

如何使用样式

WinForms版版True DBGrid 使用类似于Microsoft Word 和 Excel的样式模型，用以简化自定义的网格外观，一个Style 对象时对字体，颜色，图片以及格式信息的连接，包含如下的属性：

Property	Description
Alpha	获取或设置当样式被呈现时的透明度控件。
BackColor	获取或设置与一个Style相关联的背景颜色。
BackColor2	获取或设置与一个Style相关联的背景颜色。
BackgroundImage	获取或设置与一个Style相关联的背景图像。
BackgroundPictureDrawMode	获取或设置一个BackgroundImage的绘制方法。
Borders	获取与该样式相关联的GridBorders。
Font	获取或设置与一个Style相关联的字体。
ForeColor	获取或设置与一个Style相关联的前景颜色。
ForegroundImage	获取或设置与一个Style相关联的前景图像。
ForeGroundPicturePosition	获取或设置绘制的ForegroundImage位置。
GammaCorrection	获取或设置一个值，该值决定在线性渐变的样式呈现时是否启用伽玛校正。
GradientMode	指定线性渐变的方向。
HorizontalAlignment	获取或设置文本的水平对齐。
Locked	获取或设置一个值，该值决定在相关联的对象中能否被输入。
Name	获取或设置Style的名称。
Padding	获取或设置单元格中内容和它的边缘之间的空间。
Trimming	获取或设置修剪字符串，当它不完全适合布局形状。
VerticalAlignment	获取或设置文本的垂直对齐。
WrapText	获取或设置一个值，该值决定是否文本被换行，当它不完全适合布局形状。

内置命名样式

当网格先被创建时，它有一个用以显示的内置命名样式，例如，Heading样式决定了用以显示列头的属性，通过使用C1TrueDBGrid Styles Editor更改内置命名样式，在设计时统一更改网格的外观，在运行时，GridStyleCollection 提供了访问相同命名样式的集合，初始时所有网格包含十个内置样式，控件内包含以下用以显示的网格元素：

Element	Description
Caption	网格或者拆分的标题栏。
Editor	网格内的单元格编辑器。
EvenRow	在偶数行中的数据单元格。
Filter Bar	在过滤栏列的数据单元格。
Footer	列注脚。
Group	在网格分组区域中的分组列。
Heading	列标题。
HighlightRow	在高亮行中的数据单元格。
Inactive	当其他列被聚焦时列标题。
Normal	在非选择，非高亮行的数据单元格。

OddRow	在奇数行中的数据单元格。
Record Selector	在记录选择列的数据。
Selected	在选择行中的数据单元格。

在标签中选定的行可以被添加到SelectedRowCollection中，也可以在代码中或通过用户交互。当MarqueeStyle 属性被设置为MarqueeEnum.HighlightRow 或 MarqueeEnum.HighlightRowRaiseCell时，highlighted 标记的行引用当前行。当AlternatingRows 属性被设置为True时，偶数行和奇数行样式可以被使用。

默认命名样式

如Microsoft Word，True DBGrid 中的一个Style 对象可以继承另一个样式中的字符，我们称之为父样式，对于新创建的网格，常规的样式是所有命名样式的父（或祖）样式，它的默认属性如下：

Property	Setting
Alpha	255
BackColor	System.Drawing.Color.White
BackColor2	System.Drawing.Color.White
BackgroundImage	None
BackgroundPictureDrawMode	BackgroundPictureDrawModeEnum.Stretch
Font	Microsoft Sans Serif, 8.25pt
ForeColor	System.Drawing.Color.Black
ForegroundImage	None
ForegroundPicturePosition	ForegroundPicturePositionEnum.LeftOfText
GammaCorrection	False
GradientMode	None
HorizontalAlignment	AlignHorzEnum.General
Locked	False
Padding	0, 0, 0, 0
Trimming	Character
VerticalAlignment	AlignVertEnum.Top
WrapText	False

头与脚的样式的定义很相似，每一个常规样式的继承和每一个覆盖如下述属性：

Property	Setting
BackColor	System.Drawing.SystemColors.Control
ForeColor	System.Drawing.Color.Black
VerticalAlignment	AlignVertEnum.Center

头样式覆盖一个额外的属性，而脚样式无此属性：

Property	Setting
WrapText	True

选择的样式也可以从常规和覆盖中继承两个颜色属性：

Property	Setting
BackColor	System.Drawing.SystemColors.Highlight

ForeColor	System.Drawing.SystemColors.HighlightText
-----------	---

与HighlightRow样式的True值一样，使用颜色的逆置设置默认的Normal样式：

Property	Setting
BackColor	System.Drawing.SystemColors.Text
ForeColor	System.Drawing.SystemColors.HighlightText

偶数行，奇数行及过滤栏样式均继承自Normal，但仅有EvenRow样式覆盖了以下属性：

Property	Setting
BackColor	System.Drawing.Color.Aqua

唯一不能直接继承Normal样式的是Caption和RecordSelector样式， 原因是网格和拆分的标题默认是居中的，且Caption样式指定以下属性：

Property	Setting
HorizontalAlignment	AlignHorzEnum.Center

命名样式继承

为了参阅命名样式如何继承，需要将一个网格置于窗体内并设置网格的Caption属性和它的默认列，设置默认列的FooterText 属性以及网格的ColumnFooters 属性为True，网格外观将会如下：

Caption	
Column 0	Column 1
*	
Footer 0	Footer 1

在ClTrueDBGrid 样式编辑器样式编辑器中，在左侧面板中选择Normal 并扩展Font 节点，设置Bold 属性为True，注意到列头，列脚和网格标题都是加粗的，因而Normal样式或它的继承都内置样式。

Caption	
Column 0	Column 1
*	
Footer 0	Footer 1

接下来，从左侧面板中选择Heading，并在右侧面板中选择ForeColor 属性，点击Web 选项卡并选择Navy，注意到列头和网格标题栏的文本颜色现在都是白色，因此Caption样式从Heading样式中继承了它的颜色属性，列脚将保持相同的样式因为Footer样式继承了Normal 样式，而不是Heading样式。

Caption	
Column 0	Column 1
*	
Footer 0	Footer 1

最后，从左侧面板中选择Caption 并在右侧的面板中选择它的BackColor 属性，点击Web 选项卡，此时选择AliceBlue，注意到列头背景颜色没有改变， 并且Caption样式将从它的父样式Heading中继承他的文本颜色。

Caption	
Column 0	Column 1
*	
Footer 0	Footer 1

更改命名样式

您可以使用.NET的集合编辑器更改GridStyleCollection，以在运行时更改整个网格的外观，例如使列头中的标题文本居中，可以更改内置Heading 样式的HorizontalAlignment 属性为AlignHorzEnum.Center。

以下声明的代码结果：

To write code in Visual Basic

Visual Basic
Me.C1TrueDBGrid1.Styles("Heading").HorizontalAlignment = AlignHorzEnum.Center

To write code in C#

C#
this.c1TrueDBGrid1.Styles["Heading"].HorizontalAlignment = AlignHorzEnum.Center;

然而，它可以不必要使用C1TrueDBGrid 样式编辑器样式编辑器 或者在代码中控制GridStyleCollection 的成员命名，网格和它的控件对象给出了几个属性并返回了Style 对象，正如下一节所描述的，网格的外观可以直接通过这些对象进行微调，更多详细信息请参阅使用使用 C1TrueDBGrid样式编辑器样式编辑器 （Section 6.5）。

使用样式属性

正如在Microsoft Word中文本模板指定的单个文档图形的整体外观，the named members of the GridStyleCollection 对象的命名成员提供了C1TrueDBGrid 或 C1TrueDBDropDown的一个整体显示模板，然而为了定制单个Split 或C1DisplayColumn对象的外观，请更改适合的Style 对象属性：

Property	Description
CaptionStyle	控制一个对象的标题样式。
EditorStyle	控制一个对象的编辑样式。
EvenRowStyle	控制偶数行的行样式。
FilterBarStyle	控制过滤栏中的列样式。
FooterStyle	控制一个对象的脚样式。
HeadingStyle	控制一个对象的头样式。
HighlightRowStyle	控制选取框的样式，当被设置为Highlight Row时。
InactiveStyle	控制一个对象的交互头样式。
OddRowStyle	控制奇数行的行样式。
RecordSelectorStyle	控制一个对象的记录选择器样式。
SelectedStyle	控制一个对象中所选择的行/列的样式。
Style	控制一个对象的常规样式。

直接更改一个样式属性

定制一个网格控件的外观可以修改适当样式属性的一个或多个成员，例如，为了设置网格的标题文本加粗，需要更改Font 对象的关联属性CaptionStyle ，在设计时可以通过扩展位于属性窗口的CaptionStyle 树节点，Font 节点以及设置Bold属性为True，当您点击这个特殊的属性时，这些更改会应用于网格。

注意到当切换到C1TrueDBGrid 样式编辑器样式编辑器时，可以看到内置的Caption样式没有改变。

这就意味着以下声明并不等效：

To write code in Visual Basic

Visual Basic

```
Dim myfont As New Font(Me.C1TrueDBGrid1.Font, FontStyle.Bold)

Me.C1TrueDBGrid1.CaptionStyle.Font = myfont

Me.C1TrueDBGrid1.Styles("Caption").Font = myfont
```

To write code in C#

C#

```
Font myfont = new Font(this.c1TrueDBGrid1.Font, FontStyle.Bold); this.c1TrueDBGrid1.CaptionStyle.Font = myfont;
this.c1TrueDBGrid1.Styles["Caption"].Font = myfont;
```

第一个声明指定了网格标题栏的字体，因为这不是一个根样式，所以命名标题样式也会改变。

命名样式vs. 匿名样式

当再设计时设置了样式的属性，这对理解命名样式与匿名样式的区别至关重要。

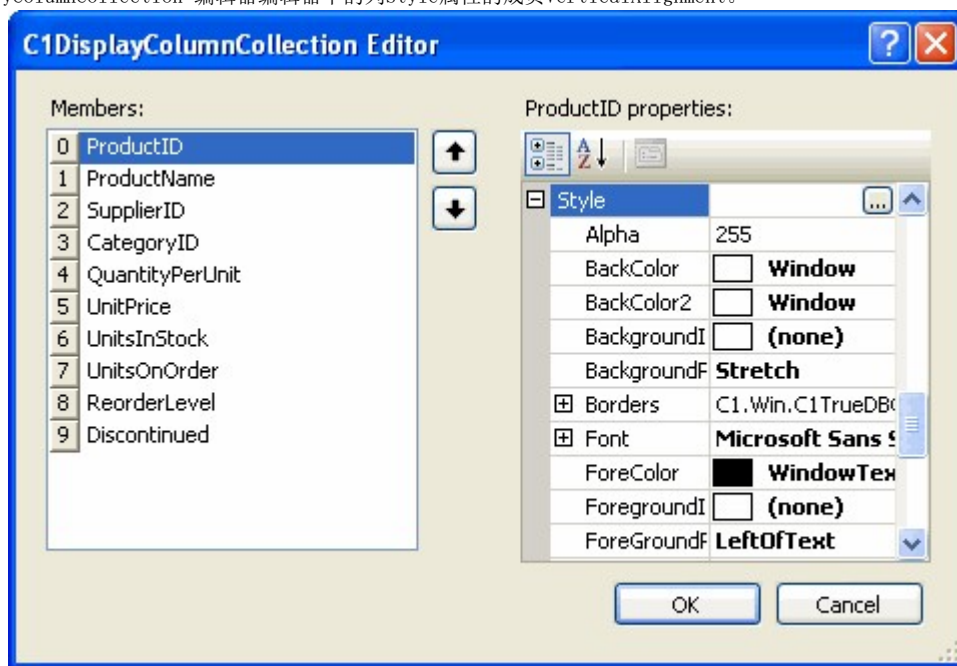
命名样式提供了模板用以管理网格，拆分以及它们的列的外观，再设计过程中，可以使用GridStyleCollection Editor创建，更改和删除命名样式，在运行时，GridStyleCollection 被用来表示命名Style对象的相同集合。

匿名样式不是GridStyleCollection的成员，然而匿名样式的提出可以使单个拆分或列可以轻松直接的定制，而不需要定义一个单独的命名样式。

以下类比可以帮助澄清命名与匿名样式之间的区别，一个Microsoft Word

文档由几个基于默认常规段落样式组成，假设一个段落需要缩进和斜体显示为一个引文，如果文档是工作的一部分，它包含几个引文，为目标定义一个特殊样式并应用它到所有引文的段落，如果文档过早的起草或它可能不能被更新，或者定义一个段落的样式，此时可以很方便的将缩进和协议应用于引文中。

在该类中，直接指定段落的属性类似于设置了属性的成员，并返回了匿名样式，例如，在一个特殊网格列来垂直居中数据，更改在C1DisplayColumnCollection 编辑器编辑器中的列Style属性的成员VerticalAlignment。



注意到更改一个匿名样式同更改一个命名样式相同，需要扩展属性树中的Style 对象节点，并选择且编辑它的一个或者多个成员属性。

匿名样式继承

正如一个命名样式可以从其他的匿名样式中继承字体，颜色及格式化字符，Split对象中的匿名样式可以

从C1TrueDBGrid 对应的部分继承。类似的，在C1DisplayColumn 对象中的匿名样式可以在Split对象的对应部分中继承，因为C1TrueDBDropDown 控件没有Splits集合，C1DisplayColumn 对象的匿名样式可以从控件自身中继承值。当一个网格先被创建，它的Style 属性将从内置的Normal样式中继承它的所有属性，用以控制所有数据单元格的外观，

Normal样式的任何改变将传递到所有拆分中，并返回每个拆分中列。然而，可以通过更改匿名Style属性成员来改变Split或C1DisplayColumn对象的数据单元格外观。

考虑以下网格布局，其使用包含两个相同拆分内置样式的默认值。

Caption			
Split 0		Split 1	
Column 0	Column 1	Column 0	Column 1
Data	Data	Data	Data
Data	Data	Data	Data
Footer 0	Footer 1	Footer 0	Footer 1

所有后续的示例都使用布局作为开始，为了澄清，示例使用代码阐述样式属性与网格显示之间的关系。然而您在设计时使用网格集合编辑器执行相同的操作。

10.2.3.1 示例1：从包含的拆分中继承

由于默认的内置样式值均有效，从包含的拆分中的继承列可以继承整个网格，因此该声明不仅会影响数据单元格，而且还有所有的标头，注脚以及标题栏，该声明有几个可见的影响，使用C1TrueDBGrid Style Editor直接改变Normal样式，然而内置的Normal样式没有被更改。

Caption			
Split 0		Split 1	
Column 0	Column 1	Column 0	Column 1
Data	Data	Data	Data
Data	Data	Data	Data
Footer 0	Footer 1	Footer 0	Footer 1

以下代码从包含的拆分中继承了值：
To write code in Visual Basic

```
Visual Basic

Dim myfont As Font
myfont = New Font (Me.C1TrueDBGrid1.Styles("Normal").Font, FontStyle.Bold)

Me.C1TrueDBGrid1.Styles("Normal").Font = myfont
```

To write code in C#

```
C#

Font myfont; myfont = new Font (this.c1TrueDBGrid1.Styles["Normal"].Font, FontStyle.Bold);
this.c1TrueDBGrid1.Styles["Normal"].Font = myfont;
```

示例2：仅影响第一个拆分的数据单元格

在本示例中，只影响第一个拆分的数据单元格，这是因为拆分标题，列头和列脚都从内置样式Caption，Heading和 Footing中分别继承字体。

Caption			
Split 0		Split 1	
Column 0	Column 1	Column 0	Column 1
Data	Data	Data	Data
Data	Data	Data	Data
Footer 0	Footer 1	Footer 0	Footer 1

以下代码仅影响第一个拆分的数据单元格：
To write code in Visual Basic

```
Visual Basic

Dim myfont As Font
myfont = New Font (Me.C1TrueDBGrid1.Splits(0).Style.Font, FontStyle.Bold)

Me.C1TrueDBGrid1.Splits(0).Style.Font = myfont
```

To write code in C#

```
C#

Font myfont; myfont = new Font (this.c1TrueDBGrid1.Splits[0].Style.Font, FontStyle.Bold);
this.c1TrueDBGrid1.Splits[0].Style.Font = myfont;
```

示例3：仅在第一个拆分中影响所有元素

该示例将第一个的粗体在第一个拆分的所有元素中呈现，此外Style 属性对CaptionStyle, HeadingStyle, 及FooterStyle 属性的设置也是必要的。

Caption			
Split 0		Split 1	
Column 0	Column 1	Column 0	Column 1
▶ Data	Data	▶ Data	Data
Data	Data	Data	Data
Footer 0	Footer 1	Footer 0	Footer 1

以下代码将仅在第一个拆分中影响所有元素：
To write code in Visual Basic

```
Visual Basic

Dim myfont As Font
Dim myfont1 As Font
Dim myfont2 As Font
Dim myfont3 As Font
myfont = New Font (Me.C1TrueDBGrid1.Splits(0).Style.Font, FontStyle.Bold) Me.C1TrueDBGrid1.Splits(0).Style.Font = myfont

myfont1 = New Font (Me.C1TrueDBGrid1.Splits(0).CaptionStyle.Font, FontStyle.Bold)
Me.C1TrueDBGrid1.Splits(0).CaptionStyle.Font = myfont1
myfont2 = New Font (Me.C1TrueDBGrid1.Splits(0).HeadingStyle.Font, FontStyle.Bold)
Me.C1TrueDBGrid1.Splits(0).HeadingStyle.Font = myfont2
myfont3 = New Font (Me.C1TrueDBGrid1.Splits(0).FooterStyle.Font, FontStyle.Bold)
Me.C1TrueDBGrid1.Splits(0).FooterStyle.Font = myfont3
```

To write code in C#

```
C#

Font myfont;
Font myfont1;
Font myfont2;
Font myfont3; myfont = new Font (this.c1TrueDBGrid1.Splits[0].Style.Font, FontStyle.Bold);
this.c1TrueDBGrid1.Splits[0].Style.Font = myfont; myfont1 = new Font (this.c1TrueDBGrid1.Splits[0].CaptionStyle.Font,
FontStyle.Bold); this.c1TrueDBGrid1.Splits[0].CaptionStyle.Font = myfont1; myfont2 = new Font
(this.c1TrueDBGrid1.Splits[0].HeadingStyle.Font, FontStyle.Bold); this.c1TrueDBGrid1.Splits[0].HeadingStyle.Font =
myfont2; myfont3 = new Font (this.c1TrueDBGrid1.Splits[0].FooterStyle.Font, FontStyle.Bold);
this.c1TrueDBGrid1.Splits[0].FooterStyle.Font = myfont3;
```

示例4：仅影响第一个拆分的第一个列中的数据单元格

本示例中，只有第一个拆分的第一个列中的数据单元格被影响，这是因为列头和列脚从内置Heading和Footing样式分别继承了字体。

Caption			
Split 0		Split 1	
Column 0	Column 1	Column 0	Column 1
▶ Data	Data	▶ Data	Data
Data	Data	Data	Data
Footer 0	Footer 1	Footer 0	Footer 1

以下代码将仅影响第一个拆分的第一个列中的数据单元格：
To write code in Visual Basic

```
Visual Basic

Dim myfont As Font
myfont = New Font (Me.C1TrueDBGrid1.Splits(0).DisplayColumns(0).Style.Font, FontStyle.Bold)

Me.C1TrueDBGrid1.Splits(0).DisplayColumns(0).Style.Font = myfont
```

To write code in C#

```
C#
```

```
Font myfont;
myfont = new Font (this.clTrueDBGrid1.Splits[0].DisplayColumns[0].Style.Font, FontStyle.Bold);
this.clTrueDBGrid1.Splits[0].DisplayColumns[0].Style.Font = myfont;
```

示例5：仅在第一个拆分的第一列影响所有元素

本示例将第一个拆分的第一列中的所有元素设置为粗体，此外Style 属性对HeadingStyle， 及FooterStyle 属性的设置也是必要的。

Caption			
Split 0		Split 1	
Column 0	Column 1	Column 0	Column 1
► Data	Data	► Data	Data
Data	Data	Data	Data
Footer 0	Footer 1	Footer 0	Footer 1

以下代码仅在第一个拆分的第一列影响所有元素：
To write code in Visual Basic

```
Visual Basic

Dim myfont As Font
Dim myfont1 As Font
Dim myfont2 As Font
myfont = New Font (Me.ClTrueDBGrid1.Splits(0).DisplayColumns(0).Style.Font, FontStyle.Bold)
Me.ClTrueDBGrid1.Splits(0).DisplayColumns(0).Style.Font = myfont
myfont1 = New Font (Me.ClTrueDBGrid1.Splits(0).DisplayColumns(0).HeadingStyle.Font, FontStyle.Bold)

Me.ClTrueDBGrid1.Splits(0).DisplayColumns(0).HeadingStyle.Font = myfont1
myfont2 = New Font (Me.ClTrueDBGrid1.Splits(0).DisplayColumns(0).FooterStyle.Font, FontStyle.Bold)
Me.ClTrueDBGrid1.Splits(0).DisplayColumns(0).FooterStyle.Font = myfont2
```

To write code in C#

```
C#

Font myfont;
Font myfont1;
Font myfont2;
myfont = new Font (this.clTrueDBGrid1.Splits[0].DisplayColumns[0].Style.Font, FontStyle.Bold);
this.clTrueDBGrid1.Splits[0].DisplayColumns[0].Style.Font = myfont;

myfont1 = new Font (this.clTrueDBGrid1.Splits[0].DisplayColumns[0].HeadingStyle.Font, FontStyle.Bold);
this.clTrueDBGrid1.Splits[0].DisplayColumns[0].HeadingStyle.Font = myfont1;
myfont2 = new Font (this.clTrueDBGrid1.Splits[0].DisplayColumns[0].FooterStyle.Font, FontStyle.Bold);
this.clTrueDBGrid1.Splits[0].DisplayColumns[0].FooterStyle.Font = myfont2;
```

示例6：更改样式属性的背景色

在第一个示例中，可以通过设置网格的Style属性的Font 成员从而影响整个网格，包括每个标题栏，列头及列脚，然而同样是没有BackColor和ForeColor属性为真，由于内置的Caption， Heading和Footting 样式均覆盖可这些属性，仅有网格的数据单元格显示为浅紫色背景。

Caption			
Split 0		Split 1	
Column 0	Column 1	Column 0	Column 1
► Data	Data	► Data	Data
Data	Data	Data	Data
Footer 0	Footer 1	Footer 0	Footer 1

以下代码将更改Style属性的字体成员：
To write code in Visual Basic

```
Visual Basic

Me.ClTrueDBGrid1.Style.BackColor = System.Drawing.Color.Lavender
```

To write code in C#

```
C#
```

```
this.clTrueDBGrid1.Style.BackColor = System.Drawing.Color.Lavender;
```

示例7：更改第一个拆分的数据单元格

在本示例中，只有第一个拆分中的数据单元格被影响，这时因为拆分标题，列头与列脚分别从内置样式 `Caption`，`Heading`，及`Footring`继承背景颜色。

Caption			
Split 0		Split 1	
Column 0	Column 1	Column 0	Column 1
Data	Data	Data	Data
Data	Data	Data	Data
Footer 0	Footer 1	Footer 0	Footer 1

以下代码将值更改第一个拆分中的数据单元格：

To write code in Visual Basic

Visual Basic

```
Me.ClTrueDBGrid1.Splits(0).Style.BackColor = System.Drawing.Color.Lavender
```

To write code in C#

C#

```
this.clTrueDBGrid1.Splits[0].Style.BackColor = System.Drawing.Color.Lavender;
```

示例8：仅更改第一个拆分的第一列中的数据单元格

在本示例中，仅有第一个拆分中的第一列中的数据可以被影响，这时因为列头和列脚从内置样式`Heading` 和 `Footring`中分别继承它们的背景颜色。

Caption			
Split 0		Split 1	
Column 0	Column 1	Column 0	Column 1
Data	Data	Data	Data
Data	Data	Data	Data
Footer 0	Footer 1	Footer 0	Footer 1

以下代码会对第一个拆分中第一列数据单元格进行更改：

To write code in Visual Basic

Visual Basic

```
Me.ClTrueDBGrid1.Splits(0).DisplayColumns(0).Style.BackColor =  
System.Drawing.Color.Lavender
```

To write code in C#

C#

```
this.clTrueDBGrid1.Splits[0].DisplayColumnsp[0].Style.BackColor =  
System.Drawing.Color.Lavender;
```

示例9：设置C1DisplayColumn对象的对齐

`C1DisplayColumn` 对象的`HorizontalAlignment` 属性设置不仅可以影响数据单元格，而且可以影响它的列头和列脚，其原因在于内置样式`Heading`和`Footring`的`HorizontalAlignment` 属性的默认设置从`Normal`样式中继承并设置为 `AlignHorzEnum.General`，对于数据单元格，一般的设置意味着潜在的数据类型决定是否单元格文档左对齐，居中对齐或者右对齐，对于列头与列脚，一般的设置意味着列数据单元对齐将会如下。

Caption			
Split 0		Split 1	
Column 0	Column 1	Column 0	Column 1
Data	Data	Data	Data
Data	Data	Data	Data
Footer 0	Footer 1	Footer 0	Footer 1

以下代码设置了C1DisplayColumn 对象的对齐：
To write code in Visual Basic

```
Visual Basic

Me.C1TrueDBGrid1.Splits(0).DisplayColumns(0).Style.HorizontalAlignment =
C1.Win.C1TrueDBGrid.AlignHorzEnum.Center
```

To write code in C#

```
C#

this.c1TrueDBGrid1.Splits[0].DisplayColumns[0].Style.HorizontalAlignment =
C1.Win.C1TrueDBGrid.AlignHorzEnum.Center;
```

示例10：设置列标题的对齐

本示例阐述了列头和列脚的普通与特定对齐之间的区别，如果HeadingStyle（或FooterStyle）属性的HorizontalAlignment 属性成员未被设置为AlignHorzEnum.General，此时头(或脚) 会根据数据单元格独立对齐。

Caption			
Split 0		Split 1	
Column 0	Column 1	Column 0	Column 1
Data	Data	Data	Data
Data	Data	Data	Data
Footer 0	Footer 1	Footer 0	Footer 1

以下代码设置了列头的对齐：
To write code in Visual Basic

```
Visual Basic

With Me.C1TrueDBGrid1.Splits(0).DisplayColumns(0)
.HeadingStyle.HorizontalAlignment = C1.Win.C1TrueDBGrid.AlignHorzEnum.Near
.FooterStyle.HorizontalAlignment = C1.Win.C1TrueDBGrid.AlignHorzEnum.Far .Style.HorizontalAlignment =
C1.Win.C1TrueDBGrid.AlignHorzEnum.Center
End With
```

To write code in C#

```
C#

this.c1TrueDBGrid1.Splits[0].DisplayColumns[0].HeadingStyle.HorizontalAlignment = C1.Win.C1TrueDBGrid.AlignHorzEnum.Near;

this.c1TrueDBGrid1.Splits[0].DisplayColumns[0].FooterStyle.HorizontalAlignment = C1.Win.C1TrueDBGrid.AlignHorzEnum.Far;

this.c1TrueDBGrid1.Splits[0].DisplayColumns[0].Style.HorizontalAlignment = C1.Win.C1TrueDBGrid.AlignHorzEnum.Center;
```

对单元格应用样式

WinForms版True DBGrid 提供了三种方法控制单元格中字符的显示：

Control	Description
按状态	每一个网格都有一个单元格状态用以识别它的性质（当前的连接，更改，选择行的部分，或高亮行的部分），使用AddCellStyle 方法设置样式属性应用于任意可能的单元格状态值连接。

按内容	指定一个模式(成为常规表达式)用于执行模式匹配单元格内容，当内容匹配了在AddRegexCellStyle 方法中的提供的模式，WinForms版版True DBGrid 将自动应用预选择的样式属性到单元格中。
按自定义准则	使用 FetchCellStyle (或 FetchRowStyle) 事件决定单元格颜色和每次一个单元格（或行）字体显示。

在设计时使用Style 对象定义作为AddCellStyle与AddRegexCellStyle方法的参数，或者在代码中创建一个临时的样式以及使用它制定一个或多个属性。
FetchCellStyle 和 FetchRowStyle 事件可以传递一个临时的Style 对象作为最终参数，通过设置它的属性，控制其他事件参数指定单元格的外观。
WinForms版版True DBGrid 中，每个单元格中的字体与颜色只能通过写代码来实现，然而在设计时创建样式，该代码可以保持最小。为了学习如何在设计时创建命名样式，请参阅使用使用C1TrueDBGrid样式编辑器样式编辑器 (Section 6.5)。

指定单元格状态值

C1TrueDBGrid 能够识别16个不同的单元格状态值，用于识别一个单元格的性质，一个单元格状态值是四个单独条件的组合，这些条件是有标签属性的枚举，这意味着它们可以使用Or运算符连接：

Condition	Description
Current Cell	当前单元格由Bookmark, Col及SplitIndex 属性指定，在任意给定的时间，只有一个单元格处于该状态。当浮动编辑器的MarqueeStyle 属性设置有效，则该条件被忽略。
Marquee Row	该单元格是高亮选取行的一部分，当MarqueeStyle 属性指出当前整个行应该被高亮，当前行的所有可见单元格会有额外的条件集。
Updated Cell	单元格内容可以通过用户更改，但不会写入数据库中，当单元格内容在代码中更改Text或Value属性时该条件也可以被设置。
Selected Row	单元格是用户或在代码中所选择行的一部分，SelectedRowCollection 包含每个所选择行的一个书签。

WinForms 版版True DBGrid 会根据单元格内容定义一下常量：

Constant	Description
CellStyleFlag.CurrentCell	应用于当前单元格。
CellStyleFlag.MarqueeRow	应用于高亮选取行中的单元格。
CellStyleFlag.UpdatedCell	应用于被更改的单元格。
CellStyleFlag.SelectedRow	应用于所选择行的单元格。

WinForms 版版True DBGrid也可以定义一下常量，这并不以为这与之前列出的连接：

Constant	Description
CellStyleFlag.AllCells	应用于所有的单元格。
CellStyleFlag.NormalCell	应用于单元格而不使用状态条件。

使用CellStyleFlag.AllCells 来引用所有的单元格而不用考虑状态，使用CellStyleFlag.NormalCell 来引用那些单元格而不用之前描述四个基本单元格条件。

按状态应用单元格样式

WinForms版版True DBGrid 中每一个单元格可以显示一个状态值，可以识别它的性质（当前的连接，一个选择行的部分，或一个高亮行的部分）。使用AddCellStyle 方法设置样式属性用于可能的单元格状态值的任意连接，AddCellStyle 方法在C1TrueDBGrid, C1TrueBDropDown, Split, 与C1DisplayColumn 对象中支持，单元格的范围可以被确定的条件控制。
对于每个唯一状态连接，您可以设置颜色，字体和图像属性用于单元格的状态，当一个单元格状态改变时，WinForms 版版True DBGrid将会检查是否样式属性覆盖会定义单元格，并在显示时应用这些属性到单元格，Style 对象被用于指定一个单元格的颜色和字体，如下示例：
To write code in Visual Basic

Visual Basic

```
Dim S As New Cl.Win.C1TrueDBGrid.Style()
Dim myfont As Font
myfont = New Font(S.Font, FontStyle.Bold) S.Font = myfont
S.ForeColor = System.Drawing.Color.Red
Me.C1TrueDBGrid1.AddCellStyle (Cl.Win.C1TrueDBGrid.CellStyleFlag.CurrentCell, S)
```

To write code in C#

C#

```
C1TrueDBGrid.Style S = new Cl.Win.C1TrueDBGrid.Style();
Font myfont;
myfont = new Font(S.Font, FontStyle.Bold); S.Font = myfont;
S.ForeColor = System.Drawing.Color.Red; this.c1TrueDBGrid1.AddCellStyle(Cl.Win.C1TrueDBGrid.CellStyleFlag.CurrentCell, S);
```

这里，一个新的临时样式对象被创建来指定颜色和字体的覆盖（红色文本，加粗）将被应用于当前整个网格的单元格，因此样式对象的BackColor 属性未被明确设置，当前单元格的背景色没有改变。

First	Last	Country
Isaac	Albeniz	Spain
Johann Sebastian	Bach	Germany
Samuel	Barber	United States
Bela	Bartok	Hungary
Ludwig van	Beethoven	Germany
Alban	Berg	Austria

设计时也会使用样式定义作为AddCellStyle 方法的参数：

To write code in Visual Basic

Visual Basic

```
Dim S As Cl.Win.C1TrueDBGrid.Style
S = Me.C1TrueDBGrid1.Styles("RedBold")
Me.C1TrueDBGrid1.AddCellStyle(Cl.Win.C1TrueDBGrid.CellStyleFlag.CurrentCell, S)
```

To write code in C#

C#

```
Cl.Win.C1TrueDBGrid.Style S; S = this.c1TrueDBGrid1.Styles("RedBold")
this.c1TrueDBGrid1.AddCellStyle(Cl.Win.C1TrueDBGrid.CellStyleFlag.CurrentCell, S);
```

之前的示例可以被简化，使用AddCellStyle 方法接收一个样式名：

To write code in Visual Basic

Visual Basic

```
Me.C1TrueDBGrid1.AddCellStyle(Cl.Win.C1TrueDBGrid.CellStyleFlag.CurrentCell,
"RedBold")
```

To write code in C#

C#

```
this.c1TrueDBGrid1.AddCellStyle(Cl.Win.C1TrueDBGrid.CellStyleFlag.CurrentCell,
"RedBold");
```

前面的示例中当前单元格中的所有文本显示为红色且粗体。然而注意到CellStyleFlag.CurrentCell 仅应用于单元格，它仅拥有该状态，因此该单元格可以更新(CellStyleFlag.CurrentCell+CellStyleFlag.UpdatedCell) 并不会再显示红色粗体，除非以下声明被执行：

To write code in Visual Basic

Visual Basic

```
Me.C1TrueDBGrid1.AddCellStyle(Cl.Win.C1TrueDBGrid.CellStyleFlag.CurrentCell +
Cl.Win.C1TrueDBGrid.CellStyleFlag.UpdatedCell, Me.C1TrueDBGrid1.Styles("RedBold"))
```

To write code in C#

C#

```
this.c1TrueDBGrid1.AddCellStyle(C1.Win.C1TrueDBGrid.CellStyleFlag.CurrentCell+  
C1.Win.C1TrueDBGrid.CellStyleFlag.UpdatedCell, this.c1TrueDBGrid1.Styles["RedBold"]);
```

注意注意：当前单元格状态当MarqueeStyle 属性没有设置为MarqueeEnum.FloatingEditor时才会实现，浮动编辑器选取框总是使用由控制面板设置的系统高亮颜色。

虽然指定单元格条件的方法提供了更多的控制和灵活性，但它也需要写额外的代码应对某些常见的情况。调用AddCellStyle 即可立即起效，它可以用于交互效果并适用于整个网格特征。

按内容应用单元格样式

WinForms版True DBGrid 能够自动应用颜色和字体到特定的单元格，基于它显示的内容，为了实现这种效果以提供一个模式，即常规表达式，用以测试每一个单元格中显示的值，使用AddRegexCellStyle 方法关联带有样式属性的常规表达式，此时应用到可能的单元格状态值连接，AddRegexCellStyle 方法由C1TrueDBGrid, C1TrueDBDropDown, 及C1DisplayColumn 对象提供，允许确定条件的单元格领域可以被控制。

AddRegexCellStyle 方法类似于AddCellStyle 方法，但它需要一个额外常规表达式字串的参数，由于使用AddCellStyle, 您可以使用临时样式或者命名样式，以下示例可以使用一个临时样式显示第一列中的所有单元格，其包含粗体的字串“Windows”:

To write code in Visual Basic

```
Visual Basic  
  
Dim S As New C1.Win.C1TrueDBGrid.Style()  
Dim myfont As Font  
myfont = New Font(S.Font, FontStyle.Bold) S.Font = myfont  
Me.C1TrueDBGrid1.AddRegexCellStyle (C1.Win.C1TrueDBGrid.CellStyleFlag.AllCells, S, "Computer")
```

To write code in C#

```
C#  
  
C1TrueDBGrid.Style S = new C1.Win.C1TrueDBGrid.Style();  
Font myfont;  
myfont = new Font(S.Font, FontStyle.Bold); S.Font = myfont; this.c1TrueDBGrid1.AddRegexCellStyle  
(C1.Win.C1TrueDBGrid.CellStyleFlag.AllCells, S, "Computer");
```

该特性允许实现“visual queries”用以区别单元格字体或颜色属性以匹配一个确定的模式。

Company	LastName	FirstName
Microcomputer Consulting	Clark	Allen
Computer Engineering	Gordon	Alan
Software Designs	Quinn	Ann
Computer Communications	Wheeler	Alice
Microcomputers Unlimited	Ardmore	Beverly
Computer Applications	Fredericks	Carey
Micro Mechanics	Ziegler	Carl
Software Specialists	Elkins	David

按自定义准则应用单元格样式

常规表达式式用于格式化表达的需求可能是不足的，使用FetchCellStyle 事件自定义每一个基本单元格的字体和颜色，该事件仅可以触发FetchStyle 属性被设置为True的列。

例如，可以在确定领域中提供值的颜色代码，以下代码假设一个单独的数字数据列的FetchStyle 属性为True ，并控制FetchCellStyle 事件用蓝色显示超过1000的值:

To write code in Visual Basic

```
Visual Basic  
  
Private Sub C1TrueDBGrid1_FetchCellStyle(ByVal sender As Object, ByVal e As  
C1.Win.C1TrueDBGrid.FetchCellStyleEventArgs) Handles C1TrueDBGrid1.FetchCellStyle  
  
Dim N As Integer  
N = Val(Me.C1TrueDBGrid1(e.Row, e.Col) If N > 1000 Then  
e.CellStyle.ForeColor = System.Drawing.Color.Blue End If  
End Sub
```

To write code in C#

C#

```
private void c1TrueDBGrid1_FetchCellStyle( object sender,
Cl.Win.C1TrueDBGrid.FetchCellStyleEventArgs e) { int N; N = (int) this.c1TrueDBGrid1[e.Row, e.Col]; if ( N > 1000 ) {

e.CellStyle.ForeColor = System.Drawing.Color.Blue; }
}
```

Split, Row和 Col 属性可以识别网格中哪一个单元格的显示, CellStyle 属性会将格式化信息从应用传递到网格中, 由于CellStyle 属性是一个Style 对象, 单元字体特性也可以在FetchCellStyle事件中更改:

To write code in Visual Basic

Visual Basic

```
If N > 1000 Then
e.CellStyle.Font.Italic = True Dim myfont As Font
myfont = New Font (e.CellStyle.Font, FontStyle.Italic) If N > 1000 Then

e.CellStyle.Font = myfont
```

To write code in C#

C#

```
if ( N > 1000 ) {
e.CellStyle.Font.Italic = true } Font myfont; myfont = new Font (e.CellStyle.Font, FontStyle.Italic); if ( N > 1000 ) {

e.CellStyle.Font = myfont;
}
```

FetchCellStyle 事件也可以用于应用基于其他单元格的格式甚至是其他控件到一个单元格中, 例如, 假设您想要:
设定如果第1列减去第2列是负值, 那么列的单元格文本则为红色。

设置如果能匹配文本框中的内容, 那么第7列中的文本加粗。

在这种情况下, 对第4列和第7列的FetchStyle 属性设置为True, 并处理FetchCellStyle 事件如下:

To write code in Visual Basic

Visual Basic

```
Private Sub C1TrueDBGrid1_FetchCellStyle(ByVal sender As Object, ByVal e As
Cl.Win.C1TrueDBGrid.FetchCellStyleEventArgs) Handles C1TrueDBGrid1.FetchCellStyle

Select Case e.Col
Case 4
Dim Col1 As Long, Col2 As Long
Col1 = CLng(Me.C1TrueDBGrid1(e.Row, 1))
Col2 = CLng(Me.C1TrueDBGrid1(e.Row, 2))
If Col1 - Col2 < 0 Then
CellStyle.ForeColor = System.Drawing.Color.Red
Case 7
Dim S As String
S = Me.C1TrueDBGrid1(e.Row, 7).ToString()
If S = TextBox1.Text Then
Dim myfont = New Font(CellStyle.Font, FontStyle.Bold)
CellStyle.Font = myfont
End If
Case Else
Debug.WriteLine ("FetchCellStyle not handled: " & e.Col)
End Select
End Sub
```

To write code in C#

C#

```
private void c1TrueDBGrid1_FetchCellStyle( object sender,
C1.Win.C1TrueDBGrid.FetchCellStyleEventArgs e) { switch (e.Col) { case 4:
long Col1, long Col2; Col1 = (long)this.c1TrueDBGrid1[e.Row, 1]; Col2 = (long)this.c1TrueDBGrid1[e.Row, 2]; if ( Col1 -
Col2 < 0 ) CellStyle.ForeColor = System.Drawing.Color.Red break; case 7:

string S; S = this.c1TrueDBGrid1[e.Row, 7].ToString(); if ( S == TextBox1.Text ) {

Font myfont = new Font(CellStyle.Font, FontStyle.Bold);

CellStyle.Font = myfont; } break; default: Console.WriteLine (”FetchCellStyle not handled: ” + e.Col);
}
}
```

出于更有效的因素，可以只设置列的FetchStyle属性为True，因而您可以在FetchCellStyle事件中处理。

注意注意：之前的示例中使用CellText 方法用于简化，然而，CellText 和 CellValue 方法每次被调用可以创建和删除一个内部的数据集的副本，但调用它们缺乏效率而不能在FetchCellStyle事件中使用，为了改善网格的显示周期的性能，非绑定应用可以直接访问基础数据源，通常会比调用CellText 或 CellValue方法更快。

为了自定义基于行的字体和颜色而不是基于单元格的，使用的FetchRowStyle 事件可以通过为网格的每一行设置FetchRowStyles 属性为True而被触发一次，事件的语法如下：
To write code in Visual Basic

```
Visual Basic

Private Sub TDBGrid1_FetchRowStyle(ByVal sender As Object, ByVal e As
C1.Win.C1TrueDBGrid.FetchRowStyleEventArgs) Handles C1TrueDBGrid1.FetchRowStyle
```

To write code in C#

```
C#

private void TDBGrid1_FetchRowStyle( object sender,

C1.Win.C1TrueDBGrid.FetchRowStyleEventArgs e)
```

虽然FetchRowStyle 事件可以用于实现间隔行颜色方案，一个更容易且更有效的方法是使用AlternatingRows属性完成相同的任务，可以同时使用内置的EvenRow 样式和 OddRow 样式。
[10.3.5 单元格样式评估顺序](#)

应用图片到网格元素中

WinForms版True DBGrid 的早前版本中，样式可以用于指定字体，颜色和对齐属性，本版本可以扩展样式概念至背景与前景图片，用以修饰于列头，列脚以及标题栏，指定数据单元格的背景模式，并且可以在不用填充ValueItems 对象的情况下在单元格中呈现图片数据，Style对象的以下属性决定了图片将如何显示：

Property	Description
BackgroundImage	设置/返回一个样式的背景图。
BackgroundPictureDrawMode	控制一个样式背景图如何显示。
ForegroundImage	设置/返回一个样式的前景图。
ForeGroundPicturePosition	控制一个样式的前景图如何被放置。

因此图片属性会遵循其他样式属性的相同继承规则，在本章中早前描述的技术可以使用图片，这意味着图片可以通过以下方法添加到一个网格元素中：
在设计器或代码中设置内置命名样式的BackgroundImage 或ForegroundImage属性。在设计器或代码中设置匿名样式的BackgroundImage 或 ForegroundImage 属性。调用AddCellStyle 或 AddRegexCellStyle 方法。为FetchCellStyle 或 FetchRowStyle 事件写一个处理程序。

显示背景图片

使用背景图片来自定义网格元素，如标题栏，列头，列脚。例如以下代码将一个彩色的渐变位图用于被网格CaptionStyle 属性返回的Style 对象的BackgroundImage 成员：
To write code in Visual Basic

```
Visual Basic
```

```
With Me.C1TrueDBGrid1.CaptionStyle
. BackgroundImage = System.Drawing.Image.FromFile("c:\bubbles.bmp")

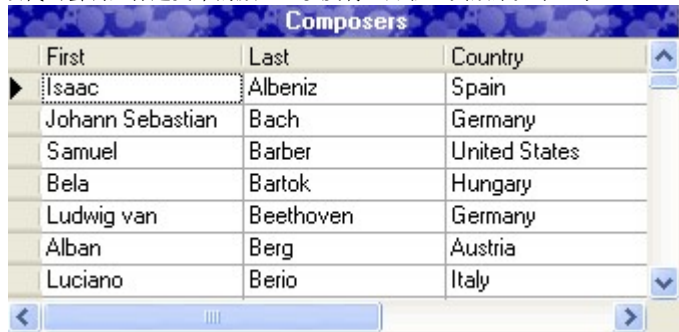
. BackgroundPictureDrawMode =
C1.Win.C1TrueDBGrid.BackgroundPictureDrawModeEnum.Tile
. ForeColor = System.Drawing.Color.White
. Font = New Font(.Font, FontStyle.Bold)
End With
```

To write code in C#

```
C#

this.c1TrueDBGrid1.CaptionStyle.BackgroundImage = System.Drawing.Image.FromFile(@"c:\bubbles.bmp");
this.c1TrueDBGrid1.BackgroundPictureDrawMode =
C1.Win.C1TrueDBGrid.BackgroundPictureDrawModeEnum.Tile; this.c1TrueDBGrid1.CaptionStyle.ForeColor =
System.Drawing.Color.White;
this.c1TrueDBGrid1.CaptionStyle.Font = new Font(this.c1TrueDBGrid1.CaptionStyle.Font,
FontStyle.Bold);
```

该代码会调整标题文本的颜色，以及将它加粗，其效果如下显示。



为了在设计时实现相同的效果，也可以编辑C1TrueDBGrid Style Editor中的内置Caption样式，或者属性窗口中的 CaptionStyle属性成员。默认情况下，网格元素中的背景图位置居中，根据背景位图的高度调整BackgroundPictureDrawMode 属性值以确保整个区域被填充，该属性决定了图片是否会居中，平铺或者拉伸以适应整个区域，其显示如下表。

居中	平铺	拉伸
----	----	----





也可以在数据单元格中使用背景图以生成有趣的视觉效果，例如，以下模式可以被设计为为相邻行的复制。



通过清除选择列的记录，以及在数据行和列头分隔之间分隔行，这些模式可以用于产生如下显示。

Title	ISBN
Advanced Html : How to Do Cool Things With Your Web Site	1-5561594-8-X
Developing Applications With Microsoft Office : Strategies for Designing, Develo	1-5561566-5-0
Developing Business Solutions With Microsoft Visual Basic and Microsoft Office	1-5561589-9-8
Developing International Software for Windows 95 and Windows Nt (Microsoft F	1-5561584-0-8
Developing Microsoft Excel 95 Solutions : With Visual Basic for Applications/Bo	1-5561589-3-9
Field Guide to Microsoft Powerpoint for Windows 95	1-5561584-1-6
Field Guide to MS Access for Windows 95	1-5561587-5-0
Field Guide to the Internet	1-5561582-2-X
Hardcore Visual Basic/Book and Disk	1-5561566-7-7
Hitchhiker's Guide to Visual Basic & Sql Server -- Fourth Edition (Book and Disk)	1-5561590-6-4
Inside Odbc/Book and Cd-Rom	1-5561581-5-7
Inside Ole/Book and Cd-Rom (Microsoft Programming	1-5561584-3-2
Inside Visual C++ (Microsoft Programming/Book and Cd-Rom	1-5561589-1-2
Learn Microsoft Visual Basic 4 Now : The Complete Learning Solution for Visual	1-5561590-5-6
Mastering Microsoft Access/Book and Cd-Rom (Microsoft Mastering Series	1-5561591-2-9
Mastering Microsoft Visual Basic/Book and Cd Rom (Microsoft Mastering Series	1-5561591-3-7

其关键在于在左侧的空白非绑定列中插入一张图以显示订书环，以下代码将演示示例：
To write code in Visual Basic

```
Visual Basic

' 给出网格的一个平面外观并记录选择，行分隔和滚动条移除。
With Me.C1TrueDBGrid1
    .InactiveStyle.ForeColor = System.Drawing.Color.White
    .RecordSelectors = False
    .RowDivider.Style = LineStyleEnum.None
    .RowHeight = 16
    .HScrollBar.Style = ScrollBarStyleEnum.None
    .VScrollBar.Style = ScrollBarStyleEnum.None
    .MarqueeStyle = MarqueeEnum.NoMarquee
End With
```

```

' 设置背景模式用于默认的数据单元格(不会打断Normal样式)。
With Me.ClTrueDBGrid1.Splits(0).Style
    .BackgroundImage = System.Drawing.Image.FromFile("paper.bmp") .BackgroundPictureDrawMode =
BackgroundPictureDrawModeEnum.Tile
End With

' 在左侧创建一个空的非绑定列用以装载订书环, 移除它的分隔线并设置它Style对象的BackgroundBitmap属性。
Dim col as New ClTrueDBGrid.ClDataColumn()
Me.ClTrueDBGrid.Columns.InsertAt(0, col) Dim C As ClTrueDBGrid.ClDisplayColumn
C = Me.ClTrueDBGrid1.Splits(0).DisplayColumns(col)
With C
    .Width = 48
    .Visible = True
    .Style.BackgroundImage = System.Drawing.Image.FromFile("rings.bmp")
    .HeaderDivider = False
    .ColumnDivider.Style = LineStyleEnum.None
End With

' 滚动被绑定列到视图中。
Me.ClTrueDBGrid1.Col = 0

' 重新制定Title列的大小并移除它的头分隔线。
Set C = Me.ClTrueDBGrid1.Splits(0).DisplayColumns("Title")
With C
    .Width = 380
    .HeaderDivider = False
End With

' 使用订书环位图的一个小角作为列标题的一个背景, 并调整相应的字体与文本颜色。
Dim myfont As Font
With Me.ClTrueDBGrid1.HeadingStyle
    .BackgroundImage = System.Drawing.Image.FromFile("corner.bmp") .BackgroundPictureDrawMode =
BackgroundPictureDrawModeEnum.Tile myfont = New Font(.Font, 10, FontStyle.Bold) .Font = myfont
    .ForeColor = System.Drawing.Color.White
End With

```

To write code in C#

C#

```

// 给出网格的一个平面外观并记录选择, 行分隔和滚动条移除。
this.clTrueDBGrid1.InactiveStyle.ForeColor = System.Drawing.Color.White; this.clTrueDBGrid1.RecordSelectors = false;
this.clTrueDBGrid1.RowDivider.Style = LineStyleEnum.None; this.clTrueDBGrid1.RowHeight = 16;
this.clTrueDBGrid1.HScrollBar.Style = ScrollBarStyleEnum.None; this.clTrueDBGrid1.VScrolBar.Style =
ScrollBarStyleEnum.None; this.clTrueDBGrid1.MarqueeStyle = MarqueeEnum.NoMarquee;

```

```

// 设置背景模式用于默认的数据单元格(不会打断Normal样式)。

```

```

this.clTrueDBGrid1.Splits[0].Style.BackgroundImage = System.Drawing.Image.FromFile("paper.bmp");
this.clTrueDBGrid1.Splits[0].Style.BackgroundPictureDrawMode = BackgroundPictureDrawModeEnum.Tile;

```

```

// 在左侧创建一个空的非绑定列用以装载订书环, 移除它的分隔线并设置它Style对象的BackgroundBitmap属性。
ClTrueDBGrid.ClDataColumn col = new ClTrueDBGrid.ClDataColumn(); this.ClTrueDBGrid.Columns.InsertAt(0, col);
ClTrueDBGrid.ClDisplayColumn C = this.clTrueDBGrid1.Splits[0].DisplayColumns[col];
C.Width = 48;
C.Visible = true;
C.Style.BackgroundImage = System.Drawing.Image.FromFile["rings.bmp"];
C.HeaderDivider = false;
C.ColumnDivider.Style = LineStyleEnum.None;

```

```

// 滚动被绑定列到视图中。
this.clTrueDBGrid1.Col = 0;

```

```

// 重新制定Title列的大小并移除它的头分隔线。
C = this.clTrueDBGrid1.Splits[0].DisplayColumns["Title"];
C.Width = 380;
C.HeaderDivider = false;

```

```

// 使用订书环位图的一个小角作为列标题的一个背景, 并调整相应的字体与文本颜色。
Font myfont;
this.clTrueDBGrid1.HeadingStyle.BackgroundImage = System.Drawing.Image.FromFile("corner.bmp");
this.clTrueDBGrid1.HeadingStyle.BackgroundPictureDrawMode = BackgroundPictureDrawModeEnum.Tile; myfont = new
Font(.Font, 10, FontStyle.Bold); this.clTrueDBGrid1.HeadingStyle.Font = myfont;
this.clTrueDBGrid1.HeadingStyle.ForeColor = System.Drawing.Color.White;

```

显示前景图片

使用前景图片对静态网格元素提供可视化提示，如标题栏，列头以及列脚。前景图片由Style的ForegroundImage 属性指定，前景图可以显示在除文本以外的地方，或者代替文本显示，但不可以覆盖文本显示。

前景图片有ForeGroundPicturePosition 属性，用以指定前景图片在单元文本中的相对位置，值与呈现如下图所示：

PositionDisplayNear



Far



LeftOfText	
RightOfText	
TopOfText	
BottomOfText	
PictureOnly	
TextOnly	