DataGrid Binding: Custom DataGrid Column Form

```
/*
User Interfaces in C#: Windows Forms and Custom Controls
by Matthew MacDonald

Publisher: Apress
ISBN: 1590590457
*/
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace DataGridBinding
{
///

/// Summary description for CustomDataGridColumnForm.
///
public class CustomDataGridColumnForm :
System.Windows.Forms.Form
{
internal System.Windows.Forms.DataGrid grid;
///

/// Required designer variable.
///
private System.ComponentModel.Container components = null;

public CustomDataGridColumnForm()
{
```

```
//
// Required for Windows Form Designer support
//
InitializeComponent();

//
// TODO: Add any constructor code after InitializeComponent
call
//
}

///

/// Clean up any resources being used.
///
protected override void Dispose( bool disposing )
{
if( disposing )
{
if(components != null)
{
components.Dispose();
}
}
base.Dispose( disposing );
}

#region Windows Form Designer generated code
///

/// Required method for Designer support – do not modify
/// the contents of this method with the code editor.
///
private void InitializeComponent()
{
this.grid = new System.Windows.Forms.DataGrid();
((System.ComponentModel.ISupportInitialize)(this.grid)).BeginI
nit();
this.SuspendLayout();
```

```
//
// grid
//
this.grid.Anchor = (((System.Windows.Forms.AnchorStyles.Top |
System.Windows.Forms.AnchorStyles.Bottom)
| System.Windows.Forms.AnchorStyles.Left)
| System.Windows.Forms.AnchorStyles.Right);
this.grid.DataMember = “”;
this.grid.HeaderForeColor                                    =
System.Drawing.SystemColors.ControlText;
this.grid.Location = new System.Drawing.Point(10, 17);
this.grid.Name = “grid”;
this.grid.Size = new System.Drawing.Size(272, 232);
this.grid.TabIndex = 1;
//
// CustomDataGridColumnForm
//
this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
this.ClientSize = new System.Drawing.Size(292, 266);
this.Controls.AddRange(new System.Windows.Forms.Control[] {
this.grid});
this.Name = “CustomDataGridColumnForm”;
this.Text = “CustomDataGridColumnForm”;
this.Load                        +=                        new
System.EventHandler(this.CustomDataGridColumnForm_Load);
((System.ComponentModel.ISupportInitialize)(this.grid)).EndIni
t();
this.ResumeLayout(false);

}
#endregion

[STAThread]
static void Main()
{
Application.Run(new CustomDataGridColumnForm());
}
```

```
private void CustomDataGridColumnForm_Load(object sender,
System.EventArgs e)
{
DataSet dsStore = new DataSet();

dsStore.ReadXmlSchema(Application.StartupPath + “store.xsd”);
dsStore.ReadXml(Application.StartupPath + “store.xml”);

// Create the column collection.
DataGridTableStyle columns = new DataGridTableStyle();
columns.MappingName = “Products”;

// Create and configure the columns you want to display.
DataGridPriceIconColumn priceCol = new
DataGridPriceIconColumn(100);
priceCol.HeaderText = “Price”;
priceCol.MappingName = “UnitCost”;

// Add the columns to the collection.
columns.GridColumnStyles.Add(priceCol);

// Configure the DataGrid to use these column settings.
grid.TableStyles.Add(columns);

grid.ReadOnly = true;

// Bind the grid.
grid.DataSource = dsStore.Tables[“Products”];

}
}

public class DataGridPriceIconColumn : DataGridColumnStyle
{
public decimal NicePrice;

public DataGridPriceIconColumn(decimal nicePrice)
{
this.NicePrice = nicePrice;
}
```

```
protected override void Abort(int rowNum)
{
// Do nothing.
}

protected override bool Commit(CurrencyManager dataSource, int
rowNum)
{
return true;
}

protected override void Edit(CurrencyManager source,
int rowNum, System.Drawing.Rectangle bounds,
bool readOnly, string instantText, bool cellIsVisible)
{
// Do nothing.
}

protected override void Edit(CurrencyManager source,
int rowNum, System.Drawing.Rectangle bounds, bool readOnly)
{
// Do nothing.
}

protected override void Edit(CurrencyManager source,
int rowNum, System.Drawing.Rectangle bounds,
bool readOnly, string instantText)
{
// Do nothing.
}
protected override int GetMinimumHeight()
{
return 20;
}

protected override int
GetPreferredHeight(System.Drawing.Graphics g,
object value)
```

```
{
return 20;
}

protected override System.Drawing.Size GetPreferredSize(
System.Drawing.Graphics g, object value)
{
return new Size(100, 20);
}

protected override void Paint(System.Drawing.Graphics g,
System.Drawing.Rectangle bounds, CurrencyManager source, int
rowNum,
System.Drawing.Brush backBrush, System.Drawing.Brush
foreBrush,
bool alignToRight)
{
// Clear the cell.
g.FillRegion(backBrush, new Region(bounds));

decimal price = (decimal)this.GetColumnValueAtRow(source,
rowNum);
Icon priceIcon;
if (price < NicePrice) { priceIcon = new
Icon(Application.StartupPath + "happy2.ico"); // Draw the
optional "nice price" icon. g.DrawIcon(priceIcon, new
Rectangle(bounds.X, bounds.Y, 16, 16)); } // Draw the text.
g.DrawString(price.ToString("C"), new Font("Tahoma",
(float)8.25), Brushes.Black, bounds.X + 20, bounds.Y + 2); }
protected override void Paint(System.Drawing.Graphics g,
System.Drawing.Rectangle bounds, CurrencyManager source, int
rowNum, bool alignToRight) { this.Paint(g, bounds, source,
rowNum, Brushes.White, Brushes.Black, alignToRight); }
protected override void Paint(System.Drawing.Graphics g,
System.Drawing.Rectangle bounds, CurrencyManager source, int
rowNum) { this.Paint(g, bounds, source, rowNum, Brushes.White,
Brushes.Black, false); } } } DataGridBinding.zip( 45
k)[/csharp]
```