单元格编辑技术

本章节将介绍WinForms版版True DBGrid 中如何自定义单元格编辑的行为，在文本输入字段中，可以在网格编辑事件中写入代码，指定一个输入掩码模板，或者在显示用于长字符串的下拉文本编辑器，为了给用户提供一系列选择，可以使用ValueItemCollection 对象，C1TrueDBDrop Down 控件，任意的内部控件或者第三方控件。

如何使用单元格编辑

行为会根据MarqueeStyle 属性的设置，如果浮动编辑器选取框样式被使用，编辑行为会不同于其他的选取框样式，以下小节总结了WinForms版版True DBGrid 的编辑行为和状态。
等多关于MarqueeStyle属性详细信息，请参阅高亮当前行或单元格高亮当前行或单元格 。

初始化单元格编辑

一个单元有显示模式和编辑模式两种，EditActive 属性设置并返回需要的模式，通过设置EditActive为True 可以将当前单元格设置为编辑格式，或者将它设置为False结束编辑格式，用户可以点击当前单元格或按F2键时，一个闪烁文本
光标（插入符号）将出现在当前单元格中—当单元格被点击的开始以及在按F2键后，当单元格进入编辑模式时BeforeColEdit 事件将被触发，当单元格处于编辑模式时EditActive 属性为True。
浮动编辑器的不同：当处于显示模式时一个闪烁的插入符将出现在高亮单元格的开始，为了进入编辑模式，用户可以在单元格文本中点击任意的字符位置来指定需要的文本插入位置，BeforeColEdit 事件将不会触发并且EditActive 属性为False 直到用户更改了单元格中的文本。

颜色与换行

在编辑模式中，单元格的颜色由EditorStyle 样式对象的ForeColor 和 BackColor 属性决定，被编辑的文本将会换行，不管列属性的WrapText 属性为何值，如果文本太多而不能适应单元格，则内置的下拉编辑控件会自动出现，更多详细信息，请参阅使用文本使用文本 (Section 11.3)。
浮动编辑不同：在编辑模式中将会出现文本高亮，单元格颜色与常规单元格颜色相同，被编辑的文本可以换行仅当列样式的WrapText 属性为True，内置下拉编辑控件不可用。

确定更改状态

在编辑过程中，会检查网格的DataChanged 属性以决定用户是否对当前行做出更改。
设置网格的DataChanged 属性为False 来退出编辑，丢弃当前行的所有更改，并从数据源中刷新当前行。
当前行中记录选择列的图标反映了网格DataChanged 属性的状态，如果DataChanged 为False，一个三角形箭头将出现在记录选择列，如果DataChanged 为True，将出现一个铅笔的图标。

决定单元格内容

在编辑过程中，列的Text 和 Value 属性包含了更改行中用户当前可以看到的文本，无论何时用户按一个键，Change 事件会触发并通知应用程序用户可以更改当前单元格，然而，Change 事件并不意味着用户结束了进城，只有一个变更会被设置并且网格仍处于编辑模式。
当网格未处于编辑模式中Change 事件不能被触发，如一个单元格的内容可以通过代码更改或用户点击一个单元格使用ValueItem 对象来循环。

终止单元格编辑

用户可以按照以下方式完成编辑过程：
按ENTER键。
按ESC键。
使用箭头键移除其他单元格，或者TAB键和鼠标。
对窗体中其他控件设置聚焦。 处理编辑事件

以下章节描述了WinForms版版True DBGrid 的默认编辑行为可以用过事件响应来更改。

标准按键事件

WinForms版版True DBGrid 支持.NET环境下包含的标准按键事件：

Event	Description
KeyDown	当用户按下一个键时触发。
KeyPress	当用户按下一个ANSI键时触发。
KeyUp	当用户释放一个键时触发。

KeyDown 和 KeyUp 事件可以捕获所有的按键，包括功能按键，ALT 和SHIFT键，以及数字键盘按键。KeyPress 事件仅可以捕获字母和数字，标点和符号以及编辑按键如TAB, ENTER,和BACKSPACE。
使用这些事件可以限制和修改用户的输入，正如您对其他内部的.NET控件操作一样，例如下述的KeyDown 事件处理可以阻止用户输入非字母数字的字符：
To write code in Visual Basic

```
Visual Basic
Private Sub C1TrueDBGrid1_KeyPress(ByVal sender As Object, ByVal e As

System.Windows.Forms.KeyPressEventArgs) Handles C1TrueDBGrid1.KeyPress


' 如果它不是一个字母或数字，则取消用户的键入。
If Not e.KeyChar.IsLetterOrDigit(e.KeyChar) Then
e.Handled = True End If
End Sub
```

To write code in C#

```
C#
private void C1trueDBGrid1_KeyPress(object sender,

System.Windows.Forms.KeyPressEventArgs e)
{
// 如果它不是一个字母或数字，则取消用户的键入。
if (! e.Keychar.IsLetterOrDigit(e.KeyChar])

{
e.Handled = true ;
}
}
```

更多关于主题相关的信息以及本地.NET事件，请参阅MSDN 或 .NET 帮助。

列编辑事件

WinForms版版True DBGrid 使用以下事件提供全控件的单元格编辑过程， 在一个成功的编辑尝试过程中以下列出的将会触发：

Event	Description
BeforeColEdit	尝试编辑列数据时触发。
ColEdit	当当前单元格进入编辑模式时触发。
AfterColEdit	在列数据被编辑后触发。

使用BeforeColEdit 事件控制每个单元格的可编辑性，或将初始按键转换为一个默认值。
当前单元格中的ColEdit 事件信号表明已经进入到编辑模式，AfterColEdit 事件信号表明编辑模式已经被终止，使用这两个事件在过程中提供额外的编辑：
To write code in Visual Basic

```
Visual Basic
Private Sub C1TrueDBGrid1_ColEdit(ByVal sender As Object, ByVal e As
C1.Win.C1TrueDBGrid.ColEventArgs) Handles C1TrueDBGrid1.ColEdit
Select Case e.Columns.DataColumn.Caption
Case "Code"
Me.Label1.Text = "Enter 4-digit company code"
Case "Description"
Me.Label1.Text = "Enter full company name"
End Select
End Sub

Private Sub C1TrueDBGrid1_AfterColEdit (ByVal sender As Object, ByVal e As

C1.Win.C1TrueDBGrid.ColEventArgs) Handles C1TrueDBGrid1.AfterColEdit

' 清除编辑指令。
Me.Label1.Text = ""
End Sub
```

To write code in C#

```
C#
```

```
private void C1trueDBGrid1_ColEdit(object sender, C1.Win.C1TrueDBGrid.ColEventArgs e) {
switch(e.Columns.DataColumn.Caption) { Case "Code":
```

```
this.Label1.Text = "Enter 4-digit company code"; break; Case "Description";
this.Label1.Text = "Enter full company name"; break; }
}
private void C1TrueDBGrid1_AfterColEdit(object sender,
C1.Win.C1TrueDBGrid.ColEventArgs e)
}
// 清除编辑指令。
this.Label1.Text = "";
}
```

使用一个按键更改单元格中的内容

您可以使用BeforeColEdit 事件自定义WinForms版版True DBGrid的编辑行为，BeforeColEdit 在其他编辑事件发生前触发，并在编辑开始前提供机会做您想要做的工作，例如，取消编辑请求和使用您的下拉列表框覆盖内置文本编辑器。
一个WinForms版版True DBGrid控件可以以四种方法进入编辑模式：
若用户鼠标点击了当前单元格，可以开始编辑当前的单元格内容。
如果用户按了F2键，编辑也可以使用当前单元格内容开始使用。
如果用户开始键入，键入的字符可以替代单元格和编辑开始的内容。
您可以在代码中设置EditActive 属性可以使编辑开始。
BeforeColEdit 事件在前三个情况中触发，但不使用于最后一个情况，因为WinForms版版True DBGrid假设您不想在代码中取消。
为了区分一个用户的是否使用鼠标或者键盘开始编辑的请求，可以设置BeforeColEdit 为KeyChar，当用户用鼠标点击单元格它将会为0，如果用户键入一个字符开始编辑它将是ASCII字符。
当BeforeColEdit 被出发了，ASCII 字符也没有被替换到当前单元格中，因此如果在BeforeColEdit中的编辑被取消，
ASCII码将被丢弃，这是一个有趣的技术。
假设存在一个布尔型名为Done的字段，并且它的NumberFormat 属性被设置为指定显示格式的Yes/No，进一步假设当用户按了Y 或
N，单元格中的内容会即刻改变而不是进入编辑模式，这个过程会在下述的BeforeColEdit 中完成:
To write code in Visual Basic

```
Visual Basic
```

```
Private Sub C1TrueDBGrid1_BeforeColEdit(ByVal sender As Object, ByVal e As
C1.Win.C1TrueDBGrid.BeforeColEditEventArgs) Handles C1TrueDBGrid1.BeforeColEdit

With Me.C1TrueDBGrid1.Columns(e.ColIndex)

' 如果这不能在"Done"列中完成，或者如果用户鼠标点击了，这时它就会继续。
If .DataField <> "Done" Or e.KeyChar = Chr(0) Then Exit Sub

' 取消常规编辑并设置字段为基于KeyChar的适当的结果，如果输入无效的字符就会蜂鸣。

Select Case UCase(e.KeyChar)
Case "Y"
.Value = -1
Case "N"
.Value = 0

Case Else
Beep()
End Select
End With
End Sub
```

To write code in C#

```
C#
```

```
private void C1TrueDBGrid1_BeforeColEdit( object sender,
C1.Win.C1TrueDBGrid.BeforeColEditEventArgs e)
{
C1.Win.C1DataColumn col = e.Column.DataColumn;

// 如果这不能在"Done"列中完成，或者如果用户鼠标点击了，这时它就会继续。
if (col.DataField != "Done"e.KeyChar == 0 ) return;

// 取消常规编辑并设置字段为基于KeyChar的适当的结果，如果输入无效的字符就会蜂鸣。

e.Cancel = true; switch (e.KeyChar. .ToUpper()) {
case "Y"; Col.Value = -1; break; case "N"; Col.Value = 0; default:; Beep();
}
}
```

注意，当KeyChar为0，事件件处理将终止，因此鼠标编辑仍然被允许。

使用文本

本节将简要介绍与文本编辑相关的属性。

限制数据输入字段的大小

使用C1DataColumn 对象的DataWidth 对象限制用户能够输入的字符数，设置该属性为0表明没有大小限制。

对长字段提供下拉编辑控件

当用户尝试编辑单元格的文本因为文本量太大而无法适应单元格时，网格将自动激活一个多行下拉文本编辑器，当编辑时，在下拉编辑控件中的文本将自动换行而不需要设置列样式的WrapText 属性，下拉文本编辑器可以关掉，编辑可以通过设置网格的EditDropDown 属性为False (默认值为True)在单元格边界内使用，如果网格的MarqueeStyle属性被设置为MarqueeEnum.FloatingEditor，下拉文本编辑器将不可用，以下代码使用网格的内置列按钮激活下拉编辑控件来更改在Comments列中的单元格数据：
To write code in Visual Basic

Visual Basic

```
Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs)

Handles MyBase.Load
With Me.C1TrueDBGrid1
.MarqueeStyle = MarqueeEnum.SolidCellBorder
.Splits(0).DisplayColumns("Comments").Button = True

' 冗余，因为默认值为True。
.EditDropDown = True
End With
End Sub

Private Sub C1TrueDBGrid1_ButtonClick(ByVal sender As Object, ByVal e As
C1.Win.C1TrueDBGrid.ColEventArgs) Handles C1TrueDBGrid1.ButtonClick
' 单元格置为编辑模式。
Me.C1TrueDBGrid1.EditActive = True
End Sub
```

To write code in C#

C#

```
private void Form1_Load(System.object sender, System.EventArgs e) {

C1TrueDBGrid1.MarqueeStyle = MarqueeEnum.SolidCellBorder;
C1TrueDBGrid1.Splits[0].DisplayColumns["Comments"].Button = true;
// 冗余，因为默认值为True。
C1TrueDBGrid1.EditDropDown = true;
}
private void C1TrueDBGrid1_ButtonClick(object sender,
C1.Win.C1TrueDBGrid.ColEventArgs e)
{
// 单元格置为编辑模式。
this.c1TrueDBGrid1.EditActive = true;
}
```

如果当前单元格在Comments 列中，可以点击当前单元格或点击内置按钮编辑初始化。

选择与替换文本

WinForms版版True DBGrid 可以在许多TextBox 类型控件中支持标准文本选择属性：

Property	Description
SelectionLength	设置/返回所选择文本的长度。
SelectionStart	设置/返回所选择文本的起始位置。
Property	Description
SelectedText	设置/返回所选择文本。

注意注意：这些属性仅在网格处于编辑模式中有效，也就是说它的EditActive 属性为True。

输入掩码

使用控件的NumberFormat 属性显示列数据的格式，如果用户需要编辑一个格式化的列，最好的方法是在编辑过程中
保持了一致的格式，WinForms版版True DBGrid 提供了一个EditMask 可选择属性与NumberFormat 属性确保数据输入的一致性。

指定列的一个输入掩码

C1DataColumn 对象的EditMask 属性用于指定一个适用于最终用户输入的输入掩码模板，输入掩码字串由特殊的字符组成，它可以代表用户必须输入字符，也可以是跳过输入的文字字符，验证模板字符如下:
EditMask 必须是由以下符号组成的字符串：

1. 通配符通配符

0 数字
9 数字或空格

1. 数字或符号
   L 字母
   ? 字母或空格 A 字母或数字
   a 字母，数字或空格
   & 任意字符
2. 本地化字符本地化字符

. 本地化十进制分隔符
, 本地化千进制分隔符
: 本地化时间分隔符
/ 本地化日期分隔符

1. 指令字符指令字符

\ 下一个字符作为文字
> 转换字母为大写

< 转换字母为小写
例如：
To write code in Visual Basic

```
Visual Basic
’ 设置掩码因而用户可以输入一个电话号码，使用可选区域代码和大写状态。

Me.C1TrueDBGrid1.Columns(0).EditMask = "(###) 000-0000 St\ate\: >LL"
```

To write code in C#

```
C#
// 设置掩码因而用户可以输入一个电话号码，使用可选区域代码和大写状态
this.c1TrueDBGrid1.Columns[0].EditMask = "(###) 000-0000 St\\ate\: >LL";
```

使用掩码来格式化

尽管EditMask 属性可以用于指定一个网格中数据的格式化显示，如果列的NumberFormat 属性未被指定，网格可以简单的显示缓存文本（剥离文字），如果NumberFormat 属性被指定，网格会发送缓存文本用以显示格式。由于常见的输入和显示格式都是相同的，NumberFormat 属性有一个Edit Mask 选项，如果该选项被选择，此时EditMask 属性设置将被用于数据输入和显示，然而输入和显示格式不需要相同，因此一个NumberFormat 选项不同于可以被选择的EditMask 属性。

11.4.3 掩码输入如何更新

通常情况下，当用户完成在列中的一个单元格的编辑后，它有它的EditMask 属性集，Winforms版版True DBGrid 缓存
被更改的单元格文本，但在输入掩码模板中的任意文字字符可以事先从被更改的单元格文本中剥离，然而，该行为可以使用EditMaskUpdate 属性覆盖。
默认情况下，EditMaskUpdate 属性为False，这意味着当被更改的单元格文本更新到数据库中，网格发送缓存文本
（从文字中剥离），而不会格式化在单元格中的文本，通过设置EditMaskUpdate属性为True覆盖默认的行为，在更新数据库之前缓存文本可以根据EditMask 属性格式化。
因此，为EditMaskUpdate 设置合适的值，以确保数据库更新时正确的数据会被发送。

单元格内按钮

WinForms版版True DBGrid 支持当前单元格或在指定列中的所有单元格中不同的单元格内选项，使用单元格内按钮指定一列可用的选择，为了执行单元格内容相关的一个指令，或者显示一个任意的控件或表格的编辑。

启用单元格内按钮

为了启用C1DisplayColumn 对象的单元格按钮，可以在代码中设置它的Button 属性为True：
To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.Splits(0).DisplayColumns(0).Button = True
```

To write code in C#

```
C#
this.c1TrueDBGrid1.Splits[0].DisplayColumns[0].Button = true;
```

当列的DropDown 属性被设置为C1TrueDBDropDown 控件名称时，Button 属性也可以被启用，或者与ValueItemCollection 对象相关联的Presentation 属性被设置为组合框选项之一。
默认情况下，单元格内按钮仅在当前单元格中显示，其显示如下图：

然而，通过设置列的ButtonAlways 属性为True，您可以使单元格内按钮显示在每一行中：

绘制单元格为指令按钮

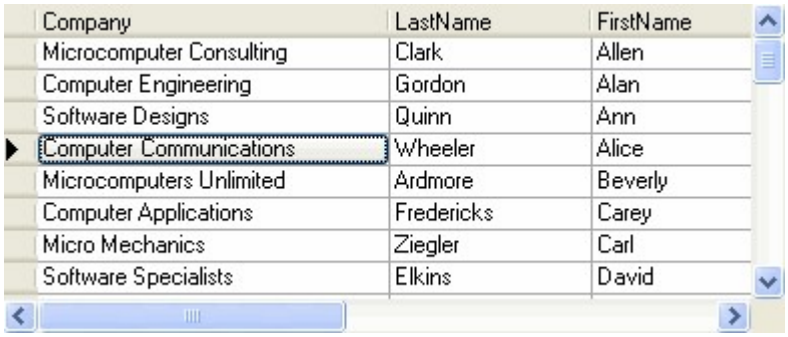
为了在一个C1DisplayColumn对象中绘制当前单元格为一个非编辑指令按钮，在代码中设置它的ButtonText 属性为True ：
To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.Splits(0).DisplayColumns(0).ButtonText = True
```

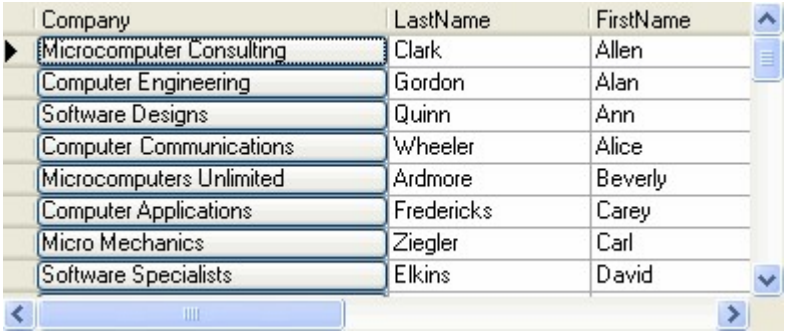
To write code in C#

```
C#
this.c1TrueDBGrid1.Splits[0].DisplayColumns[0].ButtonText = true;
```

当在列中的一个单元格被聚焦时，它会将单元格文本作为标题绘制一个标准窗口指令，单元格文本不能自动居中，但代表列的水平和垂直对齐设置：



如果Button 和ButtonText 属性被设置为True 时，ButtonText属性优先。
由于使用了默认单元格内按钮，设置列的ButtonAlways 属性为True 可以使全部单元格显示为指令按钮，然而，聚焦矩形框仅在当前单元格中绘制：



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

监控单元格内按钮点击

ButtonClick 事件可以通过代码响应用户单元格内按钮点击，其语法如下：
To write code in Visual Basic

```
Visual Basic

Private Sub C1TrueDBGrid1_ButtonClick(ByVal sender As Object, ByVal e As

C1.Win.C1TrueDBGrid.ColEventArgs) Handles C1TrueDBGrid1.ButtonClick
```

To write code in C#

```
C#

private void C1TrueDBGrid1_ButtonClick( object sender,

C1.Win.C1TrueDBGrid.ColEventArgs e)
```

当单元格内按钮被点击后它总是被触发，无论它们是否通过Button 或 ButtonText启用，ButtonClick事件示例在之前的章节使用文本使用文本(Section 11.3)中呈现了。

自定义单元格内按钮位图

默认情况下，WinForms版版True DBGrid 单元格按钮的一个向下箭头。

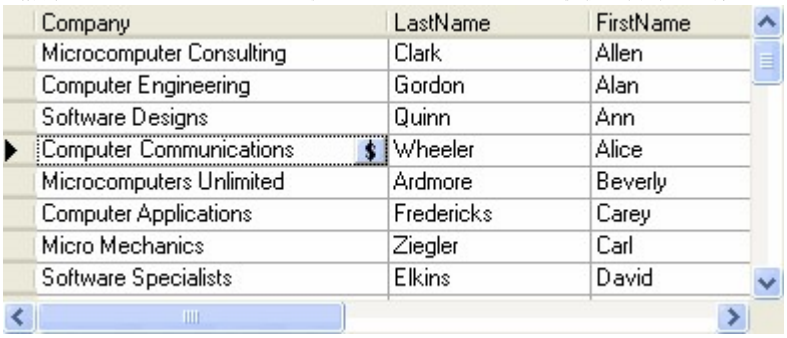
然而，按钮位图可以在设计时在代码中设置ButtonPicture属性修改C1DisplayColumn 对象修改：
To write code in Visual Basic

```
Visual Basic

Me.C1TrueDBGrid1.Columns(0).ButtonPicture =

System.Drawing.Image.FromFile("dollar.bmp")
```

To write code in C#

```
C#

this.c1TrueDBGrid1.Columns[0].ButtonPicture =System.Drawing.Image.FromFile("dollar.bmp");
```

网格会自动绘制相应的向上/向下状态地边缘，因此仅需要提供按钮内部的图像。



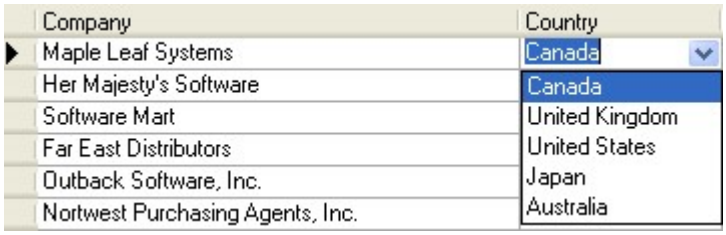
Company	LastName	FirstName
Microcomputer Consulting	Clark	Allen
Computer Engineering	Gordon	Alan
Software Designs	Quinn	Ann
Computer Communications	Wheeler	Alice
Microcomputers Unlimited	Ardmore	Beverly
Computer Applications	Fredericks	Carey
Micro Mechanics	Ziegler	Carl
Software Specialists	Elkins	David

下拉控件

WinForms版版True DBGrid 提供了大量不同的内置控件和可编程结构，可以使您完成各种各样的下拉编辑接口，使用ValueItems 对象以及它的ValueItem 对象集提供一个简单的选择列表，或者C1TrueDBDropDown 控件可以实现一个数据感知的多列组合框，任意的Visual Basic 或者第三方控件可以用于执行特定的编辑功能。

使用内置组合框

C1DataColumn 对象的ValueItems 对象提供可选择的内置组合狂接口，它可以与他的自动数据转换功能一起使用，默认情况下，Presentation 属性被设置为PresentationEnum.Normal，并且通常的单元格编辑行为会影响文本数据，然而，如果Presentation 属性被设置为PresentationEnum.ComboBox 或者 PresentationEnum.SortedComboBox，此时被影响的单元格会聚焦显示单元格内按钮，当用户点击了单元格内按钮，下拉组合框即会出现。



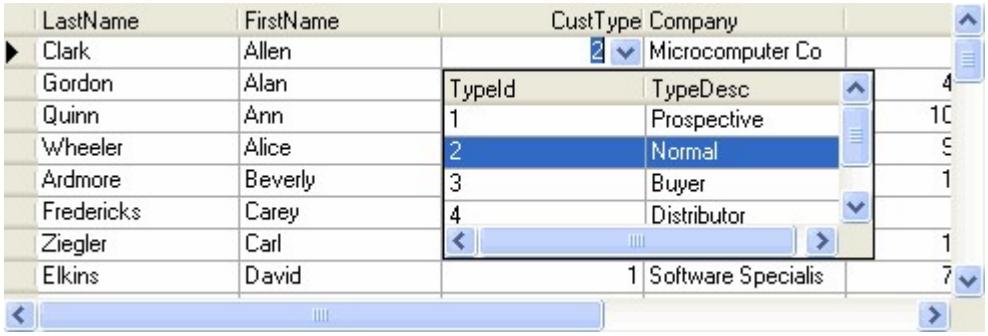
下拉组合框包含ValueItemCollection 对象中每个成员的项，如果集合的Translate 属性被设置为True，此时DisplayValue 文本用于组合框项，如果它为False，此时Value 文本被使用。
WinForms版版True DBGrid 会自动调整下拉组合框的大小以适应显示列的宽度，组合框的高度由集合中项的数量来决定，以及MaxComboItems 属性，如果项的数量小于等于MaxComboItems，其默认值为5，此时所有的项均可以显示，如果项的数量超过了 MaxComboItems，那么仅有MaxComboItems 被显示，而滚动条会出现在组合框的右边，它可以允许用户将其他项引入视图。

监控内置组合框选项

当用户从内置组合框中选择一个项时，ComboSelect 事件会被触发，该事件对于在用户退出编辑模式前决定单元格内容非常实用。
由于在内置组合框中显示项只是基本数据源的允许值，您可能会阻止您的用户在选择后单元格中输入，通过设置C1DisplayColumn 属性DropDownList 等于True，其附加C1TrueDBDropDown 控件现在将被限制使用为一个列表框，在下拉中没有新值或改变被允许，因此基本数据库不能使用错误信息更新。

使用C1TrueDBDropDown控件

之前示例中描述的内置下拉组合框，在允许值为已知或者数值相对少的情况下非常实用，ValueItem 对象的一个集合在设计器中会被显得非常笨重，并且需要大量的代码来创建，此外，内置的组合框不能绑定数据控件和自动填充。使用本章节后面列出的技术，可以创建一个二级C1TrueDBGrid 控件用于下拉，然而，为了显示其他数据源的一列值，C1TrueDBDropDown 控件提供了一个更美观的解决方案，它可以在设计时明确的被设计并且完全被创建。



为了使用下拉控件，设置网格列的DropDown 属性为设计器或者代码中的C1TrueDBDropDown 控件，在运行时，当用户点击了列的单元格内按钮，C1TrueDBDropDown 控件将出现网格当前单元格的下方，如果用户从下拉控件中选择了一个项，网格的当前单元格会被更新。
由于C1TrueDBDropDown 控件是C1TrueDBGrid的子集，它拥有很多相同的属性，方法和事件。然而以下两个属性指定给C1TrueDBDropDown 控件：

Property	Description
ValueMember	该属性指定的下拉列可以用于更新关联的网格列，当一个选择被确定时。

DisplayMember	该属性指定了下拉列的属性，用于渐进式搜索。

当一个C1TrueDBDropDown 控件变为可见，它的DropDownOpen 事件会触发，类似的当用户做出一个选择或者控件不在聚焦时，它的DropDownClose 事件会被触发。

C1TrueDBDropDown的自动数据转换

假设一个网格下拉框需要使用包含值和它相应的文本表达的数据，如下图：

TypeId	TypeDesc
1	Prospective
2	Normal
3	Buyer
4	Distributor
5	Other

在这种情况下，您不需要用户看到某些模糊的TypeId，而希望更多的可以理解的TypeDesc在下拉中显示，ValueTranslate 属性会自动映射TypeId值为TypeDesc表示，按照这种方法，当用户访问下拉时，它将显示TypeDesc 文本。

使用任意下拉控件

正常情况下，WinForms版版True DBGrid 的默认编辑行为对大多数应用程序是非常有效的，然而在很多情况下，您想要自定义该行为，一项有价值的技术可以使用下拉菜单或自合框或甚至是WinForms版版True DBGrid控件，允许从一列可能的值中选择，这可以轻松的在Visual Studio中使用WinForms版版True DBGrid控件或第三方控件，其一般方法如下，一个实例在教程教程9：将一个任意的下拉控件添加到网格单元格中：将一个任意的下拉控件添加到网格单元格中 (Section 12.9)。
一般情况下，不使用标准True DBGrid 编辑器显示一个下拉列表或组合框包含以下几步：

1. WinForms版版True DBGrid 在用户想要编辑一个单元格时会触发BeforeColEdit 事件，为了覆盖默认的编辑过程，通过设置Cancel 参数为True取消C1TrueDBGrid 的默认编辑器，将BeforeColEdit中中的代码显示您想要编辑的控件，通常情况下，您需要放置可替代的编辑控件或同网格一样的下拉表格，除非您需要他否则它是不可见的。
2. 当BeforeColEdit 被触发，有五种属性和一个方法用以确定单元格的准确坐标，五种属性包括Left (应用于网格和列)，Top (网格和列)，CellTop (仅适用于列，用于多行显示)，Width (仅适用于列)，和RowHeight(仅适用于网格)。方法有RowTop (仅适用于网格)，使用这些属性和方法来定位自定义编辑控件或网格单元格相关下拉，例如，将ListBox控件置于单元格右边并与顶边框对齐，需要使用以下代码：

To write code in Visual Basic

Visual Basic

```
Private Sub C1TrueDBGrid1_BeforeColEdit(ByVal sender As Object, ByVal e As
C1.Win.C1TrueDBGrid.BeforeColEditEventArgs) Handles C1TrueDBGrid1.BeforeColEdit

Dim r As Rectangle =
Me.C1TrueDBGrid1.Splits(0).GetCellBounds(Me.C1TrueDBGrid1.Row, e.ColIndex)

r = Me.C1TrueDBGrid1.RectangleToScreen(r)

r = Me.RectangleToClient(r)

Me.ListBox1.Left = r.Left
Me.ListBox1.Top = r.Bottom
End Sub
```

To write code in C#

C#

```
private void c1TrueDBGrid1_BeforeColEdit(object sender,
C1.Win.C1TrueDBGrid.BeforeColEditEventArgs e)
{

Rectangle r = this.c1TrueDBGrid1.Splits[0].GetCellBounds(this.c1TrueDBGrid1.Row, e.ColIndex); r =
this.c1TrueDBGrid1.RectangleToScreen(r); r = this.RectangleToClient(r); this.ListBox1.Left = r.Left; this.ListBox1.Top =
r.Bottom;

}
```

1. 将代码放在下拉或组合框中，通过将所选的值分配给正在编辑的列的文本或值属性完成编辑过程。

然而当网格的MarqueeStyle 属性被设置为MarqueeEnum.FloatingEditor，该方法无效，当浮动编辑器选取框被使用时，BeforeColEdit 事件不会触发直到单元格被用户改变，然而，使用内置列按钮功能激活在下一节中介绍的下拉框。
其他MarqueeStyle 设置的示例，请参阅高亮当前行或单元行高亮当前行或单元行 (Section 8.6)，从网格单元格中下拉一个Visual Basic ListBox控件的示例，如教程教程9：将任意的下拉控件添加到网格单元格中：将任意的下拉控件添加到网格单元格中 (Section 12.9)。

使用内置列按钮

另一种从一个单元格中的下拉控件的方法是使用WinForms版版True DBGrid 内置的列按钮功能，如果一个列的Button 属性被设置为True，一个按钮将在当前单元格右边显示，点击按钮触发网格的ButtonClick 事件，单元格的下拉控件可以使用ButtonClick 事件内的代码，也可以使用该事件触发行为或单元格内的计算。
更多详细信息，请参阅单元格内按钮单元格内按钮 (Section 11.5)。