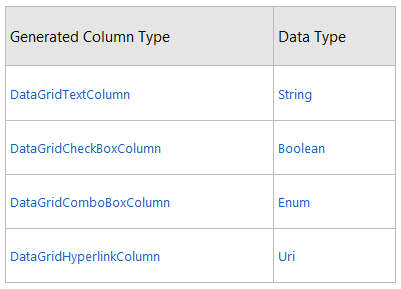
**WPF 4 DataGrid 控件（基本功能篇）**

　　提到DataGrid 不管是网页还是应用程序开发都会频繁使用。通过它我们可以灵活的在行与列间显示各种数据。本篇将详细介绍WPF 4 中DataGrid 的相关功能。

　　自定义列

　　默认情况下，当我们为DataGrid 控件设置ItemSource 属性后，DataGrid 会根据数据类型自动生成相应的列，下表列出DataGrid 支持的四种列及其数据类型。



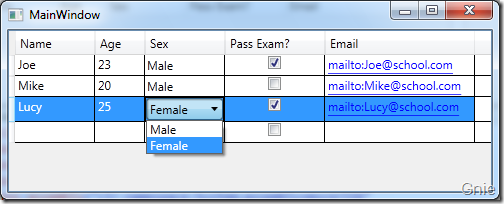
　　在创建DataGrid 时可以通过AutoGenerateColumns 属性设置列是否自动生成，从而加入自定义列。如果DataGrid 中同时包含“自动生成列”与“用户自定义列”，则首先创建“用户自定义列”。下面代码实例中分别创建这四种列：

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30 | <Window　x:Class="WPF4ControlTest.MainWindow"  　　　　xmlns="<http://schemas.microsoft.com/winfx/2006/xaml/presentation>"  　　　　xmlns:x="<http://schemas.microsoft.com/winfx/2006/xaml>"  　　　　xmlns:local="clr-namespace:WPF4ControlTest"  　　　　xmlns:assembly="clr-namespace:System;assembly=mscorlib"  　　　　Title="MainWindow"　Height="200"　Width="500">  　　<Window.Resources>  　　　　<ObjectDataProvider　x:Key="sexEnum"　MethodName="GetValues"  　　　　　　　　　　　　　　ObjectType="{x:Type　assembly:Enum}">  　　　　　　<ObjectDataProvider.MethodParameters>  　　　　　　　　<x:Type　Type="local:SexOpt"/>  　　　　　　</ObjectDataProvider.MethodParameters>  　　　　</ObjectDataProvider>  　　</Window.Resources>  　　<Grid>  　　　　<DataGrid　x:Name="dataGrid"　ItemsSource="{Binding}"　AutoGenerateColumns="False">  　　　　　　<DataGrid.Columns>  　　　　　　　　<DataGridTextColumn　Header="Name"　Width="80"　Binding="{Binding　Name}"/>  　　　　　　　　<DataGridTextColumn　Header="Age"　Width="50"　Binding="{Binding　Age}"/>  　　　　　　　　<DataGridComboBoxColumn　Width="80"　Header="Sex"  　　　　　　　　　　　　　　　　　　　　SelectedItemBinding="{Binding　Sex}"  　　　　　　　　　　　　　　　　　　　　ItemsSource="{Binding　Source={StaticResource　sexEnum}}"/>  　　　　　　　　<DataGridCheckBoxColumn　Header="Pass　Exam?"　Width="100"  　　　　　　　　　　　　　　　　　　　　Binding="{Binding　Pass}"/>  　　　　　　　　<DataGridHyperlinkColumn　Header="Email"　Width="150"  　　　　　　　　　　　　　　　　　　　　　Binding="{Binding　Email}"/>  　　　　　　</DataGrid.Columns>  　　　　</DataGrid>  　　</Grid>  </Window> |

　　其中每列都包含各自的数值类型，在C# 中创建Member 类及SexOpt 枚举，并将memberData 数据绑定到DataGrid：

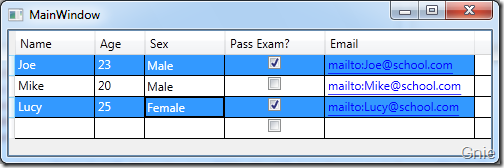
|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | using　System;  using　System.Windows;  using　System.Collections.ObjectModel;    namespace　WPF4ControlTest  {  　　///　<summary>  　　///　Interaction　logic　for　MainWindow.xaml  　　///　</summary>  　　public　partial　class　MainWindow　:　Window  　　{  　　　　public　MainWindow()  　　　　{  　　　　　　InitializeComponent();  　　　　　　ObservableCollection<Member>　memberData　=　new　ObservableCollection<Member>();  　　　　　　memberData.Add(new　Member()  　　　　　　{  　　　　　　　　Name　=　"Joe",　Age　=　"23",　Sex　=　SexOpt.Male,  　　　　　　　　Pass　=　true,　Email　=　new　Uri("mailto:Joe@school.com")  　　　　　　});  　　　　　　memberData.Add(new　Member()  　　　　　　{  　　　　　　　　Name　=　"Mike",　Age　=　"20",  　　　　　　　　Sex　=　SexOpt.Male,　Pass　=　false,  　　　　　　　　Email　=　new　Uri("mailto:Mike@school.com")  　　　　　　});  　　　　　　memberData.Add(new　Member()  　　　　　　{  　　　　　　　　Name　=　"Lucy",　Age　=　"25",  　　　　　　　　Sex　=　SexOpt.Female,　Pass　=　true,  　　　　　　　　Email　=　new　Uri("mailto:Lucy@school.com")  　　　　　　});  　　　　　　dataGrid.DataContext　=　memberData;  　　　　}  　　}    　　public　enum　SexOpt　{　Male,　Female　};    　　public　class　Member  　　{  　　　　public　string　Name　{　get;　set;　}  　　　　public　string　Age　{　get;　set;　}  　　　　public　SexOpt　Sex　{　get;　set;　}  　　　　public　bool　Pass　{　get;　set;　}  　　　　public　Uri　Email　{　get;　set;　}  　　} |

　这样我们就可以通过自定义列的方式创建出DataGrid：



　　选择模式

　　默认情况下，DataGrid 的选择模式为“全行选择”，并且可以同时选择多行（如下图所示），我们可以通过SelectionMode 和SelectionUnit 属性来修改DataGrid 的选择模式。



　　SelectionUnit：包含Cell、FullRow 和CellOrRowHeader 三种单元选择模式。

　　· Cell：选择单元格；

　　· FullRow：选择全行；

　　· CellOrRowHeader：可选择单元格，也可以通过点击行首选择全行；

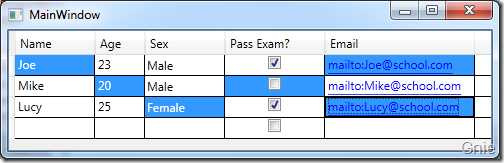
　　SelectionMode：分为Extended 和Single 两种模式。

　　· Extended：选择多个单元（单元格或全行，由SelectionUnit 定义）；

　　· Single：选择唯一单元（单元格或全行，由SelectionUnit 定义）；

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2 | <DataGrid　x:Name="dataGrid"　ItemsSource="{Binding}"　AutoGenerateColumns="False"  　　　　　SelectionUnit="Cell"　SelectionMode="Extended">…　… |

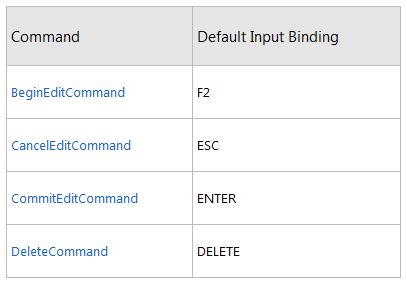
　　设置SelectionUnit 和SelectionMode 后的实例效果：



　　编辑

　　默认情况下，我们可以直接在DataGrid 中编辑数据（下表为相关的编辑命令），当然也可以通过IsReadOnly 属性将DataGrid 设置为只读方式。

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2 | <DataGrid　x:Name="dataGrid"　ItemsSource="{Binding}"　AutoGenerateColumns="False"  　　　　　SelectionUnit="Cell"　SelectionMode="Extended"　IsReadOnly="True">…　… |



# WPF 4 DataGrid 控件（自定义样式篇）

在

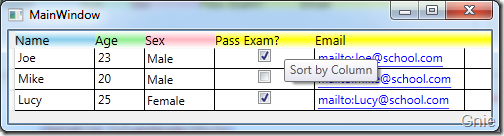
　　《WPF 4 DataGrid 控件（基本功能篇）》中我们已经学习了DataGrid 的基本功能及使用方法。本篇将继续介绍自定义DataGrid 样式的相关内容，其中将涉及到ColumnHeader、RowHeader、Row、Cell 的各种样式设置。

　　ColumnHeaderStyle 属性

　 　一般来讲列表头是用户首先注意的内容，那么如何在DataGrid 中设计一个美观的表头呢。我们既可以在<DataGrid>中通过<DataGrid.ColumnHeaderStyle>来 全局定义；也可以针对个别列在<DataGrid.Columns>中通 过<DataGridColumn.HeaderStyle>定义。

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75 | <DataGrid　x:Name="dataGrid"　ItemsSource="{Binding}"　AutoGenerateColumns="False"  　　　　　　SelectionUnit="CellOrRowHeader"　IsReadOnly="True">  　　<DataGrid.ColumnHeaderStyle>  　　　　<Style　TargetType="DataGridColumnHeader">  　　　　　　<Setter　Property="Background">  　　　　　　　　<Setter.Value>  　　　　　　　　　　<LinearGradientBrush　StartPoint="0,0"　EndPoint="0,1">  　　　　　　　　　　　　<GradientStop　Color="White"　Offset="0"/>  　　　　　　　　　　　　<GradientStop　Color="Yellow"　Offset="0.5"/>  　　　　　　　　　　　　<GradientStop　Color="White"　Offset="1"/>  　　　　　　　　　　</LinearGradientBrush>  　　　　　　　　</Setter.Value>  　　　　　　</Setter>  　　　　　　<Setter　Property="Foreground"　Value="Black"/>  　　　　　　<Setter　Property="FontSize"　Value="13"　/>  　　　　</Style>  　　</DataGrid.ColumnHeaderStyle>    　　<DataGrid.Columns>  　　　　<DataGridTextColumn　Header="Name"　Binding="{Binding　Name}">  　　　　　　<DataGridColumn.HeaderStyle>  　　　　　　　　<Style　TargetType="DataGridColumnHeader">  　　　　　　　　　　<Setter　Property="Background">  　　　　　　　　　　　　<Setter.Value>  　　　　　　　　　　　　　　<LinearGradientBrush　StartPoint="0,0"　EndPoint="0,1">  　　　　　　　　　　　　　　　　<GradientStop　Color="White"　Offset="0"/>  　　　　　　　　　　　　　　　　<GradientStop　Color="SkyBlue"　Offset="0.5"/>  　　　　　　　　　　　　　　　　<GradientStop　Color="White"　Offset="1"/>  　　　　　　　　　　　　　　</LinearGradientBrush>  　　　　　　　　　　　　</Setter.Value>  　　　　　　　　　　</Setter>  　　　　　　　　　　<Setter　Property="Foreground"　Value="Black"/>  　　　　　　　　　　<Setter　Property="FontSize"　Value="13"/>  　　　　　　　　　　<Setter　Property="Width"　Value="80"/>  　　　　　　　　　　<Style.Triggers>  　　　　　　　　　　　　<Trigger　Property="IsMouseOver"　Value="True">  　　　　　　　　　　　　　　<Setter　Property="ToolTip"　Value="Sort　by　Column"/>  　　　　　　　　　　　　</Trigger>  　　　　　　　　　　</Style.Triggers>  　　　　　　　　</Style>  　　　　　　</DataGridColumn.HeaderStyle>  　　　　</DataGridTextColumn>    　　　　<DataGridTextColumn　Header="Age"　Binding="{Binding　Age}">  　　　　　　<DataGridColumn.HeaderStyle>  　　　　　　　　<Style　TargetType="DataGridColumnHeader">  　　　　　　　　　　<Setter　Property="Background">  　　　　　　　　　　　　<Setter.Value>  　　　　　　　　　　　　　　<LinearGradientBrush　StartPoint="0,0"　EndPoint="0,1">  　　　　　　　　　　　　　　　　<GradientStop　Color="White"　Offset="0"/>  　　　　　　　　　　　　　　　　<GradientStop　Color="LightGreen"　Offset="0.5"/>  　　　　　　　　　　　　　　　　<GradientStop　Color="White"　Offset="1"/>  　　　　　　　　　　　　　　</LinearGradientBrush>  　　　　　　　　　　　　</Setter.Value>  　　　　　　　　　　</Setter>  　　　　　　　　　　<Setter　Property="Foreground"　Value="Black"/>  　　　　　　　　　　<Setter　Property="FontSize"　Value="13"/>  　　　　　　　　　　<Setter　Property="Width"　Value="50"/>  　　　　　　　　</Style>  　　　　　　</DataGridColumn.HeaderStyle>  　　　　</DataGridTextColumn>    　　　　<DataGridComboBoxColumn　Header="Sex"  　　　　　　　　　　　　　　　　SelectedItemBinding="{Binding　Sex}"  　　　　　　　　　　　　　　　　ItemsSource="{Binding　Source={StaticResource　sexEnum}}">  　　　　　　<DataGridColumn.HeaderStyle>  　　　　　　　　<Style　TargetType="DataGridColumnHeader">  　　　　　　　　　　<Setter　Property="Background">  　　　　　　　　　　　　<Setter.Value>  　　　　　　　　　　　　　　<LinearGradientBrush　StartPoint="0,0"　EndPoint="0,1">  　　　　　　　　　　　　　　　　<GradientStop　Color="White"　Offset="0"/>  　　　　　　　　　　　　　　　　<GradientStop　Color="LightPink"　Offset="0.5"/>  　　　　　　　　　　　　　　　　<GradientStop　Color="White"　Offset="1"/>  　　　　　　　　　　　　　　</LinearGradientBrush>  　　　　　　　　　　　　</Setter.Value>  　　　　　　　　　　</Setter>  　　　　　　　　　　<Setter　Property="Foreground"　Value="Black"/>  　　　　　　　　　　<Setter　Property="FontSize"　Value="13"/>  　　　　　　　　　　<Setter　Property="Width"　Value="70"/>  　　　　　　　　</Style>  　　　　　　</DataGridColumn.HeaderStyle>  　　　　</DataGridComboBoxColumn>    　　　　<DataGridCheckBoxColumn　Header="Pass　Exam?"　Width="100"  　　　　　　　　　　　　　　　　Binding="{Binding　Pass}"/>    　　　　<DataGridHyperlinkColumn　Header="Email"　Width="150"  　　　　　　　　　　　　　　　　　Binding="{Binding　Email}"/>  　　</DataGrid.Columns>  </DataGrid> |

　其效果如下图所示，Name、Age、Sex 列会按各自的样式显示。未定义样式的Pass Exam 和Email 列将按<DataGrid.ColumnHeaderStyle>全局定义显示，并且当鼠标移置到这两列上方时会有“Sort by Column”提示信息。

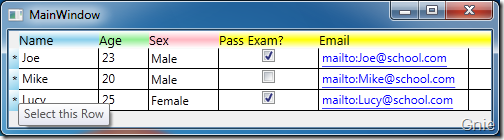


　　RowHeaderStyle 属性

　　列表头介绍完了下面再来看看行表头，其定义方式与列表头相同，直接在<DataGrid>中加入<DataGrid.RowHeaderStyle>进行全局定义，见如下代码：

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | ...　...  <DataGrid.RowHeaderStyle>  　　<Style　TargetType="DataGridRowHeader">  　　　　<Setter　Property="Content"　Value="　·¤"/>  　　　　<Setter　Property="Width"　Value="10"/>  　　　　<Setter　Property="Background">  　　　　　　<Setter.Value>  　　　　　　　　<LinearGradientBrush　StartPoint="0,0"　EndPoint="1,1">  　　　　　　　　　　<GradientStop　Color="White"　Offset="0"/>  　　　　　　　　　　<GradientStop　Color="SkyBlue"　Offset="1"/>  　　　　　　　　</LinearGradientBrush>  　　　　　　</Setter.Value>  　　　　</Setter>  　　　　<Style.Triggers>  　　　　　　<Trigger　Property="IsMouseOver"　Value="True">  　　　　　　　　<Setter　Property="ToolTip"　Value="Select　this　Row"/>  　　　　　　</Trigger>  　　　　</Style.Triggers>  　　</Style>  </DataGrid.RowHeaderStyle>  ...　... |

　　效果图：

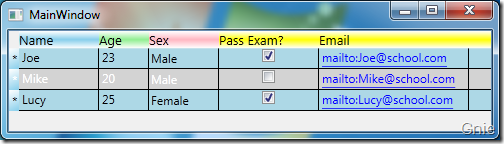


　　RowStyle 属性

　　接下来我们来为行设置样式，使每行的颜色变为蓝色，并且当鼠标移至行上方时，其背景颜色会变为灰色，文字颜色变为白色。

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | ...　...  <DataGrid.RowStyle>  　　<Style　TargetType="DataGridRow">  　　　　<Setter　Property="Background"　Value="LightBlue"　/>  　　　　<Style.Triggers>  　　　　　　<Trigger　Property="IsMouseOver"　Value="True">  　　　　　　　　<Setter　Property="Background"　Value="LightGray"/>  　　　　　　　　<Setter　Property="Foreground"　Value="White"/>  　　　　　　</Trigger>  　　　　</Style.Triggers>  　　</Style>  </DataGrid.RowStyle>  ...　... |

　　效果图：

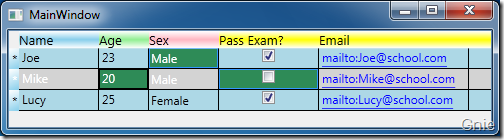


　　CellStyle 属性

　　单元格部分我们继续制作一个动态样式，当鼠标点击单元格后，其背景色将变为绿色。

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11 | ...　...  <DataGrid.CellStyle>  　　<Style　TargetType="DataGridCell">  　　　　<Style.Triggers>  　　　　　　<Trigger　Property="IsSelected"　Value="True">  　　　　　　　　<Setter　Property="Background"　Value="SeaGreen"/>  　　　　　　</Trigger>  　　　　</Style.Triggers>  　　</Style>  </DataGrid.CellStyle>  ...　... |

　　效果图：

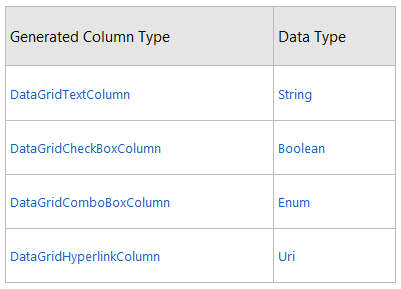


# WPF 4 DataGrid 控件（进阶篇一）

上一篇《WPF 4 DataGrid 控件（自定义样式篇）》中，我们掌握了DataGrid 列表头、行表头、行、单元格相关的自定义样式方法，本篇我们继续对自定义样式进行一些高级设置。

　　DataGridTemplateColumn 类

　　除了下表所示的列类型外，我们还可以通过DataGridTemplateColumn 自定义列样式，为DataGrid 添加更完美的数据显示方式。



　　首先我们在类中添加ExamDate 用来在DataGrid 中显示学生考试日期，并为相应学生赋值。

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | public　class　Member  {  　　public　string　Name　{　get;　set;　}  　　public　string　Age　{　get;　set;　}  　　public　SexOpt　Sex　{　get;　set;　}  　　public　bool　Pass　{　get;　set;　}  　　public　DateTime　ExamDate　{　get;　set;　}  　　public　Uri　Email　{　get;　set;　}  }    …　…  ObservableCollection<Member>　memberData　=　new　ObservableCollection<Member>();  …　…  memberData.Add(new　Member()  {  　　Name　=　"Lucy",　Age　=　"25",  　　Sex　=　SexOpt.Female,　Pass　=　true,  　　ExamDate　=　new　DateTime(2010,　4,　10),  　　Email　=　new　Uri("mailto:Lucy@school.com")  });  dataGrid.DataContext　=　memberData;  …　… |

　　接下来要在<Window.Resources> 中定义两种DataTemplate 样式模板（如下代码），第一个用来设置日期列的显示方式：

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11  12 | <DataTemplate　x:Key="DateTemplate"　>  　　<StackPanel　Width="40"　Height="30">  　　　　<Border　Background="Orange"　BorderBrush="Black"　BorderThickness="1">  　　　　　　<TextBlock　Text="{Binding　ExamDate,　StringFormat={}{0:MM-dd}}"  　　　　　　　　　　　　FontSize="10"　HorizontalAlignment="Center"/>  　　　　</Border>  　　　　<Border　Background="White"　BorderBrush="Black"　BorderThickness="1">  　　　　　　<TextBlock　Text="{Binding　ExamDate,　StringFormat={}{0:yyyy}}"  　　　　　　　　　　　　FontSize="10"　HorizontalAlignment="Center"/>  　　　　</Border>  　　</StackPanel>  </DataTemplate> |

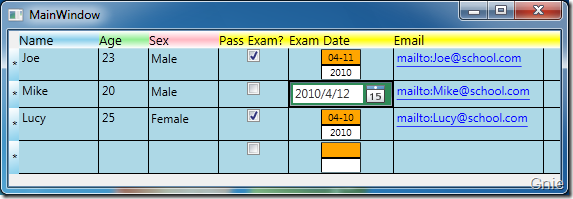
　　第二个用来设置日期列的编辑方式，这里将用到DataPicker：

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3 | <DataTemplate　x:Key="EditingDateTemplate">  　　<DatePicker　SelectedDate="{Binding　ExamDate}"/>  </DataTemplate> |

　　模板设置完成后，就要在<DataGrid>中加入DataGridTemplateColumn 列显示学生的考试日期。

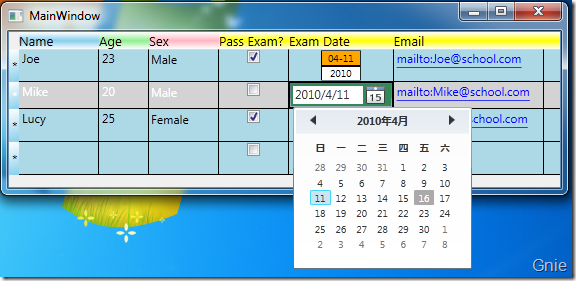
|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5 | ...　...  <DataGridTemplateColumn　Header="Exam　Date"  　　　　　　　　　　　　CellTemplate="{StaticResource　DateTemplate}"  　　　　　　　　　　　　CellEditingTemplate="{StaticResource　EditingDateTemplate}"/>  ...　... |

　　在XAML 代码中设置CellTemplate 和CellEditingTemplate 属性后便可运行程序测试一下效果。



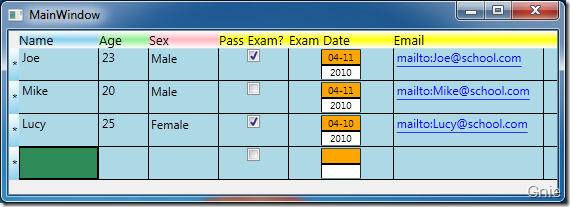
[查看原图（大图）](http://img.ddvip.com/2010_04_20/1271742757_ddvip_8824.png)

　　如上图所示，若对Exam Date 列进行修改时，其显示方式将变为DataPicker，用户可以方便的选择相应的日期进行修改。



[查看原图（大图）](http://img.ddvip.com/2010_04_20/1271742759_ddvip_6085.png)

　　修改后恢复原状：



# WPF 4 DataGrid 控件（进阶篇二）

上一篇《WPF 4 DataGrid 控件（进阶篇一）》中我们通过DataGridTemplateColumn 类自定义编辑了日期列的样式，当然也可以根据个人需要设置任何样式模板。上例中Pass Exam 列显示学生是否通过考试，但我们并不知道该学生每门学科的成绩是多少。本篇将为DataGrid 行增加这些详细信息，使得DataGrid 数据更加充实。

　　首先，我们仍然先更新一下Member 类，增加Math 和History 两门学科：

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11 | public　class　Member  {  　　public　string　Name　{　get;　set;　}  　　public　string　Age　{　get;　set;　}  　　public　SexOpt　Sex　{　get;　set;　}  　　public　bool　Pass　{　get;　set;　}  　　public　DateTime　ExamDate　{　get;　set;　}  　　public　Uri　Email　{　get;　set;　}  　　public　int　Math　{　get;　set;　}  　　public　int　History　{　get;　set;　}  } |

　　为学生赋上考试成绩：

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | …　…  memberData.Add(new　Member()  {  　　Name　=　"Lucy",  　　Age　=　"25",  　　Sex　=　SexOpt.Female,  　　Pass　=　true,  　　ExamDate　=　new　DateTime(2010,　4,　10),  　　Email　=　new　Uri("mailto:Lucy@school.com"),  　　Math　=　80,  　　History　=　85  });  dataGrid.DataContext　=　memberData; |

　　接下来就要到XAML 中为考试成绩设计样式模板：

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | <Window.Resources>  　　...　...  　　<DataTemplate　x:Key="RowDetails">  　　　　<Border　BorderThickness="0"　Background="Orchid"　Padding="10">  　　　　　　<StackPanel　Orientation="Vertical">  　　　　　　　　<StackPanel　Orientation="Horizontal">  　　　　　　　　　　<TextBlock　Text="Math:　"　VerticalAlignment="Center"/>  　　　　　　　　　　<TextBlock　Text="{Binding　Math}"　VerticalAlignment="Center"  　　　　　　　　　　　　　　　　FontSize="15"　FontWeight="Bold"/>  　　　　　　　　</StackPanel>  　　　　　　　　<StackPanel　Orientation="Horizontal">  　　　　　　　　　　<TextBlock　Text="History:　"　VerticalAlignment="Center"/>  　　　　　　　　　　<TextBlock　Text="{Binding　History}"　VerticalAlignment="Center"  　　　　　　　　　　　　　　　　FontSize="15"　FontWeight="Bold"/>  　　　　　　　　</StackPanel>  　　　　　　</StackPanel>  　　　　</Border>  　　</DataTemplate>  </Window.Resources> |

　　在<DataGrid>中为RowDetailsTemplate 属性添加RowDetails 模板:

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4 | <DataGrid　x:Name="dataGrid"　ItemsSource="{Binding}"  　　　　　AutoGenerateColumns="False"　SelectionUnit="CellOrRowHeader"  　　　　　RowDetailsTemplate="{StaticResource　RowDetails}">  …　… |

　 　当然，我们也可以直接在<DataGrid>中添加<DataGrid.RowDetailsTemplate> 完成上面所有XAML 代码。注：上面代码中<DataGrid>的RowDetailsTemplate 属性要清除。

|  |  |
| --- | --- |
| 双击代码全选 | |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | ...　...  <DataGrid.RowDetailsTemplate>  　　<DataTemplate>  　　　　<Border　BorderThickness="0"　Background="Orchid"　Padding="10">  　　　　　　<StackPanel　Orientation="Vertical">  　　　　　　　　<StackPanel　Orientation="Horizontal">  　　　　　　　　　　<TextBlock　Text="Math:　"　VerticalAlignment="Center"/>  　　　　　　　　　　<TextBlock　Text="{Binding　Math}"　VerticalAlignment="Center"  　　　　　　　　　　FontSize="15"　FontWeight="Bold"/>  　　　　　　　　</StackPanel>  　　　　　　　　<StackPanel　Orientation="Horizontal">  　　　　　　　　　　<TextBlock　Text="History:　"　VerticalAlignment="Center"/>  　　　　　　　　　　<TextBlock　Text="{Binding　History}"　VerticalAlignment="Center"  　　　　　　　　　　FontSize="15"　FontWeight="Bold"/>  　　　　　　　　</StackPanel>  　　　　　　</StackPanel>  　　　　</Border>  　　</DataTemplate>  </DataGrid.RowDetailsTemplate>  ...　... |

　　编译程序，点击行头显示详细考试成绩数据。完成这几篇开发后，我们的DataGrid 内容是不是充实了很多。

