# profit2u的博客

http://blog.sina.com.cn/profit2u ［订阅］［手机订阅］



个人资料

profit2u

profit2u

微博

加好友 发纸条

写留言 加关注

博客地图 world map

博客等级：16
博客积分：758
博客访问：112,150
关注人气：23
获赠金笔：1
赠出金笔：0
荣誉徽章：

相关博文

狗在什么情况下会伤人，该如何防
伯纳天纯

葡萄酒美人
用户360220640

2016中英文新版《奇迹课程学员练
真理之言

《一条狗的使命》：不贩卖苦情戏
崔汀

高性价比|2015年穆利内拉塞尔——
戴维-shdavid

如道的法华比喻（七）
禅门布衣

一线城市财务自由2.9个亿真的一
老龙在天

【七绝】冬养君子兰感喟——柳岸
柳岸----

正文 字体大小：大 中 小

## C# winform DataGridView 操作大全 (2012-09-04 09:59:08)

转 载 ▼

标签： 单元格 设定 新行 属性 行头 it 分类： VBvb.NET

## C# DataGridView控件动态添加新行

DataGridView控件在实际应用中非常实用，特别需要表格显示数据时。可以静态绑定数据源，这样就自动为DataGridView控件添加相应的行。假如需要动态为DataGridView控件添加新行，方法有很多种，下面简单介绍如何为DataGridView控件动态添加新行的两种方法：

方法一：

```
int index=this.dataGridView1.Rows.Add();
this.dataGridView1.Rows[index].Cells[0].Value = "1";
this.dataGridView1.Rows[index].Cells[1].Value = "2";
this.dataGridView1.Rows[index].Cells[2].Value = "监听";
```

利用dataGridView1.Rows.Add()事件为DataGridView控件增加新的行，该函数返回添加新行的索引号，即新行的行号，然后可以通过该索引号操作该行的各个单元格，如dataGridView1.Rows[index].Cells[0].Value = "1"。这是很常用也是很简单的方法。

方法二：

```
DataGridViewRow row = new DataGridViewRow();
DataGridViewTextBoxCell textboxcell = new DataGridViewTextBoxCell();
textboxcell.Value = "aaa";
row.Cells.Add(textboxcell);
DataGridViewComboBoxCell comboxcell = new DataGridViewComboBoxCell();
row.Cells.Add(comboxcell);
dataGridView1.Rows.Add(row);
```

方法二比方法一要复杂一些，但是在一些特殊场合非常实用，例如，要在新行中的某些单元格添加下拉框、按钮之类的控件时，该方法很有帮助。

DataGridViewRow row = new DataGridViewRow();是创建DataGridView的行对象，DataGridViewTextBoxCell是单元格的内容是个TextBox，DataGridViewComboBoxCell是单元格的内容是下拉列表框，同理可知，DataGridViewButtonCell是单元格的内容是个按钮，等等。textboxcell是新创建的单元格的对象，可以为该对象添加其属性。然后通过row.Cells.Add(textboxcell)为row对象添加textboxcell单元格。要添加其他的单元格，用同样的方法即可。

最后通过dataGridView1.Rows.Add(row)为dataGridView1控件添加新的行row。

## ① DataGridView 取得或者修改当前单元格的内容：

得。如果当前单元格不存在的时候，返回Nothing(C#是null)

```
// 取得当前单元格内容
Console.WriteLine(DataGridView1.CurrentCell.Value);
// 取得当前单元格的列 Index
Console.WriteLine(DataGridView1.CurrentCell.ColumnIndex);
// 取得当前单元格的行 Index
Console.WriteLine(DataGridView1.CurrentCell.RowIndex);
```

另外，使用 DataGridView.CurrentCellAddress 属性（而不是直接访问单元格）来确定单元格所在的
行： DataGridView.CurrentCellAddress.Y
列： DataGridView.CurrentCellAddress.X 。这对于避免取消共享行的共享非常有用。

当前的单元格可以通过设定 DataGridView 对象的 CurrentCell 来改变。可以通过 CurrentCell 来设定 DataGridView 的激活单元格。将 CurrentCell 设为 Nothing(null) 可以取消激活的单元格。

---

```
// 设定 (0, 0) 为当前单元格
DataGridView1.CurrentCell = DataGridView1[0, 0];
```

在整行选中模式开启时，你也可以通过 CurrentCell 来设定选定行。

```
        /// 向下遍历
        private void button4_Click(object sender, EventArgs e)
        ...{
                int row = this.dataGridView1.CurrentRow.Index + 1;
                if (row > this.dataGridView1.RowCount - 1)
                        row = 0;
                this.dataGridView1.CurrentCell = this.dataGridView1[0, row];
        }
        /// 向上遍历
        private void button5_Click(object sender, EventArgs e)
        ...{
                int row = this.dataGridView1.CurrentRow.Index - 1;
                if (row < 0)
                        row = this.dataGridView1.RowCount - 1;
                this.dataGridView1.CurrentCell = this.dataGridView1[0, row];
        }
```

* 注意: this.dataGridView 的索引器的参数是: columnIndex, rowIndex 或是 columnName, rowIndex 这与习惯不同。

---

② DataGridView 设定单元格只读：

1） 使用 ReadOnly 属性

如果希望，DataGridView 内所有单元格都不可编辑， 那么只要：

```
// 设置 DataGridView1 为只读
DataGridView1.ReadOnly = true;
```

此时，用户的新增行操作和删除行操作也被屏蔽了。

如果希望，DataGridView 内某个单元格不可编辑， 那么只要：

```
// 设置 DataGridView1 的第2列整列单元格为只读
DataGridView1.Columns[1].ReadOnly = true;
// 设置 DataGridView1 的第3行整行单元格为只读
DataGridView1.Rows[2].ReadOnly = true;
// 设置 DataGridView1 的[0, 0]单元格为只读
DataGridView1[0, 0].ReadOnly = true;
```

2） 使用 EditMode 属性

DataGridView.EditMode 属性被设置为 DataGridViewEditMode.EditProgrammatically 时，用户就不能手动编辑单元格的内容了。但是可以通过程序，调用 DataGridView.BeginEdit 方法，使单元格进入编辑模式进行编辑。

```
DataGridView1.EditMode = DataGridViewEditMode.EditProgrammatically;
```

3） 根据条件设定单元格的不可编辑状态

当一个一个的通过单元格坐标设定单元格 ReadOnly 属性的方法太麻烦的时候，你可以通过 CellBeginEdit 事件来取消单元格的编辑。

```
// CellBeginEdit 事件处理方法
private void DataGridView1_CellBeginEdit(object sender, DataGridViewCellCancelEventArgs e)
{
        DataGridView dgv = (DataGridView)sender;
        //是否可以进行编辑的条件检查
```

```
e.RowIndex].Value)
        {
                // 取消编辑
                e.Cancel = true;
        }
}
```

---

③ DataGridView 不显示最下面的新行：

通常 DataGridView 的最下面一行是用户新追加的行（行头显示 * ）。如果不想让用户新追加行即不想显示该新行，可以将 DataGridView 对象的 AllowUserToAddRows 属性设置为 False。

```
// 设置用户不能手动给 DataGridView1 添加新行
DataGridView1.AllowUserToAddRows = false;
```

但是，可以通过程序： DataGridViewRowCollection.Add 为 DataGridView 追加新行。

补足： 如果 DataGridView 的 DataSource 绑定的是 DataView，还可以通过设置 DataView.AllowAdd 属性为 False 来达到同样的效果。

---

④ DataGridView 判断新增行：

DataGridView 的AllowUserToAddRows属性为True时也就是允许用户追加新行的场合下，DataGridView的最后一行就是新追加的行(*行)。
使用 DataGridViewRow.IsNewRow 属性可以判断哪一行是新追加的行。另外，通过DataGridView.NewRowIndex 可以获取新行的行序列号.
在没有新行的时候，NewRowIndex = -1。

---

⑤ DataGridView 行的用户删除操作的自定义：

1） 无条件的限制行删除操作。
默认时，DataGridView 是允许用户进行行的删除操作的。如果设置 DataGridView对象的 AllowUserToDeleteRows属性为 False 时， 用户的行删除操作就被禁止了。

```
// 禁止DataGridView1的行删除操作。
DataGridView1.AllowUserToDeleteRows = false;
```

但是，通过 DataGridViewRowCollection.Remove 还是可以进行行的删除。
补足： 如果 DataGridView 绑定的是 DataView 的话，通过 DataView.AllowDelete 也可以控制行的删除。

2） 行删除时的条件判断处理。
用户在删除行的时候，将会引发 DataGridView.UserDeletingRow 事件。 在这个事件里，可以判断条件并取消删除操作。

```
// DataGridView1 的 UserDeletingRow 事件
private void DataGridView1_UserDeletingRow( object sender, DataGridViewRowCancelEventArgs e)
{
        // 删除前的用户确认。
        if (MessageBox.Show("确认要删除该行数据吗？", "删除确认",
                MessageBoxButtons.OKCancel, MessageBoxIcon.Question) != DialogResult.OK)
        {
                // 如果不是 OK，则取消。
                e.Cancel = true;
        }
}
```

---

⑥ DataGridView 行、列的隐藏和删除：

1） 行、列的隐藏

```
// DataGridView1的第一列隐藏
DataGridView1.Columns[0].Visible = false;

// DataGridView1的第一行隐藏
DataGridView1.Rows[0].Visible = false;
```

2） 行头、列头的隐藏

```
// 列头隐藏
DataGridView1.ColumnHeadersVisible = false;

// 行头隐藏
DataGridView1.RowHeadersVisible = false;
```

3） 行和列的删除

```
' 删除名为"Column1"的列
DataGridView1.Columns.Remove("Column1");
```

```
DataGridView1.Columns.RemoveAt(0);
' 删除第一行
DataGridView1.Rows.RemoveAt(0);
```

4） 删除选中行

```
foreach (DataGridViewRow r in DataGridView1.SelectedRows)
{
        if (!r.IsNewRow)
        {
                DataGridView1.Rows.Remove(r);
        }
}
```

---

⑦ DataGridView 禁止列或者行的Resize：

1） 禁止所有的列或者行的Resize

```
// 禁止用户改变DataGridView1的所有列的列宽
DataGridView1.AllowUserToResizeColumns = false;
//禁止用户改变DataGridView1の所有行的行高
DataGridView1.AllowUserToResizeRows = false;
```

但是可以通过 DataGridViewColumn.Width 或者 DataGridViewRow.Height 属性设定列宽和行高。

2） 禁止指定行或者列的Resize

```
// 禁止用户改变DataGridView1的第一列的列宽
DataGridView1.Columns[0].Resizable = DataGridViewTriState.False;
// 禁止用户改变DataGridView1的第一列的行宽
DataGridView1.Rows[0].Resizable = DataGridViewTriState.False;
```

关于 NoSet ：
当 Resizable 属性设为 DataGridViewTriState.NotSet 时， 实际上会默认以 DataGridView 的 AllowUserToResizeColumns 和 AllowUserToResizeRows 的属性值进行设定。
比如： DataGridView.AllowUserToResizeColumns = False 且 Resizable 是 NoSet 设定时，Resizable = False 。

判断 Resizable 是否是继承设定了 DataGridView 的 AllowUserToResizeColumns 和 AllowUserToResizeRows 的属性值，
可以根据 State 属性判断。如果 State 属性含有 ResizableSet，那么说明没有继承设定。

3） 列宽和行高的最小值的设定

```
// 第一列的最小列宽设定为 100
DataGridView1.Columns[0].MinimumWidth = 100;
// 第一行的最小行高设定为 50
DataGridView1.Rows[0].MinimumHeight = 50;
```

4) 禁止用户改变行头的宽度以及列头的高度

```
// 禁止用户改变列头的高度
DataGridView1.ColumnHeadersHeightSizeMode =
        DataGridViewColumnHeadersHeightSizeMode.DisableResizing;
// 设置用户改变行头的宽度
DataGridView1.RowHeadersWidthSizeMode =
        DataGridViewRowHeadersWidthSizeMode.EnableResizing;
```

---

**⑧ DataGridView 列宽和行高自动调整的设定：**

1) 设定行高和列宽自动调整

```
// 设定包括Header和所有单元格的列宽自动调整
DataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells;
// 设定包括Header和所有单元格的行高自动调整
DataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells;
```

AutoSizeColumnsMode 属性的设定值枚举请参照 msdn 的 DataGridViewAutoSizeRowsMode 说明。

2）指定列或行自动调整

```
// 第一列自动调整
DataGridView1.Columns[0].AutoSizeMode =
        DataGridViewAutoSizeColumnMode.DisplayedCells;
```

AutoSizeMode 设定为 NotSet 时， 默认继承的是 DataGridView.AutoSizeColumnsMode 属性。

```
// 设定列头的宽度可以自由调整
DataGridView1.ColumnHeadersHeightSizeMode =
        DataGridViewColumnHeadersHeightSizeMode.AutoSize;
// 设定行头的宽度可以自由调整
DataGridView1.RowHeadersWidthSizeMode =
        DataGridViewRowHeadersWidthSizeMode.AutoSizeToAllHeaders;
```

4） 随时自动调整

a， 临时的，让列宽自动调整，这和指定AutoSizeColumnsMode属性一样。

```
// 让 DataGridView1 的所有列宽自动调整一下。
DataGridView1.AutoResizeColumns(DataGridViewAutoSizeColumnsMode.AllCells);
// 让 DataGridView1 的第一列的列宽自动调整一下。
DataGridView1.AutoResizeColumn(0, DataGridViewAutoSizeColumnMode.AllCells);上面调用的
AutoResizeColumns 和 AutoResizeColumn
```

当指定的是DataGridViewAutoSizeColumnMode.AllCells 的时候，参数可以省略。即：

```
DataGridView1.AutoResizeColumn(0) 和 DataGridView1.AutoResizeColumns()
```

b，临时的，让行高自动调整

```
// 让 DataGridView1 的所有行高自动调整一下。
DataGridView1.AutoResizeRows(DataGridViewAutoSizeRowsMode.AllCells);
//让 DataGridView1 的第一行的行高自动调整一下。
DataGridView1.AutoResizeRow(0, DataGridViewAutoSizeRowMode.AllCells);上面调用的 AutoResizeRows 和
AutoResizeRow
```

当指定的是DataGridViewAutoSizeRowMode.AllCells 的时候，参数可以省略。即：

```
DataGridView1.AutoResizeRow (0) 和 DataGridView1.AutoResizeRows()
```

c，临时的，让行头和列头自动调整

```
// 列头高度自动调整
DataGridView1.AutoResizeColumnHeadersHeight();
// 行头宽度自动调整
DataGridView1.AutoResizeRowHeadersWidth(
        DataGridViewRowHeadersWidthSizeMode.AutoSizeToAllHeaders);
```

关于性能：

通过 AutoSizeColumnsMode 或者 AutoSizeRowsMode 属性所指定的单元格进行自动调整时，如果调整次数过于多那么将可能导致性能下降，

尤其是在行和列数比较多的情况下。在这时用 DisplayedCells 代替 AllCells 能减少非所见的单元格的调整，从而提高性能。

---

**⑨ DataGridView 冻结列或行**

1） 列冻结

DataGridViewColumn.Frozen 属性为 True 时， 该列左侧的所有列被固定， 横向滚动时固定列不随滚动条滚动而左右移动。这对于重要列固定显示很有用。

```
// DataGridView1的左侧2列固定
DataGridView1.Columns[1].Frozen = true;
```

但是，DataGridView.AllowUserToOrderColumns = True 时，固定列不能移动到非固定列， 反之亦然。

2） 行冻结

DataGridViewRow.Frozen 属性为 True 时， 该行上面的所有行被固定， 纵向滚动时固定行不随滚动条滚动而上下移动。

```
// DataGridView1 的上3行固定
DataGridView1.Rows[2].Frozen = true;
```

---

⑩ DataGridView 列顺序的调整

设定 DataGridView 的 AllowUserToOrderColumns 为 True 的时候， 用户可以自由调整列的顺序。

当用户改变列的顺序的时候，其本身的 Index 不会改变，但是 DisplayIndex 改变了。你也可以通过程序改变 DisplayIndex 来改变列的顺序。 列顺序发生改变时会引发 ColumnDisplayIndexChanged 事件：

```
// DataGridView1的ColumnDisplayIndexChanged事件处理方法
private void DataGridView1_ColumnDisplayIndexChanged(object sender,
        DataGridViewColumnEventArgs e)
{
        Console.WriteLine("{0} 的位置改变到 {1} ",
                e.Column.Name, e.Column.DisplayIndex);
}
```

⑪ 行头列头的单元格

[C#]

```
// 改变DataGridView1的第一列列头内容
DataGridView1.Columns[0].HeaderCell.Value = "第一列";
// 改变DataGridView1的第一行行头内容
DataGridView1.Rows[0].HeaderCell.Value = "第一行";
// 改变DataGridView1的左上头部单元内容
DataGridView1.TopLeftHeaderCell.Value = "左上";
```

另外你也可以通过 HeaderText 来改变他们的内容。

[C#]

```
// 改变DataGridView1的第一列列头内容
DataGridView1.Columns[0].HeaderText = "第一列";
```

⑫ DataGridView 剪切板的操作

TOP

DataGridView.ClipboardCopyMode 属性被设定为 DataGridViewClipboardCopyMode.Disable 以外的情况时，「Ctrl + C」 按下的时候，被选择的单元格的内容会拷贝到系统剪切板内。格式有： Text， UnicodeText，Html， CommaSeparatedValue。可以直接粘贴到 Excel 内。

ClipboardCopyMode 还可以设定 Header部分是否拷贝： EnableAlwaysIncludeHeaderText 拷贝Header部分、EnableWithoutHeaderText 则不拷贝。默认是 EnableWithAutoHeaderText ， Header 如果选择了的话，就拷贝。

1） 编程方式实现剪切板的拷贝

```
Clipboard.SetDataObject(DataGridView1.GetClipboardContent())
```

2) DataGridView 的数据粘贴

实现剪切板的拷贝比较容易，但是实现 DataGridView 的直接粘贴就比较难了。「Ctrl + V」按下进行粘贴时，DataGridView 没有提供方法，只能自己实现。

以下，是粘贴时简单的事例代码，将拷贝数据粘贴到以选择单元格开始的区域内。

[C#]

```
//当前单元格是否选择的判断
if (DataGridView1.CurrentCell == null)
        return;
int insertRowIndex = DataGridView1.CurrentCell.RowIndex;
// 获取剪切板的内容，并按行分割
string pasteText = Clipboard.GetText();
if (string.IsNullOrEmpty(pasteText))
        return;
pasteText = pasteText.Replace("", "");
pasteText = pasteText.Replace(' ', ' ');
pasteText.TrimEnd(new char[] { ' ' });
string[] lines = pasteText.Split(' ');
bool isHeader = true;
foreach (string line in lines)
{
        // 是否是列头
        if (isHeader)
        {
                isHeader = false;
                continue;
        }
        // 按 Tab 分割数据
        string[] vals = line.Split(' ');
        // 判断列数是否统一
        if (vals.Length - 1 != DataGridView1.ColumnCount)
                throw new ApplicationException("粘贴的列数不正确。");
        DataGridViewRow row = DataGridView1.Rows[insertRowIndex];
        // 行头设定
        row.HeaderCell.Value = vals[0];
        // 单元格内容设定
        for (int i = 0; i < row.Cells.Count; i++)
        {
                row.Cells[i].Value = vals[i + 1];
        }
```

```
        InsertRowIndex++;
}
```

---

### ⑬ DataGridView 单元格的ToolTip的设置

DataGridView.ShowCellToolTips = True 的情况下， 单元格的 ToolTip 可以表示出来。对于单元格窄小，无法完全显示的单元格， ToolTip 可以显示必要的信息。

1） 设定单元格的ToolTip内容

[C#]

```
// 设定单元格的ToolTip内容
DataGridView1[0, 0].ToolTipText = "该单元格的内容不能修改";
// 设定列头的单元格的ToolTip内容
DataGridView1.Columns[0].ToolTipText = "该列只能输入数字";
// 设定行头的单元格的ToolTip内容
DataGridView1.Rows[0].HeaderCell.ToolTipText = "该行单元格内容不能修改";
```

2） CellToolTipTextNeeded 事件

在批量的单元格的 ToolTip 设定的时候，一个一个指定那么设定的效率比较低， 这时候可以利用 CellToolTipTextNeeded 事件。当单元格的 ToolTipText 变化的时候也会引发该事件。但是，当DataGridView的DataSource被指定且VirualMode=True的时候，该事件不会被引发。

[C#]

```
// CellToolTipTextNeeded事件处理方法
private void DataGridView1_CellToolTipTextNeeded(object sender,
        DataGridViewCellToolTipTextNeededEventArgs e)
{
        e.ToolTipText = e.ColumnIndex.ToString() + ", " + e.RowIndex.ToString();
}
```

---

### ⑭ DataGridView 的右键菜单（ContextMenuStrip）

DataGridView, DataGridViewColumn, DataGridViewRow, DataGridViewCell 有 ContextMenuStrip 属性。可以通过设定 ContextMenuStrip 对象来控制 DataGridView 的右键菜单的显示。 DataGridViewColumn 的 ContextMenuStrip 属性设定了 除了列头以外的单元格的右键菜单。 DataGridViewRow 的 ContextMenuStrip 属性设定了除了行头以外的单元格的右键菜单。DataGridViewCell 的 ContextMenuStrip 属性设定了指定单元格的右键菜单。

[C#]

```
// DataGridView 的 ContextMenuStrip 设定
DataGridView1.ContextMenuStrip = this.ContextMenuStrip1;
// 列的 ContextMenuStrip 设定
DataGridView1.Columns[0].ContextMenuStrip = this.ContextMenuStrip2;
// 列头的 ContextMenuStrip 设定
DataGridView1.Columns[0].HeaderCell.ContextMenuStrip = this.ContextMenuStrip2;
// 行的 ContextMenuStrip 设定
DataGridView1.Rows[0].ContextMenuStrip = this.ContextMenuStrip3;
// 单元格的 ContextMenuStrip 设定
DataGridView1[0, 0].ContextMenuStrip = this.ContextMenuStrip4;
```

对于单元格上的右键菜单的设定，优先顺序是： Cell > Row > Column > DataGridView

⇒ CellContextMenuStripNeeded、RowContextMenuStripNeeded 事件

利用 CellContextMenuStripNeeded 事件可以设定单元格的右键菜单，尤其但需要右键菜单根据单元格值的变化而变化的时候。比起使用循环遍历，使用该事件来设定右键菜单的效率更高。但是，在DataGridView使用了DataSource绑定而且是VirtualMode的时候，该事件将不被引发。

[C#]

```
//  CellContextMenuStripNeeded事件处理方法
private void DataGridView1_CellContextMenuStripNeeded(object sender,
        DataGridViewCellContextMenuStripNeededEventArgs e)
{
        DataGridView dgv = (DataGridView)sender;
        if (e.RowIndex < 0)
        {
                //  列头的ContextMenuStrip设定
                e.ContextMenuStrip = this.ContextMenuStrip1;
        }
        else if (e.ColumnIndex < 0)
```

```
                //  行头的ContextMenuStrip设定
                e.ContextMenuStrip = this.ContextMenuStrip2;
        }
        else if (dgv[e.ColumnIndex, e.RowIndex].Value is int)
        {
                //  如果单元格值是整数时
                e.ContextMenuStrip = this.ContextMenuStrip3;
        }
}
```

同样，可以通过 RowContextMenuStripNeeded 事件来设定行的右键菜单。

```
[C#]
//  RowContextMenuStripNeeded事件处理方法
private void DataGridView1_RowContextMenuStripNeeded(object sender,
        DataGridViewRowContextMenuStripNeededEventArgs e)
{
        DataGridView dgv = (DataGridView)sender;
        //  当"Column1"列是Bool型且为True时、设定其的ContextMenuStrip
        object boolVal = dgv["Column1", e.RowIndex].Value;
        Console.WriteLine(boolVal);
        if (boolVal is bool && (bool)boolVal)
        {
                e.ContextMenuStrip = this.ContextMenuStrip1;
        }
}
```

CellContextMenuStripNeeded 事件处理方法的参数中、「e.ColumnIndex=-1」表示行头、「e.RowIndex=-1」表示列头。RowContextMenuStripNeeded则不存在「e.RowIndex=-1」的情况。

---

**⑮ DataGridView 的单元格的边框、网格线样式的设定**

1) DataGridView 的边框线样式的设定

DataGridView 的边框线的样式是通过 DataGridView.BorderStyle 属性来设定的。 BorderStyle 属性设定值是一个

BorderStyle 枚举： FixedSingle（单线，默认）、Fixed3D、None。

2) 单元格的边框线样式的设定

单元格的边框线的样式是通过 DataGridView.CellBorderStyle 属性来设定的。 CellBorderStyle 属性设定值是

DataGridViewCellBorderStyle 枚举。（详细参见 MSDN）

另外，通过 DataGridView.ColumnHeadersBorderStyle 和 RowHeadersBorderStyle 属性可以修改 DataGridView 的头部的单元格边框线样式。 属性设定值是 DataGridViewHeaderBorderStyle 枚举。（详细参见 MSDN）

3） 单元格的边框颜色的设定

单元格的边框线的颜色可以通过 DataGridView.GridColor 属性来设定的。默认是 ControlDarkDark 。但是只有在 CellBorderStyle 被设定为 Single、SingleHorizontal、SingleVertical 的条件下才能改变其边框线的颜色。同样，ColumnHeadersBorderStyle 以及 RowHeadersBorderStyle 只有在被设定为 Single 时，才能改变颜色。

4） 单元格的上下左右的边框线式样的单独设定

CellBorderStyle只能设定单元格全部边框线的式样。要单独改变单元格某一边边框式样的话，需要用到 DataGridView.AdvancedCellBorderStyle属性。如示例：

```
[VB.NET]
’ 单元格的上边和左边线设为二重线
’ 单元格的下边和右边线设为单重线
DataGridView1.AdvancedCellBorderStyle.Top = _
        DataGridViewAdvancedCellBorderStyle.InsetDouble
DataGridView1.AdvancedCellBorderStyle.Right = _
        DataGridViewAdvancedCellBorderStyle.Inset
DataGridView1.AdvancedCellBorderStyle.Bottom = _
        DataGridViewAdvancedCellBorderStyle.Inset
DataGridView1.AdvancedCellBorderStyle.Left = _
        DataGridViewAdvancedCellBorderStyle.InsetDouble
```

同样，设定行头单元格的属性是： AdvancedRowHeadersBorderStyle， 设定列头单元格属性是： AdvancedColumnHeadersBorderStyle。

---

**⑯ DataGridView 单元格表示值的自定义**

通过CellFormatting事件，可以自定义单元格的表示值。（比如：值为Error的时候，单元格被设定为红色）
下面的示例：将“Colmn1”列的值改为大写。

```
[C#]
//CellFormatting 事件处理方法
private void DataGridView1_CellFormatting(object sender,
        DataGridViewCellFormattingEventArgs e)
{
        DataGridView dgv = (DataGridView)sender;
        // 如果单元格是“Column1”列的单元格
        if (dgv.Columns[e.ColumnIndex].Name == "Column1" && e.Value is string)
        {
                // 将单元格值改为大写
                string str = e.Value.ToString();
                e.Value = str.ToUpper();
                // 应用该Format，Format完毕。
                e.FormattingApplied = true;
        }
}
```

CellFormatting事件的DataGridViewCellFormattingEventArgs对象的Value属性一开始保存着未被格式化的值。当Value属性被设定表示用的文本之后，把FormattingApplied属性做为True，告知DataGridView文本已经格式化完毕。如果不这样做的话，DataGridView会根据已经设定的Format，NullValue，DataSourceNullValue，FormatProvider属性会将Value属性会被重新格式化一遍。

---

**⑰ DataGridView 用户输入时，单元格输入值的设定**

通过 DataGridView.CellParsing 事件可以设定用户输入的值。下面的示例：当输入英文文本内容的时候，立即被改变为大写。

```
[C#]
//CellParsing 事件处理方法
private void DataGridView1_CellParsing(object sender,
        DataGridViewCellParsingEventArgs e)
{
        DataGridView dgv = (DataGridView)sender;
        //单元格列为“Column1”时
        if (dgv.Columns[e.ColumnIndex].Name == "Column1" &&
                e.DesiredType == typeof(string))
        {
                //将单元格值设为大写
                e.Value = e.Value.ToString().ToUpper();
                //解析完毕
                e.ParsingApplied = true;
        }
}
```

---

**⑱ DataGridView 新加行的默认值的设定**

需要指定新加行的默认值的时候，可以在DataGridView.DefaultValuesNeeded事件里处理。在该事件中处理除了可以设定默认值以外，还可以指定某些特定的单元格的ReadOnly属性等。

```
[C#]
// DefaultValuesNeeded 事件处理方法
private void DataGridView1_DefaultValuesNeeded(object sender,
        DataGridViewRowEventArgs e)
{
        // 设定单元格的默认值
        e.Row.Cells["Column1"].Value = 0;
        e.Row.Cells["Column2"].Value = "-";
}
```

## DataGridView的中的查找,添加，删除行

```
public class DVOP
        {
                /// <summary>
                /// DataGridView中查找指定列中的指定值，并选中该单元格
                /// </summary>
                /// <param name="dv">DataGridView控件</param>
```

```
        /// <param name= str >查找的指定值</param>
        /// <returns>如果有查找的值返回true，没有则返回false</returns>
        public bool Select(DataGridView dv, string colname, string str)
        {
                bool selected = false;
                int row = dv.Rows.Count;//得到总行数
                for (int i = 0; i < row; i++)//得到总行数并在之内循环
                {
                        if (str == dv.Rows[i].Cells[colname].Value.ToString())
                        {     //对比TexBox中的值是否与dataGridView中的值相同（上面这句）
                                dv.CurrentCell = dv[colname, i];//定位到相同的单元格
                                selected = true;
                                return selected;//返回
                        }
                }
                return selected;
        }
        /// <summary>
        /// 向DataGirdView中添加行，显示在第一行，并选中该行
        /// </summary>
        /// <param name="dv">DataGridView对向</param>
        /// <param name="colcount">总列数</param>
        /// <param name="colvalues">要添加的每列的值</param>
        public void AddRow(DataGridView dv, int colcount,params string[] colvalues)
        {
                DataGridViewRow dr = new DataGridViewRow();
                dr.CreateCells(dv);
                for (int i = 0; i < colcount; i++)
                {
                        dr.Cells[i].Value = colvalues[i];
                }
                dv.Rows.Insert(0, dr);
                dv.CurrentCell = dv.Rows[0].Cells[0];
        }
        /// <summary>
        /// 删除DV中的指定行
        /// </summary>
        /// <param name="dv">DataGridView对向</param>
        /// <param name="colindex">要删除的指定行</param>
        public void ReMoveRow(DataGridView dv, int colindex)
        {
                dv.Rows.Remove(dv.Rows[colindex]);
                dv.ClearSelection();
        }
}
```

0(最基本的技巧)、获取某列中的某行(某单元格)中的内容

```
this.currentposition = this.dataGridView1.BindingContext [this.dataGridView1.DataSource,
this.dataGridView1.DataMember].Position;bookContent = this.database.dataSet.Tables[0].Rows
[this.currentposition][21].ToString().Trim();MessageBox.Show(bookContent);
```

1、自定义列

```
//定义列宽this.dataGridView1.Columns[0].Width = 80;this.dataGridView1.Columns[1].Width =
80;this.dataGridView1.Columns[2].Width = 180;this.dataGridView1.Columns[3].Width =
120;this.dataGridView1.Columns[4].Width = 120;Customize Cells and Columns in the Windows Forms
```

DataGridView Cells

继承 DataGridViewTextBoxCell 类生成新的Cell类，然后再继承 DataGridViewColumn 生成新的Column类，并指定

CellTemplate为新的Cell类。新生成的Column便可以增加到DataGridView中去。

2、自动适应列宽

Programmatically Resize Cells to Fit Content in the Windows Forms DataGridView ControlSamples：DataGridView.AutoSizeColumns(DataGridViewAutoSizeColumnCriteria.HeaderAndDisplayedRows);DataGridView.AutoSizeColumn(DataGridViewAutoSizeColumnCriteria.HeaderOnly,2, false);DataGridView.AutoSizeRow(DataGridViewAutoSizeRowCriteria.Columns,2, false);DataGridView.AutoSizeRows(DataGridViewAutoSizeRowCriteria.HeaderAndColumns,0, dataGridView1.Rows.Count, false);

3、可以绑定并显示对象

Bind Objects to Windows Forms DataGridView Controls

4、可以改变表格线条风格

Change the Border and Gridline Styles in the Windows Forms DataGridView ControlSamples:this.dataGridView1.GridColor = Color.BlueViolet;this.dataGridView1.BorderStyle = BorderStyle.Fixed3D;this.dataGridView1.CellBorderStyle = DataGridViewCellBorderStyle.None;this.dataGridView1.RowHeadersBorderStyle = DataGridViewHeaderBorderStyle.Single;this.dataGridView1.ColumnHeadersBorderStyle = DataGridViewHeaderBorderStyle.Single;

5、动态改变列是否显示，和动态改变列的显示顺序

Change the Order of the Columns in the Windows Forms DataGridView ControlSamples：customersDataGridView.Columns["CustomerID"].Visible = false;customersDataGridView.Columns["ContactName"].DisplayIndex = 0;customersDataGridView.Columns["ContactTitle"].DisplayIndex = 1;customersDataGridView.Columns["City"].DisplayIndex = 2;customersDataGridView.Columns["Country"].DisplayIndex = 3;customersDataGridView.Columns["CompanyName"].DisplayIndex = 4;

6、可以在列中显示图像

Display Images in Cells of the Windows Forms DataGridView ControlSamples：Icon treeIcon = new Icon(this.GetType(), "tree.ico");DataGridViewImageColumn iconColumn = new DataGridViewImageColumn();iconColumn.Image = treeIcon.ToBitmap();iconColumn.Name = "Tree";iconColumn.HeaderText = "Nice tree";dataGridView1.Columns.Insert(2, iconColumn);

7、格式化显示内容:

Format Data in the Windows Forms DataGridView ControlSamples:this.dataGridView1.Columns["UnitPrice"].DefaultCellStyle.Format = "c";this.dataGridView1.Columns["ShipDate"].DefaultCellStyle.Format =

entry ;this.dataGridView1.DefaultCellStyle.WrapMode = DataGridViewWrapMode.Wrap;this.dataGridView1.Columns["CustomerName"].DefaultCellStyle.Alignment =DataGridViewContentAlignment.MiddleRight;

8、在拖动列的滚动条时可以将指定的列冻结

Freeze Columns in the Windows Forms DataGridView ControlSamples：将指定列及以前的列固定不动
this.dataGridView1.Columns["AddToCartButton"].Frozen = true;

9、获取选择的单元格,行,列

Get the Selected Cells, Rows, and Columns in the Windows Forms DataGridView ControlSamples:

10、显示录入时出现的错误信息

Handle Errors that Occur During Data Entry in the Windows Forms DataGridView ControlSamples：
private void dataGridView1_DataError(object sender,DataGridViewDataErrorEventArgs e){// If the data source raises an exception when a cell value is// commited, display an error message.if (e.Exception != null &&e.Context == DataGridViewDataErrorContext.Commit) {MessageBox.Show("CustomerID value must be unique.");}}

11、大数据量显示采用Virtual Mode

Implement Virtual Mode in the Windows Forms DataGridView Control

12、设置指定的列只读

Make Columns in the Windows Forms DataGridView Control Read-OnlySamples：
dataGridView1.Columns["CompanyName"].ReadOnly = true;

13、移去自动生成的列

Remove Autogenerated Columns from a Windows Forms DataGridView ControlSample:dataGridView1.AutoGenerateColumns = true;dataGridView1.DataSource = customerDataSet;dataGridView1.Columns.Remove ("Fax");或：
dataGridView1.Columns["CustomerID"].Visible = false;

14、自定义选择模式

Set the Selection Mode of the Windows Forms DataGridView ControlSample：
this.dataGridView1.SelectionMode = DataGridViewSelectionMod e.FullRowSelect;this.dataGridView1.MultiSelect = false;

15、自定义设定光标进入单元格是否编辑模式（编辑模式）

Specify the Edit Mode for the Windows Forms DataGridView Controlthis.dataGridView1.EditMode = DataGridViewEditMode.EditOnEnter;

16、新行指定默认值

Specify Default Values for New Rows in the Windows Forms DataGridView ControlSample：private void dataGridView1_DefaultValuesNeeded(object sender, System.Windows.Forms.DataGridViewRowEventArgs e) {e.Row.Cells["Region"].Value = "WA";e.Row.Cells["City"].Value = "Redmond";e.Row.Cells["PostalCode"].Value = "98052-6399";e.Row.Cells["Region"].Value = "NA";e.Row.Cells["Country"].Value = "USA";e.Row.Cells["CustomerID"].Value = NewCustomerId();}

17、数据验证

Validate Data in the Windows Forms DataGridView ControlSamples：private void dataGridView1_CellValidating(object sender,DataGridViewCellValidatingEventArgs e){// Validate the CompanyName entry by disallowing empty strings.if (dataGridView1.Columns[e.ColumnIndex].Name == "CompanyName"){if (e.FormattedValue.ToString() == String.Empty) {dataGridView1.Rows[e.RowIndex].ErrorText ="Company Name must not be empty";e.Cancel = true;}}}

18、数据提交到dataset中

DataSet ds = new DataSet("MyDataSet");ds.Tables[biaom.Trim()].Rows.Clear();try{for (int i = 0; i < dataGridView1.Rows.Count - 1; i++){DataTable dt = ds.Tables[biaom.Trim()];DataRow myrow = ds.Tables[biaom.Trim()].NewRow();for (int j = 0; j < dataGridView1.Columns.Count; j++){myrow[j] = Convert.ToString(dataGridView1.Rows[i].Cells[j].Value);}ds.Tables[biaom.Trim()].Rows.Add(myrow);}}c (Exception){MessageBox.Show("输入类型错误！");return;}

from http://blog.csdn.net/ibmfahsion/article/details/7891047



前一篇：【转】C#中dataGridView用法实例分析

后一篇：Oracle ERP无法导出数据的问题

〈 前一篇

【转】C#中dataGridView用法实例分析

后一篇 〉

Oracle ERP无法导出数据的问题