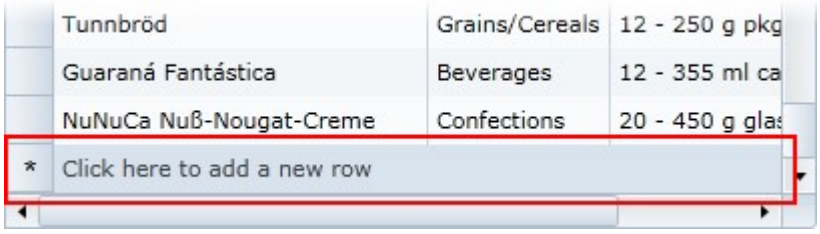
DataGrid 特性

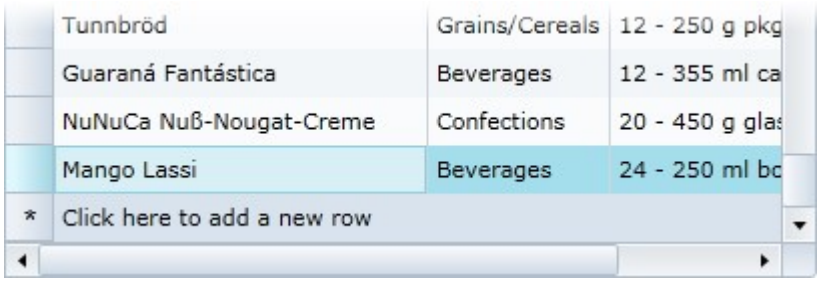
添加新的行

您可以在运行时向grid中添加行，使用新行栏。新行栏，默认情况下位于grid的底部，使用星号（*）表示，允许您在运行时通过向新行输入新的信息以添加新行至grid：



要添加一个新的行，只需在新行栏中键入文本：

按回车键结束文本编辑，并且将其添加到gird的新行：



注意：注意：CanUserAddRows属性的值必须设置为True（默认值）以便使得运行时可以执行添加行操作。做为示例，请参见禁用添加新行（第1.3.1.1章节）。

禁用添加新行

默认情况下，最终用户可以在运行时向grid添加新行以及新内容。一个新行栏将出现在grid的底部，最终用户可以在该栏输入文本以添加新行至grid。更多信息，请参见在在grid中中添加行（联机文档）。如果需要，您可以通过设置CanUserAddRows属性为False禁用运行时添加新行的功能。

在设计时

要禁用运行时添加行，须完成下列步骤：

1. 单击C1DataGrid控件以选中。
2. 导航到属性窗体并查找CanUserAddRows属性。
3. 清除位于CanUserAddRows属性右侧的复选框。

通过XAML

例如需要禁用运行时添加行功能，添加CanUserEditRows="False"至< c1:C1DataGrid>标签，使得其看起来如下面的代码片段所示：
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" CanUserAddRows="False" /> 通过代码例如，需要禁用运行时添加行功能，添加以下代码到您的工程：

```
Visual Basic
Me.C1DataGrid1.CanUserAddRows = False
```

C#

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

您所完成的步骤

运行该应用程序，如果需要，滚动grid至最后一行。可以看到新行栏将不会显示在grid底部，因此最终用户无法在运行时添加新行和内容至grid。有关单元格编辑的更多信息，请参见向grid添加行（在线文档）主题。

改变列和行的大小

默认情况下，用户可以在运行时改变grid中列和行的大小。更多信息，请参见调整列和行的大小（在线文档）。如果需要，您也可以通过设置C1DataGrid.CanUserResizeColumns以及C1DataGrid.CanUserResizeRows属性为False以禁用运行时改变行和列大小的功能。

在设计时

为禁用的列和行运行时调整大小，须完成以下步骤：

1. 单击C1DataGrid控件以选中。
2. 在"属性"窗口中，找到CanUserResizeColumns属性。
3. 清除和CanUserResizeColumns属性相邻的复选框。
4. 在"属性"窗口中，找到CanUserResizeRows 属性。
5. 清除和CanUserResizeRows 属性相邻的复选框。

通过XAML

例如，要禁用列和行运行时改变大小，添加CanUserResizeColumns="False" CanUserResizeRows="False"至<c1:C1DataGrid>标签，使得XAML代码如下所示：
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" CanUserResizeColumns="False" CanUserResizeRows="False"/> 通过代码例如，要禁用列和行运行时改变大小，须将下面的代码添加到你的工程中：

Visual Basic

```
Me.C1DataGrid1.CanUserResizeColumns = False

Me.C1DataGrid1.CanUserResizeRows = False
```

C#

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

您所完成的步骤您所完成的步骤
运行该应用程序，可以观察到您将不再可以在运行时通过拖拽操作改变列宽或者行高。有关列重新排序的更多信息，请参见调整列和行的大小（在线文档）主题。

列重新排序

最终用户可以很容易地在运行时对列进行重新排序。若要在运行时对列重新排序，须完成以下步骤：

1. 单击要排序的列的列头。
2. 将该列头拖拽到希望将要排序的目标位置。注意如果你可以将该列放置在该位置的话，会出现一条线：

1. 松开鼠标将此列放置在新位置以完成列的重排。

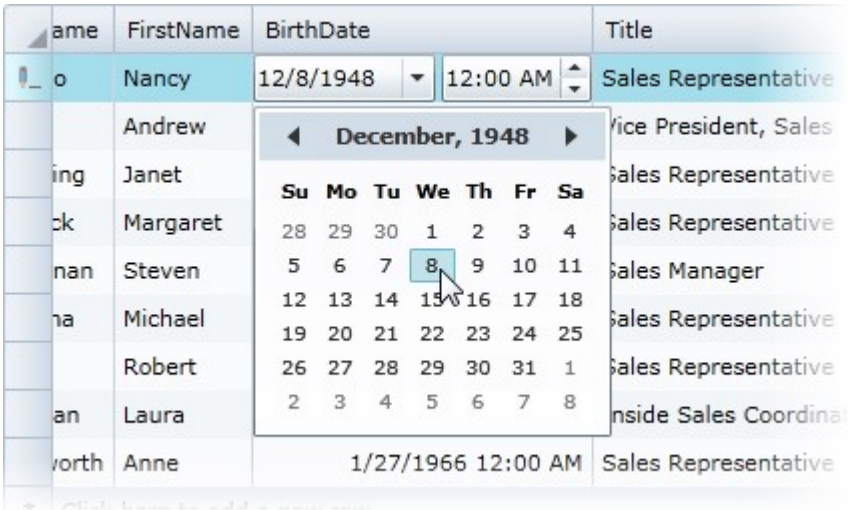
注意：注意：CanUserReorderColumns属性的值必须设置为True（默认值），以便使得可以在运行时对列进行重排序。

列类型

WPF版DataGrid
提供了一种灵活的方式通过提供许多内置的列编辑器在行和列中显示一个数据的集合，这些内置的列编辑器覆盖了全部的常见数据类型。内置的列类型包括：

列类型列类型	描述描述
DataGridBoundColumn	可以绑定到grid数据源中某个属性的列。这是为绑定到未定义数据类型准备的默认列类型。
DataGridTextColumn	文本列。这是绑定字符串数据的默认列类型。
DataGridCheckBoxColumn	一个复选框列。这是绑定布尔数据的默认列类型。
DataGridComboBoxColumn	组合框列。这是绑定枚举类型数据的默认列类型。
DataGridDateTimeColumn	日期时间列（见下图）。这是绑定日期/时间数据的默认列类型。
DataGridImageColumn	图像列。
DataGridNumericColumn	数字列。这是绑定数值数据的默认列类型（格式将从该类型推断出来的。例如，如果类型是int，格式将不
DataGridTemplateColumn	一个模版列，用作托管自定义内容。
CustomColumns	一个自定义列。参见C1DataGrid_Demo示例以查看自定义列的示例，包括一个组合列，颜色列，GIF列，Hy Text列，多行文本列，等等。

这些列类型可以提供内置的输入验证；例如DataGridDateTimeColumn 列包括用于选择日期的日历：



自定义列

自定义列的单元格内容

在本章节中，您将可以找到关于当单元格不在编辑模式时，改变做为一列中单元格内容显示的UIElement的信息。

需要着重注意的是，做为单元格内容的UIElement将被data grid回收使用；这意味着该列有可能使用由其他列创建的UIElement。
要实现自定义单元格内容，您需要重写以下方法：
GetCellContentRecyclingKey: 用于存储共享池中未来重用的单元格内容的键。
返回相同的RecyclingKey的列将可以共享相同的单元格内容实例。
CreateCellContent: 创建将用于显示单元格中的信息的UIElement。
BindCellContent:
初始化单元格内容展示器。该方法必须设置cellContent属性，SetBinding关联到的依赖属性是"row.DataItem"，该绑定源可以在绑定中直接设置或者默认的位于cellContent的DataContext。
UnbindCellContent: 该方法将在单元格内容即将被回收时调用。
在一个超链接列中，该方法的实现可能与以下示例相似：下面的方法中，此列返回不同的键（默认键是typeof(TextBlock)），这意味着该列将不和其他的列共享单元格内容元素（除非另一列也返回相同的键值，但是一般情况下，这种事情不会发生）。

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);
}
```

如果没有任何被回收的超链接，则data grid将调用CreateCellContent 方法。在这种情况下，将创建一个新的超级链接，它将被使用在单元格中包含超链接的单元格中，该超链接将被保存并可以在未来的某个时刻可以被回收供别的单元格使用：

Visual Basic

```
Public Overloads Overrides Function CreateCellContent(ByVal row As DataGridRow) As FrameworkElement

Return New HyperlinkButton()
End Function
```

C#

```
public override FrameworkElement CreateCellContent(DataGridRow row)

{ return new HyperlinkButton();
}
```

在创建了新的超链接之后，或者使用了一个被回收的超链接对象，则data grid将调用BindCellContent方法以便给超链接传递参数。在这个方法中，您应该将超链接的属性设置为与单元格的数据绑定的属性：

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;
}
```

请注意，您还可以将数据项设置为超链接的DataContext，而不是将其设置为绑定的源属性中的数据项。例如：

Visual Basic

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

C#

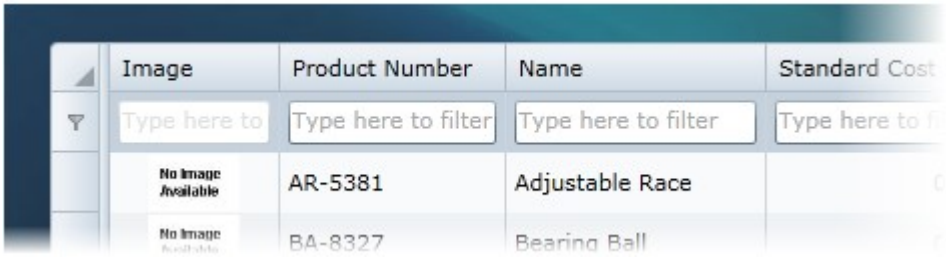
```
Hyperlink.DataContext = row.DataItem;
```

虽然您最终获得了相同的执行结果，但是这种技术和直接设置绑定源的方式运作的方式是截然不同的。

自定义行

自定义行单元格内容

本主题说明如何自定义单元格内容。例如，假设你想建立一个过滤器行。你可以创建一个grid，第一行的每个单元格中有一个文本框，当你在其中进行输入时，grid将按照输入的文本进行过滤，如下图所示：



添加一个类文件添加一个类文件

您需要添加一个新的类文件，该文件将被用做添加自定义行。例如，完成下列步骤以添加新的类文件：

2. 在解决方案资源管理器中，右键单击项目名称并选择Add|New Item。
3. 在"添加新项目"对话框中选择"可用模板"列表中的类。
4. 命名该类型，比如说"DataGridFilterRow"，并单击添加按钮以添加该类至工程。
5. 更新该列的内容，使得其类似以下所示：

Visual Basic

```
Imports C1.WPF.DataGrid
Public Class DataGridFilterRow

Inherits DataGridRow
End Class
```

C#

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

{
}
```

这将更新该类，使得其继承自DataGridRow。一旦该文件被创建它必须从DataGridRow继承。

一旦你添加了这个类，你可以用它来实现grid中的过滤。

重写一些方法重写一些方法

您需要重写一些方法，用来指定自定义行的单元格内容，这一过程和其他在自定义行中暴露的方法类似。要实现自定义单元格内容，您需要重写以下方法：

HasCellPresenter: 确定是否一个单元格出现在该行以及指定的列。

GetCellContentRecyclingKey: 用于存储共享池中未来重用的单元格内容的键。返回相同的RecyclingKey行可以共享相同的单元格内容的实例。

CreateCellContent: 创建将用于该列中的单元格中显示信息的UIElement。

BindCellContent: 初始化单元格内容展示器。

UnbindCellContent: 该方法将在单元格内容即将被回收时调用。

在过滤行中，HasCellPresenter方法将始终返回true，因为全部的列都将具有一个关联的单元格。在其他情况下，如摘要行，只有具有聚合函数的列将有一个单元格。

GetCellContentRecyclingKey方法将返回typeof(TextBox)，这将允许回收使用文本框对象，同时CreateCellContent将产生一个此类型的实例。添加以下代码：

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();
}
```

实现过滤实现过滤
在之前的步骤中，您为每一个单元格添加了一个文本框，但是这些控件目前什么都没有处理；为了实现过滤功能，须完成以下步骤：

6. 将下面的代码添加到BindCellContent方法：

Visual Basic

```
Protected Overrides Sub BindCellContent(cellContent As FrameworkElement, column As DataGridColumn)
Dim filterTextBox = DirectCast(cellContent, TextBox)
'如果该列没有指定FilterMemberPath '则将不允许在文本框中输入文本；
If String.IsNullOrEmpty(column.FilterMemberPath) Then filterTextBox.IsEnabled = False filterTextBox.Text = "Not
available" Else filterTextBox.Text = "" filterTextBox.IsEnabled = True
End If
' 处理TextChanged事件以应用过滤器至列。
filterTextBox.TextChanged += New EventHandler(Of TextChangedEventArgs) (filterTextBox_TextChanged)
End Sub
```

C#

```
protected override void BindCellContent(FrameworkElement cellContent,
DataGridColumn column) { var filterTextBox = (TextBox)cellContent;
//如果该列没有指定FilterMemberPath
//则将不允许在文本框中输入文本；
if (string.IsNullOrEmpty(column.FilterMemberPath))
{ filterTextBox.IsEnabled = false; filterTextBox.Text = "Not available";
} else { filterTextBox.Text = "";
```

```
filterTextBox.IsEnabled = true;
}
// 处理TextChanged事件以应用过滤器至列。
filterTextBox.TextChanged += new EventHandler<TextChangedEventArgs>
(filterTextBox_TextChanged);
}
```

7. 在UnbindCellContent方法中，您必须移除TextChanged事件处理，以避免内存泄露：

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 EventHandle
r<TextChangedEventArgs>
(filterTextBox_TextChanged);
}
```

添加一个自定义行

您可以替换data
grid中使用的用作展示每一个数据项数据或者分组的行，将其替换为自定义行，或者您可以在数据行的顶部或者底部添加自定义行。
更换数据项行更换数据项行为了替换由data
grid生成的行，您必须添加一个事件处理函数至CreatingRow事件。下面的代码将替换默认行为模板行：

Visual Basic

```
Private Sub C1DataGrid_CreatingRow(sender As Object, e As
DataGridCreatingRowEventArgs)
'检查是否是一个数据项行（它也可能是一个分组行）。
If e.Type = DataGridRowType.Item Then
e.Row = New DataGridTemplateRow() With { _
.RowTemplate = DirectCast(Resources("TemplateRow"), DataTemplate)


_
}
End If
End Sub
```

C#

```
private void C1DataGrid_CreatingRow(object sender, DataGridCreatingRowEventArgs e)
{
//检查是否是一个数据项行（它也可能是一个分组行）。
if (e.Type == DataGridRowType.Item)
{
e.Row = new DataGridTemplateRow()
{
<ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1"
ac:macro-id="8cf1efea-045f-4c32-a510-07773305128a"><ac:plain-text-body><![CDATA[ RowTemplate =
(DataTemplate)Resources["TemplateRow"]
]]></ac:plain-text-body></ac:structured-macro>
};
}
}
```

添加一个额外行添加一个额外行
WPF版DataGrid允许在数据的顶部或底部添加一行或多行。此功能用于新建行、总计行、摘要行和筛选器行的应用场景。
例如，通过XAML或代码：

XAML

```
<c1:C1DataGrid>
<c1:C1DataGrid.TopRows>
< local:DataGridFilterRow />
</c1:C1DataGrid.TopRows>
<c1:C1DataGrid.BottomRows>
< local:DataGridFilterRow/>
</c1:C1DataGrid.BottomRows>
</c1:C1DataGrid>
```

Visual Basic

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

C#

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

添加行细节

WPF版DataGrid中每一个grid行可以展开以显示一个行的细节区域。这个行的细节区域可以显示关于一个特定行内容的更加详细的信息。行的细节区域由一个DataTemplate定义，RowDetailsTemplate指定该区域的外观以及需要显示的数据。具体示例，请参见RowDetailsTemplate （第1.3.16.1章节）主题。
通过RowDetailsVisibilityMode属性，行细节区域可以为选中的行显示，为全部的行显示，或者可以收起。更多信息，请参见设置行细节区域可见性（第1.3.18.3.5章节）。

数据绑定

C1DataGrid控件可以绑定到任何实现了System.Collections.IEnumerable接口的对象（比如说XmlDataProvider，ObjectDataProvider，DataSet，DataView，等等）。您可以使用C1DataGrid.ItemsSource属性绑定至C1DataGrid。为绑定至grid，简单地设置ItemSource属性为一个实现了IEnumerable的实例。data grid中的每一行将绑定到数据源中的一个对象，data grid的每一列将绑定到数据对象的某一个属性。
请注意，为了使得当数据源添加或者删除项目时，C1DataGrid用户界面可以自动更新，控件必须绑定到一个实现了INotifyCollectionChanged的集合，比如说ObservableCollection<(Of <(T>)>)。
关于绑定一个C1DataGrid控件至一个XML数据源的步骤，请参见WPF版DataGrid快速入门。

延迟滚动

WPF及Silverlight版DataGrid同时支持实时滚动和延迟滚动。默认情况下，将使用实时滚动，当最终用户移动滚动滑块或者单击滚动按钮时grid将立即滚动。延迟滚动模式下，grid将保持不动，直至用户释放了滚动条滑块时滚动到目标位置；而在滑块移动过程中，grid将保持不动。如果grid包含大量数据，或者您需要优化滚动性能时，您可能需要在您的应用程序中用到延迟滚动。
您可以通过设置ScrollMode属性决定grid将如何滚动。您可以设置ScrollMode属性为C1DataGridScrollMode
枚举值中的一个，可以为RealTime（默认值），或者Differed。一下示例将grid设置为延迟滚动模式。
通过XAML
为了设置grid为延迟滚动模式，添加ScrollMode="Deferred" 至<c1:C1DataGrid> 标签，使其看起来类似以下代码：
<c1:C1DataGrid x:Name="c1DataGrid1" ScrollMode="Deferred">
通过代码
为了设置grid为延迟滚动模式，设置ScrollMode属性的值为Deferred。例如：

```
Visual Basic
Me.C1DataGrid1.ScrollMode = C1DataGridScrollMode.Deferred
```

```
C#
this.c1DataGrid1.ScrollMode = C1DataGridScrollMode.Deferred;
```

定义列

您可以使用WPF及Silverlight版DataGrid的Columns集合通过程序在运行时添加，插入，移除，以及改变任意列。您也可以通过XAML指定列，同时可以选择自动生成或者不使用自动生成的列。
创建自己的列允许您使用额外的列类型，比如说DataGridTemplateColumn或者自定义列类型。
DataGridTemplateColumn类型提供了一种简单的方式创建一个简单的自定义列。CellTemplate以及CellEditingTemplate
属性允许您指定内容模版，包括显示以及编辑模式。

生成列

默认情况下，C1DataGrid将在您设置了ItemsSource属性之后，基于数据的类型，自动生成列。对于绑定的布尔值（或者Nullable的布尔值），所生成的列为DataGridCheckBoxColumn类型，而对于绑定的字符串类型的值，所生成的列类
型为DataGridTextColumn，
对于枚举类型的值，生成的列类型为DataGridComboBoxColumn，对于绑定的未定义数据类型，将显示为一个
DataGridBoundColumn类型的列。如果一个属性不具有字符串或者数值类型的值类型，所生成的文本框列将是只读的，显示该数据对象的ToString结果的值。
你可以通过设置AutoGenerateColumns属性为false来阻止自动生成列。如果您希望显式地创建并定义全部的列，这将会非常有用。在或者，您可以让data
grid自动生成列，但是处理AutoGeneratingColumn事件，以便在列自动创建之后自定义这些自动生成的列。为了重新排布列的显示顺序，您可以设置单个列的DisplayIndex属性。

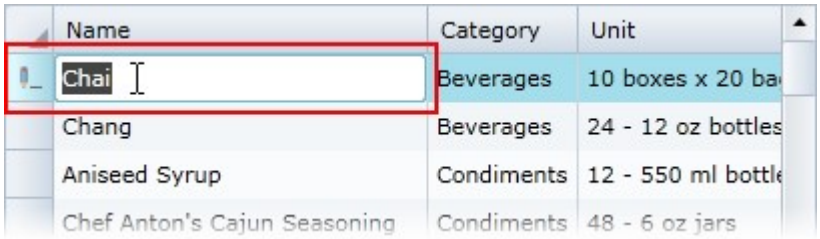
编辑

以下主题介绍如何使用C1DataGrid的编辑功能。

编辑单元格

用户可以在运行时轻松编辑单元格内容。编辑单元格的内容简单到只需选中一个单元格，并且删除或者改变该单元格的内容。完成下列步骤来编辑单元格内容：

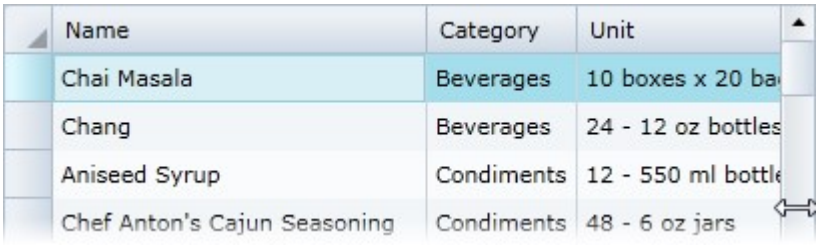
8. 双击要编辑的单元格。



一个光标将出现在该单元格中，表示它可以被编辑，与此同时铅笔图标将出现在行指示列，表明该行的某个单元格处于编辑模式。

1. 删除文本，或者键入全新的文本，再或者在现有文本之后附加文本，以完成对单元格的编辑：

1. 按下ENTER键或者离开该单元格，您对此单元格所做的更改将生效：



指示编辑状态的铅笔图标将不再可见。
请注意CanUserEditRows属性必须设置为True（默认值），以便使得最终用户可以对单元格进行编辑。具体示例，请参见禁用单元格编辑。

禁用单元格编辑

默认情况下，用户在运行时可以编辑内容。更多信息，请参见编辑单元格。如果您愿意，您可以通过设置C1DataGrid.CanUserEditRows属性为False禁用单元格编辑功能。在设计时
要禁用单元格编辑，须完成下列步骤：

1. 单击C1DataGrid控件以选中。
2. 导航至属性窗体，并找到CanUserEditRows属性。
3. 清除位于CanUserEditRows属性右侧的复选框。通过XAML

例如，为了禁用单元格编辑，添加 CanUserEditRows="False"至<c1:C1DataGrid>，因此XAML将类似以下代码片段：
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" CanUserEditRows="False" /> 通过代码
例如，要禁用单元格编辑，请将以下代码添加到项目中：

Visual Basic

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

C#

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

您所完成的步骤您所完成的步骤
运行该应用程序，并双击一个单元格；观察到单元格将不会切换到编辑模式，并且在运行时不能编辑grid内容。有关单元格编辑的更多信息，参见编辑单元格主题。

锁定grid

默认情况下，用户可以交互并编辑grid以及grid中的列。如果需要，您可以通过IsReadOnly设置grid或者grid中特定的列为不可编辑。

通过XAML

为了锁定grid不被编辑，添加IsReadOnly="True"至<c1:C1DataGrid>，代码将类似以下所示：
<c1:C1DataGrid x:Name="C1DataGrid1" IsReadOnly="True"> 通过代码
为了锁定grid不被编辑，设置IsReadOnly属性为true。例如：

Visual Basic

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

C#

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

过滤

WPF及Silverlight版DataGrid包括集中不同的选项用来过滤grid。您可以添加列过滤，行过滤，或者全文grid过滤。您可以使用基本的过滤功能或者您可以使用C1.WPF.DataGrid.Filters.dll程序集，该程序集提供了比grid内置的过滤功能更多的高级搜索选项。如何选择对grid进行过滤取决于您的需要-例如，如果您仅仅希望最终用户可以对一列进行过滤或者您的应用程序需要更多的自定义过滤。 1.3.11.1 基本列过滤
对于基本的列过滤，只需要简单地设置CanUserFilter属性的值为True。这将在grid的用户界面添加一个列的过滤元素，允许最终用户在每一列通过一个下拉框执行过滤操作。
默认情况下，CanUserFilter 属性将被设置为True，过滤将启用。如果您需要手动启用基本过滤，您可以使用以下标记或代码：

XAML

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

Visual Basic

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

C#

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

更多细节及示例，请参见对列进行过滤（第1.3.11.2章节）。

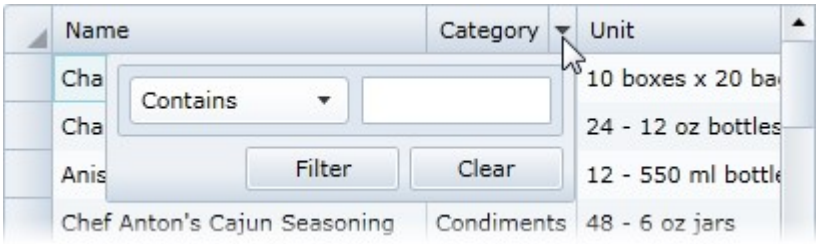
对列进行过滤

WPF版DataGrid的用户界面包括了一个过滤列元素，允许用户在运行时通过指定具体的条件对列进行过滤。
为了在运行时对列的文本进行过滤，须完成下列步骤：

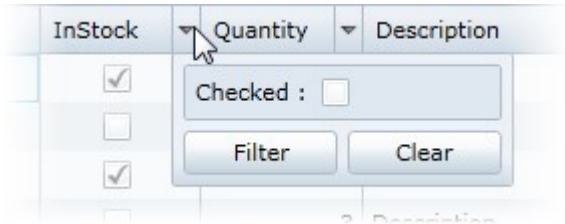
1. 在一个文本列的列头单击下拉箭头：

1. 在过滤文本框中输入您希望该列进行过滤的文本，并单击Filter按钮。该列将被过滤。

Filter选项取决于列类型。可能包括以下过滤器类型：
文本列
在文本列中，过滤器栏显示如下：

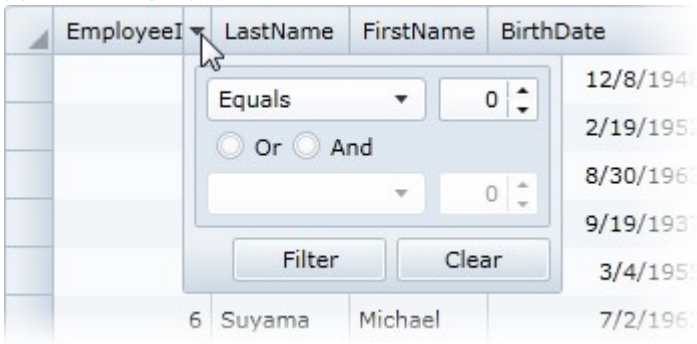


您可以通过是否列中的项目包含过滤条件，以过滤条件开头，和过滤条件相等，或者不相等，对列进行过滤。
!worddav9aacce0207087e817e2b8c475dc4e6d5.png|height=170,width=406!布尔型列布尔型列
布尔型的复选框列可以通过是否列中的项目被选中的条件进行过滤。

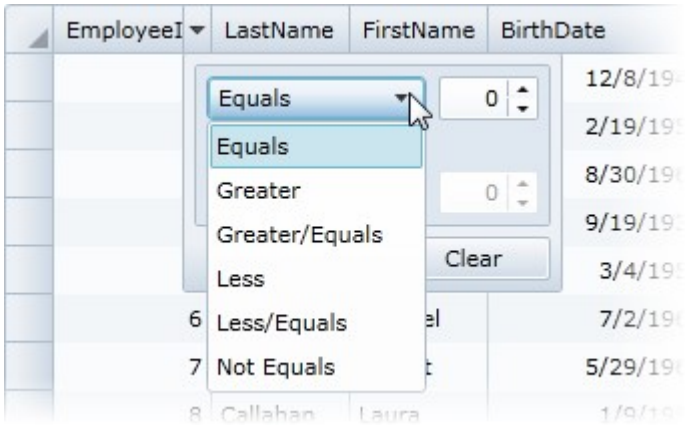


数值列数值列

数值列为过滤提供了几个选项：



您可以通过特定的条件来过滤列：

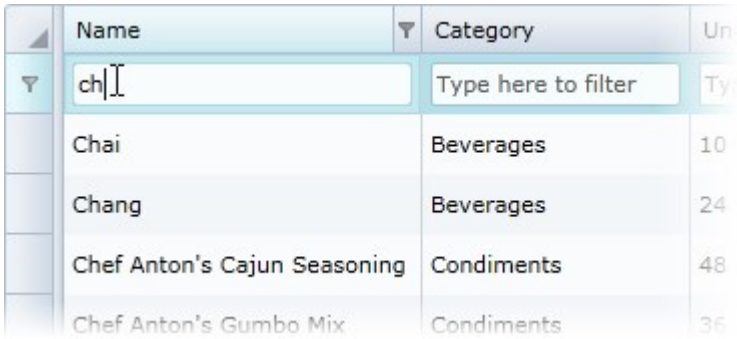


您可以使用And和Or单选按钮以便通过多个条件进行过滤：

注意：注意：CanUserFilter属性必须设置为True（默认值）以便允许过滤。

过滤行过滤

如果您愿意，您可以添加一个可见的过滤行列至grid的顶部或底部。过滤行的列将显示为每一个单元格均包含一个文本框的集合行。在文本框中输入文本时，该列的文本和grid被输入的文本进行过滤：



例如，下面的标记增加了2个过滤器行，一个在顶部，一个在grid的底部：

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>
```

您可以参见C1DataGrid_Demo2010/Filtering/FilterRow/FilterRow.xaml示例。

全文grid过滤

C1DataGrid也支持针对整个grid的全文过滤。为data grid设置一个attached属性，允许最终用户通过在一个外部的文本框输入文本，过滤整个data grid（一次过滤全部的列）。在grid中全部的匹配结果将按照用户的输入内容高亮显示。
为使用该方法用作grid过滤，您需要添加一个文本框至您的应用程序，并在FullTextSearchBehavior attached属性中引用该控件。例如，下面的XAML标记：

XAML

```
<StackPanel>
<c1:C1TextBoxBase x:Name="filterTextBox" Width="200" Watermark = "Type here to filter text"/>

<c1:C1DataGrid x:Name="c1dg" c1:C1NagScreen.Nag="True">
< c1:C1FullTextSearchBehavior.FullTextSearchBehavior>
< c1:C1FullTextSearchBehavior Filter="{Binding
ElementName=filterTextBox,Path=C1Text}"/>
< /c1:C1FullTextSearchBehavior.FullTextSearchBehavior>
</c1:C1DataGrid>
</StackPanel>
```

具体示例，请参见C1DataGrid_Demo2010/Filtering/OneTextBoxFilter/OneTextBoxFilter.xaml示例。

高级过滤

一种是通过使用C1AdvancedFiltersBehavior添加高级过滤。C1AdvancedFiltersBehavior增加了一系列的高级过滤功能至C1DataGrid的内置列。例如，这种行为增加了几个预定义的过滤器，扩展了每一列的选项：

XAML

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

</c1:C1DataGrid>
```

列过滤器列表

对grid进行过滤的一种选项是通过XAML添加一个过滤器的列表至一列。例如，以下标记语言向一个数值列添加三个过滤器，其中包括一个自定义过滤器叫做RangeFilter：

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>
```

可以通过以下路径来引用预先安装的产品示例：
Documents | ComponentOne Samples | Silverlight

禁用列过滤

默认情况下，用户可以在运行时过滤grid中的列。更多信息，请参见对列进行过滤（第1.3.11.2章节）。如果您希望这么做，您可以通过设置C1DataGrid.CanUserFilter属性为False禁用列过滤功能。

在设计时

要禁用列过滤，须完成下列步骤：

1. 单击C1DataGrid控件以选中。
2. 在"属性"窗口中，找到CanUserFilter 属性。
3. 清除紧邻在CanUserFilter属性右侧的复选框。

通过XAML

例如，为了禁用列过滤，向<c1:C1DataGrid>标签添加CanUserFilter="False"，使得代码如下所示：
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" CanUserFilter="False" /> 通过代码例如，为了禁用列过滤，请将以下代码添加到您的工程：

Visual Basic

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

C#

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

您所完成的步骤您所完成的步骤

运行该应用程序，可以观察到，您将不再可以在运行时对列进行过滤；显示过滤框的下拉箭头将不在运行时可见。有关列过滤的更多信息，请参见对列进行过滤（第1.3.11.2章节）的主题。

标签过滤列表

另一个对grid进行过滤的选项是添加显示在一个tab control的过滤器列表：

XAML

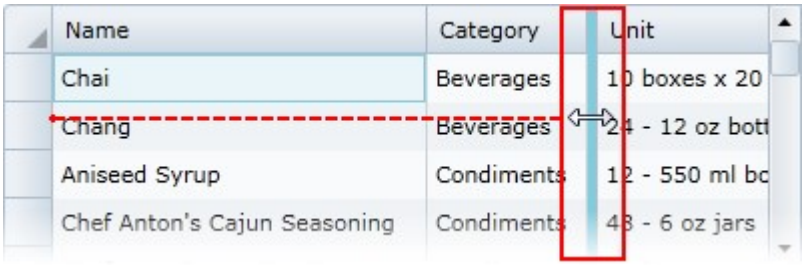
```
<c1:DataGridNumericColumn Header="Filters inside a tab control" Binding="{Binding

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>
```

您可以通过以下路径来引用预先安装的产品示例：您可以通过以下路径来引用预先安装的产品示例：
Documents | ComponentOne Samples | Silverlight

冻结

用户可以在运行时冻结列，防止它们在水平方向上发生滚动。这个功能非常有用，因为它将保证特定的列在grid改变大小或者滚动时保持始终可见。冻结栏允许用户冻结列。当可见时，默认情况下，冻结栏出现在第一列的左侧。

要冻结特定的列，将冻结栏移到要冻结的列的右边。例如，在下面的图像中，冻结栏被移到了第二列的右边：



一列被冻结，当grid水平滚动时，它们将不再滚动。例如，在下面的图像中，前两列被冻结：

请注意ShowVerticalFreezingSeparator属性必须设置为Left（默认为None），以便使得冻结栏可见，同时CanUserFreezeColumns属性必须设置为Left（默认为None）以便最终用户可以在运行时对列进行冻结。具体示例，请参见启用列冻结（第1.3.12.1章节）。

启用列冻结

您可能希望在运行时冻结一些列，使得它们在grid发生水平滚动时保持可见。更多信息，请参见冻结列（第1.3.12章节）。该功能在默认情况下没有启用，但是您可以通过设置CanUserFreezeColumns属性的值为Left启用该功能。

在设计时

为了使列冻结，须完成以下步骤：

1. 单击C1DataGrid控件以选中。
2. 在"属性"窗口中，找到CanUserFreezeColumns 属性。
3. 单击CanUserFreezeColumns 属性旁边的下拉箭头并选择Left。

通过XAML

例如，为启用列冻结，向<c1:C1DataGrid>标签添加CanUserFreezeColumns="Left"，使其结果如下所示：
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" CanUserFreezeColumns="Left" /> 通过代码
例如，为了启用列冻结，请将以下代码添加到您的工程中：

```
Visual Basic
Me.C1DataGrid1.CanUserFreezeColumns = DataGridColumnFreezing.Left
C#
this.c1DataGrid1.CanUserFreezeColumns = DataGridColumnFreezing.Left;
```

您所完成的步骤您所完成的步骤
运行该应用程序，可以观察到在运行时，冻结栏是可见的。可以移动冻结栏以选择那些列为冻结状态；该栏左侧的列将被冻结，因此当grid发生水平滚动时，该栏左侧的列将始终可见。有关列冻结的更多信息，请参见冻结列（第1.3.12章节）主题。

冻结grid行

您可能希望冻结grid顶部或者底部的行，因此当grid在运行时发生垂直方向滚动时，这些行将始终位于可见区域。默认情况下不启用此功能，但是如果您需要，可以通过设置FrozenTopRowsCount以及FrozenBottomRowsCount属性启用行冻结功能。

在设计时

要冻结顶部和底部的行，须完成下列步骤：

1. 单次点击C1DataGrid控件以选中，并导航至属性窗体。
2. 在"属性"窗口中，找到FrozenTopRowsCount属性，单击旁边的文本框属性，输入"2"，以设置顶部冻结的行的个数。
3. 找到FrozenBottomRowsCount属性，单击旁边的文本框属性，输入"2"，以设置底部冻结的行的个数。

通过XAML

例如，为了冻结顶部和底部的两行，需要添加FrozenTopRowsCount="2" FrozenBottomRowsCount="2"至
<c1:C1DataGrid>标签，添加之后XAML代码应当如下所示：
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" FrozenTopRowsCount="2"
FrozenBottomRowsCount="2" /> 通过代码例如，要冻结顶部和底部的行，将下面的代码添加到您的工程中：

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

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

您所完成的步骤您所完成的步骤
运行该应用程序，可以观察到该应用程序顶部和底部行被冻结。此时垂直方向滚动grid，可以注意到顶部和底部的两行将不会滚动，始终锁定在原地。默认情况下，添加新行将位于grid的底部，因此它将是被冻结的行中的一个。

分组

您可以启用grid的分组以及分组区域，使得最终用户可以在运行时，在您的grid中对列进行分组，以获得更佳的信息组织显示。更多信息，请参见对列进行分组。默认情况下，用户不能对grid中的列进行分组，但是您可以通过设置C1DataGrid.CanUserGroup属性的值为True以启用该功能。

在设计时

为了启用分组，须完成以下步骤：

1. 单击C1DataGrid控件以选中。
2. 在"属性"窗口中，找到CanUserGroup 属性。
3. 选中靠近CanUserGroup属性旁边的复选框。

通过XAML

例如为了启用分组，需要添加CanUserGroup="True"至< c1:C1DataGrid>，之后XAML代码应当如下所示：
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" CanUserGroup=True" /> 通过代码例如，为了启用分组，需要在工程中添加以下代码：

```
Visual Basic
Me.C1DataGrid1.CanUserGroup = True
C#
this.c1DataGrid1.CanUserGroup = true;
```

您所完成的步骤您所完成的步骤
运行该应用程序，可以注意到分组区域出现在grid的顶部。注意，您还可以自定义分组区域的可见性。关于分组区的更多信息，请参见显示分组区（第1.3.13.1章节）主题。

显示分组区

默认情况下grid中的分组是禁用的，分组区域不可见。更多信息，请参见对列进行分组（第1.3.13.2章节）。当CanUserGroup属性设置为True时，将启用分组，并且分组区域将变为可见。但是您也可以无视是否分组功能启用，单独地显示或者隐藏分组区域。默认情况下，分组区域当分组功能没有启用时为不可见，但是您仍然可以通过设置ShowGroupingPanel属性的值为True将此区域变为可见。

在设计时

要显示分组区域，须完成下列步骤：

1. 单击C1DataGrid控件以选中。
2. 在"属性"窗口中，找到ShowGroupingPanel 属性。
3. 选中ShowGroupingPanel属性相邻的复选框。

通过XAML

例如，为了显示分组区域，添加ShowGroupingPanel="True"至<c1:C1DataGrid>，使得XAML代码如下所示：
<c1:C1DataGrid Name="c1DataGrid1" Height="180" Width="250" ShowGroupingPanel="True" /> 通过代码例如，要显示分组区域，请将以下代码添加到工程中：

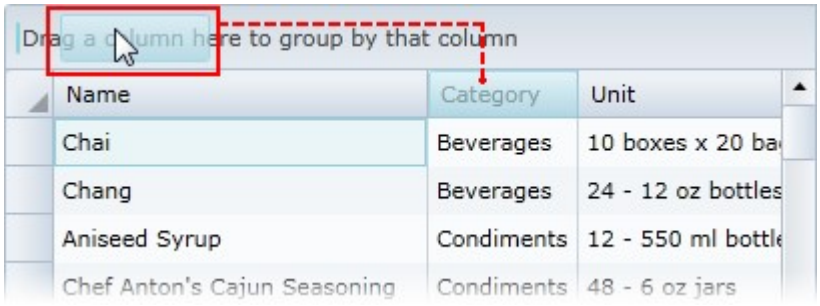
```
Visual Basic
Me.C1DataGrid1.ShowGroupingPanel = True
C#
this.c1DataGrid1.ShowGroupingPanel = true;
```

您所完成的步骤您所完成的步骤
运行该应用程序，可以注意到分组区域出现在grid的顶部。注意，即使分组区域是可见的，如果CanUserGroup 属性为False，则分组功能仍然不是启用状态。更多信息，请参见在grid中启用分组（第1.3.13章节）主题。

对列进行分组

最终用户可以在运行时，在grid中对列进行分组以获取更好的信息组织形式。grid顶部的分组区域可以让您通过简单的拖放操作来轻松地对列进行分组：

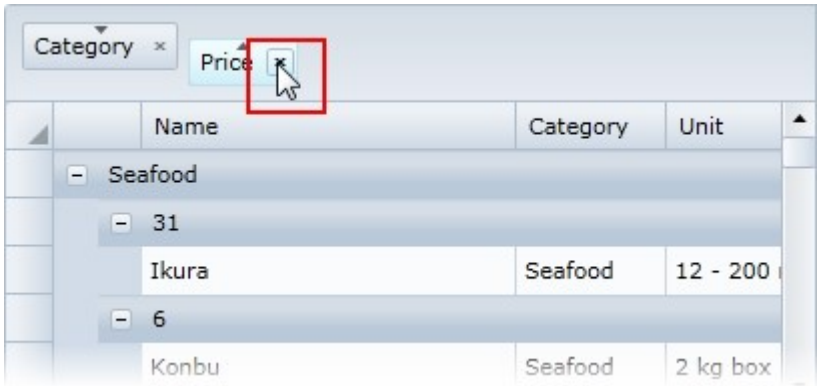
为了对一列进行分组，只需要将一列的列头拖拽到分组区域。



通过单击分组区域中的列头，可以对分组项目进行排序。在下面的图像中，分组列已被反向排序：

通过执行拖放操作，可以将更多的列拖拽到分组区域，以实现多列分组。

要删除分组，只需单击grid分组区域中紧邻分组列的"x"按钮：



请注意，CanUserGroup属性必须设置为True以便分组区域可见且分组功能启用（默认情况下该属性设置为False）。更多信息，请参见grid的启用分组。有关显示分组区域的更多信息，请参见显示分组区域主题。

分组汇总

WPF版DataGrid包含C1.WPF.DataGrid.Summaries.dll程序集，该程序集可以通过添加一个汇总行以增强grid的分组功能。
Summarize组件包含以下功能：
SummaryRow：显示关联到每一列的聚合函数的行（参见
C1DataGrid_Demo2010/Grouping/GrandTotal.xaml）
GroupRowWithSummaries：与前一个功能相同，但是汇总将显示在分组行而不是普通行。（请参见
C1DataGrid_Demo2010/Grouping/Grouping.xaml）

键盘和鼠标导航

WPF版DataGrid支持一些运行时的鼠标和键盘导航选项，这些选项提供了更多的易用性。以下主题详细介绍了一些最终用户的交互行为。

键盘导航

下表列出了可用于在运行时导航和操作grid的键盘快捷键。值得注意的是，在苹果电脑上，最终用户可以使用
Command（或Apple）键以替代这里提到的Ctrl键：

组合键组合键	描述描述
DOWN Arrow	将焦点移到当前单元格下方的单元格中。如果焦点是在最后一行，则按下向下箭头什么都不发生。
UP Arrow	将焦点移到当前单元格上方的单元格中。如果焦点是在第一行，则按下向上箭头什么都不会发生。
LEFT Arrow	将焦点移到同一行中的前一个单元格中。如果焦点是在第一个单元格中，则按下左箭头什么都不会发生。
RIGHT Arrow	将焦点移到同一行中的下一个单元格中。如果焦点是在最后一个单元格中，则按下右箭头什么都不会发生。
HOME	将焦点移到当前行中的第一个单元格。
END	将焦点移到当前行中的最后一个单元格中。
PAGE DOWN	将所显示的行数向下滚动。将焦点移动到当前显示页面的最后一行，而不会改变当前列的位置。如果最后一行只是部分显示，
PAGE UP	将所显示的行数向上滚动。将焦点移到当前显示页面的第一行，而不会改变当前列的位置。如果第一行只是部分显示，则滚动
TAB	如果当前单元格处于编辑模式，将焦点移到当前行的下一个可编辑的单元格。如果焦点已经在该行的最后一个单元格，则提交
	如果当前单元格不处于编辑模式，则将焦点移到父容器的按Tab顺序中的下一个控件中。
SHIFT+TAB	如果当前单元格处于编辑模式，将焦点移到当前行上一个可编辑的单元格。如果焦点已经在该行的第一个单元格中，则提交任 如果当前单元格不处于编辑模式，则将焦点移到父容器的按照Tab顺序中的前一个控件中。
CTRL + DOWN ARROW	将焦点移到当前列中的最后一个单元格中。
CTRL + UP ARROW	将焦点移动到当前列的第一个单元格。
CTRL + RIGHT ARROW	将焦点移到当前行中的最后一个单元格中。
CTRL + LEFT ARROW	将焦点移到当前行中的第一个单元格。
CTRL + HOME	将焦点移到控件中的第一个单元格。
CTRL + PAGE DOWN	和PAGE DOWN功能相同。
CTRL + PAGE UP	和PAGE UP功能相同。
ENTER	在选定的单元格上切换编辑模式（如果grid和对应列的IsReadOnly属性设置为False）。

F2	在选定的单元格上切换编辑模式（如果grid和对应列的IsReadOnly属性设置为False）。
ESC	取消单元格或新行编辑。
DEL	删除选定行。
INSERT	滚动到新建行，并开始编辑它。

鼠标导航

下表列出了一些可用于在运行时在grid上进行导航或者其他操作的鼠标和键盘快捷键。值得注意的是，在苹果电脑上，最终用户可以使用Command（或Apple）键以替代这里提到的Ctrl键：

鼠标动作鼠标动作	描述描述
单击一个没有被选中的行	单击一个没有被选中的行
单击当前行的某个单元格。	使得发生单击的单元格进入编辑模式。
拖动一个列头单元格	移动该列，使得可以拖放到一个新的位置（假定当前CanUserReorderColumns属性设置为True且当前被拖动行的CanUserRorder 属性为True）。
拖动列头分隔符	调整列的宽度（假定CanUserResizeColumns 属性设置为True且当前列的 CanUserResize 属性为True）。
点击列头单元格	如果ColumnHeaderClickAction属性设置为Sort，当用户单击列头时，将对列进行排序（假定此时CanUserSortCc属性设置为True且当前列的CanUserSort 属性设置为True）。 单击已排序的列的列头，将对该列进行反向排序。 按下Ctrl键的同时单击多个列头将按照单击的顺序对多列进行排序。
	如果ColumnHeaderClickAction属性设置为Select，则在SelectionMode支持列选的情况下，单击可以选中整列。
按住Ctrl键的同时单击行	修改一个非连续的多行选择（如果SelectionMode为支持多行，多单元格，或多列）。
SHIFT+单击一行	修改一个连续的多行选择（如果SelectionMode为支持多行，多单元格，或多列）。

多行选择

如果SelectionMode属性设置为MultiRow，导航行为将不会发生改变，但是在按下SHIFT（包括同时按下CTRL+SHIFT）键的情况下，键盘或鼠标导航将改变一个多行选择的状态。在导航开始前，控件将当前行标记为锚定行。当您在按下移动时，选择包括锚定行和当前行之间的所有行。
选择键选择键
以下选择键修改多行选择：
SHIFT+DOWN ARROW
SHIFT+UP ARROW
SHIFT+PAGE DOWN
SHIFT+PAGE UP
CTRL+SHIFT+DOWN ARROW
CTRL+SHIFT+UP ARROW
CTRL+SHIFT+PAGE DOWN
CTRL+SHIFT+PAGE UP
鼠标选择鼠标选择
当SelectionMode属性设置为MultiRow，单击行的同时按下CTRL或SHIFT将修改多行选择。
当您单击一行同时按下SHIFT键时，选择将包含全部的位于第一次单击确定的锚点行和当前行之间的全部行。在保持SHIFT按下的状态下继续单击将改变当前行，而不会改变最初的锚点行。
如果在导航过程中CTRL键被按下，则方向键将导航至边界单元格；例如，如果您此时位于第一行，当您按下CTRL+DOWN键，您将导航至最后一行，如果SHIFT键同时被按下，则全部的行将被选中。

自定义键盘导航

您可以添加您自己的自定义导航至C1DataGrid控件。自定义键盘导航将使您能够控制最终用户如何和grid进行交互。例如，您可以阻止用户导航到只读列或具有Null值的单元格中。在分级grid中，您可以设置父子grid之间的导航方式。为添加自定义键盘导航，您需要处理KeyDown事件，并添加代码将默认的导航逻辑替换为您的自定义导航逻辑。

添加KeyDown事件处理程序

完成以下步骤以添加KeyDown事件处理程序：

1. 切换到代码视图，添加KeyDown事件的事件处理程序，例如：
2. 切换到源代码视图并添加事件处理程序至C1DataGrid控件实例，例如：

```
<c1:C1DataGrid x:Name="c1DataGrid1" AutoGenerateColumns="True" KeyDown="c1DataGrid1_KeyDown">
</c1:C1DataGrid>
```

现在您可以将代码添加到KeyDown事件处理程序以自定义默认导航。具体示例，您可以参见位于ControlExplorer示例中的分级grid示例（C1_MDSL_RowDetail）。

本地化应用

你可以本地化（翻译）WPF版DataGrid最终用户可见的字符串。WPF版DataGrid的本地化基于标准的.NET WinForms应用程序相同的本地化机制。要本地化应用程序，您需要完成以下步骤：

1. 为您希望支持的每种文化中添加资源文件。参加添加资源文件（第1.3.15.1章节）。
2. 更新您的工程文件所支持的文化。请参见添加支持的文化（第1.3.15.2章节）。
3. 并且，根据您的工程，设置当前的文化。请参见见设置当前文化（第1.3.15.3章节）。

下面的主题描述更详细的grid本地化。

添加资源文件

和WinForms其他应用程序一样，您可以为WPF版DataGrid创建一组资源文件。您可以为每一个所需要支持的文化创建一个独立的资源文件，后缀名为.resx。当应用程序运行时，可以在这些资源和语言之间切换。请注意，您的应用程序所使用的来自于WPF版DataGrid DLL中的所有组件必须使用相同的本地化资源。
本地化规范本地化规范
要本地化grid，您需要为每个本地化的文化设置资源文件。以下为创建.resx资源文件推荐的规范：所有.resx文件应该放在工程的Resources资源子文件夹。
文件应命名如下：
XXX.YYY.resx，在这里：

1. 是相关的ComponentOne程序集的名称。
2. 是资源的文化代码。如果您所翻译的是默认的invariant文化，则.resx文件不需要包含一个文化后缀。

例如：
C1.WPF.DataGrid.de.resx –C1.WPF.DataGrid 程序集的德文 (de) 资源。
C1.WPF.DataGrid.resx –C1.WPF.DataGrid 程序集的invariant文化资源。
本地化字符串本地化字符串
下表列出的字符串可以被添加到一个.resx文件以本地化你的应用：

字符串	默认值	描述
AddNewRow	Click here to add a new row	出现在添加新行的文本。
CheckBoxFilter_Checked	Checked	在过滤器中显示复选框列的文本，以指示该列是否应该被过滤为选中的和未选中的项目。
ComboBoxFilter_SelectAll	Select All	出现在组合框中表示选择全部项目的文本。
DateTimeFilter_End	End	出现在日期时间列中的过滤器中的文本，表示日期时间范围的结束点。
DateTimeFilter_Start	Start	出现在日期时间列中的过滤器中的文本，表示日期时间范围的起始点。
EmptyGroupPanel	Drag a column here to group by that column.	当没有列进行分组时，出现在分组区域的文本。
Filter_Clear	Clear	出现在过滤栏的文本，表示清除过滤条件。
Filter_Filter	Filter	出现在过滤栏中添加过滤条件的文本。
NumericFilter_And	And	数值列过滤栏上出现的文本，表示多个过滤条件。
NumericFilter_Equals	Equals	出现在数值列过滤栏的文本，表示过滤条件应的那个应用到仅精确匹配。
NumericFilter_GreaterOrEquals	Greater/Equals	出现在数值列过滤栏的文本，表示过滤条件应当应用到值大于等于条件值的项目。
NumericFilter_Greater	Greater	出现在数值列过滤栏的文本，表示过滤条件应当应用到值大于条件值的项目。
NumericFilter_Less	Less	出现在数值列过滤栏的文本，表示过滤条件应当应用到值小于条件值的项目。
NumericFilter_LessOrEquals	Less/Equals	出现在数值列过滤栏的文本，表示过滤条件应当应用到值小于等于条件值的项目。
NumericFilter_NotEquals	Not Equals	出现在数值列过滤栏的文本，表示过滤条件应当应用到值不等于条件值的项目。
NumericFilter_Or	Or	数值列过滤栏上出现的文本，表示多个过滤条件。

TextFilter_Contains	Contains	出现在文本列过滤栏的文本，表示是否过滤条件将应用到包含指定过滤条件字符串值的
TextFilter_Equals	Equals	出现在文本列过滤栏的文本，表示是否过滤条件将应用到精确匹配指定过滤条件字符串
TextFilter_NotEquals	Not Equals	出现在文本列过滤栏的文本，表示是否过滤条件将应用到不匹配指定过滤条件字符串值
TextFilter_StartsWith	Starts With	出现在文本列过滤栏的文本，表示是否过滤条件将应用到以指定过滤条件字符串值开头

添加支持文化

一旦您为您的应用程序创建了资源文件，您需要为您的工程设置支持的文化。为这样做，须完成下列步骤：

1. 在"解决方案资源管理器"中，右键单击"工程"，选择"卸载"项目。该项目将变灰不可用。
2. 右键单击该项目，然后选择编辑ProjectName.csproj 选项（或编辑ProjectName.vbproj，那里的项目是你的项目的名称）。
3. 在.csproj文件，找到
   <SupportedCultures></SupportedCultures>标签。在标签之间，列出你想要支持的文化，每一个分号分隔。例如：

<SupportedCultures>fr;es;en;it;ru</SupportedCultures> 这将支持法语、西班牙语、英语、意大利语和俄语。

1. 保存并关闭.vbproj或.csproj文件。
2. 在"解决方案资源管理器"中，右键单击"工程"，然后从"内容"菜单中选择"重新加载"工程。

该工程将重新加载，现在将支持指定的文化。

设置当前文化

C1DataGrid控件将自动地根据应用程序选中的文化进行本地化，同时必须确保您没有将文件移动到其他位置，或者从工程中排除了某些资源文件。默认情况下，当前的文化被指定为
System.Threading.Thread.CurrentThread.CurrentUICulture。如果你想使用其他的文化，你可以在你的应用程序使用下面的代码来设置所需的文化：

Visual Basic

```
Public Sub New()
' 设置期望的文化，例如法国（法国）地区。
System.Threading.Thread.CurrentThread.CurrentUICulture = New System.Globalization.CultureInfo("fr-FR") '
InitializeComponent() 调用。
' 在调用 InitializeComponent() 之后添加任何初始化代码。
InitializeComponent()
End Sub
```

C#

```
public MainPage()
{
// 设置期望的文化，例如法国（法国）地区。
System.Threading.Thread.CurrentThread.CurrentUICulture = new System.Globalization.CultureInfo("fr-FR"); //
InitializeComponent() 调用。
InitializeComponent();
// 在调用 InitializeComponent() 之后添加任何初始化代码。 }
```

行详细信息模板

行详细信息模板

行详细信息模板控制行细节区域的外观。 行的详细信息区域显示在一行，而且可以显示附加信息。
在Expression Blend，您可以在设计时创建一个空的模板，通过选择C1DataGrid控件然后单击对象|编辑其他模板|编辑行详细信息模板|创建空。
你可以在行详细信息模板中包括文本，控件，和更多，并包括控件绑定到数据。 例如，下面的模板包括绑定和未绑定文本和复选框：

XAML

```
<c1:C1DataGrid.RowDetailsTemplate> <!-- 开始行细节部分。 -->
<DataTemplate>
<Border BorderBrush="DarkGray" BorderThickness="1" Background="Azure">

<StackPanel Orientation="Horizontal">
<StackPanel>
<StackPanel Orientation="Horizontal">
<!-- -控件被绑定到属性。 -->
<TextBlock FontSize="16" Foreground="MidnightBlue" Text="{Binding
Name}" Margin="0,0,10,0" VerticalAlignment="Bottom" />
<TextBlock FontSize="12" Text="Order Date: "
```

```
VerticalAlignment="Bottom"/>
<TextBlock FontSize="12" Text=" Complete:" VerticalAlignment="Bottom" />
CheckBox IsChecked="{Binding Complete, Mode=TwoWay}"
VerticalAlignment="Center" />
</StackPanel>
<TextBlock FontSize="12" Text="Notes: " />
<TextBox FontSize="12" Text="{Binding Notes, Mode=TwoWay}"
Width="420" TextWrapping="Wrap"/>
</StackPanel>
</StackPanel>
</Border>
</DataTemplate>
<!-- 结束行细节部分。->
</c1:C1DataGrid.RowDetailsTemplate>
```

禁用行细节切换

当网格包括子网格或者你已经创建了一个主细节网格，默认情况下该行的细节可以切换，以便它们是可见的或折叠的。如果你选择，但是，您可以禁用切换的详细信息行的特征通过设置CanUserToggleDetails属性值为false。
请注意，您将需要有一个网格和行的详细信息，以查看该示例中的更改。在设计时
禁用行详细信息的切换，完成以下步骤：

1. 点击C1DataGrid控件一次并选择它。
2. 导航到"属性"窗口中，确定CanUserToggleDetails属性位置。
3. 清空紧邻CanUserToggleDetails属性的复选框。在XAML中

例如，禁用行详细的信息的切换，添加CanUserToggleDetails ="false"到< C1：c1datagrid >标签使其出现类似下面的：
< C1：c1datagrid Name="c1datagrid1" height="180" width="250" canusertoggledetails ="false"/> 在代码中
例如，禁用行详细信息的切换，将下面的代码添加到你的项目中：

Visaul Basic

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

C#

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

你所完成的你所完成的
运行该应用程序，并观察到，您可以在运行时不再切换网格中的行详细信息。
行标头上的箭头图标表明可以被切换的行详细信息不再可视，因此切换多行不再是一种选项。

排序

排序网格列在运行时是简单的WPF数据网格。 在"列"的头上单击一次要排序的列，这样你希望的列会被排序。
你会注意到，当一列被排序的时候，会有一个排序指示符号出现：

你可以再次点击列标题来反转排序；注意到分类符号的变化方向。
通过排序一列对多个列进行排序，然后按住Ctrl键的同时单击第二栏头添加该列到你的排序条件上。
例如，在下面的图像中，类别列是先进行排序，然后将名称列为反向排序：

请注意，这个CanUserSort 属性必须设置为True（默认）的分类是可能的。

禁用列排序

默认情况下，终端用户可以在运行时对网格中的列进行排序。 更多信息，见列排序（第1.3.17.章节）。
如果你选择，但是，您可以通过设置CanUserSort属性为false禁用列排序的特征。

在设计时

要禁用列排序，完成下列步骤：

1. 点击C1DataGrid控件一次并选择它。
2. 导航到"属性"窗口中，并确定CanUserSort属性的位置。
3. 清除紧邻CanUserSort属性的复选框。

在XAML

例如，禁用列排序，添加canusersort ="false"到< C1：c1datagrid >标签使其出现类似下面的：
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" CanUserSort="False" />
代码中例如，要禁用列排序，请将以下代码添加到项目中：

```
Visual Basic
Me.C1DataGrid1.CanUserSort = False
```

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

你所完成的你所完成的
运行该应用程序，并且观察到，你可以在运行时不再进行排序列。
在运行时，单击一个列的头，将不会对网格进行排序，并且在列标题中排序指示器不可见。
有关列排序的更多信息，参见排序列（第1.3.17.章节）的话题。

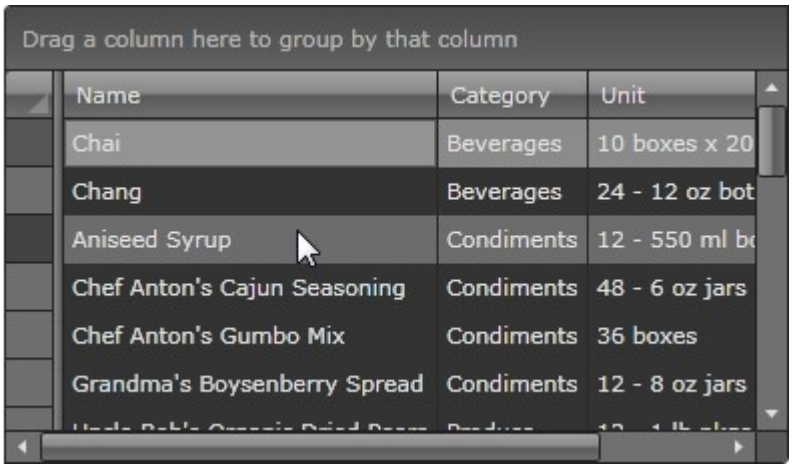
DataGrid控件的外观

以下主题描述了WPF和Silverlight的C1DataGrid外观。

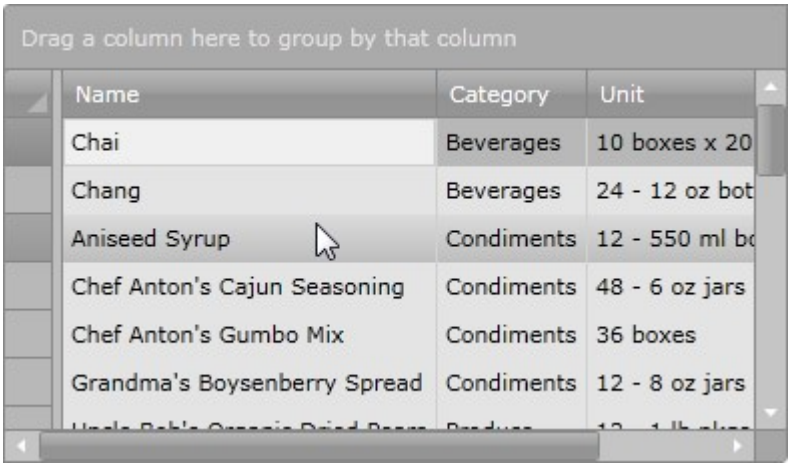
C1DataGrid主题

WPF DataGrid包含几个主题，允许您自定义你的网格外观。 当你第一次添加C1DataGrid控件到页面，看起来类似下面的图像：
这是控件的默认外观。 你可以通过使用内置的主题或者创建自己的自定义主题来改变这个外观。 所有的内置主题是基于WPF工具包的主题。
内置的主题进行了描述和下面的图片，注意下面的图片中，一个单元格已被选中，并且鼠标悬停在另一个单元格显示选定的和悬停风格：
主题名称主题预览C1ThemeBureauBlack

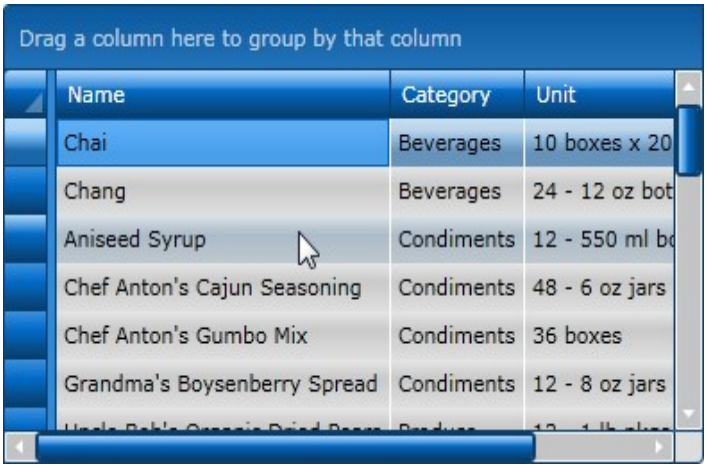
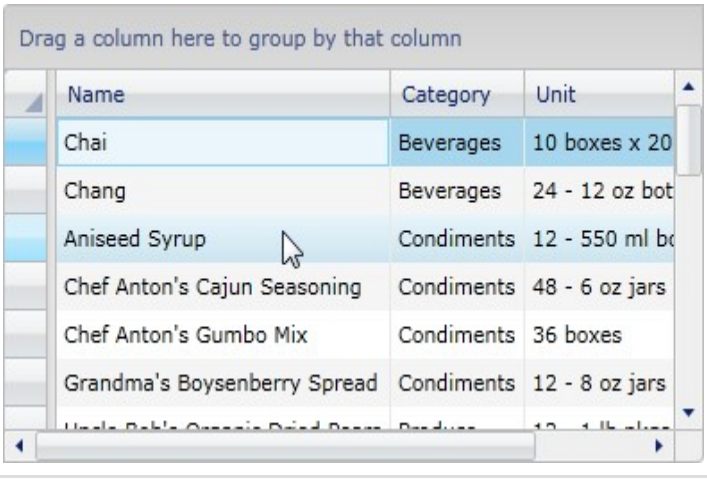
C1ThemeExpressionDark



C1ThemeExpressionLight



C1ThemeRainierOrange

C1ThemeShinyBlue	
C1ThemeWhistlerBlue	

编辑模板和风格

使用WPF控件的主要优点之一是控件是一个"看上去不那么"完全可定制的用户界面。 就像你设计自己的用户界面（UI），或者看起来和感觉到，对于WPF应用程序，您可以提供您自己的WPF DataGrid数据管理的UI。
可扩展应用程序标记语言（XAML；发音为"zammel"），基于声明的XML语言，提供了一个简单的方法来设计你的用户界面而无需编写代码。WPF DataGrid包括几个模板，这样你就不需要从头开始创建自己的用户界面。

可访问的模板

您可以通过选择C1DataGrid 控件，在DataGrid中的菜单，选择"编辑其他模板来访问微软表达混合模板。
创建一个模板的副本，您可以编辑，打开C1DataGrid菜单，选择"编辑其他模板，选择你想要编辑的模板，并选择编辑一个拷贝，来创建当前模板的一个可编辑的副本，或创建空的，来创建一个新的空白模板。

注意：注意： 如果您在菜单中创建新模板，则这个模板将自动与该模板的属性相关联。
如果你手动在XAML中创建一个模板，你不得不需要将适当的模板属性链接到你所创建的模板上。

这个C1DataGrid控制提供了多种样式属性，你可以使用并完全改变控件的外观和它的行，状态栏，标题，和单元格。
一些包含的样式在下面的表格中描述：

风格	描述
C1DataGrid.CellStyle	获取或设置用于渲染单元格时使用的样式。
C1DataGrid.ColumnHeaderStyle	获取或设置用于绘制列标题时使用的样式。
C1DataGrid.DragOverColumnStyle	风格应用于内容控件元素上，当被移动时用来显示拖拽列。
C1DataGrid.DrageSourceColumnStyle	样式应用到内容控件，在它开始进行拖放操作时放置在源列上。
C1DataGrid.DropIndicatorStyle	风格应用于内容控件元素并用于指示拖拽列被拖放的位置。
C1DataGrid.FilterStyle	获取或设置用于筛选器控制容器的样式。
C1DataGrid.FocusStyle	设置用于显示在C1DataGrid焦点上的内部矩形的风格。
C1DataGrid.GroupColumnHeaderStyle	获取或设置在组面板中显示列标题时使用的样式。
C1DataGrid.GroupRowHeaderStyle	获取设置的组行标题的样式集。
C1DataGrid.GroupRowStyle	获取设置的组行的样式。
C1DataGrid.NewRowHeaderStyle	获取或设置用于输入新项目的渲染的行标题的样式。
C1DataGrid.NewRowStyle	获取或设置为输入新项目的渲染行时使用的样式。
C1DataGrid.RowHeaderStyle	获取或设置用于渲染行标题时使用的样式。
C1DataGrid.RowStyle	获取或设置渲染行时使用的样式。

表格式选项以下主题详细表格式选项，包括网格头和表对象的位置。

设置行标题和列标题的可见度

默认行和列标题在网格中可见。 然而，如果你选择，你可以设置一个或两个标题被隐藏，通过设置HeadersVisibility属性。
你可以通过下列选项之一设置HeadersVisibility属性：

选项选项	描述描述
None	在网格中不可见行或列头文件。
Column	只有列标题在网格中可见。
Row	只有行标题在网格中可见。
All (Default)	列和行标题在网格中可见。

设置网格线的可见性

默认情况下，垂直和水平的网格线在网格中可见。
然而，如果你选择，你可以设置一个或两个设置网格线被隐藏，通过设置GridLinesVisibility属性。
你可以通过下列选项之一设置GridLinesVisibility属性：

选项选项	描述描述
None	无论是水平还是垂直的网格线在网格中是可见的。
Horizontal	只有水平的网格线在网格中可见。
Vertical	只有垂直的网格线在网格中可见。
All (Default)	在网格中，水平和垂直的网格线是可见的。

设置新行的可见度

默认情况下，添加新行设置位置在网格底部。 然而，如果你选择，你可以通过设置NewRowVisibility 属性改变其位置。
你可以通过下列选项之一设置NewRowVisibility 属性：

选项选项	描述描述
Top	添加新的行显示在网格的顶部。
Bottom（默认）	在网格的底部添加了新的行。

设置垂直和水平滚动条的可见性

默认情况下，网格的水平和垂直滚动条只有在网格内容的高度或宽度超过网格的大小时才看得见的。 然而，如果你选择，你可以设置滚动条总是不可见，而且甚至通过设置VerticalScrollbarVisibility和HorizontalScrollbarVisibility属性一起禁用它们。
你可以通过下列选项之一设置VerticalScrollbarVisibility和HorizontalScrollbarVisibility属性：

选项选项	描述描述
Disabled	禁用选择的滚动条。
Auto（default）	自动（默认）只有当选择的网格的内容超过网格窗口时滚动条才会出现。
Hidden	隐藏选择的滚动条似乎被隐藏。
Visible	可见选择的滚动条总是可见的。

设置行详细信息的可见性

默认行的详细信息是折叠的，而不是可见的。 你可以使用RowDetailsVisibilityMode属性来设置是否何时排的细节是可见的。
你可以通过下列选项之一设置RowDetailsVisibilityMode属性：

选项选项	描述描述
VisibleWhenSelected	VisibleWhenSelectedRow详细信息只有被选择时才可见。
Visible	可视行详细信息总是可见的。
Collapsed（默认）	折叠（默认）行详细信息出现折叠，并且是不可见的。

C1DataGrid画刷

WPF DataGrid提供了几种画笔的属性，你可以使用它们来完全改变控件的行，状态栏，标题，和单元格。
一些包括的画笔在下面的表格中被描述：

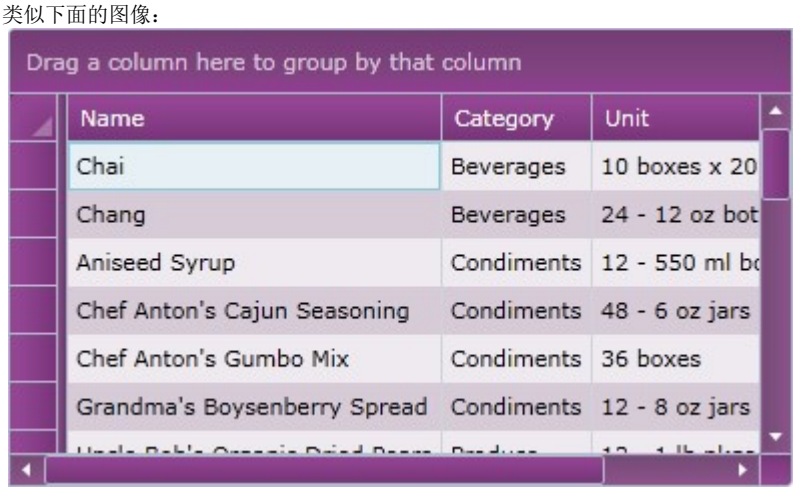
主题名称	主题预览
背景	获取或设置渲染时使用的背景画笔。 （此笔将应用于数据网格的所有部分）
前景	获取或设置渲染时使用的前景画笔。 （此画笔将应用于数据网格的所有部分）
边框画笔	获取或设置渲染时使用的边框画笔。 （此笔将应用于数据网格中的某些部分，这取决于主题）
选择的画笔	获取或设置当渲染选择的多行和单行和单列标题等时使用的选定画笔。
鼠标悬浮画笔	获取或设置当鼠标在多行和单行和单列标题等时使用的画笔。
行背景	获取或设置该行的背景画笔。
行前景	获取或设置一个前景画笔行。
交替的行背景	获取或设置一个交替行的背景画笔。
交替的行前景	获取或设置一个交替行的前景画笔。
水平的网格线画笔	将画笔应用于水平线的获取。
垂直的网格线画笔	将画笔应用于垂直线的获取。

WPF DataGrid使用ClearStyle技术设计。 有关详细信息，参见C1DataGrid （第1.3.18.5章节）。

C1DataGrid 清新的风格

WPF DataGrid支持控件一的新clearstyle技术，让你轻松控制颜色变化而无需改变控制模板。
通过设置几个颜色属性，你可以快速的设计整个网格样式。
你完全可以通过简单的设置一些属性变化的C1DataGrid控件的外观，如C1DataGrid。背景属性设置的C1DataGrid控件的颜色方案。
例如，如果你设置背景属性为"# ff663366"所以出现了类似于下面的XAML标记：
<c1:C1DataGrid HorizontalAlignment="Left" Margin="10,10,0,0" Name="c1DataGrid1" VerticalAlignment="Top"
CanUserFreezeColumns="Left" CanUserGroup="True" Background="#FFFFFFCC"/>网格将显示类似下面的图像：

如果你设置背景属性为"# ff663366"和前景的属性为"白"色，所以出现了类似于下面的XAML标记：
<c1:C1DataGrid HorizontalAlignment="Left" Margin="10,10,0,0" Name="c1DataGrid1" VerticalAlignment="Top"
CanUserFreezeColumns="Left" CanUserGroup="True" Background="#FF663366" Foreground="White"/> 网格将显示
类似下面的图像：

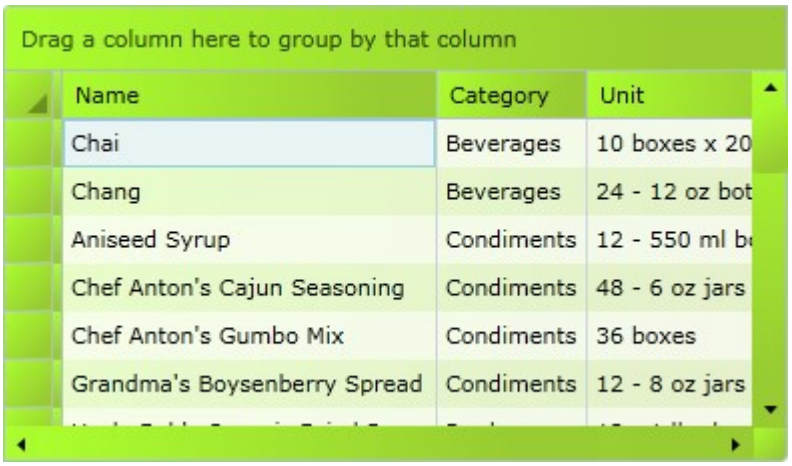


你甚至可以设置背景属性梯度值，例如，下面的代码：

XAML

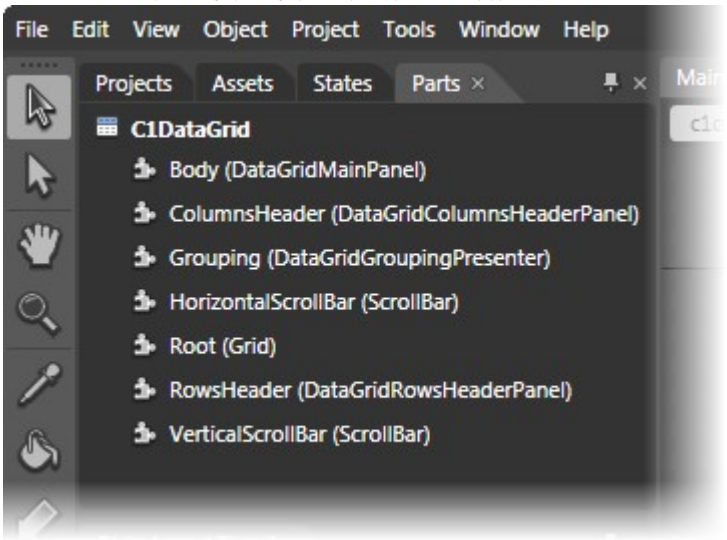
```
<c1:C1DataGrid x:Name="c1DataGrid1" HorizontalAlignment="Left" Margin="10,10,0,0"
VerticalAlignment="Top" CanUserFreezeColumns="Left" CanUserGroup="True">
<c1:C1DataGrid.Background>
<LinearGradientBrush StartPoint="0,0" EndPoint="1,1"> <GradientStop Color="GreenYellow" Offset="0.0" />

<GradientStop Color="YellowGreen" Offset="0.85" />
</LinearGradientBrush>
</c1:C1DataGrid.Background>
</c1:C1DataGrid>
```

网格将出现类似以下图像：



C1DataGrid模板部件

在微软表达配方中，你可以通过创建一个新的模板编辑模板部件（例如，单击C1DataGrid控件选择它并选择对象|编辑模板|编辑副本）。
一旦你创建了一个新的模板，模板的部分将出现在部件窗口中：



注意，你可能需要为其选择控件模板在部件窗口上作为它可见的部分。
在部件窗口中，你可以双击任何元素来创建模板中的一部分。
一旦你这样做，该部分将在部件面板上的模板和元素的图标上出现将，且根据指示选择进行改变：

在C1DataGrid控件上的可用模板部件包括：

名称名称	类型类型	描述描述
Body	DataGridMainPanel	包含网格体的面板。
ColumnsHeader	DataGridColumnsHeaderPanel	面板包含一组数据网格列标题面板。
Grouping	DataGridGroupingPresenter	显示分组的面板或另一个元素，如果有没有列在分组面板。
HorizontalScrollBar	ScrollBar	显示一个控件，该控件提供具有滑动的缩略图滚动条，其位置与值相对应。
Root	Grid	其灵活的网格区域，包含列和行。
RowsHeader	DataGridRowsHeaderPanel	面板包含数据网格行标题面板。
VerticalScrollBar	ScrollBar	显示一个控件，该控件提供具有滑动的缩略图滚动条，其位置与值相对应。

自定义网格的外观

下列主题的细节你可以通过改变网格的外观自定义C1DataGrid。 WPF

DataGrid和Silverlight控件包括几个外观选项，包括控件一的独一无二的clearstyle技术。 例如，可以更改网格的背景颜色或交替行背景。
有关更多信息clearstyle技术笔记，参见C1DataGrid clearstyle（第1.3.18.5章节）的话题。
后续主题也详细更改网格的布局，包括如何设置标题的位置和添加新的行栏。

改变网格的前景和背景颜色

WPF DataGrid包括ComponentOne的独特clearstyle技术使您能够简单地改变网格的整体外观。
以下步骤将详细介绍如何设置C1DataGrid。背景属性完全改变网格的外观。 关于ComponentOne的clearstyle技术的更多细节，看c1datagrid clearstyle（第1.3.18.5）的话题。

在设计时

要改变网格的前景色和背景色，使其显得绿色，完成下列步骤：

1. 点击C1DataGrid控件一次并选择它。
2. 导航到"属性"窗口，然后单击"背景属性"旁边的下拉箭头。
3. 点击下拉箭头在框中的十六进制代码出现，并选择绿色。
4. 导航到"属性"窗口，然后单击"前台属性"旁边的下拉箭头。
5. 点击下拉箭头的方块中的十六进制代码出现，并选择白色。

在XAML

例如，改变网格的前景颜色和背景颜色，使它呈现绿色，添加到< C1：c1datagrid >标签使其出现类似下面的：
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" Background="Green" Foreground="White" />在代码中
例如，改变网格的前景色和背景色，使其显得绿色，在项目中添加以下代码：

```
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);
```

你所完成的你所完成的
运行该应用程序，并观察到网格中的网格标题中出现了白色的文本。
值得注意的是，与C1DataGrid控件的clearstyle技术，网格的颜色，网格的滚动条，和交替行的背景网格的所有更改以反映绿色背景。
高亮显示网格中的一个项目，并注意到鼠标悬停样式并没有更改；如果您选择，您可以自定义这些样式。
看到改变网格的鼠标悬停样式（第1.3.18.7.3章节）详情。

移除网格的交替行颜色

WPF DataGrid默认出现交替行颜色。 交替行颜色是在不同颜色中出现的替代线，而不是网格的基色。
这有助于使行更容易跨网格，但如果你选择的话，你可以通过删除的交替行颜色显示网格的统一外观。

在设计时

要清除交替行颜色，设置为白色，完成下列步骤：

1. 点击C1DataGrid控件一次并选择它。
2. 在"属性"窗口中，单击旁边的下拉箭头RowBackground属性。
3. 点击下拉箭头的方块中的十六进制代码出现，并选择白色。
4. 在"属性"窗口中，单击旁边的下拉箭头AlternatingRowBackground属性。
5. 点击下拉箭头的方块中的十六进制代码出现，并选择白色。

在XAML

除去交替行颜色和设置它，所以所有的行出现白色，添加RowBackground ="white" AlternatingRowBackground ="White"到< C1：c1datagrid >标签使其出现类似下面的： <c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" RowBackground="White" AlternatingRowBackground="White" /> 在代码中要清除交替的行颜色，设置为白色，将下面的代码添加到您的项目中：

```
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.AlternatingRowB
ackground = new System.Windows.Media.
SolidColorBrush(Colors.White);
```

你所完成的你所完成的
运行应用程序，并观察到网格中的所有行现在都是白色的。

改变网格的鼠标悬停样式

默认情况下，列和行上鼠标滑过出现在不同的颜色来提示用户哪些是它们与网格交互的区域。
如果你选择你可以自定义鼠标滑过的单元格外观。 例如，您可能想要高亮这些单元格，甚至更多的或删除此效果。

在设计时

要将鼠标悬浮效果设置为黄色，完成以下步骤：

1. 点击C1DataGrid控件一次并选择它。
2. 导航到"属性"窗口中，单击紧邻MouseOverBrush属性旁边的下拉箭头。
3. 点击下拉箭头的方块中的十六进制代码出现，并选择黄色。

在XAML

设置鼠标悬浮效果为黄色，加MouseOverBrush ="Yellow"到<c1:C1DataGrid>标签使其出现类似下面的：
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" MouseOverBrush="Yellow" /> 在代码中要将鼠标设置为黄色，请将以下代码添加到项目中：

Visual Basic

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

C#

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

你所完成的
运行该应用程序，并观察到网格中所有高亮行和列都出现黄色。

改变网格的字体样式

当控件运行时你可能需要更新出现在WPF DataGrid的字体。
例如，您可能希望更改网格的样式，其中一个元素是字体样式，以匹配您的应用程序的外观。

在设计时

要更改字体样式，完成下列步骤：

1. 点击C1DataGrid控件一次并选择它。
2. 导航到"属性"窗口中，单击下拉箭头旁边的FontFamily属性并选择宋体。
3. 导航到"属性"窗口中，单击"下一步"的字体属性的下拉箭头并选择10。

在XAML

要更改字体样式，添加FontFamily="Times New Roman" FontSize="10"到<c1:C1DataGrid>标签使其出现类似下面的：
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" FontFamily="Times New Roman" FontSize="10" /> 在代码中

要清除交替的行颜色，设置为白色，将下面的代码添加到您的项目中：

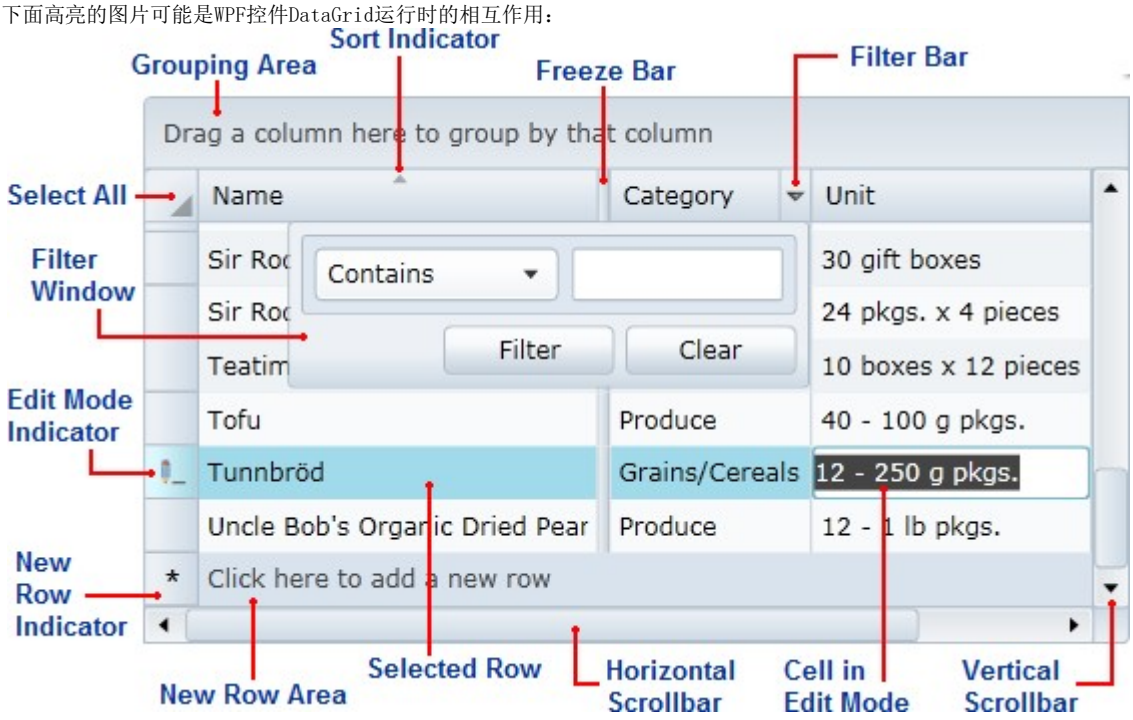
```
Visual Basic
Me.cldatagrid1.FontFamily = New FontFamily("Times New Roman")
Me.cldatagrid1.FontSize = 10
```

```
C#
this.cldatagrid1.FontFamily = new FontFamily("Times New Roman"); this.cldatagrid1.FontSize = 10;
```

你所完成的你所完成的运行应用程序，并观察到在网格中的所有行出现在新的罗马字体。

运行时间迭代

下面高亮的图片可能是WPF控件DataGrid运行时的相互作用：



以下主题详细介绍了这些运行时功能，包括筛选、排序和分组数据。

选择模式

你可以通过设置选择模型属性置网格的模式选择行为。 你可以改变用户如何与网格交互，但设置选择模型属性设置为为以下值之一：

属性属性	描述描述
None	用户不能选择任何项目。
SingleCell	用户只能选择一个单元格。
SingleRow	用户只能选择单行一次。
SingleColumn	用户只能在一个时间选择一个列。
SingleRange	用户只能在一个时间选择一个单元格区域。 （一个范围是由两个单元格分隔的矩形）
MultiRow (Default)	用户可以选择多行，同时按住相应的修改键。
MultiColumn	用户可以选择多个列，同时按住相应的修改键。
MultiRange	用户可以选择多个单元格的范围，同时按住相应的修改键。

关于修改键和多行选项的更多信息，见多行选择（第1.3.14.3章节）的话题。

自定义自动生成的列

您可以自定义列，即使是自动生成的列。 如果AutoGenerateColumns属性设置为true，列是自动生成的，你可以通过C1DataGrid.AutoGeneratingColumn事件处理自定义生成的列显示在代码中。添加AutoGeneratingColumn事件处理程序
完成以下步骤来添加AutoGeneratingColumn事件处理程序：

1. 切换到代码视图，为AutoGeneratingColumn事件添加事件处理程序，例如：

```
Visual Basic
```

```
Private Sub C1DataGrid1_AutoGeneratingColumn(ByVal sender As System.Object,
ByVal e As C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs) Handles
C1DataGrid1.AutoGeneratingColumn ' 在此处添加代码。
End Sub
```

```
C#
```

```
private void C1DataGrid1_AutoGeneratingColumn(object sender, C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs
e)
{
// 在此处添加代码。
}
```

2. 切换到源视图并添加C1DataGrid 控件的实例事件处理程序，例如：

<c1:C1DataGrid x:Name="c1DataGrid1" AutoGenerateColumns="True" AutoGeneratingColumn="
c1DataGrid1_AutoGeneratingColumn"></c1:C1DataGrid>
现在你可以将代码添加到AutoGeneratingColumn事件处理程序以自定义自动生成的列的外观和行为。 下面是自定义列格式和行为的例子。

取消列生成

你可以取消在AutoGeneratingColumn事件中产生的特定的列。 例如，你可以使用下面的代码来取消网格中的布尔列的生成：

```
Visual Basic
```

```
<c1:C1DataGrid x:Name="c1DataGrid1" AutoGenerateColumns="True"
AutoGeneratingColumn=" "
Private Sub C1DataGrid1_AutoGeneratingColumn(ByVal sender As System.Object,

ByVal e As C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs)
Handles C1DataGrid1.AutoGeneratingColumn
' 取消所有布尔列的自动生成。
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)
{
// 取消所有布尔列的自动生成。
if (e.Property.PropertyType == typeof(bool))
e.Cancel = true;
}
```

更改列标题

在AutoGeneratingColumn事件中你可以更改出现在标题自动生成的列的文本。
例如，你可以改变"产品名称"栏，以便它与"名称"标题通过使用下面的代码出现：

```
Visual Basic
```

```
Private Sub C1DataGrid1_AutoGeneratingColumn(ByVal sender As System.Object,

ByVal e As C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs)
Handles C1DataGrid1.AutoGeneratingColumn
' 修改"产品名称"列标头
If e.Column.Header.ToString() = "ProductName" Then
e.Header = "Name"
End If
End Sub
```

C#

```
private void c1DataGrid1_AutoGeneratingColumn(object sender,
C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs e)
{
// 修改"产品名称"列标头
if (e.Column.Header.ToString() == "ProductName") e.Column.Header = "Name";
}
```

阻止列之间的相互作用

使用AutoGeneratingColumn事件你可以改变最终用户如何与特定的生成的列的互动。 例如，您可以阻止用户从以下代码移动只读列：

Visual Basic

```
Private Sub C1DataGrid1_AutoGeneratingColumn(ByVal sender As System.Object,

ByVal e As C1.WPF.DataGrid.DataGridAutoGeneratingColumnEventArgs)
Handles C1DataGrid1.AutoGeneratingColumn
' 修改"产品名称"列标头
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)
{
// 修改"产品名称"列标头
if (e.Column.IsReadOnly == true)
e.Column.CanUserMove = false;
}
```

基于任务的帮助

以下任务的帮助主题假定您熟悉Visual Studio和Expression Blend和知道如何在一般中使用C1DataGrid控件。 如果你不熟悉WPF DataGrid和Silverlight的数据，请参阅快速启动的第一。
本节中的各主题提供了使用C1DataGrid 产品特定任务的解决方案。
大多数任务的帮助主题还假设您已经创建了一个新的WPF或者Silverlight项目并添加一个C1DataGrid控件项目－有关创建的控制信息，请参见创建一个DataGrid（第1.6.1章节）。

创建数据网格

你可以很容易地创建C1DataGrid控件在Expression Blend，在XAML和在代码的设计时。
请注意，如果您根据以下步骤创建了一个C1DataGrid控件，它会出现空。 您将需要绑定网格或用数据填充它。

在Blend的设计时

在Blend中创建一个C1DataGrid控件，完成以下步骤：

1. 导航到"项目"窗口中，右键单击"项目文件"列表中的"引用"文件夹。 在上下文菜单中选择"添加引用，找到并选择c1.wpf.datagrid.dll组件，然后单击"打开"。该对话框将关闭，并且将添加到您的项目中，并且该控件将在资产库中提供。
2. 在"工具箱"中，单击"资产"按钮（"双图标"图标）打开"资产"对话框。
3. 在资产库"对话框，在左窗格中选择控件项，然后单击右窗格中的C1DataGrid图标：C1DataGrid图标将在资产按钮底下的工具箱中出现。
4. 点击一次用户控件设计区域并选择它。 不像在Visual Studio中，Blend可以添加WPF控件直接设计表面作为下一步。
5. 双击工具箱中的C1DataGrid图标添加控件到面板上。 C1DataGrid控件在您的应用程序中将存在。

6. 如果您选择，可以通过在属性窗口中选择和设置属性来进行自定义的控制。
例如，设置c1datagrid控件的名称属性为"c1datagrid1"高度属性为"180"，和宽度属性"250"。

在XAML

使用XAML标记创建一个c1datagrid控件，完成以下步骤：

1. 在"可视化工作室解决方案资源管理器"中，右键单击"项目文件"列表中的"引用"文件夹。
在上下文菜单中选择"添加引用，选择c1.wpf.datagrid.dll组件，并单击"确定"。
2. 通过xmlns:c1="http://schemas.componentone.com/winfx/2006/xaml向添加项目中添加一个XAML命名空间并初始化<UserControl>
<Window>标签。 它将出现类似以下：

```
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.MainWindow" 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. 添加一个<c1:C1DataGrid>标签到你的项目中的<Grid>标签来创建一个C1DataGrid 控制。 标记将出现类似以下：

```
XAML
<Grid x:Name="LayoutRoot" Background="White">
<c1:C1DataGrid Name="c1datagrid1" Height="180" Width="250" /> </Grid>
```

这个标记将创建一个空的c1datagrid控件命名为"c1datagrid1"并设置控件的大小。

在代码中

在代码中创建一个c1datagrid控件，完成以下步骤：

4. 在"可视化工作室解决方案资源管理器"中，右键单击"项目文件"列表中的"引用"文件夹。
在上下文菜单中选择"添加引用，选择c1.wpf.dll和c1.wpf.datagrid.dll组件，并单击"确定"。
5. 右键单击mainpage.xamlmainwindow.xaml窗口内选择"查看代码，切换到代码视图
6. 向页面顶部添加下面的导入语句：

Visual Basic	
Imports C1.WPF	variable=WPF.DataGrid
C#	
using C1.WPF	variable=WPF.DataGrid;

将代码添加到页面的构造函数来创建c1datagrid控件。 它将看起来类似以下：

```
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);
}
```

此代码将创建一个空的c1datagrid控件命名为"c1datagrid1"，设置控件的大小，并在页面中添加控件。

你所完成的

运行应用程序并观察你已经创造了一个c1datagrid控件。
注意：当你创建一个c1datagrid控件在上述步骤，会出现空。 您可以将项目添加到可以在运行时进行交互的控件中。