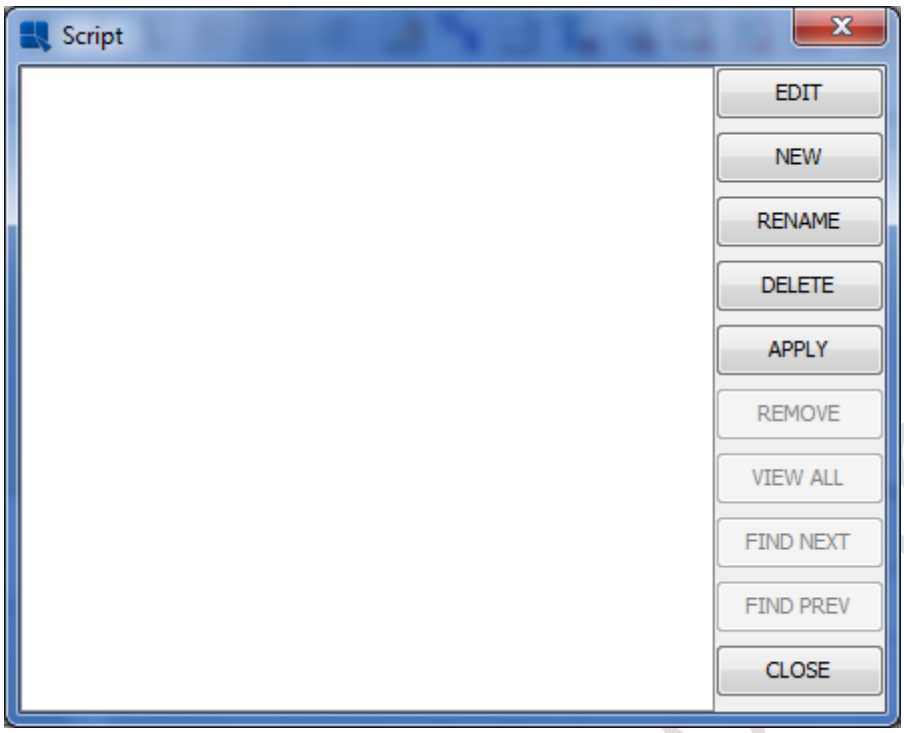
Test ボタンをクリックして、数式が正しく入力されていることを確認します。次に **OK** をクリックします。新しい数式が追加された数式リストに戻ります。数式リストから、作成した **ItemTotal** 数式を選択し、**Insert** ボタンをクリックします。ダイアログが閉じて、小さな点線の四角形がデザインウィンドウの周りのポインタの後に表示されます。**Item Total** ラベルの下および Table Data セクションの行間に数式を配置し、クリックします。計算式がレポートに追加されます。

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

2.3.5.2 スクリプトを追加する

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

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



Script List ウィンドウ

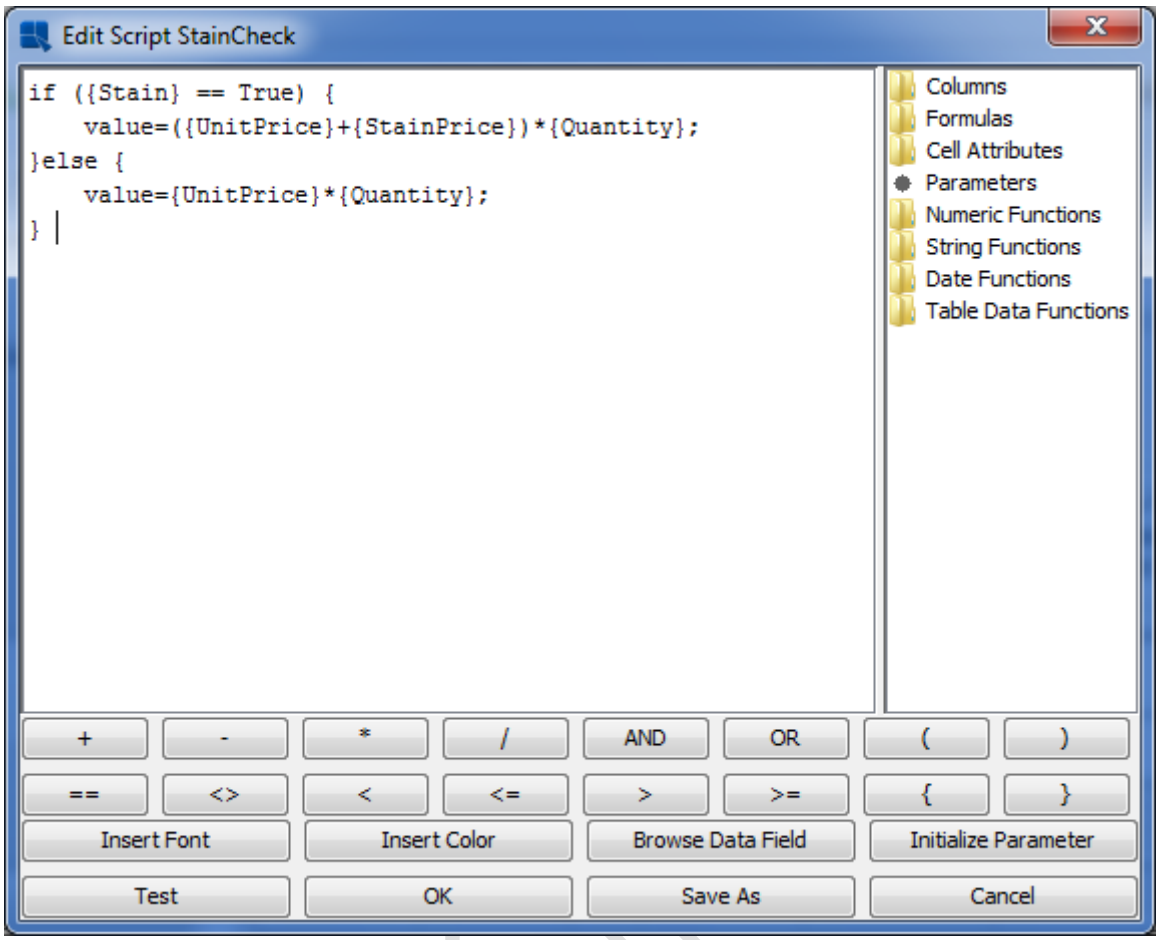
NEW をクリックして新しいスクリプトを作成します。プロンプトで、スクリプトの **StainCheck** という名前を入力します。**OK** をクリックすると、数式ビルダーが開き、スクリプトを追加できます。次のスクリプトを入力します。

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

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

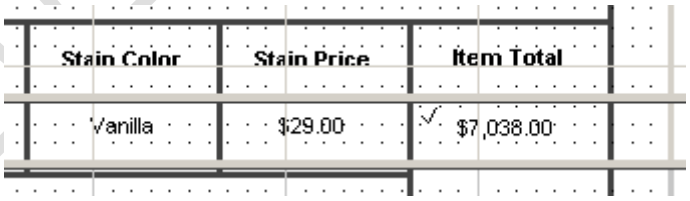
{Stain} == True

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



セルスクリプトの Formula Builder

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



デザインウィンドウでスクリプトを適用したセル

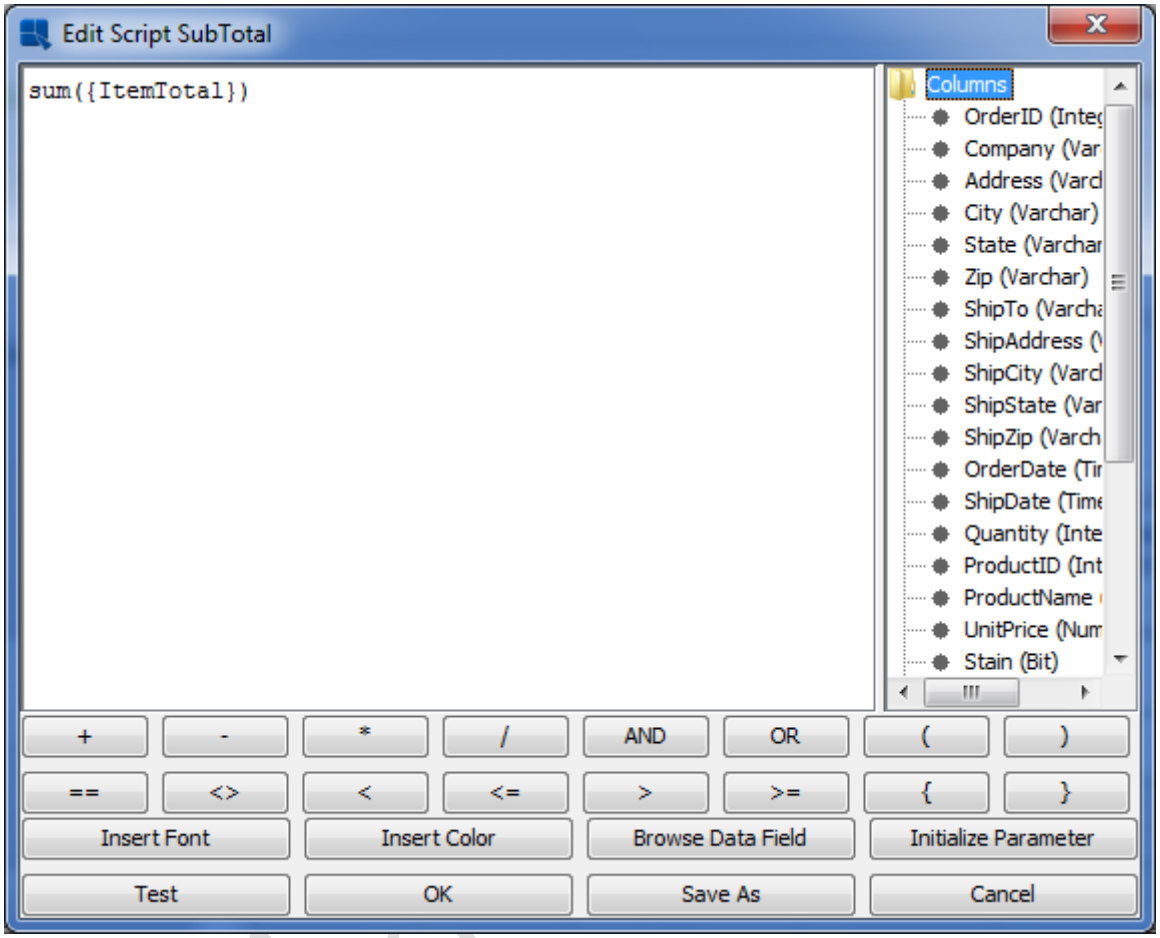
2.3.5.3 集計を追加する

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

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

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

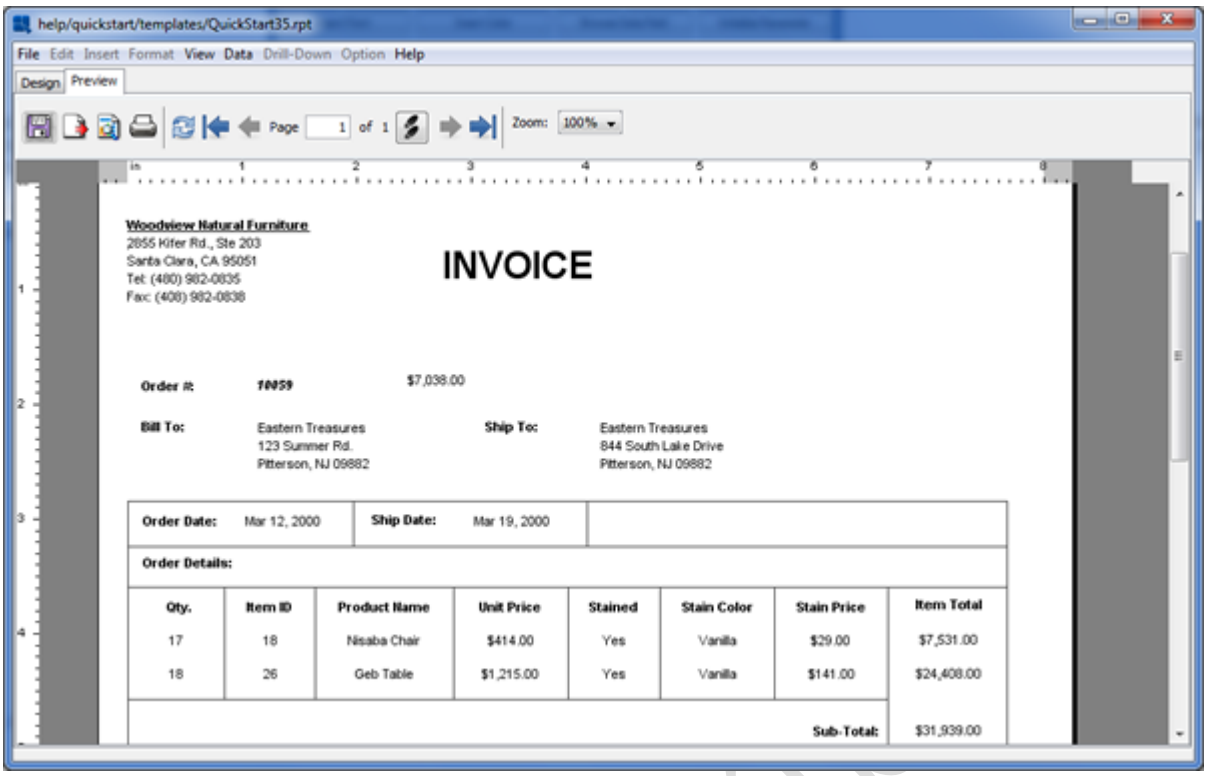
sum({ItemTotal})



集計式を含む数式ビルダー

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

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



数式によるレポート

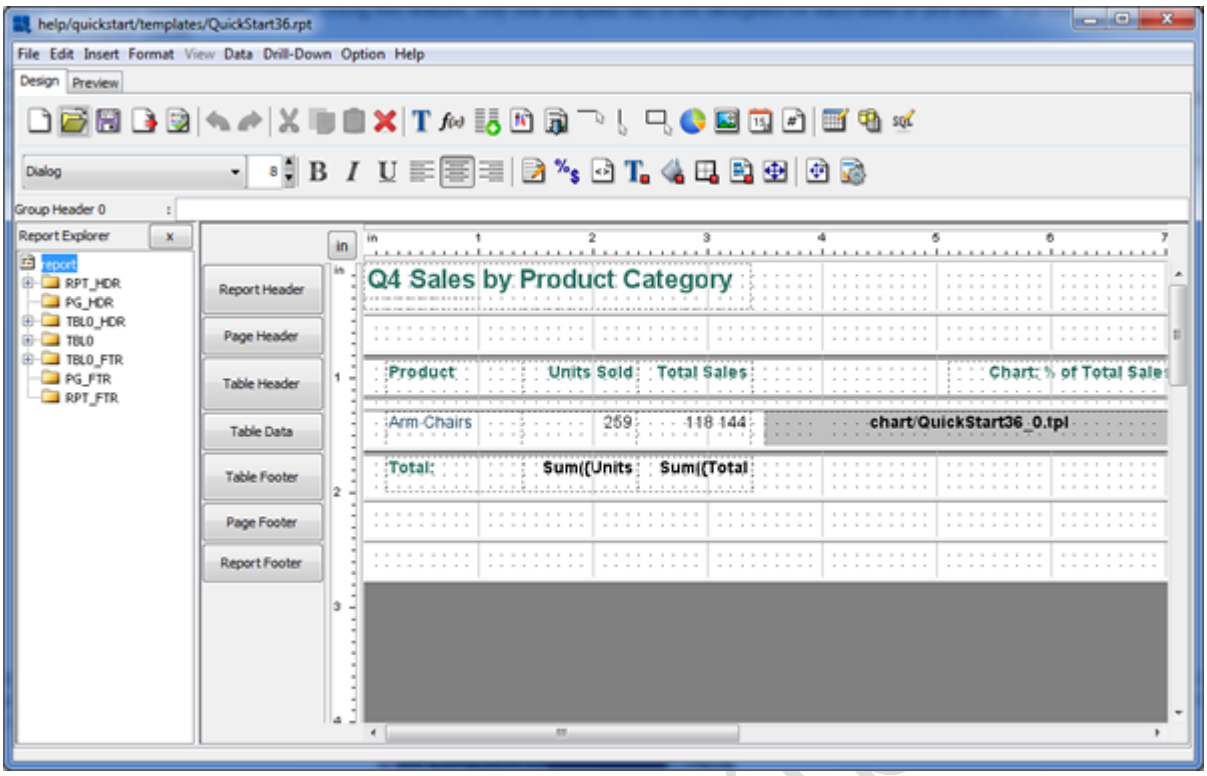
追加のチュートリアルでは、2 つの数式を Group Footer セクションに追加します。1 つは売上税を計算するもの、もう 1 つは注文の総計を計算するものです。

2.3.6 ドリルダウン

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

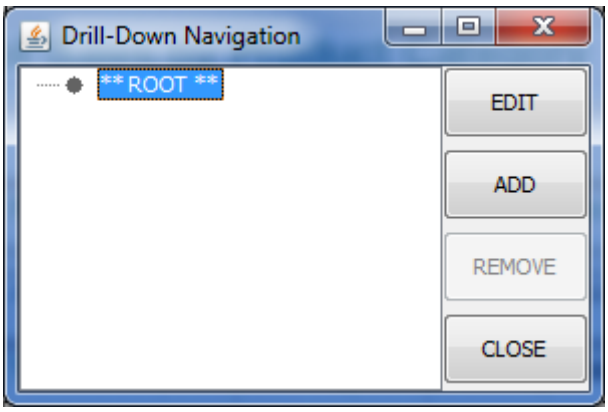
ドリルダウン機能はパラメータ化されたクエリを使用するため、このチュートリアルを実行するには、Woodview データベースをセットアップする必要があります。

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



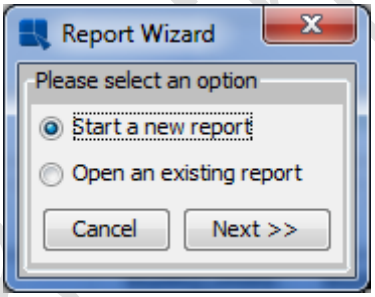
デザインウィンドウで QuickStart36.rpt

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



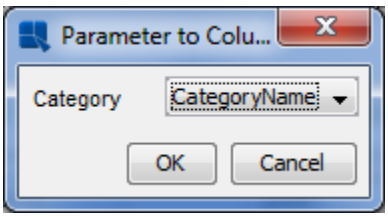
Drill-Down Navigation ダイアログ

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



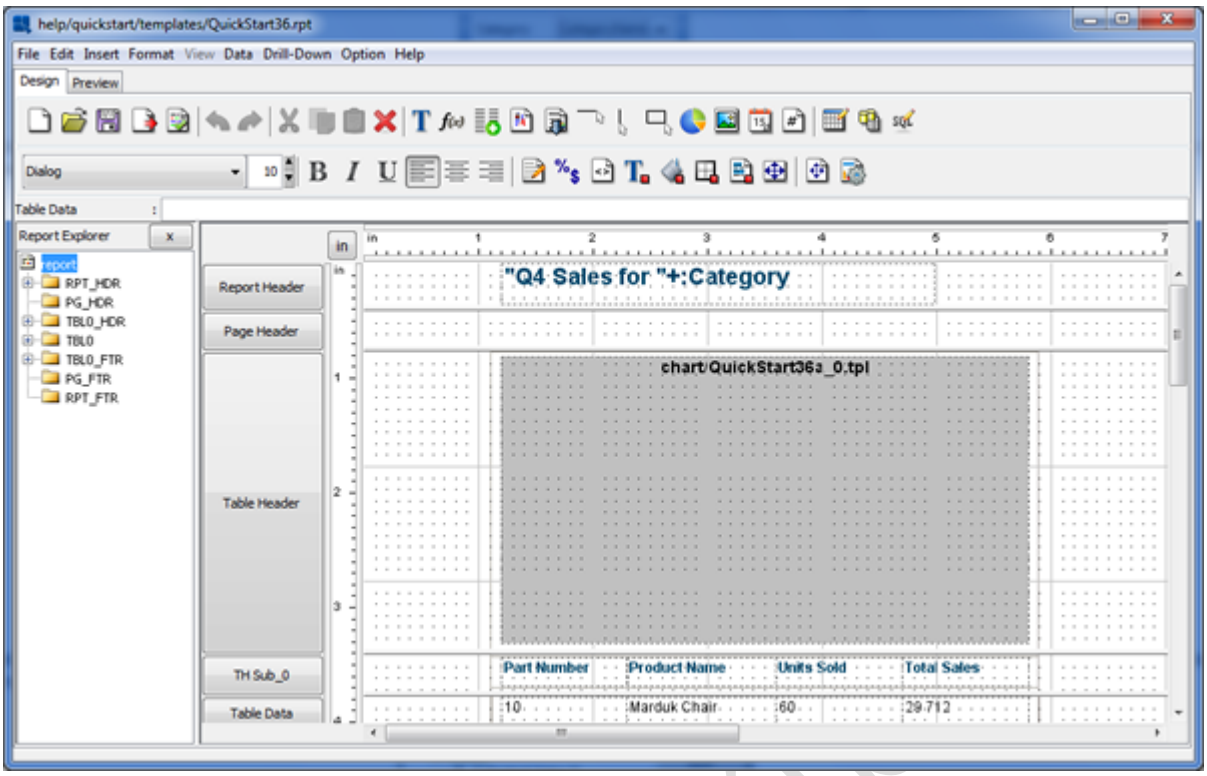
Report Options ダイアログ

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



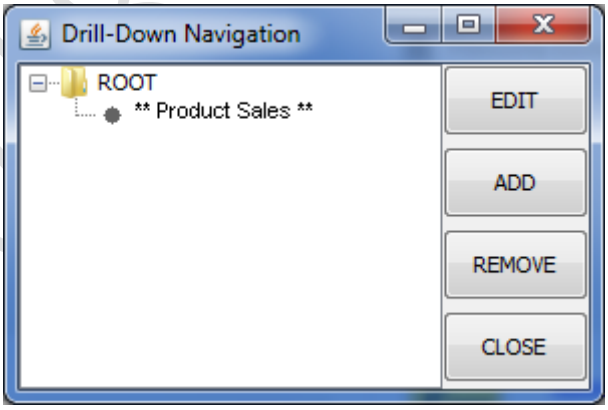
Column - Parameter Mapping ダイアログ

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



Report Designer の最初のドリルダウンレベル

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

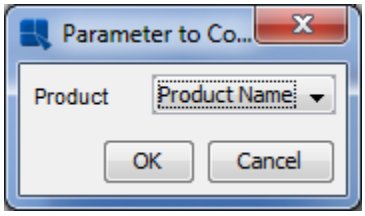


追加レイヤーを含む Drill-Down Navigation ダイアログ

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

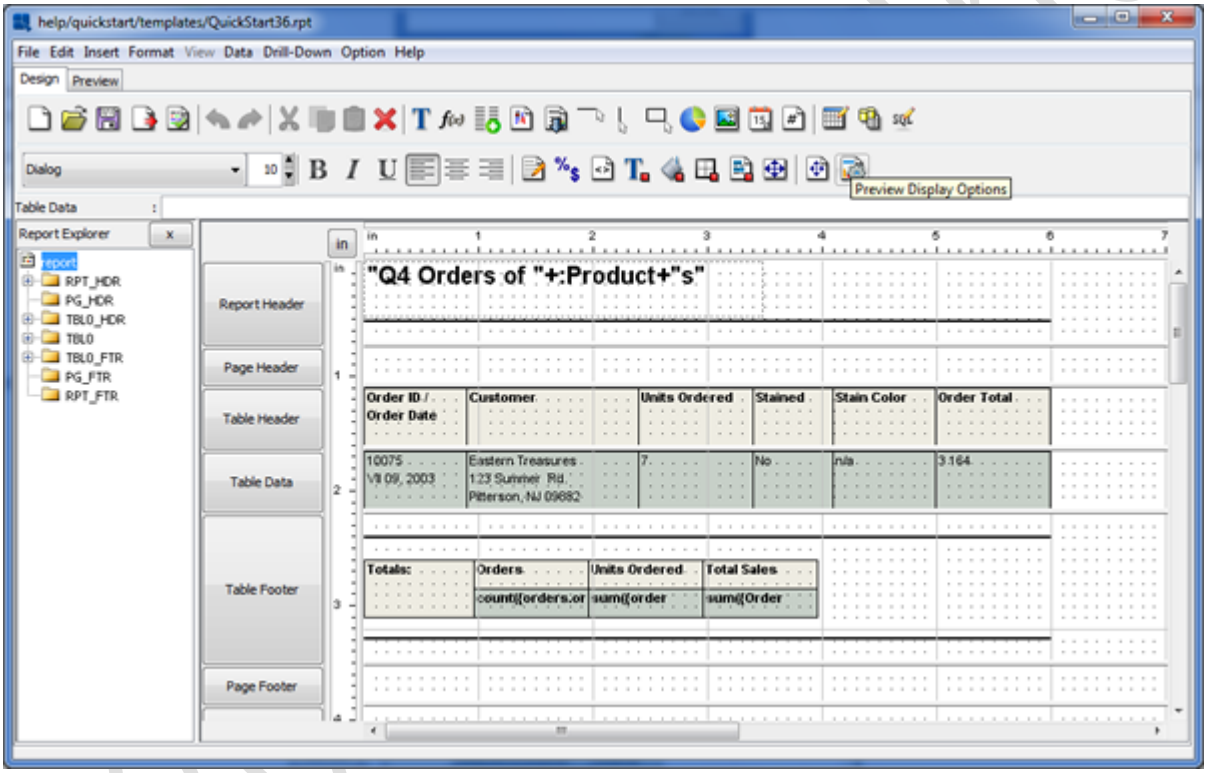
再度、新しいドリルダウンレイヤーの既存のテンプレートを開く場合に選択します。ダイアログで、**<InstallDir>/help/quickstart/templates** を参照し、QuickStart36b.rpt ファイルを選択します。(または/ Access /ディレクトリからの QuickStart36b_Acc.rpt)。これにより、列マッピングダイアログへ

のパラメータが表示されます。



Second Parameter Mapping ダイアログ

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

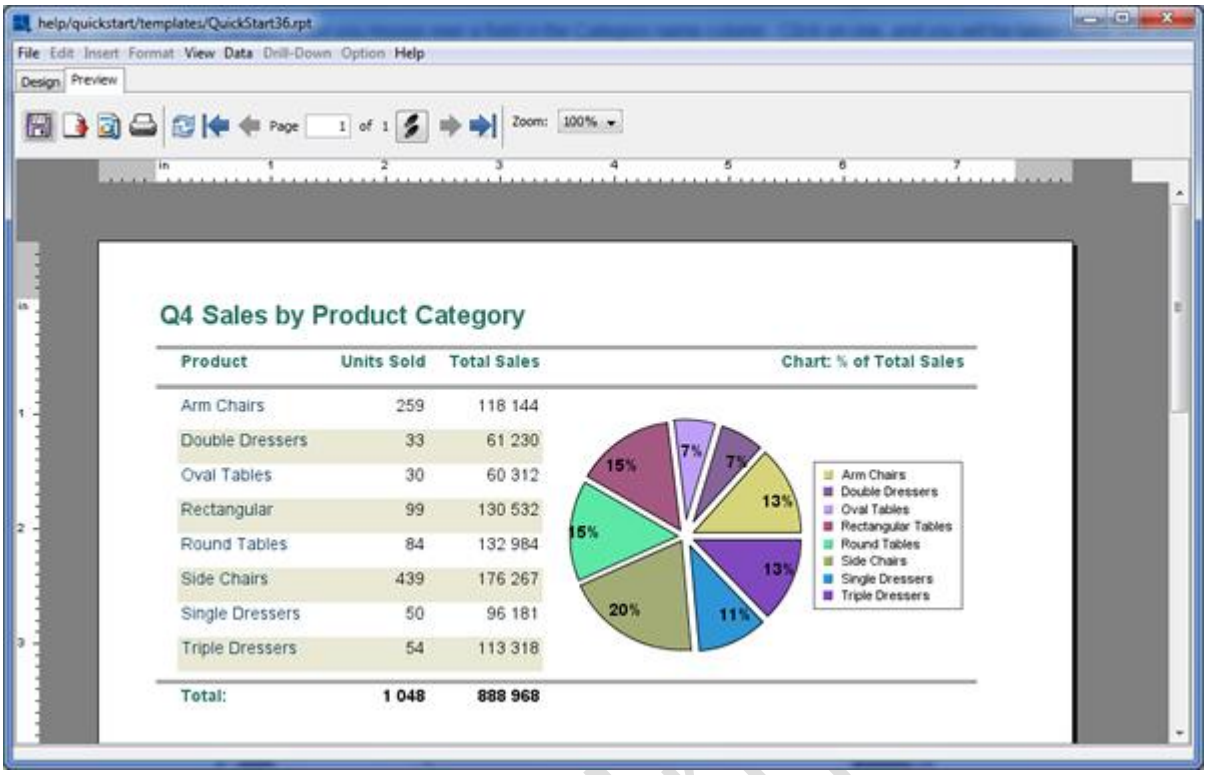


Report Designer の 2 番目のドリルダウンレベル

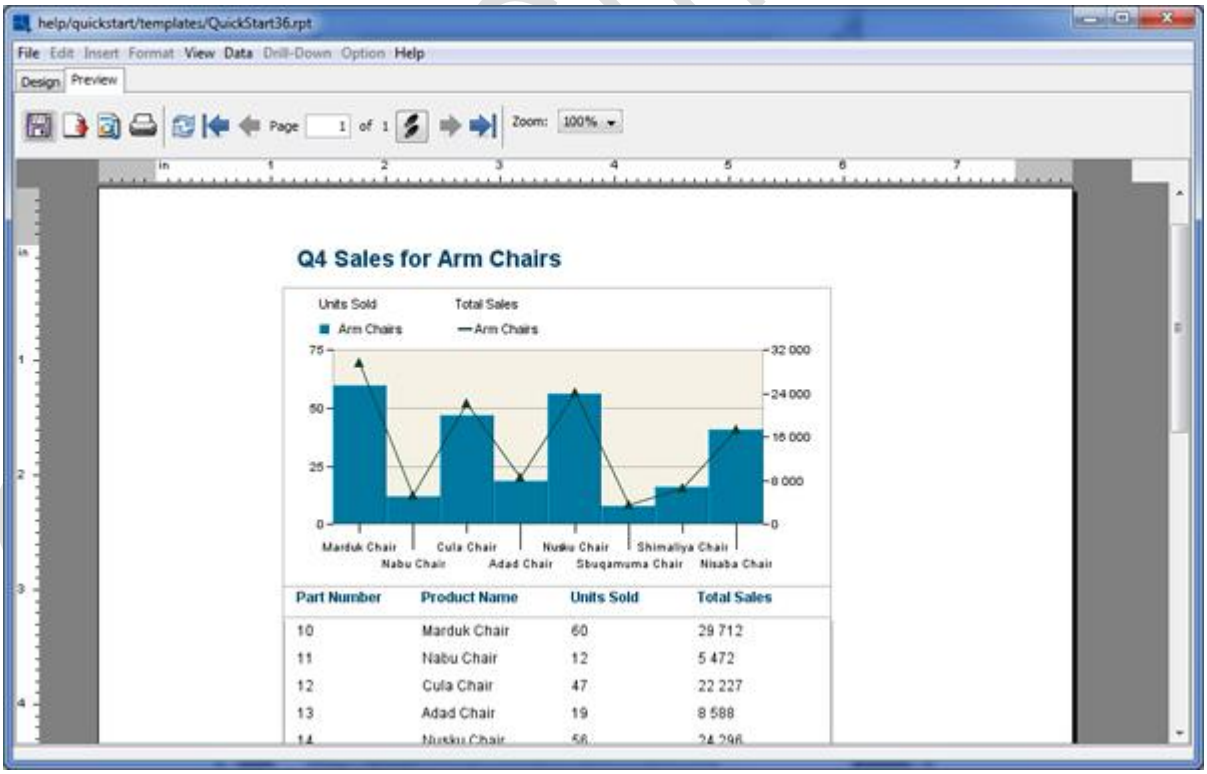
次に、**Drill-Down** > **Navigate** を選択します。今追加したレイヤーの新しいノードがあることに注目してください。ルートレポートを選択し、**EDIT** ボタンをクリックしてデザインウィンドウで開きます。最初のレポートがデザイナで表示されます。次に、**CLOSE** をクリックして、ナビゲーションダイアログを閉じます。

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

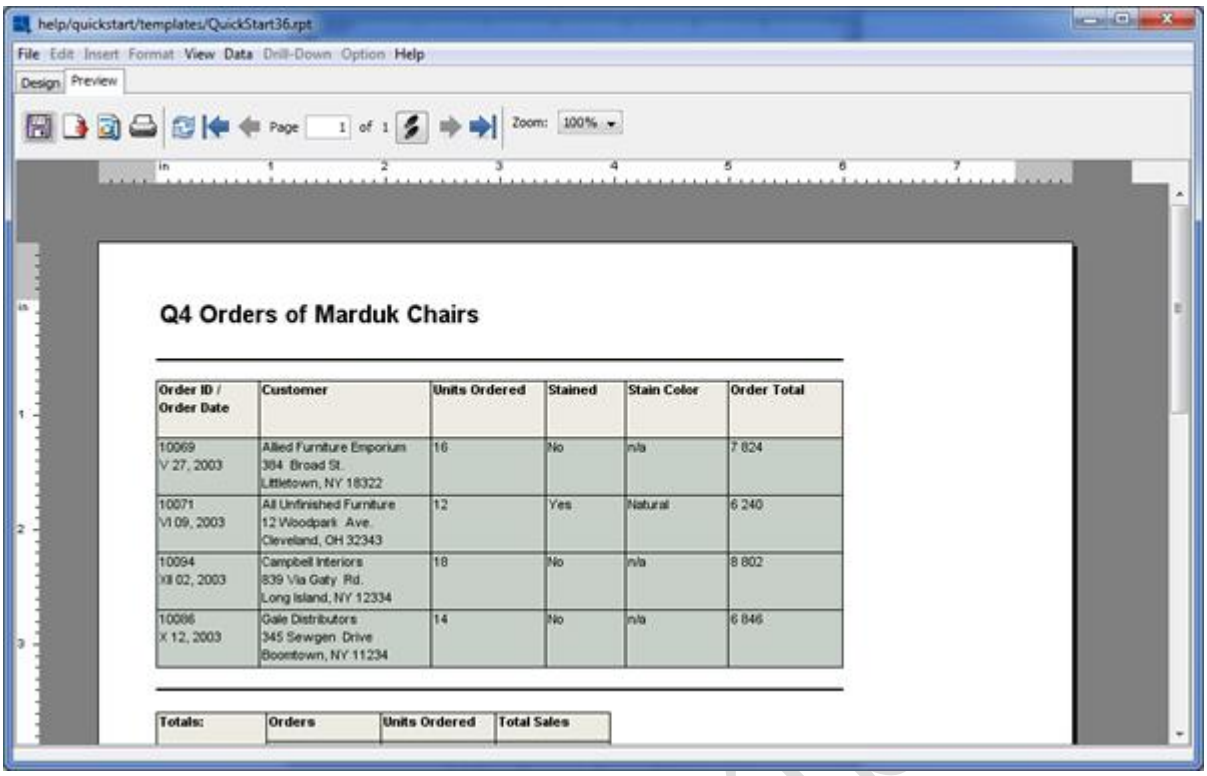
製品レポートでは、**Product name** 列のフィールドをクリックして、特定の製品の注文を示す第 3 レベルのレポートに移動できます。



ドリルダウンレベル 1



ドリルダウンレベル 1



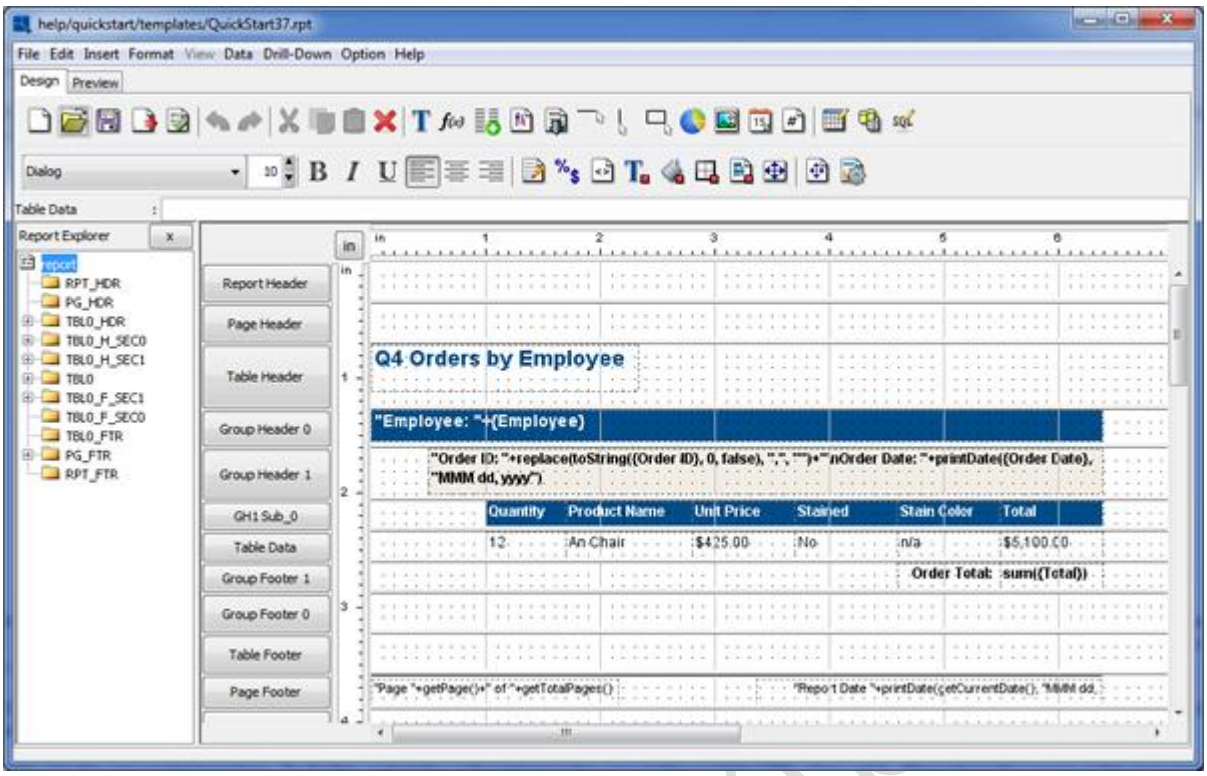
ドリルダウンレベル 1

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

2.3.7 サブレポート

EspressReport のもう一つの強力な機能は、サブレポートを使用してより複雑なレポートレイアウトを作成し、レポート内の複数のソースからのデータを結合する機能です。

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

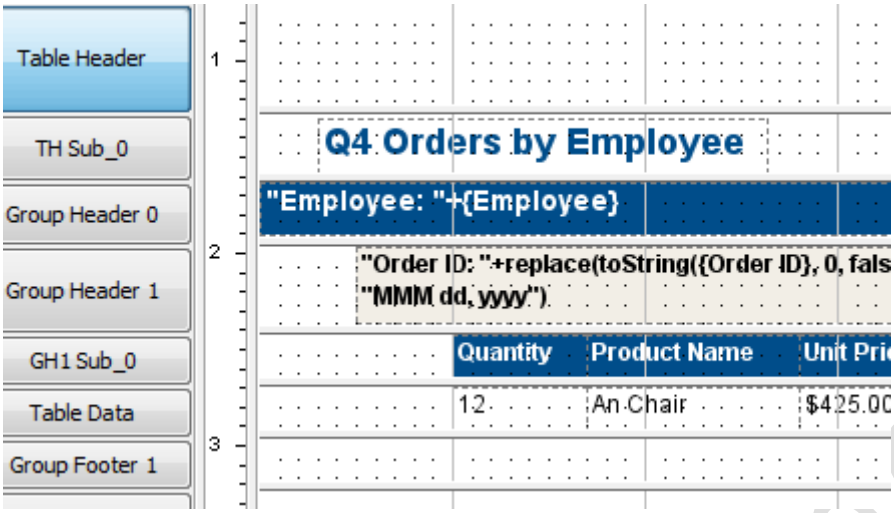


デザイナの QuickStart37.rpt

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

ネストされたセクションを挿入するには、左側の **Table Header** ボタンをクリックして、セクションオプションメニューを表示します。**Insert Section.**を選択します。これは、テーブルヘッダーのネストされたセクションを生成します。ネ

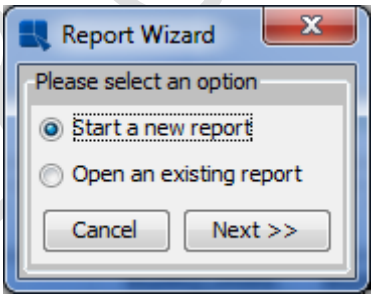
次に、タイトルを新しいネストされたセクションに移動します。これを行うには、レポートタイトルを含むセルを選択し、**CTRL + X** を押してカットします。次に、新しいセクションにカーソルを置き、**CTRL + V** を押して貼り付けます。カーソルが十字に変わります。フィールドを希望する場所に配置し、クリックして追加します。セルに合わせて新しいセクションのサイズを変更することができます。



新しいレポートセクション

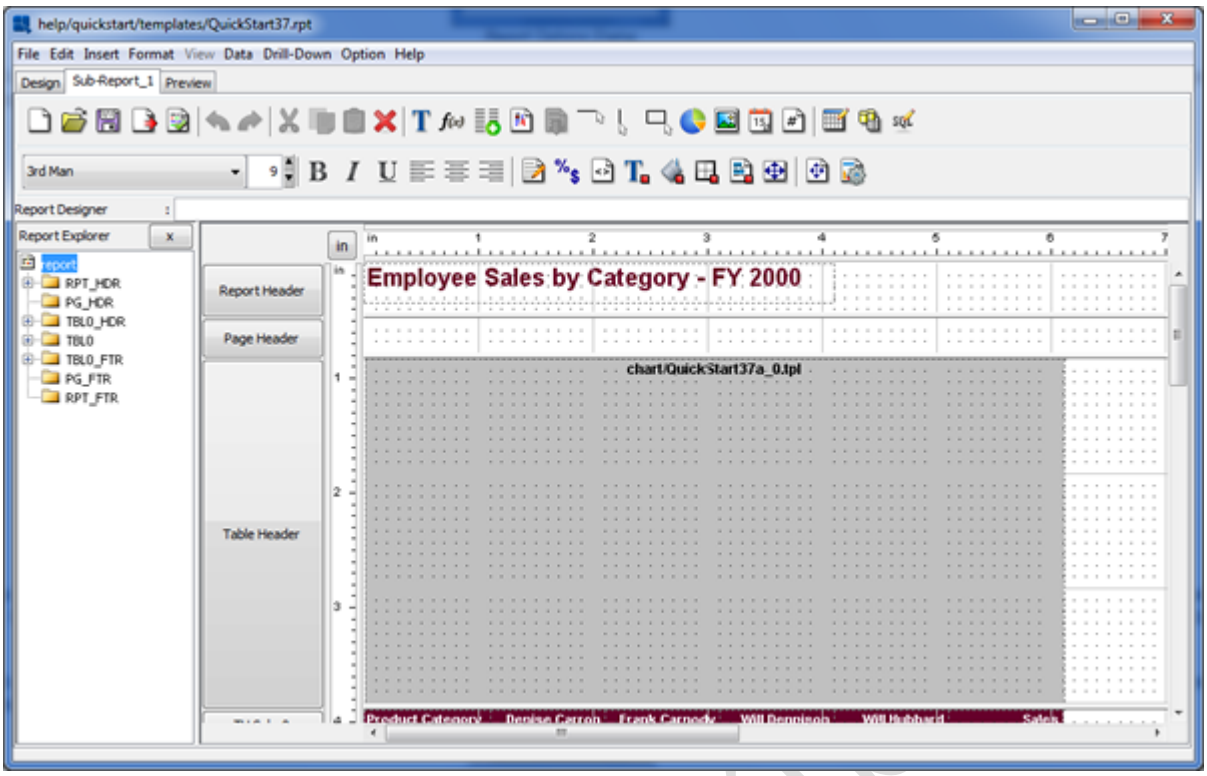
次に、ツールバーの **Insert Sub-Report** ボタン  をクリックします。続行する前に、変更をレポートに保存するように求められます。これを行うと、マウスポインタが十字に変わります。SubReport を挿入するには、Table Header セクションの左上隅にあるマウスの左ボタンを押します。

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



Report Options ダイアログ

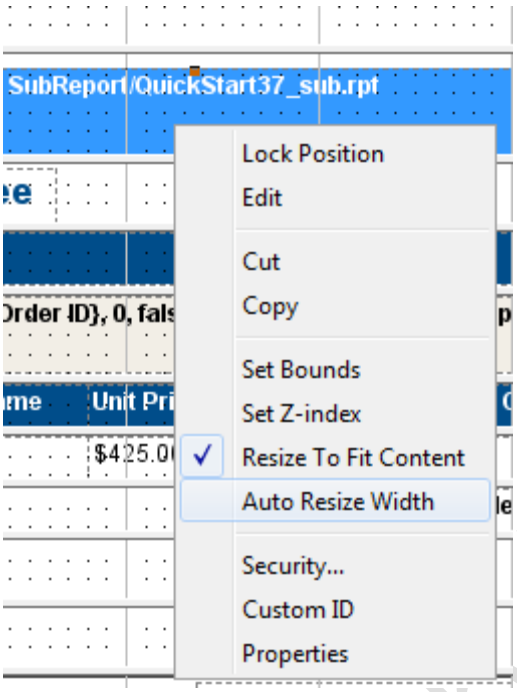
プロンプトが表示されたら、**<InstallDir>/help/quickstart/templates** ディレクトリを参照します。そこに QuickStart37a.rpt ファイルを選択し、**Open** をクリックします。サブレポートは、Report Designer の Sub-Report_1 という新しいタブで開きます。



Designer のサブレポート

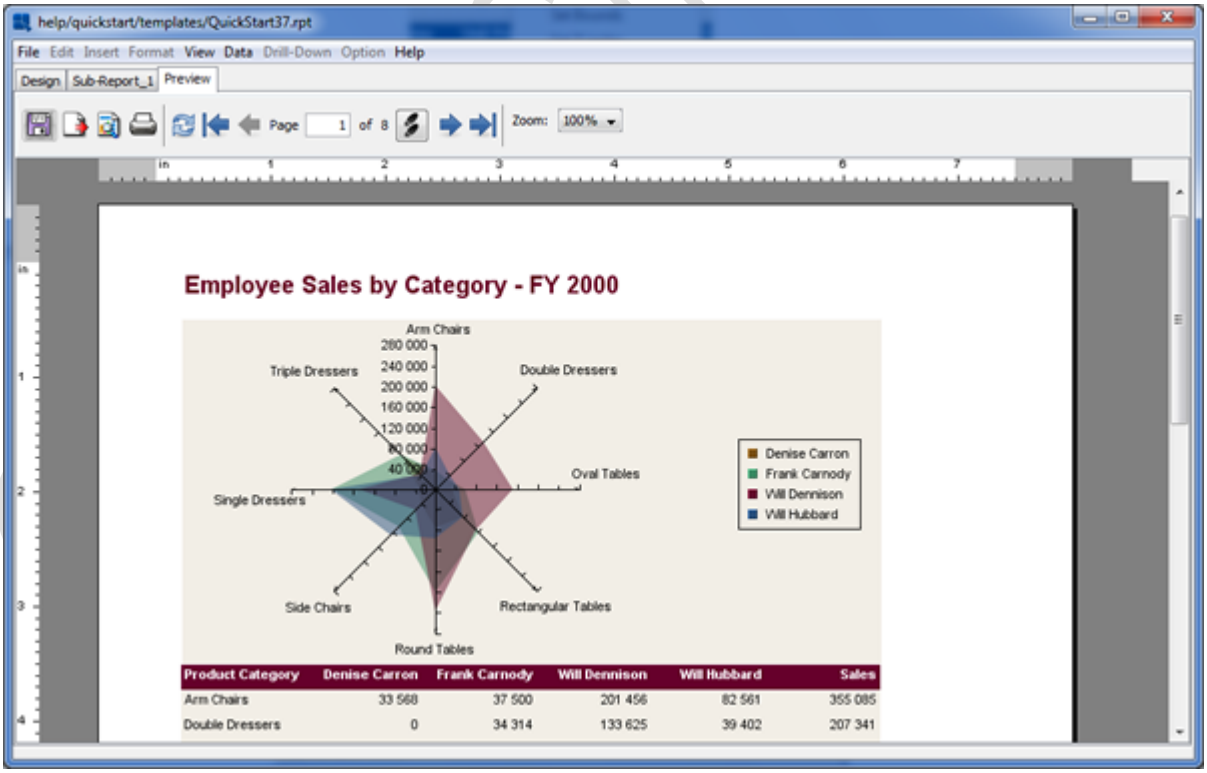
Preview タブと **Sub-Repor** タブを切り替えることで、サブレポートだけをプレビューできます。次に、**Design** タブをクリックしてメインレポートに戻ります。サブレポートは小さな灰色の四角形で表示されます。四角形をセクションの左上隅に移動し、横のルーラーをクリックしてドラッグすると、サブレポートの幅が約 7 インチに広がります。

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

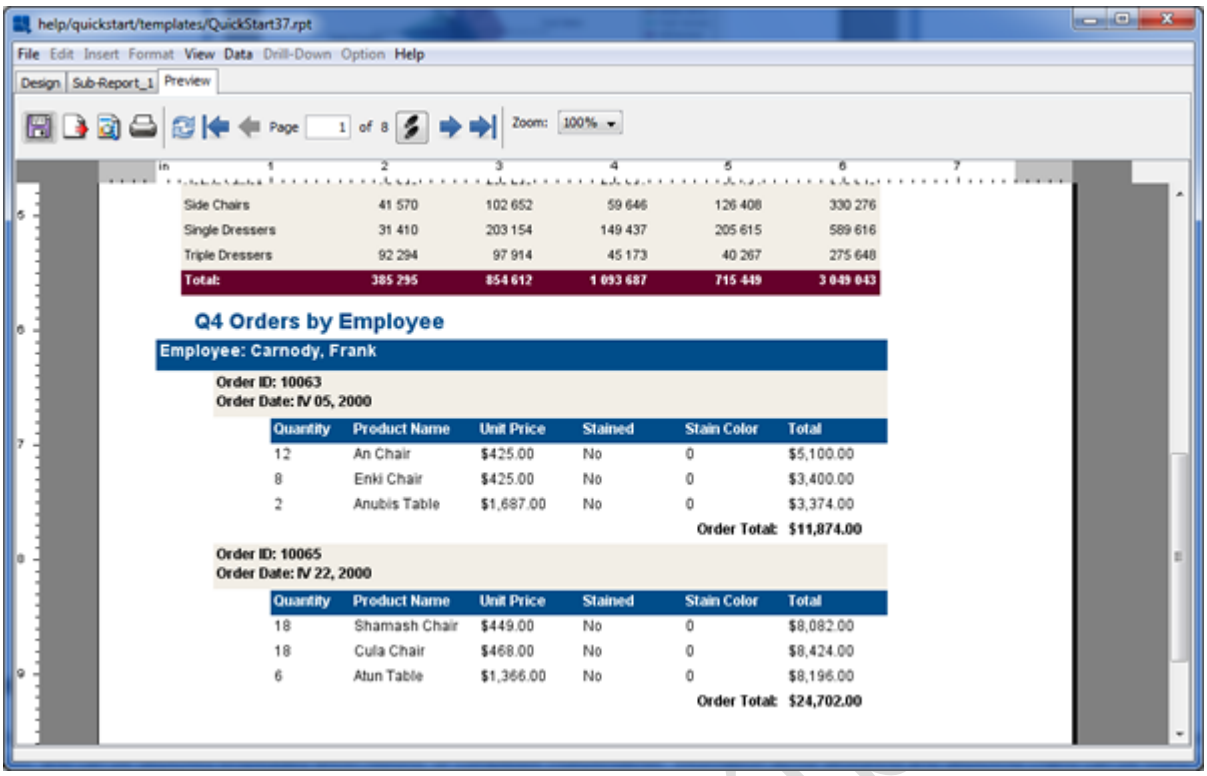


Resize to Fit Content ダイアログ

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



プレビューのメインレポート

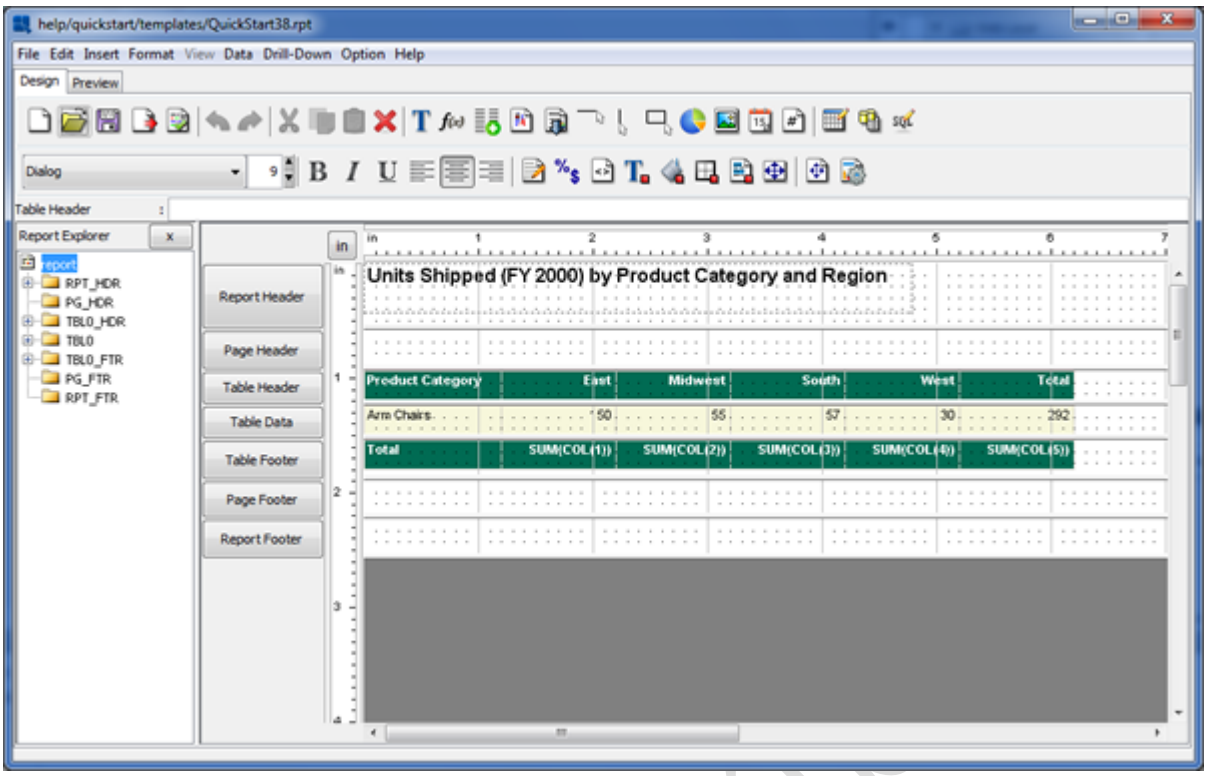


プレビューでメインレポートを含むサブレポート

2.3.8 チャートの操作

EspressReport には、30 以上の異なる 2 次元および 3 次元チャートタイプでデータをプロットするための広範囲なチャートライブラリが含まれています。チャートは独立したデータソースを使用することができ、レポート内に配置/埋め込み、または完全に独立して配置できます。

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



デザインウィンドウで QuickStart38.rpt

2.3.8.1 グラフと地図データを挿入する

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

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

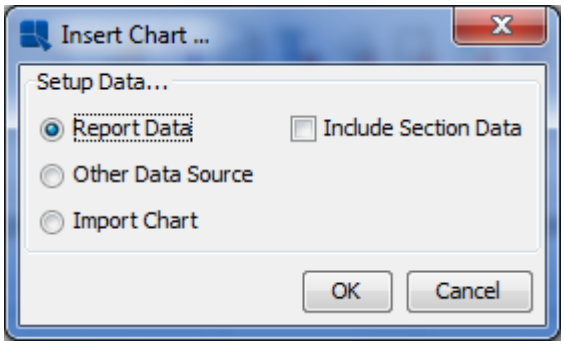


Chart Data Options

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

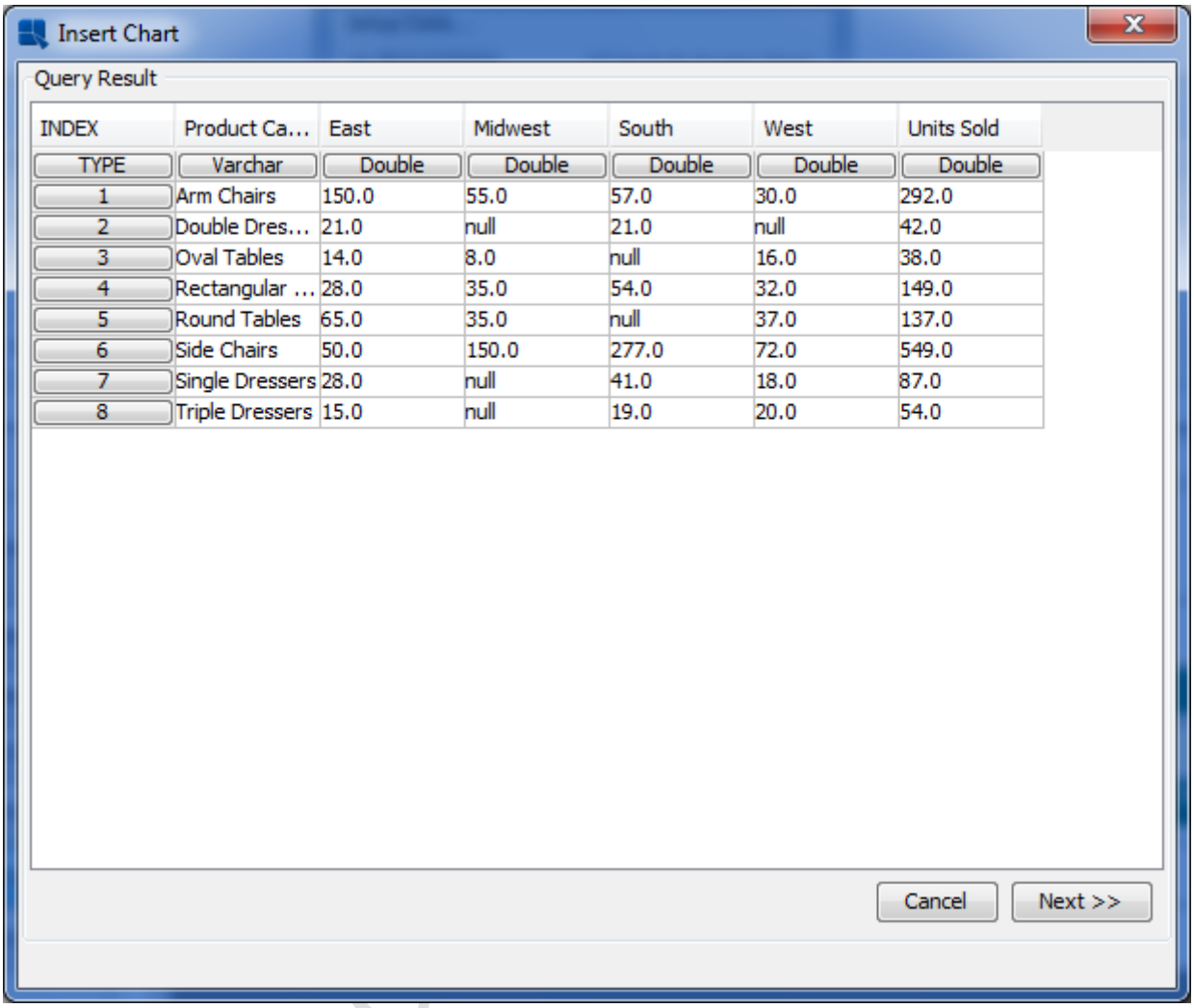


Chart Data ダイアログ

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

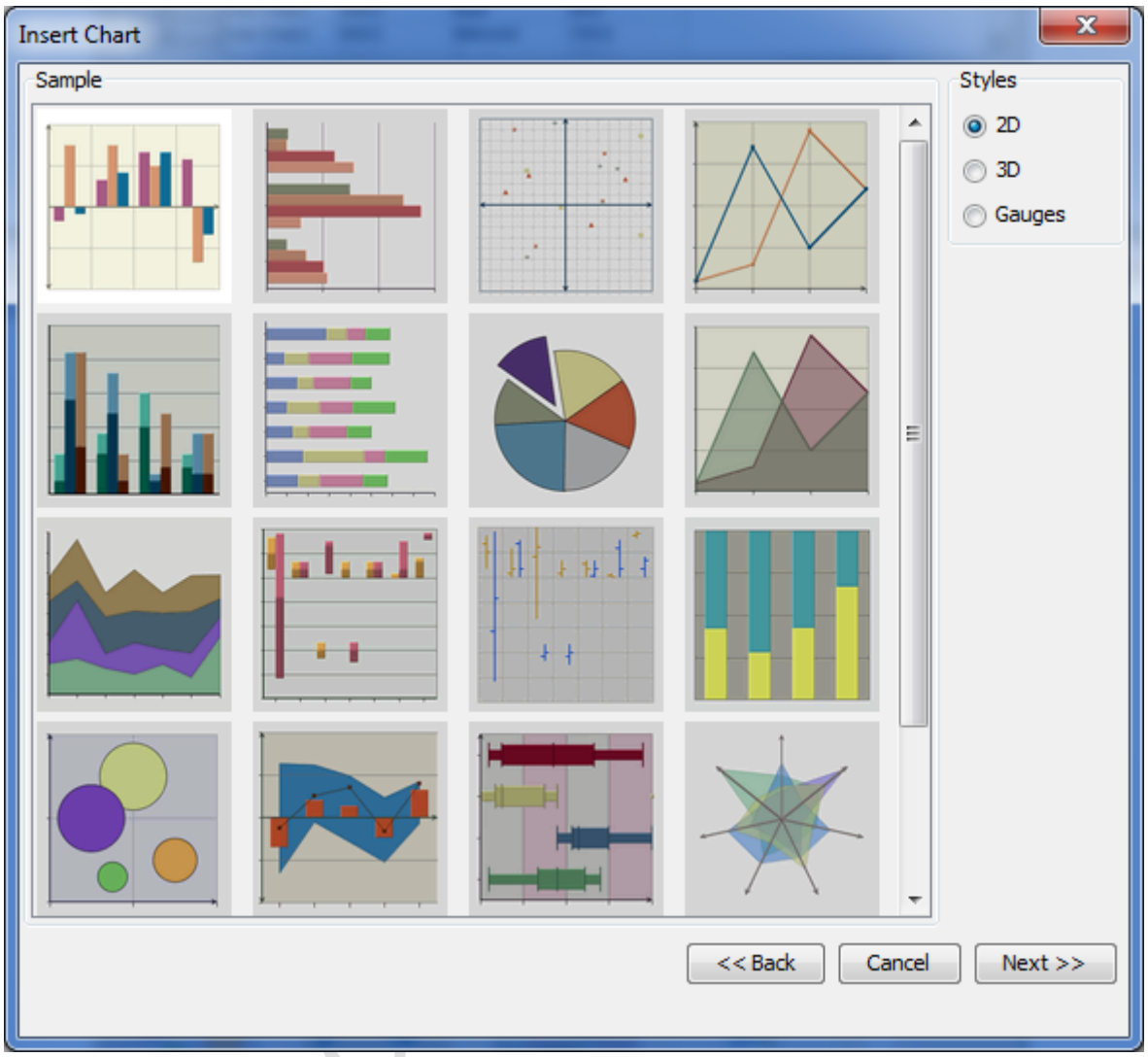
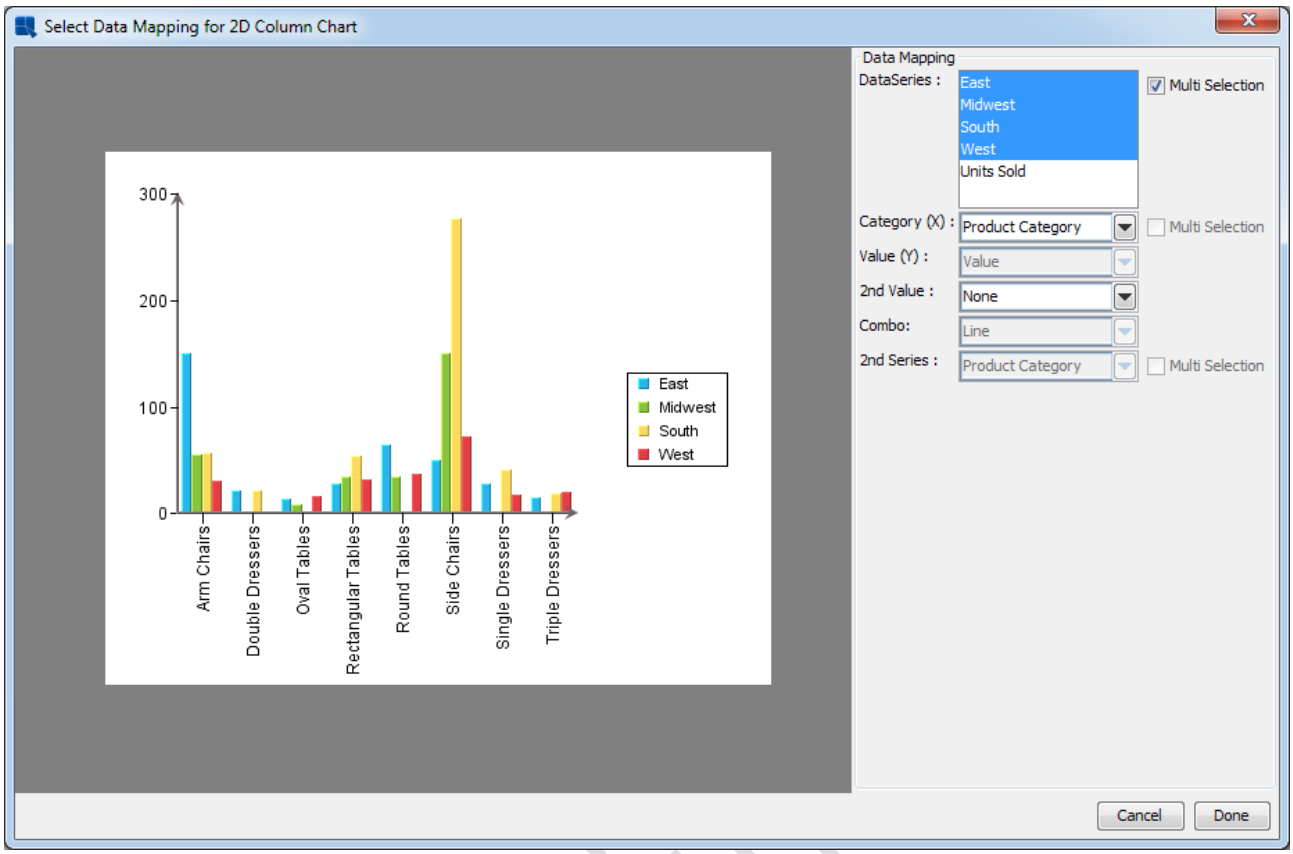


Chart Types ダイアログ

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

East
Midwest
South
West

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



Data Mapping ダイアログ

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

2.3.8.2 チャートのプロパティをカスタマイズする

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

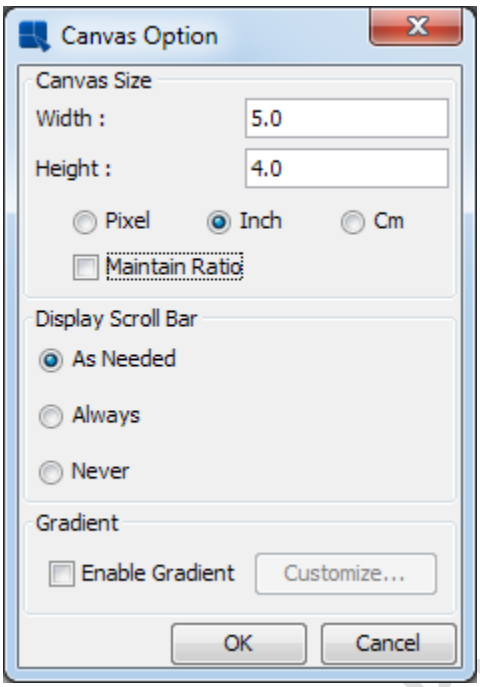
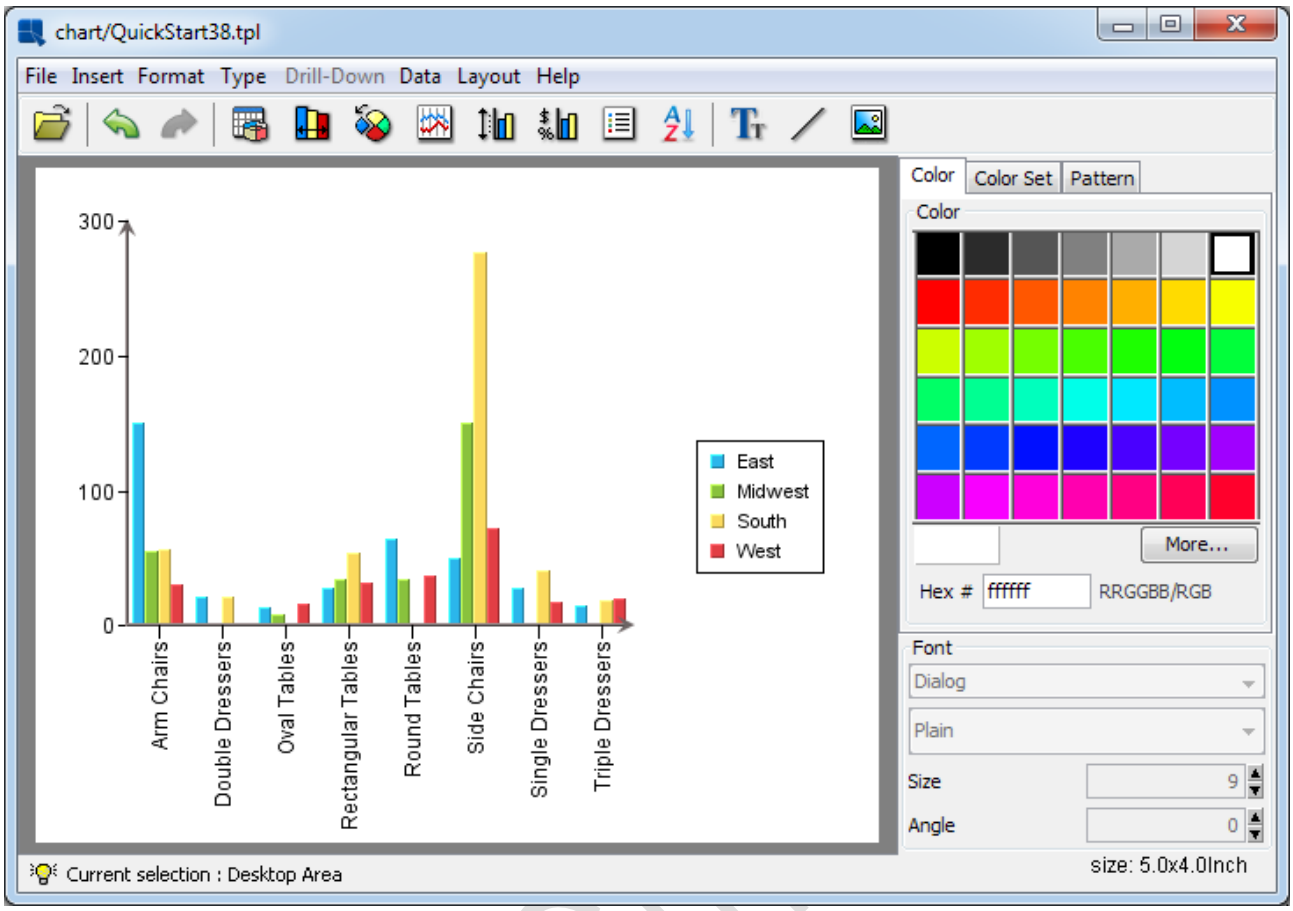


Chart Canvas ダイアログ

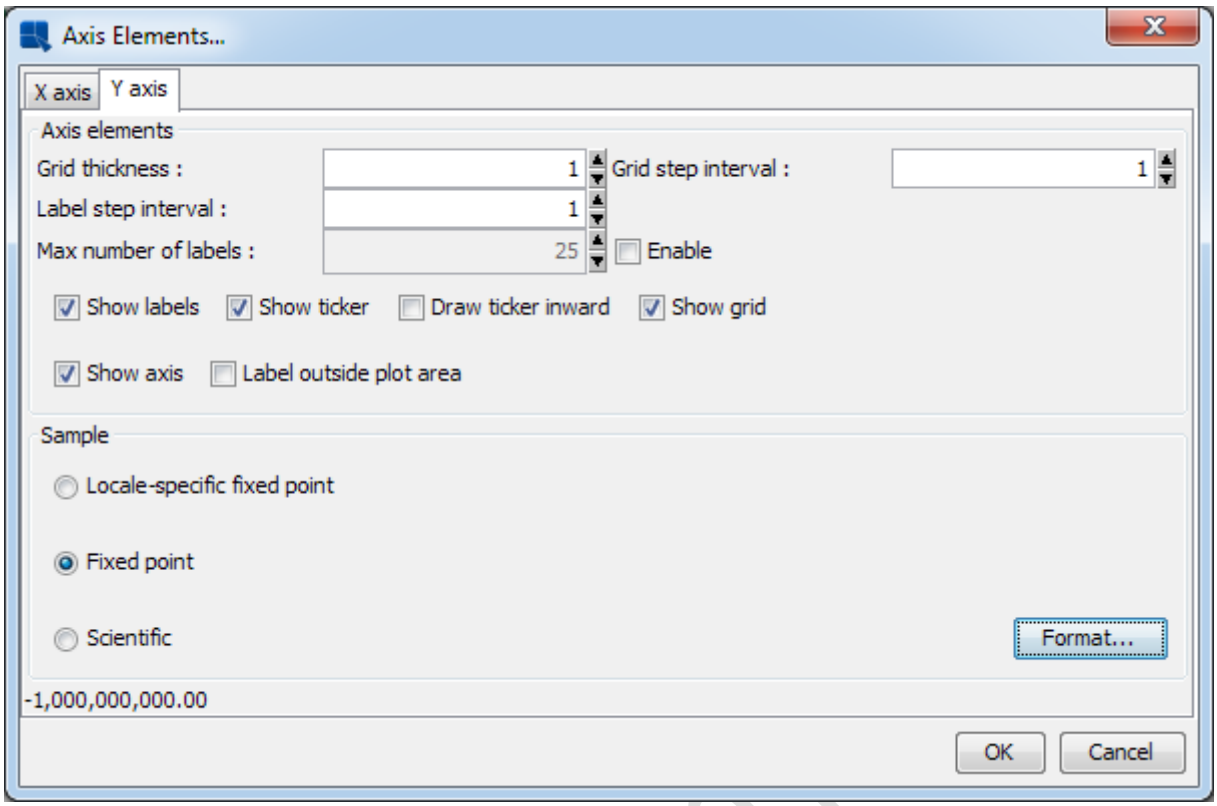
ボタンを選択して測定値をインチで指定し、**Maintain Ratio** オプションの選択を解除し、新しいチャートキャンバスサイズを 5x4 インチに設定します。**OK** をクリックします。これにより、チャートデザイナのビュー部分でキャンバスのサイズが変更されます。

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



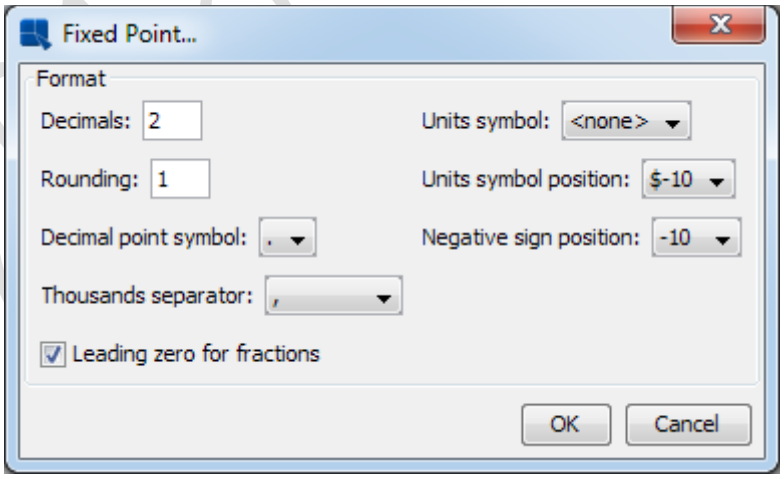
配置されたチャートと凡例を持つチャートデザイナ

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



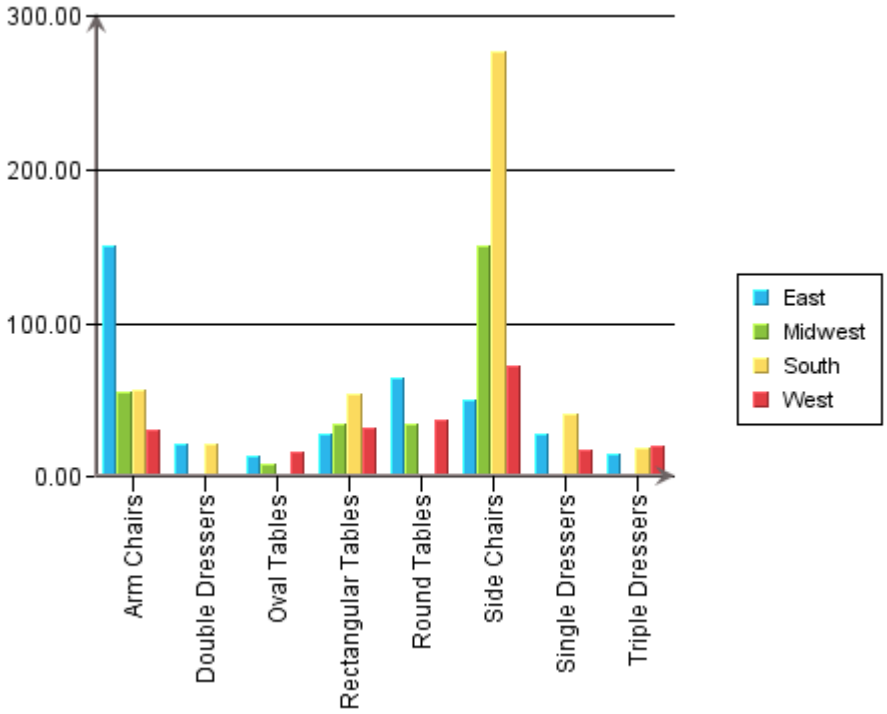
Axis Elements ダイアログ

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



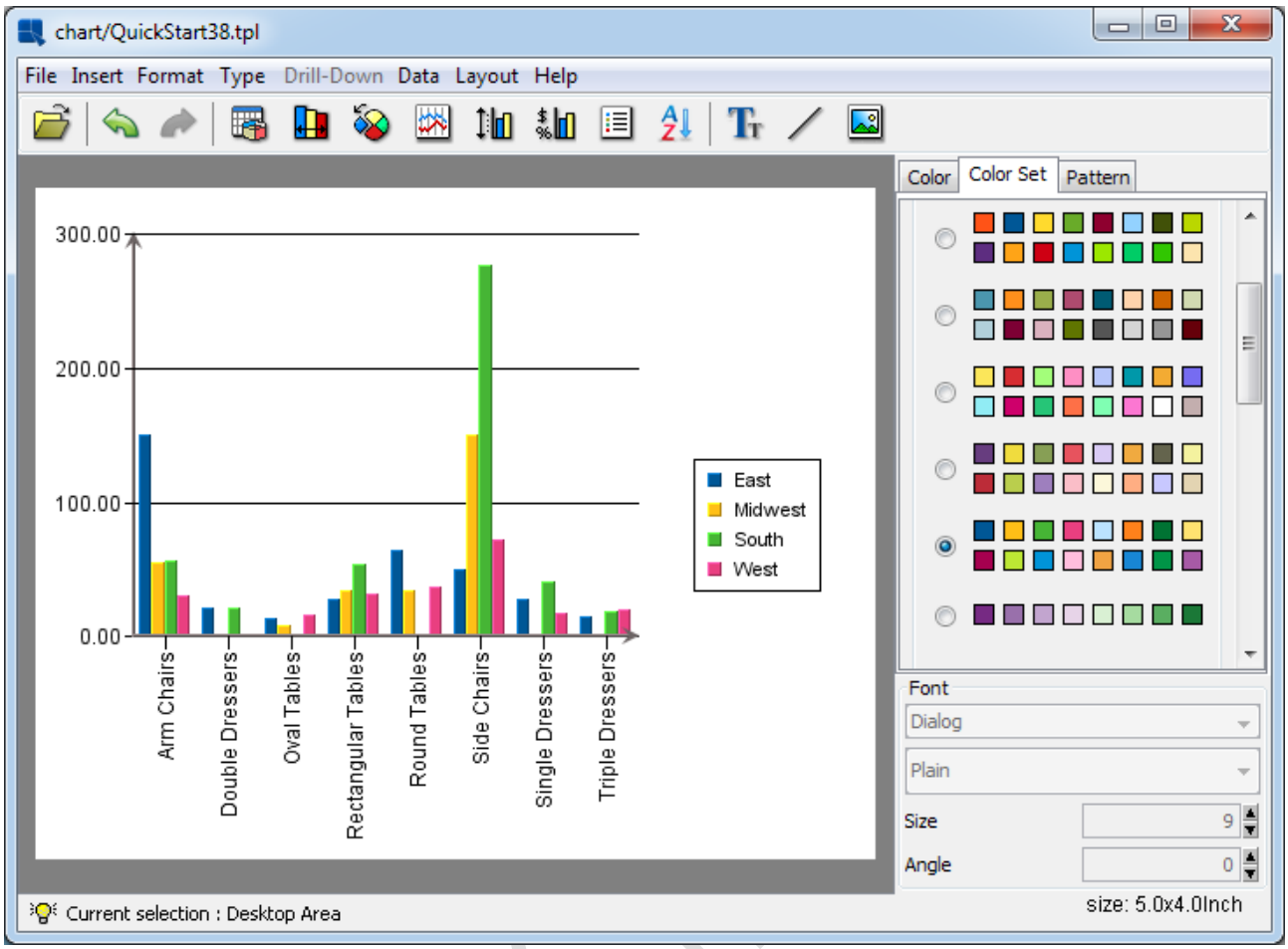
Value Axis Fixed Point Formatting ダイアログ

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



軸要素の書式設定後のグラフ

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



カラーセットをピックする

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

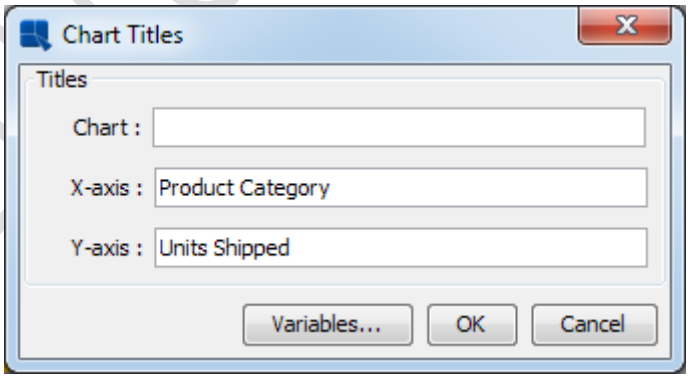
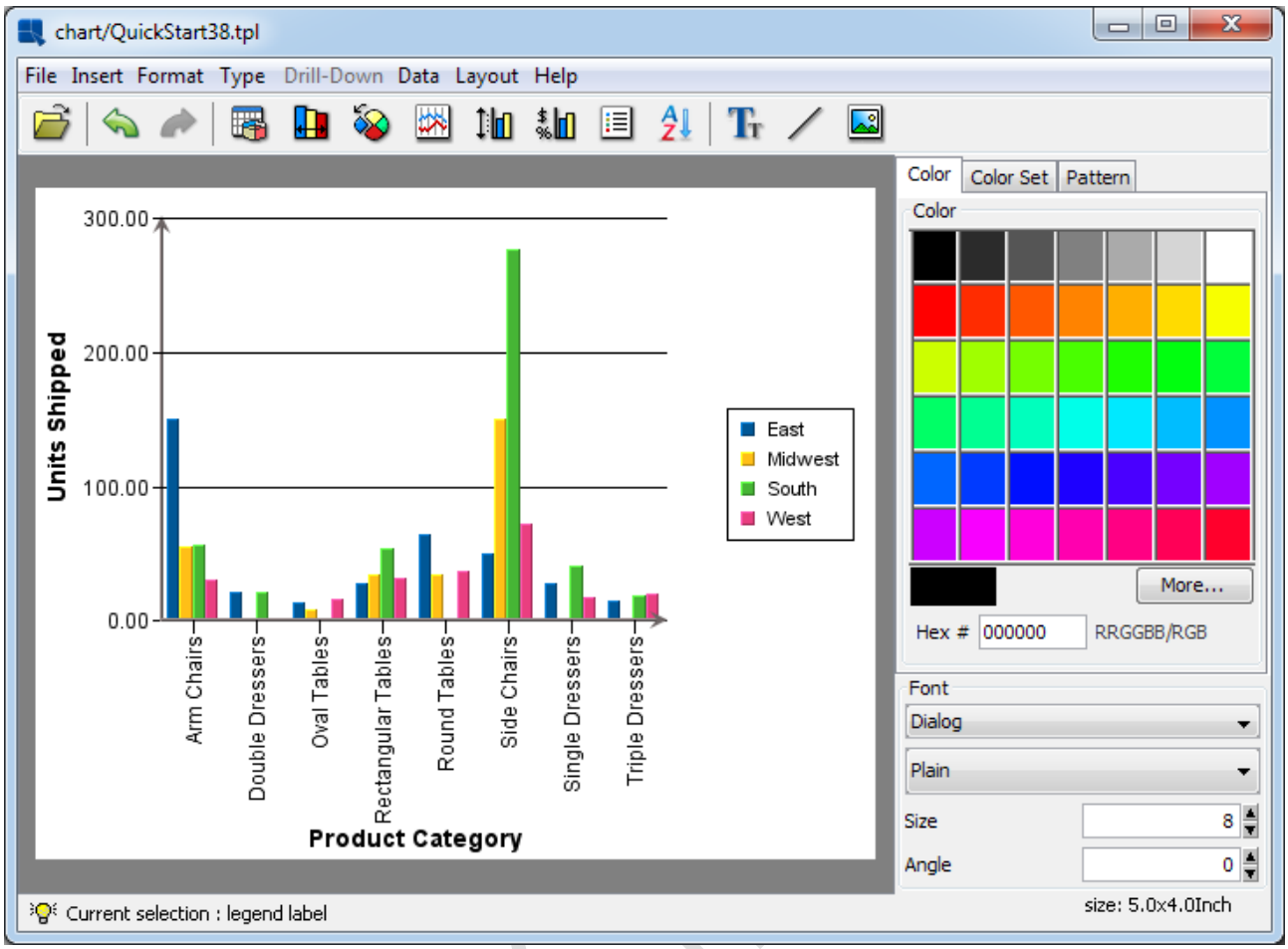


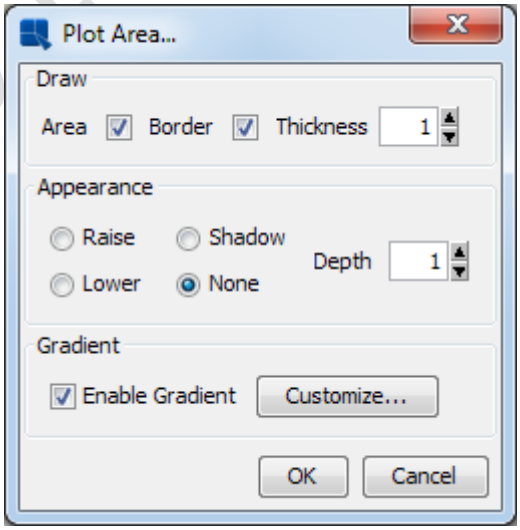
Chart Titles ダイアログ

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



軸のタイトルがあるチャート

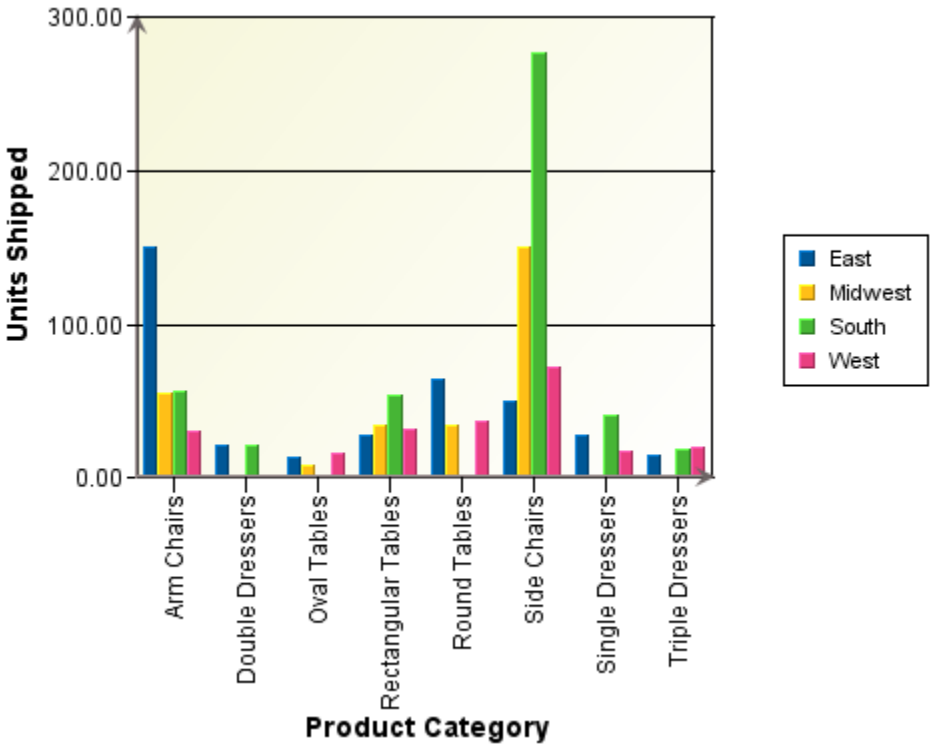
最後に、プロットの背景と境界線を追加して、プロット領域の外観をカスタマイズすることができます。これを行うには、**Format** > **Plot Area** を選択します。これにより、グラフプロットの表示オプションを設定できるダイアログが表示されます。



Plot Area ダイアログ

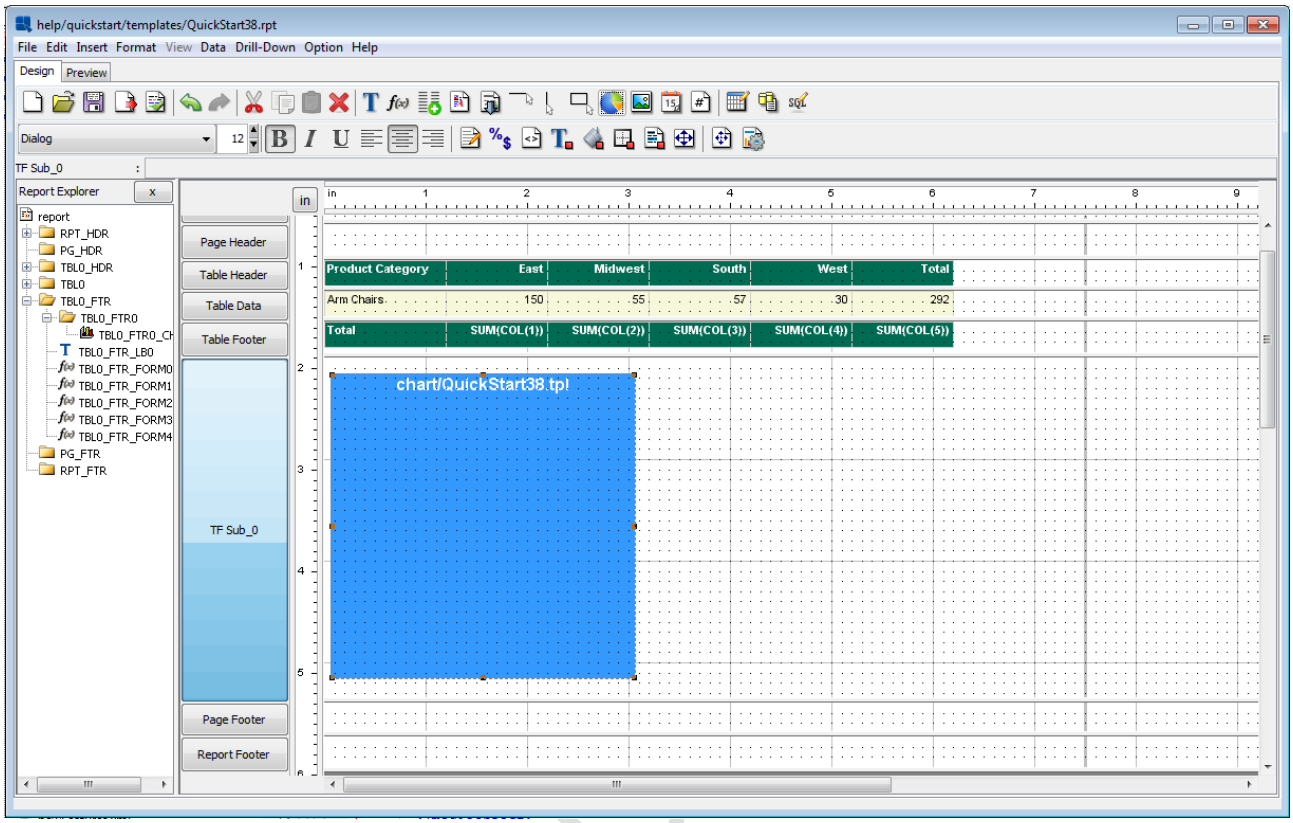
プロット領域と境界線の両方を 1 の厚さで描画する場合に選択します。外観に **None** を指定します。グラデーションを有効にするオプションを選択します。すべてのオプションを指定したら、**OK** をクリッ

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



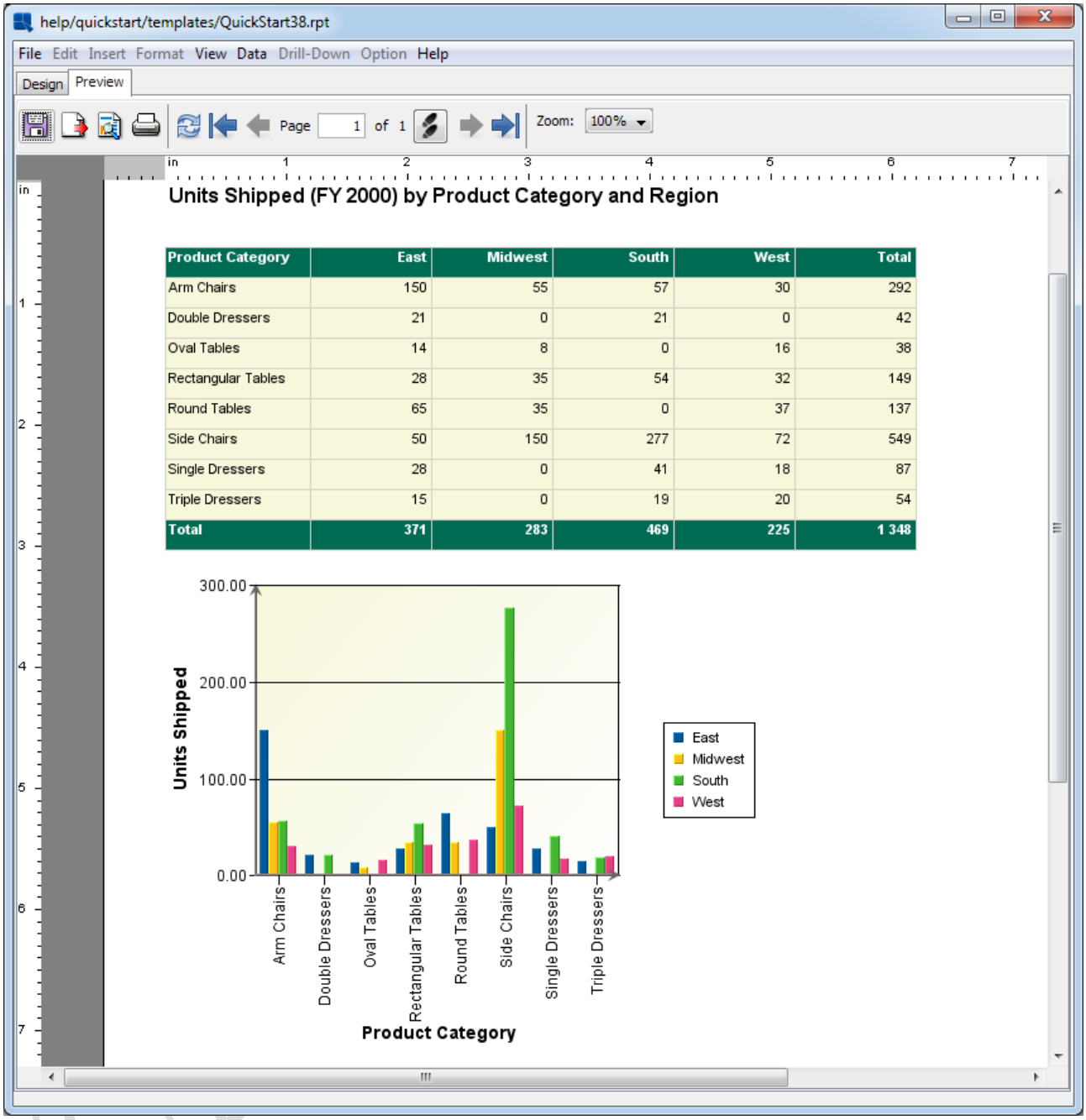
プロットエリアを含むチャート

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



Report Designer のグラフ

レポートをプレビューします。表の下にグラフが表示されます。



チャートを使用したレポートのプレビュー

2.4 API のクイックスタート

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

2.4.1 環境を設定する

このガイドに記載されているコードは、次の点を考慮して作成されています：

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

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

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

クイックスタート API ガイドを正しく使用するには：

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

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

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

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

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

かつ

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

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

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

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

コード実行中に、「Failed to read espressmanager.cfg. Use default host and port no. 22071.（espressmanager.cfg を読み込めませんでした。デフォルトホストとポート番号 22071 を使用してく

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

EspressReport API は EspressManager なしで、他のアプリケーションサーバでも使用できますが、このガイドのコードは EspressManager と Tomcat を念頭に置いて設計されています。

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

2.4.2 レポートを実行する

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

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

2.4.2.1 応用

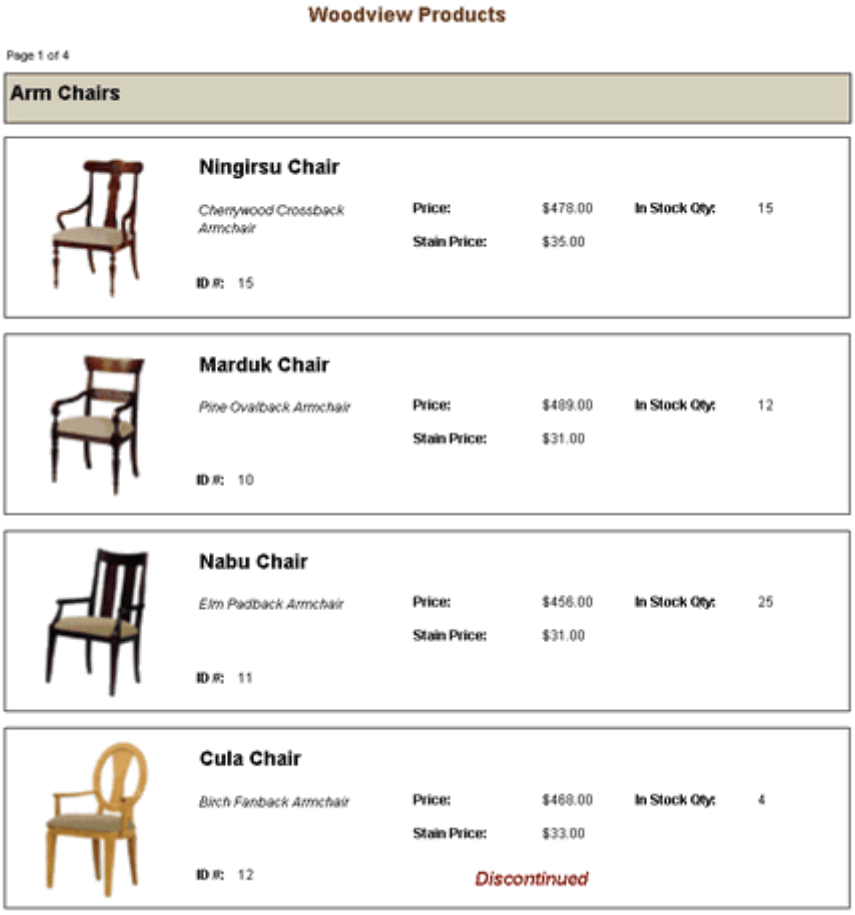
次のコードは、既存のレポートテンプレート（この場合は QuickStart42.rpt）をアプリケーションに表示する方法を示しています。

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

public class QuickStart421 extends Applet {

        public static void main(java.lang.String[] args) {
                try {
                        QuickStart421 doReport = new QuickStart421();
```

```
                    Frame frame = new Frame();
                    frame.setLayout(new BorderLayout());
                    frame.add("Center", doReport.doQuickStart421(frame));
                    frame.setSize(600, 600);
                    frame.setVisible(true);
                } catch (Exception ex) {
                    ex.printStackTrace();
                }
        }

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

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

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

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



生成されたレポート

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

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

2.4.2.2 Java Web Start アプリケーション（JNLP）

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

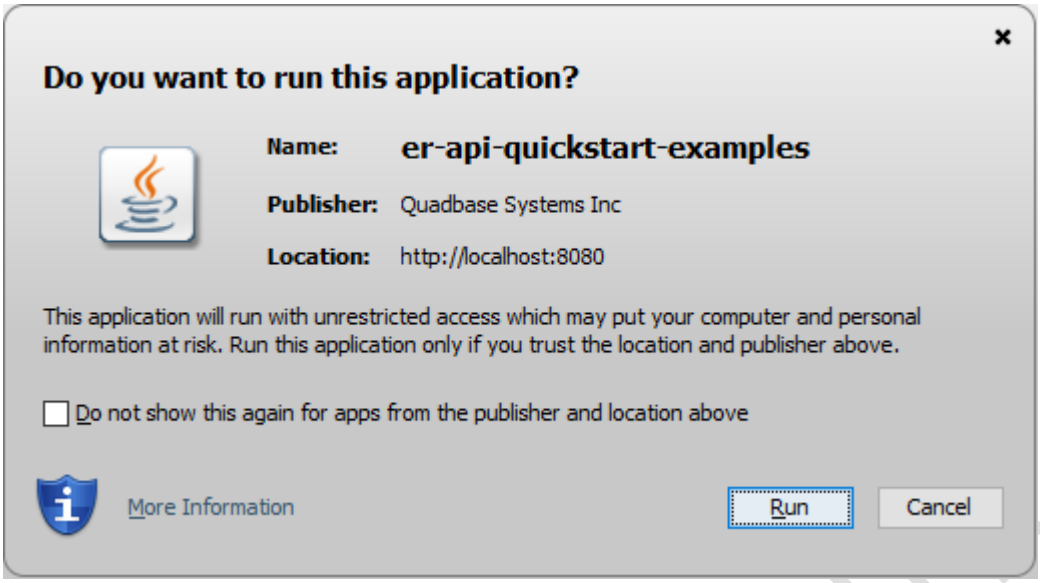
```
<?xml version="1.0" encoding="utf-8"?>
<jnlp spec="1.5" codebase="http://localhost:8080/" href="QuickStart422.jnlp">
    <information>
        <title>QuickStart422</title>
        <vendor>Quadbase Systems Inc.</vendor>
        <description>QuickStart422 JNLP</description>
        <offline-allowed/>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

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

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

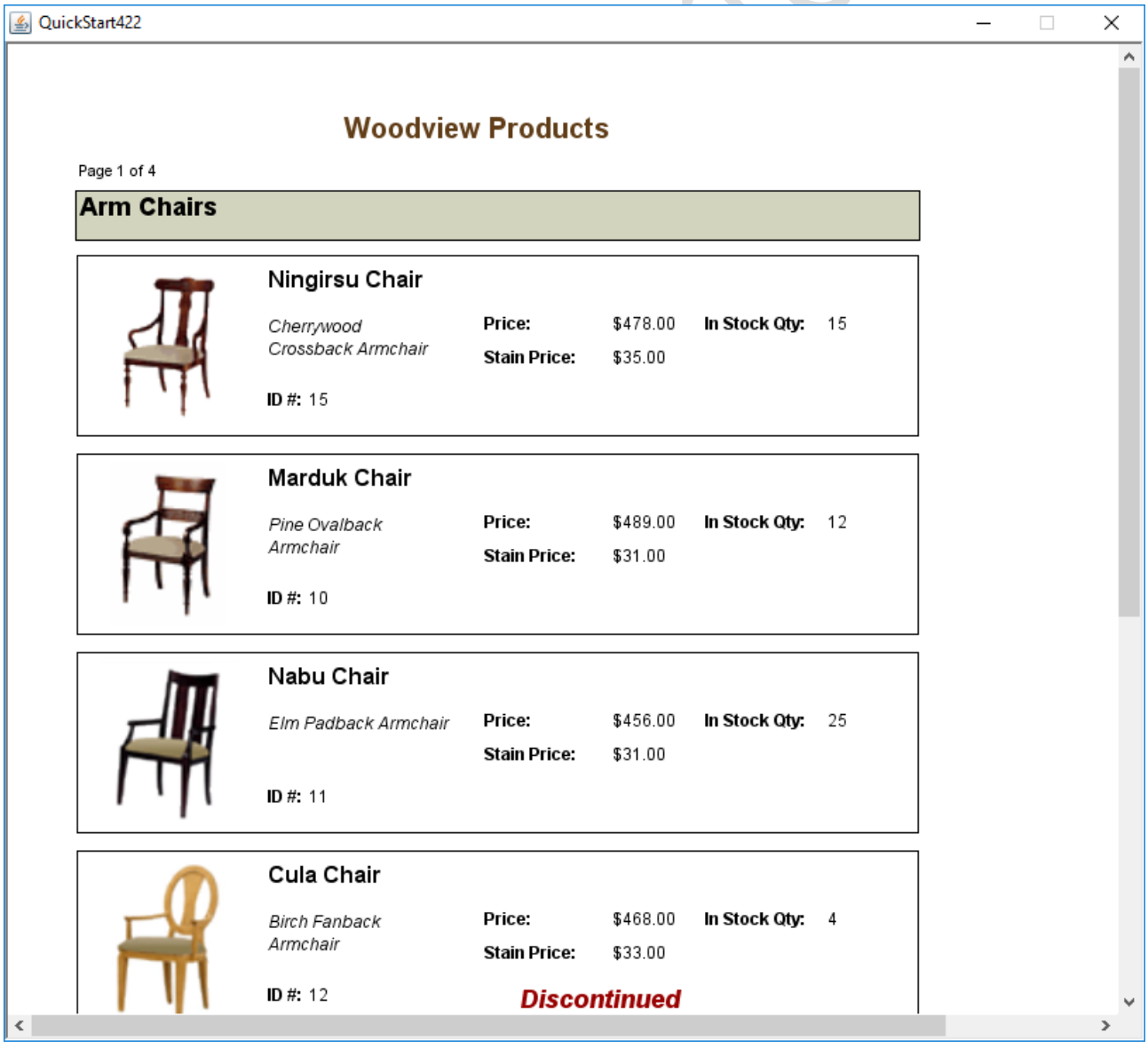
- **QuickStart.jar** を**<EspressReport InstallDir>/help/quickstart/classes** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT** ディレクトリにコピーします。
- **ReportAPIWithChart.jar** と **qblicense.jar** を**<EspressReport InstallDir>/lib** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT** ディレクトリにコピーします。
- **QuickStart422.jnlp**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

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



JNLP を実行するかどうか

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



生成されたレポート

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

2.4.2.3 サーブレット

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

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

public class QuickStart423 extends HttpServlet {

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

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

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

サーブレットをデプロイするには：

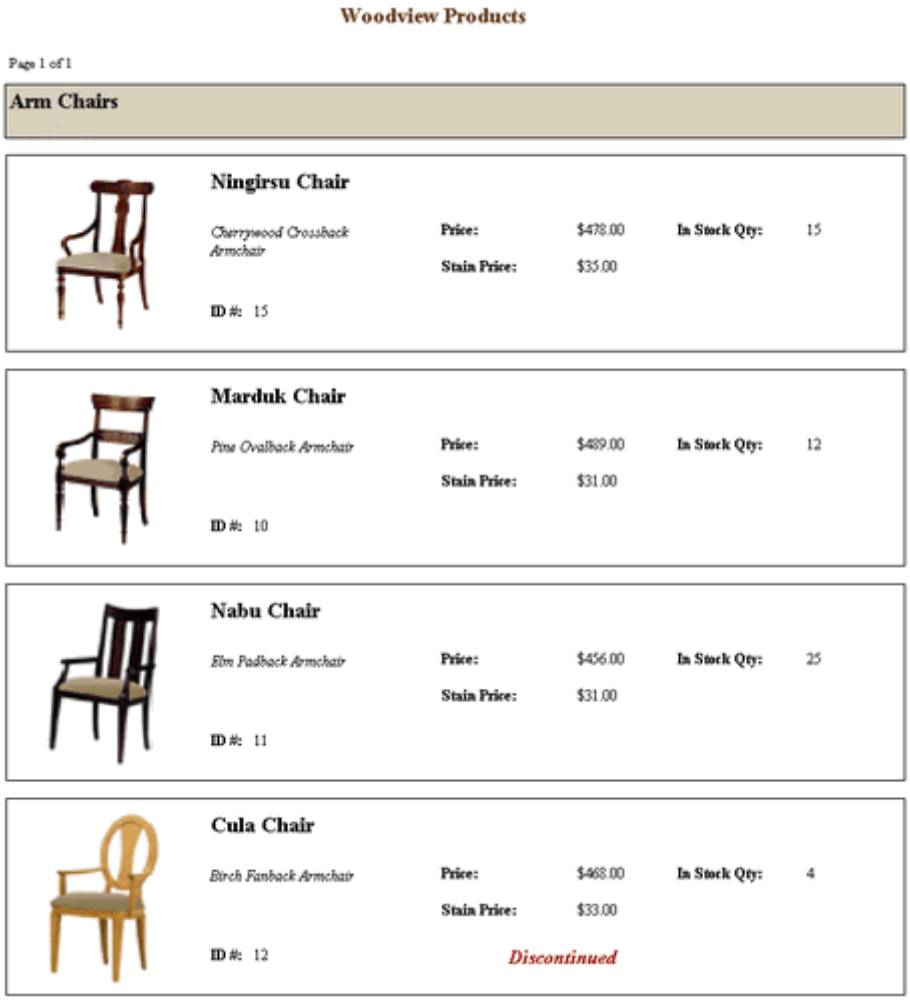
- **QuickStart423.class** を<**EspressReport InstallDir**>/**help/quickstart/classes** ディレクトリから<**Tomcat InstallDir**>/**webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- Tomcat 7.x 以上のバージョンでは、サーブレット（この例では **QuickStart423**）を web.xml に明示的にマップする必要があります。

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

かつ

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

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



生成されたレポート

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

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

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

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

2.4.2.4 JSP

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

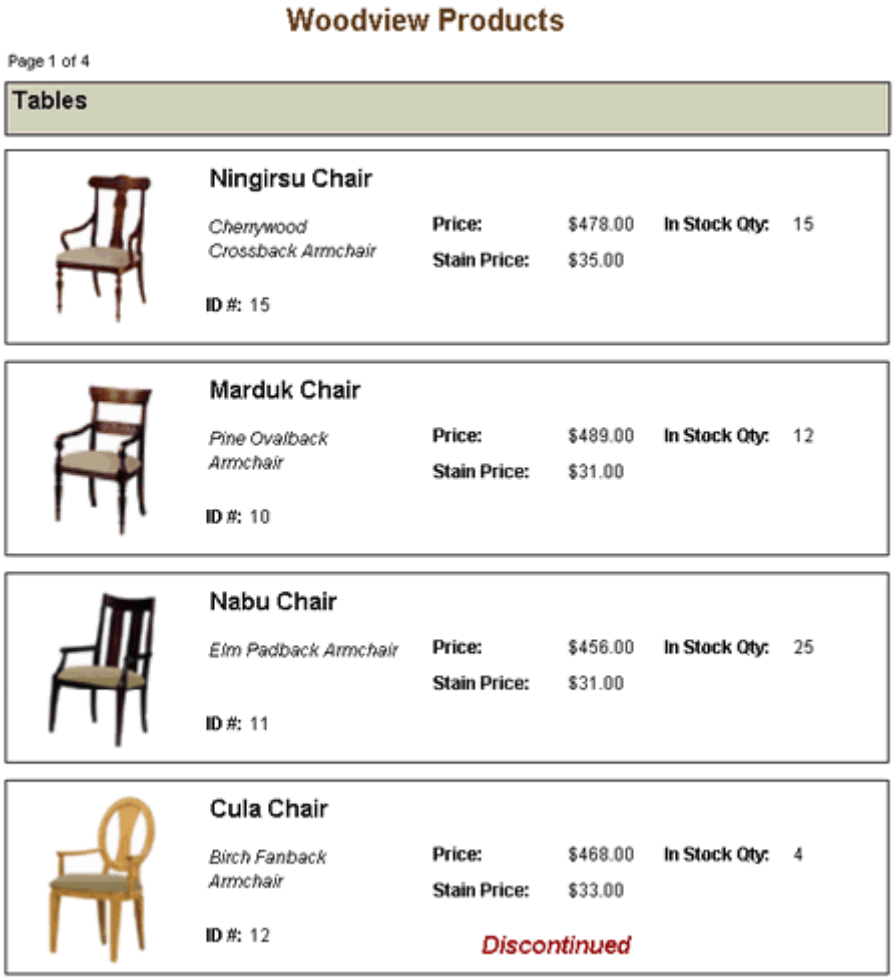
2.4.2.5 Page Viewer

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

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

public class QuickStart424 extends Applet {

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

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

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

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

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

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

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



生成されたレポート

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

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

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

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

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

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

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

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

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

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

2.4.3 プログラムでレポートを作成する

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

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

2.4.3.1 マップの概要 Break CollInfo

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

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

public class QuickStart431 extends Applet {

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

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

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

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

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

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

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

生成されたレポート

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

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

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

2.4.3.2 テンプレートを適用する

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

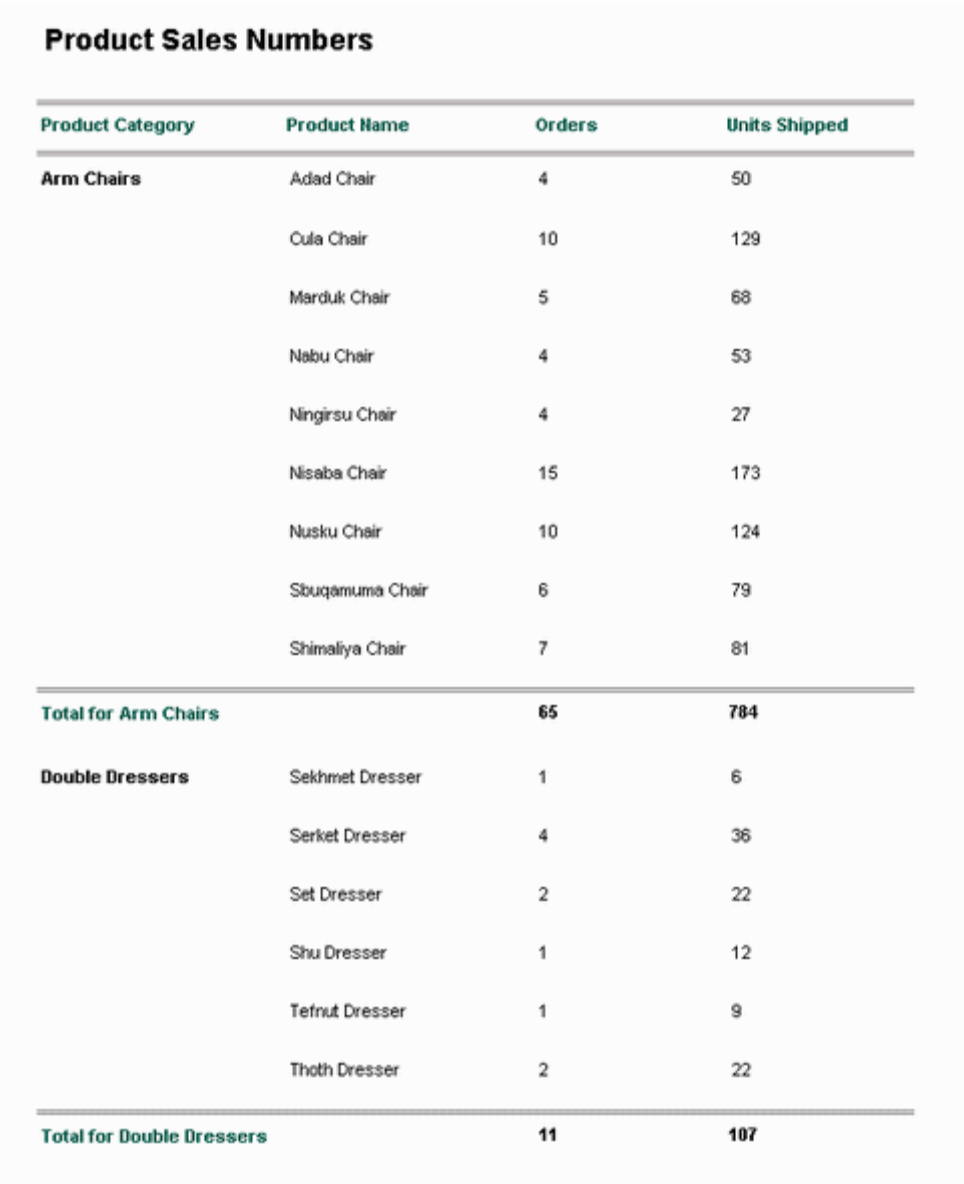
次のコードは、データソースとして **QuickStart43.txt** を使用し、テンプレートとして **QuickStart43.rpt** を使用して、プログラムでサマリブレークレポートを作成する方法を示しています。

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

public class QuickStart432 extends Applet {

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

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

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

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

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

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



Product Sales Numbers

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

生成されたレポート

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

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

2.4.4 レポートのデータソースの変更

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

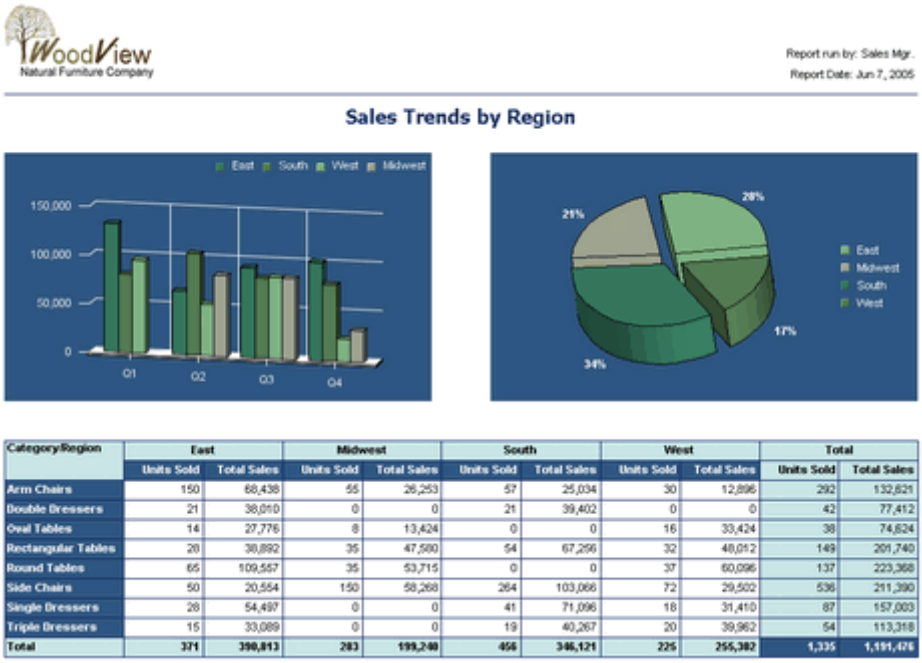
に変更されます。

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

public class QuickStart44 extends Applet {

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

		try {

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

	public void init() {

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

	Component doQuickStart44(Object object) {

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

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

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

                try {

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

                        ex.printStackTrace();
                }

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

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

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



Category/Region	East		Midwest		South		West		Total	
	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales	Units Sold	Total Sales
Arm Chairs	150	68,438	55	26,253	57	25,034	30	12,896	292	132,621
Double Dressers	21	38,010	0	0	21	39,402	0	0	42	77,412
Oval Tables	14	27,776	8	13,424	0	0	16	33,424	38	74,624
Rectangular Tables	28	38,892	35	47,580	54	67,256	32	48,012	149	201,740
Round Tables	65	109,557	35	53,715	0	0	37	60,096	137	223,368
Side Chairs	50	20,554	150	58,268	264	103,066	72	29,502	536	211,390
Single Dressers	28	54,497	0	0	41	71,096	18	31,410	87	157,003
Triple Dressers	15	33,089	0	0	19	40,267	20	39,962	54	113,318
Total	371	390,813	283	199,240	456	346,121	225	255,382	1,335	1,191,476

生成されたレポート

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

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

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

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

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

```
public class QuickStart441 extends Applet {

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

            try {

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

        public void init() {

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

        Component doQuickStart441(Object object) {

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

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

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

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

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

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

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

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

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

    ex.printStackTrace();
}
```

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

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

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

Regional sales per year

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

Detailed sales data by month

Year

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

生成されたレポート

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

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

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

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

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

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

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

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

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

から

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

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

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

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

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

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

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

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

public class QuickStart442 extends Applet {

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

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

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

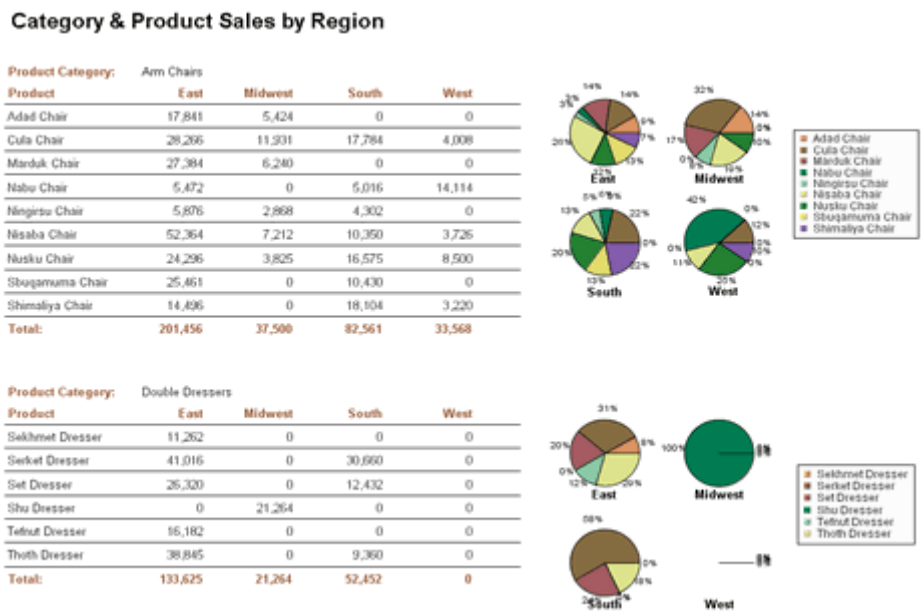
```
            }

        }

        public void init() {

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

        }

        Component doQuickStart442(Object parent) {

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

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

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

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

            try {

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

```
                                false, true, report);

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

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

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

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

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

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

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

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

            }

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

        }

}
```

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

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



Category & Product Sales by Region

Product Category: Arm Chairs

Product	East	Midwest	South	West
Adad Chair	17,841	5,424	0	0
Cula Chair	28,266	11,931	17,784	4,008
Marduk Chair	27,384	6,240	0	0
Nabu Chair	5,472	0	5,016	14,114
Ningirsu Chair	5,876	2,868	4,302	0
Nisaba Chair	52,364	7,212	10,350	3,726
Nusku Chair	24,296	3,825	16,575	8,500
Shugamuma Chair	25,461	0	10,430	0
Shimaliya Chair	14,496	0	18,104	3,220
Total:	201,456	37,500	82,561	33,568

Product Category: Double Dressers

Product	East	Midwest	South	West
Sekhmet Dresser	11,262	0	0	0
Serket Dresser	41,016	0	30,660	0
Set Dresser	26,320	0	12,432	0
Shu Dresser	0	21,264	0	0
Tefnut Dresser	16,182	0	0	0
Thoth Dresser	38,845	0	9,360	0
Total:	133,625	21,264	52,452	0

生成されたレポート

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

から

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

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

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

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

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

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

から

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

2.4.5 レポート要素の変更

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

生成されたレポート

Change ボタンをクリックすると、次のレポートが表示されます。

Top 10 Customers

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

変更ボタンがクリックされた後に生成されるレポート

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

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

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

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

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

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

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

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

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

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

2.4.6 パラメータ化されたレポート

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

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

2.4.6.1 パラメータ値を渡す

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

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

public class QuickStart461 extends Applet {

    public QuickStart461() {
    };

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

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

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

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

        }

}
```

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

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

Sales Report

Report Run By: Ivan

Orders From Jan 04, 1999 To Feb 12, 2001

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

Order Details

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

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

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

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

生成されたレポート

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

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

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

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

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

から

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

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

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

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

public class QuickStart462 extends Applet {

        public QuickStart462() {
        };

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

```
                ex.printStackTrace();
        }
}

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

        try {

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

                report.refreshWithOriginalData();

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

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

```
                return comp;
        }
}
```

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

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

Sales Report

Report Run By: Ivan

Orders From Jan 04, 1999 To Feb 12, 2001

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

Order Details

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

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

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

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

生成されたレポート

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

```
getAllParameters();
```

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

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

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

から

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

2.4.6.3 getParameterPage()と ParamReportGeneratorServlet を使用する

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

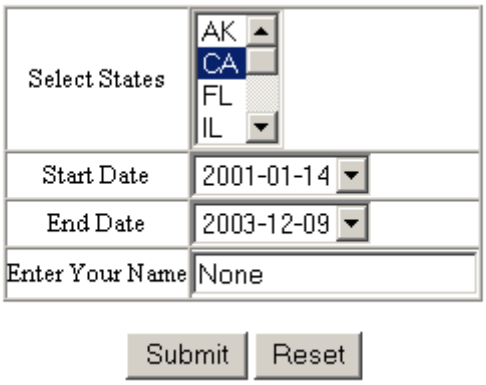
```
import quadbase.reportdesigner.ReportAPI.QbReport;
import quadbase.common.param.*;
import java.applet.*;
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.awt.Font;

public class QuickStart463 extends HttpServlet {

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

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

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

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

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

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

        }

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

サーブレットをデプロイするには：

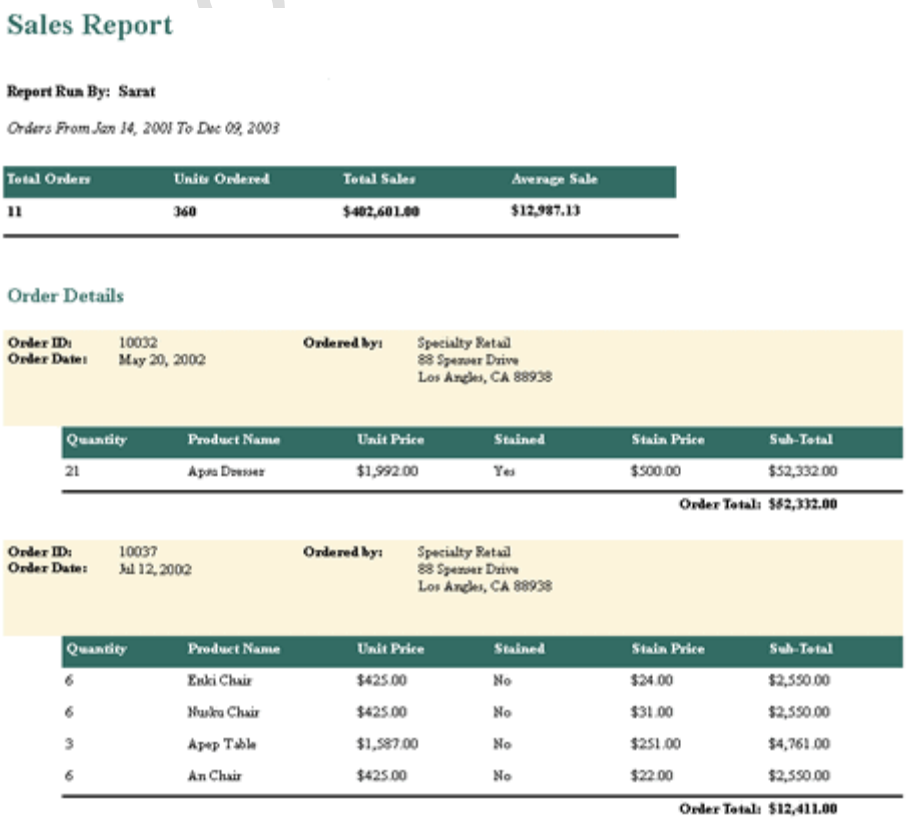
- **QuickStart463.class** を**<EspressReport InstallDir>/help/quickstart/classes** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- **ParamReportGeneratorServlet <EspressReport InstallDir>/ParamReportGenerator** をコンパイルしてください
- **ParamReportGeneratorServlet.class** を **<Tomcat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

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



生成された HTML プロンプトページ

どのパラメータ値を渡すかによって、次のようなレポートが表示されます。



Sales Report

Report Run By: Sarat

Orders From Jan 14, 2001 To Dec 09, 2003

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

Order Details

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

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

Order Total: $52,332.00

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

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

Order Total: $12,411.00

パラメータ値を渡した後に生成されるレポート

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

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

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

CSS を使用する **CssHtmlParameterPageWriter** を使用してパラメータページを生成する例については、**<EspressReport>/help/examples/servicelet/CssParamReport** にアクセスし、**javadoc** の指示に従ってください。

2.4.7 ドリルダウンの展開/エクスポート

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

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

public class QuickStart47 extends HttpServlet {

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

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

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

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

        }

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

サーブレットをデプロイする：

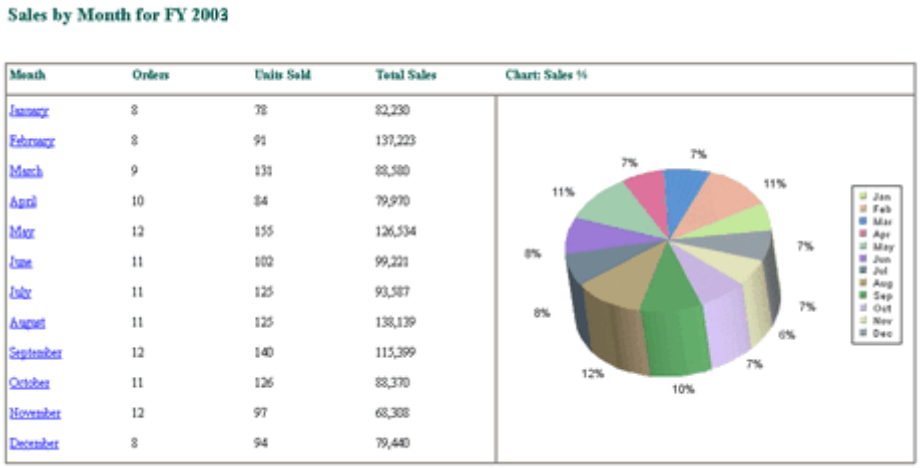
- **QuickStart47.class** を**<EspressReport InstallDir>/help/quickstart/classes** ディレクトリから**<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- **DrillDownReportServlet<EspressReport InstallDir>/DrillDownLinkGenerator** をコンパイルします。
- **DrillDownReportServlet.class** を**<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes**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

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

かつ

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

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



Sales by Month for FY 2003

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

生成されたレポート

クリックしたリンクに応じて、次のようなレポートが表示されます。

Orders for February 2003

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

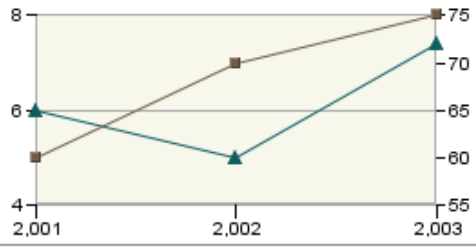
リンクをクリックした後に生成されるレポート

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

Customer Details

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

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



リンクをクリックした後に生成されるレポート

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

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

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

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

2.4.8 サーブレットの詳細

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

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

2.4.8.1 ファイルへの書き込み（PDF）

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

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

public class QuickStart481 extends HttpServlet {

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

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

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

サーブレットをデプロイするには：

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

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

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

レポートが生成された後に表示されるメッセージ

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

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

2.4.8.2 ストリームチャート（DHTML）

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

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

public class QuickStart482 extends HttpServlet {

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

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

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

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

        }

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

サーブレットをデプロイするには：

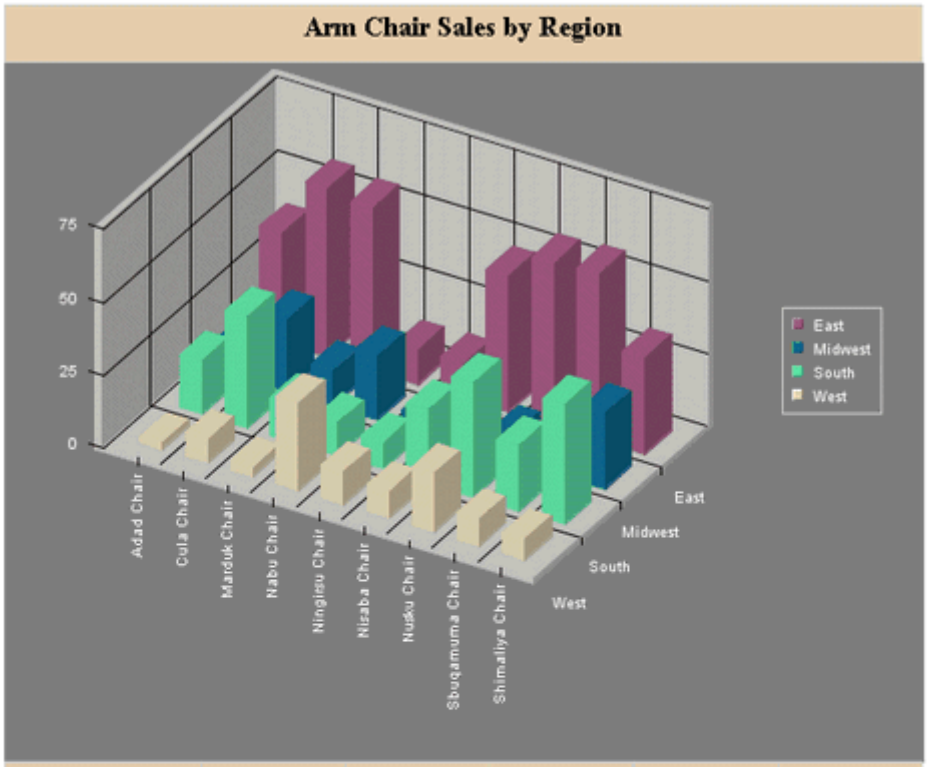
- **QuickStart482.class** を**<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- **RPTImageGenerator<EspressReport InstallDir >/ImageGenerator** をコンパイルします。
- **RPTImageGenerator.class** を**<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリに移動します。
- Tomcat 7.x 以上のバージョンでは、サーブレット（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

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

かつ

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

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



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

生成されたレポート

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

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

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

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

2.4.9 Report Designer の起動

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

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

public class QuickStart49 {

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

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

	public QuickStart49() {

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

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

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

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

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

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

3 インストール

3.1 概要

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

3.1.1 EspressReport ドキュメント

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

クイックスタートガイド:

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

デザイナーガイド:

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

プログラミングガイド:

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

3.2 アーキテクチャとインストール

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

Report Designer:

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

Chart Designer:

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

Report API:

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

Report Viewer:

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

Page Viewer:

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

EspressManager:

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

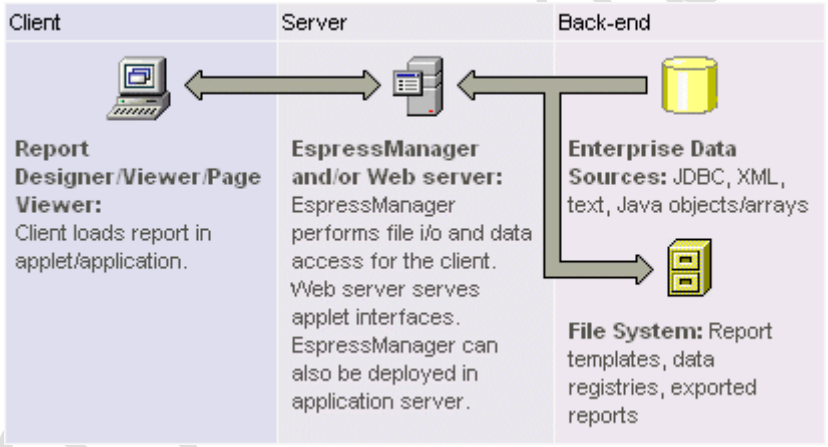
でスケジューリングプロセスを実行し、ユーザ定義のジョブに従ってレポートを実行します。サーバ上でアプリケーションプロセスとして実行することも、アプリケーションサーバ内にサーブレットとしてデプロイすることもできます。

Scheduler:

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

3.2.1 EspressReport アーキテクチャ

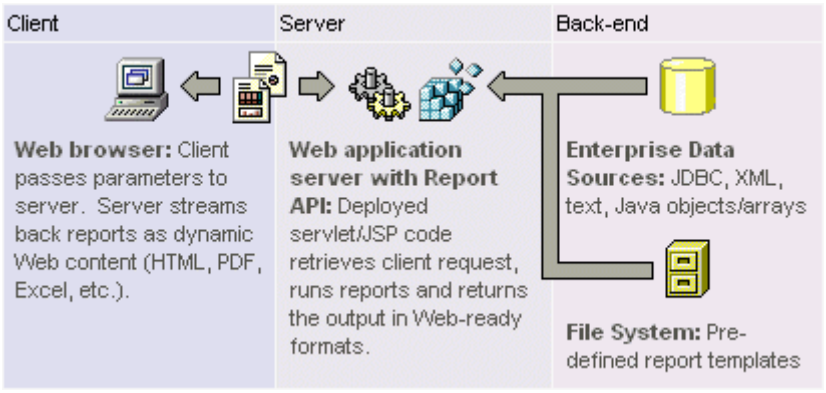
EspressReport には、設計時と実行時の両方で実行できるさまざまな構成があります。設計時に、データアクセスツールとチャートツールを含む Report Designer の GUI インタフェースは、クライアントマシン上のアプリケーションとして、またはクライアントサーバアーキテクチャ内のアプレットとしてロードできます。次の図は、設計時に Report Designer で実行されている EspressReport を示しています。



EspressReport デザイナアーキテクチャ

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

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



サーブレット環境での EspressReport

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

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

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

3.2.2 インストール

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

Windows

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

Unix

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

Mac OS X

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

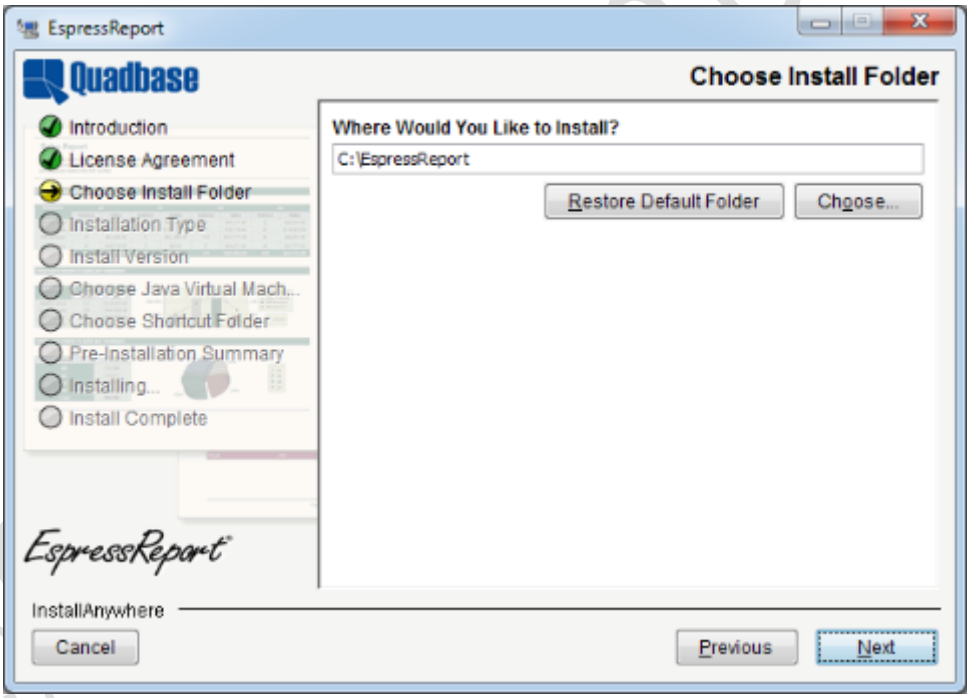
InstallERES.app ファイルを解凍します。**InstallERES.app** をダブルクリックすると、インストーラが起動します。

Pure Java:

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

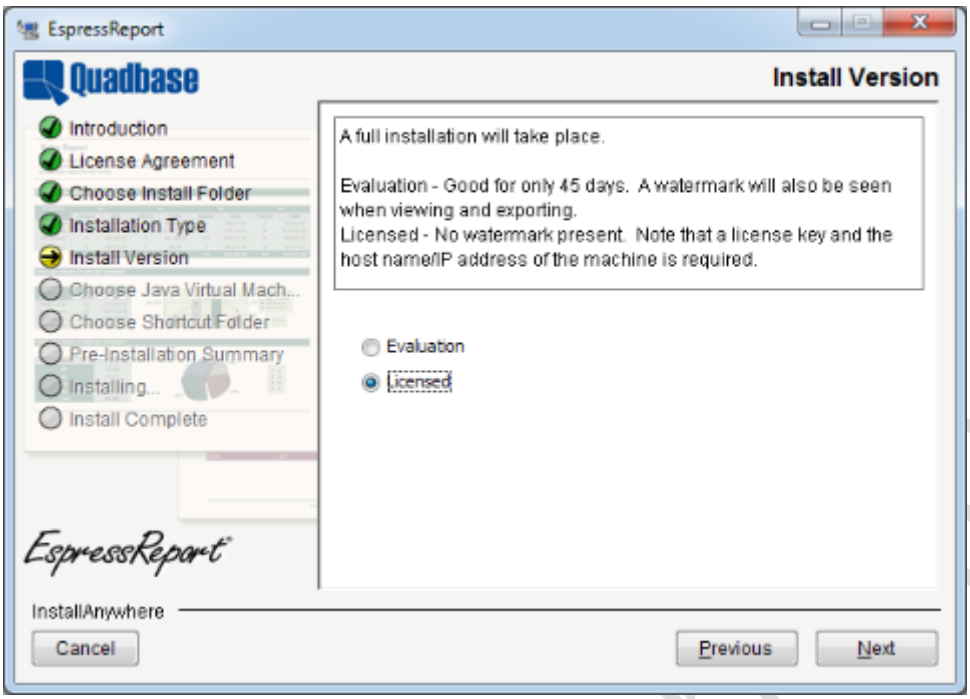
インストールプログラムが起動し、使用許諾契約書に同意すると、最初のオプションで、EspressReport をインストールするディレクトリを指定するよう求められます。

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



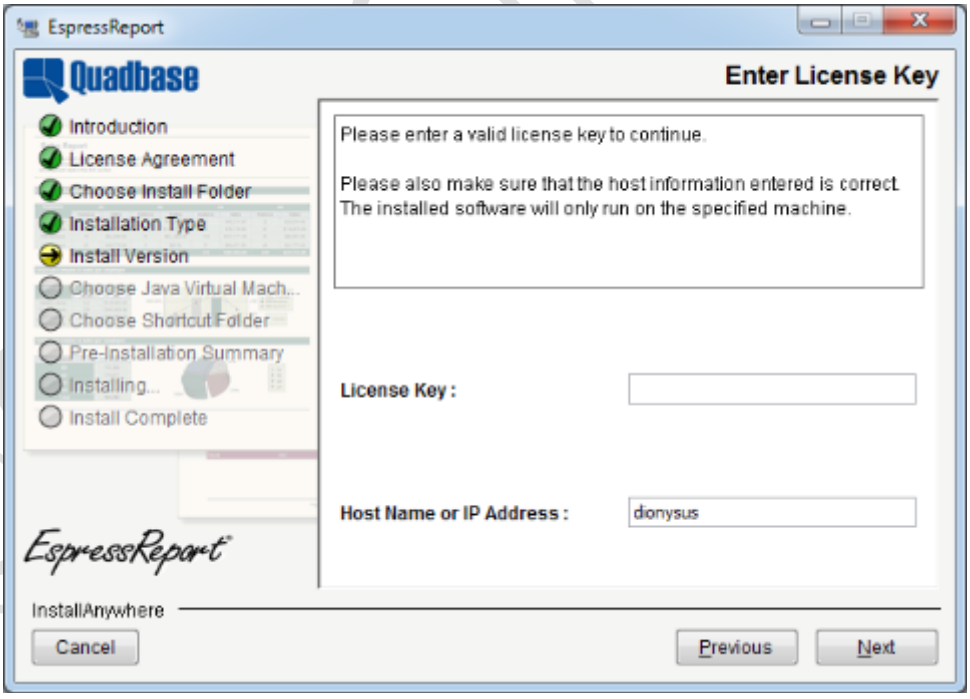
Select Location ダイアログ

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



Version Select ダイアログ

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

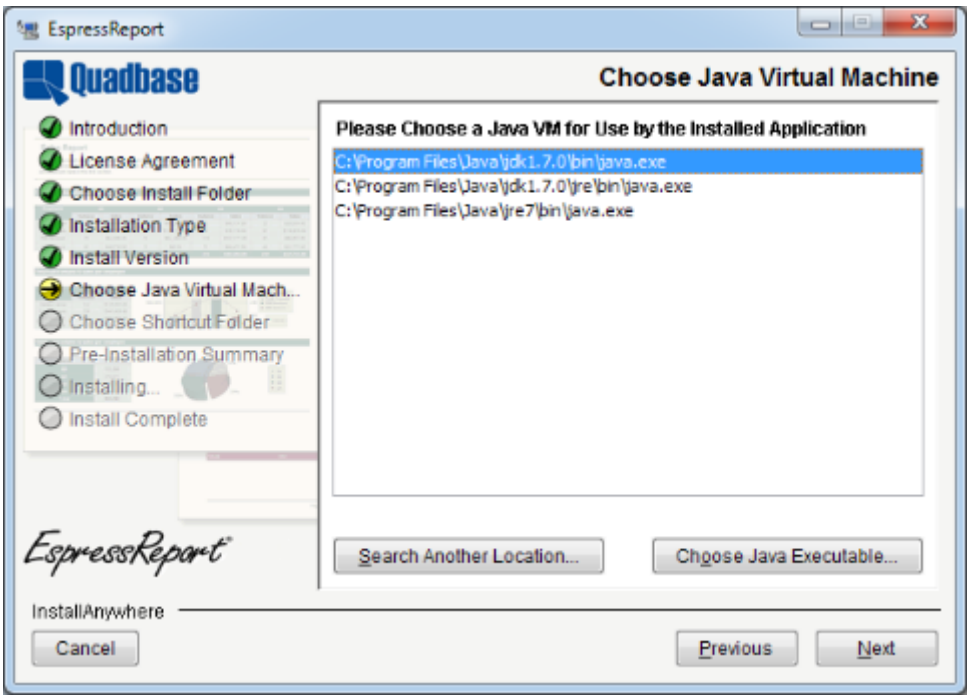


Enter License Key ダイアログ

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

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

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



Choose JVM ダイアログ

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

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

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

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

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

3.2.3 設定

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

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

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

```
[port]
22071

[webroot]
C:¥Intepub¥wwwroot

[users]
guest
Name1 Password1
Name2 Password2

[smtp host]
quadbase.com

[smtp secured]
true

[smtp ssl]
true

[smtp port]
587
```

```
[smtp username]
guest

[smtp password]
password
```

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

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

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

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

検索順序は、IP アドレス、domain_n、domain1 です。

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

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

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

[**smtp host**]セクションでは、スケジューラで電子メール通知またはレポートを送信するために使用さ

れる smtp サーバ名を指定できます。

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

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

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

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

Report API を使用する他の EspressReport コンポーネントとアプリケーション/サーブレット/JSP も（デフォルトでは）**EspressManager.cfg** ファイルを探して EspressManager に接続します。ただし、Report Viewer、Page Viewer、Report API を使用すると、コンポーネントが EspressManager に接続するために使用する IP とポートを明示的に設定できます。

3.2.3.1 アプレットの最大メモリヒープサイズの増加

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

3.2.4 EspressManager の起動

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

EspressManager を起動するには、EspressReport ルートディレクトリの **EspressManager.bat** または **EspressManager.sh** ファイルを実行します（このファイルはインストール時に自動的に生成されます）。Windows インストールの場合、インストール中に指定されたオプションに応じて、スタートメニューまたはデスクトップに追加されたプログラムショートカットを使用できます。EspressManager がデフ

ォルトのプロパティとして自動で起動します。

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

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

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

-help

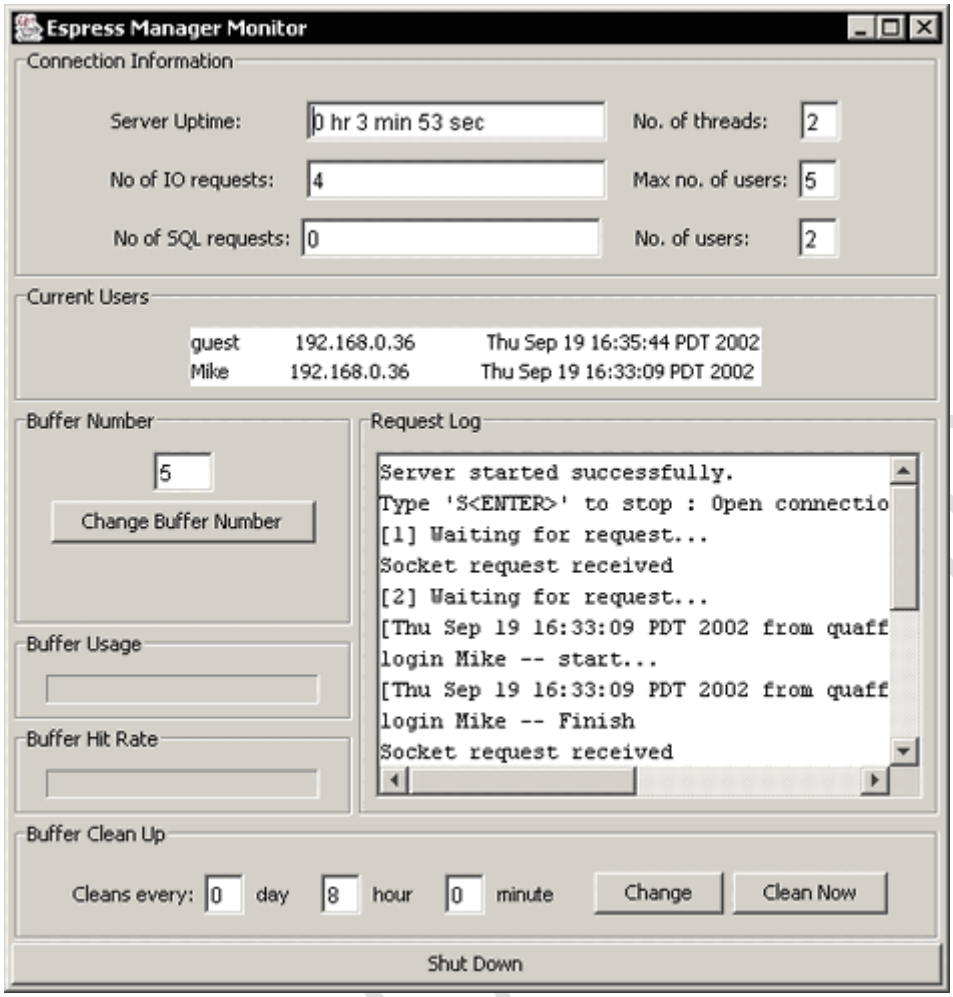
-help と入力すると、EspressManager は利用可能な引数とその意味に関する情報を提供します。**-h** もしくは**-?** を同じオンラインヘルプ情報を取得するために使用することもできます。

-log

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

-monitor:ON/OFF

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



EspressManager ウィンドウ

-runInBackground:ON/OFF

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

-recordLimit:nn

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

-queryTimeout:sss

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

-DBBuffer:nnn

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

-DBCleanAll:ddhhmm

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

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

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

EspressManager を起動すると、更新間隔が表示されます。

-RequireLogin

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

-QbDesignerPassword

この引数を使用すると、**-RequireLogin** 引数がオンのときにパスワードを設定できます。**config.txt** ファイルからのログインを使用する代わりに、**QbReportDesigner** オブジェクトを表示するときにメソッド呼び出しを介してサーバに提供する必要がある特定のパスワードを設定できます。

-ScheduleCallBackClass:classfile:

この引数を使用すると、スケジューラリスナーメカニズムのクラスを指定できます。リスナーは、スケジュールが実行される前にスケジュールされたレポートへのハンドルを返します。

-ListenerManagerClass:classfile:

この引数を使用すると、スケジューラリスナーメカニズムのクラスを指定できます。リスナーは、スケジュールが実行される前にスケジュールされたレポートへのハンドルを返します。

-PageCleanUpTime:ddhhmm:

この引数を指定すると、EspressReport の/pages/ディレクトリのクリーンアップ間隔を指定できます。レポートテンプレート（.pak/.rpt）または **QbReport** オブジェクトを使用して Page Viewer コンポーネントを呼び出すと、このディレクトリにファイルが生成されます。

-MaxRecordInMemory:nnn

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

-MaxCharForRecordFile:nnn

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

-FileRecordBufferSize:nnn

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

-globalFormat:xmlfile:

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

-fontMapping:xmlfile:

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

-htmlDpi:nn:

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

-singleTableForDistinctParamValue

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

-schedulerBuffer:nnn:

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

-xmlEncoding:encoding:

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

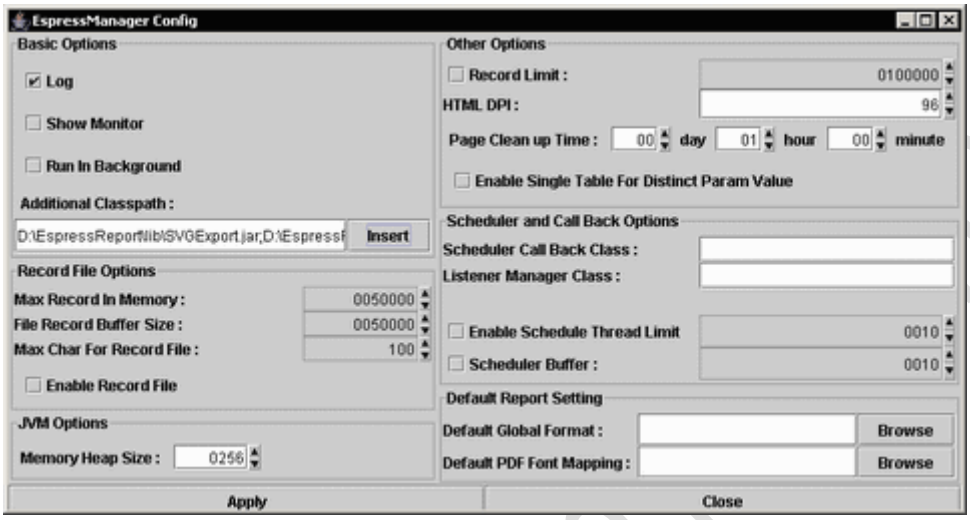
-paperSize:LETTER/A4

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

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

3.2.4.1 EspressManager 設定インタフェース

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



EspressManager 設定ウィンドウ

以下の設定をインタフェースで設定できます：

- Log
- EspressManager Monitor on/off
- Run in Background on/off
- Record Limit
- HTML DPI
- Page Clean-up Time
- Single Table for Distinct Param Value
- Scheduler Callback Class
- Listener Manager Class
- Scheduler Thread Limit
- Schedule Buffer
- Record File Settings
- Default Global Format
- Default PDF Font Mapping

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

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

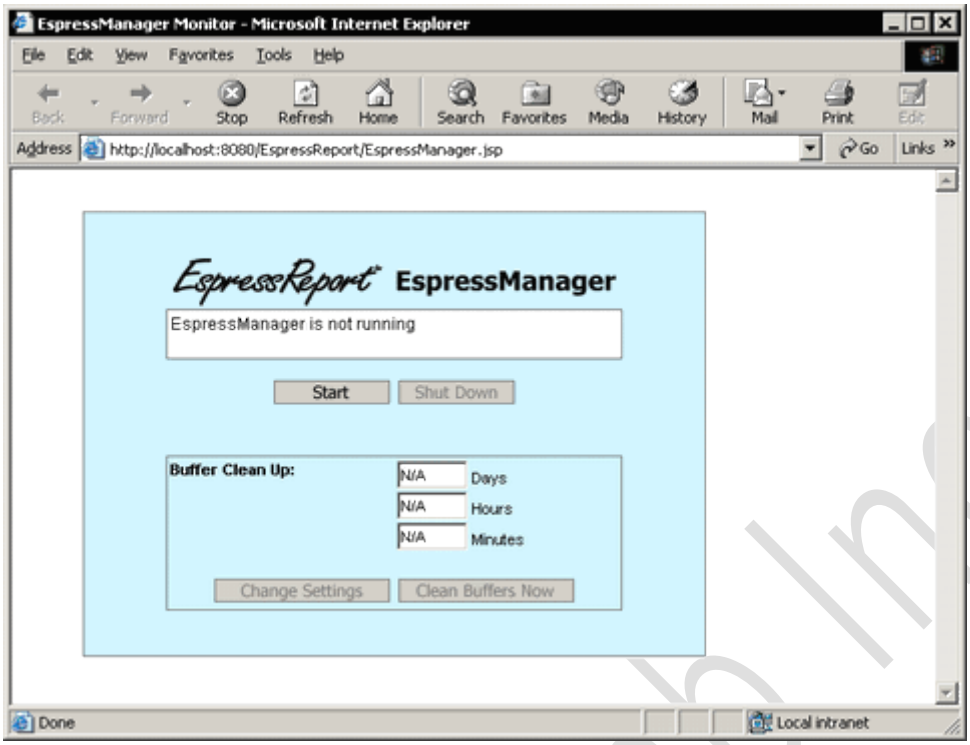
3.2.4.2 サーブレットでの EspressManager の起動

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

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

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

これらのすべての手順が完了したら、Step1 で説明したように、Web ブラウザでアプリケーションサーバ上で使用できる **EspressManager.jsp** ページを指定して、EspressManager ページを読み込むことができます。



EspressManager JSP ウィンドウ

このページが読み込まれたら、**Start** ボタンをクリックして EspressManager を起動します。起動したら、バッファ設定を設定/変更し、このウィンドウからシャットダウンすることができます。
EspressManager が起動したら、このウィンドウを閉じることができます。JSP を再度呼び出してシャットダウンするまで実行を続けます。

EspressManager をサーブレットとして実行する場合、EspressManager に接続するすべてのコンポーネントを考慮して変更する必要があります。これには、Report Designer、スケジューラ、レポートビューア、Page Viewer、および Report API を使用するすべてのアプリケーション/サーブレット/JSP が含まれます。

3.2.4.2.1 Tomcat 4.x/5.x でのサーブレットとしての EspressManager のデプロイ

次のセクションでは、Tomcat 4.x/5.x でサーブレットとして EspressManager を設定して実行する方法について説明します。手順は前のセクションと同じです。この例では、ポート 8080 で Tomcat をローカル（IP 127.0.0.1）で実行していることを前提としています。設定が異なる場合は、これらの手順の IP とポートを使用するポートと置き換えてください。

1. Tomcat のルートフォルダ（**<Tomcat InstallDir>/webapps/ROOT**）の下に **EspressReport** という名前のディレクトリを作成します。
   EspressReport インストール（<EspressReport InstallDir>）から、**EspressManager.jsp**、**MenuError.jsp** ファイル、および/**Web_Component**/ディレクトリを、Tomcat ルートフォルダの下の新しい/**EspressReport**/ディレクトリにコピーします。

2. **<EspressReport InstallDir>/classes** ディレクトリの内容を

<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes ディレクトリにコピーします。実行者が**<Tomcat InstallDir>/conf/web.xml** ファイルでコメントアウトされていないことを確認します（Tomcat ではデフォルトでコメントアウトされていて、**/servlet/context** を利用できません）。

3. **QuadbaseDirectory.cfg** ファイルを EspressReport ディレクトリから**<Tomcat InstallDir>/bin** ディレクトリにコピーします。（通常、これは Tomcat の作業ディレクトリですが、別の場所で Tomcat を起動すると作業ディレクトリが異なる場合があります）作業ディレクトリを確認するには、Web ブラウザを開いて Tomcat を実行し、**http://127.0.0.1:8080/servlet/whatIsMyWorkingDirectory** と入力します。サーブレットは作業ディレクトリへのパスを表示します）作業ディレクトリが**/bin**/以外の場合は、**QuadbaseDirectory.cfg** ファイルをそこに置きます。

4. **servlet-api.jar**（**<Tomcat InstallDir>/common/lib** の下）と同様に、**EspressManager.jar** と **qblicense.jar**（**<EspressReport InstallDir>/lib** の下）を Tomcat の CLASSPATH に追加します。一般に、これは<Tomcat InstallDir>/bin ディレクトリの **setclasspath.bat** または **setclasspath.sh** ファイルを編集することによって行われます。

5. Tomcat サーバを再起動します。

EspressManager を正しくデプロイする必要があります。EspressManager を起動するには、Web ブラウザを開き、次のアドレスに移動します：**http://127.0.0.1:8080/EspressReport/EspressManager.jsp** 画面がロードされ、**Start** ボタンをクリックして EspressManager を起動することができます。

3.2.4.2.2 Tomcat 8.x/9.x でのサーブレットとしての EspressManager のデプロイ

次のセクションでは、Tomcat 8.x/9.x でサーブレットとして EspressManager をセットアップして実行する方法について説明します。手順は前のセクションと同様です。この例では、ポート 8080 で Tomcat をローカル（IP 127.0.0.1）で実行していることを前提としています。設定が異なる場合は、これらの手順の IP とポートを使用するポートと置き換えてください。

1. Tomcat のルートフォルダ（**<Tomcat InstallDir>/webapps/ROOT**）の下に **EspressReport** という名前のディレクトリを作成します。EspressReport インストール（**<EspressReport InstallDir>**）から、**EspressManager.jsp**、**MenuError.jsp** ファイル、**/Web_Component/** ディレクトリを、Tomcat ルートフォルダの下の新しい**/EspressReport/**ディレクトリにコピーします。
2. **<EspressReport InstallDir>/classes** ディレクトリの内容を **<Tomcat InstallDir>/webapps/ROOT/WEB-INF/classes** ディレクトリにコピーします。

3. **servlet-api.jar**（**<Tomcat InstallDir>/lib** の下）と同様に、**EspressManager.jar** と **qblicense.jar**（**<EspressReport InstallDir>/lib** の下）を Tomcat の CLASSPATH に追加します。一般的には、**<Tomcat InstallDir>/bin** ディレクトリに **setenv.bat** を作成することで

行います。Tomcat の CLASSPATH に、**EspressManager.jar**、**qblicense.jar**、および **servlet-api.jar** の値を書き込みます。

setenv.bat の例：
CLASSPATH=C:¥EspressReport70¥lib¥EspressManager.jar;C:¥EspressReport70¥lib¥qblicense.jar;C:¥tomcat9¥lib¥servlet-api.jar

4. **<Tomcat InstallDir>/bin** ディレクトリに **html** という空のフォルダを 1 つ作成します。

5. **<Tomcat InstallDir>/bin** ディレクトリに **userdb** という名前のフォルダを 1 つ作成します。**<EspressReportInstallDir>/userdb** ディレクトリにある **config.txt** を新しく作成したフォルダにコピーします。

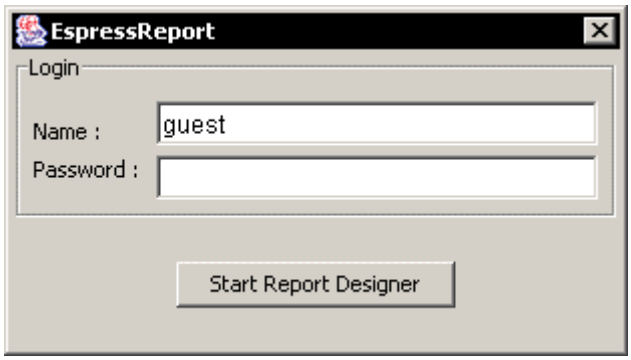
6. Tomcat サーバを再起動します。

今度は EspressManager を正しく配備する必要があります。EspressManager を起動するには、Web ブラウザを開き、次のアドレスに移動します：
http://127.0.0.1:8080/EspressReport/EspressManager.jsp
画面がロードされ、**Start** ボタンをクリックして EspressManager を起動することができます。

3.2.5 Report Designer の起動

スタンドアロン

Report Designer をローカルで実行している場合は、まず **espressmanager.bat** または **.sh** ファイルを実行して EspressManager を起動し、**reportdesigner.bat** または **.sh** ファイルを実行します（Windows または Mac のショートカット/エイリアスも実行できます）。EspressManager にログオンするように促すダイアログボックスが表示されます。**config.txt** ファイルでユーザを変更していない場合は、パスワードなしでユーザ名として **guest** を入力し、**Start Report Designer** ボタンをクリックします。これでアプリケーションが起動します。



Report Designer ログインウィンドウ

ブラウザ

Report Designer をリモートで実行している場合は、EspressManager がリモートマシン上で実行されていることを確認し、ブラウザに EspressReport の URL（**http//machinename/espressreport/index.html**）を入力します。このページには、EspressManager にログオンするためのダイアログが表示されます。**config.txt** ファイルでユーザを変更していない場合は、パスワードなしでユーザ名として **guest** を入力し、**Start Report Designer** ボタンをクリックします。これにより、新しいウィンドウでアプリケーションが起動します。

EspressReport は JDK 1.2 以上を必要とするため、ブラウザ経由で Report Designer を実行するには、Java プラグインをダウンロードする必要があります。また、index.html ページのアプレットは、Windows クライアント上で実行するように設定されています。別のプラットフォームでアプレットを実行している場合は、ページの HTML ソースを変更する必要があります。HTML ソースには 2 つのアプレットがありますが、そのうちの 1 つはコメントアウトされています。Windows 以外のクライアントで実行するには、最初のアプレットをコメントアウトし、2 番目のアプレットをコメント解除します。

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

サーブレットとしての EspressManager の起動で説明されているように、EspressManager がサーブレットとして実行されている場合、ReportDesigner を正常に起動するには、いくつかの変更を加える必要があります。

スタンドアロン

.bat または.sh ファイルを使用して ReportDesigner を実行している場合は、EspressReport がサーブレットとして実行されていることを示すためにファイルを変更し、EspressManager サーブレットの場所をポイントする必要があります。これを行うには、次の引数を.bat ファイルまたは.sh ファイルに追加します。**-servlet:http://IP Address:Port/Context**
例：**-servlet:http://127.0.0.1:8080/servlet**

EspressManager が/**servlet/context** のポート 8080 でローカルホスト上で実行されていることを示します。

ブラウザ

サーブレットとして実行されている EspressManager に接続している Java Web Start アプリケーションを使用して ReportDesigner を実行している場合は、次のものが必要です。

/lib/directory、**index.html**、**ReportDesigner.jsp** ファイル（これらの 3 つはすべて **<EspressReport InstallDir>** にあります）を **<Tomcat InstallDir>/webapps/ROOT/EspressReport** ディレクトリにコピーします。

ReportDesigner.jsp で **jnlp** の内容を変更します。
要素 **jnlp** を **href** = **""**に設定します。

例：<jnlp spec="1.0+" codebase="<%=codebase%>" href="">

jnlp コンテンツの **applet-desc** の要素内に、次のパラメータタグを追加する必要があります。

```
<PARAM NAME="comm_protocol" VALUE="servlet">
<PARAM NAME="comm_url" VALUE="<%=codebase%>">
<PARAM NAME="servlet_context" VALUE="servlet">
```

最初のパラメータは、EspressManager がサーブレットとして実行されていることを示します。2 つ目は、アプリケーションサーバが使用している URL（JSP 式<**%codebase%**>に保存されている URL）で、3 つ目は EspressManager がデプロイされているサーブレットのコンテキストです。

/browserimages/ ディレクトリ（**<EspressReport InstallDir>** の下）を **<Tomcat InstallDir>/bin** ディレクトリにコピーします。

<web-app> と **</web-app>** の間に次のサーブレット定義を追加して、**web.xml** の ReportDesigner（**<Tomcat InstallDir>/webapps/ROOT/WEB-INF** ディレクトリ下）で使用するサーブレットを定義します。

```
<display-name>EspressReport</display-name>
  <description>EspressReport</description>

  <servlet>
                <servlet-name>whatIsMyWorkingDirectory</servlet-name>
                <servlet-class>whatIsMyWorkingDirectory</servlet-class>
                <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet>
                <servlet-name>ESMBaseServlet</servlet-name>
                <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。

ReportDesigner にアクセスするには、Web ブラウザを開き、次のアドレスに移動します：
http://127.0.0.1:8080/EspressReport/index.html

EspressReport.jnlp（Firefox）を開くか、ダウンロードディレクトリに保存してそこから実行して ReportDesigner（Chrome/Microsoft Edge）を開きます。

3.2.6 下位互換性パッチ

古いバージョンからアップグレードした場合、デフォルトの動作にいくつかの変更があることがあります。下位互換性は可能な限り保持されますが、新しい動作が適用される場合があります。新しい動作推奨がされますが、以前のバージョンのチャートやレポートが既にある場合は、古い動作を維持して、チャートとレポートがまったく同じに見えるようにすることができる下位互換性のパッチを提供しています。

パッチは上級ユーザ向けです。パッチを必要としている場合、**操作内容を理解している場合**にのみ、それらを適用してください。ご不明な点がある場合は、サポートに連絡してください。

パッチは**<EspressReportInstallDir>/lib/Patches** ディレクトリにあります。それらは JAR アーカイブに格納されます。パッチを適用するには、アプリケーションのクラスパスに適切な JAR ファイルを追加する必要があります。

すべてのデザイナ（レポートとグラフ用）とビューアにパッチを適用する場合は、EspressReport インストールディレクトリに **espressmanager.bat** ファイルと **reportdesigner.bat** ファイル（Windows を使用している場合）、**espressmanager.sh** ファイルと **reportdesigner.sh** ファイル（他の OS を使用している場合）を編集し、パッチ JAR ファイルへの相対パス（例：**./lib/Patches/patch1.jar**）を classpath パラメータに追加する必要があります。HTML ページから Report Designer を実行する場合は、EspressReport インストールディレクトリの **index.html** ファイルを編集し、パッチ JAR ファイルの相対パスをアプレットタグのアーカイブ属性に追加する必要があります。

API を使用している場合は、アプリケーションのクラスパスにパッチ JAR ファイルを含める必要があります。

以下は現在のバージョンで使用可能なすべてのパッチのリストです。

patch1.jar…グラフの軸のパディングをデフォルトで無効にします

- デフォルトの動作（パッチなし）…軸のパディングはデフォルトでオンです
- パッチ適用後…軸のパディングはデフォルトではオフです
- 新しい動作はバージョン 4.0 で導入されました
- この機能は、API の **IAxis.setAxisPaddingAdded** メソッドまたは Chart Designer の Axis Scale ダイアログを使用して設定することもできます

patch2.jar…注釈テキストの左余白をチャートに追加します

- デフォルトの動作（パッチなし）…注釈テキストに余白が残りません
- パッチ適用後…注釈テキストに余白が残ります
- 新しい動作はバージョン 5.0 で導入されました
- この機能はパブリック API または UI で設定することはできません
- 注釈テキストは、凡例テキスト、チャートタイトル、および Chart Designer の **Insert** >

Text を使用して挿入されたテキストです

patch3.jar…軸 pt のチャート軸オートスケールが 1 未満の場合、最小値/最大値として 0～1 を使用します

- デフォルトの動作（パッチなし）…自動スケールを使用すると、データに応じて最大値と最小値が常に設定されます
- パッチ適用後…max-min が 1 未満で autoscale が使用されている場合、min は 0 に設定され、max は 1 に設定されます
- 新しい動作はバージョン 5.4 で導入されました
- この機能はパブリック API または UI で設定することはできません
- **このパッチは使用しないことをお勧めします**

patch4.jar…新しい円グラフのラベル配置アルゴリズムをオフにします（円グラフの位置に基づいてラベルの配置を計算する）

- デフォルトの動作（パッチなし）…新しい円のラベル配置アルゴリズムが使用されます
- パッチ適用後…古いパイのラベル配置アルゴリズムが使用されます
- 新しい動作はバージョン 6.0 で導入されました
- この機能はパブリック API または UI で設定することはできません

patch5.jar…常にチャート軸の自動スケールに整数値を使用します

- デフォルトの動作（パッチなし）…整数は常に軸自動スケールに使用されます
- パッチ適用後…軸の自動スケールには軸の値のデータ型が使用されます
- 新しい動作はバージョン 6.0 で導入されました
- この機能はパブリック API または UI で設定することはできません

patch6.jar…グラフ軸スケールの最小および最大エラーチェックを無効にします

- デフォルトの動作（パッチなし）…エラーチェックが有効になります
- パッチ適用後…エラーチェックは無効になります（データセットに設定された最大値よりも低い最大値を設定できます）
- 新しい動作はバージョン 6.2 で導入されました
- このパッチは API 専用です
- この機能はパブリック API または UI で設定することはできません

patch7jar…デフォルトでは、円柱チャートと棒グラフのカテゴリ機能のために単一色をオフにします

- デフォルトの動作（パッチなし）…カテゴリ機能の単一色は、デフォルトでオンになっています
- パッチ適用後…カテゴリ機能の単一色は、デフォルトでオフになっています
- 新しい動作はバージョン 6.3 で導入されました
- このパッチは API 専用です
- この機能はパブリック API または UI で設定することはできません
- この機能は、API の **IDataPointSet.setSingleColorForCategories** メソッドまたは Chart

Designer の Chart Options ダイアログで設定することもできます

patch8jar…線グラフの端から端まで単一点データを左軸に表示します

patch9jar…スタックに十分なスペースがなくてもスタックラベルを表示します

4 更新履歴

版	修正日	修正者	内容
1.0	2018/03/01(木)	Climb	初版