DataGrid for WPF/Silverlight

2018.04.12 更新

グレープシティ株式会社

目次

製品の概要

DataGrid for WPF/Silverを使用して、WPF/Silverlight アプリケーションに高度なデータの視覚化を追加します。堅牢なデータ連結 **C1DataGrid** コントロールを使用して、表形式データを WPF/Silverlight アプリケーションで簡単に表示、編集、分析できます。

主な特長

DataGrid for WPF/Silverlight には、次の主要な機能があります。

- **完全な対話式グリッド**

  エンドユーザーの使用感を高めるために、完全な対話式グリッドを作成します。**DataGrid for WPF/Silverlight** には、列のサイズ変更と並べ替え、行の編集、ソート、フィルタ処理、グループ化、フリーズ、選択など多くの対話式機能が組み込まれています。詳細については、「実行時の操作」を参照してください。

- **データのグループ化と集計**

  DataGrid for WPF/Silverlight は、Outlook スタイルのグループ化をサポートします。列ヘッダーをグリッドの上にある領域にドラッグするだけでデータをグループ化できます。展開/折りたたみ可能なノードが自動的に生成されます。グループ化したヘッダー行に、集計関数の計算結果や合計を表示することもできます。詳細については、「列のグループ化」を参照してください。

- **Excel 形式のフィルタ機能**

  デフォルトでは、**DataGrid for WPF/Silverlight** は Excel 形式のフィルタ機能をサポートしています。このタイプのフィルタ機能では、各列でドロップダウンメニューを使用できるため、ユーザーはフィルタ条件を作成できます。詳細については、「列のフィルタ処理」を参照してください。

- **高パフォーマンス**

  DataGrid for WPF/Silverlight では、行と列の両方を再利用することで（UI の仮想化）、大規模なデータセットを処理する際に最適なパフォーマンスを得ることができます。

- **複数の組み込みの列タイプ**

  DataGrid for WPF/Silverlight には、多くの列エディタが組み込まれており、それによってすべての標準データ型をサポートすることができます。テキスト、チェックボックス、DateTime ピッカー、コンボボックス、および画像のためのエディタが組み込まれています。マスク付きテキスト、ハイパーリンク、複数行テキスト、カラーピッカーなどカスタムの列エディタから、必要なエディタを選択することもできます。詳細については、「列型」を参照してください。

- **RowDetailsTemplate および階層のサポート**

  また、DataGrid では、各行の折りたたみ可能なセクション内に **UIElements** を埋め込むことができる **RowDetails**Template テンプレートがサポートされています。たとえば、別の DataGrid を埋め込むだけで、マスター/詳細グリッドを作成して階層データを表示できます。詳細については、「行の詳細の追加」を参照してください。

- **上端行テンプレートと下端行テンプレート**

  DataGrid for WPF/Silverlight の上端行テンプレートと下端行テンプレートを使用して、カスタム行をグリッドに簡単に作成および追加することができます。たとえば、独自のフィルタの設計、合計行の指定、UIElements の埋め込みなどを行うことができます。

- **複数選択モード**

  すべてのセル選択オプションをエンドユーザーに提供します。具体的には、単一セル、単一行、単一列、単一範囲、複数行、複数列、および複数範囲です。**DataGrid for WPF/Silverlight** コントロールはクリップボードをサポートするので、エンドユーザーは選択したセルを Microsoft Excel などの任意のテキストエディタに簡単に貼り付けることができま

す。

- **新規行**

  ユーザーは、グリッドの上部または下部のいずれかに空の新規行を表示することで、新規行を **DataGrid for WPF/Silverlight** に追加できます。詳細については、「行の追加」および「新規行の表示/非表示の設定」を参照してください。

- **カスタム行とカスタム列**

  各 DataGrid 行に対して独自のデータテンプレートを設計し、複数のデータフィールドのデータを結合できる複合列を作成します。

- **ClearStyle を使って簡単に色を変更する**

  DataGrid for WPF/Silverlight は、コントロールのテンプレートを変更することなくコントロールの色を簡単に変更できる ComponentOne 社の新しい ClearStyle 技術をサポートします。色のプロパティをいくつか設定するだけで、グリッド全体のスタイルを簡単に設定できます。詳細については、「ClearStyle」を参照してください。

メモ:説明内に含まれるクラスおよびメンバーに対するリファレンスへのリンクは、原則としてWPF版のリファレンスページを参照します。Silverlight版については、目次から同名のメンバーを参照してください。

はじめに

クイックスタート

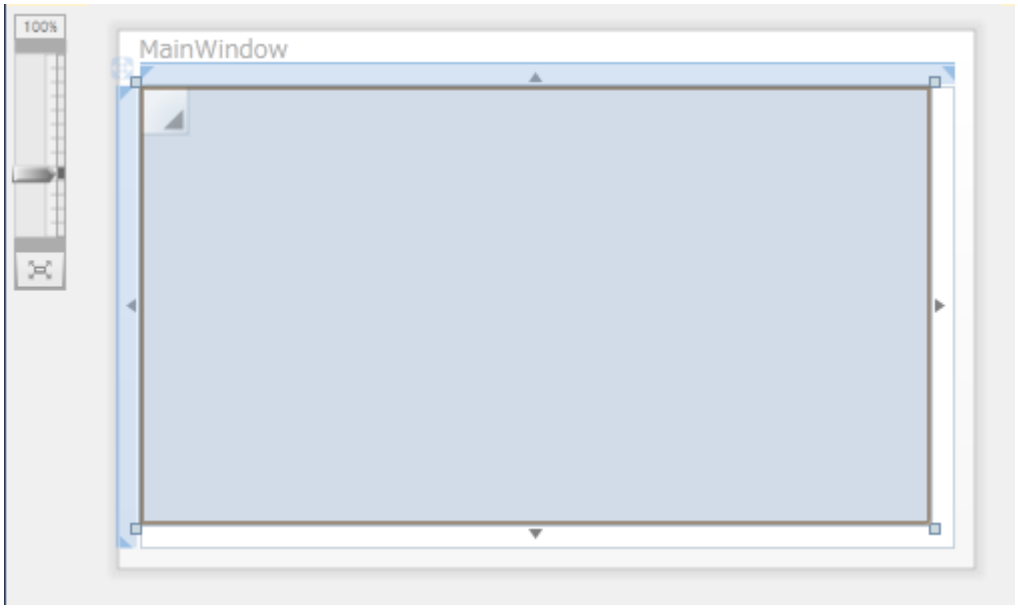
このクイックスタートは、**DataGrid for WPF/Silverlight** を初めて使用するユーザーのために用意されています。このクイックスタートは、最初に Visual Studio で新しいプロジェクトを作成し、**DataGrid for WPF/Silverlight** をアプリケーションに追加し、データソースを追加します。その後、Microsoft Expression Blend に移動して、グリッドをデータソースに連結し、グリッドをカスタマイズした後、グリッドアプリケーションを実行して実行時の動作を確認します。

手順 1:コントロールの追加

この手順では、最初に Visual Studio で **DataGrid for WPF/Silverlight** を使用するグリッドアプリケーションを作成します。C1DataGrid コントロールをアプリケーションに追加するだけで、完全な機能を備えたグリッドとして使用できます。さらに、そのグリッドをアプリケーションに合わせてカスタマイズできます。

プロジェクトをセットアップし、C1DataGrid コントロールをアプリケーションに追加するには、次の手順に従います。

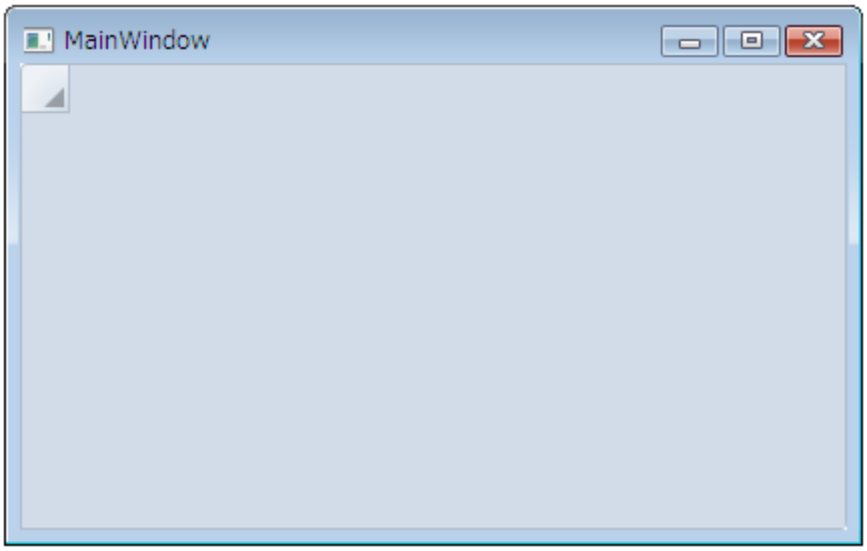
1. Visual Studio で新しい WPF プロジェクトを作成します。
2. ツールボックスに移動し、**[C1DataGrid]**アイコンをダブルクリックして、**Window1** にグリッドコントロールを追加します。
3. 次の図のように、ウィンドウとウィンドウ内の C1DataGrid のサイズを変更します。



4. C1DataGrid のName プロパティを設定して、コントロールを ProductsDataGrid と名前付けます。

ここまでの成果

アプリケーションを実行し、次の図のようなグリッドアプリケーションが表示されることを確認します。



これで、ごく基本的なグリッドアプリケーションを作成できました。ただし、グリッドは空白です。次の手順では、プロジェクトにデータソースを追加し、グリッドをデータソースに連結します。

手順 2: データソースへのグリッドの連結

前の手順では、グリッドアプリケーションを設定しました。ただし、この基本的なグリッドは機能しますが、データは含まれていません。この手順では、引き続き Visual Studio で作業し、プロジェクトにデータソースを追加します。その後、プロジェクトを Microsoft Expression Blend で開き、データソースにグリッドを連結します。

Visual Studio でデータソースを追加し、データ連結を設定するには、次の手順に従います。

1. [**データ**]メニューから[**新しいデータ ソースの追加**]を選択します。[**データ ソース構成ウィザード**]が表示されます。
2. **[データ ソース構成ウィザード]**で**[データベース]**が選択されていることを確認し、**[次へ]**をクリックします。
3. **[データベース モデルの選択]**画面が表示された場合は、**[データセット]**を選択し、**[次へ]**をクリックします。

4. **[データ接続の選択]**画面で**[新しい接続]**ボタンをクリックし、データベースを探して接続します。**[データ ソースの選択]**ダイアログボックスが表示された場合は、**[Microsoft Access データベース ファイル]**を選択し、**[続行]**をクリックします。**[接続の追加]**ダイアログボックスが表示されます。
5. **[接続の追加]**ダイアログボックスで**[参照]**ボタンをクリックして、**NWind.mdb** を見つけます。これを選択し、**[開く]**をクリックします。
6. **[テスト接続]**ボタンをクリックしてデータベースまたはサーバーに正しく接続されていることを確認し、**[OK]**をクリックします。
7. **[OK]**をクリックして、**[接続の追加]**ダイアログボックスを閉じます。**[データ接続の選択]**ページのデータ接続ドロップダウンリストに新しい文字列が表示されます。
8. **[次へ]**ボタンをクリックして続行します。データファイルをプロジェクトに追加し、接続文字列を修正するかどうかを確認するダイアログボックスが表示されたら、**[いいえ]**をクリックします。これは、プロジェクトにデータベースをコピーする必要がないためです。
9. 次のウィンドウでは、**[次の名前で接続を保存する]**チェックボックスが「オン」になっており、テキストボックスに自動的に名前が入力されていることを確認します（"NWindConnectionString"）。[**次へ**]をクリックして続行します。
10. [**データベース オブジェクトの選択**]ウィンドウで、データセットに必要なテーブルとフィールドを選択します。必要な場合は最初に**[テーブル]**ノードを展開し、**[Products]**テーブルを選択して、データセット名を **ProductsDS** に変更します。
11. **[完了]**をクリックしてウィザードを終了します。**ProductsDS.xsd** ファイルがソリューションエクスプローラに表示されます。
12. ソリューションエクスプローラで、**MainWindow.xaml.cs**（または MainWindow.xaml.vb）ファイルをダブルクリックしてコードビューに切り替えます。
13. Window1.xaml.cs（または Window1.xaml.vb）ファイルの先頭に次の参照を追加します。ProjectName は、実際のプロジェクトの名前に置き換えてください。

WPF

Visual Basic

```
Imports C1.WPF.DataGrid
Imports ProjectName.ProductsDSTableAdapters
```

C#

```
using C1.WPF.DataGrid;
using ProjectName.ProductsDSTableAdapters;
```

Silverlight

Visual Basic

```
Imports C1.Silverlight.DataGrid
Imports ProjectName.ProductsDSTableAdapters
```

C#

```
using C1.Silverlight.DataGrid;
using ProjectName.ProductsDSTableAdapters;
```

14. データベースから製品と受注明細のデータを取得するために、次のコードを **MainWindow** クラスに追加します。

Visual Basic

```
Class MainWindow
```

```
    Inherits Window
    Private _productsDataSet As ProductsDS = Nothing
    Public ReadOnly Property ProductsDataSet() As ProductsDS
        Get
            If _productsDataSet Is Nothing Then
                _productsDataSet = New ProductsDS()
                Dim prodTA As New ProductsTableAdapter()
                prodTA.Fill(_productsDataSet.Products)
            End If
            Return _productsDataSet
        End Get
    End Property

    Public Sub New()
        InitializeComponent()
    End Sub
End Class
```

C#

```
public partial class MainWindow : Window
{
    private ProductsDS _productsDataSet = null;
    public ProductsDS ProductsDataSet
    {
        get
        {
            if (_productsDataSet == null)
            {
                _productsDataSet = new ProductsDS();
                ProductsTableAdapter prodTA = new ProductsTableAdapter();
                prodTA.Fill(_productsDataSet.Products);
            }
            return _productsDataSet;
        }
    }

    public MainWindow()
    {
        InitializeComponent();
    }
}
```

15. [F5]キーを押してプロジェクトを実行し、すべての処理が正しく実行されることを確認します。実行中のアプリケーションには空のグリッドが表示されることに注意してください。コンテンツを表示するには、連結を完了する必要があります。
16. 実行中のアプリケーションを閉じ、プロジェクトに戻ります。
17. コードを **MainWindow** コンストラクタに追加します。次のようになります。

Visual Basic

```
Public Sub New()
    InitializeComponent()
    Me.C1DataGrid1.ItemsSource = ProductsDataSet.Products
End Sub
```

C#

```
public MainWindow()
{
   InitializeComponent();
   this.c1DataGrid1.ItemsSource = ProductsDataSet.Products;
}
```

このコードにより、NWind データベースの **Products** テーブルにグリッドが連結されます。**[XAML]**ビューでは、**C1DataGrid** タグが次のように表示されます。

XAML

```
<datagrid:C1DataGrid HorizontalAlignment="Left" Name="C1DataGrid1"
VerticalAlignment="Top" Height="215" Width="384"/>
```

プログラムの実行と動作の確認

グリッドには **Products** テーブルのデータが挿入されています。

これで、**DataGrid for WPF/Silverlight** の C1DataGrid コントロールをデータソースに連結できました。次の手順では、グリッドアプリケーションで使用できる実行時操作をいくつか試してみます。

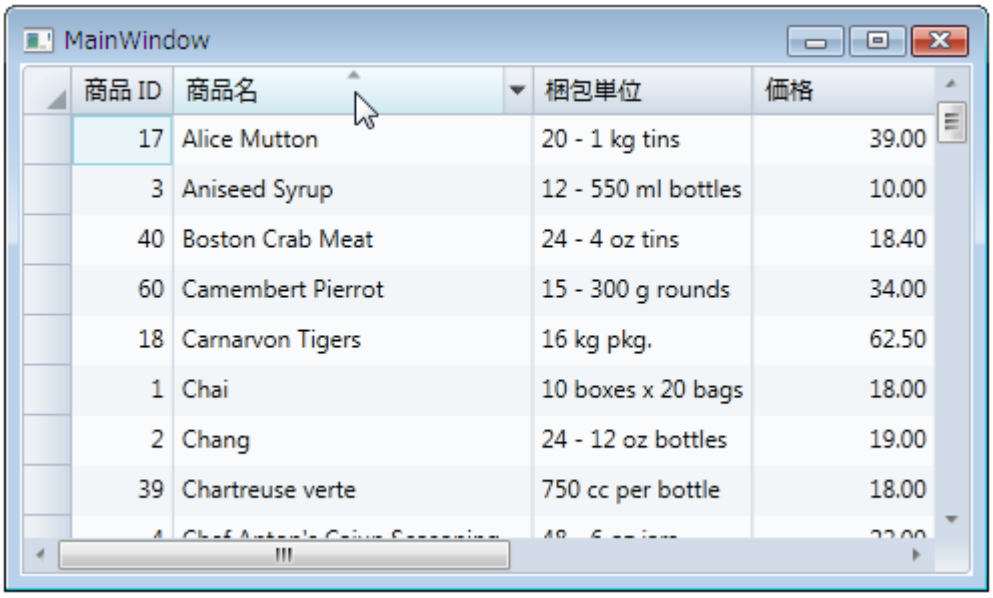
手順 3：アプリケーションの実行

グリッドアプリケーションを作成して、グリッドをデータベースに連結しました。次に、アプリケーションを実行します。グリッドアプリケーションを実行して **Grid for WPF/Silverlight** の実行時の動作を確認するには、次の手順に従います。

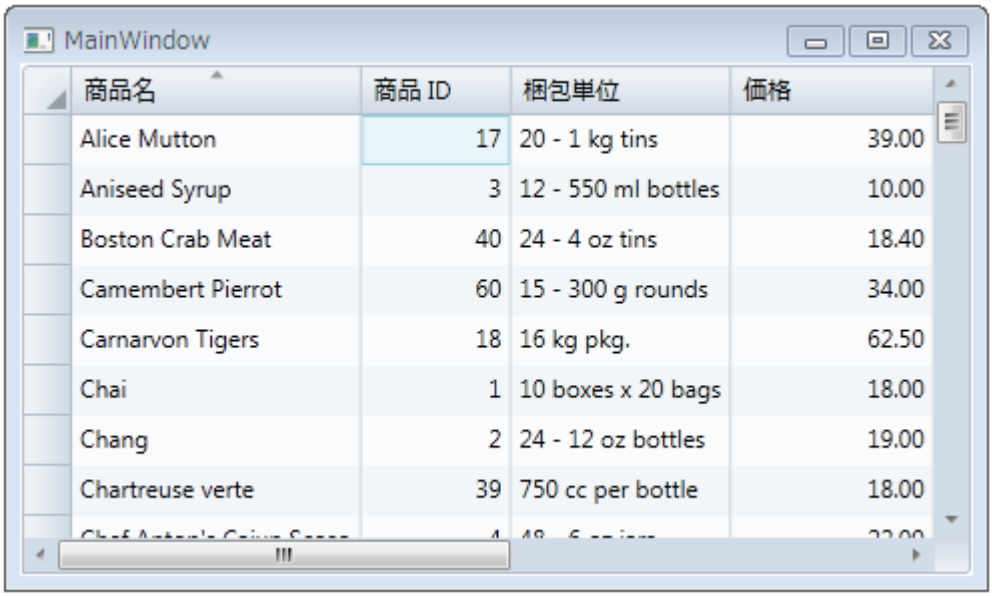
1. **[デバッグ]**メニューから**[デバッグ開始]**を選択し、実行時にグリッドアプリケーションがどのように表示されるかを確認

します。

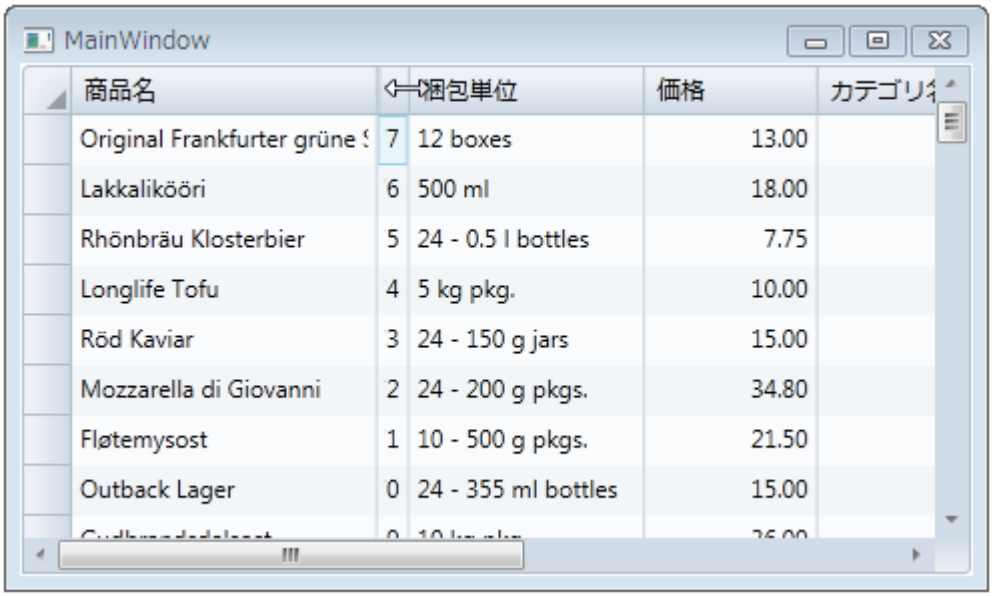
2. 「商品名」ヘッダーをクリックして、グリッドを製品名でソートします。ソートインジケータグリフが表示され、ソートされている列やソートの方向が示されます。



「商品名」列ヘッダーをクリックし、「商品 ID」列ヘッダーの手前までドラッグして、列を並べ替えます。「商品名」列がグリッドの最初の列として表示されます。



3. 列（ここでは［商品 ID］列）のサイズを変更します。それには、列の右端をクリックし、端を新しい位置までドラッグします。



4. セルを1回クリックし、セルの内容を編集して、[Enter]キーを押します。

おめでとうございます!

これで、**DataGrid for WPF/Silverlight** クイックスタートは終了です。**DataGrid for WPF/Silverlight** グリッドアプリケーションを作成し、グリッドをデータソースに連結し、グリッドアプリケーションの実行時機能をいくつか確認することができました。

クラス階層

次のリストに、**DataGrid for WPF/Silverlight** に含まれる重要なクラス間のクラス関係をまとめます。

- **C1.WPF/Silverlight.DataGrid.C1DataGrid : System.Windows.Controls.Control**

  グリッド機能のほとんどをカプセル化します。このコンポーネントは、Visual Studio のツールボックスに表示されます。

- **C1.WPF/Silverlight.DataGrid.DataGridColumn : System.Object**

  グリッド内の1つの列を表します。

- **C1.WPF/Silverlight.DataGridColumnCollection : System.Object**

データグリッドの列のコレクションを表します。

- **C1.WPF/Silverlight.DataGrid.DataGridColumnHeaderPresenter : System.Windows.Controls.Control**

  列のヘッダーを表すコンテンツコントロールです。このコントロールは、ソート、サイズ変更、およびフィルタの要素を含みます。

- **C1.WPF/Silverlight.DataGrid.DataGridRow : System.Object**

  グリッド内の1つの行を表します。

- **C1.WPF/Silverlight.DataGrid.DataGridRowCollection : System.Object**

  行のコレクションです。

- **C1.WPF/Silverlight.DataGrid.DataGridCell : System.Object**

  個々のグリッドセルを表します。

Expression Blend に C1DataGrid コントロールの作成

C1DataGrid コントロールは、Expression Blend で設計時に、XAML、およびコードで簡単に作成できます。次の手順で作成した C1DataGrid コントロールは、空で表示されます。グリッドを連結するか、グリッドにデータを設定する必要があります。

Blend での設計時

C1DataGrid コントロールを Blend で作成するには、次の手順に従います。

1. **[プロジェクト]**ウィンドウに移動し、プロジェクトファイルリストで**[参照]**フォルダを右クリックします。コンテキストメニューから**[参照の追加]**を選択し、**C1.WPF.DataGrid.dll** または **C1.Silverlight.DataGrid.dll** アセンブリを見つけて選択し、**[開く]**をクリックします。**[ツール]**パネルが閉じ、プロジェクトに参照が追加されて、**[アセット]**パネルでコントロールを利用できるようになります。
2. **[ツール]**パネルで、**[アセット]**ボタン（二重山かっこアイコン）をクリックして、**[アセット]**パネルを開きます。
3. **[アセット]**パネルで、左ペインから**[コントロール]**→**[すべて]**項目を選択し、右ペインで**[C1DataGrid]**アイコンをクリックします。**[C1DataGrid]**アイコンが**[ツール]**パネルの**[アセット]**ボタンの下に表示されます。
4. **UserControl** のデザイン領域をクリックして選択します。Visual Studio とは異なり Blend では、次の手順に示すように、WPF コントロールを直接デザインサーフェスに追加できます。
5. **[ツール]**パネルの**[C1DataGrid]**アイコンをダブルクリックして、コントロールをパネルに追加します。これで、C1DataGrid コントロールがアプリケーションに追加されました。
6. 必要に応じて、コントロールを選択し、**[プロパティ]**ウィンドウでプロパティを設定することにより、コントロールをカスタマイズすることもできます。たとえば、C1DataGrid コントロールの **Name** プロパティを「c1datagrid1"、**Height** プロパティを「180"、**Width** プロパティを「250」に設定します。

主な概念とプロパティ

C1DataGridを使用してほとんどのデータベースの読み書きを可能にするアプリケーションを作成するには、主なプロパティがどのようにデータグリッドの要素にマップされるかを理解する必要があります。

データグリッドを表示および編集する手順は次のとおりです。

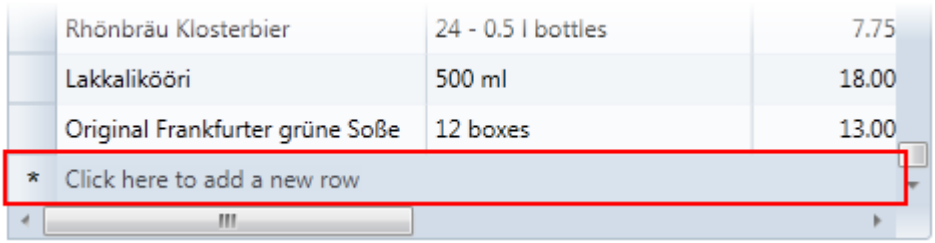
1. EditorItemsSourceプロパティをIEnumberableの実装に設定してグリッドをバインドするか、**DataGridColumn**コレクションエディタを使用して、グリッド内の列のヘッダーを自動的に生成できます。
2. グリッドに対してデータが設定されたら、列を自動的に生成するか、列を明示的に構成するかを決定できます。列を明示的に構成するには、**AutoGenerateColumns**プロパティをFalseに設定します。または、列を自動的に生成するには、Trueを設定します。
3. 次に、データグリッドの列を編集（削除、並べ替え、追加）することができます。詳細については、「実行時の操作」を参

照してください。.

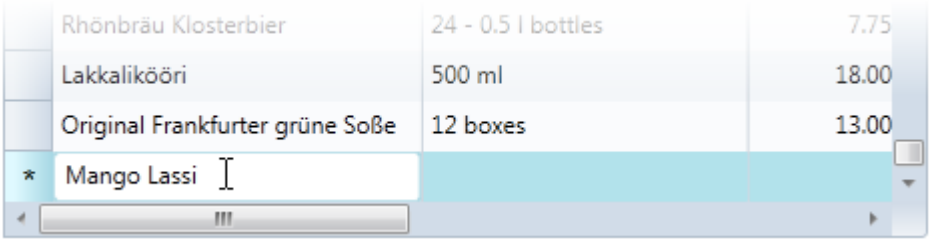
DataGrid の機能

行の追加

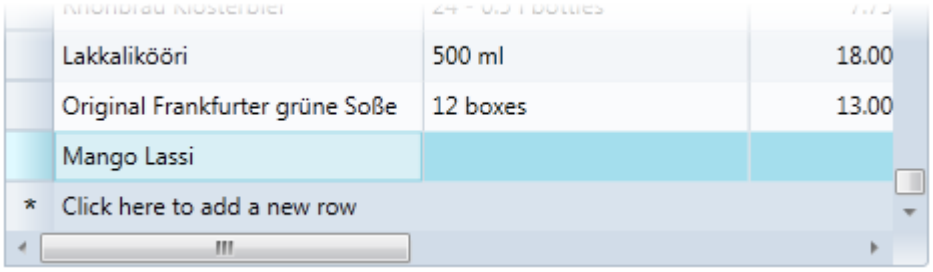
新規行バーを使って実行時にグリッドに行を追加できます。新規行バーはデフォルトでグリッドの最下部に置かれ、アスタリスク記号(*)が付けられています。これを使用して、実行時に新しい情報を入力してグリッドに追加できます。



新しい行を追加するには、新規行バーにテキストを入力するだけです。



[Enter]キーを押すと、テキストがグリッドの新しい行に追加されます。



行の追加を有効にするには、**CanUserAddRows** プロパティを **True**(デフォルト)に設定する必要があります。サンプルについては、「行の追加を無効にする」を参照してください。

行の追加を無効にする

デフォルトでは、エンドユーザーは、実行時にグリッドに新しい行とコンテンツを追加できます。グリッドの下端に新規行バーが表示され、ユーザーはこのバーにテキストを入力することにより、グリッドに新しいコンテンツを追加できます。詳細については、「行の追加」を参照してください。しかし、必要に応じて、CanUserAddRows プロパティを **False** に設定することにより、新規行バー機能を無効にすることができます。

設計時

行の追加を無効にするには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウに移動し、CanUserAddRows プロパティを見つけます。
3. CanUserAddRows プロパティの横にあるチェックボックスを「オフ」にします。

XAML の場合

たとえば、行の追加を無効にするには、CanUserAddRows="False" を < datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250"
CanUserAddRows="False" />
```

コードの場合

たとえば、行の追加を無効にするには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.C1DataGrid1.CanUserAddRows = False
```

C#

```
this.c1DataGrid1.CanUserAddRows = false;
```

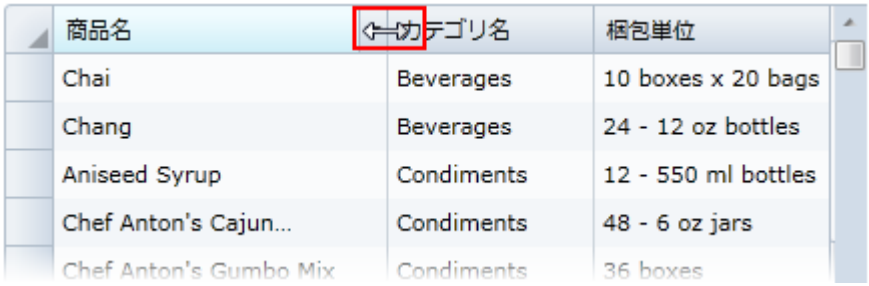
ここまでの成果

アプリケーションを実行し、必要に応じて、グリッドの末尾までスクロールします。グリッドに新規行バーが表示されず、ユーザーがグリッドに新しい行とコンテンツを追加できないことを確認します。セルの編集の詳細については、「行の追加」トピックを参照してください。

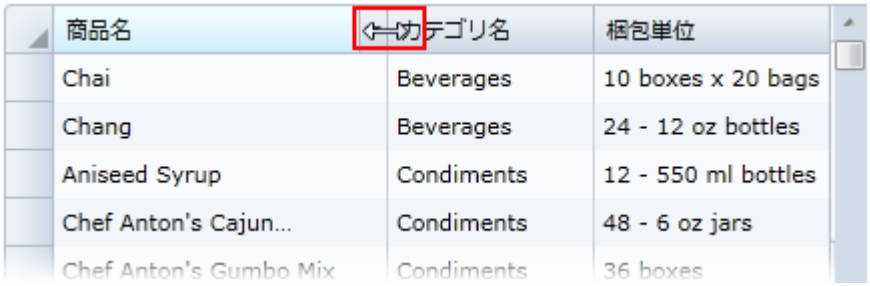
行および列のサイズ変更

ユーザーは、実行時にドラッグアンドドロップ操作で簡単に列と行をサイズ変更できます。実行時に列をサイズ変更するには、次の手順に従います。

1. マウスを列のヘッダーの右境界線に移動します。列のサイズ変更カーソルが表示されます。



2. マウスをクリックし、カーソルを左右にドラッグして、列のサイズを変更します。



3. マウスを放すと、列のサイズ変更操作が終了します。

行インジケータ列をドラッグすると、行を同様の方法でサイズ変更できます。列と行のサイズ変更を有効にするには、**CanUserResizeColumns** プロパティと **CanUserResizeRows** プロパティを **True**（デフォルト）に設定する必要があります。詳細については、「行および列のサイズ変更を無効にする」のトピックを参照してください。

行および列のサイズ変更を無効にする

デフォルトでは、エンドユーザーは、実行時にグリッド内の列および行のサイズを変更できます。詳細については、「行および列のサイズ変更」を参照してください。ただし、必要に応じて、**CanUserResizeColumns** プロパティと**CanUserResizeRows** プロパティを **False** に設定して、行および列のサイズ変更機能を無効にすることができます。

設計時

行および列のサイズ変更を無効にするには、次の手順に従います。

1. **C1DataGrid** コントロールをクリックして選択します。
2. [プロパティ]ウィンドウに移動し、**CanUserResizeColumns** プロパティを見つけます。
3. **CanUserResizeColumns** プロパティの横にあるチェックボックスを「オフ」にします。
4. [プロパティ]ウィンドウで、**CanUserResizeRows** プロパティを見つけます。
5. **CanUserResizeRows** プロパティの横にあるチェックボックスを「オフ」にします。

XAML の場合

たとえば、行および列のサイズ変更を無効にするには、CanUserResizeColumns="False" CanUserResizeRows="False" を <datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250"
CanUserResizeColumns="False" CanUserResizeRows="False"/>
```

コードの場合

たとえば、行および列のサイズ変更を無効にするには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.C1DataGrid1.CanUserResizeColumns = False
Me.C1DataGrid1.CanUserResizeRows = False
```

C#

```
this.c1DataGrid1.CanUserResizeColumns = false;
this.c1DataGrid1.CanUserResizeRows = false;
```

ここまでの成果

アプリケーションを実行し、ドラッグアンドドロップ操作を実行して、実行時に列または行のサイズ変更ができなくなったことを確認します。列の並べ替えの詳細については、「行および列のサイズ変更」トピックを参照してください。

列の並べ替え

エンドユーザーは、実行時に列を簡単に並べ替えることができます。実行時に列を並べ替えるには、次の手順に従います。

1. 並べ替える列の列ヘッダーをクリックします。
2. 列ヘッダーを列の並べ替え後の位置までドラッグします。列を配置できる場所には、ラインが表示されます。

3. マウスボタンを放すと、列が新しい位置に配置され、並べ替えられます。

列の並べ替えを有効にするには、**CanUserReorderColumns** プロパティを **True**（デフォルト）に設定する必要があります。

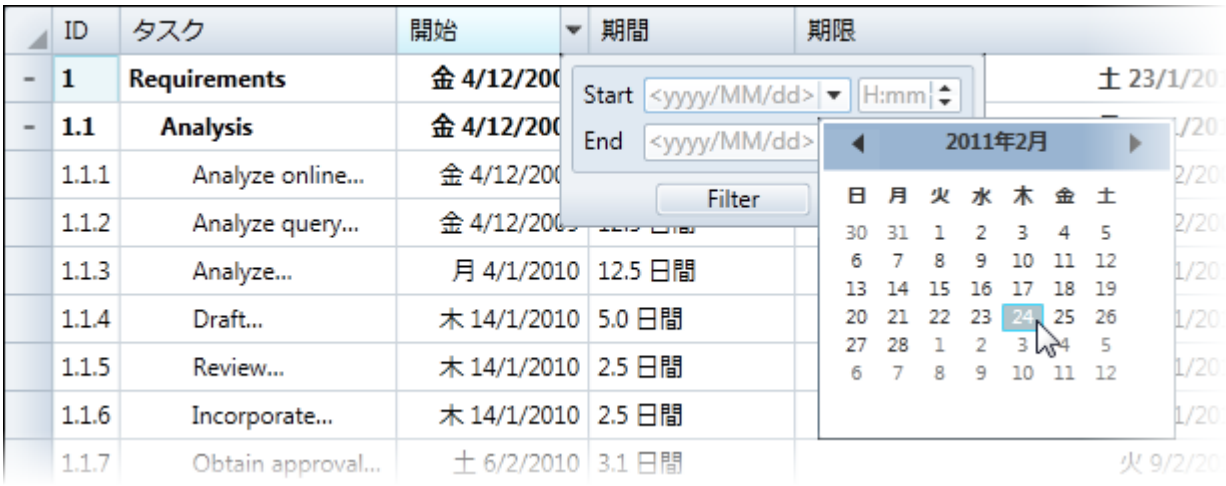
列型

DataGrid for WPF/Silverlight の **C1DataGrid** コントロールを使用すると、柔軟な方法で行と列にデータのコレクションを表示できます。このコントロールには多くの列エディタが組み込まれており、それによってすべての標準データ型がサポートされます。組み込みの列型は次のとおりです。

列型	説明
DataGridBoundColumn	グリッドのデータソース内の1つのプロパティに連結できる列。これは、連結未定義データに対するデフォルトの列型です。
DataGridTextColumn	テキスト列。これは、連結文字列データに対するデフォルトの列型です。
DataGridCheckBoxColumn	チェックボックス列。これは、連結ブール値データに対するデフォルトの列型です。
DataGridComboBoxColumn	コンボボックス列。これは、連結列挙型データに対するデフォルトの列型です。
DataGridDateTimeColumn	日時列（画像については下記を参照）。これは、連結日付/時刻データに対するデフォルトの列型です。
DataGridImageColumn	画像列。
DataGridNumericColumn	数値列。これは、連結数値データに対するデフォルトの列型です。書式は型に基づいて決定されます。たとえば、型が整数の場合、書式には小数点以下が含まれません。
DataGridTemplateColumn	カスタムコンテンツをホストするテンプレート列。
カスタム列	カスタム列。複合列、色の列、Gif 列、ハイパーリンク列、マスク付きテキスト列、複数行テキ

	スト列などのカスタム列の例については、C1DataGrid_Demo サンプルを参照してください。

これらの列型には、入力検証機能が組み込まれています。たとえば、**DataGridDateTimeColumn** 列には、日付を選択するためのカレンダーがあります。



カスタム列

列セルコンテンツのカスタマイズ

このセクションでは、列に含まれるセルが編集モードになっていないときに、セルコンテンツとして表示される UI 要素を変更する方法について説明します。

セルコンテンツの UI 要素はデータグリッドによって再利用されることに注意してください。つまり、この列で使用されている UI 要素は、他の列によって作成された可能性があります。

カスタムのセルコンテンツを実装するには、以下のメソッドをオーバーライドする必要があります。

- **GetCellContentRecyclingKey**:後から再利用するためにセルコンテンツを共有プールに保存する際に使用するキー。同じ **RecyclingKey** を返す列は、同じセルコンテンツのインスタンスを共有する候補になります。
- **CreateCellContent**:セル内に情報を表示するために使用されるビジュアル要素を作成します。
- **BindCellContent**:セルコンテンツプレゼンタを初期化します。このメソッドは、対応する依存プロパティの **SetBinding** を「row.DataItem」に設定した状態で **cellContent** のプロパティを設定する必要があります。row.DataItem は、連結の中で直接、または **cellContent** の **DataContext** によって設定可能なソースです。
- **UnbindCellContent**:このメソッドは、セルコンテンツを再利用する前に呼び出されます。

ハイパーリンク列の実装では、メソッドは次の例のようになります。次のメソッドでは、この列に対して別のキーが返されます(デフォルトキーは typeof(TextBlock))。これは、この列が他の列とセルコンテンツの要素を共有しないことを意味します。別の列が同じキーを返す場合は例外ですが、そのようなことはほとんど起こりません。

Visual Basic

```
Public Overloads Overrides Function GetCellContentRecyclingKey(ByVal row As
DataGridRow) As Object
    Return (GetType(HyperlinkButton))
End Function
```

C#

```
public override object GetCellContentRecyclingKey(DataGridRow row)
{
    return typeof(HyperlinkButton);
}
```

CreateCellContent メソッドは、再利用されるハイパーリンクがない場合に、データグリッドによって呼び出されます。この場合、ハイパーリンクを含むセルがアンロードされると、そのセルで使用される新しいハイパーリンクが作成されます。作成されたハイパーリンクは、後から他のセルで使用するために保存されます。

Visual Basic

```
Public Overloads Overrides Function CreateCellContent(ByVal row As DataGridRow) As
FrameworkElement
    Return New HyperlinkButton()
End Function
```

C#

C#

```
public override FrameworkElement CreateCellContent(DataGridRow row)
{
   return new HyperlinkButton();
}
```

ハイパーリンクが作成されるか、再利用されたハイパーリンクが取得されると、データグリッドはパラメータとしてハイパーリンクを渡して **BindCellContent** メソッドを呼び出します。このメソッドでは、ハイパーリンクのプロパティを設定してセルのデータに連結する必要があります。

Visual Basic

```
Public Overloads Overrides Sub BindCellContent(ByVal cellContent As FrameworkElement,
ByVal row As DataGridRow)
    Dim hyperlink = DirectCast(cellContent, HyperlinkButton)
    If Binding IsNot Nothing Then
        Dim newBinding As Binding = CopyBinding(Binding)
        newBinding.Source = row.DataItem
        hyperlink.SetBinding(HyperlinkButton.NavigateUriProperty, newBinding)
    End If
    hyperlink.HorizontalAlignment = HorizontalAlignment
    hyperlink.VerticalAlignment = VerticalAlignment
End Sub
```

C#

```
public override void BindCellContent(FrameworkElement cellContent, DataGridRow row)
{
```

```
    var hyperlink = (HyperlinkButton)cellContent;
    if (Binding != null)
    {
        Binding newBinding = CopyBinding(Binding);
        newBinding.Source = row.DataItem;
        hyperlink.SetBinding(HyperlinkButton.NavigateUriProperty, newBinding);
    }
    hyperlink.HorizontalAlignment = HorizontalAlignment;
    hyperlink.VerticalAlignment = VerticalAlignment;
}
```

連結の **Source** プロパティにデータ項目を設定するのではなく、ハイパーリンクのデータコンテキストとしてデータ項目を設定することもできます。次に例を示します。

Visual Basic

```
Hyperlink.DataContext = row.DataItem
```

C#

```
Hyperlink.DataContext = row.DataItem;
```

結果は同じですが、この方法のパフォーマンスは、直接連結ソースのプロパティを設定するより低くなります。

カスタム行

行セルコンテンツのカスタマイズ

このトピックでは、セルコンテンツをカスタマイズする方法について説明します。たとえば、フィルタ行を作成する場合は、次の手順に従います。次の図のように、最初の行の各セルに **TextBox** を配置したグリッドを作成して、そこに入力したテキストによってグリッドがフィルタ処理されるようにすることができます。

クラスファイルの追加

カスタム行が記述された新しいクラスファイルを追加することが必要になる場合があります。たとえば、次の手順に従って新しいクラスファイルを追加します。

1. ソリューションエクスプローラで、プロジェクト名を右クリックし、**[追加]**→**[新しい項目]**を選択します。
2. **[新しい項目の追加]**ダイアログボックスで、利用可能なテンプレートのリストから**[クラス]**を選択します。

3. クラスに「DataGridFilterRow」などの名前を付け、**[追加]**ボタンをクリックしてプロジェクトにクラスを追加します。
4. クラスを更新します。次のようになります。

WPF

Visual Basic

```
Imports C1.WPF.DataGrid
Public Class DataGridFilterRow
   Inherits DataGridRow
End Class
```

C#

```
using C1.WPF.DataGrid;
public class DataGridFilterRow : DataGridRow
{
}
```

Silverlight

Visual Basic

```
Imports C1.Silverlight.DataGrid
Public Class DataGridFilterRow
   Inherits DataGridRow
End Class
```

C#

```
using C1.Silverlight.DataGrid;
public class DataGridFilterRow : DataGridRow
{
}
```

これにより、**DataGridRow** から継承するようにクラスが更新されます。ファイルは、作成後に **DataGridRow** から継承する必要があります。

クラスを追加すると、それを使ってグリッドにフィルタ処理を実装できます。

メソッドのオーバーライド

カスタム行のセルコンテンツを指定するためにオーバーライドする必要があるメソッドは、カスタム列で公開されているメソッドとほぼ同じです。カスタムのセルコンテンツを実装するには、以下のメソッドをオーバーライドする必要があります。

- **HasCellPresenter**:この行の指定された列にセルを含めるかどうかを決定します。
- **GetCellContentRecyclingKey**:後から再利用するためにセルコンテンツを共有プールに保存する際に使用するキー。同じ **RecyclingKey** を返す行は、同じセルコンテンツのインスタンスを共有できます。
- **CreateCellContent**:この列のセル内に情報を表示するために使用されるビジュアル要素を作成します。
- **BindCellContent**:セルコンテンツプレゼンタを初期化します。
- **UnbindCellContent**:このメソッドは、セルコンテンツを再利用する前に呼び出されます。

フィルタ行では、すべての列に対応するセルがあるので、**HasCellPresenter** メソッドは常に True を返します。サマリー行など

他のシナリオでは、セルは集計関数がある列にのみ存在します。

GetCellContentRecyclingKey メソッドは typeof(TextBox) を返すので、それによってテキストボックスを再利用できます。**CreateCellContent** は、その新しいインスタンスを作成します。次のコードを追加します。

Visual Basic

```
Protected Overrides Function GetCellContentRecyclingKey(column As DataGridColumn) As
Object
      Return GetType(TextBox)
End Function
Protected Overrides Function CreateCellContent(column As DataGridColumn) As
FrameworkElement
      Return New TextBox()
End Function
```

C#

```
protected override object GetCellContentRecyclingKey(DataGridColumn column)
{
    return typeof(TextBox);
}
protected override FrameworkElement CreateCellContent(DataGridColumn column)
{
    return new TextBox();
}
```

フィルタ処理の実装

前の手順では各セルに **TextBox** を追加しましたが、これらのコントロールは、現時点では何も実行しません。フィルタ処理を実装するには、次の手順に従います。

1. **BindCellContent** メソッドに次のコードを追加します。

Visual Basic

```
Protected Overrides Sub BindCellContent(cellContent As FrameworkElement, column
As DataGridColumn)
   Dim filterTextBox = DirectCast(cellContent, TextBox)
   '列に FilterMemberPath が指定されていない場合、
   'TextBox にテキストは入力できません。
   If String.IsNullOrEmpty(column.FilterMemberPath) Then
         filterTextBox.IsEnabled = False
         filterTextBox.Text = "利用不可能"
   Else
         filterTextBox.Text = ""
         filterTextBox.IsEnabled = True
   End If
   ' TextChanged を処理し、列にフィルタを適用します。
```

```
   filterTextBox.TextChanged += New EventHandler(Of TextChangedEventArgs)
(filterTextBox_TextChanged)
End Sub
```

C#

```
protected override void BindCellContent(FrameworkElement cellContent,
DataGridColumn column)
{
    var filterTextBox = (TextBox)cellContent;
    //列に FilterMemberPath が指定されていない場合、
    //TextBox にテキストは入力できません。
    if (string.IsNullOrEmpty(column.FilterMemberPath))
    {
        filterTextBox.IsEnabled = false;
        filterTextBox.Text = "利用不可能";
    }
    else
    {
        filterTextBox.Text = "";
        filterTextBox.IsEnabled = true;
    }
    // TextChanged を処理し、列にフィルタを適用します。
    filterTextBox.TextChanged += new EventHandler(filterTextBox_TextChanged);
}
```

2. **UnbindCellContent** では、メモリリークを防止するために、テキスト変更ハンドラを削除する必要があります。

Visual Basic

```
Protected Overrides Sub UnbindCellContent(cellContent As FrameworkElement,
column As DataGridColumn)
   Dim filterTextBox = DirectCast(cellContent, C1SearchBox)
   filterTextBox.TextChanged -= New EventHandler(Of TextChangedEventArgs)
(filterTextBox_TextChanged)
End Sub
```

C#

```
protected override void UnbindCellContent(FrameworkElement cellContent,
DataGridColumn column) { var filterTextBox = (C1SearchBox)cellContent;
filterTextBox.TextChanged -= new EventHandler(filterTextBox_TextChanged); }
```

カスタム行の追加

各データ項目やグループのデータを表示するためにデータグリッドで使用される行をカスタム行に置き換えたり、データ項目行

の上端または下端にカスタム行を追加することができます。

データ項目行の置き換え

データグリッドによって生成される行を置き換えるには、**CreatingRow** イベントにハンドラを追加する必要があります。たとえば、次の図では、行がテンプレート行に置き換えられています。

次のコードでは、デフォルト行がテンプレート行に置き換えられます。

Visual Basic

```
Private Sub C1DataGrid_CreatingRow(sender As Object, e As
DataGridCreatingRowEventArgs)
      '項目行かどうかをチェックします(グループ行の場合もあります)。
      If e.Type = DataGridRowType.Item Then
            e.Row = New DataGridTemplateRow() With { _
                  .RowTemplate = DirectCast(Resources("TemplateRow"), DataTemplate) _
            }
      End If
End Sub
```

C#

C#

```
private void C1DataGrid_CreatingRow(object sender, DataGridCreatingRowEventArgs e)
 {
   //項目行かどうかをチェックします(グループ行の場合もあります)。
   if (e.Type == DataGridRowType.Item)
    {
      e.Row = new DataGridTemplateRow()
       {
         RowTemplate = (DataTemplate)Resources["TemplateRow"]
       };
    }
 }
```

行の追加

DataGrid for WPF/Silverlight を使用すると、データの上端または下端に1つ以上の行を追加できます。この機能は、新しい行、集計行、サマリー行、およびフィルタ行のシナリオで使用されます。

たとえば、XAML またはコードで、次の手順に従います。

XAML

```
<datagrid:C1DataGrid>
    <datagrid:C1DataGrid.TopRows>
        <local:DataGridFilterRow />
    </datagrid:C1DataGrid.TopRows>
    <datagrid:C1DataGrid.BottomRows>
        <local:DataGridFilterRow/>
    </datagrid:C1DataGrid.BottomRows>
</datagrid:C1DataGrid>
```

Visual Basic

```
grid.Rows.TopRows.Add(New DataGridFilterRow())
```

C#

C#

```
grid.Rows.TopRows.Add(new DataGridFilterRow());
```

行の詳細の追加

DataGrid for WPF/Silverlight 内の各グリッド行を展開して、行の詳細セクションを表示できます。この行の詳細セクションに、特定の行の内容に関する詳細情報を表示できます。行の詳細セクションは、**DataTemplate** の1つである **RowDetailsTemplate** によって定義されます。これは、表示されるセクションとデータの外観を決定します。

RowDetailsVisibilityMode プロパティを使用すると、選択した行またはすべての行に行の詳細セクションを表示したり、行の詳細セクションを折りたたむことができます。

データ連結

DataGrid for WPF/Silverlight の C1DataGrid コントロールは、**System.Collections.IEnumerable** インターフェイスを実装するすべてのオブジェクト（**XmlDataProvider**、**ObjectDataProvider**、**DataSet**、**DataView** など）に連結できます。C1DataGrid を連結するには、**ItemsSource** プロパティを使用します。

グリッドを連結するには、**ItemsSource** プロパティを **IEnumerable** の実装に設定するだけです。データグリッドの各行は、データソース内の1つのオブジェクトに連結され、データグリッドの各列は、データオブジェクトの1つのプロパティに連結されます。

ソースデータに項目を追加したりソースデータから項目を削除するときに **C1DataGrid** ユーザーインターフェイスを自動更新するには、**INotifyCollectionChanged** を実装するコレクション（**ObservableCollection(Of T)** など）にコントロールを連結する必要があります。

RIA Services データソースへの連結については、「WCF RIA Services のデータ連結」と「WCF RIA Services のデータソースへのグリッドの連結」を参照してください。データ連結の例については、「RSS フィードへのグリッドの連結」と「Web サービスへの

グリッドの連結」を参照してください。C1DataGrid コントロールを XML データソースに連結する手順については、「クイックスタート」を参照してください。

WCF RIA Services のデータ連結 (Silverlight のみ)

DataGrid for Silverlight の C1DataGrid コントロールは、WCF RIA Services の DomainDataSoure に直接連結できます。XAML でコードを記述することなくこれを行う方法が2つあります。DomainDataSource に直接連結するか（フィルタ処理の一部の機能を使用できなくなります）、または Adaptor クラスを使ってグリッドを連結します。

次のような XAML マークアップを使用すると、グリッドを DomainDataSource に直接連結できます。

XAML

```
<ria:DomainDataSource x:Name="_myDataSource" QueryName="GetProducts" PageSize="8">
    <ria:DomainDataSource.DomainContext>
        <local:NorthwindContext/>
    </ria:DomainDataSource.DomainContext>
</ria:DomainDataSource>
<c1:C1DataGrid x:Name="_dataGrid" ItemsSource="{Binding Data,
ElementName=_myDataSource}">
```

この方法でもグリッドを連結して使用できますが、RIA サービスは標準の CollectionView 以外のフィルタ処理方法を使用するので、C1DataGrid の組み込みフィルタ処理機能は使用できません。

すべての機能を維持するには、RIA サービスがフィルタ処理を実行できるように、フィルタ処理情報を "変換" するクラスを追加する必要があります。具体的には、C1RiaAdapter クラスを使用します。このクラスは、RIA で C1DataGrid のフィルタ処理を実行するために必要な変換を実行します。

次のような XAML マークアップを使用します。

DomainDataSource への間接連結（Adapter を使用した連結）

XAML

```
<adapter:C1RiaAdapter x:Name="_adapter" DataGrid="{Binding ElementName=_dataGrid}">
    <ria:DomainDataSource x:Name="_myDataSource" QueryName="GetProducts"
PageSize="8">
        <ria:DomainDataSource.DomainContext>
            <local:NorthwindContext/>
        </ria:DomainDataSource.DomainContext>
    </ria:DomainDataSource>
</adapter:C1RiaAdapter>
<c1:C1DataGrid x:Name="_dataGrid" ItemsSource="{Binding Data, ElementName=_adapter}">
```

この方法でグリッドを連結すると、初期設定のままでフィルタ処理を使用できます。もちろん、フィルタ処理のサポートを必要としない場合は、C1RiaAdapter を使用することなく直接グリッドを連結できます。

例については、「WCF RIA Services のデータソースへのグリッドの連結」を参照してください。

メモ:このトピックの内容は、ComponentOne for Silverlight にのみ適用されます。

遅延スクロールとリアルタイムスクロール

DataGrid for WPF/Silverlight は、遅延スクロールとリアルタイムスクロールの両方をサポートします。デフォルトではリアルタイムスクロールが使用され、ユーザーがサムボタンを移動するかスクロールボタンをクリックすると、グリッドがスクロールします。遅延スクロールでは、スクロールバーのサムボタンを放すまでグリッドはスクロールされません。つまり、グリッドはスク

ロールバーのサムボタンに連動しません。グリッド内のデータが大量である場合、またはスクロールを最適化する場合は、アプリケーションに遅延スクロールを実装することもできます。

ScrollMode プロパティを設定すると、グリッドのスクロール方法を指定できます。次の例では、グリッドを遅延スクロールモードに設定しています。

XAML の場合

グリッドを遅延スクロールモードに設定するには、次のように タグに ScrollMode="Deferred" を追加します。

XAML

```
<datagrid:C1DataGrid x:Name="c1DataGrid1" ScrollMode="Deferred">
```

コードの場合

グリッドを遅延スクロールモードに設定するには、ScrollMode プロパティを **Deferred** に設定します。次に例を示します。

Visual Basic

```
Me.C1DataGrid1.ScrollMode = DataGridScrollMode.Deferred
```

C#

```
this.c1DataGrid1.ScrollMode = DataGridScrollMode.Deferred;
```

列の定義

DataGrid for WPF/Silverlight の **Columns** コレクションを使用して、実行時にプログラムでコントロールに列を追加、挿入、削除、および変更できます。列の自動生成の有無に関係なく、XAML で列を指定することもできます。

独自の列を作成すると、他の列型（**DataGridTemplateColumn** 型やカスタム列型）を使用できます。**DataGridTemplateColumn** タイプを使用すると、単純なカスタム列を簡単な方法で作成できます。**CellTemplate** プロパティと **CellEditingTemplate** プロパティを使用すると、表示モードと編集モードの両方でコンテンツテンプレートを指定できます。

列の生成

デフォルトでは、**C1DataGrid** コントロールは、**ItemsSource** プロパティが設定されたときにデータの型に基づいて列を自動生成します。生成される列のタイプは、連結ブール値プロパティ（および null 可能なブール値プロパティ）に対応する **DataGridCheckBoxColumn**、連結文字列データに対応する **DataGridTextColumn**、連結列挙型データに対応する **DataGridComboBoxColumn**、連結日付／時刻データに対応する **DataGridDateTimeColumn**、および連結数値データに対応する **DataGridNumericColumn** です。連結未定義データは、**DataGridBoundColumn** タイプの列に表示されます。プロパティが文字列型や数値型以外の場合、生成されるテキストボックス列は読み取り専用になり、データオブジェクトの **ToString** 値が表示されます。

列の自動生成を禁止することができます。それには、**AutoGenerateColumns プロパティ**を False に設定します。これは、すべての列を明示的に作成および設定する場合に便利です。また、データグリッドで自動的に列を生成し、**AutoGeneratingColumn** イベントを処理して生成後に列をカスタマイズすることもできます。列の表示順序を並べ替えるには、各列の **DisplayIndex** プロパティを設定します。

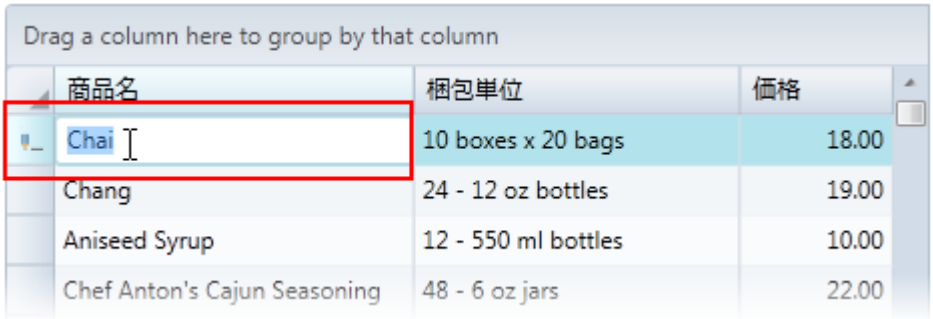
編集

次のトピックでは、**C1DataGrid**の編集機能の使い方について説明します。

セルの編集

セルの内容は、実行時に簡単に編集できます。内容の編集は、セルを選択し、そのセル内の内容を削除したり変更するだけで簡単に実行できます。セルの内容を編集するには、次の手順に従います。

1. 編集するセルをダブルクリックします。



そのセルにカーソルが表示され、編集可能であることがわかります。行インジケータ列には鉛筆アイコンが表示され、その行のセルが編集モードにあることを示します。

2. テキストの削除、新しいテキストの入力、テキストの追加などを行って、セルの内容を編集します。

3. [Enter]キーを押すか、編集中のセルの外をクリックしてセルから移動すると、行った変更が有効になります。

編集中であることを示す鉛筆アイコンは表示されなくなります。

編集を有効にするには、CanUserEditRows プロパティを True（デフォルト）に設定する必要があります。例については、「セルの編集を無効にする」を参照してください。

セルの編集を無効にする

デフォルトでは、エンドユーザーは、実行時にグリッド内のコンテンツを編集できます。詳細については、「セルの編集」を参照してください。しかし、必要に応じて、CanUserEditRows プロパティを **False** に設定することにより、セルの編集機能を無効にすることができます。

設計時

セルの編集を無効にするには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウに移動し、CanUserEditRows プロパティを見つけます。
3. CanUserEditRows プロパティの横にあるチェックボックスを「オフ」にします。

XAML の場合

たとえば、セルの編集を無効にするには、CanUserEditRows="False" を < datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250"
CanUserEditRows="False" />
```

コードの場合

たとえば、セルの編集を無効にするには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.C1DataGrid1.CanUserEditRows = False
```

C#

```
this.c1DataGrid1.CanUserEditRows = false;
```

ここまでの成果

アプリケーションを実行し、1つのセルをダブルクリックします。このセルが編集モードにならず、実行時にグリッドのコンテンツを編集できないことを確認します。セルの編集の詳細については、「セルの編集」トピックを参照してください。

グリッドのロック

デフォルトでは、ユーザーはグリッドやグリッド内の列を操作したり編集することができます。必要に応じて、グリッドまたはグリッド内の特定の列を IsReadOnly プロパティで編集不能に設定できます。

XAML の場合

グリッドの編集をロックするには、<c1:C1DataGrid> タグに IsReadOnly="True" を追加します。次のようになります。

XAML

```
<c1:C1DataGrid x:Name="C1DataGrid1" IsReadOnly="True">
```

コードの場合

グリッドの編集をロックするには、IsReadOnly プロパティを True に設定します。次に例を挙げます。

Visual Basic

```
Me.C1DataGrid1.IsReadOnly = True
```

C#

```
this.c1DataGrid1.IsReadOnly = true;
```

グリッドのフィルタ処理

DataGrid for WPF/Silverlight には、グリッドをフィルタ処理するためのオプションがいくつかあります。列フィルタ処理、フィルタ行、またはフルテキストグリッドフィルタ処理を追加できます。グリッドには基本的なフィルタ処理が組み込まれていますが、高度なフィルタオプションを提供する C1.WPF.DataGrid.Filters.dll または C1.Silverlight.DataGrid.Filters.dll アセンブリを使用することもできます。グリッドをフィルタ処理する方法はニーズによって異なります。たとえば、エンドユーザーが列内のテキストをフィルタ処理するだけの場合や、アプリケーションで高度なカスタムフィルタ処理が必要な場合があります。

基本的な列フィルタ処理

基本的な列フィルタ処理の場合は、**CanUserFilter** プロパティを **True** に設定するだけです。これで、グリッドのユーザーインターフェイスに列フィルタ処理要素が追加され、エンドユーザーは、各列のヘッダーのドロップダウンボックスを使用してグリッドをフィルタ処理できます。

デフォルトでは、**CanUserFilter** プロパティは **True** に設定され、フィルタ処理は有効になります。基本的なフィルタ処理を手作業で有効にする場合は、次のマークアップまたはコードを使用できます。

XAML

```
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" CanUserFilter="True" />
```

Visual Basic

```
Me.C1DataGrid1.CanUserFilter = True
```

C#

```
this.c1DataGrid1.CanUserFilter = true;
```

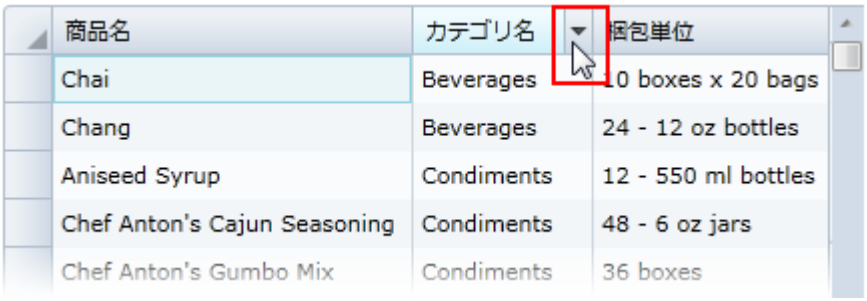
詳細および例については、「列のフィルタ処理」のトピックを参照してください。

列のフィルタ処理

DataGrid for WPF/Silverlight では、ユーザーインターフェイスに列フィルタ処理要素が組み込まれており、実行時に条件を指定して列をフィルタ処理できます

実行時に列のテキストをフィルタ処理するには、次の手順に従います。

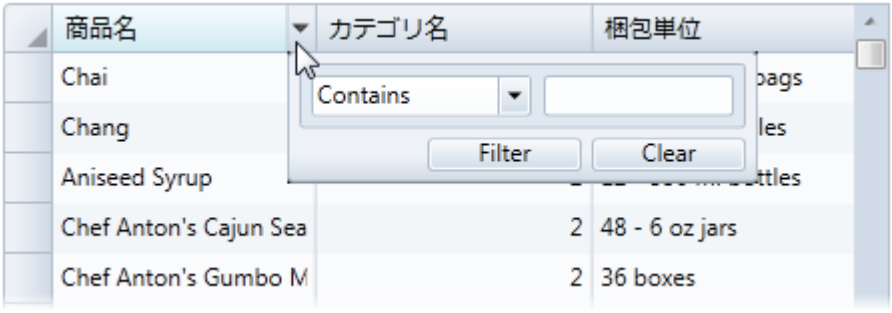
1. テキスト列のヘッダーにあるドロップダウン矢印をクリックします。



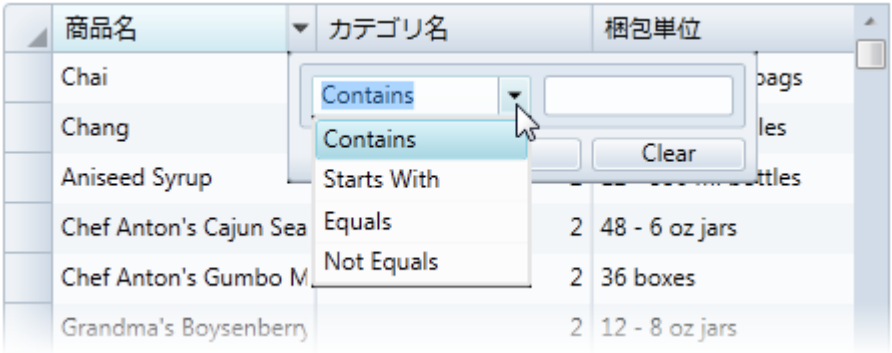
2. フィルタテキストボックスに、列をフィルタ処理するためのテキストを入力し、[フィルタ]ボタンをクリックします。列がソートされます。

フィルタオプションは、列タイプによって異なります。次はフィルタオプションの一部です。

- **テキスト列**
  テキスト列では、フィルタバーが次のように表示されます。

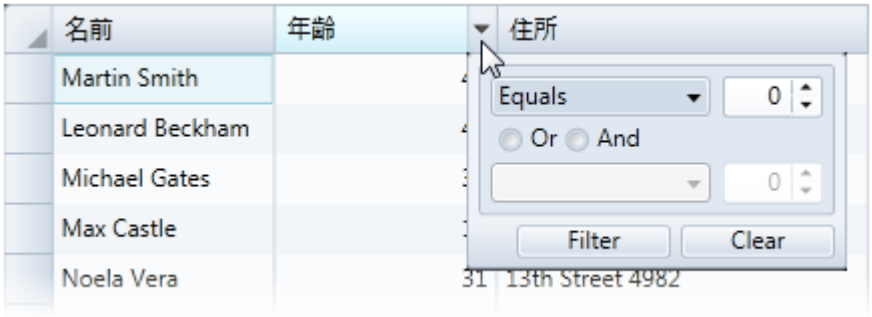


  列内の項目がフィルタ条件を含む、始まる、等しい、等しくないのいずれかで、列のフィルタ処理を行います。

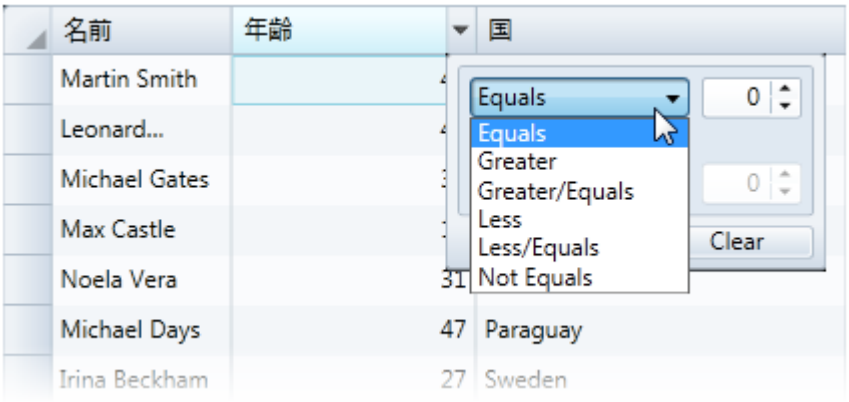


- ブール値列
  列内の項目がオンかオフかで、ブール値チェックボックス列のフィルタ処理を行います。

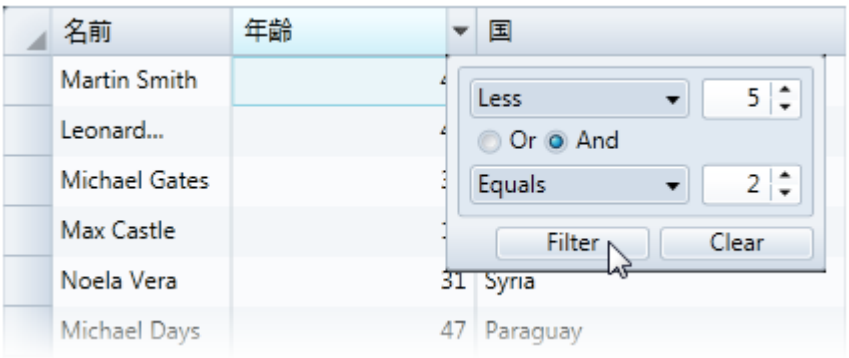
- 数値列
  数値列にはフィルタ処理のオプションが複数あります。



列を特定の条件でフィルタ処理できます。



また、[And]ラジオボタンと[Or]ラジオボタンを使用して、複数の条件でフィルタ処理できます。



フィルタ処理を有効にするには、CanUserFilter プロパティを True（デフォルト）に設定する必要があります。

フィルタ行フィルタ処理

必要な場合は、フィルタ行をグリッドの上部または下部に追加して表示できます。フィルタ行は、各セルにテキストボックスがある1つの行として表示されます。テキストボックスにテキストを入力すると、入力されたテキストに基づいてフィルタ処理されます。

たとえば、次のマークアップは、グリッドの上部と下部に1つずつフィルタ行を追加します。

XAML

```
<c1:C1DataGrid x:Name="grid" Grid.Row="1" CanUserAddRows="False"
CanUserFreezeColumns="True" FrozenTopRowsCount="1" FrozenBottomRowsCount="1"
RowHeight="30" >
    <c1:C1DataGrid.TopRows>
        <c1:DataGridFilterRow />
    </c1:C1DataGrid.TopRows>
    <c1:C1DataGrid.BottomRows>
        <c1:DataGridFilterRow/>
    </c1:C1DataGrid.BottomRows>
</c1:C1DataGrid>
```

フルテキストグリッドフィルタ処理

C1DataGrid は、グリッド全体のフルテキストフィルタ処理もサポートします。データグリッドに1つの添付プロパティを設定することで、エンドユーザーが外部のテキストボックスに入力したテキストでデータグリッド全体（すべての列を一度に）をフィルタ処理できるようになります。キー入力のたびに、グリッド内のすべての一致する結果が強調表示されます。

このグリッドフィルタ処理を使用するには、アプリケーションにテキストボックスコントロールを追加し、**FullTextSearchBehavior** 添付プロパティでそのコントロールを参照します。たとえば、XAML マークアップは次のようになります。

XAML

```
<StackPanel>
    <c1:C1TextBoxBase x:Name="filterTextBox" Width="200" Watermark = "フィルタするテキストはここに入力"/>
    <c1:C1DataGrid x:Name="c1DataGrid1" c1:C1NagScreen.Nag="True">

    <c1:C1FullTextSearchBehavior.FullTextSearchBehavior>
        <c1:C1FullTextSearchBehavior Filter="{Binding ElementName=filterTextBox,Path=C1Text}"/>

    </c1:C1FullTextSearchBehavior.FullTextSearchBehavior>
</c1:C1DataGrid>
</StackPanel>
```

高度なフィルタ処理

高度なフィルタ処理を追加する場合は、**C1AdvancedFiltersBehavior** を使用します。**C1AdvancedFiltersBehavior** は、さまざまな高度なフィルタ処理を **C1DataGrid** の組み込み列に追加します。たとえば、この Behavior クラスはいくつかの定義済みフィルタを追加し、それらのオプションを各列で展開します。

XAML

```
<c1:C1DataGrid>

    <c1:C1AdvancedFiltersBehavior.AdvancedFiltersBehavior>
        <c1:C1AdvancedFiltersBehavior/>

    </c1:C1AdvancedFiltersBehavior.AdvancedFiltersBehavior>
</c1:C1DataGrid>
```

列フィルタリスト

グリッドをフィルタ処理するためのオプションとして、XAML で列にフィルタリストを追加できます。たとえば、次のマークアップは、RangeFilter という名前のカスタムフィルタを含む3つのフィルタを数値列に追加します。

XAML

```
<c1:DataGridNumericColumn Header="Range filter" Binding="{Binding StandardCost}" FilterMemberPath="StandardCost">
    <c1:DataGridNumericColumn.Filter>
        <c1:DataGridContentFilter>
            <c1:DataGridFilterList>
                <local:DataGridRangeFilter Minimum="0" Maximum="1000"/>
                <c1:DataGridNumericFilter/>
                <c1:DataGridTextFilter/>
            </c1:DataGridFilterList>
        </c1:DataGridContentFilter>
    </c1:DataGridNumericColumn.Filter>
</c1:DataGridNumericColumn>
```

列のフィルタ処理を無効にする

デフォルトでは、エンドユーザーは、実行時にグリッド内の列をフィルタ処理できます。詳細については、「列のフィルタ処理」を参照してください。しかし、必要に応じて、CanUserFilter プロパティを False に設定することにより、列のフィルタ処理機能を無効にすることができます。

設計時

列のフィルタ処理を無効にするには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウに移動し、CanUserFilter プロパティを見つけます。
3. CanUserFilter プロパティの横にあるチェックボックスを「オフ」にします。

XAML の場合

たとえば、列のフィルタ処理を無効にするには、CanUserFilter="False" を <datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250"
CanUserFilter="False" />
```

コードの場合

たとえば、列のフィルタ処理を無効にするには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.C1DataGrid1.CanUserFilter = False
```

C#

```
this.c1DataGrid1.CanUserFilter = false;
```

ここまでの成果

アプリケーションを実行し、実行時に列のフィルタ処理を行うことができないこと、つまり実行時にフィルタボックスを表示するためのドロップダウン矢印が表示されないことを確認します。列のフィルタ処理の詳細については、「列のフィルタ処理」トピックを参照してください。

ベクターオブジェクト

グリッドをフィルタ処理するためのオプションとして、タブコントロールに表示されるフィルタリストを追加できます。

XAML

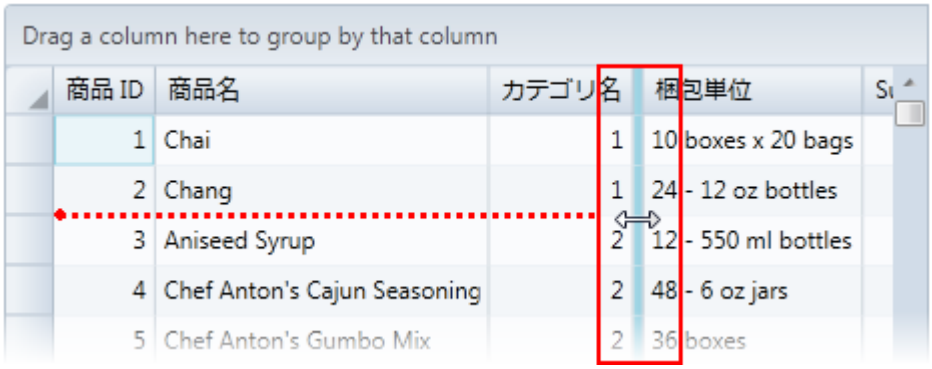
```
<c1:DataGridNumericColumn Header="タブコントロール内のフィルタ" Binding="{Binding
StandardCost}" FilterMemberPath="StandardCost">
    <c1:DataGridNumericColumn.Filter>
        <c1:DataGridContentFilter>
            <local:DataGridTabFilters Width="250">
                <local:DataGridRangeFilter Minimum="0" Maximum="1000"/>
```

```
                <c1:DataGridNumericFilter/>
                <c1:DataGridTextFilter/>
            </local:DataGridTabFilters>
        </c1:DataGridContentFilter>
    </c1:DataGridNumericColumn.Filter>
</c1:DataGridNumericColumn>
```

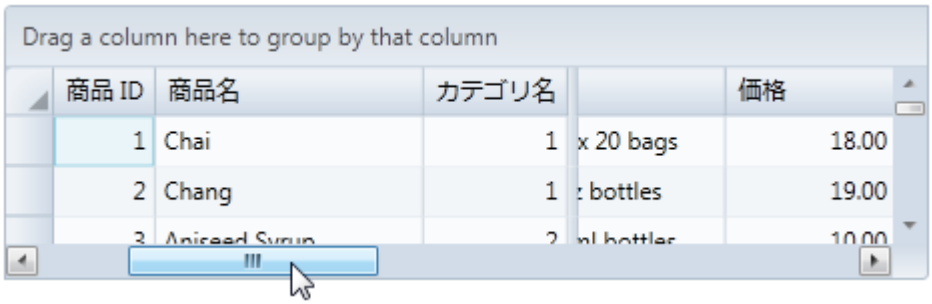
列のフリーズ

列が水平方向にスクロールされないように、実行時に列をフリーズできます。これは、グリッドをサイズ変更またはスクロールする際に、特定の列を表示したままにできるので便利です。ユーザーは、フリーズバーを使って列をフリーズできます。フリーズバーを表示可能にすると、デフォルトで最初の列の左側に表示されます。

特定の列をフリーズするには、フリーズする列の右側までフリーズバーを移動します。たとえば、次の図では、フリーズバーを2番目の列の右側まで移動しました。



フリーズされた列は、グリッドを水平方向にスクロールしてもスクロールされません。たとえば、次の図では、最初から2つの列がフリーズされています。



フリーズバーを表示するには、**ShowVerticalFreezingSeparator** プロパティが **Left**（デフォルトは **None**）に設定されている必要があります。また、ユーザーが実行時に列をフリーズできるようにするには、**CanUserFreezeColumns** プロパティが **Left**（デフォルトは **None**）に設定されている必要があります。サンプルについては、「列のフリーズを有効にする」を参照してください。

列のフリーズを有効にする

実行時にグリッドの列をフリーズして、グリッドを水平方向にスクロールしてもそれらが常に表示されるようにすることができます。詳細については、「列のフリーズ」を参照してください。デフォルトでは、この機能は有効ではありませんが、必要に応じて、CanUserFreezeColumns プロパティを Left に設定して、列のフリーズ機能を有効にすることができます。

設計時

列のフリーズを有効にするには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウに移動し、CanUserFreezeColumns プロパティを見つけます。
3. CanUserFreezeColumns プロパティの横にあるドロップダウン矢印をクリックし、**Left** を選択します

XAML の場合

たとえば、列のフリーズを有効にするには、CanUserFreezeColumns="Left" を <datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250"
CanUserFreezeColumns="Left" />
```

コードの場合

たとえば、列のフリーズを有効にするには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.C1DataGrid1.CanUserFreezeColumns = DataGridColumnFreezing.Left
```

C#

```
this.c1DataGrid1.CanUserFreezeColumns = DataGridColumnFreezing.Left;
```

ここまでの成果

アプリケーションを実行し、実行時にフリーズバーが表示されることを確認します。フリーズバーを移動することによってフリーズ対象の列を選択できます。このバーよりも左にある列はフリーズされて、グリッドを水平方向にスクロールしても常に表示されます。列のフリーズの詳細については、「列のフリーズ」トピックを参照してください。

行のフリーズ

グリッドの上端または下端の数行をフリーズして、実行時にグリッドを垂直方向にスクロールしてもそれらが常に表示されるようにすることができます。デフォルトでは、この機能は有効ではありませんが、必要に応じて、FrozenTopRowsCount プロパティと FrozenBottomRowsCount プロパティを設定して、行のフリーズ機能を有効にすることができます。

設計時

上端および下端からの2行をフリーズするには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択し、[**プロパティ**]ウィンドウに移動します。
2. [**プロパティ**]ウィンドウで、FrozenTopRowsCount プロパティを見つけて、このプロパティの横にあるテキストボックスをクリックし、上端にあるフリーズ対象の行数を設定するために「2」と入力します。
3. FrozenBottomRowsCount プロパティを見つけて、このプロパティの横にあるテキストボックスをクリックし、下端にあるフリーズ対象の行数を設定するために「2」と入力します。

XAML の場合

たとえば、上端または下端からの2行をフリーズするには、FrozenTopRowsCount="2" FrozenBottomRowsCount="2" を <datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250"
FrozenTopRowsCount="2" FrozenBottomRowsCount="2" />
```

コードの場合

たとえば、上端または下端からの2行をフリーズするには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.C1DataGrid1.FrozenTopRowsCount = True
Me.C1DataGrid1.FrozenBottomRowsCount = True
```

C#

```
this.c1DataGrid1.FrozenTopRowsCount = true;
this.c1DataGrid1.FrozenBottomRowsCount = true;
```

ここまでの成果

アプリケーションを実行し、上端または下端からの2行がフリーズされていることを確認します。グリッドを垂直方向にスクロールし、上端または下端からの2行がスクロールされず、元の位置にロックされていることを確認します。デフォルトでは、グリッドの最後の行として新規追加行が表示されるので、この行がフリーズ行の1行になります。

グループ化を有効にする

グループ化およびグリッドのグループ化領域を有効にすることにより、実行時にユーザーがグリッドの列をグループ化して、情報をわかりやすく整理できるようにすることができます。詳細については、「列のグループ化」を参照してください。デフォルトでは、ユーザーはグリッドの列をグループ化できませんが、CanUserGroup プロパティを **True** に設定することにより、この機能を有効にできます。

設計時

グループ化を有効にするには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウに移動し、CanUserGroup プロパティを見つけます。
3. CanUserGroup プロパティの横にあるチェックボックスを「オン」にします。

XAML の場合

たとえば、グループ化を有効にするには、CanUserGroup="True" を <datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250" CanUserGroup=True"
/>
```

コードの場合

たとえば、グループ化を有効にするには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.C1DataGrid1.CanUserGroup = True
```

C#

```
this.c1DataGrid1.CanUserGroup = true;
```

ここまでの成果

アプリケーションを実行し、グリッドの上部にグループ化領域が表示されることを確認します。グループ化領域の表示状態をカスタマイズすることもできます。グループ化領域の詳細については、「グループ化領域の表示」トピックを参照してください。

グループ化領域の表示

デフォルトでは、グリッドでのグループ化は無効であり、グループ化領域は表示されません。詳細については、「列のグループ化」を参照してください。CanUserGroup プロパティを **True** に設定し、グループ化を有効にしてあると、グループ化領域が表示されます。ただし、必要に応じて、グループ化が有効かどうかに関係なく、グループ化領域の表示と非表示を切り替えることができます。デフォルトでは、グループ化が有効でない場合、グループ化領域は表示されませんが、ShowGroupingPanel プロパティを **True** に設定して、この領域を表示することができます。

設計時

グループ化領域を表示するには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウに移動し、ShowGroupingPanel プロパティを見つけます。
3. ShowGroupingPanel プロパティの横にあるチェックボックスを「オン」にします。

XAML の場合

たとえば、グループ化領域を表示するには、ShowGroupingPanel="True" を <datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1DataGrid1" Height="180" Width="250"
ShowGroupingPanel="True" />
```

コードの場合

たとえば、グループ化領域を表示するには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.C1DataGrid1.ShowGroupingPanel = True
```

C#

```
this.c1DataGrid1.ShowGroupingPanel = true;
```

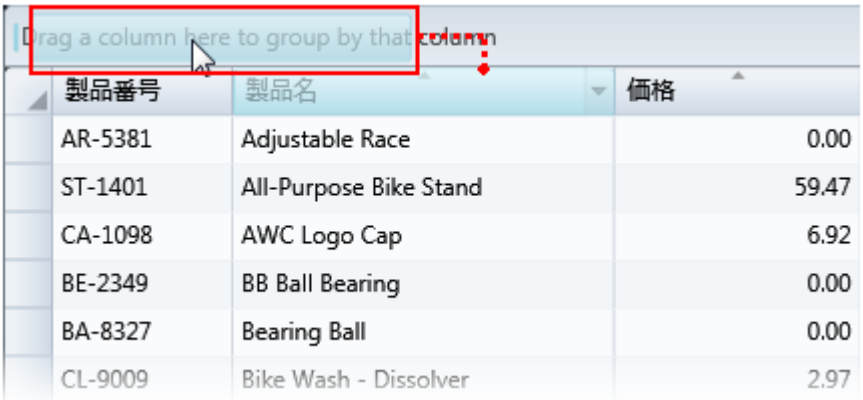
ここまでの成果

アプリケーションを実行し、グリッドの上部にグループ化領域が表示されることを確認します。グループ化領域が表示されている場合でも、CanUserGroup プロパティが **False** の場合、グループ化は有効ではありません。詳細については、「グループ化を有効にする」を参照してください。

列のグループ化

グリッド内の列を実行時にグループ化して、情報をわかりやすく整理できます。グリッド上部にあるグループ化領域を使用すると、簡単なドラッグアンドドロップ操作で列を簡単にグループ化できます。

列をグループ化するには、列ヘッダーをグループ化領域にドラッグします。



グループ化された項目の表示をソートするには、グループ化領域にある列ヘッダーをクリックします。次の図では、グループ化

された列が逆順にソートされています。

ドラッグアンドドロップ操作で追加の列をグループ化領域にドラッグすると、複数の列をグループ化できます。

グループ化を解除するには、グリッドのグループ化領域で、グループ化された列の横にある[X]ボタンをクリックするだけです。

グループ化領域を表示し、グループ化を有効にするには、**CanUserGroup** プロパティを True に設定する必要があります（デフォルトは **False**）。詳細については、「グループ化を有効にする」を参照してください。グループ化領域の表示の詳細については、「グループ化領域の表示」トピックを参照してください。

グリッドの集計

DataGrid for WPF/Silverlight には、グリッドの機能を強化するためにサマリー行を追加できる C1.WPF.DataGrid.Summaries.dll または C1.Silverlight.DataGrid.Summaries.dll アセンブリが含まれます。

Summaries アセンブリには、次の機能が含まれます。

- SummaryRow - 各列に対応する集計関数を表示する行。（C1DataGrid_Demo2010/Grouping/GrandTotal.xaml を参照）
- GroupRowWithSummaries - サマリーを通常の行ではなくグループ行に表示すること以外は前の機能と同じ。（C1DataGrid_Demo2010/Grouping/Grouping.xaml を参照）

キーボードとマウスによる移動

DataGrid for WPF/Silverlight は、実行時のキーボードとマウスによる移動のオプションを複数サポートして、ユーザー補助機能を高めています。以下のトピックでは、これらのエンドユーザー操作の一部について詳しく説明します。

キーボードによる移動

次の表は、キーボードのショートカットを示します。これらのショートカットを使用して、実行時にグリッド内を移動したりグリッドを操作することができます。Apple のコンピュータでは、[Ctrl]キーの代わりにコマンドキー（またはアップルキー）を使用する必要があります。

キーの組み合わせ	説明
↓	フォーカスを現在のセルの真下のセルに移動します。フォーカスが最後の行にある場合は、[↓]キーを押してもフォーカスは移動しません。
↑	フォーカスを現在のセルの真上のセルに移動します。フォーカスが最初の行にある場合は、[↑]キーを押してもフォーカスは移動しません。
←	フォーカスを行内の前のセルに移動します。フォーカスが行の最初のセルにある場合は、[←]キーを押してもフォーカスは移動しません。
→	フォーカスを行内の次のセルに移動します。フォーカスが行の最後のセルにある場合は、[→]キーを押してもフォーカスは移動しません。
Home	フォーカスを現在の行内の最初のセルに移動します。
End	フォーカスを現在の行内の最後のセルに移動します。
Page Down	コントロールを表示されている行数分下にスクロールします。フォーカスは、最後に表示されている行の同じ列に移動します。最後の行の一部が表示されていない場合は、最後の行が完全に表示されるまでグリッドがスクロールします。
Page Up	コントロールを表示されている行数分上にスクロールします。フォーカスは、最初に表示されている行の同じ列に移動します。最初の行の一部が表示されていない場合は、最初の行が完全に表示されるまでグリッドがスクロールします。
Tab	現在のセルが編集モードにある場合は、フォーカスを現在の行の次の編集可能なセルに移動します。フォーカスが既に行の最後のセルにある場合は、実行された変更をコミットし、フォーカスを次の行の最初の編集可能

	なセルに移動します。フォーカスがコントロールの最後のセルにある場合は、フォーカスを親コンテナのタブオーダーで次のコントロールに移動します。現在のセルが編集モードにない場合は、フォーカスを親コンテナのタブオーダーで次のコントロールに移動します。
Shift+Tab	現在のセルが編集モードにある場合は、フォーカスを現在の行の前の編集可能なセルに移動します。フォーカスが既に行の最初のセルにある場合は、実行された変更をコミットし、フォーカスを前の行の最後のセルに移動します。フォーカスがコントロールの最初のセルにある場合は、フォーカスを親コンテナのタブオーダーで前のコントロールに移動します。 現在のセルが編集モードにない場合は、フォーカスを親コンテナのタブオーダーで前のコントロールに移動します。
Ctrl+↓	フォーカスを現在の列内の最後のセルに移動します。
Ctrl+↑	フォーカスを現在の列内の最初のセルに移動します。
Ctrl+→	フォーカスを現在の行内の最後のセルに移動します。
Ctrl+←	フォーカスを現在の行内の最初のセルに移動します。
Ctrl+Home	フォーカスをコントロール内の最初のセルに移動します。
Ctrl+Page Down	Page Down と同じです。
Ctrl+Page Up	Page Up と同じです。
Enter	選択されたセルで編集モードに入るか、編集モードを終了します（グリッドと列の **IsReadOnly** プロパティが **False** の場合）。
F2	選択されたセルで編集モードに入ります（グリッドと列の **IsReadOnly** プロパティが **False** の場合）。フォーカスが新規行にある場合、新規行の最初の編集可能なセルの編集を開始します。
Esc	セルまたは新規行の編集をキャンセルします。
Del	選択された行を削除します。
Insert	新規行までスクロールし、編集を開始します。

マウスによる移動

次の表は、マウスとキーボードのショートカットを示します。これらのショートカットを使用して、実行時にグリッド内を移動したりグリッドを操作することができます。Apple のコンピュータでは、[Ctrl]キーの代わりにコマンドキー（またはアップルキー）を使用する必要があります。

マウス操作	説明
未選択の行をクリック	クリックされた行を現在の行にします。
現在の行内のセルをクリック	クリックされたセルを編集モードにします。
列ヘッダーのセルをドラッグ	新しい位置にドロップできるように列を移動します（**CanUserReorderColumns** プロパティが True で、現在の列の **CanUserReorder** プロパティが **True** の場合）。
列ヘッダーのセパレータをドラッグ	列をサイズ変更します（**CanUserResizeColumns** プロパティが **True** で、現在の列の **CanUserResize** プロパティが True の場合）。

列ヘッダーのセルをクリック	**ColumnHeaderClickAction** プロパティが **Sort** に設定されているとき、ユーザーが列のヘッダーをクリックすると、その列がソートされます（**CanUserSortColumns** プロパティが **True** に設定され、現在の列の **CanUserSort** プロパティが **True** に設定されている場合）。既にソートされている列のヘッダーをクリックすると、その列のソート方向が逆順になります。[Ctrl]キーを押しながら複数の列ヘッダーをクリックすると、クリックした順番に複数の列がソートされます。プロパティ **ColumnHeaderClickAction** が **Select** に設定されているときは、列が選択されます（**SelectionMode** で列の選択がサポートされている場合）。
Ctrl+行をクリック	連続しない複数の行の選択を変更します（**SelectionMode** で複数行、セル、または列の選択がサポートされている場合）。
Shift+行をクリック	連続する複数の行の選択を変更します（**SelectionMode** で複数行、セル、または列の選択がサポートされている場合）。

複数行の選択

SelectionMode プロパティに **MultiRow** を設定しても、移動動作は変わりませんが、[Shift]キー（[Ctrl]+[Shift]も含む）を押しながらキーボードやマウスで移動すると、複数行が選択されます。移動を開始する前に、コントロールは現在の行にアンカー行のマークを付けます。[Shift]キーを押しながら移動すると、アンカー行と現在の行の間のすべての行が選択されます。

選択キー

次の選択キーは複数行を選択します。

- Shift+↓
- Shift+↑
- Shift+Page Down
- Shift+Page Up
- Ctrl+Shift+↓
- Ctrl+Shift+↑
- Ctrl+Shift+Page Down
- Ctrl+Shift+Page Up

マウスによる選択

SelectionMode プロパティに **MultiRow** が設定されている場合、[Ctrl]キーまたは[Shift]キーを押しながら行をクリックすると、複数行が選択されます。

[Shift]キーを押しながら行をクリックすると、現在の行と、最初にクリックする前に現在の行の位置にあったアンカー行との間のすべての行が選択されます。[Shift]キーを押しながらクリックを続けると、現在の行は変わりますが、アンカー行は変わりません。

ナビゲーションの際、[Ctrl]キーが押されていると、矢印キーを使って境界セルまで移動できます。たとえば、最初の行で[Ctrl]キーを押しながら下向き矢印キーを押すと、最後の行まで移動します。[Shift]キーを押しながら矢印キーを押すと、移動先までのすべての行が選択されます。

キーボードによる移動のカスタマイズ

独自のカスタム移動を **C1DataGrid** コントロールに追加できます。カスタムキーボード移動を使用すると、ユーザーがグリッドを操作する方法を制御できます。たとえば、読み取り専用の列や null 値が入ったセルに移動することを禁止できます。階層グリッドでは、親グリッドと子グリッドの間の移動を設定することもできます。カスタムキーボード移動を追加するには、**KeyDown** イベントを処理し、カスタマイズした移動でデフォルトの移動を上書きするためのコードを追加する必要があります。

KeyDown イベントハンドラの追加

次の手順に従って、**KeyDown** イベントハンドラを追加します。

1. **[コード]**ビューに切り替え、**KeyDown** イベントのイベントハンドラを追加します。たとえば、次のようになります。

Visual Basic

```
Private Sub C1DataGrid1_KeyDown(ByVal sender As System.Object, ByVal e As

System.Windows.Input.KeyEventArgs) Handles C1DataGrid1.KeyDown
    ' コードをここに追加します。
End Sub
```

C#

```
private void c1DataGrid1_KeyDown(object sender, KeyEventArgs e)
{
    // コードをここに追加します。
}
```

2. **[ソース]**ビューに切り替え、イベントハンドラを **C1DataGrid** コントロールのインスタンスに追加します。たとえば、次のようになります。

XAML

```
<datagrid:C1DataGrid x:Name="c1DataGrid1" AutoGenerateColumns="True"

KeyDown="c1DataGrid1_KeyDown"></datagrid:C1DataGrid>
```

これで、デフォルトの移動をカスタマイズするためのコードを **KeyDown** イベントハンドラに追加できます。たとえば、「高度な機能」サンプルにあるカスタム列の例を参照してください。

アプリケーションのローカライズ

DataGrid for WPF/Silverlight では、エンドユーザーに表示される文字列をローカライズ（翻訳）できます。DataGrid for WPF/Silverlight によるローカライズは、.NET Windows フォームの標準のローカライズと同じ方法に基づきます。

アプリケーションをローカライズするには、次の手順に従う必要があります。

1. サポートするカルチャごとにリソースファイルを追加します。リソースファイルの追加をご参照ください。
2. サポートするカルチャをプロジェクトファイルで更新します。リソースファイルの追加をご参照ください。
3. プロジェクトに応じて、現在のカルチャを設定します。現在のカルチャの設定をご参照ください。

リソースファイルの追加

Windows フォームと同様に、**DataGrid for WPF/Silverlight** アセンブリに1組のリソースファイルを作成できます。必要なカルチャごとに個別のリソースファイル（拡張子 .resx）を作成できます。アプリケーションの実行時に、これらのリソースや言語を切り替えることができます。1つの DataGrid for WPF/Silverlight DLL のコンポーネントを使用するアプリケーションのすべての部分で、同じローカライズリソースを使用する必要があります。

ローカライズの表記規則

グリッドをローカライズするには、ローカライズするカルチャごとにリソースファイルを設定する必要があります。.resx リソースファイルの作成時には、次の表記規則の使用をお勧めします。

- すべての .resx ファイルをプロジェクトの **Resources** サブフォルダに置く必要があります。
- ファイル名は「XXX.YYY.resx」のように指定します。
  各部分の意味は次のとおりです。
  - XXX は、**ComponentOne for WPF/Silverlight** アセンブリの名前です。
  - YYY は、リソースのカルチャコードです。翻訳が不変カルチャだけに使用される場合は、.resx ファイルの名前の末尾にカルチャ名を付ける必要はありません。

  次に例を挙げます。
  - C1.WPF.DataGrid.de.resx または C1.Silverlight.DataGrid.de.resx - C1.WPF.DataGrid または C1.Silverlight.DataGrid アセンブリのドイツ語（de）リソース。
  - C1.WPF.DataGrid.resx または C1.Silverlight.DataGrid.resx - C1.WPF.DataGrid または C1.Silverlight.DataGrid アセンブリの不変カルチャリソース。

ローカライズ文字列

次の表は、アプリケーションをローカライズするために .resx ファイルに追加できる文字列を示します。

文字列	デフォルト値	説明
AddNewRow	Click here to add new row	新規追加行に表示されるテキスト。
CheckBoxFilter_Checked	Checked:	チェックボックス列のフィルタに表示されるテキスト。オンまたはオフに設定された項を基準に列をフィルタ処理することを示します。
ComboBoxFilter_SelectAll	SelectAll	チェックボックス列のフィルタに表示されるテキスト。すべての項目を選択します。
DateTimeFilter_End	End	日時列のフィルタに表示されるテキスト。日時範囲の終了を表します。
DateTimeFilter_Start	Start	日時列のフィルタに表示されるテキスト。日時範囲の開始を表します。
EmptyGroupPanel	Drag a column here to group by that column	グリッドのグループ化領域に表示されるテキスト。列がグループ化されていない場合に表示されます。
Filter_Clear	Clear	フィルタバーに表示されるテキスト。フィルタ条件をクリアします。
Filter_Filter	Filter	フィルタバーに表示されるテキスト。フィルタ条件を追加します。
NumericFilter_And	And	数値列のフィルタバーに表示されるテキスト。複数のフィルタ条件を示します。
NumericFilter_Equals	Equals	数値列のフィルタバーに表示されるテキスト。正確な一致にのみフィルタ条件を適用することを示します。
NumericFilter_GraterorEquals	Greater/Equals	数値列のフィルタバーに表示されるテキスト。条件値より大きい値を持つか正確に一致する項目にフィルタ条件を適用することを示します。
NumericFilter_Greater	Greater	数値列のフィルタバーに表示されるテキスト。条件値より大きい値を持つ項目にフィルタ条件を適用することを示します。
NumericFilter_Less	Less	数値列のフィルタバーに表示されるテキスト。条件値より小さい値を持つ項目にフィルタ条件を適用することを示します。
NumericFilter_LessorEquals	Less/Equals	数値列のフィルタバーに表示されるテキスト。条件値より小さい値を持つか正確に一致する項目にフィルタ条件を適用することを示します。

NumericFilter_NotEquals	Not Equals	数値列のフィルタバーに表示されるテキスト。正確に一致しない項目にフィルタ条件を適用することを示します。
NumericFilter_Or	Or	数値列のフィルタバーに表示されるテキスト。複数のフィルタ条件を示します。
TextFilter_Contains	Contains	テキスト列のフィルタに表示されるテキスト。条件値を含む項目にフィルタ条件を適用することを示します。
TextFilter_Equals	Equals	テキスト列のフィルタバーに表示されるテキスト。正確な一致にのみフィルタ条件を適用することを示します。
TextFilter_NotEquals	NotEquals	テキスト列のフィルタバーに表示されるテキスト。正確に一致しない項目にフィルタ条件を適用することを示します。
TextFilter_StartsWith	StartsWith	テキスト列のフィルタに表示されるテキスト。条件値で始まる項目にフィルタ条件を適用することを示します。

サポートするカルチャの追加

アプリケーションのリソースファイルを作成したら、サポートするカルチャをプロジェクトに設定します。それには、次の手順に従います。

1. ソリューションエクスプローラで、プロジェクトを右クリックし、[プロジェクトのアンロード]を選択します。
   プロジェクトは淡色表示され、使用できなくなります。
2. プロジェクトをもう一度クリックし、[ProjectName.csproj の編集]オプション(もしくは[ProjectName.vbproj の編集]オプション)を選択します(ProjectName はプロジェクトの名前)。
3. .csproj または .vbproj ファイルで、<SupportedCultures> </SupportedCultures> タグを見つけます。タグとタグの間に、サポートするカルチャをリストし、各カルチャをセミコロンで区切ります。
   次に例を挙げます。

   XAML
   ```
   <SupportedCultures>fr;es;en;it;ru</SupportedCultures>
   ```

   これは、フランス語、スペイン語、英語、イタリア語、およびロシア語をサポートします。
4. .csproj または .vbproj ファイルを保存して閉じます。
5. ソリューションエクスプローラで、プロジェクトを右クリックし、コンテキストメニューから[プロジェクトの再ロード]を選択します。
   プロジェクトが再ロードされ、指定されたカルチャがサポートされるようになります。

現在のカルチャの設定

ファイルを別の場所に移動したり、ファイルをプロジェクトから除外しない限り、**C1DataGrid** コントロールは、アプリケーションで選択されたカルチャに基づいて自動的にローカライズファイルを使用します。デフォルトでは、現在のカルチャが **System.Threading.Thread.CurrentThread.CurrentUICulture** で指定されます。現在のカルチャ以外のカルチャを使用する場合は、次のコードを使用して、アプリケーションにで使用するカルチャを設定できます。

Visual Basic

```
Public Sub New()

    ' 使用するカルチャを設定します。たとえば、ここでは、フランス語(フランス)のロケールを設定します。

 System.Threading.Thread.CurrentThread.CurrentUICulture = New
```

```
System.Globalization.CultureInfo("fr-FR")

    ' InitializeComponent() の呼び出し。

    InitializeComponent()

    ' InitializeComponent() 呼び出しの後に初期化を追加します。

End Sub
```

C#

```
public MainPage()

{

    // 使用するカルチャを設定します。たとえば、ここでは、フランス語（フランス）のロケールを設定します。

    System.Threading.Thread.CurrentThread.CurrentUICulture = new
System.Globalization.CultureInfo("fr-FR");

    // InitializeComponent() の呼び出し。

    InitializeComponent();

    // InitializeComponent() 呼び出しの後に初期化を追加します。

}
```

行の詳細

RowDetailsTemplate テンプレート

RowDetailsTemplate テンプレートは、行の詳細領域の外観を制御します。行の詳細セクションは、行の下に表示され、追加情報を表示します。

MS Expression Blend では、設計時に空のテンプレートを作成できます。それには、**[C1DataGrid]**コントロールを選択し、**[オブジェクト]→[追加テンプレートの編集]→[RowDetailsTemplate の編集]→[空アイテムの作成]**をクリックします。

RowDetailsTemplate には、テキスト、コントロールなど（データに連結されたコントロールも含む）を入れることができます。たとえば、次のテンプレートには、連結および非連結のテキストとチェックボックスが含まれます。

XAML

```
<datagrid:C1DataGrid.RowDetailsTemplate>
    <!-- 行の詳細セクションを開始します。-->
    <DataTemplate>
        <Border BorderBrush="DarkGray" BorderThickness="1" Background="Azure">
        <StackPanel Orientation="Horizontal">
            <StackPanel>
                <StackPanel Orientation="Horizontal">
                <!-- コントロールはプロパティに連結されます。-->
```

```
                <TextBlock FontSize="16" Foreground="MidnightBlue" Text="{Binding
Name}" Margin="0,0,10,0" VerticalAlignment="Bottom" />
                <TextBlock FontSize="12" Text="注文日:" VerticalAlignment="Bottom"/>
                <TextBlock FontSize="12" Text="    完了:" VerticalAlignment="Bottom"
/>
                <CheckBox IsChecked="{Binding Complete, Mode=TwoWay}"
VerticalAlignment="Center" />
                </StackPanel>
                <TextBlock FontSize="12" Text="備考:" />
                <TextBox FontSize="12" Text="{Binding Notes, Mode=TwoWay}"
Width="420" TextWrapping="Wrap"/>
            </StackPanel>
        </StackPanel>
        </Border>
    </DataTemplate>
    <!-- 行の詳細セクションを終了します。-->
</datagrid:C1DataGrid.RowDetailsTemplate>
```

詳細行の切り替えを無効にする

グリッドに子グリッドが含まれている場合、またはマスター/詳細グリッドを作成した場合、デフォルトでは、詳細行を切り替えることによって表示するか折りたたむことができます。しかし、必要に応じて、**CanUserToggleDetails** プロパティを False に設定することにより、詳細行の切り替え機能を無効にすることができます。このサンプルで変更を確認するには、詳細行を含むグリッドが必要です。

設計時

詳細行の切り替えを無効にするには、次の手順に従います。

1. **C1DataGrid** コントロールをクリックして選択します。
2. [プロパティ]ウィンドウに移動し、**CanUserToggleDetails** プロパティを見つけます。
3. **CanUserToggleDetails** プロパティの横にあるチェックボックスをオフにします。

XAML の場合

たとえば、詳細行の切り替えを無効にするには、**CanUserToggleDetails="False"** を <c1:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250"
CanUserToggleDetails="False" />
```

コードの場合

たとえば、詳細行の切り替えを無効にするには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.C1DataGrid1.CanUserToggleDetails = False
```

C#

```
this.c1DataGrid1.CanUserToggleDetails = false;
```

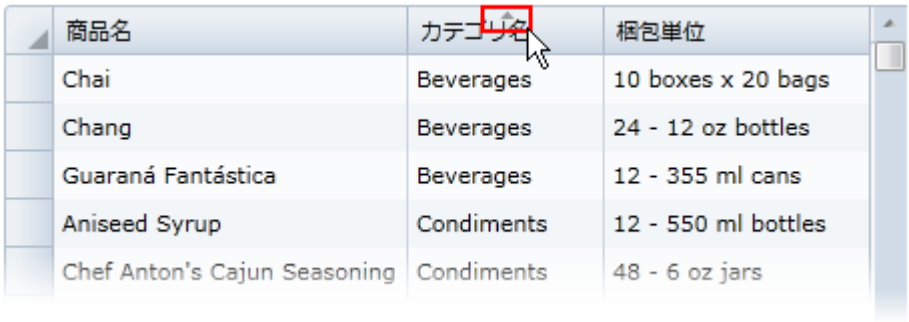
ここまでの成果

アプリケーションを実行し、実行時にグリッドで詳細行の切り替えを行うことができないことを確認します。行ヘッダーに、詳細行が切り替え可能であることを示す矢印アイコンが表示されなくなるので、行の切り替えを選択することはできなくなります。

列のソート

DataGrid for WPF/Silverlight では、実行時に簡単にグリッド列をソートできます。列をソートするには、ソートする列のヘッダーを1回クリックします。

列がソートされると、ソートグリフ(ソート方向インジケータ)が表示されます。



列ヘッダーをもう一度クリックすると、ソートを逆順にできます。ソートグリフの方向も変わります。

複数の列をソートするには、1つの列をソートし、次に[Ctrl]キーを押しながら2番目の列ヘッダーをクリックして、その列をソート条件に追加します。たとえば、次の図では、「カテゴリ名」列がまずソートされ、次に「商品名」列が逆順でソートされています。

ソートを有効にするには、**CanUserSort** プロパティを **True**(デフォルト)に設定する必要があります。

列のソートを無効にする

デフォルトでは、エンドユーザーは、実行時にグリッド内の列をソートできます。詳細については、「列のソート」を参照してください。しかし、必要に応じて、CanUserSort プロパティを **False** に設定することにより、列のソート機能を無効にすることができます。

設計時

列のソートを無効にするには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウに移動し、CanUserSort プロパティを見つけます。
3. CanUserSort プロパティの横にあるチェックボックスを「オフ」にします。

XAML の場合

たとえば、列のソートを無効にするには、CanUserSort="False" を <datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250" CanUserSort="False"
/>
```

コードの場合

たとえば、列のソートを無効にするには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.C1DataGrid1.CanUserSort = False
```

C#

```
this.c1DataGrid1.CanUserSort = false;
```

ここまでの成果

アプリケーションを実行し、実行時に列のソートを行うことができないことを確認します。実行時に列のヘッダーをクリックしても、グリッドはソートされず、列ヘッダーにはソートインジケータが表示されません。列のソートの詳細については、「列のソート」トピックを参照してください。

レイアウトおよび外観

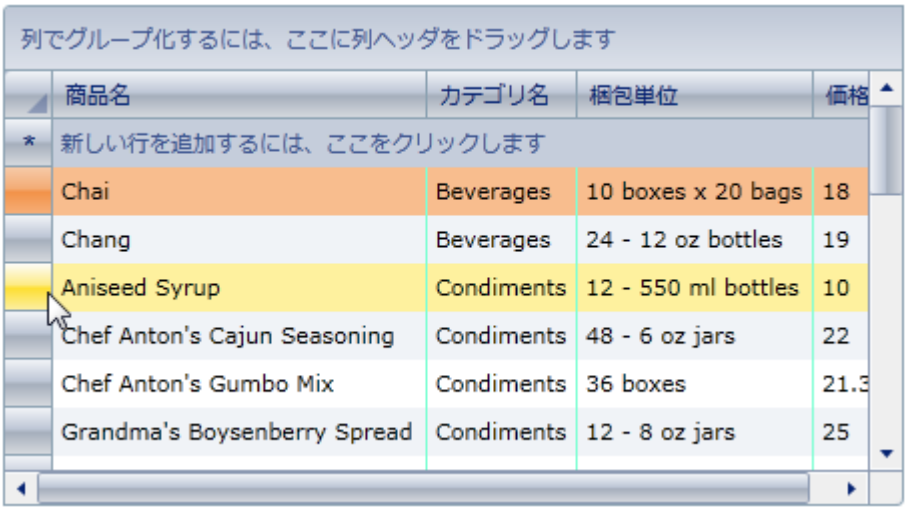
C1DataGrid コントロールは、一般的なテーブル書式設定オプション（1行ごとの背景色の変更、ヘッダーの表示／非表示、グリッド線、スクロールバーなど）をサポートしています。さらに、ブラシ、スタイル、およびテンプレートのプロパティで、コントロールとその行、列、ヘッダー、セルの外観を完全に変更できます。

DataGrid for WPF/Silverlight は、スタイル設定に ClearStyle 技術を使用することに注意してください。詳細については、「ClearStyle」を参照してください。

テーマ

DataGrid for WPF/Silverlight には、グリッドの外観をカスタマイズできるいくつかのテーマが組み込まれています。**C1DataGrid** コントロールを初めてページに追加すると、次の図のように表示されます。

これは、このコントロールのデフォルトの外観です。この外観は、組み込みテーマの1つを使用したり、独自のカスタムテーマを作成することで変更できます。すべての組み込みテーマは、Silverlight Toolkit テーマに基づいています。以下に、組み込みテーマについて説明および図示します。以下の図では、1つのセルが選択され、別のセルにポインタが置かれているため、選択スタイルとホバースタイルの両方が示されています。

テーマ名	テーマのプレビュー
C1ThemeBureauBlack	
C1ThemeExpressionDark	

C1ThemeExpressionLight

C1ThemeRainierOrange
(Silverlight *のみ*)

C1ThemeShinyBlue

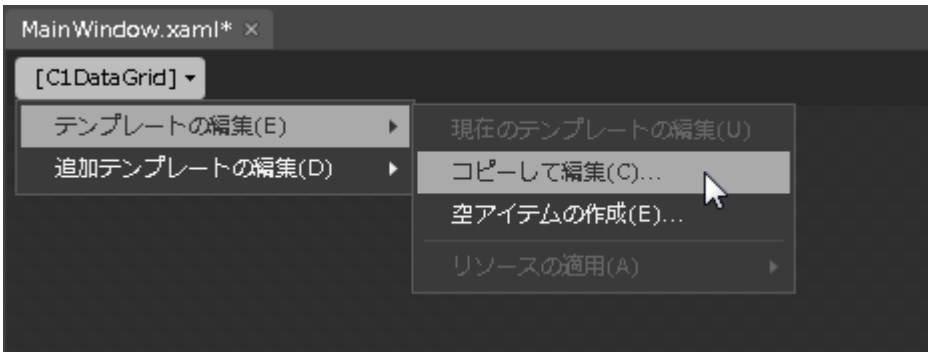
C1ThemeWhistlerBlue	

テンプレートとスタイルの編集

WPF コントロールを使用する主な利点の1つは、これが自由にカスタマイズできるユーザーインターフェイスを持つ「外観のない (lookless)」コントロールであることです。WPF/Silverlight アプリケーションのユーザーインターフェイスであるルックアンドフィールを独自に設計するのと同様に、**DataGrid for WPF/Silverlight** で管理されるデータに関して独自の UI を提供できます。Extensible Application Markup Language（XAML。「ザムル」と発音する）は、コードを記述することなく独自の UI を設計するための簡単な方法を提供する XML ベースの宣言型言語です。**DataGrid for WPF/Silverlight** には複数のテンプレートが含まれているため、独自の UI を何もない状態から作成する必要はありません。

テンプレートへのアクセス

テンプレートにアクセスするには、Microsoft Expression Blend で C1DataGrid コントロールを選択し、DataGrid のメニューで**[テンプレートの編集]**を選択します。編集可能なテンプレートのコピーを作成するには、[C1DataGrid]メニューを開き、**[テンプレートの編集]**を選択し、編集するテンプレートを選択します。次に、**[コピーして編集]**を選択して現在のテンプレートの編集可能なコピーを作成するか、**[空アイテムの作成]**を選択して新しい空のテンプレートを作成します。



メモ: メニューから新しいテンプレートを作成する場合、テンプレートはそのテンプレートのプロパティに自動的にリンクされます。手作業でテンプレートの XAML を作成する場合は、作成したテンプレートに適切なTemplate プロパティをリンクする必要があります。

DataGrid for WPF/Silverlight の **C1DataGrid** コントロールが提供するスタイルの複数のプロパティを使用して、コントロールとその行、列、ヘッダー、セルの外観を完全に変更できます。これらのスタイルの一部について、次の表で説明します。

スタイル	説明
CellStyle	セルのレンダリング時に使用されるスタイルを取得または設定します。
ColumnHeaderStyle	列ヘッダーのレンダリング時に使用されるスタイルを取得または設定します。

DragOverColumnStyle	ドラッグされて移動中の列の表示に使用される **ContentControl** 要素に適用されるスタイル。
DragSourceColumnStyle	ドラッグアンドドロップ操作の開始時にソース列の上に置かれた **ContentControl** 要素に適用されるスタイル。
DropIndicatorStyle	ドラッグされた列がドロップされる位置の指示に使用される **ContentControl** 要素に適用されるスタイル。
FilterStyle	フィルタコントロールコンテナに使用されるスタイルを取得または設定します。
FocusStyle	**C1DataGrid** にフォーカスを表示するために使用される内部 **Rectangle** のスタイルを設定します。
GroupColumnHeaderStyle	グループパネルの列ヘッダーのレンダリング時に使用されるスタイルを取得または設定します。
GroupRowHeaderStyle	グループ行のヘッダーのスタイルを取得または設定します。
GroupRowStyle	グループ行のスタイルを取得または設定します。
NewRowHeaderStyle	新しい項目を入力するための行ヘッダーのレンダリング時に使用されるスタイルを取得または設定します。
NewRowStyle	新しい項目を入力するための行のレンダリング時に使用されるスタイルを取得または設定します。
RowHeaderStyle	行ヘッダーのレンダリング時に使用されるスタイルを取得または設定します。
RowStyle	行のレンダリング時に使用されるスタイルを取得または設定します。

テーブル書式設定オプション

以下のトピックでは、テーブル書式設定オプション（グリッドヘッダー、テーブルオブジェクトの配置など）について詳しく説明します。

行および列ヘッダーの表示／非表示の設定

デフォルトでは、行ヘッダーと列ヘッダーがグリッドに表示されます。ただし、必要に応じて、一方または両方のヘッダーを非表示に設定できます。それには、**HeadersVisibility** プロパティを設定します。**HeadersVisibility** プロパティには、次のオプションの1つを設定できます。

オプション	説明
None	行ヘッダーと列ヘッダーの両方がグリッドで非表示になります。
Column	列ヘッダーのみがグリッドに表示されます。
Row	行ヘッダーのみがグリッドに表示されます。
All（デフォルト）	列ヘッダーと行ヘッダーの両方がグリッドに表示されます。

グリッド線の表示／非表示の設定

デフォルトでは、垂直方向と水平方向のグリッド線がグリッドに表示されます。ただし、必要に応じて、一方または両方のグリッド線を非表示に設定できます。それには、**GridLinesVisibility** プロパティを設定します。**GridLinesVisibility** プロパティには、次のオプションの1つを設定できます。

オプション	説明
None	水平方向と垂直方向の両方のグリッド線がグリッドで非表示になります。
Horizontal	水平方向のグリッド線のみがグリッドに表示されます。
Vertical	垂直方向のグリッド線のみがグリッドに表示されます。
All（デフォルト）	水平方向と垂直方向の両方のグリッド線がグリッドに表示されます。

新規行の表示／非表示の設定

デフォルトでは、グリッドの末尾に新規追加行が配置されます。ただし、必要に応じてその位置を変更できます。それには、**NewRowVisibility** プロパティを設定します。**NewRowVisibility** プロパティには、次のオプションの1つを設定できます。

オプション	説明
Top	新規追加行がグリッドの先頭に表示されます。
Bottom（デフォルト）	新規追加行がグリッドの末尾に表示されます。

垂直および水平スクロールバーの表示／非表示の設定

デフォルトでは、グリッドの内容の高さまたは幅がグリッドのサイズを超えたときにのみ、グリッドの水平スクロールバーと垂直スクロールバーが表示されます。ただし、必要に応じて、スクロールバーを常に表示または非表示になるように設定したり、まったく無効になるように設定できます。それには、**VerticalScrollbarVisibility** プロパティと **HorizontalScrollbarVisibility** プロパティを設定します。**VerticalScrollbarVisibility** プロパティと **HorizontalScrollbarVisibility** プロパティには、次のオプションの1つを設定できます。

オプション	説明
Disabled	選択したスクロールバーは無効になります。
Auto（デフォルト）	選択したスクロールバーは、グリッドの内容がグリッドウィンドウの大きさを超えるときだけ表示されます。
Hidden	選択したスクロールバーは非表示になります。
Visible	選択したスクロールバーは常に表示されます。

行の詳細の表示／非表示の設定

デフォルトでは、行の詳細は折りたたまれて表示されません。**RowDetailsVisibilityMode** プロパティを使用して、行の詳細を表示するかどうかと、いつ表示するかを設定できます。**RowDetailsVisibilityMode** プロパティには、次のオプションの1つを設定できます。

オプション	説明
VisibleWhenSelected	行の詳細は選択されたときにのみ表示されます。
Visible	行の詳細は常に表示されます。
Collapsed（デフォルト）	行の詳細は折りたたまれて表示されません。

ブラシ

DataGrid for WPF/Silverlight の **C1DataGrid** コントロールが提供するブラシの複数のプロパティを使用して、コントロールとその行、列、ヘッダー、セルの外観を完全に変更できます。次の表では、いくつかのブラシについて説明します。

ブラシ	説明
Background	レンダリング時に使用される背景ブラシを取得または設定します。(このブラシは、データグリッドのすべてのパーツに適用されます。)
Foreground	レンダリング時に使用される前景ブラシを取得または設定します。(このブラシは、データグリッドのすべてのパーツに適用されます。)
BorderBrush	レンダリング時に使用される境界線ブラシを取得または設定します。(このブラシは、テーマに基づいて、データグリッドの一部のパーツに適用されます。)
SelectedBrush	選択された行、行ヘッダー、列ヘッダーなどのレンダリング時に使用される、選択されたブラシを取得または設定します。
MouseOverBrush	マウスが行、行ヘッダー、列ヘッダーなどの上にあるときに使用される、マウスオーバーブラシを取得または設定します。
RowBackground	行の背景ブラシを取得または設定します。
RowForeground	行の前景ブラシを取得または設定します。
AlternatingRowBackground	交互表示行の背景ブラシを取得または設定します。
AlternatingRowForeground	交互表示行の前景ブラシを取得または設定します。
HorizontalGridLinesBrush	水平線に適用されるブラシを取得または設定します。
VerticalGridLinesBrush	垂直線に適用されるブラシを取得または設定します。

DataGrid for WPF/Silverlight は、スタイル設定に ClearStyle 技術を使用します。詳細については、「ClearStyle」を参照してください。

ClearStyle

DataGrid for WPF/Silverlight は、コントロールのテンプレートを変更することなくコントロールの色を簡単に変更できる ComponentOne 社の新しい ClearStyle 技術をサポートします。色のプロパティをいくつか設定するだけで、グリッド全体のスタイルを簡単に設定できます。

C1DataGrid コントロールの配色を設定する **C1DataGrid.Background** プロパティなど、いくつかのプロパティを設定するだけで、**C1DataGrid** コントロールの外観を全面的に変更できます。たとえば、**Background** プロパティを「#FFFFFFCC」に設定する場合の XAML マークアップは次のようになります。

```
<datagrid:C1DataGrid HorizontalAlignment="Left" Margin="10,10,0,0" Name="c1DataGrid1"
VerticalAlignment="Top" CanUserFreezeColumns="Left" CanUserGroup="True"
Background="#FFFFFFCC"/>
```

グリッドは次の図のように表示されます。

Background プロパティを「#FF663366」に設定し、**Foreground** プロパティを「White」に設定する場合の XAML マークアップは次のようになります。

XAML

```
<datagrid:C1DataGrid HorizontalAlignment="Left" Margin="10,10,0,0" Name="c1DataGrid1"
VerticalAlignment="Top" CanUserFreezeColumns="Left" CanUserGroup="True"
Background="#FF663366" Foreground="White"/>
```

グリッドは次の図のように表示されます。

次の XAML に示すように、**Background** プロパティをグラデーションの値に設定することもできます。

XAML

```
<datagrid:C1DataGrid x:Name="c1DataGrid1" HorizontalAlignment="Left"
Margin="10,10,0,0" VerticalAlignment="Top" CanUserFreezeColumns="Left"
CanUserGroup="True">
    <datagrid:C1DataGrid.Background>
        <LinearGradientBrush StartPoint="0,0" EndPoint="1,1">
            <GradientStop Color="GreenYellow" Offset="0.0" />
            <GradientStop Color="YellowGreen" Offset="0.85" />
        </LinearGradientBrush>
    </datagrid:C1DataGrid.Background>
</datagrid:C1DataGrid>
```

グリッドは次の図のように表示されます。

テンプレートパーツ

Microsoft Expression Blend では、新しいテンプレートを作成して、テンプレートパーツを表示および編集できます。たとえば、**C1DataGrid** コントロールをクリックして選択し、**[オブジェクト]**→**[テンプレートの編集]**→**[空アイテムの作成]**を選択します。新しいテンプレートを作成すると、テンプレートのパーツが**[パーツ]**ウィンドウに表示されます。

[パーツ]ウィンドウにパーツを表示するには、**ControlTemplate** を選択する必要があります。

[パーツ]ウィンドウで、任意の要素をダブルクリックすると、そのパーツをテンプレートに作成できます。これで、そのパーツがテンプレートに表示されます。また、**[パーツ]**ペイン内の要素アイコンが選択状態に変更されます。

C1DataGrid コントロールでは、次のテンプレートパーツを使用きます。

名前	タイプ	説明
Body	**DataGridMainPanel**	グリッド本体を含むパネル。
ColumnsHeader	DataGridColumnsHeaderPanel	DataGridColumnsHeaderPanel のコレクションを含むパネル。
Grouping	DataGridGroupingPresenter	グループ化パネル、またはグループ化パネルに列がない場合は別の要素を表示するプレゼンタ。
HorizontalScrollBar	ScrollBar	Thumb がスライドし、その位置が値に対応するスクロールバーを備えたコントロールを表します。

Root	Grid	列と行で構成される柔軟なグリッド領域を定義します。
RowsHeader	DataGridRowsHeaderPanel	DataGridRowsHeaderPanel を含むパネル。
VerticalScrollBar	ScrollBar	Thumb がスライドし、その位置が値に対応するスクロールバーを備えたコントロールを表します。

外観のカスタマイズ

以下のタスク別ヘルプのトピックでは、グリッドの外観を変更して、**DataGrid for WPF/Silverlight** をカスタマイズする方法について詳しく説明します。**DataGrid for WPF/Silverlight** には、ComponentOne 社のユニークな ClearStyle 技術を組み込むための複数の外観オプションが含まれています。たとえば、グリッドの背景色や交互表示の背景色を変更できます。ClearStyle 技術の詳細については、「ClearStyle」トピックを参照してください。また、以下のトピックでは、ヘッダーの位置の設定方法や新規行バーの追加方法など、グリッドのレイアウトの変更について詳しく説明します。

背景色と前景色の変更

DataGrid for WPF/Silverlight には、グリッドの外観全体を簡単に漏れなく変更するための ComponentOne 社のユニークな ClearStyle 技術が含まれています。以下の手順では、**C1DataGrid.Background** プロパティを設定して、グリッドの外観を全面的に変更する方法について詳しく説明します。ComponentOne 社の ClearStyle 技術の詳細については、「ClearStyle」トピックを参照してください。

設計時

グリッドの前景色と背景色が緑色になるように変更するには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウに移動し、**Background** プロパティの横にあるドロップダウン矢印をクリックします。
3. ボックス内のドロップダウン矢印をクリックすると、16 進数のコードが表示されるので、**緑色**を選択します。
4. **[プロパティ]**ウィンドウに移動し、**Foreground** プロパティの横にあるドロップダウン矢印をクリックします。
5. ボックス内のドロップダウン矢印をクリックすると、16 進数のコードが表示されるので、**白色**を選択します。

XAML の場合

たとえば、グリッドの前景色と背景色が緑色になるように変更するには、Background="Green" Foreground="White" を <datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250" Background="Green"
Foreground="White" />
```

コードの場合

たとえば、グリッドの前景色と背景色が緑色になるように変更するには、プロジェクトに次のコードを追加します。

Visual Basic

```
Me.C1DataGrid1.Background = New System.Windows.Media.SolidColorBrush(Colors.Green)
Me.C1DataGrid1.ForeGround = New System.Windows.Media.SolidColorBrush(Colors.White)
```

C#

```
this.c1DataGrid1.Background = new System.Windows.Media.
SolidColorBrush(Colors.Green);
this.c1DataGrid1.Foreground = new System.Windows.Media.
SolidColorBrush(Colors.White);
```

ここまでの成果

アプリケーションを実行して、グリッドが緑色になり、グリッドヘッダーに白色のテキストが表示されることを確認します。

C1DataGrid コントロールの ClearStyle 技術により、グリッドの色、グリッドのスクロールバー、およびグリッドの1行ごとの背景色がすべて、緑色の背景色を反映するように変更されることに注意してください。グリッド内の1つの項目を強調表示して、マウスのホバースタイルが変更されていないことを確認します。必要に応じて、これらのスタイルもカスタマイズできます。詳細については、「マウスホバースタイルの変更」を参照してください。

交互表示の色の除去

デフォルトでは、**DataGrid for WPF/Silverlight** は、1行おきに別の色で表示されます。1行おきの色の変更とは、1行おきにグリッドの基本色とは異なる色で表示されることです。これは、グリッド内で行が区別しやすくなるという点で便利ですが、1行おきの色の変更を除去してグリッドの外観を均一にすることもできます。

設計時

1行おきの色の設定を除去して、すべての行が白色で表示されるように設定するには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウに移動し、**RowBackground** プロパティの横にあるドロップダウン矢印をクリックします。
3. ボックス内のドロップダウン矢印をクリックすると、16 進数のコードが表示されるので、**白色**を選択します。
4. **[プロパティ]**ウィンドウに移動し、**AlternatingRowBackground** プロパティの横にあるドロップダウン矢印をクリックします。
5. ボックス内のドロップダウン矢印をクリックすると、16 進数のコードが表示されるので、**白色**を選択します。

XAML の場合

1行おきの色の変更を除去し、すべての行が白色で表示されるように設定するには、RowBackground="White" AlternatingRowBackground="White" を <datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250"
```

```
RowBackground="White" AlternatingRowBackground="White" />
```

コードの場合

1行おきの色の設定を除去して、すべての行が白色で表示されるように設定するには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.C1DataGrid1.RowBackground = New System.Windows.Media.SolidColorBrush(Colors.White)
Me.C1DataGrid1.AlternatingRowBackground = New
System.Windows.Media.SolidColorBrush(Colors.White)
```

C#

```
this.c1DataGrid1.RowBackground = new System.Windows.Media.
SolidColorBrush(Colors.White);
this.c1DataGrid1.AlternatingRowBackground = new System.Windows.Media.
SolidColorBrush(Colors.White);
```

ここまでの成果

アプリケーションを実行して、グリッド内のすべての行が白色で表示されることを確認します。

マウスホバースタイルの変更

デフォルトでは、マウスを置いた列と行は、操作対象のグリッドの領域をユーザーに示すために別の色で表示されます。必要に応じて、マウスを置いたセルの外観をカスタマイズできます。たとえば、セルの強調表示の度合いを高めたり、強調表示を除去することができます。

設計時

マウスオーバー効果を黄色に設定するには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択します。
2. **[プロパティ]**ウィンドウに移動し、**MouseOverBrush** プロパティの横にあるドロップダウン矢印をクリックします。
3. ボックス内のドロップダウン矢印をクリックすると、16 進数のコードが表示されるので、**黄色**を選択します。

XAML の場合

マウスオーバー効果を黄色に設定するには、MouseOverBrush="Yellow" を <datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250"
MouseOverBrush="Yellow" />
```

コードの場合

マウスオーバー効果を黄色に設定するには、プロジェクトに次のコードを追加します。

Visual Basic

```
Me.c1datagrid1.MouseOverBrush = New
System.Windows.Media.SolidColorBrush(Colors.Yellow)
```

C#

```
this.c1datagrid1.MouseOverBrush = new
System.Windows.Media.SolidColorBrush(Colors.Yellow);
```

ここまでの成果

アプリケーションを実行して、グリッド内の強調表示されたすべての行と列が黄色で表示されることを確認します。

フォントスタイルの変更

コントロールの実行時に **DataGrid for WPF/Silverlight** に表示されるフォントスタイルを更新することができます。たとえば、アプリケーションの外観に合わせて、グリッドのスタイル（フォントスタイルはその要素の1つとして含まれる）を変更できます。

設計時

フォントスタイルを変更するには、次の手順に従います。

1. C1DataGrid コントロールをクリックして選択します。

2. **[プロパティ]**ウィンドウに移動し、**FontFamily** プロパティの横にあるドロップダウン矢印をクリックします。
3. ボックス内のドロップダウン矢印をクリックして、ドロップダウンリストから「Times New Roman」を選択します。
4. **[プロパティ]**ウィンドウに移動し、**FontSize** プロパティの横にあるドロップダウン矢印をクリックします。
5. ボックス内のドロップダウン矢印をクリックして、ドロップダウンリストから「10」を選択します。

XAML の場合

フォントスタイルを変更するには、FontFamily="Times New Roman" FontSize="10" を <datagrid:C1DataGrid> タグに追加します。次のようになります。

XAML

```
<datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250" FontFamily="Times New Roman" FontSize="10" />
```

コードの場合

フォントスタイルを変更するには、次のコードをプロジェクトに追加します。

Visual Basic

```
Me.c1datagrid1.FontFamily = New FontFamily("Times New Roman")
Me.c1datagrid1.FontSize = 10
```

C#

```
this.c1datagrid1.FontFamily = new FontFamily("Times New Roman");
this.c1datagrid1.FontSize = 10;
```

ここまでの成果

アプリケーションを実行し、グリッド内のすべての行が Times New Roman フォントで表示されることを確認します。

実行時の操作

次の図には、DataGrid for WPF/Silverlight の C1DataGrid コントロールで可能な実行時の操作の一部が強調表示されてい

ます。

以下のトピックでは、データのフィルタ処理、ソート、グループ化などの実行時機能について詳しく説明します。

選択モードの設定

SelectionMode プロパティを設定して、グリッドの選択モードの動作を設定できます。ユーザーがグリッドを操作する方法を変更できますが、**SelectionMode** プロパティを次のいずれかの値に設定します。

オプション	説明
None	ユーザーはどの項目も選択できません。
SingleCell	ユーザーは、一度に1つのセルのみを選択できます。
SingleRow	ユーザーは、一度に1つの行のみを選択できます。
SingleColumn	ユーザーは、一度に1つの列のみを選択できます。
SingleRange	ユーザーは、一度に1つのセル範囲のみを選択できます。(範囲は、2つのセルで区切られた四角形です。)
MultiRow(デフォルト)	ユーザーは、対応する修飾キーを押しながら複数の行を選択できます。
MultiColumn	ユーザーは、対応する修飾キーを押しながら複数の列を選択できます。
MultiRange	ユーザーは、対応する修飾キーを押しながら複数のセル範囲を選択できます。

修飾キーと **MultiRow** オプションの詳細については、「複数行の選択」のトピックを参照してください。

自動生成された列のカスタマイズ

列が自動生成される場合も列のカスタマイズは可能です。**AutoGenerateColumns** プロパティを True に設定し、列が自動生

成されたら、生成された列の表示方法をコードでカスタマイズできます。それには、**C1DataGrid.AutoGeneratingColumn** イベントを制御します。

AutoGeneratingColumn イベントハンドラの追加

次の手順に従って、**AutoGeneratingColumn** イベントハンドラを追加します。

1. コードビューに切り替え、AutoGeneratingColumn イベントのイベントハンドラを追加します。たとえば、次のようになります。

WPF

Visual Basic

```
Private Sub C1DataGrid1_AutoGeneratingColumn(ByVal sender As System.Object,
ByVal e As C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs) Handles
C1DataGrid1.AutoGeneratingColumn
    ' コードをここに追加します。
End Sub
```

C#

```
private void C1DataGrid1_AutoGeneratingColumn(object sender,
C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs e)
{
    // コードをここに追加します。
}
```

Silverlight

Visual Basic

```
Private Sub c1DataGrid1_AutoGeneratingColumn(ByVal sender As System.Object,
ByVal e As C1.Silverlight.DataGrid.DataGridAutoGeneratingColumnEventArgs)
Handles c1dg.AutoGeneratingColumn
    ' コードをここに追加します。
End Sub
```

C#

```
private void c1DataGrid1_AutoGeneratingColumn(object sender,
C1.Silverlight.DataGrid.DataGridAutoGeneratingColumnEventArgs e)
{
    // コードをここに追加します。
}
```

2. ソースビューに切り替え、イベントハンドラを **C1DataGrid** コントロールのインスタンスに追加します。たとえば、次のようになります。

XAML

```
<datagrid:C1DataGrid x:Name="c1DataGrid1" AutoGenerateColumns="True"
AutoGeneratingColumn=" c1DataGrid1_AutoGeneratingColumn"></datagrid:C1DataGrid>
```

これで、自動生成された列の外観と動作をカスタマイズするためのコードを **AutoGeneratingColumn** イベントハンドラに追加できます。以下は、列の書式設定と動作をカスタマイズする例です。

列生成のキャンセル

特定の列の生成を **AutoGeneratingColumn** イベントでキャンセルできます。たとえば、次のコードを使用して、グリッド内のブール値列の生成をキャンセルできます。

WPF

Visual Basic

```
Private Sub C1DataGrid1_AutoGeneratingColumn(ByVal sender As System.Object, ByVal e
As C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs) Handles
C1DataGrid1.AutoGeneratingColumn
    ' すべてのブール値列の自動生成をキャンセルします。
    If e.Property.PropertyType Is GetType(Boolean) Then
        e.Cancel = True
    End If
End Sub
```

C#

```
private void c1DataGrid1_AutoGeneratingColumn(object sender,
C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs e)
{
    // すべてのブール値列の自動生成をキャンセルします。
    if (e.Property.PropertyType == typeof(bool))
        e.Cancel = true;
}
```

Silverlight

Visual Basic

```
Private Sub c1DataGrid1_AutoGeneratingColumn(ByVal sender As System.Object, ByVal e
As C1.Silverlight.DataGrid.DataGridAutoGeneratingColumnEventArgs) Handles
c1dg.AutoGeneratingColumn
    ' すべてのブール値列の自動生成をキャンセルします。
    If e.Property.PropertyType Is GetType(Boolean) Then
        e.Cancel = True
    End If
End Sub
```

C#

```
private void c1DataGrid1_AutoGeneratingColumn(object sender,
C1.Silverlight.DataGrid.DataGridAutoGeneratingColumnEventArgs e)
{
    // すべてのブール値列の自動生成をキャンセルします。
    if (e.Property.PropertyType == typeof(bool))
        e.Cancel = true;
}
```

列ヘッダーの変更

AutoGeneratingColumn イベントで、自動生成された列のヘッダーに表示されるテキストを変更できます。たとえば、次のコードを使用すると、列に「名前」というヘッダーが表示されるように「商品名」列を変更できます。

Visual Basic

```
Private Sub C1DataGrid1_AutoGeneratingColumn(ByVal sender As System.Object, ByVal e
As C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs) Handles
C1DataGrid1.AutoGeneratingColumn
    '「商品名」列のヘッダーを変更します。
    If e.Column.Header.ToString() = "ProductName" Then
        e.Header = "名前"
    End If
End Sub
```

C#

```
private void c1DataGrid1_AutoGeneratingColumn(object sender,
C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs e)
{
    // 「商品名」列のヘッダーを変更します。
    if (e.Column.Header.ToString() == "ProductName")
        e.Column.Header = "名前";
}
```

列操作の禁止

AutoGeneratingColumn イベントを使用して、生成された特定の列に対してエンドユーザーが実行できる操作を変更できます。たとえば、次のコードを使用すると、ユーザーが読み取り専用の列を移動することを禁止できます。

Visual Basic

```
Private Sub C1DataGrid1_AutoGeneratingColumn(ByVal sender As System.Object, ByVal e
As C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs) Handles
C1DataGrid1.AutoGeneratingColumn
    ' 「商品名」列のヘッダーを変更します。
    If e.Column.IsReadOnly = True Then
       e.Column.CanUserMove = False
    End If
End Sub
```

C#

```
private void c1DataGrid1_AutoGeneratingColumn(object sender,
```

```
C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs e)
{
    //「商品名」列のヘッダーを変更します。
    if (e.Column.IsReadOnly == true)
        e.Column.CanUserMove = false;
}
```

チュートリアル (Silverlight のみ)

以下のチュートリアルでは、**DataGrid for Silverlight** について詳しく説明し、グリッドを詳細にカスタマイズするための手順を示します。チュートリアルには、追加的なデータ連結、スタイル設定、および動作のカスタマイズ手順が含まれるほか、高度な機能も紹介します。

メモ:このセクションの内容は、ComponentOne for Silverlight にのみ適用されます。

Web サービスへのグリッドの連結

次のチュートリアルでは、**C1DataGrid** コントロールを標準の Northwind データベースに連結し、Web サービスを作成するプロセスを示します。

手順 1:ユーザーインターフェイスの作成

この手順では、最初に Visual Studio で Silverlight グリッドアプリケーションを作成します。引き続き、アプリケーションのユーザーインターフェイス(UI)を作成してカスタマイズし、プロジェクトに **C1DataGrid** コントロールを追加します。

プロジェクトを設定するには、次の手順に従います。

1. Visual Studio で、[ファイル]→**[新しいプロジェクト]**を選択します。
2. **[新しいプロジェクト]**ダイアログボックスで、左ペインから言語を選択し、テンプレートリストから**[Silverlight アプリケーション]**を選択します。プロジェクトの名前(たとえば、「C1DataGrid」)を入力し、**[OK]**をクリックします。[新しい Silverlight アプリケーション]ダイアログボックスが表示されます。
3. 既定の設定のまま**[OK]**をクリックすると、**[新しい Silverlight アプリケーション]**ダイアログボックスが閉じ、プロジェクトが作成されます。
4. **MainPage.xaml** ファイルが開いていない場合は、ソリューションエクスプローラに移動し、**MainPage.xaml** 項目をダブルクリックします。
5. XAML ビューで、タグを探します。
6. <UserControl>タグで、Width="400" Height="300"(もしくは d:DesignWidth="400" d:DesignHeight="300")を Width="600" Height="400" に置き換えます。
   これで、Silverlight アプリケーションのサイズが大きくなります。
7. カーソルを <Grid x:Name="LayoutRoot" Background="White"> タグの直後に置き、次のマークアップを追加します。

   XAML
   ```
   <!-- グリッドレイアウト-->
   <Grid.RowDefinitions>
   <RowDefinition Height="Auto" />
   <RowDefinition Height="*" />
   <RowDefinition Height="Auto" />
   </Grid.RowDefinitions>
   ```
   この行定義は、グリッドのレイアウトを定義します。
8. タイトルをアプリケーションに追加します。それには、次の **TextBlock** を </Grid.RowDefinitions> タグの真下に追加します。

XAML

```
<!-- タイトル -->
<TextBlock Text="DataGrid for Silverlight" Margin="5"   FontSize="16"/>
```

9. プロジェクトの XAML ウィンドウで、カーソルを </Grid>タグの真上に置き1回クリックします。
10. ツールボックスに移動し、**[C1DataGrid]**アイコンをダブルクリックして、**MainPage.xaml** にグリッドコントロールを追加します。XAML マークアップは次のようになります。

XAML

```
<UserControl
 xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
 x:Class="C1DataGrid.MainPage"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Width="600"
Height="400">
    <Grid x:Name="LayoutRoot" Background="White">
            <!-- グリッドレイアウト-->
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="*" />
                <RowDefinition Height="Auto" />
            </Grid.RowDefinitions>
            <!-- タイトル -->
            <TextBlock Text="DataGrid for Silverlight" Margin="5"
FontSize="16"/>
        <c1:C1DataGrid></c1:C1DataGrid>
    </Grid>
</UserControl>
```

C1.Silverlight.DataGrid 名前空間と タグがプロジェクトに追加されていることがわかります。

11. <c1:C1DataGrid>タグが既存のコンテンツを含む場合、コンテンツを削除します。タグは次のように表示されます。

XAML

```
<c1:C1DataGrid>
```

12. x:Name="_c1DataGrid" を <c1:C1DataGrid> タグに追加して、グリッドに名前を付けます。次のようになります。

XAML

```
<c1:C1DataGrid x:Name="_c1DataGrid">
```

コントロールに一意の識別子を付けると、コードでその **C1DataGrid** コントロールにアクセスできるようになります。

13. グリッドの位置を定義します。それには、<c1:C1DataGrid> タグを Grid.Row="1" に追加します。次のようになります。

XAML

```
<c1:C1DataGrid x:Name="_c1DataGrid" Grid.Row="1">
```

14. </c1:C1DataGrid>タグの直後に次のマークアップを追加します。

XAML

```
<TextBlock x:Name="_tbStatus" Text="Ready"
   VerticalAlignment="Center" FontSize="12" Foreground="Gray" Margin="5"
Grid.Row="2" />
```

この TextBlock は、ステータス情報テキストを表示するために使用されます。

ここまでの成果

アプリケーションを実行し、ページにタイトル、グリッド、テキスト（グリッドの下）が含まれていることを確認します。これで、基本的なグリッドアプリケーションを作成できました。ただし、グリッドは空白でデータが入っていません。次の手順では、プロジェク

トにデータベースを追加し、グリッドをデータソースに連結します。

手順 2:Web サービスの追加

この手順では、データベースをプロジェクトに追加し、グリッドを連結するプロセスを開始します。この手順では、標準の Northwind データベースとサンプルコードファイルを使用します。これらは、**ComponentOne for Silverlight** のサンプルにインストールされています。

プロジェクトを設定するには、次の手順に従います。

1. ソリューションエクスプローラで、.Web プロジェクト（たとえば、ComponentOneDataGrid.Web）を展開します。**App_Data** フォルダが表示されていない場合、.Web プロジェクトを右クリックして、**[追加]**→**[新しいフォルダ]**を選択して、フォルダ名を**「App_Data」**に設定します。
2. ソリューションエクスプローラで、App_Data ノードを右クリックして、**[追加]**→**[既存の項目]**を選択します。
3. **[既存のアイテムの追加]**ダイアログボックスで、**C:\Users\<ユーザー名>\Documents\ComponentOne Samples\Common** ディレクトリに移動します。**Nwind.mdb** ファイルを選択し、[追加]をクリックしてプロジェクトに追加します。
4. ソリューションエクスプローラで、今追加した **Nwind.mdb** ファイルを選択し、プロパティウィンドウで、[ビルドアクション]プロパティを**[なし]**に設定します。
5. ソリューションエクスプローラで、.Web プロジェクト（たとえば、ComponentOneDataGrid.Web）を右クリックし、**[追加]**→**[既存の項目]**を選択します。
6. **[既存のアイテムの追加]**ダイアログボックスで、製品サンプルの **C1_MDSL\C1_MDSLWeb** ディレクトリに移動します。SmartDataSet.cs ファイルを選択し、**[追加]**をクリックしてプロジェクトに追加します。
   このファイルには、データベースと双方向のデータ転送を行うためのコードが含まれています。
7. ソリューションエクスプローラで、.Web プロジェクトを右クリックし、**[追加]**→**[新しい項目]**を選択します。
8. **[新しい項目の追加]**ダイアログボックスの左ペインで、**[Web]**項目を選択します。
9. テンプレートリストで、**[Web サービス]**を選択します。その Web サービスの名前を**「DataService.asmx」**とし、**[追加]**ボタンをクリックします。Web サービスファイルがプロジェクトに追加され、自動的に開かれます。
10. DataService.asmx ファイルで、ファイルの先頭に次の using 文を追加します。

Visual Basic

```
Imports System.IO

Imports System.Data

Imports C1_MDSLWeb ' SmartDataSet 名前空間
```

C#

```
using System.IO;

using System.Data;

using C1_MDSLWeb; // SmartDataSet 名前空間
```

11. 次に、**[System.Web.Script.Services.ScriptService]** または <System.Web.Script.Services.ScriptService()> 行のコメントを外します。
    これによりスクリプトからの Web サービス呼び出しが有効になります。
12. 既存の HelloWorld メソッドを削除し、次のコードに置き換えます。

Visual Basic

```
<WebMethod> _

Public Function GetData(tables As String) As Byte()

    ' 接続文字列を含む DataSet を作成します。

    Dim ds = GetDataSet()

    ' DataSet にデータを読み込みます

    ds.Fill(tables.Split(","C))

    ' ストリームに接続します

    Dim ms = New System.IO.MemoryStream()

    ds.WriteXml(ms, XmlWriteMode.WriteSchema)

    ' ストリームデータを返却します

    Return ms.ToArray()

End Function

Private Function GetDataSet() As SmartDataSet

    ' mdb ファイルの物理的な場所を取得します

    Dim mdb As String = Path.Combine(Context.Request.PhysicalApplicationPath,
"App_Data\nwind.mdb")

    ' このファイルの存在を確認します

    If Not File.Exists(mdb) Then

        Dim msg As String = String.Format("Cannot find database file {0}.", mdb)

        Throw New FileNotFoundException(msg)

    End If

    ' ファイルが読み取り専用でないことを確認します（ソースコントロールによって読み取り専用にされる場合があります）

    Dim att As FileAttributes = File.GetAttributes(mdb)

    If (att And FileAttributes.[ReadOnly]) <> 0 Then

        att = att And Not FileAttributes.[ReadOnly]

        File.SetAttributes(mdb, att)
```

```
    End If

    ' SmartDataSet を作成および初期化します

    Dim dataSet = New SmartDataSet()

    dataSet.ConnectionString = "provider=microsoft.jet.oledb.4.0;data source=" &
mdb

    Return dataSet

End Function
```

C#

```
[WebMethod]

public byte[] GetData(string tables)

{

    // 接続文字列を含む DataSet を作成します

    var ds = GetDataSet();

    // DataSet にデータを読み込みます

    ds.Fill(tables.Split(','));

    // ストリームに接続します

    var ms = new System.IO.MemoryStream();

    ds.WriteXml(ms, XmlWriteMode.WriteSchema);

    // ストリームデータを返却します

    return ms.ToArray();

}

SmartDataSet GetDataSet()

{

    // mdb ファイルの物理的な場所を取得します

    string mdb = Path.Combine(

    Context.Request.PhysicalApplicationPath, @"App_Data\nwind.mdb");
```

```
    // このファイルの存在を確認します
    if (!File.Exists(mdb))
    {
        string msg = string.Format("Cannot find database file {0}.", mdb);
        throw new FileNotFoundException(msg);
    }
    // ファイルが読み取り専用でないことを確認します（ソースコントロールによって読み取り専用にされる場合があります）
    FileAttributes att = File.GetAttributes(mdb);
    if ((att & FileAttributes.ReadOnly) != 0)
    {
        att &= ~FileAttributes.ReadOnly;
        File.SetAttributes(mdb, att);
    }
    // SmartDataSet を作成および初期化します
    var dataSet = new SmartDataSet();
    dataSet.ConnectionString = "provider=microsoft.jet.oledb.4.0;data source=" + mdb;
    return dataSet;
}
```

このコードは、データセットを作成し、データベースからデータを受け取ります。

13. .Web プロジェクト（たとえば、ComponentOneDataGrid.Web）を右クリックし、コンテキストメニューから**[ビルド]**を選択します。これで、ComponentOneDataGrid.Web プロジェクトの作業は終了し、ComponentOneDataGrid プロジェクトの作業に戻ります。

ここまでの成果

この手順では、プロジェクトにデータベースを追加し、Web サービスを作成しました。次の手順では、最後にプロジェクトに Web サービスを接続し、アプリケーションを実行します。

手順 3：Web サービスの接続

前の手順では、Web サービスを作成し、プロジェクトにデータベースを追加しました。この手順では、引き続き、Web サービス

をアプリケーションにリンクします。この手順では **Data for Silverlight** が必要です。

プロジェクトを設定するには、次の手順に従います。

1. ソリューションエクスプローラでプロジェクトノードを展開し、プロジェクト名（たとえば、ComponentOneDataGrid）を右クリックし、コンテキストメニューから**[参照の追加]**を選択します。
2. **[参照の追加]**ダイアログボックスで、**C1.Silverlight.Data** アセンブリへの参照を追加し、**[OK]**をクリックします。
3. ソリューションエクスプローラで、プロジェクト名を右クリックし、コンテキストメニューから**[サービス参照の追加]**を選択します。
4. **[サービス参照の追加]**ダイアログボックスで、**[検出]**ボタンをクリックします。サービスのリストに DataService.asmx ファイルが表示されます。
5. **[名前空間]**テキストボックスで、デフォルト値を「DataService」に変更します。**[OK]**ボタンをクリックして設定を保存し、ダイアログボックスを閉じます。
6. ソリューションエクスプローラで、**MainPage.xaml** ノードを展開し、**MainPage.xaml.cs** または **MainPage.xaml.vb** ファイルをダブルクリックしてコードエディタで開きます。
7. 次の using 文または Imports 文をファイルの先頭に追加します。

VisualBasic

```
Imports System.IO
Imports C1.Silverlight.Data
Imports ComponentOneDataGrid.DataService ' ComponentOneDataGrid はプロジェクトの名前空間です。プロジェクト名に合わせて変更してください。
```

C#

```
using System.IO;
using C1.Silverlight.Data;
using ComponentOneDataGrid.DataService; // ComponentOneDataGrid はプロジェクトの名前空間です。プロジェクト名に合わせて変更してください。
```

8. **MainPage** コンストラクタに LoadData(); を追加します。次のようになります。

VisualBasic

```
Public Sub New()
    InitializeComponent()
    LoadData()
End Sub
```

C#

```
public MainPage()
{
    InitializeComponent();
    LoadData();
}
```

9. Web サービスからデータを取得するために、**LoadData** メソッドと **svc_GetDataCompleted** メソッドを追加します。

VisaulBasic

```
Private _ds As DataSet = Nothing
Private Sub LoadData()
    ' Web サービスを呼び出します
    Dim svc = GetDataService()
    AddHandler svc.GetDataCompleted, AddressOf svc_GetDataCompleted
    'svc.GetDataAsync("Categories,Products,Employees");
    svc.GetDataAsync("Employees")
End Sub
Private Sub svc_GetDataCompleted(sender As Object, e As
GetDataCompletedEventArgs)
    ' エラーを処理します
    If e.[Error] IsNot Nothing Then
        _tbStatus.Text = "データダウンロード中にエラーが発生しました..."
        Return
    End If
    ' サーバーからのデータストリームを解析します(DataSet は XML)
    _tbStatus.Text = String.Format("データは, {0:n0} kBytes", e.Result.Length /
1024)
    Dim ms = New MemoryStream(e.Result)
    _ds = New DataSet()
    _ds.ReadXml(ms)
    ' コントロールをデータに連結します
    BindData()
End Sub
```

C#

```
DataSet _ds = null;
void LoadData()
{
    // Web サービスを呼び出します
    var svc = GetDataService();
    svc.GetDataCompleted += svc_GetDataCompleted;
    //svc.GetDataAsync("Categories,Products,Employees");
    svc.GetDataAsync("Employees");
}
void svc_GetDataCompleted(object sender, GetDataCompletedEventArgs e)
{
    // エラーを処理します
    if (e.Error != null)
    {
        _tbStatus.Text = "データダウンロード中にエラーが発生しました...";
        return;
    }
    // サーバーからのデータストリームを解析します(DataSet は XML)
    _tbStatus.Text = string.Format("データは, {0:n0} kBytes", e.Result.Length /
1024);
    var ms = new MemoryStream(e.Result);
```

```
    _ds = new DataSet();
    _ds.ReadXml(ms);
    // コントロールをデータに連結します
    BindData();
}
```

10. **GetDataService()** メソッドを実装します。それには、次のコードを追加します。

VisualBasic

```
' 現在のホスト/ドメインに関連するデータサービスを取得します
Private Function GetDataService() As DataServiceSoapClient
    ' バッファサイズを増やします
    Dim binding = New System.ServiceModel.BasicHttpBinding()
    binding.MaxReceivedMessageSize = 2147483647
    ' int.MaxValue
    binding.MaxBufferSize = 2147483647
    ' サービスの絶対アドレスを取得します
    Dim uri As Uri =
C1.Silverlight.Extensions.GetAbsoluteUri("DataService.asmx")
    Dim address = New System.ServiceModel.EndpointAddress(uri)
    ' 新しいサービスクライアントを返します
    Return New DataServiceSoapClient(binding, address)
End Function
```

C#

```
// 現在のホスト/ドメインに関連するデータサービスを取得します
DataServiceSoapClient GetDataService()
{
    // バッファサイズを増やします
    var binding = new System.ServiceModel.BasicHttpBinding();
    binding.MaxReceivedMessageSize = 2147483647;
    // int.MaxValue
    binding.MaxBufferSize = 2147483647;
    // サービスの絶対アドレスを取得します
    Uri uri = C1.Silverlight.Extensions.GetAbsoluteUri("DataService.asmx");
    var address = new System.ServiceModel.EndpointAddress(uri);
    // 新しいサービスクライアントを返します
    return new DataServiceSoapClient(binding, address);
}
```

11. **BindData()** メソッドを実装します。それには、次のコードを追加します。

VisualBasic

```
Private Sub BindData()
    ' テーブルを取得します
    Dim dtEmployees As DataTable = _ds.Tables("Employees")
    ' カテゴリグリッドにデータを挿入します
```

```
    _c1DataGrid.ItemsSource = dtEmployees.DefaultView
End Sub
```

C#

```
void BindData()
{
    // テーブルを取得します
    DataTable dtEmployees = _ds.Tables["Employees"];
    // カテゴリグリッドにデータを挿入します
    _c1DataGrid.ItemsSource = dtEmployees.DefaultView;
}
```

12. アプリケーションを実行し、グリッドが Northwind データベースの Employees テーブルに連結していることを確認します。

ここまでの成果

おめでとうございます。このチュートリアルは終了です。このチュートリアルでは、新しい Silverlight プロジェクトを作成し、Access データベースを追加し、Web サービスを作成して、**C1DataGrid** コントロールを連結しました。

RSS フィードへのグリッドの連結

次のチュートリアルでは、C1DataGrid コントロールを RSS フィードに連結するプロセスを説明します。この例では、グリッドを GrapeCity RSS ニュースフィード（http://feeds.feedburner.com/http/wwwgrapecitycom/japan）に連結します。URL を変更すれば、他の RSS フィードを購読することも可能です。

次の手順に従います。

1. Visual Studio で、**[ファイル]**→**[新しいプロジェクト]**を選択します。
2. **[新しいプロジェクト]**ダイアログボックスで、左ペインから言語を選択し(この例では **C#** を使用)、テンプレートリストから**[Silverlight アプリケーション]**を選択します。プロジェクトの**名前**(たとえば、**「C1DataGridRSS」**)を入力し、**[OK]**をクリックします。**[新しい Silverlight アプリケーション]**ダイアログボックスが表示されます。
3. 既定の設定のまま**[OK]**をクリックすると、**[新しい Silverlight アプリケーション]**ダイアログボックスが閉じ、プロジェクトが作成されます。
4. ソリューションエクスプローラウィンドウで、プロジェクト名(たとえば、C1DataGridRSS)を右クリックし、**[参照の追加]**を選択します。
5. **[参照の追加]**ダイアログボックスで **System.Xml.Linq** ライブラリを見つけ、**[OK]**をクリックすると、参照がプロジェクトに追加されます。
6. プロジェクトの XAML ウィンドウで、カーソルを タグと タグの間に置き、1回クリックします。
7. ツールボックスに移動し、**[C1DataGrid]**アイコンをダブルクリックして、**MainPage.xaml** にグリッドコントロールを追加します。XAML マークアップは次のようになります。

XAML

```
<UserControl
 xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
 x:Class="C1DataGrid.MainPage"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Width="400"
Height="300">
    <Grid x:Name="LayoutRoot" Background="White">
        <c1:C1DataGrid></c1:C1DataGrid>
    </Grid>
</UserControl>
```

C1.Silverlight.DataGrid 名前空間と<c1:C1DataGrid> </c1:C1DataGrid>タグがプロジェクトに追加されていることがわかります。

8. <c1:C1DataGrid>タグが既存のコンテンツを含む場合、コンテンツを削除します。タグは次のように表示されます。

XAML

```
<c1:C1DataGrid></c1:C1DataGrid>
```

9. x:Name="c1DataGrid1" を<c1:C1DataGrid>タグに追加して、グリッドに名前を付けます。次のようになります。

XAML

```
<c1:C1DataGrid x:Name="c1DataGrid1">
```

コントロールに一意の識別子を付けると、コードでその **C1DataGrid** コントロールにアクセスできるようになります。

10. <c1:C1DataGrid>タグに AutoGenerateColumns="True" を追加します。次のようになります。

XAML

```
<c1:C1DataGrid x:Name="c1DataGrid1" AutoGenerateColumns="True">
```

このように、グリッドはデータをデータソースから自動的に生成して表示します。

11. ソリューションエクスプローラで、**MainPage.xaml** ノードを展開し、**MainPage.xaml.cs** または **MainPage.xaml.vb** ファイルをダブルクリックしてコードエディタで開きます。
12. 次の using 文または Imports 文をファイルの先頭に追加します。

VisualBasic

```
Imports System.Xml.Linq
```

C#

```
using System.Xml.Linq;
```

13. **MainPage** コンストラクタで、イベントハンドラを追加し、RSS フィードから読み取るために **WebClient** オブジェクトを設定します。次のコードを使用します。

VisualBasic

```
Public Sub New()
    InitializeComponent()
    Dim client As New WebClient()
    Dim uri As New Uri("http://feeds.feedburner.com/http/wwwgrapecitycom/japan")
    AddHandler client.DownloadStringCompleted, AddressOf
client_DownloadStringCompleted
    client.DownloadStringAsync(uri)
End Sub
```

C#

```
public Page()
{
    InitializeComponent();
    WebClient client = new WebClient();
    Uri uri = new Uri("http://feeds.feedburner.com/http/wwwgrapecitycom/japan");
    client.DownloadStringCompleted +=
     new DownloadStringCompletedEventHandler(client_DownloadStringCompleted);
    client.DownloadStringAsync(uri);
}
```

14. **News** クラスを追加します。

VisualBasic

```
Public Class News
    Public Property Title() As String
        Get
            Return m_Title
        End Get
        Set(ByVal value As String)
            m_Title = Value
        End Set
    End Property
    Private m_Title As String
    Public Property Link() As String
        Get
            Return m_Link
        End Get
        Set(ByVal value As String)
            m_Link = Value
        End Set
    End Property
```

```
    Private m_Link As String
End Class
```

C#

```
public class News
{
    public string Title { get; set; }
    public string Link { get; set; }
}
```

15. **client_DownloadStringCompleted** イベントハンドラを追加します。

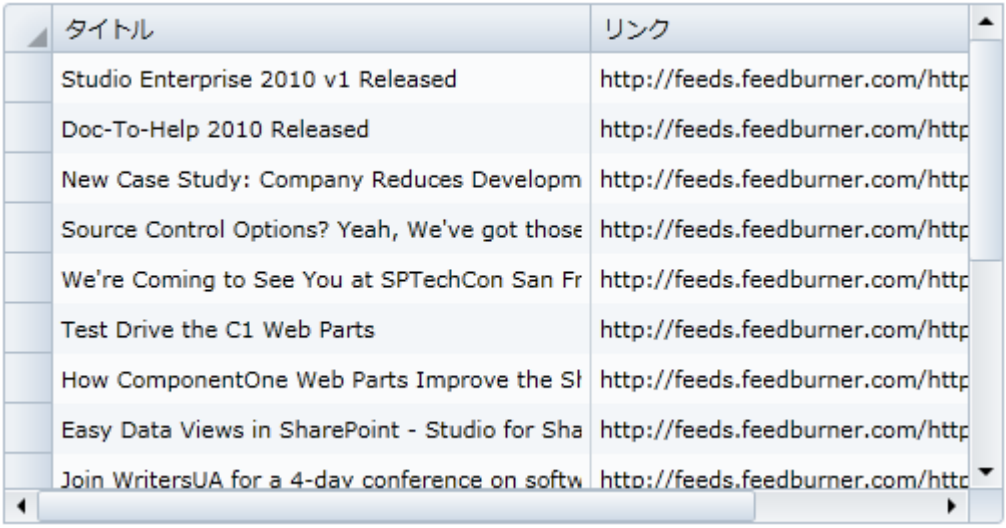
VisualBasic

```
Private Sub client_DownloadStringCompleted(ByVal sender As Object, ByVal e As
DownloadStringCompletedEventArgs)
    Dim xmlNews As XDocument = XDocument.Parse(e.Result)
    Dim news = From story In xmlNews.Descendants("item") _
    Select New News With {.Title = story.Element("title").Value, .Link =
story.Element("link").Value}
    c1DataGrid1.ItemsSource = news
End Sub
```

C#

```
void client_DownloadStringCompleted(object sender,
DownloadStringCompletedEventArgs e)
{
    XDocument xmlNews = XDocument.Parse(e.Result);
    var news = from story in xmlNews.Descendants("item")
        select new News
        {
           Title = (string)story.Element("title"),
           Link = (string)story.Element("link")
        };
    c1DataGrid1.ItemsSource = news;
}
```

16. アプリケーションを実行し、グリッドが Silverlight RSS ニュースフィードに連結していることを確認します。



ここまでの成果

おめでとうございます。このチュートリアルは終了です。このトピックでは、新しい Silverlight プロジェクトを作成し、**C1DataGrid** コントロールを追加し、グリッドを RSS フィードに連結する方法を学びました。

マスター/詳細ビューの作成

次のチュートリアルでは、行の詳細機能を使ってデータをマスター/詳細ビューに表示するための **C1DataGrid** コントロールの使用方法について説明します。

このサンプルには、XML への LINQ を使って XML ファイルからロードされた一連の製品カテゴリが表示されます。メイングリッド(カテゴリ)の各行に対して製品のリストがロードされ、それが2つ目の **C1DataGrid** コントロールを使って詳細ビューに表示されます。カテゴリ行の詳細ビューが変更されると、詳細データがロードされます。

手順 1:マスター/詳細グリッドの設定

この手順では、最初に Visual Studio で **DataGrid for Silverlight** を使用する Silverlight グリッドアプリケーションを作成します。新しい Silverlight プロジェクトを作成し、**C1DataGrid** コントロールをアプリケーションに追加します。

プロジェクトをセットアップし、**C1DataGrid** コントロールをアプリケーションに追加するには、次の手順に従います。

1. Visual Studio で、**[ファイル]**→**[新しいプロジェクト]**を選択します。
2. **[新しいプロジェクト]**ダイアログボックスで、左ペインから言語を選択し(この例では **C#** を使用)、テンプレートリストから**[Silverlight アプリケーション]**を選択します。**[名前]**テキストボックスに「MasterDetail」と入力し、**[OK]**をクリックします。**[新しい Silverlight アプリケーション]**ダイアログボックスが表示されます。
3. 既定の設定のまま**[OK]**をクリックすると、**[新しい Silverlight アプリケーション]**ダイアログボックスが閉じ、プロジェクトが作成されます。
4. プロジェクトの XAML ウィンドウで、カーソルを<Grid> タグと </Grid>と タグの間に置き、1回クリックします。
5. ツールボックスに移動し、**[C1DataGrid]**アイコンをダブルクリックして、**MainPage.xaml** にグリッドコントロールを追加します。XAML マークアップは次のようになります。

XAML

```
<UserControl xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
x:Class="MasterDetail.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
```

```
    mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
  <Grid x:Name="LayoutRoot">
      <c1:C1DataGrid></c1:C1DataGrid>
  </Grid>
</UserControl>
```

6. <c1:C1DataGrid> タグが既存のコンテンツを含む場合、コンテンツを削除します。タグは次のように表示されます。

   XAML

   ```
   <c1:C1DataGrid></c1:C1DataGrid>
   ```

7. x:Name="c1DataGrid1" を <c1:C1DataGrid> タグに追加して、グリッドに名前を付けます。次のようになります。

   XAML

   ```
   <c1:C1DataGrid x:Name="c1DataGrid1">
   ```

   コントロールに一意の識別子を付けると、コードでその **C1DataGrid** コントロールにアクセスできるようになります。
8. <c1:C1DataGrid> タグに CanUserAddRows="False" を追加します。次のようになります。

   XAML

   ```
   <c1:C1DataGrid x:Name="c1DataGrid1" CanUserAddRows="False">
   ```

   ユーザーはグリッドに新規行を追加できなくなります。
9. <c1:C1DataGrid> タグに Margin="5" を追加します。次のようになります。

   XAML

   ```
   <c1:C1DataGrid x:Name="c1DataGrid1" CanUserAddRows="False" Margin="5">
   ```

   これで、グリッドの周囲にマージンが追加されます。

ここまでの成果

これで、基本的なグリッドアプリケーションを作成できました。次の手順では、プロジェクトに XML データソースを追加します。

手順 2：プロジェクトへのデータソースの追加

この手順では、アプリケーションにデータソースを追加し、外部ファイルを追加してデータソースを設定します。この手順では、チュートリアルを簡略化して、**ComponentOne for Silverlight** インストールに付属する **ControlExplorer** サンプルのファイルを使用します。products.xml と **Data.cs** ファイル
は、**ControlExplorer\C1.Silverlight.DataGrid\C1DataGrid_Demo\C1DataGrid_Demo** ディレクトリに含まれます。

データソースを追加するには、次の手順に従います。

1. ソリューションエクスプローラウィンドウで、**MasterDetail** プロジェクトを右クリックし、**[追加]→[新規フォルダ]**を選択します。フォルダの名前を「Resources」に変更します。
2. ソリューションエクスプローラウィンドウで、**Resources** フォルダを右クリックし、[追加]→[既存の項目]を選択します。
3. **[既存のアイテムの追加]**ダイアログボックスで、**C1DataGrid_Demo\Resources** サンプルフォルダに移動し、**products.xml** ファイルを選択して、**[追加]**をクリックします。このファイルは、プロジェクトで使用するデータを提供します。
4. ソリューションエクスプローラで **products.xml** ファイルを選択し、プロパティウィンドウで**[ビルドアクション]**プロパティを**[埋め込まれたリソース]**に設定します。
5. ソリューションエクスプローラウィンドウで、**MasterDetail** プロジェクトを右クリックし、**[追加]→[既存の項目]**を選択します。
6. **[既存のアイテムの追加]**ダイアログボックスで、**C1DataGrid_Demo** サンプルフォルダに移動し、**Data.cs** ファイルを選択して、**[追加]**をクリックします。このファイルには、データソースを設定するコードが入っています。

ここまでの成果

この手順では、XML データソースを追加しました。次の手順では、行の詳細セクションを設定し、アプリケーションを完成させます。

手順 3：行の詳細の設定

この手順では、グリッドの行の詳細セクションの設定を完成させます。**RowDetailsTemplate** を追加して詳細行の外観を設定し、詳細行の動作を設定するコードを追加します。

行の詳細を設定するには、次の手順に従います。

1. 次の<c1:C1DataGrid.RowDetailsTemplate>を<c1:C1DataGrid></c1:C1DataGrid>タグの間に追加します。次のようになります。

XAML

```
<c1:C1DataGrid x:Name="c1DataGrid1" CanUserAddRows="False" Margin="5">
    <c1:C1DataGrid.RowDetailsTemplate>
        <DataTemplate>
            <c1:C1DataGrid HeadersVisibility="Column"  Margin="5"
CanUserAddRows="False"/>
        </DataTemplate>
    </c1:C1DataGrid.RowDetailsTemplate>
</c1:C1DataGrid>
```

このテンプレートは、行の詳細セクションの表示をカスタマイズします。

2. <c1:C1DataGrid>タグに LoadedRowDetailsPresenter="c1dg_LoadedRowDetailsPresenter" LoadingRow="c1dg_LoadingRow" を追加します。次のようになります。

XAML

```
<c1:C1DataGrid x:Name="c1DataGrid1" CanUserAddRows="False"
LoadedRowDetailsPresenter="c1dg_LoadedRowDetailsPresenter"
LoadingRow="c1dg_LoadingRow">
```

後で、これらのイベントのハンドラをコードに追加します。

3. ソリューションエクスプローラで、プロジェクトを右クリックして**[参照の追加]**を選択します。**[参照の追加]**ダイアログボックスで **System.Xml.Linq** と **System.ComponentModel.DataAnnotations** を見つけ、**[OK]**をクリックすると、参照が追加されます。
4. ページを右クリックし、コンテキストメニューから[コードの表示]を選択して、コードエディタを開きます。
5. コードエディタで、次の名前空間をインポートします。

C#

```
using System.Xml.Linq;
using C1.Silverlight.DataGrid;
using C1DataGrid;
```

6. **MainPage** コンストラクタに **ItemsSource** プロパティを設定するコードを追加します。

C#

```
public MainPage()
{
    InitializeComponent();
    c1dg.ItemsSource = Data.GetSubCategories().Take(10);
}
```

7. **MainPage** クラスに **c1dg_LoadedRowDetailsPresenter** イベントのコードを追加します。

C#

```
private void c1dg_LoadedRowDetailsPresenter(object sender,
C1.Silverlight.DataGrid.DataGridRowDetailsEventArgs e)
{
    if (e.Row.DetailsVisibility == Visibility.Visible)
    {
```

```
        C1.Silverlight.DataGrid.C1DataGrid detailGrid = e.DetailsElement as
C1.Silverlight.DataGrid.C1DataGrid;
        if (detailGrid.ItemsSource == null)
        {
            int subcategory = (e.Row.DataItem as
Subcategory).ProductSubcategoryID;
            detailGrid.ItemsSource = Data.GetProducts((product) =>
product.Element("ProductSubcategoryID") != null &&
product.Element("ProductSubcategoryID").Value != "" &&
int.Parse(product.Element("ProductSubcategoryID").Value) ==
subcategory).Take(10);
        }
    }
}
```

8. **c1dg_LoadingRow** イベントのコードを **MainPage** クラスに追加して、最初の行について行の詳細の表示/非表示を設定します。

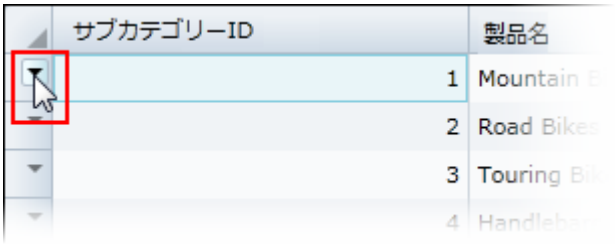
C#

```
private void c1dg_LoadingRow(object sender, DataGridRowEventArgs e)
{
    if (e.Row.Index == 0)
    {
        e.Row.DetailsVisibility = Visibility.Visible;
    }
}
```

ここまでの成果

アプリケーションを保存して実行すると、グリッドに **products.xml** ファイルのデータが表示され、最初の行の詳細セクションが表示されていることを確認できます。

行の詳細セクションを折りたたんだり、別の行の詳細セクションを展開するには、行の行ヘッダーの矢印アイコンをクリックします。



グリッドのローカライズ

さまざまなユーザーのために **DataGrid for Silverlight** をローカライズするプロセスは、たいへんシンプルです。Silverlight のローカライズは標準の .NET のローカライズに基づいています。ローカライズの詳細については、「アプリケーションのローカライズ」を参照してください。このチュートリアルでは、既存のアプリケーションに表示されるグリッド文字列をローカライズします。

手順 1：ローカライズされたグリッドの設定

この手順では、**DataGrid for Silverlight** を使って Silverlight グリッドアプリケーションを作成します。新しい Silverlight プロジェクトを作成し、**C1DataGrid** コントロールをアプリケーションに追加して、グリッドを連結します。

次の手順に従います。

1. Visual Studio で、**[ファイル]**→**[新しいプロジェクト]**を選択します。
2. **[新しいプロジェクト]**ダイアログボックスで、左ペインから言語を選択し、テンプレートリストから**[Silverlight アプリケーション]**を選択します。[名前]テキストボックスに「C1DataGridLocalization」と入力し、**[OK]**をクリックします。**[新しい Silverlight アプリケーション]**ダイアログボックスが表示されます。
3. **[OK]**をクリックすると、**[新しい Silverlight アプリケーション]**ダイアログボックスが閉じ、プロジェクトが作成されます。
4. <UserControl>タグで、Width="400"（またはd:DesignWidth="400"）を Width="450" に置き換えて、サイズを大きくします。
5. プロジェクトの XAML ウィンドウで、カーソルを<Grid> タグと </Grid>タグの間に置き、1回クリックします。
6. ツールボックスに移動し、**[C1DataGrid]**アイコンをダブルクリックして、**MainPage.xaml** にグリッドコントロールを追加します。XAML マークアップは次のようになります。

XAML

```
<UserControl xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
 x:Class="C1DataGridLocalization.MainPage"
 xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
 xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml" Width="450"
Height="300">
    <Grid x:Name="LayoutRoot">
        <c1:C1DataGrid></c1:C1DataGrid>
    </Grid>
</UserControl>
```

7. <c1:C1DataGrid>タグが既存のコンテンツを含む場合、コンテンツを削除します。タグは次のように表示されます。

```
<c1:C1DataGrid></c1:C1DataGrid>
```

8. x:Name="c1DataGrid1" を<c1:C1DataGrid> タグに追加して、グリッドに名前を付けます。次のようになります。

```
<c1:C1DataGrid x:Name="c1DataGrid1">
```

コントロールに一意の識別子を付けると、コードでその **C1DataGrid** コントロールにアクセスできるようになります。

9. <c1:C1DataGrid>タグに CanUserGroup="True" を追加します。次のようになります。

```
<c1:C1DataGrid x:Name="c1DataGrid1" CanUserGroup="True">
```

10. ソリューションエクスプローラで、**C1DataGridLocalization** プロジェクトを右クリックし、**[ビルド]**を選択します。
11. ソリューションエクスプローラで、**MainPage.xaml.cs** または **MainPage.xaml.vb** ファイルを右クリックし、コンテキストメニューから**[コードの表示]**をクリックして、コードエディタを開きます。
12. 次のコードをプロジェクトに追加して、**Data** クラスを作成します。

VisualBasic

```
Public Class Data
    Private _ProductName As String
    Public Property ProductName() As String
        Get
            Return _ProductName
        End Get
        Set(ByVal value As String)
            _ProductName = value
        End Set
    End Property
    Private _Description As String
    Public Property Description() As String
        Get
            Return _Description
        End Get
        Set(ByVal value As String)
            _Description = value
        End Set
    End Property
    Private _Quantity As Integer
    Public Property Quantity() As Integer
        Get
            Return _Quantity
        End Get
        Set(ByVal value As Integer)
            _Quantity = value
        End Set
    End Property
    Private _InStock As Boolean
    Public Property InStock() As Boolean
        Get
            Return _InStock
        End Get
        Set(ByVal value As Boolean)
            _InStock = value
        End Set
    End Property
End Class
```

C#

```
public class Data
{
```

```
    public string ProductName { get; set; }
    public string Description { get; set; }
    public int Quantity { get; set; }
    public bool InStock { get; set; }
}
```

13. 次のコードを **MainPage** コンストラクタに追加して、グリッドにデータを設定します。

VisualBasic

```
Public Sub New()
    InitializeComponent()
    ' データソースにデータを追加します。
    Dim source As New List(Of Data)()
    Dim itemsCount As Integer = 25
    For i As Integer = 0 To itemsCount - 1
        source.Add(New Data With
            {
                .ProductName = "Name",
                .Description = "Description",
                .Quantity = i,
                .InStock = (i Mod 2 = 0)
            })
    Next
    ' グリッドの ItemsSource プロパティを設定します。
    c1dg.ItemsSource = source
End Sub
```

C#

```
public Page()
{
    InitializeComponent();
    // データソースにデータを追加します。
    List<Data> source = new List<Data>();
    int itemsCount = 25;
    for (int i = 0; i < itemsCount; i++)
    {
        source.Add(new Data()
        {
            ProductName = "Name",
            Description = "Description",
            Quantity = i,
            InStock = (i % 2 == 0)
        });
    }
    // グリッドの ItemsSource プロパティを設定します。
    c1dg.ItemsSource = source;
}
```

ここまでの成果

この手順では、新しい Silverlight アプリケーションを作成し、**C1DataGrid** コントロールを追加して、コントロールをデータソースに連結しました。次の手順では、グリッドをローカライズするためのリソースファイルを追加します。

手順 2:リソースファイルの追加

この手順では、最初にリソースファイルをアプリケーションに追加します。必要に応じて、複数のリソースファイルをプロジェクトに追加できます。

1. ソリューションエクスプローラで、**C1DataGridLocalization** プロジェクトを右クリックし、**[追加]**→**[新規フォルダ]**を選択します。
2. 作成したフォルダに「Resources」という名前を付けます。
3. **Resources** フォルダを右クリックし、コンテキストメニューで、**[追加]**→**[新しい項目]**を選択します。
4. **[新しい項目の追加]**ダイアログボックスで、テンプレートペインから[リソースファイル]を選択します。そのファイルに「C1.Silverlight.DataGrid.resx」という名前を付け、**[追加]**をクリックして、ファイルをプロジェクトに追加します。
5. リソースファイルが自動的に開かれなかった場合は、ソリューションエクスプローラでファイル名をダブルクリックします。
6. **C1.Silverlight.DataGrid.resx** ファイルに次の名前と値を追加して、スペイン語のローカライズを行います。

名前	値
AddNewRow	Click here to add a new row
CheckBoxFilter_Checked	Checked:
ComboBoxFilter_SelectAll	Select All
DateTimeFilter_End	End
DateTimeFilter_Start	Start
EmptyGroupPanel	Drag a column here to group by that column
Filter_Clear	Clear
Filter_Filter	Filter
NumericFilter_And	And
NumericFilter_Equals	Equals
NumericFilter_GraterOrEquals	Greater/Equals
NumericFilter_Greater	Greater
NumericFilter_Less	Less
NumericFilter_LessOrEquals	Less/Equals
NumericFilter_NotEquals	Not Equals
NumericFilter_Or	Or
TextFilter_Contains	Contains
TextFilter_StartsWith	Start With
TextFilter_Equals	Equals
TextFilter_NotEquals	Not Equals

7. リソースファイルを保存して閉じます。
8. **Resources** フォルダを右クリックし、コンテキストメニューで、**[追加]**→**[新しい項目]**を選択します。

9. **[新しい項目の追加]**ダイアログボックスで、テンプレートペインから**[リソースファイル]**を選択します。そのファイルに「C1.Silverlight.DataGrid.es.resx」という名前を付け、**[追加]**をクリックして、ファイルをプロジェクトに追加します。このファイルは、アプリケーションをスペイン語にローカライズします。ファイルの名前付けについては、「リソースファイルの追加」を参照してください。
10. リソースファイルが自動的に開かれなかった場合は、ソリューションエクスプローラでファイル名をダブルクリックします。
11. **C1.Silverlight.DataGrid.es.resx** ファイルに次の名前と値を追加して、スペイン語のローカライズを行います。

名前	値
AddNewRow	Cliquee aquí para agregar un nuevo renglón
CheckBoxFilter_Checked	Seleccionado:
ComboBoxFilter_SelectAll	Seleccionar todo
DateTimeFilter_End	Fin
DateTimeFilter_Start	Inicio
EmptyGroupPanel	Arrastre una columna aquí para agrupar
Filter_Clear	Borrar
Filter_Filter	Filtrar
NumericFilter_And	Y
NumericFilter_Equals	Igual
NumericFilter_GraterOrEquals	Mayor o igual
NumericFilter_Greater	Mayor
NumericFilter_Less	Menor
NumericFilter_LessOrEquals	Menor o igual
NumericFilter_NotEquals	Diferente
NumericFilter_Or	O
TextFilter_Contains	Contiene
TextFilter_StartsWith	Empieza con
TextFilter_Equals	Igual
TextFilter_NotEquals	Diferente

12. リソースファイルを保存して閉じます。

ここまでの成果

この手順では、新しいリソースファイルをアプリケーションに追加しました。次の手順では、プロジェクトがサポートするカルチャにこのファイルのカルチャを追加し、そのカルチャを現在のカルチャとして設定します。

手順 3:カルチャの設定

アプリケーションのリソースファイルを作成したら、サポートするカルチャをプロジェクトに設定し、プロジェクトの現在のカルチャを明示的に設定します。それには、次の手順に従います。

1. ソリューションエクスプローラで、**C1DataGridLocalization** プロジェクトを右クリックし、**[プロジェクトのアンロード]**を選択します。Visual Studio からプロジェクトを保存するかどうか確認された場合、**[はい]**をクリックします。プロジェクトは淡色表示され、使用できなくなります。

2. プロジェクトをもう一度右クリックし、**[C1DataGridLocalization.csproj の編集]**オプションを選択します。
.csproj ファイルで、<SupportedCultures> </SupportedCultures>タグを見つけます。タグの間に「es」を追加します。
次のようになります。

XAML

```
<SupportedCultures>es</SupportedCultures>
```

3. .csproj ファイルを保存して閉じます。
4. ソリューションエクスプローラで、プロジェクトを右クリックし、コンテキストメニューから[プロジェクトの再ロード]を選択します。
プロジェクトが再ロードされ、指定されたカルチャがサポートされるようになります。
5. ソリューションエクスプローラで、**MainPage.xaml.cs** ファイルを右クリックし、コンテキストメニューから**[コードの表示]**をクリックして、コードエディタで開きます。
6. 次の using 文または Imports 文をファイルの先頭に追加します。

VisualBasic

```
Imports System.Globalization
Imports System.Threading
```

C#

```
using System.Globalization;
using System.Threading;
```

7. 次のコードを **InitializeComponent()** 呼び出しの上にある **MainPage** コンストラクタに追加して、**CurrentUICulture** プロパティを設定します。

VisualBasic

```
Thread.CurrentThread.CurrentUICulture = New CultureInfo("es")
```

C#

```
Thread.CurrentThread.CurrentUICulture = new CultureInfo("es");
```

次のようになります。

VisualBasic

```
Public Sub New()
    ' カルチャを設定します。
    Thread.CurrentThread.CurrentUICulture = New CultureInfo("es")
    InitializeComponent()
    ' データソースにデータを追加します。
    Dim source As New List(Of Data)()
    Dim itemsCount As Integer = 25
    For i As Integer = 0 To itemsCount - 1
        source.Add(New Data())
```

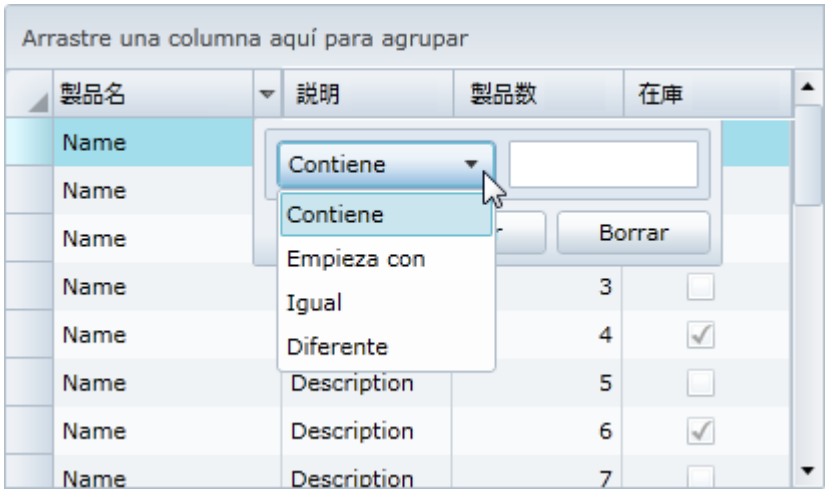
```
    Next
    ' グリッドの ItemsSource プロパティを設定します。
    c1dg.ItemsSource = source
End Sub
```

C#

```
public Page()
{
    // カルチャを設定します。
    Thread.CurrentThread.CurrentUICulture = new CultureInfo("es");
    InitializeComponent();
    // データソースにデータを追加します。
    List<Data> source = new List<Data>();
    int itemsCount = 25;
    for (int i = 0; i < itemsCount; i++)
    {
        source.Add(new Data()
        {
            ProductName = "Name",
            Description = "Description",
            Quantity = i,
            InStock = (i % 2 == 0)
        });
    }
    // グリッドの ItemsSource プロパティを設定します。
    c1dg.ItemsSource = source;
}
```

8. アプリケーションを保存し、実行します。
9. ローカライズされた言語の文字列を確認するには、グリッドのドロップダウンフィルタアイコンを選択してみます。

10. フィルタボックスでドロップダウン矢印をクリックして、追加された文字列を表示します。



ここまでの成果

この手順では、プロジェクトがサポートするカルチャにこのファイルのカルチャを追加し、そのカルチャを現在のカルチャとして設定しました。このチュートリアルでは、アプリケーションをローカライズする方法について学びました。リソースファイルを作成し、プロジェクトがサポートするカルチャを設定し、現在のカルチャをコードで明示的に設定しました。

WCF RIA Services のデータソースへのグリッドの連結

次のチュートリアルでは、**C1DataGrid** コントロールを WCF RIA Services のデータソースに連結するプロセスについて説明します。詳細については、「WCF RIA Services のデータ連結」を参照してください。この例では、**ComponentOne for Silverlight** と共にインストールされた **C1DataGrid_Ria2010** サンプルに含まれているファイルを使用します。

手順 1：アプリケーションの作成とデータソースの追加

この手順では、WCF RIA Services を有効にして新しい Silverlight プロジェクトを作成し、データソースを追加し、クライアント側プロジェクトを設定します。次の手順に従います。

1. Visual Studio で、**[ファイル]**→**[新しいプロジェクト]**を選択します。
2. **[新しいプロジェクト]**ダイアログボックスの左ペインから[Visual C#]を選択し、テンプレートリストから**[Silverlight アプリケーション]**を選択します。[名前]テキストボックスに「C1DataGridRIA」と入力し、**[OK]**をクリックします。**[新しい Silverlight アプリケーション]**ダイアログボックスが表示されます。
3. **[新しい Silverlight アプリケーション]**ダイアログボックスで、**[WCF RIA Services を有効にする]**チェックボックスをオンにし、[OK]をクリックして**[新しい Silverlight アプリケーション]**ダイアログボックスを閉じます。次に、プロジェクトを作成します。
4. ソリューションエクスプローラで、**C1DataGridRIA.Web** プロジェクトを右クリックし、**[追加]**→**[新しいフォルダ]**を選択します。フォルダの名前を "App_Data" に変更します。
5. **App_Data** フォルダを右クリックし、**[追加]**→**[既存の項目]**を選択します。
6. **[既存のアイテムの追加]**ダイアログボックスで、ComponentOne サンプルがインストールされている場所（デフォルトは **Documents** フォルダ）に移動し、**ComponentOne Samples\Silverlight\C1.Silverlight.DataGrid\C1DataGrid_Ria\C1DataGrid_Ria2010Web\App_Data** フォルダに移動します。**NORTHWND.MDF** ファイルを選択し、**[追加]**ボタンをクリックします。

   プロジェクトにデータベースが追加されます。これは、標準の Microsoft Northwind データベースです。
7. ソリューションエクスプローラで、**C1DataGridRIA.Web** プロジェクトを右クリックし、**[追加]**→**[新しい項目]**を選択しま

す。

8. [新しい項目の追加]ダイアログボックスで、左側のリストから[データ]を選択し、データテンプレートのリストから**[ADO.NET エンティティデータモデル]**を選択します。ファイル名を "NorthwindModel" と指定し、**[追加]**をクリックしてプロジェクトにこのファイルを追加します。

9. **[エンティティデータモデルウィザード]**が表示されます。**[データベースから生成]**オプションを選択し、**[次へ]**をクリックします。

10. **[データ接続の選択]**画面で、**NORTHWND.MDF** ファイルが選択されていることを確認します。選択されていない場合は、**[新しい接続]**を選択してそのファイルを見つけます。接続文字列をデフォルト名「NORTHWNDEntities」で保存し、**[次へ]**をクリックします。

11. **[データベースオブジェクトの選択]**画面で、**[テーブル]**チェックボックスをオンにし、**[製品]**テーブルを選択します。**[完了]**をクリックします。

12. **[ビルド]**→**[ソリューションの再ビルド]**を選択してソリューション全体をビルドし、自動生成された RIA サービスファイルが作成されていることを確認します。

13. ソリューションエクスプローラで、**C1DataGridRIA.Web** プロジェクトを右クリックし、**[追加]**→**[新しい項目]**を選択します。

14. **[新しい項目の追加]**ダイアログボックスで、左側のリストから**[Web]**を選択し、コードテンプレートのリストから**[ドメインサービスクラス]**を選択します。ファイル名を "NorthwindService" と指定し、**[追加]**をクリックしてプロジェクトにこのファイルを追加します。**[新しいドメインサービスクラスの追加]**ダイアログボックスが表示されます。

15. **[新しいドメインサービスクラスの追加]**ダイアログボックスで、DataContext 項目として **NorthwindEntities** を選択し、**[クライアントアクセスを有効にする]**チェックボックスをオンにします。**[製品]**エンティティと**[編集を有効にする]**チェックボックスをオンにし、**[OK]**をクリックします。

16. プロジェクトを保存し、**[ビルド]**→**[ソリューションの再ビルド]**を選択して、すべてが正しく動作することを確認します。

ここまでの成果

この手順では、新しい RIA データソースをアプリケーションに追加しました。次の手順では、このアプリケーションに **C1DataGrid** コントロールを追加します。

手順 2：C1DataGrid コントロールの追加

前の手順では、WCF RIA Services を有効にして新しい Silverlight アプリケーションを作成し、新しいデータソースを追加しました。この手順では、アプリケーションを設定し、そのアプリケーションに C1DataGrid コントロールを追加します。次の手順に従います。

1. ソリューションエクスプローラで、C1DataGridRIA プロジェクトを右クリックし、[参照の追加]を選択します。[参照の追加]ダイアログボックスが表示されます。
2. [参照の追加]ダイアログボックスで、次のアセンブリを選択したら、[OK]をクリックします。
   - System.Windows.Controls.Data
   - System.Windows.Controls.DomainServices
   - C1.Silverlight
   - C1.Silverlight.DataGrid
   - C1.Silverlight.DataGrid.Ria

   これで、選択したアセンブリの参照がプロジェクトに追加されます。
3. ソリューションエクスプローラで、MainPage.xaml ファイルをダブルクリックして開きます。
4. プロジェクトの XAML ウィンドウで、UserControl タグを更新します。次のようになります。

XAML

```
<UserControl x:Class="C1DataGridRIA.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
```

```
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    xmlns:data="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.Data"
    xmlns:ria="clr-
namespace:System.Windows.Controls;assembly=System.Windows.Controls.DomainServices"
    xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
    xmlns:adapter="clr-
namespace:C1.Silverlight.DataGrid.Ria;assembly=C1.Silverlight.DataGrid.Ria"
    xmlns:local="clr-namespace:C1DataGridRIA.Web"
    mc:Ignorable="d" d:DesignWidth="640" d:DesignHeight="480">
```

このマークアップは、ユーザーが追加したアセンブリへの参照を追加し、UserControl をサイズ変更します。

5. Grid タグの直後に次のマークアップを追加して、行定義を作成します。

XAML

```
<Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition Height="*"/>
</Grid.RowDefinitions>
```

このマークアップは、ページのレイアウトを設定します。

6. Grid タグ内の行定義の直下に次のマークアップを追加して、C1RiaAdaptor を作成します。

XAML

```
<!-- RIA データソース -->
<adapter:C1RiaAdapter x:Name="_adapter" DataGrid="{Binding
ElementName=_dataGrid}">
    <ria:DomainDataSource x:Name="_myDataSource" QueryName="GetProducts"
PageSize="8">
        <ria:DomainDataSource.DomainContext>
            <local:NorthwindContext/>
        </ria:DomainDataSource.DomainContext>
        <ria:DomainDataSource.GroupDescriptors>
            <ria:GroupDescriptor PropertyPath="CategoryID"/>
            <ria:GroupDescriptor PropertyPath="Discontinued"/>
        </ria:DomainDataSource.GroupDescriptors>
        <ria:DomainDataSource.SortDescriptors>
            <ria:SortDescriptor PropertyPath="ProductName"
Direction="Descending"/>
        </ria:DomainDataSource.SortDescriptors>
        <ria:DomainDataSource.FilterDescriptors>
            <ria:FilterDescriptor PropertyPath="UnitPrice"
Operator="IsGreaterThanOrEqualTo" Value="18"/>
            <ria:FilterDescriptor PropertyPath="ProductName" Operator="Contains"
Value="C"/>
        </ria:DomainDataSource.FilterDescriptors>
    </ria:DomainDataSource>
</adapter:C1RiaAdapter>
```

このマークアップは、RIA データソースを追加します。

7. Grid タグ内の C1RiaAdaptor タグの下に次のマークアップを追加して、ページにヘッダーを追加します。

XAML

```
<!-- ヘッダー -->
<Border Grid.Row="0" Height="40" Background="LightBlue">
    <TextBlock Text="CollectionView adapter for C1DataGrid: RIA Services"
       Margin="10 0 0 0" FontSize="15" FontWeight="Bold"
VerticalAlignment="Center"/>
</Border>
```

8. Grid タグ内の Header の下に次のマークアップを追加して、ページにレイアウト Grid を追加します。

XAML

```
<!-- コンテンツ -->
<Grid Grid.Row="1" Margin="20">
    <Grid.RowDefinitions>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="Auto"/>
        <RowDefinition Height="*"/>
    </Grid.RowDefinitions>
</Grid>
```

このレイアウトグリッド内に C1DataGrid コントロールを追加します。

9. 追加したコンテンツレイアウト Grid（ タグの直前）内に次のマークアップを追加して、ページに標準の DataPager コントロールを追加します。

XAML

```
<!-- DataPager -->
    <data:DataPager x:Name="_dataPager" Source="{Binding Data,
ElementName=_myDataSource}" BorderThickness="0" Background="White"/>
```

10. コンテンツレイアウト Grid 内の DataPager の直後に次のマークアップを追加して、ページに C1DataGrid コントロールを追加します。

XAML

```
<!-- C1DataGrid -->
<c1:C1DataGrid x:Name="_dataGrid" CanUserGroup="True" AutoGenerateColumns="False"
Grid.Row="1"
       CanUserAddRows="True" CanUserEditRows="True" CanUserRemoveRows="True"
       ItemsSource="{Binding Data, ElementName=_adapter}"
       BeginningRowEdit="_dataGrid_BeginningRowEdit"
CommittingRowEdit="_dataGrid_CommittingRowEdit"
       CancelingRowEdit="_dataGrid_CancelingRowEdit"
RowsDeleted="_dataGrid_RowsDeleted" >
    <c1:C1DataGrid.Columns>
        <c1:DataGridNumericColumn Binding="{Binding CategoryID, Mode=TwoWay}"
SortMemberPath="CategoryID" FilterMemberPath="CategoryID" Header="CategoryID"/>
        <c1:DataGridCheckBoxColumn Binding="{Binding Discontinued, Mode=TwoWay}"
SortMemberPath="Discontinued" FilterMemberPath="Discontinued"
Header="Discontinued"/>
        <c1:DataGridTextColumn Binding="{Binding ProductName, Mode=TwoWay}"
SortMemberPath="ProductName" FilterMemberPath="ProductName" Header="ProductName"/>
        <c1:DataGridTextColumn Binding="{Binding QuantityPerUnit, Mode=TwoWay}"
SortMemberPath="QuantityPerUnit" FilterMemberPath="QuantityPerUnit"
Header="QtyPerUnit"/>
```

```
        <c1:DataGridNumericColumn Binding="{Binding UnitPrice, Mode=TwoWay}"
SortMemberPath="UnitPrice" FilterMemberPath="UnitPrice" Header="UnitPrice"/>
    </c1:C1DataGrid.Columns>
</c1:C1DataGrid>
```

この C1DataGrid コントロールは、追加済みのデータベースに連結され、定義済みの連結列を含みます。

11. コンテンツレイアウト Grid 内の C1DataGrid の直後に次のマークアップを追加して、ページに1つのテキストボックスと2つのボタンを追加します。

XAML

```
<!-- テキストの変更 -->
<TextBox x:Name="_changeText" Margin="0 4 0 0" Grid.Row="2"/>
<!-- 拒否ボタン -->
<Button x:Name="_rejectButton" Content="Reject Changes" IsEnabled="False"
Click="_rejectButton_Click" Width="120" HorizontalAlignment="Right" Margin="0 4
130 0" Grid.Row="3"/>
<!-- 送信ボタン -->
<Button x:Name="_submitButton" Content="Submit Changes" IsEnabled="False"
Click="_submitButton_Click" Width="120" HorizontalAlignment="Right" Margin="0 4 0
0" Grid.Row="3"/>
```

実行時に、このテキストボックスにはグリッドに加えられたすべての変更の場所が表示されます。また、これらのボタンを使用して、実行時にグリッドに加えられた変更を拒否または適用できます。次の手順では、アプリケーションに追加した XAML を実装するためのコードを追加します。

12. MainPage.xaml ページを右クリックし、[コードの表示]を選択して、コードエディタで MainPage.xaml.cs（または MainPage.xaml.vb）ページを開きます。
13. ページの先頭に imports 文を追加します。

Visual Basic

```
Imports C1.Silverlight.DataGrid
Imports System.ServiceModel.DomainServices.Client
```

C#

```
using C1.Silverlight.DataGrid;
using System.ServiceModel.DomainServices.Client;
```

14. MainPage クラス内に次のコードを追加して、XAML で追加されたコントロールを実装します。

Visual Basic

```
Private Sub _submitButton_Click(sender As Object, e As RoutedEventArgs)
    ' サーバーに変更を送信します
    _dataGrid.IsLoading = True
    _myDataSource.DomainContext.SubmitChanges(AddressOf OnSubmitCompleted, Nothing)
End Sub
Private Sub _rejectButton_Click(sender As Object, e As RoutedEventArgs)
    ' 変更を拒否します
    _myDataSource.DomainContext.RejectChanges()
```

```
    CheckChanges()
    _dataGrid.Reload(False)
End Sub
' 行内に保留中の変更がある場合は、送信/拒否ボタンを無効にします
Private Sub _dataGrid_BeginningRowEdit(sender As Object, e As
DataGridEditingRowEventArgs)
    _submitButton.IsEnabled = False
    _rejectButton.IsEnabled = False
End Sub
' 保留中の変更がコミットされた後で、送信/拒否ボタンを有効または無効にします
Private Sub _dataGrid_CommittingRowEdit(sender As Object, e As
DataGridEditingRowEventArgs)
    CheckChanges()
End Sub
' 保留中の変更がキャンセルされた後で、送信/拒否ボタンを有効または無効にします
Private Sub _dataGrid_CancelingRowEdit(sender As Object, e As
DataGridEditingRowEventArgs)
    CheckChanges()
End Sub
' 行が削除された後で、送信/拒否ボタンを有効または無効にします
Private Sub _dataGrid_RowsDeleted(sender As Object, e As
DataGridRowsDeletedEventArgs)
    CheckChanges()
End Sub
' 保留中の変更を送信/拒否するかをチェックし、それに応じてボタンを有効または無効にします。
Private Sub CheckChanges()
    Dim changeSet As EntityChangeSet =
_myDataSource.DomainContext.EntityContainer.GetChanges()
    _changeText.Text = changeSet.ToString()
    Dim hasChanges As Boolean = _myDataSource.HasChanges
    _submitButton.IsEnabled = hasChanges
    _rejectButton.IsEnabled = hasChanges
End Sub
' サーバーへの変更の送信時にエラーをチェックします
Private Sub OnSubmitCompleted(so As SubmitOperation)
    _dataGrid.IsLoading = False
    If so.HasError Then
          MessageBox.Show(String.Format("Submit Failed: {0}", so.[Error].Message))
          so.MarkErrorAsHandled()
    End If
    CheckChanges()
End Sub
```

C#

```
private void _submitButton_Click(object sender, RoutedEventArgs e)
{
    // サーバーに変更を送信します
    _dataGrid.IsLoading = true;
    _myDataSource.DomainContext.SubmitChanges(OnSubmitCompleted, null);
}
private void _rejectButton_Click(object sender, RoutedEventArgs e)
```

```
{
    // 変更を拒否します
    _myDataSource.DomainContext.RejectChanges();
    CheckChanges();
    _dataGrid.Reload(false);
}
// 行内に保留中の変更がある場合は、送信/拒否ボタンを無効にします
private void _dataGrid_BeginningRowEdit(object sender, DataGridEditingRowEventArgs
e)
{
    _submitButton.IsEnabled = false;
    _rejectButton.IsEnabled = false;
}
// 保留中の変更がコミットされた後で、送信/拒否ボタンを有効または無効にします
private void _dataGrid_CommittingRowEdit(object sender,
DataGridEditingRowEventArgs e)
{
    CheckChanges();
}
// 保留中の変更がキャンセルされた後で、送信/拒否ボタンを有効または無効にします
private void _dataGrid_CancelingRowEdit(object sender, DataGridEditingRowEventArgs
e)
{
    CheckChanges();
}
// 行が削除された後で、送信/拒否ボタンを有効または無効にします
private void _dataGrid_RowsDeleted(object sender, DataGridRowsDeletedEventArgs e)
{
    CheckChanges();
}
// 保留中の変更を送信/拒否するかをチェックし、それに応じてボタンを有効または無効にします。
private void CheckChanges()
{
    EntityChangeSet changeSet =
_myDataSource.DomainContext.EntityContainer.GetChanges();
    _changeText.Text = changeSet.ToString();
    bool hasChanges = _myDataSource.HasChanges;
    _submitButton.IsEnabled = hasChanges;
    _rejectButton.IsEnabled = hasChanges;
}
// サーバーへの変更の送信時にエラーをチェックします
private void OnSubmitCompleted(SubmitOperation so)
{
    _dataGrid.IsLoading = false;
    if (so.HasError)
    {
        MessageBox.Show(string.Format("Submit Failed: {0}", so.Error.Message));
        so.MarkErrorAsHandled();
    }
    CheckChanges();
}
```

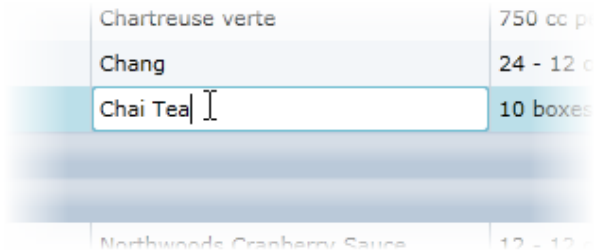
手順 3:アプリケーションの実行

前の手順では、WCF RIA Services を有効にして新しい Silverlight アプリケーションを作成し、新しいデータソースを追加し、アプリケーションに C1DataGrid コントロールを追加しました。この手順では、このアプリケーションを実行して実行時の操作を確認します。次の手順に従います。

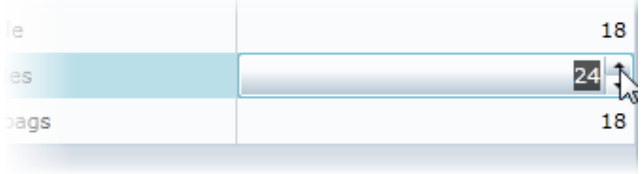
1. プロジェクトを保存し、[デバッグ]→[デバッグ開始]を選択してアプリケーションを実行します。次の画像のように表示されます。

2. 実行時に、製品名列の1つのセルをクリックし、セルからテキストを削除します。確認テキストが表示されます。

3. 削除した 製品名列のセルにテキストを入力します。



4. [変更の拒否]ボタンをクリックして、行った変更を破棄します。
5. 単価列内の項目をクリックし、上向き/下向き矢印ボタンを使用して、セルの値を変更します。



6. セルの外をクリックし、[変更の送信]ボタンをクリックして、変更をデータに保存します。

ここまでの成果

このチュートリアルでは、C1DataGrid コントロールを RIA サービスのデータソースに連結する方法について学習しました。Silverlight アプリケーションを作成してデータソースを追加し、C1DataGrid コントロールを追加して実装しました。

ステルスページングの実装

ページングを使用すると、1つのページを表示するために必要なデータのみをロードできます。詳細については、「グリッドデータのページング」を参照してください。ステルスページングはこれと多少異なり、スクロールバーによるページング機能を使用できます。ユーザーがグリッドを下方にスクロールすると、ページング機能の場合と同様に、必要に応じて追加のデータが取得されます。**C1DataGrid** は、パフォーマンスを低下させることなくソートとフィルタ処理の機能を実行できるように、サーバー側でこれらをサポートします。このチュートリアルでは、ステルスページング機能を実装する **C1DataGrid** コントロールを含む Silverlight アプリケーションを作成します。

手順 1:ユーザーインターフェイスの作成

この手順では、最初に Visual Studio で Silverlight グリッドアプリケーションを作成します。引き続き、アプリケーションのユーザーインターフェイス(UI)を作成してカスタマイズし、プロジェクトに **C1DataGrid** コントロールを追加します。

プロジェクトを設定するには、次の手順に従います。

1. Visual Studio で、**[ファイル]**→**[新しいプロジェクト]**を選択します。
2. **[新しいプロジェクト]**ダイアログボックスの左ペインから言語を選択し、テンプレートリストから**[Silverlight アプリケーション]**を選択します。プロジェクトの**[名前]**「StealthPaging」を入力し、**[OK]**をクリックします。**[新しい Silverlight アプリケーション]**ダイアログボックスが表示されます。
3. **[OK]**をクリックしてデフォルト設定を受け入れ、**[新しい Silverlight アプリケーション]**ダイアログボックスを閉じると、プロジェクトが作成されます。
4. ソリューションエクスプローラに移動して **StealthPaging** プロジェクトを右クリックし、コンテキストメニューから**[参照の追加]**を選択します。
5. **[参照の追加]**ダイアログボックスで、**System.Runtime.Serialization** アセンブリを見つけ、**[OK]**ボタンをクリックして参照をプロジェクトに追加します。ダイアログボックスが閉じ、参照が追加されます。
6. **MainPage.xaml** ファイルが開いていない場合は、ソリューションエクスプローラに移動し、**MainPage.xaml** 項目をダブルクリックします。
7. XAML ビューで、カーソルを<Grid x:Name="LayoutRoot" Background="White">タグの直後に置き、次のマークアップを追加します。

XAML

```
<!-- グリッドレイアウト-->
<Grid.RowDefinitions>
    <RowDefinition Height="*" />
    <RowDefinition Height="Auto" />
</Grid.RowDefinitions>
```

この行定義は、アプリケーションのレイアウトを定義します。

8. プロジェクトの XAML ウィンドウで、カーソルを</Grid>タグの真上に置き1回クリックします。
9. ツールボックスに移動し、**[C1DataGrid]**アイコンをダブルクリックして、**MainPage.xaml** にグリッドコントロールを追加します。XAML マークアップは次のようになります。

XAML

```
<UserControl x:Class="StealthPaging.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    d:DesignHeight="300" d:DesignWidth="400"
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml">
```

```
    <Grid x:Name="LayoutRoot" Background="White">
        <!-- グリッドレイアウト-->
        <Grid.RowDefinitions>
            <RowDefinition Height="*" />
            <RowDefinition Height="Auto" />
        </Grid.RowDefinitions>
        <c1:C1DataGrid />
    </Grid>
</UserControl>
```

C1.Silverlight.DataGrid 名前空間と タグがプロジェクトに追加されていることがわかります。

10. <c1:C1DataGrid>タグに既存のコンテンツが含まれている場合は、そのコンテンツを削除します。次のようになります。

```
XAML
<c1:C1DataGrid />
```

11. x:Name="peopleDataGrid" を<c1:C1DataGrid>タグに追加して、グリッドに名前を付けます。次のようになります。

```
XAML
<c1:C1DataGrid x:Name="peopleDataGrid" />
```

コントロールに一意の識別子を付けると、コードでその **C1DataGrid** コントロールにアクセスできるようになります。

12. AutoGenerateColumns="True" CanUserAddRows="False" を<c1:C1DataGrid>タグに追加して、グリッドをカスタマイズします。次のようになります。

```
XAML
<c1:C1DataGrid x:Name="peopleDataGrid" AutoGenerateColumns="True"
CanUserAddRows="False" />
```

このマークアップは、列を自動的に生成するようにグリッドを設定し、新しい行の追加を無効にします。

13. </c1:C1DataGrid>タグの直後に次のマークアップを追加します。

```
XAML
<TextBlock x:Name="txtStatus" Grid.Row="1" Text="準備完了"  Margin="0,5,0,0" />
```

この **TextBlock** は、ステータス情報テキストを表示するために使用されます。

ここまでの成果

アプリケーションを実行すると、ページにグリッドとテキスト(グリッドの下)が含まれていることがわかります。これで、基本的なグリッドアプリケーションを作成できました。ただし、グリッドは空白でデータが入っていません。この後の手順では、グリッドをデータソースに連結し、コードにステルスページングを追加します。

手順 2:Web サービスの追加

この手順では、データベースをプロジェクトに追加し、グリッドを連結するプロセスを開始します。この手順では、標準の Northwind データベースとサンプルコードファイルを使用します。これらは、**ComponentOne for Silverlight** のサンプルにインストールされています。

プロジェクトを設定するには、次の手順に従います。

1. ソリューションエクスプローラで、.Web プロジェクト(たとえば、ComponentOneDataGrid.Web)を展開します。**App_Data** フォルダが表示されていない場合、.Web プロジェクトを右クリックして、**[追加]→[新しいフォルダ]**を選択して、フォルダ名を**「App_Data」**に設定します。
2. ソリューションエクスプローラで、App_Data ノードを右クリックして、**[追加]→[既存の項目]**を選択します。
3. **[既存のアイテムの追加]**ダイアログボックスで、**C:\Users\<ユーザー名>\Documents\ComponentOne Samples\Common** ディレクトリに移動します。**Nwind.mdb** ファイルを選択し、[追加]をクリックしてプロジェクトに追加します。
4. ソリューションエクスプローラで、今追加した **Nwind.mdb** ファイルを選択し、プロパティウィンドウで、[ビルドアクショ

ン]プロパティを**[なし]**に設定します。

5. ソリューションエクスプローラで、.Web プロジェクト（たとえば、ComponentOneDataGrid.Web）を右クリックし、**[追加]→[既存の項目]**を選択します。
6. **[既存のアイテムの追加]**ダイアログボックスで、製品サンプルの **C1_MDSL\C1_MDSLWeb** ディレクトリに移動します。SmartDataSet.cs ファイルを選択し、**[追加]**をクリックしてプロジェクトに追加します。
このファイルには、データベースと双方向のデータ転送を行うためのコードが含まれています。
7. ソリューションエクスプローラで、.Web プロジェクトを右クリックし、**[追加]→[新しい項目]**を選択します。
8. **[新しい項目の追加]**ダイアログボックスの左ペインで、**[Web]**項目を選択します。
9. テンプレートリストで、**[Web サービス]**を選択します。その Web サービスの名前を**「DataService.asmx」**とし、**[追加]**ボタンをクリックします。Web サービスファイルがプロジェクトに追加され、自動的に開かれます。
10. DataService.asmx ファイルで、ファイルの先頭に次の using 文を追加します。

Visual Basic

```
Imports System.IO

Imports System.Data

Imports C1_MDSLWeb ' SmartDataSet 名前空間
```

C#

```
using System.IO;

using System.Data;

using C1_MDSLWeb; // SmartDataSet 名前空間
```

11. 次に、**[System.Web.Script.Services.ScriptService]** または <System.Web.Script.Services.ScriptService()> 行のコメントを外します。
これによりスクリプトからの Web サービス呼び出しが有効になります。
12. 既存の HelloWorld メソッドを削除し、次のコードに置き換えます。

Visual Basic

```
<WebMethod> _

Public Function GetData(tables As String) As Byte()

    ' 接続文字列を含む DataSet を作成します。

    Dim ds = GetDataSet()

    ' DataSet にデータを読み込みます

    ds.Fill(tables.Split(","C))

    ' ストリームに接続します

    Dim ms = New System.IO.MemoryStream()
```

```
    ds.WriteXml(ms, XmlWriteMode.WriteSchema)

    ' ストリームデータを返却します

    Return ms.ToArray()

End Function

Private Function GetDataSet() As SmartDataSet

    ' mdb ファイルの物理的な場所を取得します

    Dim mdb As String = Path.Combine(Context.Request.PhysicalApplicationPath,
"App_Data\nwind.mdb")

    ' このファイルの存在を確認します

    If Not File.Exists(mdb) Then

        Dim msg As String = String.Format("Cannot find database file {0}.", mdb)

        Throw New FileNotFoundException(msg)

    End If

    ' ファイルが読み取り専用でないことを確認します（ソースコントロールによって読み取り専用にされる場合があります）

    Dim att As FileAttributes = File.GetAttributes(mdb)

    If (att And FileAttributes.[ReadOnly]) <> 0 Then

        att = att And Not FileAttributes.[ReadOnly]

        File.SetAttributes(mdb, att)

    End If

    ' SmartDataSet を作成および初期化します

    Dim dataSet = New SmartDataSet()

    dataSet.ConnectionString = "provider=microsoft.jet.oledb.4.0;data source=" &
mdb

    Return dataSet

End Function
```

C#

```
[WebMethod]

public byte[] GetData(string tables)

{

    // 接続文字列を含む DataSet を作成します

    var ds = GetDataSet();

    // DataSet にデータを読み込みます

    ds.Fill(tables.Split(','));

    // ストリームに接続します

    var ms = new System.IO.MemoryStream();

    ds.WriteXml(ms, XmlWriteMode.WriteSchema);

    // ストリームデータを返却します

    return ms.ToArray();

}

SmartDataSet GetDataSet()

{

    // mdb ファイルの物理的な場所を取得します

    string mdb = Path.Combine(

    Context.Request.PhysicalApplicationPath, @"App_Data\nwind.mdb");

    // このファイルの存在を確認します

    if (!File.Exists(mdb))

    {

        string msg = string.Format("Cannot find database file {0}.", mdb);

        throw new FileNotFoundException(msg);

    }

    // ファイルが読み取り専用でないことを確認します(ソースコントロールによって読み取り専用にされる場合があります)

    FileAttributes att = File.GetAttributes(mdb);
```

```
    if ((att & FileAttributes.ReadOnly) != 0)
    {
        att &= ~FileAttributes.ReadOnly;
        File.SetAttributes(mdb, att);
    }
    // SmartDataSet を作成および初期化します
    var dataSet = new SmartDataSet();
    dataSet.ConnectionString = "provider=microsoft.jet.oledb.4.0;data source=" +
mdb;
    return dataSet;
}
```

このコードは、データセットを作成し、データベースからデータを受け取ります。

13. .Web プロジェクト（たとえば、ComponentOneDataGrid.Web）を右クリックし、コンテキストメニューから**[ビルド]**を選択します。これで、ComponentOneDataGrid.Web プロジェクトの作業は終了し、ComponentOneDataGrid プロジェクトの作業に戻ります。

ここまでの成果

この手順では、プロジェクトにデータベースを追加し、Web サービスを作成しました。次の手順では、最後にプロジェクトに Web サービスを接続し、アプリケーションを実行します。

手順 3：Web サービスの接続とステルスページングの追加

前の手順では、Web サービスを作成し、プロジェクトにデータソースを追加しました。この手順では、引き続き、Web サービスをアプリケーションにリンクします。

プロジェクトを設定するには、次の手順に従います。

1. **MainPage.xaml** ファイルに戻ります。
2. ソリューションエクスプローラで、プロジェクト名を右クリックし、コンテキストメニューから**[サービス参照の追加]**を選択します。
3. **[サービス参照の追加]**ダイアログボックスで、**[検出]**ボタンをクリックします。サービスのリストに DataWebService.asmx ファイルが表示されます。
4. **[名前空間]**テキストボックスで、デフォルト値を "DataService" に変更します。**[OK]**ボタンをクリックして設定を保存し、ダイアログボックスを閉じます。
5. LoadedRowPresenter="peopleDataGrid_LoadedRowPresenter" を<c1:C1DataGrid>タグに追加して、グリッドをカスタマイズします。次のようになります。

```
<c1:C1DataGrid x:Name="peopleDataGrid" AutoGenerateColumns="True"
CanUserAddRows="False" LoadedRowPresenter="peopleDataGrid_LoadedRowPresenter">
```

このマークアップは、イベントハンドラを追加します。次の手順では、このイベントハンドラのコードを追加します。

6. ソリューションエクスプローラで、**MainPage.xaml** ノードを展開し、**MainPage.xaml.cs** または **MainPage.xaml.vb**

ファイルをダブルクリックしてコードエディタで開きます。

7. ファイルの先頭に次の import 文を追加します。

VisualBasic

```
Imports System.Runtime.Serialization
Imports System.Collections.ObjectModel
Imports System.ServiceModel
Imports C1.Silverlight
Imports C1.Silverlight.DataGrid
Imports StealthPaging.DataService ' プロジェクトの名前が異なる場合は、ここを変更します。
```

C#

```
using System.Runtime.Serialization;
using System.Collections.ObjectModel;
using System.ServiceModel;
using C1.Silverlight;
using C1.Silverlight.DataGrid;
using StealthPaging.DataService; // プロジェクトの名前が異なる場合は、ここを変更します。
```

8. **MainPage** クラスに次の変数を追加します。

VisualBasic

```
Dim _startRow As Integer = 0
Dim _pageSize As Integer = 20
Dim _people As New ObservableCollection(Of ServerPerson)()
Dim _loading As Boolean
```

C#

```
int _startRow = 0;
int _pageSize = 20;
ObservableCollection<ServerPerson> _people = new
ObservableCollection<ServerPerson>();
bool _loading;
```

9. **MainPage** コンストラクタにコードを追加します。次のようになります。

VisualBasic

```
Public Sub New()
   InitializeComponent()
   AddHandler peopleDataGrid.LoadedRowPresenter, AddressOf
peopleDataGrid_LoadedRowPresenter
   peopleDataGrid.ItemsSource = _people
   GetData(_startRow, _pageSize)
```

```
End Sub
```

C#

```
public MainPage()
{
    InitializeComponent();
    peopleDataGrid.LoadedRowPresenter += new EventHandler<DataGridRowEventArgs>
(peopleDataGrid_LoadedRowPresenter);
    peopleDataGrid.ItemsSource = _people;
    GetData(_startRow, _pageSize);
}
```

10. **LoadedRowPresenter** イベントハンドラを **MainPage** コンストラクタ内のコードに追加します。

VisualBasic

```
Private Sub peopleDataGrid_LoadedRowPresenter(ByVal sender As System.Object,
ByVal e As C1.Silverlight.DataGrid.DataGridRowEventArgs)
   If _loading OrElse _people.Count < _pageSize Then
         Return
   End If
   If _people.Count - 5 < e.Row.Index Then
         GetData(_startRow, _startRow + _pageSize)
   End If
End Sub
```

C#

```
private void peopleDataGrid_LoadedRowPresenter(object sender,
C1.Silverlight.DataGrid.DataGridRowEventArgs e)
{
    if (_loading || _people.Count < _pageSize)
    {
        return;
    }
    if (_people.Count - 5 < e.Row.Index)
    {
        GetData(_startRow, _startRow + _pageSize);
    }
}
```

11. サーバーからデータを取得するために、次のコードを追加します。

VisualBasic

```
#Region "retrieve data from the server"
Private Sub GetData(startRow As Integer, endRow As Integer)
   UpdateState(True, startRow, endRow)
```

```
    ' Web サービスを呼び出します
    Dim proxy = New DataWebServiceSoapClient(New BasicHttpBinding(), New
EndpointAddress(Extensions.GetAbsoluteUri("DataWebService.asmx")))
    AddHandler proxy.GetDataCompleted, AddressOf proxy_GetDataCompleted
    proxy.GetDataAsync(startRow, endRow)
End Sub
Private Sub proxy_GetDataCompleted(sender As Object, e As
GetDataCompletedEventArgs)
    If e.[Error] IsNot Nothing Then
        MessageBox.Show(e.[Error].Message, "データ取得にエラーが発生しました。",
MessageBoxButton.OK)
        Return
    End If
    ' データが正しく取得されたら、Observable コレクションに追加します
    _startRow += _pageSize
    For Each person As ServerPerson In e.Result
        _people.Add(person)
    Next
    UpdateState(False, 0, 0)
End Sub
' 読み込みステータスを設定します
' ここで VisualState を使用することもできます
Private Sub UpdateState(loading As Boolean, startRow As Integer, endRow As
Integer)
    If loading Then
        txtStatus.Text = String.Format("{0}行～{1}行を取得しています...", startRow,
endRow)
        Cursor = Cursors.Wait
        _loading = True
    Else
        _loading = False
        txtStatus.Text = "準備完了"
        Cursor = Cursors.Arrow
    End If
End Sub
#End Region
```

C#

```
#region retrieve data from the server
private void GetData(int startRow, int endRow)
{
    UpdateState(true, startRow, endRow);
    // Web サービスを呼び出します
    var proxy = new DataWebServiceSoapClient(new BasicHttpBinding(), new
EndpointAddress(Extensions.GetAbsoluteUri("DataWebService.asmx")));
    proxy.GetDataCompleted += new EventHandler<GetDataCompletedEventArgs>
(proxy_GetDataCompleted);
    proxy.GetDataAsync(startRow, endRow);
}
void proxy_GetDataCompleted(object sender, GetDataCompletedEventArgs e)
```

```
{
    if (null != e.Error)
    {
        MessageBox.Show(e.Error.Message, "データ取得にエラーが発生しました。",
MessageBoxButton.OK);
        return;
    }
    // データが正しく取得されたら、Observable コレクションに追加します
    _startRow += _pageSize;
    foreach (ServerPerson person in e.Result)
    {
        _people.Add(person);
    }
    UpdateState(false, 0, 0);
}
// 読み込みステータスを設定します
// ここで VisualState を使用することもできます
private void UpdateState(bool loading, int startRow, int endRow)
{
    if (loading)
    {
        txtStatus.Text = string.Format("{0}行～{1}行を取得しています...", startRow,
endRow);
        Cursor = Cursors.Wait;
        _loading = true;
    }
    else
    {
        _loading = false;
        txtStatus.Text = "準備完了";
        Cursor = Cursors.Arrow;
    }
}
#endregion
```

12. アプリケーションを実行し、グリッドがデータソースに連結されていることを確認します。

名前	名字	年齢	住所
名前 0	名字 0	0	住所 0
名前 1	名字 1	1	住所 1
名前 2	名字 2	2	住所 2
名前 3	名字 3	3	住所 3
名前 4	名字 4	4	住所 4
名前 5	名字 5	5	住所 5
名前 6	名字 6	6	住所 6
名前 7	名字 7	7	住所 7
名前 8	名字 8	8	住所 8

準備完了

13. アプリケーションを実行し、グリッドをスクロールするとグリッドに行が追加されて表示されることを確認します。

また、スクロールするたびに、グリッドの下のテキストに、追加される行の番号が示されることも確認してください。

ここまでの成果

おめでとうございます。このチュートリアルは終了です。このチュートリアルでは、新しい Silverlight プロジェクトを作成し、データソースを追加してから、Web サービスを作成して C1DataGrid コントロールを連結しました。さらに、ステルスページングを実装しました。これにより、実行時にグリッドをスクロールするとグリッドがページスルーされるため、パフォーマンスが向上します。

タスク別ヘルプ

タスク別ヘルプは、ユーザーの皆様が Visual Studio および Expression Blend に精通しており、**C1DataGrid** コントロールの一般的な使用方法を理解していることを前提としています。**DataGrid for WPF/Silverlight** 製品を初めて使用される場合は、まず「クイックスタート」を参照してください。

このセクションの各トピックは、**DataGrid for WPF/Silverlight** 製品を使って特定のタスクを実行するためのソリューションを提供します。

また、タスク別ヘルプのトピックは、新しい WPF/Silverlight プロジェクトが作成されており、プロジェクトに **C1DataGrid** コントロールが追加されていることを前提としています。コントロールの作成の詳細については、「C1DataGrid コントロールの作成」を参照してください。

C1DataGrid コントロールの作成

C1DataGrid コントロールは、Expression Blend で設計時に、XAML、およびコードで簡単に作成できます。次の手順で作成した C1DataGrid コントロールは、空で表示されます。グリッドを連結するか、グリッドにデータを設定する必要があります。

Blend での設計時

C1DataGrid コントロールを Blend で作成するには、次の手順に従います。

1. **[プロジェクト]**ウィンドウに移動し、プロジェクトファイルリストで**[参照]**フォルダを右クリックします。コンテキストメニューから**[参照の追加]**を選択し、**C1.WPF.DataGrid.dll** または **C1.Silverlight.DataGrid.dll** アセンブリを見つけて選択し、**[開く]**をクリックします。

   [ツール]パネルが閉じ、プロジェクトに参照が追加されて、**[アセット]**パネルでコントロールを利用できるようになります。

2. **[ツール]**パネルで、**[アセット]**ボタン（二重山かっこアイコン）をクリックして、**[アセット]**パネルを開きます。
3. **[アセット]**パネルで、左ペインから**[コントロール]**→**[すべて]**項目を選択し、右ペインで**[C1DataGrid]**アイコンをクリックします。**[C1DataGrid]**アイコンが**[ツール]**パネルの**[アセット]**ボタンの下に表示されます。
4. **UserControl** のデザイン領域をクリックして選択します。Visual Studio とは異なり Blend では、次の手順に示すように、WPF コントロールを直接デザインサーフェスに追加できます。
5. **[ツール]**パネルの**[C1DataGrid]**アイコンをダブルクリックして、コントロールをパネルに追加します。これで、C1DataGrid コントロールがアプリケーションに追加されました。
6. 必要に応じて、コントロールを選択し、**[プロパティ]**ウィンドウでプロパティを設定することにより、コントロールをカスタマイズすることもできます。たとえば、C1DataGrid コントロールの **Name** プロパティを「c1datagrid1"、**Height** プロパティを「180"、**Width** プロパティを「250」に設定します。

XAML の場合

C1DataGrid コントロールを XAML マークアップを使って作成するには、次の手順に従います。

1. Visual Studio ソリューションエクスプローラで、プロジェクトファイルリスト内の**[参照]**フォルダを右クリックします。コンテキストメニューから**[参照の追加]**を選択し、**C1.WPF.DataGrid.dll** または **C1.Silverlight.DataGrid.dll** アセンブリを選択して、**[OK]**をクリックします。
2. `xmlns:c1grid="clr-namespace:C1.WPF.DataGrid;assembly=C1.WPF.DataGrid"` または `xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"` を初期状態の `<UserControl>` タグに追加することで、プロジェクトに XAML 名前空間を追加します。次のようになります。

WPF

XAML

```
<UserControl xmlns=http://schemas.microsoft.com/winfx/2006/xaml/presentation
xmlns:x=http://schemas.microsoft.com/winfx/2006/xaml
xmlns:c1grid="clr-namespace:C1.WPF.DataGrid;assembly=C1.WPF.DataGrid"
x:Class="C1DataGrid.MainPage" Width="640" Height="480">
```

Silverlight

XAML

```
<Window xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml"
x:Class="C1DataGrid.MainPage" Width="640" Height="480"><UserControl
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml
x:Class="C1DataGrid.MainPage" Width="640" Height="480">
```

3. `<c1:C1DataGrid>` タグをプロジェクトの `<Grid>` タグ内に追加して、C1DataGrid コントロールを作成します。マークアップは次のようになります。

XAML

```
<Grid x:Name="LayoutRoot" Background="White">
    <datagrid:C1DataGrid Name="c1datagrid1" Height="180" Width="250" />
</Grid>
```

このマークアップは、「c1datagrid1」という名前の空の C1DataGrid コントロールを作成し、コントロールのサイズを設定します。

コードの場合

C1DataGrid コントロールをコードで作成するには、次の手順に従います。

1. Visual Studio ソリューションエクスプローラで、プロジェクトファイルリスト内の**[参照]**フォルダを右クリックします。コンテキストメニューから**[参照の追加]**を選択し、**C1.WPF.4.dll** および **C1.WPF.DataGrid.4.dll** アセンブリを選択して、**[OK]**をクリックします。
2. **[MainPage.xaml]**ウィンドウ内で右クリックし、**[コードの表示]**を選択してコードビューに切り替えます。
3. 次の **import** 文をページの先頭に追加します。

WPF

Visual Basic

```
Imports C1.WPF.DataGrid
```

C#

```
using C1.WPF.DataGrid;
```

Silverlight

Visual Basic

```
Imports C1.Silverlight.DataGrid
```

C#

```
using C1.Silverlight.DataGrid;
```

4. ページのコンストラクタに、C1DataGrid コントロールを作成するコードを追加します。次のようになります。

Visual Basic

```
Public Sub New()
    InitializeComponent()
    Dim c1datagrid1 As New C1DataGrid
    c1datagrid1.Height = 180
    c1datagrid1.Width = 250
    LayoutRoot.Children.Add(c1datagrid1)
End Sub
```

C#

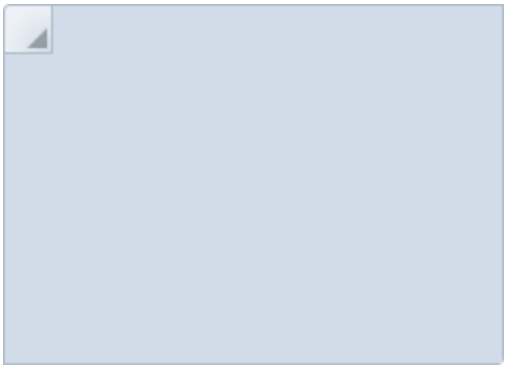
```
public MainPage()
{
    InitializeComponent();
```

```
    C1DataGrid c1datagrid1 = new C1DataGrid();
    c1datagrid1.Height = 180;
    c1datagrid1.Width = 250;
    LayoutRoot.Children.Add(c1datagrid1);
}
```

このコードは、「c1datagrid1」という名前の空の C1DataGrid コントロールを作成し、コントロールのサイズを設定し、コントロールをページに追加します。

ここまでの成果

アプリケーションを実行し、C1DataGrid コントロールが作成されたことを確認します。



上記の手順で作成した C1DataGrid コントロールは、空で表示されます。実行時に操作できる項目をコントロールに追加できます。