WinForms版True DBGrid教程

本章节将详细介绍二十二个教程。这些教程假定你熟悉在Visual
Studio中编程，知道数据集是什么，并且知道如何使用绑定的控件。这些教程提供了逐步介绍-不需要提前了解WinForms版版True DBGrid的知识。通过此章节中以下步骤的大
纲，你将可以创建工程，来展示WinForms版版True DBGrid的各种特征，并且对WinForms版版True DBGrid能做什么和如何做得到一个好的了解
。
此教程使用一个数据库 C1NWind.mdb。这个数据库文件C1NWind.mdb在ComponentOne Samples 文件夹
的Common 子目录中，并且此教程中的工程在Documents\ComponentOne Samples\WinForms\C1TrueDBGrid目录下的Tutorials文件夹。尽管此教程编了号，你没有必要按顺序完成它们；此教程的编号引用了Tutorials 文件夹中的文件。
我们鼓励你在Visual Studio中运行该教程中的工程，检查代码，并且你自己可以修改代码进行实验。这是最好的而且是最快的方法来发掘True DBGrid的潜能。你将发现True DBGrid很容易使用，并且它能使你创建强大的数据库应用。此教程假定数据库在Documents\ComponentOne Samples\Common目录的目录的C1NWind.mdb 数据库文件中，并且为了暂时的安全，通过文件名引用它来代替全部路径名。

注意注意: 依赖你所存储的工程和数据库文件，你可能需要改变在数据集中C1NWind.mdb 引用的路径。

下面的教程中创建的工程，有些会被其他的教程使用；例如，教程1, 3, 6, 7, 8, 10创建的工程被使用在其他的教程中。
因为这些工程创建在一些教程中，且被多个教程重新使用，建议你保存你的文件在完成每一个教程之后。

绑定True DBGrid到数据集

在这个教程中，你将学到如何绑定WinForms版版True DBGrid 控件到数据集上，并且创建功能完整的数据库浏览器。 你也可以学到WinForms版版True DBGrid的基础属性，紧接着可以运行程序并观察网格的运行时特征。

注意注意:  在ComponentOne Videos[	http://helpcentral.componentone.com/Videos.aspx]网页上的教程中的视频可用。

完成下面的步骤:

1. 创建一个新的.NET 工程。
2. 打开工具框，该工具框初始位置在IDE的左边，它有一个锤子和一个扳手作为它的图标。从这个工具框中，定位

并双击 C1TrueDBGrid 图标

。
一个网格控件被增加到了窗体上，并且出现了C1TrueDBGrid 任务（任务（C1TrueDBGrid Tasks））菜单。

1. 在这个菜单中，选择选择数据源（Choose Data Source））下拉框箭头并且单击添加工程数据源（Add Project

Data Source）。）。

1. 出现数据源配置向导（Data Source Configuration Wizard）） 并且数据库（Database））被选中。单击下一步

（Next））.

1. 单击新的连接（New Connection））按钮以定位并连接数据库。
2. 单击浏览（Browse）） 按钮并在Documents\ComponentOne Samples\Common 目录中定位C1NWind.mdb文件。选择它并单击打开（Open））.
3. 单击测试连接（Test Connection））按钮以确认你已经成功连接到了数据库或者服务器，并单击确定（OK）。）。新的字符串出现在数据连接的下拉列表中。
4. 单击下一步（Next））按钮并继续。一个对话框出现并询问你是否想要在你的工程中添加一个数据文件并修改连接字符串。单击取消（No）。）。
5. 在下个窗口中，Yes, save the connection as复选框被默认选中，并且一个名称为

("TDBGDemoConnectionString")的字符串已经被自动键入到文本框中。单击下一步（Next ）并继续。

1. 在选择你的数据库对象（Choose Your Database Objects））窗口中，在你的数据库中选择你想要的表和字段。

在数据集名称（DataSet name））文本框中此数据集被赋予了一个默认的名称("TDBGDemoDataSet")。

1. 单击完成（Finish ）退出向导. 此数据集（DataSet））, 绑定源（BindingSource ）和表适配器（TableAdapter））目前已经出现在你的窗体上。
2. 改变网格的大小并且双击此窗体。注意Visual Studio已经添加了如下的代码到Form_Load 事件上：

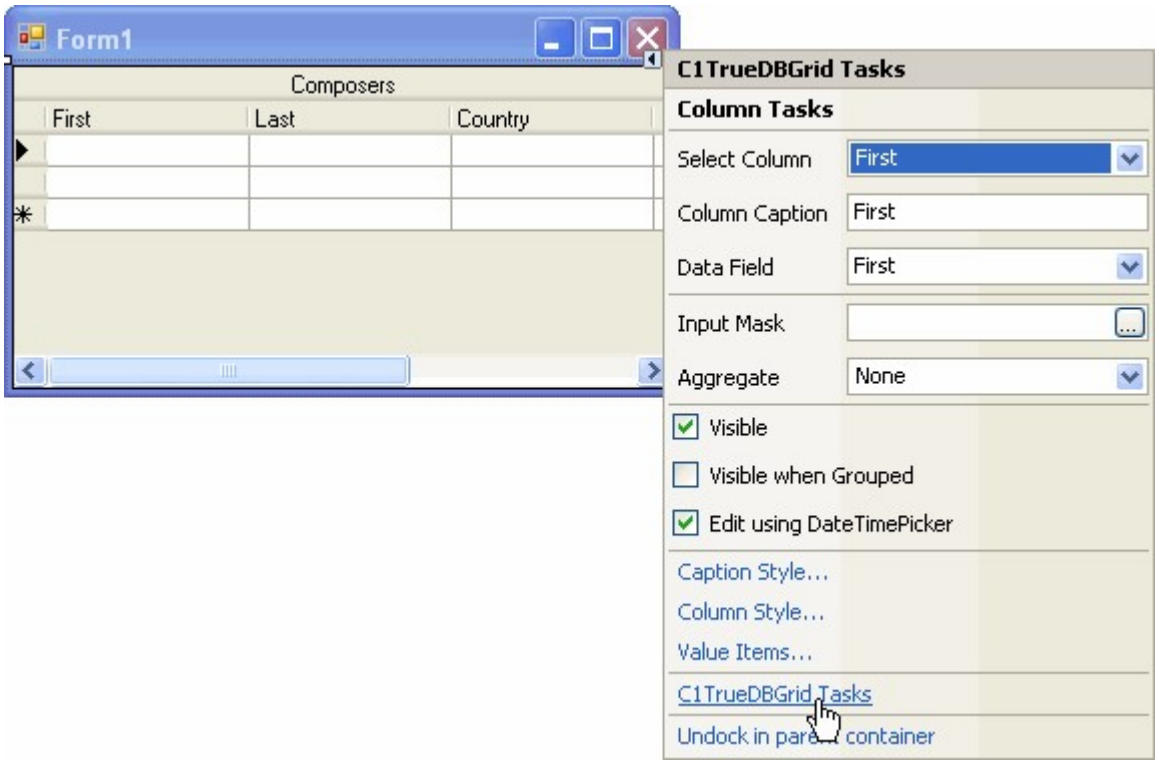
To write code in Visual Basic

Visual Basic
Me.ComposerTableAdapter.Fill(Me.DsComposer.Composer)

To write code in C#

C#
this.ComposerTableAdapter.Fill(this.DsComposer.Composer);

1. 单击设计（Design））Tab返回到的设计器并且选择网格。
2. 打开 C1TrueDBGrid 任务任务（C1TrueDBGrid Tasks））菜单并且选择C1TrueDBGrid 任务任务（C1TrueDBGrid Tasks））.



1. 选中Enable Adding和Enable Deleting复选框。这样会设置C1TrueDBGrid1 的AllowAddNew和AllowDelete属性为True，，允许用户添加或者删除网格上的记录。

运行此程序并观察下列各项:
True DBGrid 从数据集中检索数据库结构信息并自动配置自己以显示包含在数据库表中的所有字段。 注意字段名被作为默认的列标题使用。
True DBGrid 自动与数据集进行通信。任何在数据集上的行为标记将被映射到网格中。
一个功能完整的数据库浏览器只需要写的四行简单代码就可创建。
参考Run-Time Interaction 在运行时试用指令导航，编辑，配置网格。
在程序的末尾，关闭窗口或者 按下Visual Basic Toolbar上的停止按钮。
恭喜你，你已经完成了绑定True DBGrid到数据集上!

教程2:为WinForms和SQL查询结果使用TureDBGrid

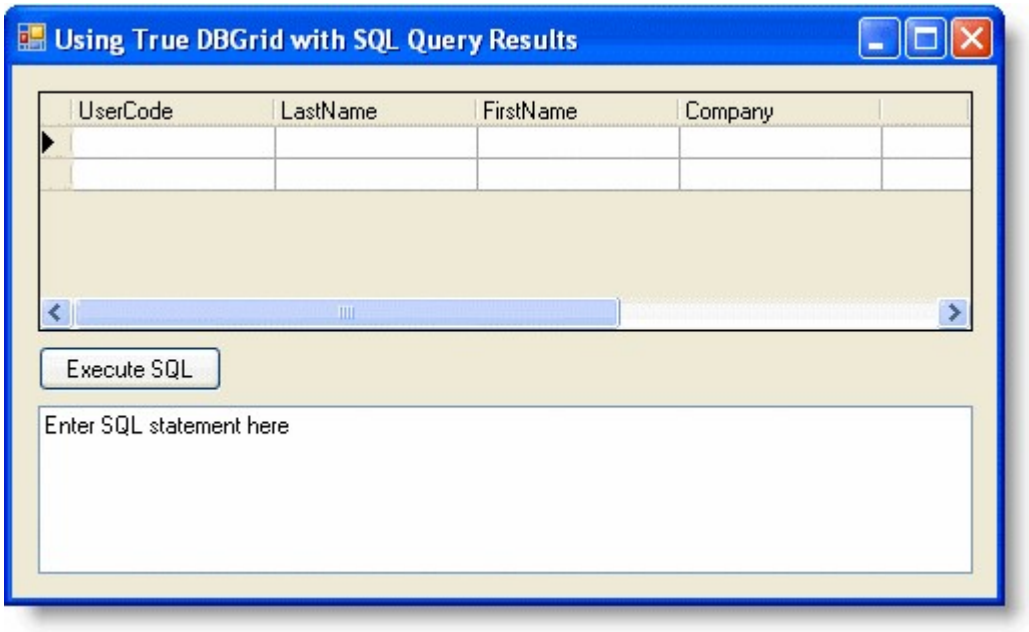
WinForms版版True DBGrid 的一个重要特性是在运行时它能自动感知并改变数据库。 在这个教程中，你将学习到如何使用 WinForms版版True DBGrid 以显示ad-hoc SQL 查询的结果。
另外，它也将强调如何设置一个连接在运行时的数据库上。注意网格的顺序以自动响应字段布局的更改。你不必在设计时定义任何列属性。如果一个布局已经被定义，使用网格的Clear Fields 上下文菜单命令去删除它。这样将引起网格在运行时自动配置自己。

Note: 在ComponentOne Videos[	http://helpcentral.componentone.com/Videos.aspx]网页上的教程中的视频可用。

完成下面的步骤:

1. 创建一个新的.NET工程.
2. 放置一个 C1TrueDBGrid 控件(C1TrueDBGrid1), 一个Button (Button1), 和一个TextBox 控件(TextBox1) 到表单上.

设置命令行按钮的Text属性为"Execute SQL"、设置TextBox1 的Text 属性为"Enter SQL statement here":



1. 跳转到 数据源（DataSource））属性并从下拉框中选择添加工程数据源（Add Project Data Source）。）。在适配器的数据源配置向导（Data Source Configuration Wizard））， 或者选择一个到C1NWind.mdb的连接，或者创建一个新的连接到数据库。在向导的Choose your database objects页面上，选择在Customers 表和类型"DsCustomers"的所有字段到数据集名称（DataSet name））框中，并完成该向导。
2. Visual Studio将增加下面的代码Form_Load 事件:

To write code in Visual Basic

```
Visual Basic
Me.CustomersTableAdapter.Fill(Me.DsCustomers.Customers)
```

To write code in C#

```
C#
this.CustomersTableAdapter.Fill(this.DsCustomers.Customers);
```

1. 增加下面的代码为Button1添加Click事件 ：

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 sqlStr As String = TextBox1.Text
Dim da as Oledb.OleDbDataAdapter = New Oledb.OleDbDataAdapter (sqlStr,

Me.CustomersTableAdapter.Connection)
Dim ds As DataSet = New DataSet()
ds.Clear() Try
da.Fill(ds, "mySQL") Me.C1TrueDBGrid1.DataSource = Nothing
Me.C1TrueDBGrid1.ClearFields()

Me.C1TrueDBGrid1.SetDataBinding(ds.Tables("mySQL"), "", False)
Catch
MessageBox.Show("Error in SQL clause")
End Try
End Sub
```

To write code in C#

```
C#
```

```
private void button1_Click(System.object sender, System.EventArgs e)
{ string sqlStr = TextBox1.Text;
da as Oledb.OleDbDataAdapter = New Oledb.OleDbDataAdapter (sqlStr, this.CustomersTableAdapter.Connection);
DataSet DataSet ds = new DataSet();
ds.Clear();
try {
da.Fill(ds, "mySQL");
this.c1TrueDBGrid1.DataSource = null;
this.c1TrueDBGrid1.ClearFields();
this.c1TrueDBGrid1.SetDataBinding(ds.Tables["mySQL"], "", false); }
catch () {
MessageBox.Show ("Error in SQL clause");
}
}
```

运行此程序并观察下列各项:
在教程教程1: 绑定绑定True DBGrid到数据集到数据集 (Section 12.1)中, True DBGrid for WinForms 从数据集中检索数据库结构信息并自动配置自己以显示包含在数据库表中的所有字段。 注意字段名被作为默认的列标题使用。

1. a. 在TextBox 控件上, 键入下面的SQL 语句:

Select * from Customer
按下Execute SQL 按钮. 上面的SQL 语句显示了从Customer 表中的所有字段，并相当于默认的显示。

1. a. 在TextBox 控件上,键入下面的SQL 语句:

Select Company from Customer
按下 Execute SQL 按钮。此网格为Company 字段只显示一列作出了响应。

1. a. 在TextBox 控件上, 键入下面的SQL 语句:

Select LastName, Company from Customer 按下Execute SQL 按钮. 此语句与之前的SQL 语句相似，并期待两列(LastName and Company)被显示.

1. a. 在TextBox 控件上, 键入下面的SQL 语句:

Select Count⭐ from Customer
按下Execute SQL 按钮. 上面的SQL 语句使用了聚合函数 Count⭐, 返回了在Customer 表中记录的总共数字. 即使此SQL结果不是记录的集合，但是网格在单列中如实地响应并显示了记录总数。默认Expr1000被作为列头使用，表明表达式结果的展示。

1. a. 在TextBox 控件上, 键入下面的SQL 语句:

Select UCase(LastName) as ULAST, UCase(FirstName) AS UFIRST from Customer
按下Execute SQL 按钮. 上面的SQL 语句产生出两个计算的列，LastName 和FirstName 并以大写字母显示。网格也显示（分配的）计算的列名称ULAST 和UFIRST，作为列标头。

1. a. 在TextBox 控件上,键入下面的SQL 语句:

SELECT * FROM Customer WHERE FirstName = "Jerry" 按下Execute SQL 按钮. 上面的SQL 语句只显示记录FirstName 为Jerry的值.

1. a. 在TextBox 控件上, 键入下面的SQL 语句:

SELECT * FROM Customer ORDER BY LastName 按下Execute SQL 按钮. 上面的SQL 语句根据LastName 字段以字母顺序显示记录。
你可以使用一条SQL 语句连接两个数据库表，正如在Tutorial 3: Linking Multiple True DBGrid Controls (Section 12.3)中演示.
结论教程2; 你已经完成使用了True DBGrid 和SQL查询结果.

教程3：连接多个True DBGrid控件

该教程演示了如何使用Master Detail 数据集链接多个WinForms版版True DBGrid控件。完成下面的步骤:

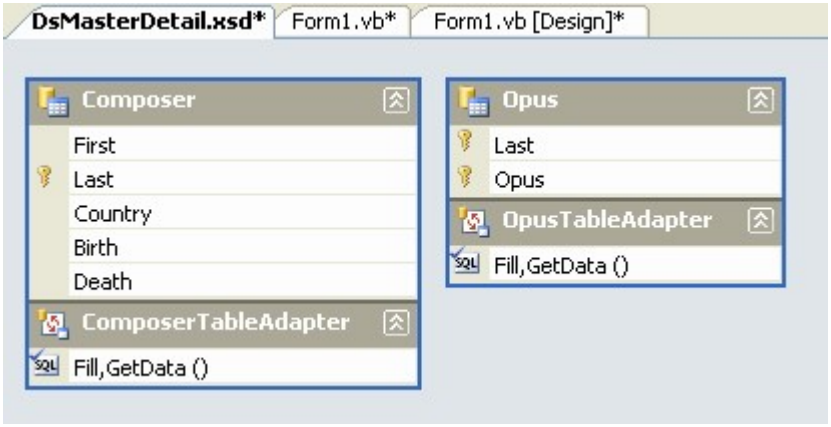
1. 创建一个新的.NET工程.
2. 导航到Visual Studio工具框并双击C1TrueDBGrid项，添加两个C1TrueDBGrid 控件到窗体上

(C1TrueDBGrid1和C1TrueDBGrid2).

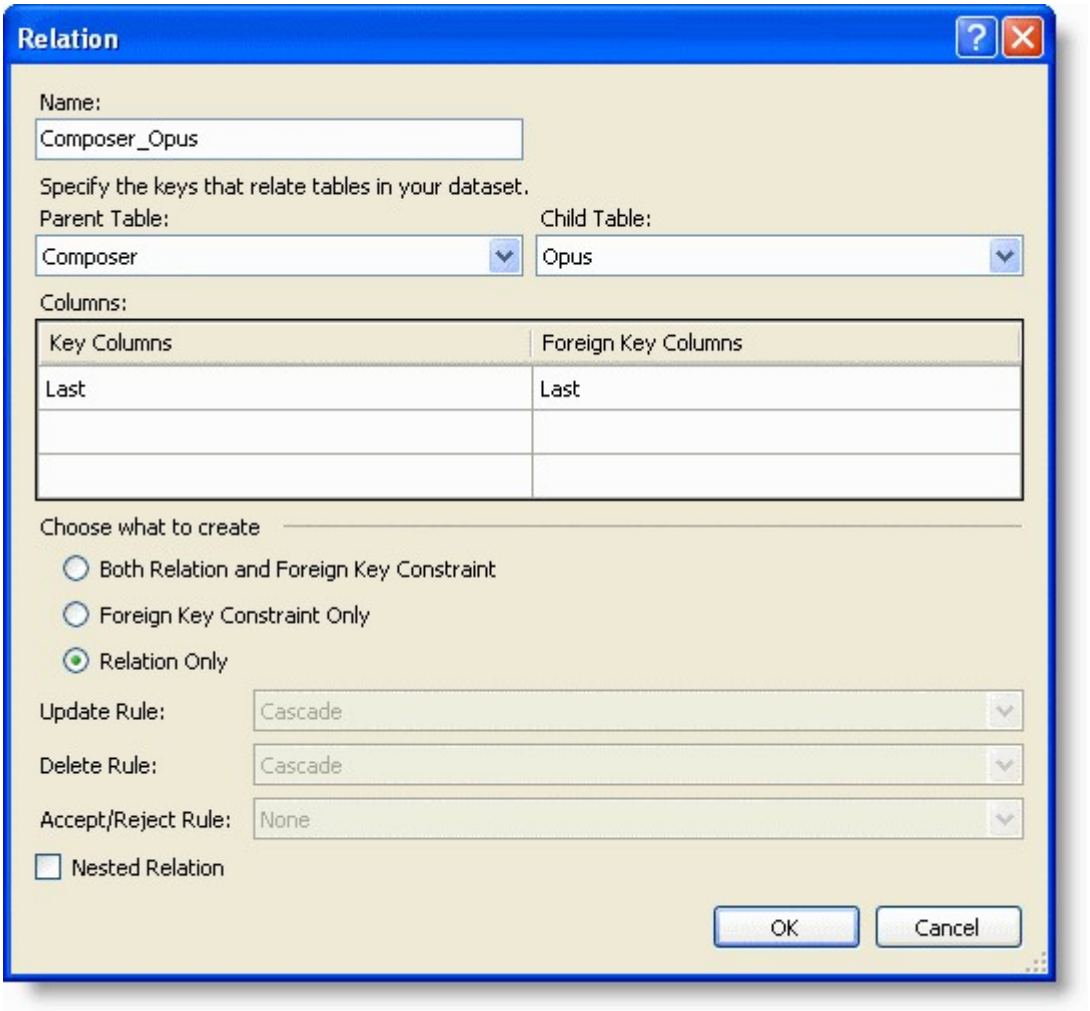
1. 在 C1TrueDBGrid1 控件的C1TrueDBGrid Tasks 菜单中，鼠标放置 Choose Data Source下拉框上并选择Add

Project Data Source.

1. 在适配器的Data Source Configuration Wizard中，选择一个连接到C1NWind.mdb或者创建一个新的连接到数据库中。在向导的Choose your database objects 页面，选择Composer 表中的所有字段和Opus 表中的所有字段并在DataSet name 框中键入"DsMasterDetail"，然后完成向导。
2. 在Solution Explorer 窗口中双击 DsMasterDetail.xsd。将打开将打开DsMasterDetail.xds文件并同下面一样:

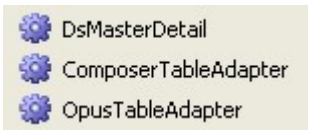


1. 将两个表做关联，在Composer中紧挨着Last字段的区域按下鼠标。然后在Composers 表格上拖拽鼠标到 Opus 表格上，并在紧挨着Last 字段的区域释放鼠标。将产生Relation 对话框。

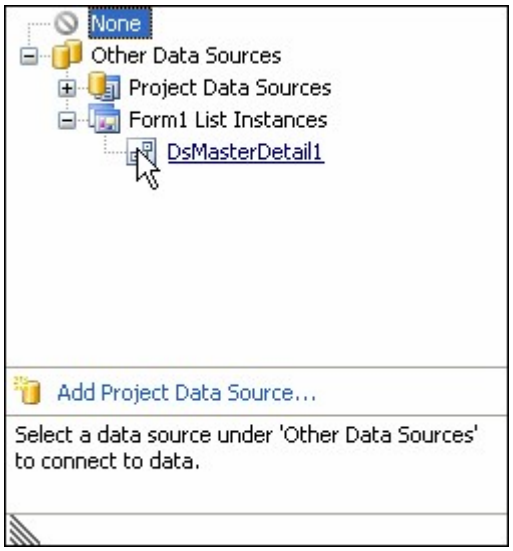


确认Parent Table被设置为Composer并且Child Table被设置为Opus。另外，确认两个字段被设置为Last列和Relation Only被选中(正如在先前的截图). 单击OK并退出Edit Relation 对话框。

1. 现在跳转到Visual Studio 中的 Build 菜单并选择 Build Solution。这样将确认这个新关系在工程中可用。
2. 返回到窗体的设计视图，并在工具框中放置 <Your Project Name> Components 标签。添加DsMasterDetail的一个实例, ComposerTable和OpusTableAdapter到表单中。

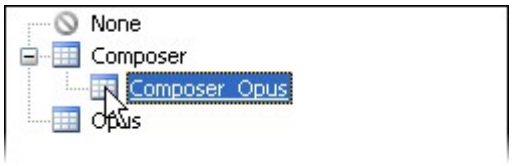


1. 现在在属性窗口，为第一个C1TrueDBGrid控件设置DataSource 属性为DsMasterDetail1 并且设置DataMember 属性为 Composer.



如果提示要替换列外观，单击Yes.

1. 对第二个C1TrueDBGrid 控件，设置DataSource 属性为DsMasterDetail1 并且设置DataMember 属性为Composer.Composer_Opus.



如果提示要替换列外观，单击Yes.

1. 所有这些被留下是为了计算DataAdapters.
2. 双击窗体转换代码视图并且创建Form_Load事件处理器。为Form1的 Load 事件添加下面的代码:

To write code in Visual Basic

```
Visual Basic
Me.ComposerTableAdapter1.Fill(Me.DsMasterDetail1.Composer)

Me.OpusTableAdapter1.Fill(Me.DsMasterDetail1.Opus)
```

To write code in C#

```
C#
this.composerTableAdapter1.Fill(this.dsMasterDetail1.Composer); this.opusTableAdapter1.Fill(this.dsMasterDetail1.Opus);
```

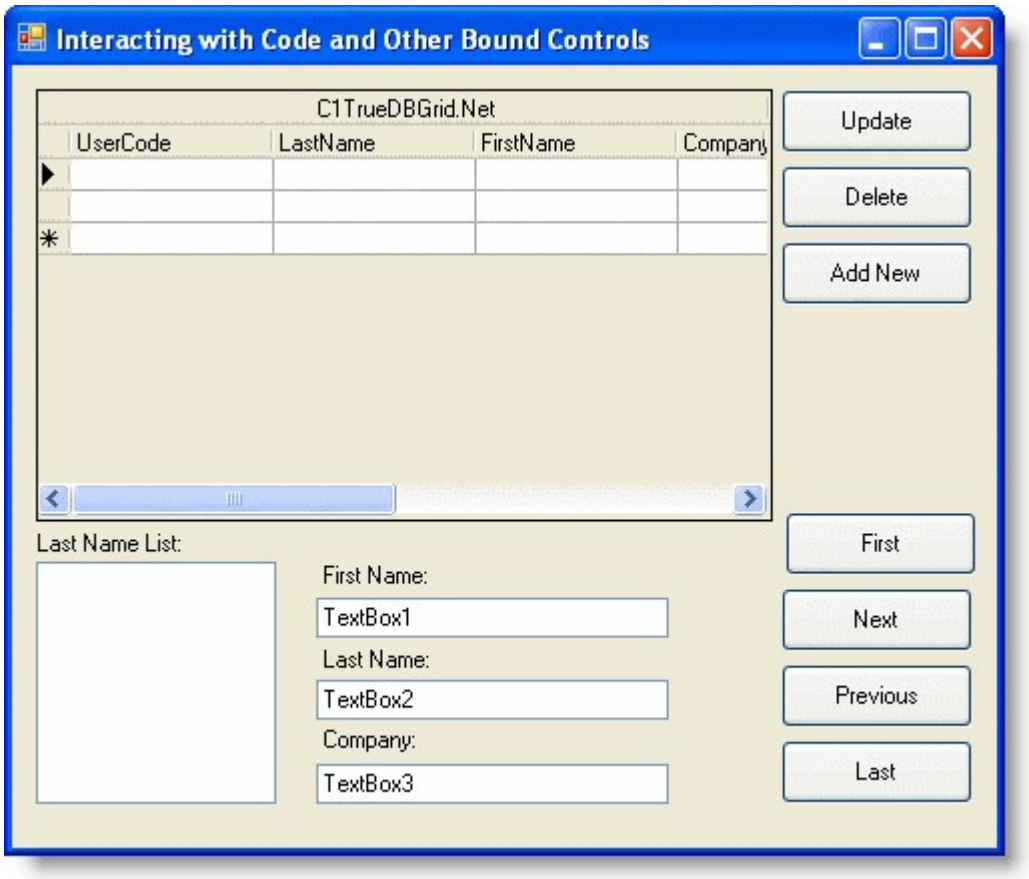
运行此程序并观察下列各项:
当Form1加载成功，C1TrueDBGrid1和C1TrueDBGrid2从DsMasterDetail1检索数据库结构信息 :
!worddav682d7feef9bf86f05db06671fc7b1b31.png|height=316,width=466!通过点击不同的行改变第一个网格的当前记录位置。观察C1TrueDB Grid2 (detail 网格) 将配置自己以在
C1TrueDBGrid1 (the master grid)中每当行变化时显示一个新的记录集合。
总结此教程，你已经成功地完成了连接多个True DBGrid 控件.

教程4：代码和其他绑定控件的相互作用

在此教程中， 你将学习到如何使True DBGrid与其他的绑定控件相互作用，并使用代码操纵相同的数据集到绑定的网格中。完成下面的步骤:

1. 创建新的.NET 工程.
2. 放置下面的控件到窗体(Form1)， 正如下面的图形中显示的一样:

C1TrueDBGrid 控件(C1TrueDBGrid1)
列表框控件(ListBox1) 三个文本控件(TextBox1 到 3)
七个按钮(命名为 btnFirst, btnNext, btnPrevious, btnLast, btnDelete, btnUpdate, btnAdd) 四个标签行(Label1 到 4)
如下所示，为每一个标签和按钮设置Text 属性:



1. 在 C1TrueDBGrid Tasks菜单中，放置Choose Data Source下拉列表并选择Add Project Data Source。 在适配器的Data Source Configuration Wizard，，或者选择一个连接到C1NWind.mdb或者创建一个新的连接到此数据库。在向导的Choose your database objects页面，选择Customers 表的所有字段并键入"DsCustomers"到DataSet name框，然后完成了此向导。
2. Visual Studio添加下面的代码到Form_Load 事件:

To write code in Visual Basic

```
Visual Basic
Me.CustomersTableAdapter.Fill(Me.DsCustomers.Customers)
```

To write code in C#

```
C#
this.CustomersTableAdapter.Fill(this.DsCustomers.Customers);
```

1. 现在，为上面图片中右下角的四个按钮(Button4, 5, 6, 7) 添加下面的代码：

To write code in Visual Basic

```
Visual Basic
Private Sub btnFirst_Click(ByVal sender As System.Object, ByVal e As

System.EventArgs) Handles btnFirst.Click

' Move to the first record in the grid.
Me.C1TrueDBGrid1.MoveFirst()
End Sub


Private Sub btnNext_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnNext.Click
' Move to the next record in the grid.
Me.C1TrueDBGrid1.MoveNext()
End Sub
Private Sub btnPrevious_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnPrevious.Click
' Move to the previous record in the grid.
Me.C1TrueDBGrid1.MovePrevious()
```

```
End Sub
Private Sub btnLast_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnLast.Click
' Move to the last record in the grid.
Me.C1TrueDBGrid1.MoveLast()
End Sub
```

To write code in C#

C#

```
private void btnFirst_Click(System.object sender, System.EventArgs e) {
// Move to the first record in the grid.
this.c1TrueDBGrid1.MoveFirst(); } private void btnNext_Click(System.object sender, System.EventArgs e) {
// Move to the next record in the grid.
this.c1TrueDBGrid1.MoveNext(); } private void btnPrevious_Click(System.object sender, System.EventArgs e) {

// Move to the previous record in the grid.
this.c1TrueDBGrid1.MovePrevious(); } private void btnLast_Click(System.object sender, System.EventArgs e) {

// Move to the last record in the grid.
this.c1TrueDBGrid1.MoveLast();
}
```

1. 为上面图片中右上角的三个按钮设置如下代码：

To write code in Visual Basic

Visual Basic

```
Private Sub btnDelete_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnDelete.Click

' Delete the record and save the changes to the database.
Me.C1TrueDBGrid1.Delete()
Call UpdateCustomers()
End Sub

Private Sub BtnUpdate_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnUpdate.Click

' Update the grid, call UpdateCustomers to save.
Me.C1TrueDBGrid1.UpdateData()
Call UpdateCustomers()
End Sub

Private Sub BtnAdd_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnAdd.Click

' This "Add New" button moves the cursor to the "new (blank) row" at the end so that user can start adding data to the new
record.
' Move to last record, "new row" will be visible.
Me.C1TrueDBGrid1.MoveLast()

' Move to the "addnew row", and set focus to the grid.
Me.C1TrueDBGrid1.Row = Me.C1TrueDBGrid1.Row + 1
Me.C1TrueDBGrid1.Select()
End Sub
```

To write code in C#

C#

```
private void btnDelete_Click(System.object sender, System.EventArgs e) {
// Delete the record and save the changes to the database. this.c1TrueDBGrid1.Delete(); UpdateCustomers();
} private void BtnUpdate_Click(System.object sender, System.EventArgs e) {
// Update the grid, call UpdateCustomers to save. this.c1TrueDBGrid1.UpdateData(); UpdateCustomers();
} private void BtnAdd_Click(System.object sender, System.EventArgs e) {
// This "Add new" button moves the cursor to the "new (blank) row" at the
```

```
end so that user can start adding data to the new record.
// Move to last record, "new row" will be visible. this.c1TrueDBGrid1.MoveLast();
// Move to the "addnew row", and set focus to the grid. this.c1TrueDBGrid1.Row = this.c1TrueDBGrid1.Row + 1;
this.c1TrueDBGrid1.Select();
}
```

1. 现在添加UpdateCustomers子程序。 这样将从暂时的数据表（DataTable）中返回数据集（DataSet）信息：

To write code in Visual Basic

Visual Basic

```
Private Sub UpdateCustomers()
Me.CustomersTableAdapter.Connection.Open()
Dim UpdatedRows As System.Data.DataSet
Dim InsertedRows As System.Data.DataSet
Dim DeletedRows As System.Data.DataSet

’ These Data Tables hold any changes that have been made to the dataset since the last update. UpdatedRows =
Me.DsCustomers.GetChanges(DataRowState.Modified)
InsertedRows = Me.DsCustomers.GetChanges(DataRowState.Added) DeletedRows = Me.DsCustomers.GetChanges(DataRowState.Deleted)


Try

’ For each of these, we have to make sure that the Data Tables contain any records, otherwise, we will get an error. If
Not UpdatedRows Is Nothing Then
Me.CustomersTableAdapter.Update(UpdatedRows)
If Not InsertedRows Is Nothing Then
Me.CustomersTableAdapter.Update(InsertedRows)
If Not DeletedRows Is Nothing Then
Me.CustomersTableAdapter.Update(DeletedRows)
Catch eUpdate As System.Exception
MessageBox.Show ("Caught exception: " & eUpdate.Message)
End Try

Me.CustomersTableAdapter.Connection.Close()
End Sub
```

To write code in C#

C#

```
private void UpdateCustomers() {
this.CustomersTableAdapter.Connection.Open(); System.Data.DataSet UpdatedRows;
System.Data.DataSet InsertedRows;
System.Data.DataSet DeletedRows;


// These Data Tables hold any changes that have been made to the dataset since the last update. UpdatedRows =
this.DsCustomers.GetChanges(DataRowState.Modified);
InsertedRows = this.DsCustomers.GetChanges(DataRowState.Added); DeletedRows =
this.DsCustomers.GetChanges(DataRowState.Deleted); try {
// For each of these, we have to make sure that the Data Tables contain any records, otherwise, we will get an error.
if (! UpdatedRows == null ) this.CustomersTableAdapter.Update(UpdatedRows) if (! InsertedRows == null )
this.CustomersTableAdapter.Update(InsertedRows) if (! DeletedRows == null ) this.CustomersTableAdapter.Update(DeletedRows)
} catch (System.Exception eUpdate) {
MessageBox.Show ("Caught exception: " + eUpdate.Message); } this.CustomersTableAdapter.Connection.Close();
}
```

1. 现在回退到.NET 设计器上，单击ListBox1 控件。在属性窗口中设置它的DataSource 属性为CustomersBindingSource，，并且设置它的DisplayMember 属性为LastName.
2. 在窗体上单击第一个文本框(TextBox1)。在属性窗口扩展它的DataBindings 对象。设置在DataBindings 对象之下的Text 属性为CustomersBindingSource - FirstName。。为下两个文本框在关联的DataBindings对象之下，以相似的方式设置相同的Text 属性。TextBox2 应该被设置为LastName，，并且TextBox3 应该被设置为Company.
3. 最后在属性窗口中，为C1TrueDBGrid控件设置AllowAddNew，AllowDelete， 和AllowUpdate 属性为True。

运行此程序并观察下列各项:
使用鼠标或者键盘以改变网格中当前行的位置，并且观察其他绑定的控件 (ListBox1 and the TextBoxes)随着网格改变它的记录位置，即使他们没有包含代码 。
单击Next, Previous, Last，和First 按钮，并且观察它们已经有与数据控件上相应的按钮一样的效果。
在网格中的几个单元格中修改数据 (在相同行中) 。按下Update 按钮，观察被修改的数据已经被更改到数据库中，并且铅笔按钮在记录选择器的列中消失。其他在窗体中绑定的控件目前显示修改的数据。
在一个或者更多的文本控件上修改数据。按下Update或者Next 按钮。网格将自动更新它的数据以反映新的改变。
移动网格中当前的单元格到任何你想删除的记录上，然后单击Delete 按钮。此记录将被删除并且从网格中消失。在删除记录后网格自动移除当前的行到记录上. 其他在床上绑定的控件也相应的移除了它们的记录位置。
总结此教程，你已经成功完成了通过代码与其他绑定的控件之间的交互。

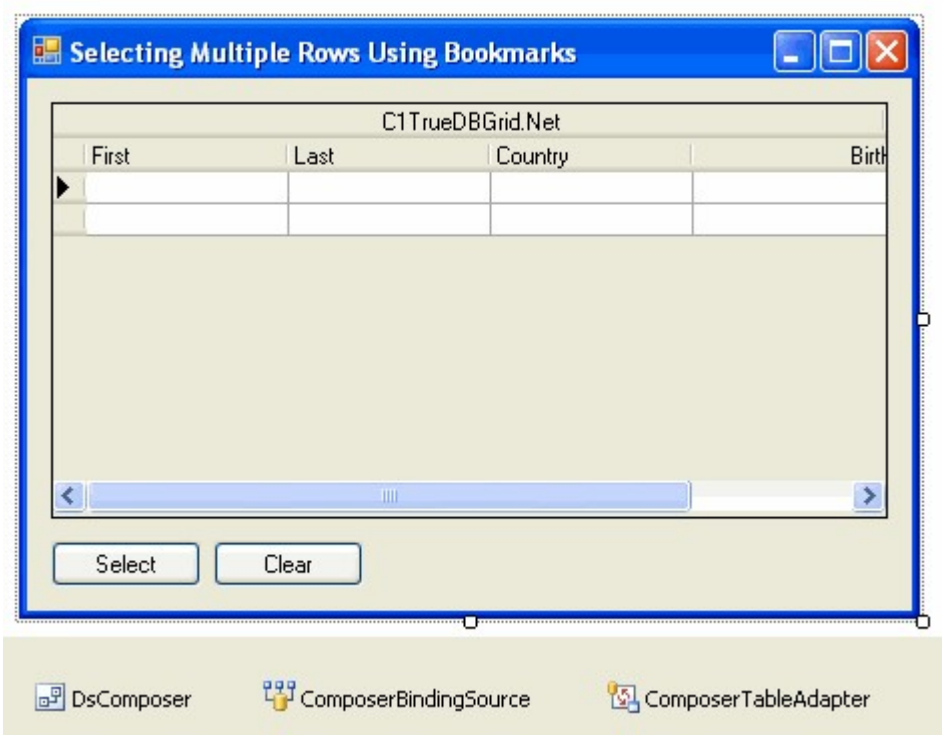
教程5：使用书签选择多行

在此教程中，你将学到如何选择和使记录高亮来满足指定的标准。在True DBGrid中相似的一组项目普遍以集合形式实现。当在True DBGrid中操纵一组项目时，使用技巧和在此描述的相似。

完成下面的步骤:

1. 创建一个新的.NET 工程.
2. 从IDE左边的工具框中，添加两个命令按钮和C1TrueDBGrid控件到窗体上。 C1TrueDBGrid图标看起来像这样:

1. 设置Button1的 Text 属性为"Select"并且设置Button2的Text属性为"Clear."
2. 在C1TrueDBGrid Tasks菜单上，查找Choose Data Source下拉框的位置并选择Add Project Data Source。。在适配器的Data Source Configuration Wizard，，或者选择一个连接到 C1NWind.mdb 或者创建一个新的连接到数据库。在导航上的Choose your database objects页面，选择 Composers表的所有字段并且键入"DsComposers"到DataSet name 框中，然后完成了此向导。



1. Visual Studio添加下面的代码到Form_Load 事件:

To write code in Visual Basic

```
Visual Basic
Me.ComposerTableAdapter.Fill(Me.DsComposer.Composer)
```

To write code in C#

```
C#
this.ComposerTableAdapter.Fill(this.DsComposer.Composer);
```

1. 我们可以通过操纵SelectedRowCollection 很容易的选择或者不选择True DBGrid 中的行。选择行，添加下面的代码到Button1的 Click事件 :

To write code in Visual Basic

```
Visual Basic
Dim l As Integer
For l = 0 To Me.DsComposer.Composer.Rows.Count - 1
If Me.DsComposer.Composer.Rows(l).Item("Country") = "Germany" Then

Me.C1TrueDBGrid1.SelectedRows.Add(l)
End If
```

```
Next
Me.C1TrueDBGrid1.Refresh()
```

To write code in C#

```
C#
int l; for (l = 0 ; l < this.DsComposer.Composer.Rows.Count; l++) { if (this.DsComposer.Composer.Rows[l].["Country"] == "Germany") { this.c1TrueDBGrid1.SelectedRows.Add(l); } } this.c1TrueDBGrid1.Refresh();
```

1. 在Button2的Click事件中增加下面的代码，取消选中行:

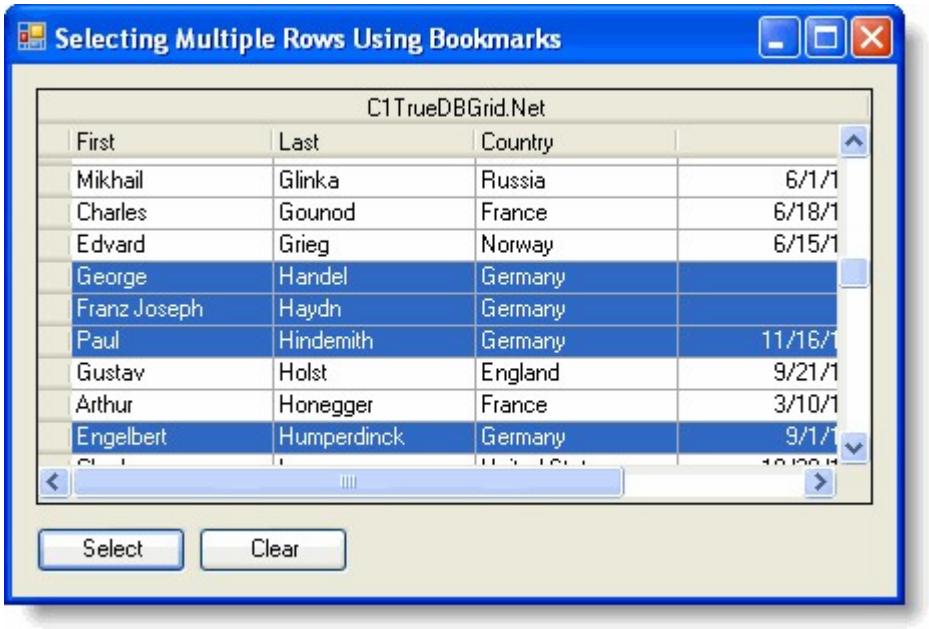
To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.SelectedRows.Clear()
```

To write code in C#

```
C#
this.c1TrueDBGrid1.SelectedRows.Clear();
```

运行此程序并观察下列各项:
C1TrueDBGrid1从数据集（DataSet）中检索数据库结构信息，并且自动配置它自己以显示内连接的数据库表中的所有字段。这是和教程1中网格的行为很相似。单击 Select 按钮并且观察所有记录的 Country 字段为 Germany 的行将以高亮显示。



取消选中的高亮记录，单击Clear 按钮，或者点击网格上单元格的任何一处，也将清除选中的行。总结此教程，你使用书签已经成功完成了选择多个行。

教程6：在绑定的网格中定义未绑定的列

在此教程中，你将学习到如何让使用UnboundColumnFetch 事件在一列中显示两个字段(FirstName和LastName)。你也将学习到如何使用一个SQL语句在数据库中对两个表创建内连接。完成下面的步骤:

1. 创建一个新的.NET 工程.
2. 从IDE左边的工具框中，双击图标以添加一个控件到窗体上。 C1TrueDBGrid图标看起来像这样:

1. 在C1TrueDBGrid 任务任务菜单中，查找选择数据源选择数据源下拉框的位置并选择添加工程数据源添加工程数据源。在适配器的数据源配置向数据源配置向导，或者选择一个连接到 C1NWind.mdb或者创建一个新的连接到数据库。在向导的选择你的数据对象选择你的数据对象页面上，选择Contacts表中的所有字段，并键入"DsContacts"到数据集名称数据集名称框，然后完成向导。
2. 在解决方案资源管理器窗口双击DsContacts.xsd以编辑设计器。右键Contacts表并从上下文菜单中选择配置。配置。
3. 在表适配器配置向导表适配器配置向导中修改SQL字符串为:

```
SELECT Customers.FirstName, Customers.LastName, Customers.CustType,
Contacts.ContactType, Contacts.Callback, Contacts.ContactDate, Contacts.UserCode,
Customers.UserCode AS Expr1 FROM Contacts INNER JOIN Customers ON
Contacts.UserCode = Customers.UserCode
```

1. Contacts表目前与Customers 表进行了内连接。单击Finish按钮退出向导。
2. 返回到设计视图并且如果提示替换存在的列外观，单击Yes.

注意注意: 如果所有的列没有显示在C1TrueDBGrid中，在C1TrueDBGrid 任务任务菜单的下拉框中重新选择数据源。这样，你将在DSContacts之下重新选择Contacts 表。

1. 在Form1中声明一个新的全局数据表（DataTable ）对象：

To write code in Visual Basic

```
Visual Basic
Dim dtCopy As New DataTable
```

To write code in C#

```
C#
DataTable dtCopy = new DataTable;
```

1. 现在在Form_Load 事件中添加下面的代码。在第一行填充数据集，第二行拷贝这个数据集，我们将使用后者来计算未绑定的列：

To write code in Visual Basic

```
Visual Basic
Me.ContactsTableAdapter.Fill(Me.DsContacts.Contacts) dtCopy = Me.DsContacts.Tables(0).Copy()
```

To write code in C#

```
C#
this.ContactsTableAdapter.Fill(this.DsContacts.Contacts);
```

```
dtCopy = this.DsContacts.Tables(0).Copy();
```

1. 创建未绑定的列，在属性窗口中通过点击紧邻着Columns 属性的ellipsis 按钮 (…) 打开C1TrueDBGrid 设计器设计器。下一步单击 Appendcolumn 按钮来创建新的列。在左窗格设置新的列的Caption属性为"Name"。注意存在Caption字段的值，但是在DataField中没有值，这样如何让知道这个网格是一个未绑定的列。此网格目前知道触发UnboundColumnFetch 事件。单击 OK 按钮以便关闭C1TrueDB Grid 设计器设计器.
2. 在属性窗口上通过点击紧邻着Splits 属性的ellipsis 按钮来打开SplitCollection编辑器。现在通过点击紧邻着DisplayColumns 属性的ellipsis 按钮来打开C1DisplayColumnCollection 编辑器。在此编辑器中，在左窗格中

查找你刚刚创建的未绑定的列。它被放置在网格中的最后一列。在DisplayColumns集合中决定字段的位置。通过在左窗格中使用上下箭头按钮来调整列到想要的位置。然后在右窗格中，设置它的Visible 属性为True。。现在我们的未绑定的列对用户可见，并且不只是在True DBGrid for WinForms 控件上.
你可以在此隐藏被使用的未绑定的列。从左边窗格中选择 FirstName列，然后在右边窗格，设置它的Visible 属性为False。这样从视图中隐藏FirstName列。重复的选择 LastName 列.
选择OK来关闭C1DisplayColumnCollection 编辑器并且再次单击OK来关闭SplitCollection 编辑器.

1. 添加下面的代码到UnboundColumnFetch 事件。此代码使用dtCopy
   来收集值并放置到未绑定的列，然后设置这些值等于e.Value，放置值到未绑定的列：

To write code in Visual Basic

```
Visual Basic
Private Sub C1TrueDBGrid1_UnboundColumnFetch(ByVal sender As System.Object, ByVal e As
C1.Win.C1TrueDBGrid.UnboundColumnFetchEventArgs) Handles
C1TrueDBGrid1.UnboundColumnFetch If e.Column.Caption = "Name" AndAlso e.Row < dtCopy.Rows.Count Then
e.Value = Me.C1TrueDBGrid1(e.Row, "FirstName").ToString + " " +
Me.C1TrueDBGrid1(e.Row, "LastName").ToString
End If
End Sub
```

To write code in C#

```
C#
private void c1TrueDBGrid1_UnboundColumnFetch(object sender,
C1.Win.C1TrueDBGrid.UnboundColumnFetchEventArgs e) { if(e.Column.Caption == "Name" && e.Row < dtCopy.Rows.Count) {
e.Value = this.c1TrueDBGrid1[e.Row, "FirstName"].ToString()+ " " + this.c1TrueDBGrid1[e.Row, "LastName"].ToString(); }

}
```

运行此程序并观察下列各项:
当应用运行时，应该与下面的图一样:
!worddav6f6b848051fff7fc103b814826fb71a0.png|height=316,width=466!正如定义在UnboundColumnFetch事件中，第一列显示组合的_First Name_和_LastName_字段。CustType, ContactType_和_Callback 列显示数字的值，这些值对用户十分模糊并且提供了一个无吸引力的数据呈C1TrueDBGrid1根据在设计时配置的五列，从连接的表中显示数据。
现。在接下来的三个教程中(7, 8, 和9)，将图解提高显示和用户界面。.
总结教程;你已经成功地完成了在绑定的网格中定义未绑定的列。

教程7：用内建的组合框显示翻译的数据

在此教程中，你将学习到如何让使用ValueItems对象来显示翻译的文本和为编辑功能的网格内建的下拉框可用--所有这些都不需要写一行代码。
完成下面的步骤：

1. 在教程教程6: 在绑定的网格中定义未绑定的列在绑定的网格中定义未绑定的列 (Section 12.6)中，开始创建的工程。
2. 确认C1TrueDBGrid 已经获得焦点，然后在属性窗口中单击紧邻着Columns 属性的ellipsis 按钮。这样将产生C1TrueDBGrid 设计器。设计器。
3. 选择 CustType 列数字，然后在左边窗格，单击紧邻着Valueitems 属性的扩展图标。这样将会显示

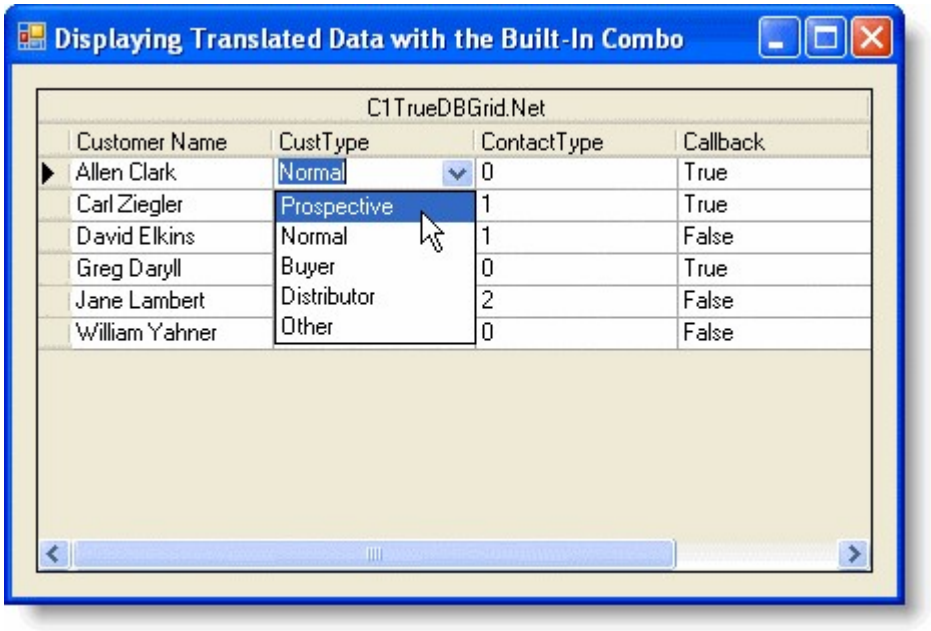
Valueitems 对象的所有成员。

1. C1TrueDBGrid 设计器设计器中单击紧邻着Values 属性的ellipsis 按钮。这样会产生ValueitemCollection 编辑器编辑器.
2. 在左边窗格中通过点击Add 按钮五次来创建五个新的ValueItem 对象。注意一个ValueItem 有两个属性：DisplayValue 和Value.
3. 在右边窗格中键入下面的成对DisplayValue/Value pairs，然后关闭ValueItemCollection 编辑器编辑器:

DisplayValue	Value
Value	0
Prospective	1
Normal	2
Buyer	3
Distributor	4
Other	5

4. 在 C1DataColumn编辑器编辑器中Valueitems对象下面，设置Presentation属性到ComboBox，，并且设置Translate 属性为True.
5. 单击属性页面属性页面对话框底部的 OK 按钮以接受改编。

运行程序并观察下面的步骤:
正如在教程教程6: 在绑定的网格中定义未绑定的列在绑定的网格中定义未绑定的列 (Section 12.6)中，C1TrueDBGrid1从连接的多个表中显示数据。
CustType 列目前显示翻译的文本代替数字值。
在CustType 列中单击单元格使它成为当前的单元格。注意下拉框按钮显示在单元格的右边边缘。
单击下拉按钮框或者按下ALT+DOWN
箭头来显示包含翻译值的组合框，正如下面的图形中显示的。通过从组合框中选择期望的项目，在当前的单元格中改变数据。



总结教程; 你已经成功完成了在内建的组合框中显示翻译的数据。

教程8：将下拉框控件附加到网格单元格

在此教程中，你将学习到如何附加一个多列的True DBDropDown 控件到网格单元格。不像在教程教程7: 通过内建的组合通过内建的组合框显示翻译的数据框显示翻译的数据 (Section 12.7)中的内建的组合框演示的那样，C1TrueDBDropDown 控件可以绑定数据源，使得它非常适合数据条目涉及到一个二次查找表.
无论何时用户在当前单元格中点击按钮，都会显示下拉框控件。当用户使一个已经有下拉框控件连接到的列获得焦点，这个按钮会自动显示。
完成下面的步骤:

1. 在教程教程6: 在绑定的网格中定义未绑定的列在绑定的网格中定义未绑定的列 (Section 12.6)，从工程结构开始
2. 添加一个 C1TrueDBDropDown 控件(C1TrueDBDropDown1)到窗体上C1TrueDBDropDown的图标与下面的看起来相似:

1. 跳转到数据源数据源属性并且从下拉列表中选择添加工程数据源添加工程数据源。
   在适配器的数据源配置向导数据源配置向导，或者选择连接到C1NWind.mdb 或者创建一个新的连接到数据库。在向导的选择你的数据对象选择你的数据对象页面 ，在CustType 表中选择TypeID 和TypeDesc 字段并且键入"DsCustType"到数据集名称数据集名称框，然后完成向导.
2. Visual Studio 添加下面的代码到Form1 的Load 事件来填充新的数据集：

To write code in Visual Basic

```
Visual Basic
Me.CustTypeTableAdapter.Fill(Me.DsCustType.CustType)
```

To write code in C#

```
C#
this.CustTypeTableAdapter.Fill(this.DsCustType.CustType);
```

1. 然后再次在Form1的Load 事件中，添加下面的代码以设置C1TrueDBDropDown1到CustType 列:

To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.Columns("CustType").DropDown = Me.C1TrueDBDropdown1
```

To write code in C#

```
C#
this.c1TrueDBGrid1.Columns("CustType").DropDown = this.c1TrueDBDropdown1;
```

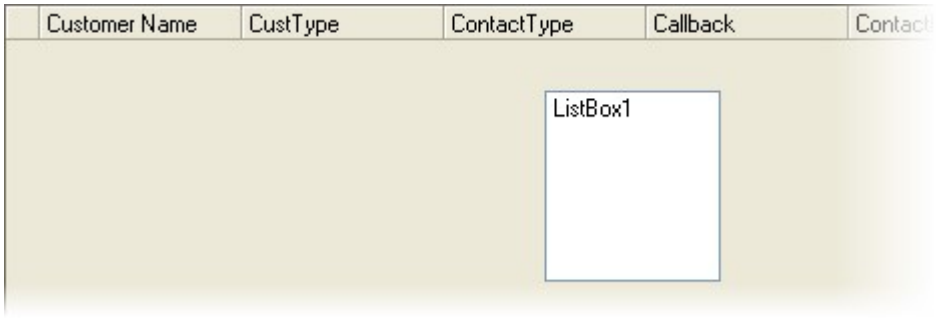
1. 在属性窗口中设置C1TrueDBDropDown1 的DisplayMember属性为TypeID。。此属性通知下拉框列将使下拉框绑定的网格列同步. 运行程序并观察下面的步骤:

正如在教程教程*6:在绑定的网格中定义未绑定的列在绑定的网格中定义未绑定的列 *(Section 12.6)，C1TrueDBGrid1从连接的表中显示数据。在_CustType_ 列中单击单元格以使它作为高亮指示为当前的单元格。一个按钮将显示在单元格的右边缘。单击此按钮显示*True DBDropDown* 控件，正如下面图形中所显示的。 !worddav4a7f5254cdbbe9a284af511bf0c8070e.png|height=316,width=466!使用UP ARROW 和DOWN ARROW 关键字来移动C1TrueDBDropDown控件的高亮条。如果在网格中的另一个单元格被点击，C1TrueDBDropDown将失去焦点并且不可见。在C1TrueDBDropDown中选择任何一个条目。在网格中的当前单元格将与被选择的条目一同更新。并且C1TrueDBDropDown将消失直到编辑被重新初始化。如果当前单元格被移动到另一个单元格，按钮将从_CustType_ 列中的单元格消失。
你已经成功完成了附加下拉框控件到网格单元格中；此总结在教程8中。

教程9：将渐变的下拉框控件附加到网格单元格

在教程中，你将学习到如何在网格单元格中下拉一个任意的控件。这个例子使用一个列表框控件，该控件包含预定义输入值并为了促进用户的数据输入。无论何时当用户开始编辑时该列表下拉，正如通过点击当前单元格。你也将学习到如何放置一个按钮到单元格中，且该按钮被点击时，将引起下拉框出现。你也将从网格单元格中下拉任何控件，使用的技术在此教程中描述如下。
完成下面的步骤:

1. 在教程教程6: 在绑定的网格中定义未绑定的列在绑定的网格中定义未绑定的列 (Section 12.6)中开始工程的构建。
2. 正如下图所示，从表单中添加列表框控件(ListBox1)。



1. 设置ListBox1 的 Visible 属性为False。。

添加代码以下拉一个列表框控件

在网格第二列 (Column1)中的CustType 字段，显示从1到5的数字值，这些值以下面的自定义类型呈现：

1. = Prospective
2. = Normal
3. = Buyer
4. = Distributor
5. = Other

下拉框ListBox1包含文本的自定义类型描述，并且允许用户双击一个条目，是为了键入相关的值到网格中。

1. 在Form1的一般声明部分包含下面的代码。添加这些命名空间将简化我们在之后需要写的代码。

To write code in Visual Basic

Visual Basic
Imports System.Data Imports System.Data.Oledb Imports System.IO Imports System.Collections

To write code in C#

C#
using System.Data; using System.Data.Oledb; using System.IO;

using System.Collections;

1. 在Form1中的Load事件，添加代码以包含自定义的类型到ListBox1。并且使用Button属性在CustType 列放置一个按钮。现在Form1_Load 事件处理器与下面的看起来一样：

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 Me.ContactsTableAdapter.Fill(Me.DsContacts.Contacts) ' Add customer types to ListBox1. With Me.ListBox1 .Items.Add("Prospective") .Items.Add("Normal") .Items.Add("Buyer") .Items.Add("Distributor") .Items.Add("Other") .Visible = False End With ' Place a button in the CustType column. Me.C1TrueDBGrid1.Splits(0).DisplayColumns("CustType").Button = True End Sub

To write code in C#

C#

private void Form1_Load(System.object sender, System.EventArgs e) {
this.ContactsTableAdapter.Fill(this.DsContacts.Contacts);
// Add customer types to ListBox1. this.listBox1.Items.Add("Prospective"); this.listBox1.Items.Add("Normal");
this.listBox1.Items.Add("Buyer"); this.listBox1.Items.Add("Distributor"); this.listBox1.Items.Add("Other");
this.listBox1.Visible = false;
// Place a button in the CustType column.
this.c1TrueDBGrid1.Splits[0].DisplayColumns["CustType"].Button = true;
 	
}

1. 如果在CustType 列中的单元格成为了当前单元格，一个按钮将被放置在单元格的右边缘。单击按钮将触发网格的ButtonClick 事件。当一个按钮无论何时被点击时，我们将下拉该ListBox1列表框：

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
```

```
With ListBox1
.Left = Me.C1TrueDBGrid1.Left + Me.C1TrueDBGrid1.RecordSelectorWidth +
Me.C1TrueDBGrid1.Splits(0).DisplayColumns(0).Width +
Me.C1TrueDBGrid1.Splits(0).DisplayColumns(1).Width
.Top = Me.C1TrueDBGrid1.Top +
Me.C1TrueDBGrid1.RowTop(Me.C1TrueDBGrid1.Row)
.Visible = True
.Select()
End With
End Sub
```

To write code in C#

```
C#
```

```
private void c1TrueDBGrid1_ButtonClick(object sender,
C1.Win.C1TrueDBGrid.ColEventArgs e) { this.listBox1.Left = this.c1TrueDBGrid1.Left +
this.c1TrueDBGrid1.RecordSelectorWidth + this.c1TrueDBGrid1.Splits[0].DisplayColumns[0].Width +
this.c1TrueDBGrid1.Splits[0].DisplayColumns[1].Width; this.listBox1.Top = this.c1TrueDBGrid1.Top +
this.c1TrueDBGrid1.RowTop(this.c1TrueDBGrid1.Row); this.listBox1.Visible = true; this.listBox1.Select();

}
```

1. 在ListBox1列表框的DoubleClick 事件，添加下面的代码。当一个ListBox1列表框中的条目被选中时，这些代码将复制它的下标到C1TrueDBGrid1中的合适的列，然后使ListBox1 列表框不可见。

To write code in Visual Basic

```
Visual Basic
```

```
Private Sub ListBox1_DoubleClick(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ListBox1.DoubleClick
Me.C1TrueDBGrid1.Columns("CustType").Text = Me.ListBox1.SelectedIndex + 1

Me.ListBox1.Visible = False
End Sub
```

To write code in C#

```
C#
```

```
private void listBox1_DoubleClick(object sender, System.EventArgs e) { this.c1TrueDBGrid1.Columns["CustType"].Text =
(this.listBox1.SelectedIndex +
1).ToString(); this.listBox1.Visible = false;
}
```

1. 然后在ListBox1列表框的Leave 事件，添加下面的代码，以确信一旦选中的部分产生，列表框不可见：

To write code in Visual Basic

```
Visual Basic
```

```
Private Sub ListBox1_Leave(ByVal sender As Object, ByVal e As System.EventArgs) Handles ListBox1.Leave

Me.ListBox1.Visible = False
End Sub
```

To write code in C#

```
C#
```

```
private void listBox1_Leave(object sender, System.EventArgs e) { this.listBox1.Visible = false;

}
```

1. 然后在C1TrueDBGrid1的Scroll事件中，添加下面的代码，以确信一旦网格滚动，列表框不可见：

To write code in Visual Basic

```
Visual Basic
```

```
Private Sub C1TrueDBGrid1_Scroll(ByVal sender As Object, ByVal e As

C1.Win.C1TrueDBGrid.CancelEventArgs) Handles C1TrueDBGrid1.Scroll
Me.ListBox1.Visible = False
End Sub
```

To write code in C#

```
C#
```

```
private void c1TrueDBGrid1_Scroll(object sender,
C1.Win.C1TrueDBGrid.CancelEventArgs e) { this.listBox1.Visible = false;

}
```

运行程序并观察下面的步骤:
C1TrueDBGrid1从连接的表中显示数据，正如在教程教程6: 在绑定的网格中定义未绑定的列在绑定的网格中定义未绑定的列 (Section 12.6)。在CustType列中点击一个单元格使它成为当前单元格，该单元格以高亮显示。一个按钮将显示在单元格的右边缘。单击按钮以触发ButtonClick事件。正如下面图中显示的那样，ListBox1 将在单元格的右边缘下拉。.
!worddava93a34919a7919682eac03ae487582f1.png|height=316,width=466!使用鼠标或者向上箭头UP ARROW 和向下箭头DOWN ARROW关键字来移除ListBox1列表框中的高亮条。如果网格中的另一个单元格被点击，ListBox1列表框将失去焦点并且不可见。双击ListBox1中的任何条目。网格中的当前单元格将随着被选中的条目被更新，并且ListBox1将消失，直到编辑功能重新被初始化。如果当前单元格被移动到另一个列，该按钮将从_CustType_列中的单元格中消失。
你已经成功完成了附带一个任意的下拉控件到一个网格单元格；这些总结了教程9。

教程10：通过在单元格中的位图加强界面

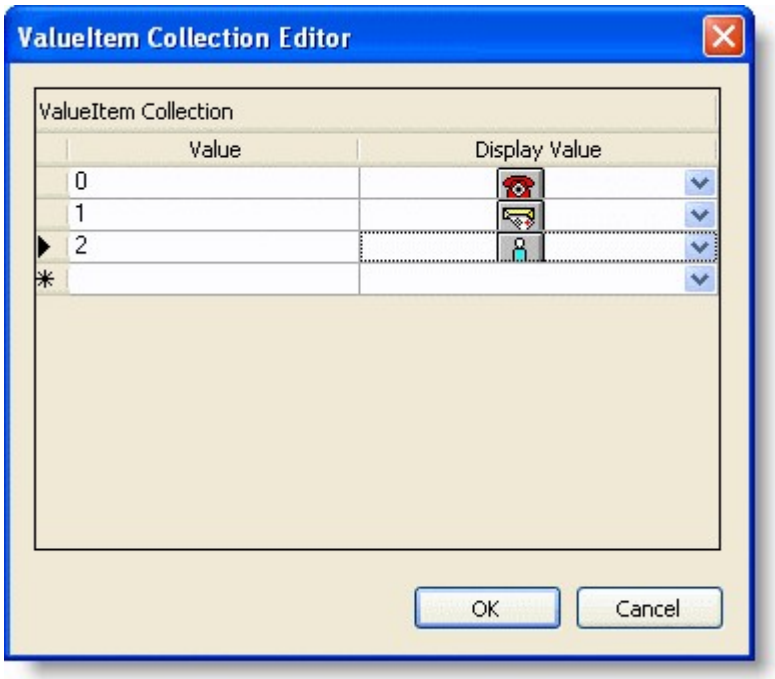
在此教程中，你将学习到如何使用ValueItems 对象和它的ValueItem对象集合来显示位图，并检查单元格中的复选框-没有写一行代码!
完成下面的步骤：Complete the following steps:

1. 以教程教程7: 用嵌入的选框显示翻译的数据用嵌入的选框显示翻译的数据 (Section 12.7)中使用的工程开始。
2. 第一，改变ContactType 和Callback 列的说明文字，在属性窗口中通过点击紧邻着Columns列的ellipsis 按钮打开C1TrueDBGrid Designer 。
3. 选择ContactType列，然后在左边窗格改变它的Caption 属性为 "How"。然后用类似的方式，改变CallBack列的说明文字为 "Call?"
4. 通过点击Align center 按钮，改变两个列的HorizontalAlignment 属性，以便位图将被设置在单元格的中心。在属性窗口通过点击紧邻着Splits属性的ellipsis 按钮打开SplitCollection Editor 。下一步，在编辑器中，通过点

击紧邻着DisplayColumn属性的ellipsis按钮打开C1DisplayColumnCollection Editor。在左边窗格选择How列，然后在右边窗格，点击紧邻着Styles属性的扩展按钮，在该列的样式对象下面，设置HorizontalAlignment属性为Center。然后设置VerticalAlignment 属性为Center。以相似的方式，设置 HorizontalAlignment和VerticalAlignment 属性为Call? 列。

1. 下一步，通过计算相应的ValueItems对象，分配位图和复选框为选中的列。我们将在列2中开始位图。在属性窗口，通过点击紧邻着Columns 属性的ellipsis 按钮打开C1TrueDBGrid Designer。选择How 列，然后再左边窗格，单击紧邻着Valueitem对象的扩展按钮。并点击紧邻着Values属性的ellipsis按钮打开ValueItemCollection

Editor:



1. 在左边窗格，通过点击Add 按钮创建三个ValueItem 对象。ContactType 字段可能的值分别为0, 1, 和2，这些值

呈现电话号码，邮箱和个人联系方式，各自的位图将会代替数字值在单元格中显示。如果全部产品被安装，下面的文件将在教程10的安装目录的子目录：PHONE.BMP, MAIL.BMP和PERSON.BMP中被发现。

1. 在右边窗格，为第一个ValueItem键入0作为值，然后在DisplayValue属性框，单击ellipsis按钮，在单元格中搜索图片文件并显示。在教程10 WinForms Edition 安装目录的子目录中定位Phone.bmp文件。以相同的方式设置其它的两个ValueItem 对象的值分别为1，Mail.bmp的DisplayValue和一个为2的值，Person.BMP的

DisplayValue。在C1TrueDBGrid Designer中返回到ValueItems 对象，并且设置Translate和CycleOnClick 属性为True.

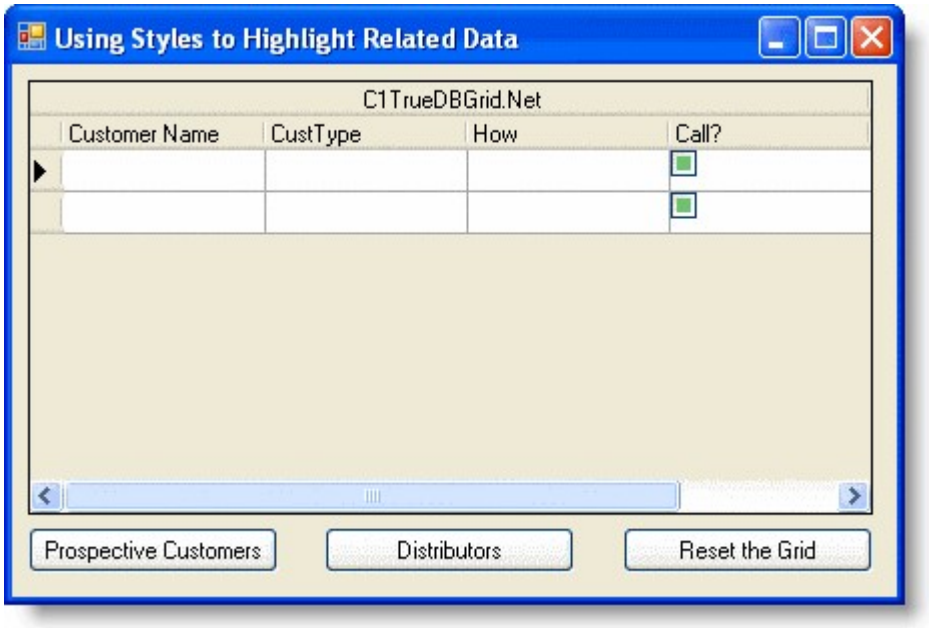
1. 为列3设置复选框，在C1TrueDBGrid Designer中，选择Call 列。在左边窗格，扩展ValueItems对象并设置Presentation属性为CheckBox。这些将显示列的布尔值如复选框的值。最后，在同一个对象下面设置Translate和CycleOnClick属性为True. 运行程序并观察下面的步骤:

C1TrueDBGrid1从连接的表中显示数据。
正如下面的图形中显示的那样，How 和Call? columns 目前用位图代替数字值进行显示。
!worddav104226d4f63ca1b1ec20fc98abfe2b76.png|height=316,width=466!在_How_
列单击一个单元格使它成为当前列。然后多次重复单击它，并观察该单元格如何通过PHONE, MAIL, 和PERSON 位图循环。在_Call?_
列单击一个单元格并观察该单元格如何通过复选框位图进行循环。
你已经成功完成了通过在单元格中的位图加强用户界面；这是教程10的总结。

教程11：使用样式高亮相关的数据

在此教程中，你将学习到如何通过依赖网格中的值创建行样式，改变网格的显示来高亮这些行，True DBGrid使用FetchRowStyle 事件来创建样式特征并且动态地应用到行上。
完成下面的步骤:

1. 从教程教程 10: 用单元格中的位图加强用户界面用单元格中的位图加强用户界面 (Section 12.10)中使用的工程开始。
2. 添加3个按钮到表单上。改变Button1的说明文字为Prospective Customers，Button2的说明文字为Distributors，和Button3的说明文字为Reset the Grid，窗体如下所示。



1. 为Form1的General部分添加下面的申明：

To write code in Visual Basic

```
Visual Basic
Dim bflag As Integer
```

To write code in C#

```
C#
int bflag;
```

1. 在Button1的Click 事件中键入下面的代码:

To write code in Visual Basic

```
Visual Basic
' Prospective Customers. Me.C1TrueDBGrid1.FetchRowStyles = True bFlag = 1
Me.C1TrueDBGrid1.Refresh()
```

To write code in C#

```
C#
// Prospective Customers. this.c1TrueDBGrid1.FetchRowStyles = true; bFlag = 1; this.c1TrueDBGrid1.Refresh();
```

1. 在Button2的Click 事件中键入下面的代码：

To write code in Visual Basic

```
Visual Basic
' Distributors. Me.C1TrueDBGrid1.FetchRowStyles = True bFlag = 2
Me.C1TrueDBGrid1.Refresh()
```

To write code in C#

```
C#
// Distributors. this.c1TrueDBGrid1.FetchRowStyles = true; bFlag = 2; this.c1TrueDBGrid1.Refresh();
```

1. 在Button3的Click 事件中键入下面的代码：

To write code in Visual Basic

```
Visual Basic

' Reset the grid.
Me.C1TrueDBGrid1.FetchRowStyles = False
Me.C1TrueDBGrid1.Refresh()
```

To write code in C#

```
C#
// Reset the grid. this.c1TrueDBGrid1.FetchRowStyles = false; this.c1TrueDBGrid1.Refresh();
```

1. 下一步键入下面的代码到FetchRowStyles 事件。这些代码在单击事件中与FetchRowStyles 属性的设置进行交互。当FetchRowStyles 被设置为True，，当它需要重画单元格时，网格触发FetchRowStyle 事件。这样行样式根据bflag整形标记被应用：

To write code in Visual Basic

```
Visual Basic
Private Sub C1TrueDBGrid1_FetchRowStyle(ByVal sender As Object, ByVal e As
C1.Win.C1TrueDBGrid.FetchRowStyleEventArgs) Handles C1TrueDBGrid1.FetchRowStyle


If bFlag = 1 And Me.C1TrueDBGrid1 (e.Row,"CustType") = 1 Then
Dim fntFont As New Font(e.CellStyle.Font.Name, e.CellStyle.Font.Size,
FontStyle.Bold)
e.CellStyle.Font = fntFont
e.CellStyle.ForeColor = System.Drawing.Color.Blue End If
If bFlag = 2 And Me.C1TrueDBGrid1 (e.Row, "CustType") = 4 Then
e.CellStyle.ForeColor = System.Drawing.Color.White
e.CellStyle.BackColor = System.Drawing.Color.Red End If
End Sub
```

To write code in C#

```
C#
```

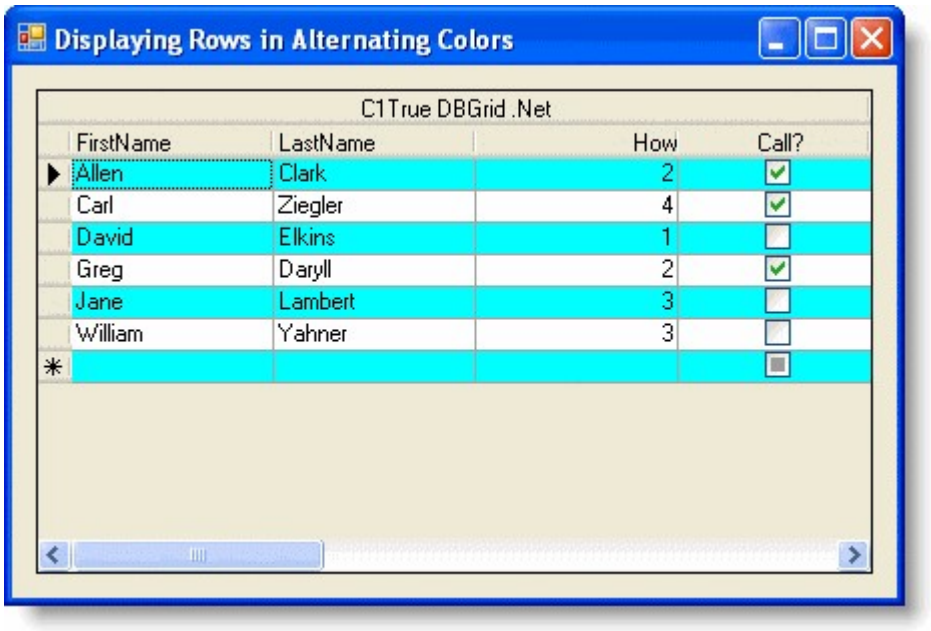
```
private void C1TrueDBGrid1_FetchRowStyle(object sender,
C1.Win.C1TrueDBGrid.FetchRowStyleEventArgs e) { if (bFlag == 1 && (int)this.c1TrueDBGrid1 [e.Row, "CustType"] == 1 ) {


Font fntFont = new Font(e.CellStyle.Font.Name, e.CellStyle.Font.Size,
FontStyle.Bold);
e.CellStyle.Font = fntFont;
e.CellStyle.ForeColor = System.Drawing.Color.Blue; } if (bFlag == 2 && this.c1TrueDBGrid1 [e.Row, "CustType"] == 4 ) {


e.CellStyle.ForeColor = System.Drawing.Color.White;
```

e.CellStyle.BackColor = System.Drawing.Color.Red; }
}
运行程序并观察下面的步骤:
C1TrueDBGrid1 如教程教程 10: 内置单元格位图加强用户界面内置单元格位图加强用户界面 (Section 12.10)*中一样显示数据.单击*Prospective Customers 按钮，网格应该如下所示。
!worddavc9530fb687ddac1e9d949dfb4e4e1edf.png|height=316,width=466!单击*Distributors* 按钮。网格现在应该如下所示:
!worddav89fe01e64e47e5ace17919cc93cf7af8.png|height=316,width=466!最后单击*Reset the Grid* 按钮。网格线应该清除它自己的样式。
你已经成功完成了使用样式来高亮相关的数据；这些总结了教程11.

教程12：在交互的颜色中显示行

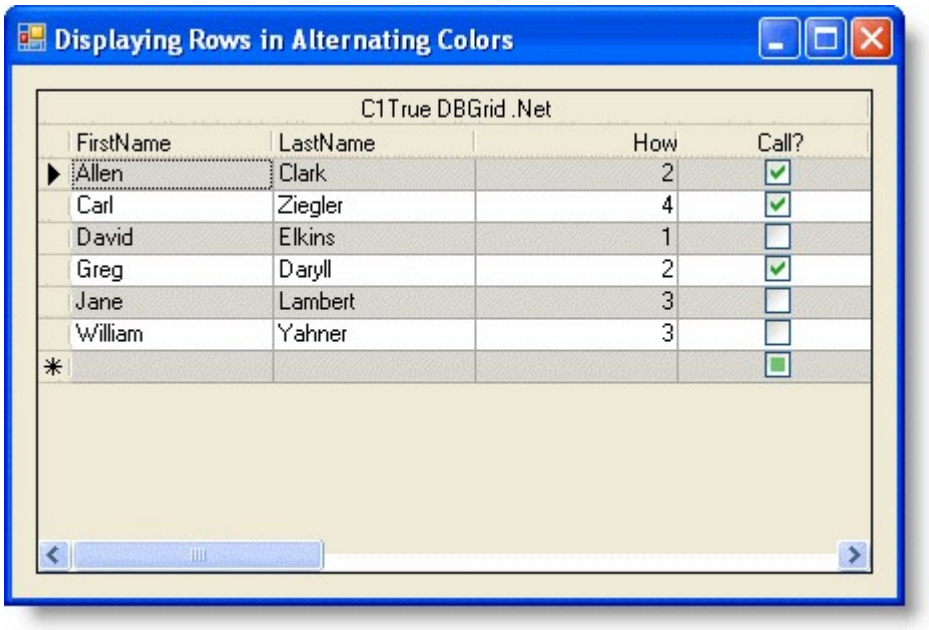
在此教程中，你将学习到如何让使用AlternatingRows属性和内建的样式来应用交替的颜色到网格的行上，并提高它们的可读性。
完成下面的步骤:

1. 以教程教程 10: 使用使用In-Cell位图增强用户界面位图增强用户界面 (Section 12.10)中使用的工程开始.
2. 在属性窗口，设置AlternatingRows 属性为True。网格为EvenRow和OddRow的样式有默认的设置。首先使用默认的设置，然后为EvenRowStyle改变设置。



1. 在教程教程 10: 使用使用In-Cell位图增强用户界面位图增强用户界面 (Section 12.10)中运行程序并观察C1TrueDBGrid1展示数据。除过偶数行有青绿色背景。
2. 在属性窗口点击紧邻着样式样式属性的省略号省略号按钮。将产生C1TrueDBGrid 样式编辑器。样式编辑器。
3. 在左边窗格选择EvenRowStyle ，并在右边窗格改变它的BackColor 为 LightGray 。点击OK并关闭编辑器。

运行程序并观察下面的步骤:
正如在先前的图片中，C1TrueDBGrid1展示数据，目前除过偶数行有青灰色背景:



总结该教程。

教程13：实现拖拽下拉功能

在此教程中，你将在True DBGrid for WinForms中学习到如何实现drag-and-drop功能性.
为WinForms 控件安装True DBGrid 完成下面的步骤:

1. 开始一个新的.NET工程
2. 放置两个 C1TrueDBGrid 控件(C1TrueDBGrid1, C1TrueDBGrid2)到窗体上。并在窗体上添加三个标签，并安排它们与下面的图片看起来一样.
3. 在C1TrueDBGrid1的C1TrueDBGrid Tasks菜单上，定位选择数据源选择数据源下拉框并选择添加项目数据源添加项目数据源。在适配器的数据源的配置向导数据源的配置向导，或者选择一个连接到 C1NWind.mdb，或者创建一个连接到数据库。在向导的选择数据选择数据库对象库对象页面，在Customers 表选择所有字段，并在数据集名称数据集名称框中键入"DsCustomers" ，然后完成向导。
4. 在C1TrueDBGrid2的C1TrueDBGrid Tasks菜单上，定位选择数据源选择数据源下拉框并选择添加工程数据源添加工程数据源. 在适配器的数数据源配置向导据源配置向导，或者选择一个连接到C1NWind.mdb，或者创建一个新的连接到数据库。在向导的选择你的数据选择你的数据库对象库对象页面，在CallList 表选择所有字段，并在数据集名称数据集名称框中键入"DsCallList" ，然后退出向导。
5. 在Form中，添加下面的声明:

To write code in Visual Basic

```
Visual Basic
Dim _ptStartDrag As Point

Dim _dragRow As Long
```

To write code in C#

```
C#
Point _ptStartDrag; long _dragRow;
```

1. Visual Studio添加下面的代码到Form1 的Load 事件来填充新的数据集:

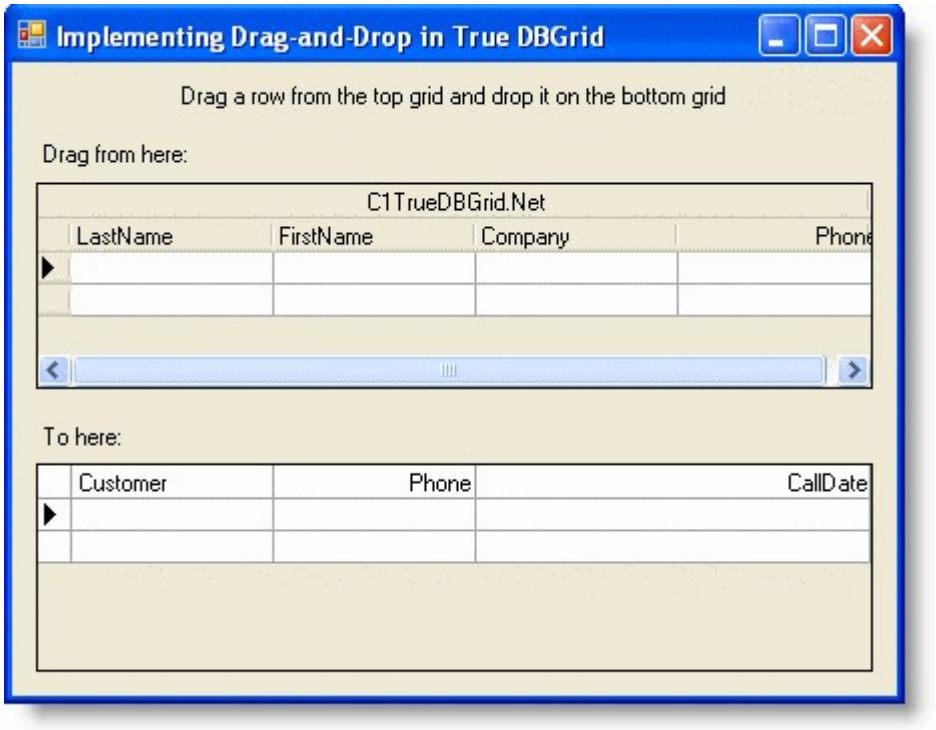
To write code in Visual Basic

```
Visual Basic
Me.CallListTableAdapter.Fill(Me.DsCallList.CallList)
Me.CustomersTableAdapter.Fill(Me.DsCustomers.Customers)
```

To write code in C#

```
C#
this.CallListTableAdapter.Fill(this.DsCallList.CallList); this.CustomersTableAdapter.Fill(this.DsCustomers.Customers);
```

1. 对于第一个网格(C1TrueDBGrid1) 设置AllowDrag 属性为True。同时对于第二个网格，设置AllowDrop 属性为True.
2. 右键C1TrueDBGrid1并选择Retrieve Fields。其它网格与之相同。
3. 在属性窗口，通过点击紧邻着Columns属性的省略号省略号按钮，来打开C1TrueDBGrid1 的C1TrueDBGrid 设计器设计器。
4. 通过点击每一列的Remove Column按钮，按钮，从网格中移除除了LastName, FirstName, Company, 和Phone的所有列。为其它网格键入C1TrueDBGrid Designer 并移除除了Customer, Phone, 和CallDate的所有列。
5. 在属性窗口设置C1TrueDBGrid1的MarqueeStyle值为o SolidCellBorder。在C1TrueDBGrid Designer 设置列3 的 (Phone) NumberFormat 属性为"(###)#######"。为第二个网格打开C1TrueDBGrid Designer并设置它的列 2的NumberFormat 属性为"(###)#######"。网格应该与下面的看起来一样：



添加代码到你的工程

该章节描述了代码需要从C1TrueDBGrid1到C1TrueDBGrid2拖动单元格或者行的内容。此代码设想，如果你想拖动数据的全部行到C1TrueDBGrid2并为了添加新的纪录。

1. 添加下面的子程序到工程中，为每一个网格重设MarqueeStyle属性，此方法被用作在程序中拖拽时提供可见的反馈。这个重设的程序将在一个拖拽和下拉操作总结之后被调用来执行清除。

To write code in Visual Basic

Visual Basic

```
Private Sub ResetDragDrop()
' Turn off drag-and-drop by resetting the highlight and label text.
Me._ptStartDrag = Point.Empty
Me._dragRow = - 1
Me.C1TrueDBGrid1.MarqueeStyle =
C1.Win.C1TrueDBGrid.MarqueeEnum.SolidCellBorder Me.C1TrueDBGrid2.MarqueeStyle =
C1.Win.C1TrueDBGrid.MarqueeEnum.SolidCellBorder
Me.Label3.Text = "Drag a row from the top grid and drop it on the bottom grid." End Sub
```

To write code in C#

C#

```
private void ResetDragDrop() {
// Turn off drag-and-drop by resetting the highlight and label text. this._ptStartDrag = Point.Empty; this._dragRow = - 1;

this.c1TrueDBGrid1.MarqueeStyle =
C1.Win.C1TrueDBGrid.MarqueeEnum.SolidCellBorder; this.c1TrueDBGrid2.MarqueeStyle =
C1.Win.C1TrueDBGrid.MarqueeEnum.SolidCellBorder; this.label3.Text = "Drag a row from the top grid and drop it on the
bottom grid.";
}
```

1. 键入下面的代码来处理鼠标相关联的事件:

To write code in Visual Basic

Visual Basic

```
Private Sub C1TrueDBGrid1_MouseDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles C1TrueDBGrid1.MouseDown
Dim row, col As Integer
Me._ptStartDrag = Point.Empty
Me._dragRow = - 1
If Me.C1TrueDBGrid1.CellContaining(e.X, e.Y, row, col) Then

' Save the starting point of the drag operation.
Me._ptStartDrag = New Point(e.X, e.Y)
Me._dragRow = row
End If
End Sub

Private Sub C1TrueDBGrid1_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles C1TrueDBGrid1.MouseMove

' If we don't have an empty start drag point, then the drag has been
initiated. If Not Me._ptStartDrag.IsEmpty Then

' Create a rectangle that bounds the start of the drag operation by 2 pixels. Dim r As New Rectangle(Me._ptStartDrag,
Size.Empty)
r.Inflate(2, 2)
' If we've moved more than 2 pixels, start the drag operation.
If Not r.Contains(e.X, e.Y) Then
Me.C1TrueDBGrid1.Row = Me._dragRow
Me.C1TrueDBGrid1.MarqueeStyle =
C1.Win.C1TrueDBGrid.MarqueeEnum.HighlightRow
Me.C1TrueDBGrid1.DoDragDrop(Me._dragRow, DragDropEffects.Copy)
End If
End If
End Sub
```

To write code in C#

C#

```
private void C1TrueDBGrid1_MouseDown(object sender,
System.Windows.Forms.MouseEventArgs e) { int row, col; this._ptStartDrag = Point.Empty; this._dragRow = - 1; if
(this.c1TrueDBGrid1.CellContaining(e.X, e.Y, row, col) ) {
// Save the starting point of the drag operation.
this._ptStartDrag = new Point(e.X, e.Y); this._dragRow = row; }
}
private void C1TrueDBGrid1_MouseMove(object sender,
System.Windows.Forms.MouseEventArgs e)
{
// If we don't have an empty start drag point, then the drag has been initiated.
if (!this._ptStartDrag.IsEmpty ) {
// Create a rectangle that bounds the start of the drag operation by 2 pixels. Rectangle r = new
Rectangle(this._ptStartDrag, Size.Empty); r.Inflate(2, 2);
// If we've moved more than 2 pixels, start the drag operation. if (!r.Contains(e.X, e.Y) ) { this.c1TrueDBGrid1.Row =
this._dragRow;
this.c1TrueDBGrid1.MarqueeStyle = C1.Win.C1TrueDBGrid.MarqueeEnum.HighlightRow; this.c1TrueDBGrid1.DoDragDrop(this._dragRow,
DragDropEffects.Copy); }
}
}
```

1. 键入下面的代码来处理拖拽和下拉事件:

To write code in Visual Basic

Visual Basic

```
Private Sub C1TrueDBGrid2_DragEnter(ByVal sender As Object, ByVal e As
System.Windows.Forms.DragEventArgs) Handles C1TrueDBGrid2.DragEnter Me.Label3.Text = "Create a new record when dropped..."

e.Effect = DragDropEffects.Copy End Sub

Private Sub C1TrueDBGrid2_DragDrop(ByVal sender As Object, ByVal e As
System.Windows.Forms.DragEventArgs) Handles C1TrueDBGrid2.DragDrop
Try
Dim row As Integer = CInt(e.Data.GetData(GetType(Integer)))

' Use the grid's indexer to get some data.
Dim custname As String = Me.C1TrueDBGrid1(row, "FirstName").ToString()

' Use the CellText() method to get some data.
custname += " " + Me.C1TrueDBGrid1.Columns("LastName").CellText(row)
' Use the CellValue() method to get some data. custname += " " +
Me.C1TrueDBGrid1.Columns("Company").CellValue(row).ToString()

' Add a new row to the data set for the bottom grid.
Dim drv As DataRowView = Me.DsCallList.CallList.DefaultView.AddNew() drv("CallDate") = DateTime.Now drv("Customer") =
custname drv("Phone") = Me.C1TrueDBGrid1.Columns("Phone").Value.ToString() drv.EndEdit() Me.C1TrueDBGrid2.MoveLast()
Me.C1TrueDBGrid2.Select()

' Commit changes to the database.
Dim inserted As DataSet = Me.DsCallList.GetChanges(DataRowState.Added)
If Not (inserted Is Nothing) Then
Me.CallListTableAdapter.Update(inserted)
End If
Me.ResetDragDrop()
Catch ex As System.Exception
MessageBox.Show(ex.Message)
End Try
End Sub
```

To write code in C#

```
C#
```

```
private void C1TrueDBGrid2_DragEnter(object sender,
System.Windows.Forms.DragEventArgs e) { this.label3.Text = "Create a new record when dropped...";

e.Effect = DragDropEffects.Copy; }
private void C1TrueDBGrid2_DragDrop(object sender,
System.Windows.Forms.DragEventArgs e) { try { int row = (int)e.Data.GetData(typeof(int));
// Use the grid's indexer to get some data.
string custname = this.c1TrueDBGrid1[row, "FirstName"].ToString();

// Use the CellText() method to get some data.
custname += " " + this.c1TrueDBGrid1.Columns["LastName"].CellText(row);

// Use the CellValue() method to get some data.

custname += " " + this.c1TrueDBGrid1.Columns["Company"].CellValue(row).ToString();

// Add a new row to the data set for the bottom grid.
DataRowView drv = this.DsCallList.CallList.DefaultView.AddNew();

drv["CallDate"] = DateTime.Now; drv["Customer"] = custname;

drv["Phone"] = this.c1TrueDBGrid1.Columns["Phone"].Value.ToString();

drv.EndEdit();

this.c1TrueDBGrid2.MoveLast();

this.c1TrueDBGrid2.Select();

// Commit changes to the database.
DataSet inserted = this.DsCallList.GetChanges(DataRowState.Added);

if (! (inserted == null) )

{ this.CallListTableAdapter.Update(inserted); }

this.ResetDragDrop(); }

catch (System.Exception ex) {
MessageBox.Show(ex.Message);
}
}
```

运行程序并观察下面的步骤:
如果在C1TrueDBGrid1中的列的一个条目被拖拽，在C1TrueDBGrid1中的全部行以高亮显示，以表明数据的全部行被拖拽了。当拖拽TDBGrid2，当前的单元格选取框(单元格周围固体的边框) 消失.
!worddav2af74a67d590133617671d11b50bdeb5.png|height=366,width=466!如果数据在C1TrueDBGrid2中被下拉，一个新的纪录从C1TrueDBGrid1的当前行使用的数据被创建.
你已经成功地完成了在C1TrueDBGrid中实现拖拽和下拉；总结教程13.

教程14：创建固定的，无滚动列的网格

经常地你想要阻止一个或者多个列的水平滚动或者垂直滚动，以便它们始终在视图中。True DBGrid的SplitCollection提供了一个机制为相邻列的定义一个组，并且实现列固定，无滚动的列。在这个教程中，你将学习到如何编写代码来创建有两个Split的网格，并且在最左边的Split中列固定。
完成下面的步骤:

1. 根据教程教程1: 绑定绑定True DBGrid (Section 12.1)到数据集，用绑定到数据集来创建一个工程。
2. 在Form1的Load 事件，放置下面的代码创建两个Split，最左边Split列0和列1固定和另外一个Split:

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 ' Create an additional split.
Me.C1TrueDBGrid1.InsertHorizontalSplit(0)

' Hide all columns in the leftmost split except 0 and 1.
Dim x As Integer
For x = 2 To Me.C1TrueDBGrid1.Columns.Count - 1

Me.C1TrueDBGrid1.Splits(0).DisplayColumns
```

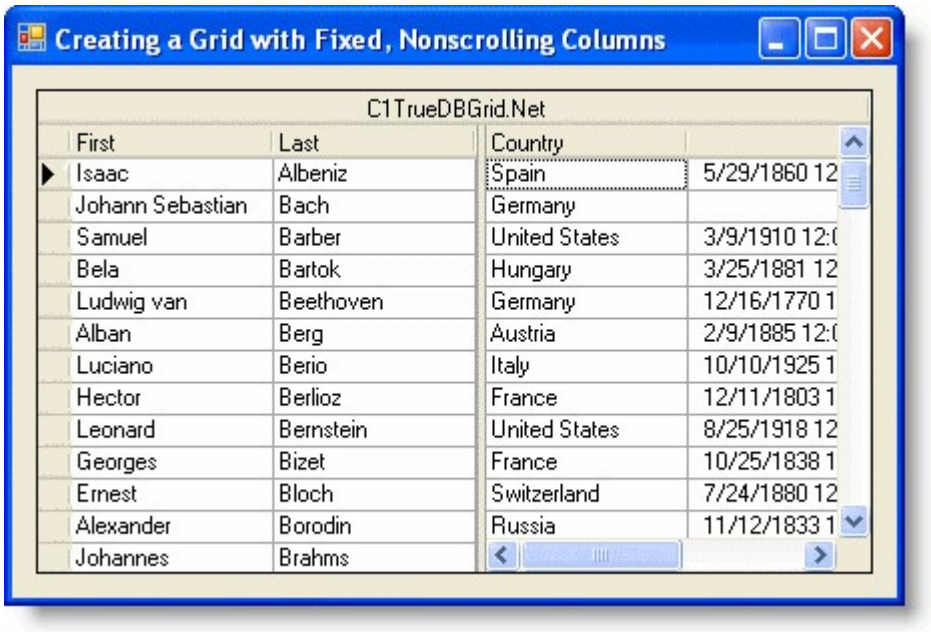
```
.Visible = False
Next
' Configure split 0 to display exactly 2 columns.
With Me.C1TrueDBGrid1.Splits(0)
.SplitSizeMode = C1.Win.C1TrueDBGrid.SizeModeEnum.NumberOfColumns
.SplitSize = 2
.AllowHorizontalSizing = False
End With
' Make columns 0 and 1 invisible in split 1.
Me.C1TrueDBGrid1.Splits(1).DisplayColumns(0).Visible = False Me.C1TrueDBGrid1.Splits(1).DisplayColumns(1).Visible =
False
' Turn off record selectors in split 1.
Me.C1TrueDBGrid1.Splits(1).RecordSelectors = False
End Sub
```

To write code in C#

C#

```
private void Form1_Load(System.object sender, System.EventArgs e) {
// Create an additional split. this.c1TrueDBGrid1.InsertHorizontalSplit(0);
// Hide all columns in the leftmost split except 0 and 1.

int x; for (x = 2 ; x < this.c1TrueDBGrid1.Columns.Count; x++) { this.c1TrueDBGrid1.Splits[0].DisplayColumns[x].Visible =
false; }

// Configure split 0 to display exactly 2 columns.
this.c1TrueDBGrid1.Splits[0].SplitSizeMode = C1.Win.C1TrueDBGrid.SizeModeEnum.NumberOfColumns;
this.c1TrueDBGrid1.Splits[0].SplitSize = 2; this.c1TrueDBGrid1.Splits[0].AllowHorizontalSizing = false;

// Make columns 0 and 1 invisible in split 1.

this.c1TrueDBGrid1.Splits[1].DisplayColumns[0].Visible = false; this.c1TrueDBGrid1.Splits[1].DisplayColumns[1].Visible =
false;

// Turn off record selectors in split 1.
this.c1TrueDBGrid1.Splits[1].RecordSelectors = false;
}
```

运行程序并观察下面的步骤:
C1TrueDBGrid 显示来自数据控件的数据，正如在 教程教程1: 绑定绑定True DBGrid到数据集到数据集 (Section 12.1).
在最左边的Split的两列(First 和_Last_)
被固定并且不能滚动。在左边Split的下面没有水平滚动条呈现。一个水平的滚动条出现在最右边Split的下面，允许用户滚动Split中的列。



在任何网格中使用Split来创建固定的，不滚动的列，甚至在中间也使用分隔符呈现数据的不同视图。例如，Split能创建
独立地滚动（在垂直方向）以便用户能比较纪录在数据库的最开始和最后。为了让获取更多信息，参见怎样使用分隔符怎样使用分隔符
(Section 9)。
你已经成功地完成了创建固定的，无滚动列的网格；总结教程14.

教程15：使用打印信息和打印预览

在此教程中，你可以学习到如何使用True DBGrid for WinForms的打印和输出能力。
完成下面的步骤：

1. 以在教程教程1：绑定绑定True DBGrid到数据集到数据集 (Section 12.1)中创建的工程开始.
2. 添加按钮到窗体上(Button1) 并且改变它的Text 属性为"Print Preview".
3. 键入下面的代码到Form1的Load事件。它改变了列的BackColor，改变了列的字体，为列设置 NumberFormat 属性到FormatText 事件，并且改变HeadingStyle：

To write code in Visual Basic

Visual Basic

```
' Change the presentation of the grid.
With Me.C1TrueDBGrid1.Splits(0).DisplayColumns
.Item("Country").Style.BackColor = System.Drawing.Color.Cyan Dim fntFont As Font
fntFont = New Font("Times New Roman", .Item("Country").Style.Font.Size, FontStyle.Regular)

.Item("Country").Style.Font = fntFont
.Item("Last").Style.ForeColor = System.Drawing.Color.Red
End With
Me.C1TrueDBGrid1.Columns("last").NumberFormat = "FormatText Event"
With Me.C1TrueDBGrid1.HeadingStyle
Dim fntfont As Font

fntfont = New Font(.Font.Name, .Font.Size, FontStyle.Bold) .Font = fntfont
.BackColor = System.Drawing.Color.Blue
.ForeColor = System.Drawing.Color.Yellow
End With
```

To write code in C#

C#

```
// Change the presentation of the grid. C1DisplayColumn col = this.c1TrueDBGrid1.Splits[0].DisplayColumns["Country"];
col.Style.BackColor = System.Drawing.Color.Cyan; Font fntFont; fntFont = new Font("Times new Roman", col.Style.Font.Size,
FontStyle.Regular); col.Style.Font = fntFont;
c1TrueDBGrid1.Splits[0].DisplayColumns["Last"].Style.ForeColor = System.Drawing.Color.Red;
this.c1TrueDBGrid1.Columns["last"].NumberFormat = "FormatText event"; Font fntfont;
fntfont = new Font(Font.Name, this.c1TrueDBGrid1.HeadingStyle.Font.Size, FontStyle.Bold);
this.c1TrueDBGrid1.HeadingStyle.Font = fntfont; this.c1TrueDBGrid1.HeadingStyle.BackColor = System.Drawing.Color.Blue;
this.c1TrueDBGrid1.HeadingStyle.ForeColor = System.Drawing.Color.Yellow;
```

1. 在先前的代码中，列的NumberFormat属性被设置为FormatText。这意味着FormatText 事件将触发可用的程序来改变列的样式和格式。键入下面的代码到FormatText 事件，这样会改变列的值为大写字体:

To write code in Visual Basic

```
Visual Basic
Private Sub C1TrueDBGrid1_FormatText(ByVal sender As Object, ByVal e As C1.Win.C1TrueDBGrid.FormatTextEventArgs) Handles
C1TrueDBGrid1.FormatText e.Value = UCase(e.Value)
End Sub
```

To write code in C#

```
C#
private void C1TrueDBGrid1_FormatText(object sender,

C1.Win.C1TrueDBGrid.FormatTextEventArgs e) {
e.Value = e.Value.ToUpper();
}
```

1. 添加下面的代码到Button1的Click事件。它使用PrintInfo 对象和他的属性和方法来创建页头和页脚。它以调用PrintPreview方法结束来实行打印先前的窗口：

To write code in Visual Basic

```
Visual Basic
With Me.C1TrueDBGrid1.PrintInfo

Dim fntFont As Font
fntFont = New Font(.PageHeaderStyle.Font.Name, .PageHeaderStyle.Font.Size,
FontStyle.Italic)
.PageHeaderStyle.Font = fntFont
.PageHeader = "Composers Table"
' Column headers will be on every page.
.RepeatColumnHeaders = True
' Display page numbers (centered).
.PageFooter = "Page: \p"
' Invoke print preview.
.UseGridColors = True
.PrintPreview()
End With
```

To write code in C#

```
C#
Font fntFont; fntFont = new Font(this.c1TrueDBGrid1.PrintInfo.PageHeaderStyle.Font.Name,
this.c1TrueDBGrid1.PrintInfo.PageHeaderStyle.Font.Size, FontStyle.Italic);
this.c1TrueDBGrid1.PrintInfo.PageHeaderStyle.Font = fntFont; this.c1TrueDBGrid1.PrintInfo.PageHeader = "Composers Table";
// Column headers will be on every page.
this.c1TrueDBGrid1.PrintInfo.RepeatColumnHeaders = true; // Display page numbers (centered). this.c1TrueDBGrid1.PrintInfo.
PageFooter = "Page:
p"; // Invoke print preview. this.c1TrueDBGrid1.PrintInfo.UseGridColors = true;
this.c1TrueDBGrid1.PrintInfo.PrintPreview();
```

运行程序并观察下面的步骤:
C1TrueDBGrid1 展示在第4步中使用字体和颜色改变指定的数据。
!worddavcd94522237b5e76d15d5b4a597be3804.png|height=316,width=466!点击 *Print
Preview*按钮来显示一个单独的应用窗体。输出反映了网格的格式。你已经成功地完成了使用打印信息和打印预览;总结教程15.

教程16：使用分层的显示

在此教程中，你将学习到如何通过网格的分层的展示来显示Master Detail DataSet信息。此教程与教程教程3: 链接多个链接多个True DBGrid控件控件 (Section 12.3)相似，但是此教程只能使用一个C1TrueDBGrid对象来显示教程教程3: 链接多个链接多个True DBGrid控控件件 (Section 12.3)一样的Master Detail DataSet信息。完成下面的步骤:

1. 用教程教程3:链接多个链接多个True DBGrid 控件控件 (Section 12.3)开始下面的步骤，设置一个Form有一个网格和Master Detail DataSet。
2. 在属性窗口，设置网格的DataSource属性为dsMasterDetail并且设置DataMember属性为Composer.
3. 下一步，在Form1的Load 事件，添加下面的代码，填充两个数据适配器并设置网格的显示为分层的：

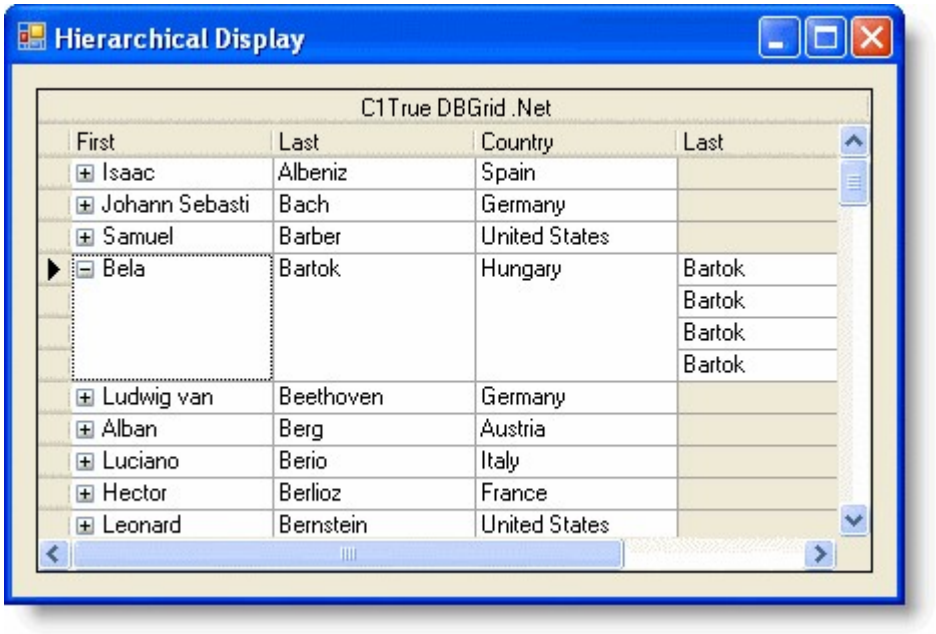
To write code in Visual Basic

```
Visual Basic
Me.ComposerTableAdapter.Fill(Me.DsMasterDetail.Composer)
Me.C1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.Hierarchical
```

To write code in C#

```
C#
this.ComposerTableAdapter.Fill(this.DsMasterDetail.Composer);
this.c1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.Hierarchical;
```

运行程序并观察下面的步骤:
C1TrueDBGrid1显示Composers 表的数据，但是每一行有扩展的图标。展开一行数据，我们注意到关联的Opus 的数据显示在网格的较远的列中。这些数据显示与Composers 表中的关联的那些记录而且都被扩展了:



你已经成功地完成了使用分层的显示；总结教程。

教程17：创建分组的显示

在教程中，你将学习到如何使用DataView 属性在网格上创建一个分组区域，使得用户可以在运行时对列数据排序。
完成下面的步骤:

1. 以在教程教程1: 绑定绑定True DBGrid到数据集到数据集 (Section 12.1) 中创建的工程开始。
2. 添加下面的代码到Form1的Load事件，在当前的数据适配器代码之后：

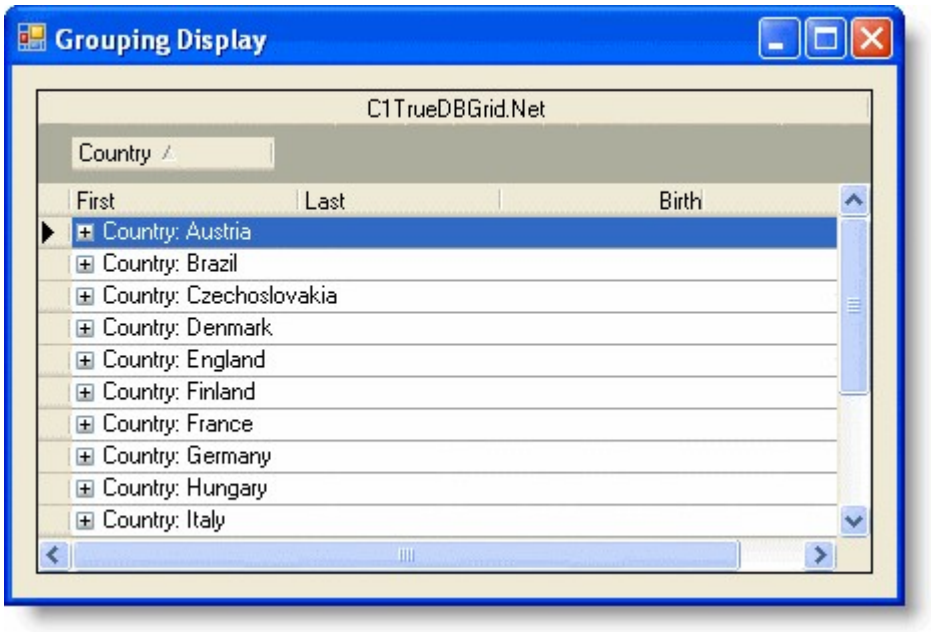
To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.GroupBy
```

To write code in C#

```
C#
this.c1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.GroupBy;
```

运行程序并观察下面的步骤:
C1TrueDBGrid1 显示在教程教程1: 绑定绑定True DBGrid到数据集到数据集 (Section 12.1)中指定的数据。
注意目前在网格上有一个分组区域。
点击Country列头并拖拽它分到组区域。你的网格目前应该与下面的看起来相似：



你已经成功地完成了在网格中创建一个分组区域；总结该教程。

教程18：使用ValueTranslate

在此教程中，你将学习到如何使用C1TrueDBDropDowns的ValueTranslate 属性自动地翻译从下拉框中的Detail数据到网格中的Master数据。完成下面的数据：

1. 以教程教程8：附加下拉框控件到网格单元格附加下拉框控件到网格单元格 (Section 12.8) 中创建的工程开始。
2. 在窗体的Load事件中添加下面的代码到已存在的代码中：

To write code in Visual Basic

```
Visual Basic
```

```
Me.C1TrueDBDropDown1.ValueTranslate = True
Me.C1TrueDBDropDown1.ListField = "TypeDesc"

Me.C1TrueDBDropDown1.DataField = "TypeID"
```

To write code in C#

```
C#
```

```
this.c1TrueDBDropDown1.ValueTranslate = true; this.c1TrueDBDropDown1.ListField = "TypeDesc";
this.c1TrueDBDropDown1.DataField = "TypeID";
```

运行程序并观察下面的步骤：
C1TrueDBGrid1 显示在教程教程8：附加下拉框控件到网格单元格附加下拉框控件到网格单元格 (Section 12.8)中指定的数据。
网格的CustType 列的值目前显示了一个很长的描述已显示在下拉框中。在运行时，从下拉框到网格列这些值被自动地翻译。
你已经成功地完成了使用C1TrueDBDropDowns ValueTranslate属性；总结该教程。

教程19：使用范围选择

在此教程中，你将学习到如何使用SelectedRows 和SelectedCols 对象，用一定的格式从网格拷贝一个范围，它能够被粘贴到Microsoft Excel. 完成下面的步骤：

1. 以教程教程1：绑定绑定True DBGrid到数据集到数据集 (Section 12.1) 中创建的工程开始。
2. 添加命令行按钮到窗体中，将鼠标放置窗体的左下角，并且设置它的Text 属性为"Copy".
3. 下一步添加下面的代码到Button1的Click事件：

To write code in Visual Basic

```
Visual Basic
```

```
' String to be copied to the clipboard.
Dim strTemp As String

Dim row As Integer
Dim col As C1.Win.C1TrueDBGrid.C1DataColumn
Dim cols As Integer, rows As Integer
If Me.C1TrueDBGrid1.SelectedRows.Count > 0 Then For Each row In Me.C1TrueDBGrid1.SelectedRows

' Copy everything here.
For Each col In Me.C1TrueDBGrid1.SelectedCols strTemp = strTemp & col.CellText(row) & vbTab Next
strTemp = strTemp & vbCrLf
Next

System.Windows.Forms.Clipboard.SetDataObject(strTemp, False)
MessageBox.Show ("Range of " & Me.C1TrueDBGrid1.SelectedCols.Count & " x " & C1TrueDBGrid1.SelectedRows.Count & " cells
have been copied to the clipboard in
TAB delimited format")
Else
MessageBox.Show ("Please select a range of cells")
End If
```

To write code in C#

```
C#
```

```
// String to be copied to the clipboard.

string strTemp;

int row;

C1.Win.C1TrueDBGrid.C1DataColumn col;

int cols, rows;

if (this.c1TrueDBGrid1.SelectedRows.Count > 0 )

{ foreach (row in this.c1TrueDBGrid1.SelectedRows) {

// Copy everything here.

foreach (col in this.c1TrueDBGrid1.SelectedCols)

{ strTemp = strTemp + col.CellText(row) + "\t";

}

strTemp = strTemp + "\n";

}
System.Windows.Forms.Clipboard.SetDataObject(strTemp, false);

MessageBox.Show ("Range of " + this.c1TrueDBGrid1.SelectedCols.Count.ToString() + " x " +
this.c1TrueDBGrid1.SelectedRows.Count.ToString() + " cells have been copied to the clipboard in TAB delimited format"); }

else {
MessageBox.Show ("Please select a range of cells");
}
```

运行程序并观察下面的步骤:
C1TrueDBGrid1显示在教程教程1: 绑定绑定True DBGrid到数据集到数据集 (Section 12.1)中指定的数据.
如果你选择了一个在True DBGrid中单元格范围，然后按下拷贝按钮，一个信息框将出现并描述你已经拷贝到剪切板的单元格的信息。
!worddav01e9d1bf2adcef2283a537c4e42ae008.png|height=316,width=466!现在打开Microsoft
Excel。选择一个确定的行数和列数的单元格，正如你在*True DBGrid*中中选择的，然后单
击Paste 按钮。你已经拷贝到网格中的单元格现在被粘贴到Microsoft Excel。
你已经成功地完成了使用范围选择；总结此教程。

教程20：显示多个数据视图

在此教程中，你将学习到如何使用网格的DataView属性以不同寻常的格式展示数据，如倒转视图，分组视图，窗体视图。完成下面的步骤:

1. 以教程教程1: 绑定绑定True DBGrid到数据集到数据集 (Section 12.1) 中创建的工程开始。
2. 添加一个ComboBox 控件(ComboBox1到工程，并且设置它的Text属性为"Data View".
3. 在属性窗口，通过点击紧邻着Items 属性的ellipsis 按钮来为组合框打开编辑器。在编辑器中添加下面的条目:

Normal

Inverted
Form
GroupBy
MultipleLines
Hierarchical

1. 现在为Form1的Load 事件添加下面的代码到存在的代码中:

To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.Normal

Me.ComboBox1.SelectedIndex = 0
```

To write code in C#

```
C#
this.c1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.Normal; this.comboBox1.SelectedIndex = 0;
```

1. 现在为ComboBox1的SelectedIndexChanged 事件添加下面的代码。为用户在组合框中选择的每一个值改变网格的DataView 属性:

To write code in Visual Basic

```
Visual Basic
Private Sub ComboBox1_SelectedIndexChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ComboBox1.SelectedIndexChanged
Select Case ComboBox1.SelectedItem
Case "Normal"
Me.C1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.Normal
Case "Inverted"
Me.C1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.Inverted
Case "Form"
Me.C1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.Form
Case "GroupBy"
Me.C1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.GroupBy
Case "MultipleLines"
Me.C1TrueDBGrid1.DataView =
C1.Win.C1TrueDBGrid.DataViewEnum.MultipleLines
Case "Hierarchical"
MessageBox.Show ("Hierarchical View can't be set at run time. Please see the Hierarchical Display tutorial") End Select

End Sub
```

To write code in C#

```
C#
```

```
private void ComboBox1_SelectedIndexChanged(object sender, System.EventArgs e)

{ switch (ComboBox1.SelectedItem) { case "Normal":
this.c1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.Normal; break;

case "Inverted":
this.c1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.Inverted;

break;

case "Form": this.c1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.Form;

break;

case "GroupBy":
this.c1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.GroupBy;

break;

case "MultipleLines":
this.c1TrueDBGrid1.DataView = C1.Win.C1TrueDBGrid.DataViewEnum.MultipleLines;

break;

case "Hierarchical";
MessageBox.Show ("Hierarchical View can't be set at run time. Please see the Hierarchical Display tutorial");

break; }
}
```

运行程序并观察下面的步骤:
C1TrueDBGrid1 显示在教程教程*1:* 绑定绑定*True DBGrid*到数据集到数据集 (Section 12.1)*中指定的数据 。改变组合框为*Inverted。Inverted视图显示网格的列作为行，网格的行作为列。这个网格现在将看起来与下面的一样：
!worddave1fe6fe690227bec0443078c31acadcf.png|height=313,width=461!改变组合框为*Form*。Form视图显示每一个纪录在类似窗体的视图上，这是一个最佳的数据入口。该网格现在看起来与下面的一样：
!worddav5be98623dcc916559e2b412d6d2fd1d4.png|height=313,width=461!改变组合框为GroupBy。。GroupBy视图在网格上面包含一个分组区域，该区域的列可以被拖拽。拖拽一个列到这个区域，并通过列分组网格的剩余部分。拖拽Company
列到分组区域。该网格应该看起来与下面的一样：
!worddavd9e5c205e30db2a1ff27ce26421ebc1f.png|height=313,width=461!改变组合框为*MultipleLines*。MultipleLines视图显示当前网格区域中所有的列，调整这些列显示连续的多行中。注意到有两列被挤出去显示在第2行。网格应该看起来与下面的一样：
!worddav5936cb607863c51412ddfd956c1efeca.png|height=313,width=461!现在设置组合框为Hierarchical。没有改变发生，由于Hierarchical不能在运行时设置，事件中设置的
MessageBox被弹出。分层的数据必须在应用运行之前被设置。为了在视图上获取更多的信息，参见 教程教程16: 使用使用分层的显示分层的显示 (Section 12.16)。
你已经成功地完成了显示多个数据视图；总结该教程20。

教程21：增加过滤条

在此教程中，你将学习到如何使用网格的过滤条功能允许最终用户在运行时对列数据排序。
完成下面的步骤:

1. 以教程教程1: 绑定绑定True DBGrid到数据集到数据集 (Section 12.1) 中创建的工程开始。
2. 在Form1的Load事件中存在的代码之后，添加下面的行：

To write code in Visual Basic

```
Visual Basic
Me.C1TrueDBGrid1.FilterBar = True
```

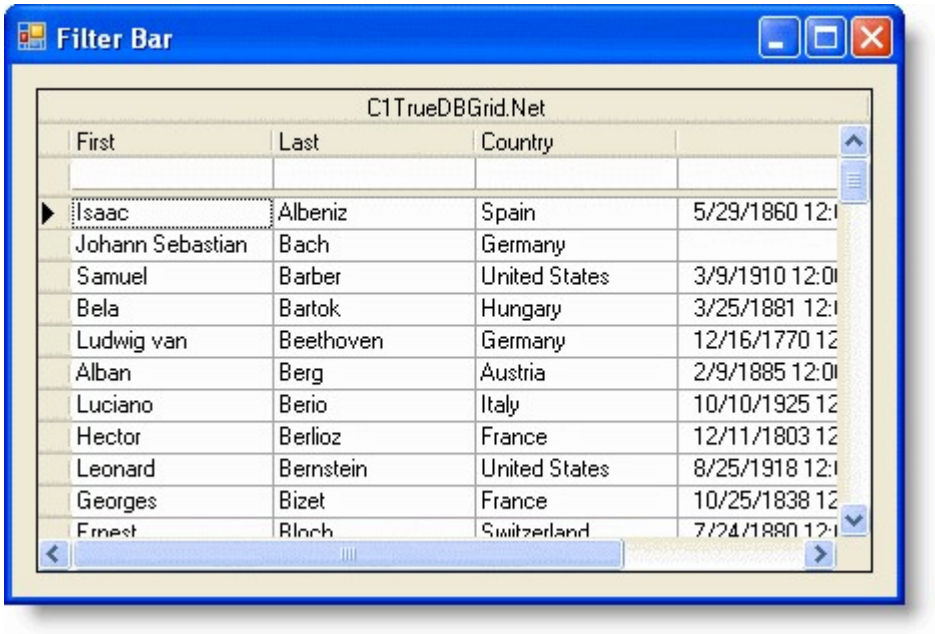
To write code in C#

```
C#
this.c1TrueDBGrid1.FilterBar = true;
```

运行程序并观察下面的步骤:

C1TrueDBGrid1显示在教程教程1: 绑定绑定True DBGrid到数据集到数据集 (Section 12.1)中指定的数据。
在网格数据上面，出现一个行接受用户的输入，这就是过滤条。当一个用户键入数据到过滤条

上，网格自动地过滤列数据。
过滤之前过滤之前:



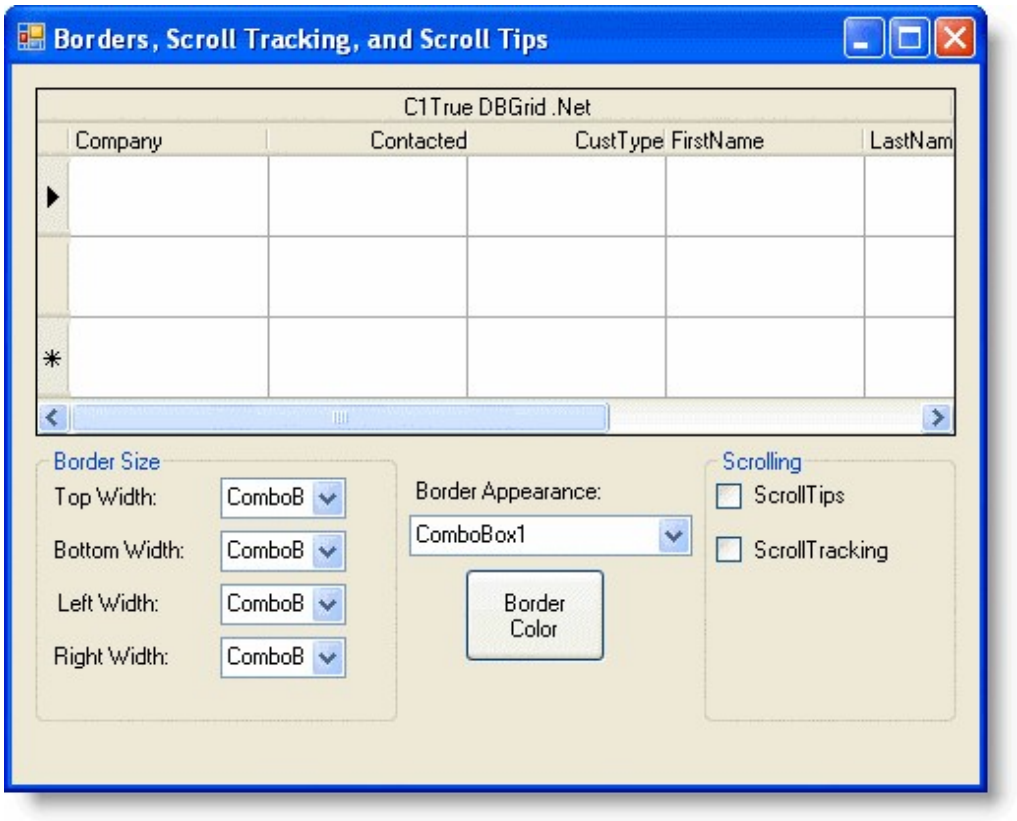
过滤之后过滤之后:
!worddav4162319b01c6654438be11313959e9c1.png|height=316,width=466!如果你想要自己处理过滤，必须修改*AllowFilter*属性为*False* (保持*FilterBar*
值为*True*)。然后你将处
理FilterChange事件，过滤条状态的每次改变都将引发该事件。你已经成功地完成了添加一个过滤条；总结此教程21。

教程22：边框，滚动跟踪和滚动提示

在此教程中，你将学习到如何调整边框，添加滚动跟踪，并且添加滚动提示到网格上。完成下面的步骤:

1. 创建一个新的工程。添加一个 C1TrueDBGrid 控件到窗体上。
2. 添加下面的条目到窗体上，并且使它们位于与下面的图片上出现的位置一样。

5个组合框(ComboBox1 - 5).
2个组框(GroupBox1 - 2) 并且分别设置它们的Text 属性值为"Border Size" 和"Scrolling"。 4个标签(Label1 - 5)
并且分别设置它们的文本属性值为"Top Width", "Bottom Width", "Left Width",
"Right Width", 和"Border Appearance" 。按钮(Button1) 并且设置它的Text属性值为"Border Color".
2个复选框并且分别设置它们的文本属性值为"ScrollTips" 和"ScrollTracking".



1. 添加一个颜色对话框颜色对话框控件到窗体上(ColorDialog1).
2. 在C1TrueDBGrid Tasks 菜单上，定位选择数据源选择数据源下拉框并且选择添加工程数据源添加工程数据源。在适配器的数据源配置向导数据源配置向导，或者选择一个连接到 C1NWind.mdb，，或者创建一个新的连接到该数据库。在向导的选择你的数据库对象选择你的数据库对象页面，选择Customer 表的所有字段，并键入"DsCustomer"到数据集名称数据集名称框，并然后完成向导。
3. 单击网格使其获得焦点，然后在属性窗口设置RowHeight属性为40。。
4. Visual Studio添加下面的代码到Form_Load 事件:

To write code in Visual Basic

```
Visual Basic
Me.CustomerTableAdapter.Fill(Me.DsCustomer.Customer)
```

To write code in C#

```
C#
this.CustomerTableAdapter.Fill(this.DsCustomer.Customer);
```

1. 在Form1中添加下面的声明：

To write code in Visual Basic

```
Visual Basic
' Copy the data.
Dim dbTable As DataTable

Dim borderColor As Color
Dim borderLeft As Integer, borderTop As Integer, borderRight As Integer, borderBottom As Integer

Dim borderType As C1.Win.C1TrueDBGrid.BorderTypeEnum
```

To write code in C#

```
C#
// Copy the data.
DataTable dbTable;
Color borderColor;
int borderLeft, int borderTop, int borderRight, int borderBottom;

C1.Win.C1TrueDBGrid.BorderTypeEnum borderType;
```

1. 在Form1的Load 事件中添加下面的代码:

To write code in Visual Basic

Visual Basic

```
dbTable = Me.DsCustomer.Tables(0).Copy() ' Fill each combobox.

FillComboBox1()
FillCombo(ComboBox2)
FillCombo(ComboBox3)
FillCombo(ComboBox4)
FillCombo(ComboBox5)
Me.CheckBox2.Checked = True
' Initalize border sizes.
Me.borderBottom = 1
Me.borderLeft = 1
Me.borderRight = 1
Me.borderTop = 1
```

To write code in C#

C#

```
dbTable = this.DsCustomer.Tables[0].Copy(); // Fill each combobox.
FillComboBox1();
FillCombo(comboBox2);
FillCombo(comboBox3);
FillCombo(comboBox4); FillCombo(comboBox5); this.checkBox2.Checked = true; // Initalize border sizes.

this.borderBottom = 1;

this.borderLeft = 1;

this.borderRight = 1;

this.borderTop = 1;
```

1. 现在添加函数它将填充组合框:

To write code in Visual Basic

Visual Basic

```
' Fill each combo with numbers from 1 to 10.
Private Sub FillCombo(ByRef com As ComboBox) Dim i As Integer com.Text = 1 For i = 1 To 10 com.Items.Add Next

End Sub
' Fill the first combo with border types.
Private Sub FillComboBox1()
Me.ComboBox1.Text = "None"
With Me.ComboBox1.Items
.Add("Fillet")
.Add("Flat")
.Add("Groove")
.Add("Inset")
.Add("InsetBevel")
.Add("None")
.Add("Raised")
.Add("RaisedBevel")
End With
End Sub
```

To write code in C#

C#

```
// Fill each combo with numbers from 1 to 10.
private void FillCombo(ref ComboBox com) { int i; com.Text = 1;
for (i = 1 ; i <= 10; i++) { com.Items.Add[I]; }

}
// Fill the first combo with border types. private void FillComboBox1() { this.comboBox1.Text = "None";
this.comboBox1.Items.Add("Fillet"); this.comboBox1.Items.Add("Flat"); this.comboBox1.Items.Add("Groove");
this.comboBox1.Items.Add("Inset"); this.comboBox1.Items.Add("InsetBevel"); this.comboBox1.Items.Add("None");
this.comboBox1.Items.Add("Raised"); this.comboBox1.Items.Add("RaisedBevel");
```

}

1. 现在为Button1的Click 事件创建一个处理器，这将通过使用颜色对话框设置边框的颜色:

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 result As DialogResult
result = Me.ColorDialog1.ShowDialog() If result = DialogResult.OK Then borderColor = Me.ColorDialog1.Color
Button1.BackColor = borderColor
End If
UpdateBorder()
End Sub
```

To write code in C#

C#

```
private void button1_Click(System.object sender, System.EventArgs e) button1.Click { DialogResult result;
result = this.colorDialog1.ShowDialog(); if (result == DialogResult.OK ) { borderColor = this.colorDialog1.Color;
button1.BackColor = borderColor; }
UpdateBorder();
}
```

1. 现在包括一个函数，它更新边框：

To write code in Visual Basic

Visual Basic

```
Private Sub UpdateBorder()
With
Me.C1TrueDBGrid1.Splits(0).DisplayColumns(Me.C1TrueDBGrid1.Col).Style.Borders

.Color = ColorDialog1.Color
.BorderType = borderType
.Bottom = borderBottom
.Left = borderLeft
.Right = borderRight
.Top = borderTop
End With
End Sub
```

To write code in C#

C#

```
  private void UpdateBorder()

{ C1.Win.C1TrueDBGrid.GridBorders b;

b = this.c1TrueDBGrid1.Splits[0].DisplayColumns(this.c1TrueDBGrid1.Col).Style.Borders;
b.Color = colorDialog1.Color;
b.BorderType = borderType;
b.Bottom = borderBottom;
b.Left = borderLeft;
b.Right = borderRight;
b.Top = borderTop; }
```

Visual Basic

```
Private Sub ComboBox1_SelectionChangeCommitted(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ComboBox1.SelectionChangeCommitted
Select Case Me.ComboBox1.SelectedItem
Case "Fillet"
Me.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.Fillet
Case "Flat"
Me.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.Flat
Case "Groove"
Me.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.Groove
Case "Inset"
Me.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.Inset
Case "InsetBevel"
Me.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.InsetBevel
Case "None"
Me.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.None
Case "Raised"
Me.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.Raised
Case "RaisedBevel"
Me.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.RaisedBevel
End Select
Me.UpdateBorder()
End Sub

Private Sub ComboBox2_SelectionChangeCommitted(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ComboBox2.SelectionChangeCommitted
Me.borderTop = Me.ComboBox2.SelectedItem
Me.UpdateBorder()
End Sub

Private Sub ComboBox3_SelectionChangeCommitted(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ComboBox3.SelectionChangeCommitted
Me.borderBottom = Me.ComboBox3.SelectedItem
Me.UpdateBorder()
End Sub

Private Sub ComboBox4_SelectionChangeCommitted(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ComboBox4.SelectionChangeCommitted
Me.borderLeft = Me.ComboBox4.SelectedItem
Me.UpdateBorder()
End Sub
Private Sub ComboBox5_SelectionChangeCommitted(ByVal sender As Object, ByVal e As
System.EventArgs) Handles ComboBox5.SelectionChangeCommitted
Me.borderRight = Me.ComboBox5.SelectedItem
Me.UpdateBorder()
End Sub
```

To write code in C#

```
C#
```

```
private void ComboBox1_SelectionChangeCommitted(object sender, System.EventArgs e) { switch (this.comboBox1.SelectedItem)
{
case "Fillet";

this.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.Fillet;

break;

case "Flat";

this.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.Flat;

break;

case "Groove";

this.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.Groove;

break;

case "Inset";

this.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.Inset;

break;

case "InsetBevel";

this.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.InsetBevel;

break;

case "None";

this.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.None;

break;

case "Raised";

this.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.Raised;

break;

case "RaisedBevel";

this.borderType = C1.Win.C1TrueDBGrid.BorderTypeEnum.RaisedBevel;

break;

}

this.UpdateBorder();

}

private void comboBox2_SelectionChangeCommitted(object sender, System.EventArgs e) { this.borderTop =
this.comboBox2.SelectedItem; this.UpdateBorder();

}

private void comboBox3_SelectionChangeCommitted(object sender, System.EventArgs e) {
this.borderBottom = this.comboBox3.SelectedItem; this.UpdateBorder(); }
private void comboBox4_SelectionChangeCommitted(object sender, System.EventArgs e) { this.borderLeft =
this.comboBox4.SelectedItem; this.UpdateBorder(); } private void comboBox5_SelectionChangeCommitted(object sender,
System.EventArgs e) {
this.borderRight = this.comboBox5.SelectedItem; this.UpdateBorder();
}
```

13. 最后包括的代码是为了处理复选框和FetchScrollTips 事件以设置当用户滚动时工具提示框的改变：
To write code in Visual Basic

```
Visual Basic
```

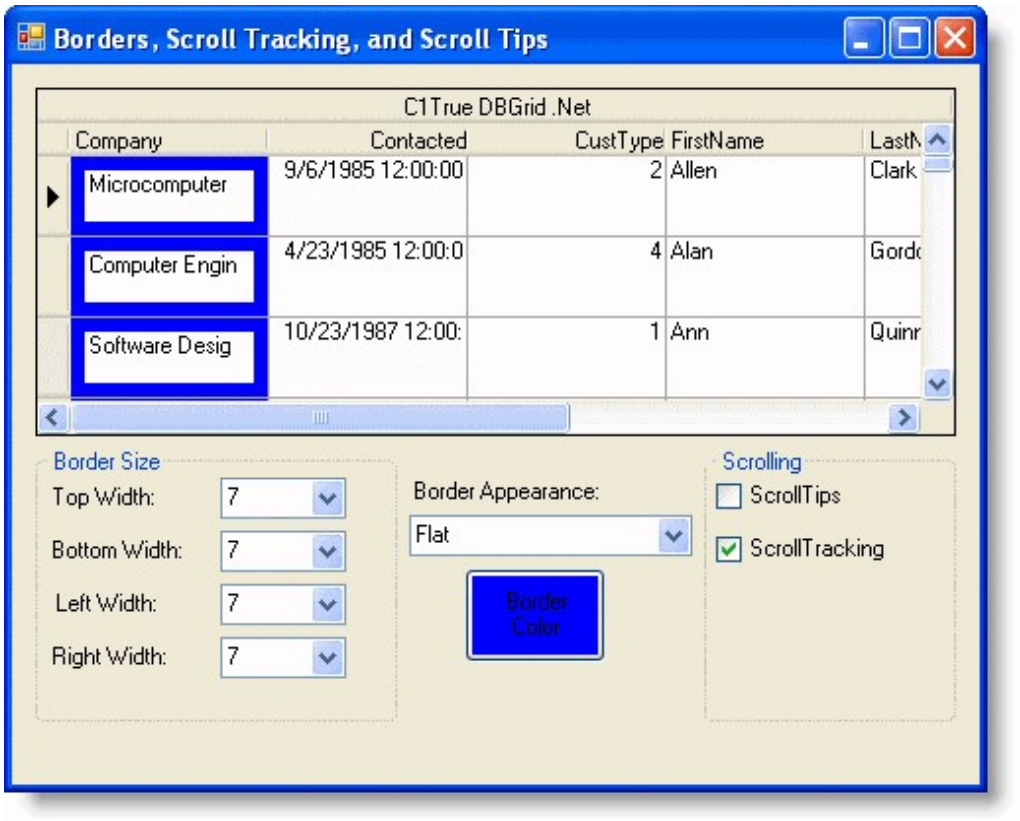
```
Private Sub CheckBox1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles CheckBox1.Click
Me.C1TrueDBGrid1.ScrollTips = Me.CheckBox1.Checked
End Sub

Private Sub CheckBox2_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles CheckBox2.Click
Me.C1TrueDBGrid1.ScrollTrack = Me.CheckBox2.Checked
End Sub

Private Sub C1TrueDBGrid1_FetchScrollTips(ByVal sender As System.Object, ByVal e
As C1.Win.C1TrueDBGrid.FetchScrollTipsEventArgs) Handles
C1TrueDBGrid1.FetchScrollTips

' Set the ScrollTip depending on which scroll bar was moved.
Select Case e.ScrollBar Case C1.Win.C1TrueDBGrid.ScrollBarEnum.Horizontal
e.ScrollTip = Me.C1TrueDBGrid1.Columns(e.ColIndex).Caption Case C1.Win.C1TrueDBGrid.ScrollBarEnum.Vertical

e.ScrollTip = "Record: " & CStr(e.Row + 1) & " of " & CStr(Me.dbTable.Rows.Count) & vbCrLf & "Company: " &

Me.dbTable.Rows(e.Row).Item("Company") & vbCrLf & "User code: " &
Me.dbTable.Rows(e.Row).Item("UserCode") End Select
e.TipStyle.ForeColor = Color.Blue
End Sub
```

To write code in C#

```
C#
```

```
private void checkBox1_Click(object sender, System.EventArgs e) { this.c1TrueDBGrid1.ScrollTips = this.checkBox1.Checked;
} private void checkBox2_Click(object sender, System.EventArgs e) {
this.c1TrueDBGrid1.ScrollTrack = this.checkBox2.Checked; }
private void c1TrueDBGrid1_FetchScrollTips(System.object sender,
C1.Win.C1TrueDBGrid.FetchScrollTipsEventArgs e)
{
// Set the ScrollTip depending on which scroll bar was moved. switch (e.ScrollBar) { case
C1.Win.C1TrueDBGrid.ScrollBarEnum.Horizontal:
e.ScrollTip = this.c1TrueDBGrid1.Columns[e.ColIndex].Caption; break; case C1.Win.C1TrueDBGrid.ScrollBarEnum.Vertical:

e.ScrollTip = "Record: " + (e.Row + 1).ToString() + " of " + this.dbTable.Rows.Count.ToString() + "\n" + "Company: " +
this.dbTable.Rows[e.Row] ["Company"].ToString() + "\n" + "User code: " + this.dbTable.Rows[e.Row] ["UserCode"].ToString();
break; }

e.TipStyle.ForeColor = Color.Blue;
}
```

运行程序并观察下面的步骤:
C1TrueDBGrid1显示指定的数据。
!worddav729bdd0411865672e43c5f10e34e708f.png|height=63,width=228!设置ScrollTracks值为 True，并使你可以看到当它被滚动时的值。
设置ScrollTips值为True，并在用户滚动时显示列信息的工具提示框。
通过模拟组合框和颜色对话框，创建一个边框围绕的列的单元格，并且设置它们的系统色。



你已经成功地调整了边框，添加了滚动跟踪，并添加了滚动提示到网格上；总结此教程22。