TrueDBGrid数据绑定

以下主题描述如何绑定一个数据源，创建并使用非绑定列，以及不绑定数据源显示数据。 7.1 为WinForms版True DBGrid绑定一个数据源
令人惊讶的易用性，WinForms版版True DBGrid可以绑定到任意的.NET数据源中，仅需要很少的代码或者不需要代码， C1TrueDBGrid 控件可以在几秒钟内创建一个全功能导航数据库浏览器。
WinForms版版True DBGrid 完全支持数据绑定到ADO.NET对象，如DataTable, DataView 和 DataSet对象，您有一个更容易绑定ComponentOne DataObjects Express 数据源的选项，
如C1ExpressTable和C1ExpressConnection，C1TrueDBGrid 也完全支持ComponentOne Studio Enterprise中强大的WinForms版版DataObjects框架。
为了将WinForms版版True DBGrid控件关联一个ADO.NET 或WinForms版版DataObjects数据源，为相同的窗体中的网格设置DataSource 属性为一个数据集，如果DataSet 包含多张表，您可以在DataMember 属性组合框中选择一个表名，
DataSource与DataMember属性均可以通过代码被设置，通过Visual Studio的属性窗口，这需要WinForms版版True DBGrid
充分使用数据库或应用中的DataTable 。
一旦有链接存在，WinForms版版True DBGrid和DataSet 将自动通知和响应其他执行的操作，最终简化您应用程序的开发。

保存网格布局

您可以在运行时使用SetDataBinding 方法绑定网格，例如以下代码绑定C1TrueDBGrid 控件到在DSCustomers 数据源中的Customers 表格:
To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.SetDataBinding(Me.DsCustomers.Customers, "")
```

To write code in C#

```
C#
this.c1TrueDBGrid1.SetDataBinding(this.DsCustomers.Customers, "");
```

如果数据源在代码中被重置，它将显示网格中的所有数据，并且不会保持设计器初始创建的布局，您可以使用
SetDataBinding 方法保持网格设计时的布局，通过设置HoldFields 参数为True，例如
To write code in Visual Basic

```
Visual Basic
C1TrueDBGrid1.SetDataBinding(Me.DsCustomers.Customers, "", True)
```

To write code in C#

```
C#
this.c1TrueDBGrid1.SetDataBinding(this.DsCustomers.Customers, "", true);
```

另一个使用 SetDataBinding(Object, String, Boolean) 方法的示例，请参阅教程教程 2: 带有带有SQL查询结果的查询结果的WinForms版版DBGrid。

使用非绑定列

通常情况下，WinForms版版True DBGrid 会自动显示绑定数据字段的数据，然而，您可能需要扩展您布局中的字段集，列均来自于数据库字段或未关联（或仅松关联）数据库信息的列，例如如果您的数据库包含一个Balance 字段，您可能换成显示两个列Credit和Debit，以单独显示正数和负数。或者您想要在其他的数据库中查找数据，或者转换字段数据为其他的形式，如将数字转化为文本描述。
为了完成非绑定列中的任务，单词unbound column 关联一个bound grid的部分列，但它并不是直接绑定一个数据库字段。
列没有DataField 属性集(这就是说DataField 属性等于一个空字串)，但列的Caption 属性集被理解为非绑定列，网格将通过UnboundColumnFetch 事件响应这些列中的数据。带有DataField属性集的列将被绑定，如果DataField属性与数据源中的某个字段相同。
带有DataField属性集的列设置了一个非数据集中的值，该数据集被忽略了获取数据的目的，类似地，列中DataField 和Caption 属性集没有值。

创建非绑定列

第一步使用一个非绑定列创建一个列，这可以通过C1TrueDBGrid Designer 在设计器中添加一个列，在代码中，非绑定列可以使用C1DataColumnCollection的Insert 方法添加，列必须为它的Caption 属性设置一个给定的名称，在设计器中，这个过程需要使用C1TrueDBGrid Designer，在代码中，C1DataColumn对象的Caption 属性被设置，C1DataColumn 对象被加入到C1DataColumnCollection 引起C1DisplayColumn 被添加到所有拆分的C1DisplayColumnCollection 的一个响应，最新加入C1DisplayColumn 的默认可见属性将为False。
当在代码中尝试插入一个非绑定列，使用Rebind 方法确保列出现在网格中需要的位置：
To write code in Visual Basic

```
Visual Basic
```

```
Dim Col As New C1.Win.C1TrueDBGrid.C1DataColumn Dim dc As C1.Win.C1TrueDBGrid.C1DisplayColumn

With Me.C1TrueDBGrid1
.Columns.Insert(0, Col) Col.Caption = "Unbound" dc = .Splits(0).DisplayColumns.Item("Unbound")

' 移除网格最左侧的新加入的列。
.Splits(0).DisplayColumns.RemoveAt(.Splits(0).DisplayColumns.IndexOf(dc)) .Splits(0).DisplayColumns.Insert(0, dc)
dc.Visible = True .Rebind(True)
End With
```

To write code in C#

```
C#

C1.Win.C1TrueDBGrid.C1DataColumn Col = new C1.Win.C1TrueDBGrid.C1DataColumn();

C1.Win.C1TrueDBGrid.C1DisplayColumn dc;

c1TrueDBGrid1.Columns.Insert(0, Col);

Col.Caption = "Unbound";

dc = c1TrueDBGrid1.Splits[0].DisplayColumns["Unbound"];
```

// 移除网格最左侧的新加入的列。
c1TrueDBGrid1.Splits[0].DisplayColumns.RemoveAt(C1TrueDBGrid1.Splits[0].DisplayColumns.IndexOf(dc));
c1TrueDBGrid1.Splits[0].DisplayColumns.Insert(0, dc); dc.Visible = true; c1TrueDBGrid1.Rebind(true);
当网格需要显示非绑定列中的值，它将触发UnboundColumnFetch事件，该事件提供给用户行和列的索引，用以识别所请求的网格中的单元格，事件的Value属性是类型对象，默认值为Null，但可以更改为任何所需的值，并将用于填充指定行和列索引单元格的内容。
To write code in Visual Basic

```
Visual Basic

Private Sub C1TrueDBGrid1_UnboundColumnFetch(ByVal sender As Object, ByVal e As
C1.Win.C1TrueDBGrid.UnboundColumnFetchEventArgs) Handles C1TrueDBGrid1.UnboundColumnFetch
```

To write code in C#

```
C#

private void c1TrueDBGrid1_UnboundColumnFetch(object sender,

C1.Win.C1TrueDBGrid.UnboundColumnFetchEventArgs e)
```

实现多非绑定列

目前，我们的示例仅使用一个单独的非绑定列演示UnboundColumnFetch 事件，但仅使用了一个非绑定列，由于UnboundColumnFetch触发了每个非绑定列的每一行，只要一个列值某时被设置，每一列就必须确定一个合适的值，第二个UnboundColumnFetch属性Column被用于识别网格中的列，其值是必须的。
To write code in Visual Basic

```
Visual Basic
```

```
' 将用作拷贝。
Dim dtCopy As Data.DataTable

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load

dtCopy = Me.DataSet11.Tables(0).Copy() End Sub

Private Sub C1TrueDBGrid1_UnboundColumnFetch(ByVal sender As System.Object, ByVal e
As C1.Win.C1TrueDBGrid.UnboundColumnFetchEventArgs) Handles
C1TrueDBGrid1.UnboundColumnFetch
Select Case e.Column.Caption
Case "Area"

' 计算网格中"Area"列值。
e.Value = dtCopy.Rows(e.Row).Item("Length") * dtCopy.Rows(e.Row).Item("Width") Case "Perimeter"

' 计算网格中"Perimeter"的列值。
e.Value = 2 * (dtCopy.Rows(e.Row).Item("Length") + dtCopy.Rows(e.Row).Item("Width")) End Select
End Sub
```

To write code in C#

C#

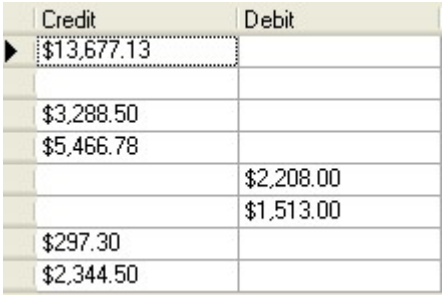
```
// 将用作拷贝。
Data.DataTable dtCopy;

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

{ dtCopy = this.DataSet11.Tables[0].Copy(); }

private void C1TrueDBGrid1_UnboundColumnFetch(object sender,
C1.Win.C1TrueDBGrid.UnboundColumnFetchEventArgs e)

{

switch (e.Column.Caption)

{ case "Area";
// 计算网格中"Area"列值。
e.value = dtCopy.Rows[e.Row].Item["Length"] * dtCopy.Rows[e.Row].Item["Width"];

break;

case "Perimeter";

// 计算网格中"Perimeter"的列值。
e.value = 2 * (dtCopy.Rows[e.Row].Item["Length"] + dtCopy.Rows[e.Row].Item["Width"]);

break;

}

}
```

更新非绑定列

在大多数情况下，非绑定列是只读的，并作为源自网格中其他数据的值，在这些情况下，设置列样式的Locked 属性为True。
若Locked 为False 且更新被允许，用户可以编辑非绑定列的值，当编辑非绑定列值时，行被标记为已更新(在记录选择器列中将显示一个铅笔图标)并且更新序列将照常执行，然而网格并不了解修改后的数据将如何处理，因此不需要数据库字段来保存它，在这种情况下，UnboundColumnUpdated 事件将被触发。
BeforeUpdate 事件将用来取消更新操作，因此，如果非绑定列与其他数据库绑定，则非绑定列的更新将在
BeforeUpdate中执行，如果操作失败，那么事件将被取消，但如果操作成功了，绑定更新将被允许继续，绑定更新也有可能在这时失败，因此非绑定列的数据库关联最好在事务基础上被处理。
如果绑定更新成功了，则AfterUpdate 事件将被触发，并且非绑定列事务应该被确认，如果绑定更新失败，非绑定列事务应将在
.NET的错误捕捉处理中回滚，依赖于更新初始化的状态，如果事务不可用，此时将之前非绑定原始列值存储到更新中，执行其他更新来存储绑定更新失败的这些值。

编辑非绑定列

更新一个非绑定列的其他方法还可以使用AfterColUpdate 事件来调整其他（绑定）列的值，例如一组列值Debit 与Credit的图像，其网格显示如下：



Credit	Debit
$13,677.13	
$3,288.50	
$5,466.78	
	$2,208.00
	$1,513.00
$297.30	
$2,344.50	

假如没有这些数据库字段，但非绑定列的值来源于单个的Balance列，它可能是正的也有可能是负的，从用户的角度来看，希望它可以自动更新网格的Balance列。
WinForms版版True DBGrid可以这件任务更加容易实现，通过以下代码放到网格的AfterColUpdate事件中，当Balance 列被更新时也会引发列的改变：
To write code in Visual Basic

```
Visual Basic
```

```
Private Sub C1TrueDBGrid1_AfterColUpdate(ByVal sender As Object, ByVal e As

C1.Win.C1TrueDBGrid.ColEventArgs) Handles C1TrueDBGrid1.AfterColUpdate
Dim row as Integer = Me.C1TrueDBGrid1.Row
Me.C1TrueDBGrid1(row, "Balance") = -e.Column.DataColumn.Value
End Sub
```

To write code in C#

```
C#
```

```
private void C1TrueDBGrid1_AfterColUpdate(object sender, C1.Win.C1TrueDBGrid.ColEventArgs e)

{

int row = this.c1TrueDBGrid1.Row;

this.c1TrueDBGrid1[row, "Balance"] = -e.Column.DataColumn.Value;

}
```

注意，当更新这些列时，代码事实上改变了Balance 列中的值，包括绑定的和不可见的值。

创建一个非绑定网格

WinForms版版True DBGrid 允许不用绑定一个数据源就可以显示数据，创建一个非绑定网格需要完成以下几个步骤。
创建一个非绑定网格，需要完成以下步骤：

1. 开始创建您的列，这个过程可以在设计器或者代码中完成，更多关于创建列的详细信息，请参阅创建非绑定列。创建非绑定列。

(Section 7.3.1)
To write code in Visual Basic

```
Visual Basic
```

```
Me.C1TrueDBGrid1.Columns.Add(New C1.Win.C1TrueDBGrid.C1DataColumn("FirstName",
```

```
GetType(String)))
Me.C1TrueDBGrid1.Columns.Add(New C1.Win.C1TrueDBGrid.C1DataColumn("LastName",
GetType(String)))
Me.C1TrueDBGrid1.Columns.Add(New C1.Win.C1TrueDbGrid.C1DataColumn("DateOfBirth",
GetType(DateTime)))
```

To write code in C#

```
C#
```

```
this.c1TrueDBGrid1.Columns.Add(new
C1.Win.C1TrueDBGrid.C1DataColumn("FirstName",typeof(string))); this.c1TrueDBGrid1.Columns.Add(new

C1.Win.C1TrueDBGrid.C1DataColumn("LateName",typeof(string))); this.c1TrueDBGrid1.Columns.Add(new

C1.Win.C1TrueDBGrid.C1DataColumn("DateOfBirth",typeof(DateTime)));
```

1. 调用非参数的SetDataBinding方法。

To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.SetDataBinding()
```

To write code in C#

```
C#
this.c1TrueDBGrid1.SetDataBinding();
```

1. 使用AddRow或AddRows 方法填充网格。

To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.AddRow("John;Doe;11/29/1985")
Me.C1TrueDBGrid1.AddRow("Jane;Doe;7/12/1980")

Dim index As Integer = Me.C1TrueDBGrid1.AddRows(2)
Dim i As Integer
For i = index To 1
Me.C1TrueDBGrid1(i, "FirstName") = "Joe"
Me.C1TrueDBGrid1(i, "LastName") = "Doe"
Me.C1TrueDBGrid1(i, "DateOfBirth") = New DateTime(2000, 1, 15)

Next i
```

To write code in C#

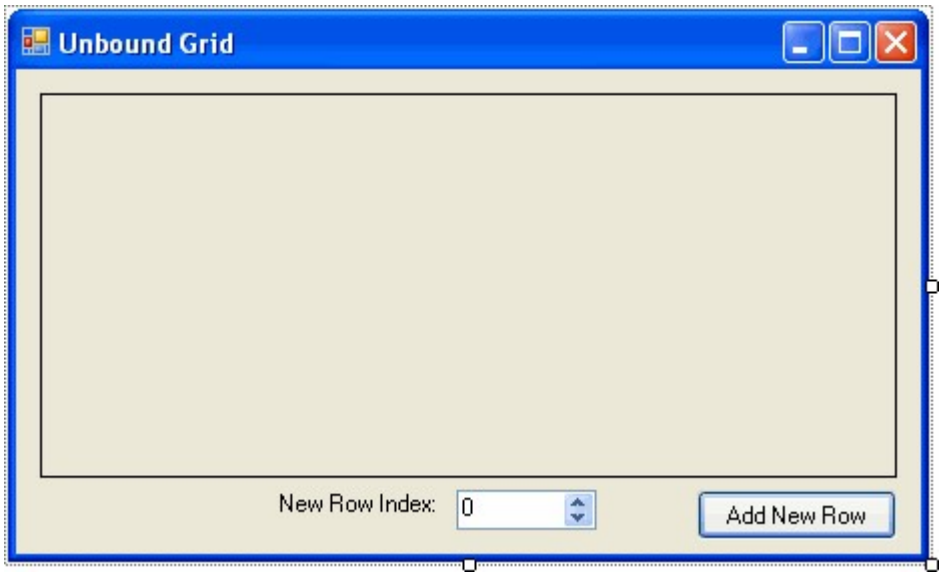
```
C#
this.c1TrueDBGrid1.AddRow("John;Doe;11/29/1985"); this.c1TrueDBGrid1.AddRow("Jane;Doe;7/12/1980"); int index =
this.c1TrueDBGrid1.AddRows(2); for(int i=index; i < 2; i++)
```

{ this.c1TrueDBGrid1[i,"FirstName"] = "Joe"; this.c1TrueDBGrid1[i, "LastName"] = "Doe"; this.c1TrueDBGrid1[i, "DateOfBirth"] = new DateTime(2000,1, 15); }
您已经成功创建了一个非绑定网格。

添加新行到非绑定网格

通过使用C1TrueDBGrid.NewRow 方法创建一个非绑定网格相同模式的新System.Data.DataRow，可以轻松的为非绑定网格添加新的行，在以下步骤中您将使用C1TrueDBGrid.Rows集合获取非绑定网格的DataRowCollection，并且使用C1TrueDBGrid.NewRow 方法插入一个新的行到非绑定网格的指定索引。完成以下步骤：

1. 创建一个新的.NET工程。
2. 导航到工具栏，并添加C1TrueDBGrid, Label, NumericUpDown及Button 控件到窗体中。
3. 设置Button1.Text 属性为"Add New Row"并将Label1.Text 属性为"New Row Index"。
4. 窗体中布置的控件将与下图类似：



1. 切换到代码视图并添加以下导入（或使用）声明到工程中：

To write code in Visual Basic

```
Visual Basic
imports C1.Win.C1TrueDBGrid
```

To write code in C#

```
C#
using C1.Win.C1TrueDBGrid;
```

1. 添加以下代码来创建Form_Load 事件并添加数据到网格中：

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
' 添加一个标题到网格中。
Me.C1TrueDBGrid1.Caption = "Unbound Grid"
' 添加列到网格中。
Me.C1TrueDBGrid1.Columns.Add(New C1.Win.C1TrueDBGrid.C1DataColumn("Col 1", GetType(String)))
Me.C1TrueDBGrid1.Columns.Add(New C1.Win.C1TrueDBGrid.C1DataColumn("Col 2", GetType(String)))
Me.C1TrueDBGrid1.Columns.Add(New C1.Win.C1TrueDBGrid.C1DataColumn("Col 3", GetType(String)))
Me.C1TrueDBGrid1.Columns.Add(New C1.Win.C1TrueDBGrid.C1DataColumn("Col 4",
GetType(String)))
' 调用无参的SetDataBinding方法。
Me.C1TrueDBGrid1.SetDataBinding()
' 填充网格。
Dim i As Integer
For i = 0 To 20 - 1
Dim s As String = String.Format("Data {0};Data {1};Data {2}; Data {3}",
New Object() {i, i, i, i})
Me.C1TrueDBGrid1.AddRow(s)
Next i
End Sub
```

To write code in C#

```
C#
```

```
private void Form1_Load(object sender, EventArgs e)
{
// 添加一个标题到网格中。
this.c1TrueDBGrid1.Caption = "Unbound Grid"

// 添加列到网格中。
this.c1TrueDBGrid1.Columns.Add(new C1.Win.C1TrueDBGrid.C1DataColumn("Col 1", typeof(string)));
this.c1TrueDBGrid1.Columns.Add(new C1.Win.C1TrueDBGrid.C1DataColumn("Col 2", typeof(string)));
this.c1TrueDBGrid1.Columns.Add(new C1.Win.C1TrueDBGrid.C1DataColumn("Col 3", typeof(string)));
this.c1TrueDBGrid1.Columns.Add(new C1.Win.C1TrueDBGrid.C1DataColumn("Col 4", typeof(string)));

// 调用无参的SetDataBinding方法。
this.c1TrueDBGrid1.SetDataBinding();

// 填充网格。
for (int i = 0; i < 20; i++)
```

```
{ string s = String.Format("Data {0};Data {1};Data {2}; Data {3}", i, i, i, i); this.c1TrueDBGrid1.AddRow(s); }
}
```

7. 添加以下代码创建Button_Click 事件并在按钮点击时在指定索引中创建一个新的行：

To write code in Visual Basic

Visual Basic

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As

System.EventArgs) Handles Button1.Click
Dim idx As Integer = CInt(Me.NumericUpDown1.Value)
' 创建一个新行。
Dim dr As DataRow = Me.C1TrueDBGrid1.NewRow dr.Item(0) = "new row"
' 在所选择的检索中添加新的行。
Me.C1TrueDBGrid1.Rows.InsertAt(dr, idx)
End Sub
```

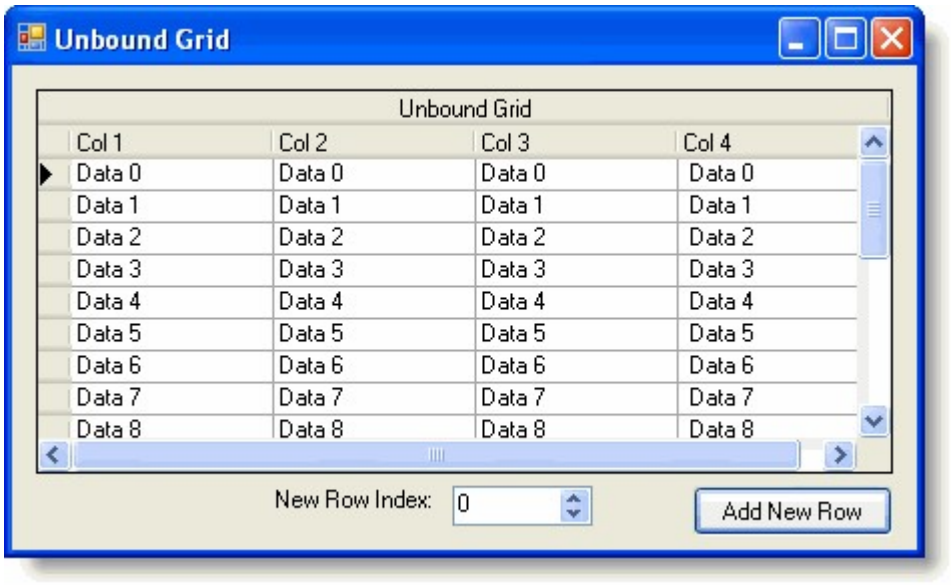
To write code in C#

C#

```
private void button1_Click(object sender, EventArgs e) { int idx = (int) this.numericUpDown1.Value;

// 创建一个新行。

DataRow dr = this.c1TrueDBGrid1.NewRow();

dr[0] = "new row";
// 在选择的索引中添加新行。
this.c1TrueDBGrid1.Rows.InsertAt(dr, idx);
}
```

运行程序并观察：

出现的窗体与下图相类似：



在New Row Index 框中使用箭头更改数值，此时选择Add New Row 按钮，在您选择的索引中将出现新的箭头。