# DataGridView 控件用法合集

## 目　录

# 正　文

## DataGridView 控件用法合集(一)

1. DataGridView 当前的单元格属性取得、变更
2. DataGridView 编辑属性
3. DataGridView 最下面一列新追加行非表示
4. DataGridView 判断当前选中行是否为新追加的行
5. DataGridView 删除行可否设定
6. DataGridView 行列不表示和删除

**1.当前的单元格属性取得、变更**

[VB.NET]

```
'当前选中单元的值
Console.WriteLine(DataGridView1.CurrentCell.Value)
'当前列的 Index 值
  Console.WriteLine(DataGridView1.CurrentCell.ColumnIndex)
'当前单元的行 Index 值
Console.WriteLine(DataGridView1.CurrentCell.RowIndex)
'将控件中(0, 0)处的值，赋给当前单元格.
DataGridView1.CurrentCell = DataGridView1(0, 0)
```

**2.DataGridView 编辑属性**

全部单元格编辑属性

[VB.NET]

```
'DataGridView1 只读属性
```

DataGridView1.ReadOnly = True
指定行列单元格编辑属性
[VB.NET]

```
DataGridView1.Columns(1).ReadOnly = True

DataGridView1.Rows(2).ReadOnly = True

DataGridView1(0, 0).ReadOnly = True
```

根据条件判断单元格的编辑属性
下例中 column2 的值是 True 的时候，Column1 设为可编辑
[VB.NET]

```
Private Sub DataGridView1_CellBeginEdit(ByVal sender As Object, _
        ByVal e As DataGridViewCellCancelEventArgs) _
        Handles DataGridView1.CellBeginEdit
    Dim dgv As DataGridView = CType(sender, DataGridView)

    If dgv.Columns(e.ColumnIndex).Name = "Column1" AndAlso _
        Not CBool(dgv("Column2", e.RowIndex).Value) Then

        e.Cancel = True
    End If
End Sub
```

**3.DataGridView 最下面一列新追加行非表示**
[VB.NET]

```
DataGridView1.AllowUserToAddRows = False
```

**4.判断当前选中行是否为新追加的行**
[VB.NET]

```
If DataGridView1.CurrentRow.IsNewRow Then
    Console.WriteLine("現在のセルがある行は、新しい行です。")
Else
    Console.WriteLine("現在のセルがある行は、新しい行ではありません。")
End If
```

**5. DataGridView 删除行可否设定**
[VB.NET]

```
DataGridView1.AllowUserToDeleteRows = False
```

根据条件判断当前行是否要删除
[VB.NET]

```
Private Sub DataGridView1_UserDeletingRow(ByVal sender As Object, _
        ByVal e As DataGridViewRowCancelEventArgs) _
        Handles DataGridView1.UserDeletingRow

    If MessageBox.Show("この列を削除しますか？", "削除の確認", _
        MessageBoxButtons.OKCancel, MessageBoxIcon.Question) <> _
            Windows.Forms.DialogResult.OK Then
        e.Cancel = True
```

```
    End If
End Sub
```

**6. DataGridView 行列不表示和删除**

行列不表示

[VB.NET]

```
'DataGridView1 のはじめの列を非表示にする
DataGridView1.Columns(0).Visible = False
'DataGridView1 のはじめの行を非表示にする
DataGridView1.Rows(0).Visible = False
```

行列表头部分不表示

[VB.NET]

```
DataGridView1.ColumnHeadersVisible = False

DataGridView1.RowHeadersVisible = False
```

指定行列删除

[VB.NET]

```
DataGridView1.Columns.Remove("Column1")

DataGridView1.Columns.RemoveAt(0)

DataGridView1.Rows.RemoveAt(0)
```

选择的行列删除（多行列）

[VB.NET]

```
'DataGridView1 で選択されているすべての行を削除する
Dim r As DataGridViewRow
For Each r In DataGridView1.SelectedRows
    If Not r.IsNewRow Then
        DataGridView1.Rows.Remove(r)
    End If
Next r
```

**DataGridView 控件用法合集(二)**

7. DataGridView 行列宽度高度设置为不能编辑
8. DataGridView 行高列幅自动调整
9. DataGridView 指定行列冻结
10. DataGridView 列顺序变更可否设定
11. DataGridView 行复数选择
12. DataGridView 选择的行、列、单元格取得

**7. DataGridView 行列宽度高度设置为不能编辑**

[VB.NET]

```
'DataGridView1 の列の幅をユーザーが変更できないようにする
DataGridView1.AllowUserToResizeColumns = False
'DataGridView1 の行の高さをユーザーが変更できないようにする
DataGridView1.AllowUserToResizeRows = False
```

指定行列宽度高度设置为不能编辑

[VB.NET]

```
'DataGridView1 のはじめの列の幅をユーザーが変更できないようにする
DataGridView1.Columns(0).Resizable = DataGridViewTriState.False
```

'DataGridView1 のはじめの行の高さをユーザーが変更できないようにする
DataGridView1.Rows(0).Resizable = DataGridViewTriState.False
列幅行高最小值设定
[VB.NET]
'一番はじめの列の幅の最小を 100 ピクセルとする
DataGridView1.Columns(0).MinimumWidth = 100
'一番はじめの行の高さの最小を 50 ピクセルとする
DataGridView1.Rows(0).MinimumHeight = 50
行列表头部分行高列幅设置为不能编辑
[VB.NET]
'列ヘッダーの高さを変更できないようにする
DataGridView1.ColumnHeadersHeightSizeMode = _
    DataGridViewColumnHeadersHeightSizeMode.DisableResizing
'行ヘッダーの幅を変更できるようにする
DataGridView1.RowHeadersWidthSizeMode = _
    DataGridViewRowHeadersWidthSizeMode.EnableResizing

**8. DataGridView 行高列幅自动调整**

[VB.NET]
'ヘッダーとすべてのセルの内容に合わせて、列の幅を自動調整する
DataGridView1.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.AllCells
'ヘッダーとすべてのセルの内容に合わせて、行の高さを自動調整する
DataGridView1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.AllCells
表头部分行高列幅自动调整
[VB.NET]
'列ヘッダーの高さが自動調整されるようにする
DataGridView1.ColumnHeadersHeightSizeMode = _
    DataGridViewColumnHeadersHeightSizeMode.AutoSize
'行ヘッダーの幅が自動調整されるようにする
DataGridView1.RowHeadersWidthSizeMode = _
    DataGridViewRowHeadersWidthSizeMode.AutoSizeToAllHeaders
指定列自动调整
[VB.NET]
'はじめの列の幅を自動調整する
DataGridView1.Columns(0).AutoSizeMode = _
    DataGridViewAutoSizeColumnMode.DisplayedCells

**9. DataGridView 指定行列冻结**

列冻结（当前列以及左侧做所有列）
[VB.NET]
'DataGridView1 の左側 2 列を固定する
DataGridView1.Columns(1).Frozen = True
行冻结（当前行以及上部所有行）
[VB.NET]
'DataGridView1 の上部 2 行を固定する
DataGridView1.Rows(2).Frozen = True
指定单元格冻结（单元格所在行上部分所有行，列左侧所有列）
[VB.NET]
DataGridView1(0, 0). Frozen = True

**10. DataGridView 列顺序变更可否设定**

[VB.NET]
'DataGridView1 の列の位置をユーザーが変更できるようにする

DataGridView1.AllowUserToOrderColumns = True
但是如果列冻结的情况下，冻结的部分不能变更到非冻结的部分。
变更后列位置取得
[VB.NET]
'列"Column1"の現在の位置を取得する
Console.WriteLine(DataGridView1.Columns("Column1").DisplayIndex)
'列"Column1"を先頭に移動する
DataGridView1.Columns("Column1").DisplayIndex = 0

**11. DataGridView 行复数选择**

复数行选择不可
[VB.NET]
'DataGridView1 でセル、行、列が複数選択されないようにする
DataGridView1.MultiSelect = False
单元格选择的时候默认为选择整行
[VB.NET]
'セルを選択すると行全体が選択されるようにする
DataGridView1.SelectionMode = DataGridViewSelectionMode.FullRowSelect

**12. DataGridView 选择的行、列、单元格取得**

[VB.NET]
'選択されているセルを表示
Console.WriteLine("選択されているセル")
For Each c As DataGridViewCell In DataGridView1.SelectedCells
    Console.WriteLine("{0}, {1}", c.ColumnIndex, c.RowIndex)
Next c
'選択されている行を表示
Console.WriteLine("選択されている行")
For Each r As DataGridViewRow In DataGridView1.SelectedRows
    Console.WriteLine(r.Index)
Next r
'選択されている列を表示
Console.WriteLine("選択されている列")
For Each c As DataGridViewColumn In DataGridView1.SelectedColumns
    Console.WriteLine(c.Index)
Next c
指定行、列、单元格取得
[VB.NET]
'(0, 0)のセルを選択する
DataGridView1(0, 0).Selected = True
'インデックス 1 の行を選択する
DataGridView1.Rows(1).Selected = True
'インデックス 2 の列を選択する
DataGridView1.Columns(2).Selected = True

**DataGridView 控件用法合集(三)**

13. DataGridView 指定单元格是否表示
14. DataGridView 表头部单元格取得
15. DataGridView 表头部单元格文字列设定
16. DataGridView 选择的部分拷贝至剪贴板
17.DataGridView 粘贴
18. DataGridView 单元格上 ToolTip 表示设定(鼠标移动到相应单元格上时，弹出说明信息)

**13. DataGridView 指定单元格是否表示**

[VB.NET]

```
If Not DataGridView1(0, 0).Displayed AndAlso _
        DataGridView1(0, 0).Visible Then
    DataGridView1.CurrentCell = DataGridView1(0, 0)
End If
```

**14. DataGridView 表头部单元格取得**

[VB.NET]

```
'DataGridView1 のはじめの列のテキストを変更する
DataGridView1.Columns(0).HeaderCell.Value = "はじめの列"
'DataGridView1 のはじめの行のテキストを変更する
DataGridView1.Rows(0).HeaderCell.Value = "はじめの行"
'DataGridView1 の左上隅のセルのテキストを変更する
DataGridView1.TopLeftHeaderCell.Value = "左上"
```

**15. DataGridView 表头部单元格文字列设定**

更改列 Header 表示文字列

[VB.NET]

```
'DataGridView1 のはじめの列のテキストを変更する
DataGridView1.Columns(0).HeaderText = "はじめの列"
```

更改行 Header 表示文字列

[VB.NET]

```
'DataGridView1 の行ヘッダーに行番号を表示する
Dim i As Integer
For i = 0 To DataGridView1.Rows.Count - 1
    DataGridView1.Rows(i).HeaderCell.Value = i.ToString()
Next i
'行ヘッダーの幅を自動調節する
DataGridView1.AutoResizeRowHeadersWidth( _
    DataGridViewRowHeadersWidthSizeMode.AutoSizeToAllHeaders)
```

最左上 Header 单元格文字列

[VB.NET]

```
'左上隅のヘッダーセルに"/"と表示する
DataGridView1.TopLeftHeaderCell.Value = "/"
```

**16. DataGridView 选择的部分拷贝至剪贴板**

拷贝模式设定

[VB.NET]

```
'ヘッダーをコピーしないようにする
DataGridView1.ClipboardCopyMode = _
    DataGridViewClipboardCopyMode.EnableWithoutHeaderText
```

选中部分拷贝

[VB.NET]

```
'選択されたセルをクリップボードにコピーする
Clipboard.SetDataObject(DataGridView1.GetClipboardContent())
```

**17.DataGridView 粘贴**

[VB.NET]

```
'現在のセルのある行から下にペーストする
If DataGridView1.CurrentCell Is Nothing Then
    Return
End If
Dim insertRowIndex As Integer = DataGridView1.CurrentCell.RowIndex
```

```
'クリップボードの内容を取得して、行で分ける
Dim pasteText As String = Clipboard.GetText()
If String.IsNullOrEmpty(pasteText) Then
    Return
End If
pasteText = pasteText.Replace(vbCrLf, vbLf)
pasteText = pasteText.Replace(vbCr, vbLf)
pasteText.TrimEnd(New Char() {vbLf})
Dim lines As String() = pasteText.Split(vbLf)
Dim isHeader As Boolean = True
For Each line As String In lines
    '列ヘッダーならば飛ばす
    If isHeader Then
        isHeader = False
    Else
        'タブで分割
        Dim vals As String() = line.Split(ControlChars.Tab)
        '列数が合っているか調べる
        If vals.Length - 1 <> DataGridView1.ColumnCount Then
            Throw New ApplicationException("列数が違います。")
        End If
        Dim row As DataGridViewRow = DataGridView1.Rows(insertRowIndex)
        'ヘッダーを設定
        row.HeaderCell.Value = vals(0)
        '各セルの値を設定
        Dim i As Integer
        For i = 0 To row.Cells.Count - 1
            row.Cells(i).Value = vals((i + 1))
        Next i
        '次の行へ
        insertRowIndex += 1
    End If
Next line
```

**18. DataGridView 单元格上 ToolTip 表示设定(鼠标移动到相应单元格上时，弹出说明信息)**

[VB.NET]

```
'セルに表示する ToolTip を設定する
DataGridView1(0, 0).ToolTipText = "このセルは変更できません"
'列ヘッダーに表示する ToolTip を設定する
DataGridView1.Columns(0).ToolTipText = "この列には数字を入力できます"
'行ヘッダーに表示する ToolTip を設定する
DataGridView1.Rows(0).HeaderCell.ToolTipText = "この行のセルは変更できません"
```

CellToolTipTextNeeded 事件，在多个单元格使用相同的 ToolTips 的时候，可以用该事件，下例为显示当前单元格的行号和列号

[VB.NET]

```
'CellToolTipTextNeeded イベントハンドラ
Private Sub DataGridView1_CellToolTipTextNeeded(ByVal sender As Object, _
        ByVal e As DataGridViewCellToolTipTextNeededEventArgs) _
        Handles DataGridView1.CellToolTipTextNeeded
    e.ToolTipText = e.ColumnIndex.ToString() + ", " + e.RowIndex.ToString()
End Sub
```

## DataGridView 控件用法合集(四)



### 19. DataGridView 中的 ContextMenuStrip 属性

[VB.NET]

```
'DataGridView の ContextMenuStrip を設定する
DataGridView1.ContextMenuStrip = Me.ContextMenuStrip1
'列の ContextMenuStrip を設定する
DataGridView1.Columns(0).ContextMenuStrip = Me.ContextMenuStrip2
'列ヘッダーの ContextMenuStrip を設定する
DataGridView1.Columns(0).HeaderCell.ContextMenuStrip = Me.ContextMenuStrip2
'行の ContextMenuStrip を設定する
DataGridView1.Rows(0).ContextMenuStrip = Me.ContextMenuStrip3
'セルの ContextMenuStrip を設定する
DataGridView1(1, 0).ContextMenuStrip = Me.ContextMenuStrip4
```

也可以用 CellContextMenuStripNeeded、RowContextMenuStripNeeded 属性进行定义

[VB.NET]

```
'CellContextMenuStripNeeded イベントハンドラ
Private Sub DataGridView1_CellContextMenuStripNeeded( _
        ByVal sender As Object, _
        ByVal e As DataGridViewCellContextMenuStripNeededEventArgs) _
        Handles DataGridView1.CellContextMenuStripNeeded
    Dim dgv As DataGridView = CType(sender, DataGridView)
    If e.RowIndex < 0 Then
        '列ヘッダーに表示する ContextMenuStrip を設定する
        e.ContextMenuStrip = Me.ContextMenuStrip1
    ElseIf e.ColumnIndex < 0 Then
        '行ヘッダーに表示する ContextMenuStrip を設定する
        e.ContextMenuStrip = Me.ContextMenuStrip2
    ElseIf TypeOf (dgv(e.ColumnIndex, e.RowIndex).Value) Is Integer Then
        'セルが整数型のときに表示する ContextMenuStrip を変更する
        e.ContextMenuStrip = Me.ContextMenuStrip3
    End If
End Sub
```

### 20. DataGridView 指定滚动框位置

[VB.NET]

```
'先頭の行までスクロールする
DataGridView1.FirstDisplayedScrollingRowIndex = 0
'先頭の列までスクロールする
DataGridView1.FirstDisplayedScrollingColumnIndex = 0
```

### 21. DataGridView 手动追加列

[VB.NET]

```
'列が自動的に作成されないようにする
DataGridView1.AutoGenerateColumns = False
```

```
'データソースを設定する
DataGridView1.DataSource = BindingSource1
'DataGridViewTextBoxColumn 列を作成する
Dim textColumn As New DataGridViewTextBoxColumn()
'データソースの"Column1"をバインドする
textColumn.DataPropertyName = "Column1"
'名前とヘッダーを設定する
textColumn.Name = "Column1"
textColumn.HeaderText = "Column1"
'列を追加する
DataGridView1.Columns.Add(textColumn)
```

**22. DataGridView 全体分界线样式设置**

[VB.NET]

```
'DataGridView の境界線を 3D にする
DataGridView1.BorderStyle = BorderStyle.Fixed3D
```

单元格上下左右分界线样式设置

[VB.NET]

```
'セルの上と左を二重線のくぼんだ境界線にし、
'下と右を一重線のくぼんだ境界線にする
DataGridView1.AdvancedCellBorderStyle.Top = DataGridViewAdvancedCellBorderStyle.InsetDouble
DataGridView1.AdvancedCellBorderStyle.Right = DataGridViewAdvancedCellBorderStyle.Inset
DataGridView1.AdvancedCellBorderStyle.Bottom = DataGridViewAdvancedCellBorderStyle.Inset
DataGridView1.AdvancedCellBorderStyle.Left = DataGridViewAdvancedCellBorderStyle.InsetDouble
```

**23. DataGridView 根据单元格属性更改显示内容**

如下例，当该列是字符串时，自动转换文字大小写

[VB.NET]

```
'CellFormatting イベントハンドラ
Private Sub DataGridView1_CellFormatting(ByVal sender As Object, _
        ByVal e As DataGridViewCellFormattingEventArgs) _
        Handles DataGridView1.CellFormatting
    Dim dgv As DataGridView = CType(sender, DataGridView)
    'セルの列を確認
    If dgv.Columns(e.ColumnIndex).Name = "Column1" AndAlso _
            TypeOf e.Value Is String Then
        '大文字にして表示する
        Dim str As String = e.Value.ToString()
        e.Value = str.ToUpper()
        'フォーマットの必要がないことを知らせる
        e.FormattingApplied = True
    End If
End Sub
```

**24. DataGridView 新追加行的行高样式设置**

行高设置

[VB.NET]

```
'行テンプレートの高さを設定する
DataGridView1.RowTemplate.Height = 50
'行の最低の高さを設定する
DataGridView1.RowTemplate.MinimumHeight = 50
```

样式设置

[VB.NET]

'行テンプレートのセルスタイルの背景色を黄色にする
DataGridView1.DefaultCellStyle.BackColor = Color.Yellow

**25. DataGridView 新追加行单元格默认值设置**

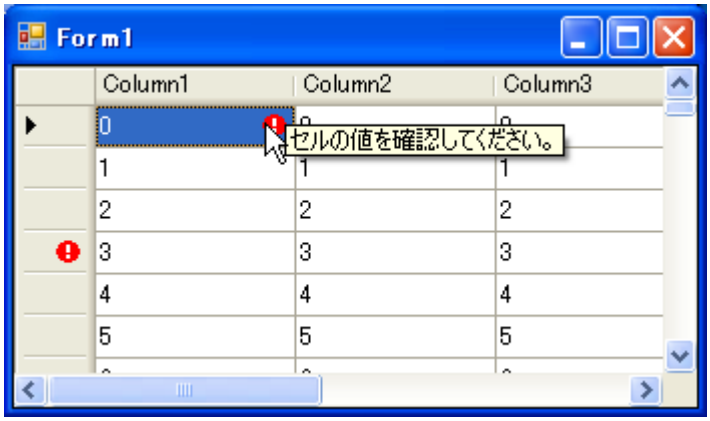
[VB.NET]

```
'DefaultValuesNeeded イベントハンドラ
Private Sub DataGridView1_DefaultValuesNeeded(ByVal sender As Object, _
        ByVal e As DataGridViewRowEventArgs) _
        Handles DataGridView1.DefaultValuesNeeded
    'セルの既定値を指定する
    e.Row.Cells("Column1").Value = 0
    e.Row.Cells("Column2").Value = "-"
End Sub
```

**DataGridView 中输入错误数据的处理（五）**

26. DataGridView 单元格数据错误标签表示
27. DataGridView 单元格内输入值正确性判断
28. DataGridView 单元格输入错误值事件的捕获

**26. DataGridView 单元格数据错误标签表示**




VB.NET]

```
'(0, 0)のセルにエラーアイコンを表示する
DataGridView1(0, 0).ErrorText = "セルの値を確認してください。"
'インデックスが 3 の行にエラーアイコンを表示する
DataGridView1.Rows(3).ErrorText = "負の値は入力できません。"
```

[C#]

```
//(0, 0)のセルにエラーアイコンを表示する
DataGridView1[0, 0].ErrorText = "セルの値を確認してください。";
//インデックスが 3 の行にエラーアイコンを表示する
DataGridView1.Rows[3].ErrorText = "負の値は入力できません。";
```

在大量单元格需要错误提示时，也可以用 CellErrorTextNeeded、RowErrorTextNeeded 事件

[VB.NET]

```
'CellErrorTextNeeded イベントハンドラ
Private Sub DataGridView1_CellErrorTextNeeded(ByVal sender As Object, _
        ByVal e As DataGridViewCellErrorTextNeededEventArgs) _
        Handles DataGridView1.CellErrorTextNeeded
    Dim dgv As DataGridView = CType(sender, DataGridView)
    'セルの値が負の整数であれば、エラーアイコンを表示する
    Dim cellVal As Object = dgv(e.ColumnIndex, e.RowIndex).Value
    If TypeOf cellVal Is Integer AndAlso CInt(cellVal) < 0 Then
        e.ErrorText = "負の整数は入力できません。"
    End If
```

```
End Sub
'RowErrorTextNeeded イベントハンドラ
Private Sub DataGridView1_RowErrorTextNeeded(ByVal sender As Object, _
        ByVal e As DataGridViewRowErrorTextNeededEventArgs) _
        Handles DataGridView1.RowErrorTextNeeded
    Dim dgv As DataGridView = CType(sender, DataGridView)
    If dgv("Column1", e.RowIndex).Value Is DBNull.Value AndAlso _
        dgv("Column2", e.RowIndex).Value Is DBNull.Value Then
        e.ErrorText = _
            "少なくとも Column1 と Column2 のどちらかには値を入力してください。"
    End If
End Sub
```

[C#]

```
//CellErrorTextNeeded イベントハンドラ
private void DataGridView1_CellErrorTextNeeded(object sender,
    DataGridViewCellErrorTextNeededEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    //セルの値が負の整数であれば、エラーアイコンを表示する
    object cellVal = dgv[e.ColumnIndex, e.RowIndex].Value;
    if (cellVal is int && ((int)cellVal) < 0)
    {
        e.ErrorText = "負の整数は入力できません。";
    }
}
//RowErrorTextNeeded イベントハンドラ
private void DataGridView1_RowErrorTextNeeded(object sender,
    DataGridViewRowErrorTextNeededEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    if (dgv["Column1", e.RowIndex].Value == DBNull.Value &&
        dgv["Column2", e.RowIndex].Value == DBNull.Value)
    {
        e.ErrorText =
            "少なくとも Column1 と Column2 のどちらかには値を入力してください。";
    }
}
```

**27. DataGridView 単元格内输入值正确性判断**

[VB.NET]

```
'CellValidating イベントハンドラ
Private Sub DataGridView1_CellValidating(ByVal sender As Object, _
        ByVal e As DataGridViewCellValidatingEventArgs) _
        Handles DataGridView1.CellValidating
    Dim dgv As DataGridView = CType(sender, DataGridView)
    If dgv.Columns(e.ColumnIndex).Name = "Column1" AndAlso _
            e.FormattedValue.ToString() = "" Then
        '行にエラーテキストを設定
        dgv.Rows(e.RowIndex).ErrorText = "値が入力されていません。"
        '入力した値をキャンセルして元に戻すには、次のようにする
        'dgv.CancelEdit()
```

```
        'キャンセルする
        e.Cancel = True
    End If
End Sub
'CellValidated イベントハンドラ
Private Sub DataGridView1_CellValidated(ByVal sender As Object, _
        ByVal e As DataGridViewCellEventArgs) _
        Handles DataGridView1.CellValidated
    Dim dgv As DataGridView = CType(sender, DataGridView)
    'エラーテキストを消す
    dgv.Rows(e.RowIndex).ErrorText = Nothing
End Sub
```

[C#]

```
//CellValidating イベントハンドラ
private void DataGridView1_CellValidating(object sender,
    DataGridViewCellValidatingEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    if (dgv.Columns[e.ColumnIndex].Name == "Column1" &&
        e.FormattedValue.ToString() == "")
    {
        //行にエラーテキストを設定
        dgv.Rows[e.RowIndex].ErrorText = "値が入力されていません。";
        //入力した値をキャンセルして元に戻すには、次のようにする
        //dgv.CancelEdit();
        //キャンセルする
        e.Cancel = true;
    }
}
//CellValidated イベントハンドラ
private void DataGridView1_CellValidated(object sender,
    DataGridViewCellEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    //エラーテキストを消す
    dgv.Rows[e.RowIndex].ErrorText = null;
}
```

**28. DataGridView 单元格输入错误值事件的捕获**

[VB.NET]

```
'DataError イベントハンドラ
Private Sub DataGridView1_DataError(ByVal sender As Object, _
        ByVal e As DataGridViewDataErrorEventArgs) _
        Handles DataGridView1.DataError
    If Not (e.Exception Is Nothing) Then
        MessageBox.Show(Me, _
            String.Format("({0}, {1}) のセルでエラーが発生しました。" + _
                vbCrLf + vbCrLf + "説明: {2}", _
                e.ColumnIndex, e.RowIndex, e.Exception.Message), _
            "エラーが発生しました", _
            MessageBoxButtons.OK, _
```

```
                MessageBoxIcon.Error)
    End If
End Sub
```

[C#]

```
//DataError イベントハンドラ
private void DataGridView1_DataError(object sender,
    DataGridViewDataErrorEventArgs e)
{
    if (e.Exception != null)
    {
        MessageBox.Show(this,
            string.Format("({0}, {1}) のセルでエラーが発生しました。\n\n 説明: {2}",
            e.ColumnIndex, e.RowIndex, e.Exception.Message),
            "エラーが発生しました",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

输入错误值时返回原先数据

[VB.NET]

```
'DataError イベントハンドラ
Private Sub DataGridView1_DataError(ByVal sender As Object, _
        ByVal e As DataGridViewDataErrorEventArgs) _
        Handles DataGridView1.DataError
    e.Cancel = False
End Sub
```

[C#]

```
//DataError イベントハンドラ
private void DataGridView1_DataError(object sender,
    DataGridViewDataErrorEventArgs e)
{
    e.Cancel = false;
}
```

**DataGridView 控件用法合集(六)**

29. DataGridView 行排序（点击列表头自动排序的设置）
30. DataGridView 自动行排序（新追加值也会自动排序）
31. DataGridView 自动行排序禁止情况下的排序
32. DataGridView 指定列指定排序

**29. DataGridView 行排序（点击列表头自动排序的设置）**

[VB.NET]

```
'並び替えができないようにする
For Each c As DataGridViewColumn In DataGridView1.Columns
    c.SortMode = DataGridViewColumnSortMode.NotSortable
Next c
```

**30. DataGridView 自动行排序（新追加值也会自动排序）**

[VB.NET]

```
'フォームの Load イベントハンドラ
Private Sub Form1_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
    '自動的に並び替えられるようにする
```

```
    Dim c As DataGridViewColumn
    For Each c In DataGridView1.Columns
        c.SortMode = DataGridViewColumnSortMode.Automatic
    Next c
End Sub
'Button1 の Click イベントハンドラ
Private Sub Button1_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles Button1.Click
    If DataGridView1.CurrentCell Is Nothing Then
        Return
    End If
    '並び替える列を決める
    Dim sortColumn As DataGridViewColumn = _
        DataGridView1.CurrentCell.OwningColumn
    '並び替えの方向（昇順か降順か）を決める
    Dim sortDirection As System.ComponentModel.ListSortDirection = _
        System.ComponentModel.ListSortDirection.Ascending
    If Not (DataGridView1.SortedColumn Is Nothing) AndAlso _
            DataGridView1.SortedColumn.Equals(sortColumn) Then
        sortDirection = IIf(DataGridView1.SortOrder = SortOrder.Ascending, _
            System.ComponentModel.ListSortDirection.Descending, _
            System.ComponentModel.ListSortDirection.Ascending)
    End If
    '並び替えを行う
    DataGridView1.Sort(sortColumn, sortDirection)
End Sub
```

**31. DataGridView 自动行排序禁止情况下的排序**

```
'ColumnHeaderMouseClick イベントハンドラ
Private Sub DataGridView1_ColumnHeaderMouseClick(ByVal sender As Object, _
        ByVal e As DataGridViewCellMouseEventArgs) _
        Handles DataGridView1.ColumnHeaderMouseClick
    Dim clickedColumn As DataGridViewColumn = _
        DataGridView1.Columns(e.ColumnIndex)
    If clickedColumn.SortMode <> DataGridViewColumnSortMode.Automatic Then
        Me.SortRows(clickedColumn, True)
    End If
End Sub
'RowsAdded イベントハンドラ
Private Sub DataGridView1_RowsAdded(ByVal sender As Object, _
        ByVal e As DataGridViewRowsAddedEventArgs) _
        Handles DataGridView1.RowsAdded
    Me.SortRows(DataGridView1.SortedColumn, False)
End Sub
'CellValueChanged イベントハンドラ
Private Sub DataGridView1_CellValueChanged(ByVal sender As Object, _
        ByVal e As DataGridViewCellEventArgs) _
        Handles DataGridView1.CellValueChanged
    If Not (DataGridView1.SortedColumn Is Nothing) AndAlso _
        e.ColumnIndex = DataGridView1.SortedColumn.Index Then
        Me.SortRows(DataGridView1.SortedColumn, False)
```

```
        End If
    End Sub
    ''' <summary>
    ''' 指定された列を基準にして並び替えを行う
    ''' </summary>
    ''' <param name="sortColumn">基準にする列</param>
    ''' <param name="orderToggle">並び替えの方向をトグルで変更する</param>
    Private Sub SortRows(ByVal sortColumn As DataGridViewColumn, _
            ByVal orderToggle As Boolean)
        If sortColumn Is Nothing Then
            Return
        End If
        '今までの並び替えグリフを消す
        If sortColumn.SortMode = DataGridViewColumnSortMode.Programmatic AndAlso _
            Not (DataGridView1.SortedColumn Is Nothing) AndAlso _
            Not DataGridView1.SortedColumn.Equals(sortColumn) Then
            DataGridView1.SortedColumn.HeaderCell.SortGlyphDirection = _
                SortOrder.None
        End If
        '並び替えの方向（昇順か降順か）を決める
        Dim sortDirection As System.ComponentModel.ListSortDirection
        If orderToggle Then
            sortDirection = IIf(DataGridView1.SortOrder = SortOrder.Descending, _
                System.ComponentModel.ListSortDirection.Ascending, _
                System.ComponentModel.ListSortDirection.Descending)
        Else
            sortDirection = IIf(DataGridView1.SortOrder = SortOrder.Descending, _
                System.ComponentModel.ListSortDirection.Descending, _
                System.ComponentModel.ListSortDirection.Ascending)
        End If
        Dim sOrder As SortOrder = _
            IIf(sortDirection = System.ComponentModel.ListSortDirection.Ascending, _
                SortOrder.Ascending, SortOrder.Descending)
        '並び替えを行う
        DataGridView1.Sort(sortColumn, sortDirection)
        If sortColumn.SortMode = DataGridViewColumnSortMode.Programmatic Then
            '並び替えグリフを変更
            sortColumn.HeaderCell.SortGlyphDirection = sOrder
        End If
    End Sub
```

[C#]

```
//フォームの Load イベントハンドラ
private void Form1_Load(object sender, EventArgs e)
{
    //イベントハンドラの追加
    DataGridView1.RowsAdded += new DataGridViewRowsAddedEventHandler(
        DataGridView1_RowsAdded);
    DataGridView1.CellValueChanged += new DataGridViewCellEventHandler(
        DataGridView1_CellValueChanged);
    DataGridView1.ColumnHeaderMouseClick += new DataGridViewCellMouseEventHandler(
```

```
            DataGridView1_ColumnHeaderMouseClick);
}
//ColumnHeaderMouseClick イベントハンドラ
private void DataGridView1_ColumnHeaderMouseClick(object sender,
    DataGridViewCellMouseEventArgs e)
{
    DataGridViewColumn clickedColumn = DataGridView1.Columns[e.ColumnIndex];
    if (clickedColumn.SortMode != DataGridViewColumnSortMode.Automatic)
        this.SortRows(clickedColumn, true);
}
//RowsAdded イベントハンドラ
private void DataGridView1_RowsAdded(object sender,
    DataGridViewRowsAddedEventArgs e)
{
    this.SortRows(DataGridView1.SortedColumn, false);
}
//CellValueChanged イベントハンドラ
private void DataGridView1_CellValueChanged(object sender,
    DataGridViewCellEventArgs e)
{
    if (DataGridView1.SortedColumn != null &&
        e.ColumnIndex == DataGridView1.SortedColumn.Index)
        this.SortRows(DataGridView1.SortedColumn, false);
}
/// <summary>
/// 指定された列を基準にして並び替えを行う
/// </summary>
/// <param name="sortColumn">基準にする列</param>
/// <param name="orderToggle">並び替えの方向をトグルで変更する</param>
private void SortRows(DataGridViewColumn sortColumn, bool orderToggle)
{
    if (sortColumn == null)
        return;
    //今までの並び替えグリフを消す
    if (sortColumn.SortMode == DataGridViewColumnSortMode.Programmatic &&
        DataGridView1.SortedColumn != null &&
        !DataGridView1.SortedColumn.Equals(sortColumn))
    {
        DataGridView1.SortedColumn.HeaderCell.SortGlyphDirection =
            SortOrder.None;
    }
    //並び替えの方向（昇順か降順か）を決める
    ListSortDirection sortDirection;
    if (orderToggle)
    {
        sortDirection =
            DataGridView1.SortOrder == SortOrder.Descending ?
            ListSortDirection.Ascending : ListSortDirection.Descending;
    }
    else
```

```
        {
            sortDirection =
                DataGridView1.SortOrder == SortOrder.Descending ?
                ListSortDirection.Descending : ListSortDirection.Ascending;
        }
        SortOrder sortOrder =
            sortDirection == ListSortDirection.Ascending ?
            SortOrder.Ascending : SortOrder.Descending;
        //並び替えを行う
        DataGridView1.Sort(sortColumn, sortDirection);
        if (sortColumn.SortMode == DataGridViewColumnSortMode.Programmatic)
        {
            //並び替えグリフを変更
            sortColumn.HeaderCell.SortGlyphDirection = sortOrder;
        }
}
```

**32. DataGridView 指定列指定排序**

[VB.NET]

```
'DataGridView1 にバインドされている DataTable を取得
Dim dt As DataTable = CType(DataGridView1.DataSource, DataTable)
'DataView を取得
Dim dv As DataView = dt.DefaultView
'Column1 と Column2 で昇順に並び替える
dv.Sort = "Column1, Column2 ASC"
'2 つの列のヘッダーに並び替えグリフを表示する
DataGridView1.Columns("Column1").HeaderCell.SortGlyphDirection = _
    SortOrder.Ascending
DataGridView1.Columns("Column2").HeaderCell.SortGlyphDirection = _
    SortOrder.Ascending
```

[C#]

```
//DataGridView1 にバインドされている DataTable を取得
DataTable dt = (DataTable)DataGridView1.DataSource;
//DataView を取得
DataView dv = dt.DefaultView;
//Column1 と Column2 で昇順に並び替える
dv.Sort = "Column1, Column2 ASC";
//2 つの列のヘッダーに並び替えグリフを表示する
DataGridView1.Columns["Column1"].HeaderCell.SortGlyphDirection =
    SortOrder.Ascending;
DataGridView1.Columns["Column2"].HeaderCell.SortGlyphDirection =
    SortOrder.Ascending;
```

**DataGridView 控件用法合集(七)**

33. DataGridView 单元格样式设置
34. DataGridView 文字表示位置的设定
35. DataGridView 单元格内文字列换行
36. DataGridView 单元格 DBNull 值表示的设定
37. DataGridView 单元格样式格式化
38. DataGridView 指定单元格颜色设定
39. DataGridView 单元格文字字体设置
40. DataGridView 根据单元格值设定单元格样式

**33. DataGridView 单元格样式设置**

指定行列的样式设定

[VB.NET]

```
'インデックス 0 の列のセルの背景色を水色にする
DataGridView1.Columns(0).DefaultCellStyle.BackColor = Color.Aqua
'インデックス 0 の行のセルの背景色を薄い灰色にする
DataGridView1.Rows(0).DefaultCellStyle.BackColor = Color.LightGray
```

[C#]

```
//インデックス 0 の列のセルの背景色を水色にする
DataGridView1.Columns[0].DefaultCellStyle.BackColor = Color.Aqua;
//インデックス 0 の行のセルの背景色を薄い灰色にする
DataGridView1.Rows[0].DefaultCellStyle.BackColor = Color.LightGray;
```

奇数行样式设定

[VB.NET]

```
'奇数行のセルの背景色を黄緑色にする
DataGridView1.AlternatingRowsDefaultCellStyle.BackColor = Color.GreenYellow
```

[C#]

```
//奇数行のセルの背景色を黄緑色にする
DataGridView1.AlternatingRowsDefaultCellStyle.BackColor = Color.GreenYellow;
```

行，列表头部的样式设定

[VB.NET]

```
'列ヘッダーの背景色をアイボリーにする
DataGridView1.ColumnHeadersDefaultCellStyle.BackColor = Color.Ivory
'行ヘッダーの背景色をライムにする
DataGridView1.RowHeadersDefaultCellStyle.BackColor = Color.Lime
```

[C#]

```
//列ヘッダーの背景色をアイボリーにする
DataGridView1.ColumnHeadersDefaultCellStyle.BackColor = Color.Ivory;
//行ヘッダーの背景色をライムにする
DataGridView1.RowHeadersDefaultCellStyle.BackColor = Color.Lime;
```

样式的优先顺序

一般单元格的样式优先顺位

DataGridViewCell.Style

DataGridViewRow.DefaultCellStyle

DataGridView.AlternatingRowsDefaultCellStyle

DataGridView.RowsDefaultCellStyle

DataGridViewColumn.DefaultCellStyle

DataGridView.DefaultCellStyle

表头部的样式优先顺位

DataGridViewCell.Style

DataGridView.RowHeadersDefaultCellStyle

DataGridView.ColumnHeadersDefaultCellStyle

DataGridView.DefaultCellStyle

下例说明

[VB.NET]

```
'1 列目を水色にする
DataGridView1.Columns(0).DefaultCellStyle.BackColor = Color.Aqua
'全ての列の背景色を黄色にする
DataGridView1.RowsDefaultCellStyle.BackColor = Color.Yellow
```

```
'奇数行を黄緑色にする
DataGridView1.AlternatingRowsDefaultCellStyle.BackColor = Color.GreenYellow
'3 行目をピンクにする
DataGridView1.Rows(2).DefaultCellStyle.BackColor = Color.Pink
'自身のセルスタイルと継承されたセルスタイルの背景色を取得する
'1 列目のセルスタイル
'"[Aqua]"と"[Aqua]"と表示される
Console.WriteLine(DataGridView1.Columns(0).DefaultCellStyle.BackColor)
Console.WriteLine(DataGridView1.Columns(0).InheritedStyle.BackColor)
'1 行目のセルスタイル
'"[Empty]"と"[Yellow]"と表示される
Console.WriteLine(DataGridView1.Rows(0).DefaultCellStyle.BackColor)
Console.WriteLine(DataGridView1.Rows(0).InheritedStyle.BackColor)
'2 行目のセルスタイル
'"[Empty]"と"[GreenYellow]"と表示される
Console.WriteLine(DataGridView1.Rows(1).DefaultCellStyle.BackColor)
Console.WriteLine(DataGridView1.Rows(1).InheritedStyle.BackColor)
'3 行目のセルスタイル

'"[Pink]"と"[Pink]"と表示される
Console.WriteLine(DataGridView1.Rows(2).DefaultCellStyle.BackColor)
Console.WriteLine(DataGridView1.Rows(2).InheritedStyle.BackColor)
'(0, 3)のセルスタイル
'"[Empty]"と"[Pink]"と表示される
Console.WriteLine(DataGridView1(0, 2).Style.BackColor)
Console.WriteLine(DataGridView1(0, 2).InheritedStyle.BackColor)
[C#]
//1 列目を水色にする
DataGridView1.Columns[0].DefaultCellStyle.BackColor = Color.Aqua;
//全ての列の背景色を黄色にする
DataGridView1.RowsDefaultCellStyle.BackColor = Color.Yellow;
//奇数行を黄緑色にする
DataGridView1.AlternatingRowsDefaultCellStyle.BackColor = Color.GreenYellow;
//3 行目をピンクにする
DataGridView1.Rows[2].DefaultCellStyle.BackColor = Color.Pink;
//自身のセルスタイルと継承されたセルスタイルの背景色を取得する
//1 列目のセルスタイル
//"[Aqua]"と"[Aqua]"と表示される
Console.WriteLine(DataGridView1.Columns[0].DefaultCellStyle.BackColor);
Console.WriteLine(DataGridView1.Columns[0].InheritedStyle.BackColor);
//1 行目のセルスタイル
//"[Empty]"と"[Yellow]"と表示される
Console.WriteLine(DataGridView1.Rows[0].DefaultCellStyle.BackColor);
Console.WriteLine(DataGridView1.Rows[0].InheritedStyle.BackColor);
//2 行目のセルスタイル
//"[Empty]"と"[GreenYellow]"と表示される
Console.WriteLine(DataGridView1.Rows[1].DefaultCellStyle.BackColor);
Console.WriteLine(DataGridView1.Rows[1].InheritedStyle.BackColor);
//3 行目のセルスタイル
//"[Pink]"と"[Pink]"と表示される
```

```
Console.WriteLine(DataGridView1.Rows[2].DefaultCellStyle.BackColor);
Console.WriteLine(DataGridView1.Rows[2].InheritedStyle.BackColor);
//(0, 3)のセルスタイル
//"[Empty]"と"[Pink]"と表示される
Console.WriteLine(DataGridView1[0, 2].Style.BackColor);
Console.WriteLine(DataGridView1[0, 2].InheritedStyle.BackColor);
```

复数行列的样式设定

```
[VB.NET]
'奇数列の背景色を変更する
'効率的な方法
Dim cellStyle As New DataGridViewCellStyle()
cellStyle.BackColor = Color.Yellow
For i As Integer = 0 To DataGridView1.Columns.Count - 1
    If i Mod 2 = 0 Then
        DataGridView1.Columns(i).DefaultCellStyle = cellStyle
    End If
Next i
'非効率的な方法
For i As Integer = 0 To DataGridView1.Columns.Count - 1
    If i Mod 2 = 0 Then
        DataGridView1.Columns(i).DefaultCellStyle.BackColor = Color.Yellow
    End If
Next i
[C#]
//奇数列の背景色を変更する
//効率的な方法
DataGridViewCellStyle cellStyle = new DataGridViewCellStyle();
cellStyle.BackColor = Color.Yellow;
for (int i = 0; i < DataGridView1.Columns.Count; i++)
{
    if (i % 2 == 0)
        DataGridView1.Columns[i].DefaultCellStyle = cellStyle;
}
//非効率的な方法
for (int i = 0; i < DataGridView1.Columns.Count; i++)
{
    if (i % 2 == 0)
        DataGridView1.Columns[i].DefaultCellStyle.BackColor = Color.Yellow;
}
```

**34. DataGridView 文字表示位置的设定**

单元格的设定

```
[VB.NET]
'"Column1"列のセルのテキストの配置を上下左右とも中央にする
DataGridView1.Columns("Column1").DefaultCellStyle.Alignment = _
    DataGridViewContentAlignment.MiddleCenter
[C#]
//"Column1"列のセルのテキストの配置を上下左右とも中央にする
DataGridView1.Columns["Column1"].DefaultCellStyle.Alignment =
    DataGridViewContentAlignment.MiddleCenter;
```

表头的设定

[VB.NET]
'"Column1"列のヘッダーのテキストの配置を上下左右とも中央にする
DataGridView1.Columns("Column1").HeaderCell.Style.Alignment = _
    DataGridViewContentAlignment.MiddleCenter
[C#]
//"Column1"列のヘッダーのテキストの配置を上下左右とも中央にする
DataGridView1.Columns["Column1"].HeaderCell.Style.Alignment =
    DataGridViewContentAlignment.MiddleCenter;

**35. DataGridView 单元格内文字列换行**

[VB.NET]
'"Column1"列のセルのテキストを折り返して表示する
DataGridView1.Columns("Column1").DefaultCellStyle.WrapMode = _
    DataGridViewTriState.True
'ヘッダーも折り返して表示するなら、次のようにする
DataGridView1.Columns("Column1").HeaderCell.Style.WrapMode = _
    DataGridViewTriState.True
[C#]
//"Column1"列のセルのテキストを折り返して表示する
DataGridView1.Columns["Column1"].DefaultCellStyle.WrapMode =
    DataGridViewTriState.True;
//ヘッダーも折り返して表示するなら、次のようにする
DataGridView1.Columns["Column1"].HeaderCell.Style.WrapMode =
    DataGridViewTriState.True;

**36. DataGridView 单元格 DBNull 值表示的设定**

[VB.NET]
DataGridView1.DefaultCellStyle.NullValue = "（指定されていません）"
[C#]
DataGridView1.DefaultCellStyle.NullValue = "（指定されていません）";
单元格内 NullValue 属性设定的值输入，表示单元格内为 Null 值
[VB.NET]
DataGridView1.DefaultCellStyle.NullValue = "-"
DataGridView1.DefaultCellStyle.DataSourceNullValue = "X"
[C#]
DataGridView1.DefaultCellStyle.NullValue = "-";
DataGridView1.DefaultCellStyle.DataSourceNullValue = "X";

**37. DataGridView 单元格样式格式化**

[VB.NET]
'列のセルのテキストの書式を地域通貨として指定する
DataGridView1.Columns(0).DefaultCellStyle.Format = "c"
DataGridView1.Columns(1).DefaultCellStyle.Format = "c"
'2 列目のカルチヤを変更する
DataGridView1.Columns(1).DefaultCellStyle.FormatProvider = _
    New System.Globalization.CultureInfo("en-US")
[C#]
//列のセルのテキストの書式を地域通貨として指定する
DataGridView1.Columns[0].DefaultCellStyle.Format = "c";
DataGridView1.Columns[1].DefaultCellStyle.Format = "c";
//2 列目のカルチヤを変更する
DataGridView1.Columns[1].DefaultCellStyle.FormatProvider =
    new System.Globalization.CultureInfo("en-US");

Format 的参数一览（整数）

書式

説明

値が"123456"の時

書式なし

123456

C

通貨

\123,456

D

10 進数

123456

E

指数

1.234560E+005

F

固定小数点

123456.00

G

一般

123456

N

数値

123,456.00

P

パーセント

12,345,600.00%

R

ラウンドトリップ

（エラーが出る）

X

16 進数

1E240

0

123456

00000000

00123456

########

123456

#,##0

123,456

%0

%12345600

00.000E0

12.346E4

プラス#;マイナス#;ゼロ

プラス 123456

i の値は「#」です。

i の値は「123456」です

Format 的参数一览（小数）

書式
 説明
 値が"1.23456789"の時
 書式なし
 1.23456789
 C
 通貨
 \1
D
 10 進数
 （エラーが出る）
E
 指数
 1.234568E+000
F
 固定小数点
 1.23
G
 一般
 1.23456789
N
 数値
 1.23
P
 パーセント
 123.46%
R
 ラウンドトリップ
 1.23456789
X
 16 進数
 （エラーが出る）
00.0000000000
01.2345678900
##.##########
1.23456789
#,##0.000
1.235
%0.##
%123.46
00.000E0
12.346E-1
プラス#;マイナス#;ゼロ
プラス 1.23
d の値は「#.##」です。
d の値は「1.23」です。

**38. DataGridView 指定单元格颜色设定**

光标下的单元格颜色自动变换
[VB.NET]

```
'DataGridView1 の CellMouseEnter イベントハンドラ
Private Sub DataGridView1_CellMouseEnter(ByVal sender As Object, _
        ByVal e As DataGridViewCellEventArgs) _
        Handles DataGridView1.CellMouseEnter
    'ヘッダー以外のセル
    If e.ColumnIndex >= 0 And e.RowIndex >= 0 Then
        Dim dgv As DataGridView = CType(sender, DataGridView)
        'セルスタイルを変更する
        dgv(e.ColumnIndex, e.RowIndex).Style.BackColor = Color.Red
        dgv(e.ColumnIndex, e.RowIndex).Style.SelectionBackColor = Color.Red
    End If
End Sub
'DataGridView1 の CellMouseLeave イベントハンドラ
Private Sub DataGridView1_CellMouseLeave(ByVal sender As Object, _
        ByVal e As DataGridViewCellEventArgs) _
        Handles DataGridView1.CellMouseLeave
    'ヘッダー以外のセル
    If e.ColumnIndex >= 0 And e.RowIndex >= 0 Then
        Dim dgv As DataGridView = CType(sender, DataGridView)
        'セルスタイルを元に戻す
        'セルスタイルを削除するなら、null を設定してもよい
        dgv(e.ColumnIndex, e.RowIndex).Style.BackColor = Color.Empty
        dgv(e.ColumnIndex, e.RowIndex).Style.SelectionBackColor = Color.Empty
    End If
End Sub
```

[C#]

```
//DataGridView1 の CellMouseEnter イベントハンドラ
private void DataGridView1_CellMouseEnter(object sender,
    DataGridViewCellEventArgs e)
{
    //ヘッダー以外のセル
    if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
    {
        DataGridView dgv = (DataGridView)sender;
        //セルスタイルを変更する
        dgv[e.ColumnIndex, e.RowIndex].Style.BackColor = Color.Red;
        dgv[e.ColumnIndex, e.RowIndex].Style.SelectionBackColor = Color.Red;
    }
}
//DataGridView1 の CellMouseLeave イベントハンドラ
private void DataGridView1_CellMouseLeave(object sender,
    DataGridViewCellEventArgs e)
{
    //ヘッダー以外のセル
    if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
    {
        DataGridView dgv = (DataGridView)sender;
        //セルスタイルを元に戻す
        //セルスタイルを削除するなら、null を設定してもよい
        dgv[e.ColumnIndex, e.RowIndex].Style.BackColor = Color.Empty;
```

```
            dgv[e.ColumnIndex, e.RowIndex].Style.SelectionBackColor = Color.Empty;
        }
}
```

表头部单元格颜色设定

[VB.NET]

```
'列ヘッダーの背景色を黄色にする
DataGridView1.ColumnHeadersDefaultCellStyle.BackColor = Color.Yellow
'行ヘッダーの背景色を黄緑色にする
DataGridView1.RowHeadersDefaultCellStyle.BackColor = Color.YellowGreen
'左上隅のヘッダーセルの背景色を青にする
DataGridView1.TopLeftHeaderCell.Style.BackColor = Color.Blue
```

[C#]

```
//列ヘッダーの背景色を黄色にする
DataGridView1.ColumnHeadersDefaultCellStyle.BackColor = Color.Yellow;
//行ヘッダーの背景色を黄緑色にする
DataGridView1.RowHeadersDefaultCellStyle.BackColor = Color.YellowGreen;
//左上隅のヘッダーセルの背景色を青にする
DataGridView1.TopLeftHeaderCell.Style.BackColor = Color.Blue;
```

**39. DataGridView 单元格文字字体设置**

光标下单元格字体设置为粗体

[VB.NET]

```
'デフォルトのセルスタイル
Private defaultCellStyle As DataGridViewCellStyle
'マウスポインタの下にあるセルのセルスタイル
Private mouseCellStyle As DataGridViewCellStyle
'フォームの Load イベントハンドラ
Private Sub Form1_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
    'デフォルトのセルスタイルの設定
    Me.defaultCellStyle = New DataGridViewCellStyle()
    '現在のセルのセルスタイルの設定
    Me.mouseCellStyle = New DataGridViewCellStyle()
    Me.mouseCellStyle.Font = New Font(DataGridView1.Font, _
        DataGridView1.Font.Style Or FontStyle.Bold)
End Sub
'DataGridView1 の CellMouseEnter イベントハンドラ
Private Sub DataGridView1_CellMouseEnter(ByVal sender As Object, _
        ByVal e As DataGridViewCellEventArgs) _
        Handles DataGridView1.CellMouseEnter
    'ヘッダー以外のセル
    If e.ColumnIndex >= 0 And e.RowIndex >= 0 Then
        Dim dgv As DataGridView = CType(sender, DataGridView)
        'セルスタイルを変更する
        dgv(e.ColumnIndex, e.RowIndex).Style = Me.mouseCellStyle
    End If
End Sub
'DataGridView1 の CellMouseLeave イベントハンドラ
Private Sub DataGridView1_CellMouseLeave(ByVal sender As Object, _
        ByVal e As DataGridViewCellEventArgs) _
        Handles DataGridView1.CellMouseLeave
```

```
    'ヘッダー以外のセル
    If e.ColumnIndex >= 0 And e.RowIndex >= 0 Then
        Dim dgv As DataGridView = CType(sender, DataGridView)
        'セルスタイルを元に戻す
        'セルスタイルを削除するなら、null を設定してもよい
        dgv(e.ColumnIndex, e.RowIndex).Style = Me.defaultCellStyle
    End If
End Sub
```

[C#]

```
//デフォルトのセルスタイル
private DataGridViewCellStyle defaultCellStyle;
//マウスポインタの下にあるセルのセルスタイル
private DataGridViewCellStyle mouseCellStyle;
//フォームの Load イベントハンドラ
private void Form1_Load(object sender, EventArgs e)
{
    //デフォルトのセルスタイルの設定
    this.defaultCellStyle = new DataGridViewCellStyle();
    //現在のセルのセルスタイルの設定
    this.mouseCellStyle = new DataGridViewCellStyle();
    this.mouseCellStyle.Font = new Font(DataGridView1.Font,
        DataGridView1.Font.Style | FontStyle.Bold);
}
//DataGridView1 の CellEnter イベントハンドラ
private void DataGridView1_CellEnter(object sender,
    DataGridViewCellEventArgs e)
{
    //ヘッダー以外のセル
    if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
    {
        DataGridView dgv = (DataGridView)sender;
        //セルスタイルを変更する
        dgv[e.ColumnIndex, e.RowIndex].Style = this.mouseCellStyle;
    }
}
//DataGridView1 の CellLeave イベントハンドラ
private void DataGridView1_CellLeave(object sender,
    DataGridViewCellEventArgs e)
{
    //ヘッダー以外のセル
    if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
    {
        DataGridView dgv = (DataGridView)sender;
        //セルスタイルを元に戻す
        //セルスタイルを削除するなら、null を設定してもよい
        dgv[e.ColumnIndex, e.RowIndex].Style = this.defaultCellStyle;
    }
}
```

**40. DataGridView 根据单元格值设定单元格样式**

单元格负数情况下显示黄色，0 的情况下显示红色

[VB.NET]

```
'CellFormatting イベントハンドラ
Private Sub DataGridView1_CellFormatting(ByVal sender As Object, _
        ByVal e As DataGridViewCellFormattingEventArgs) _
        Handles DataGridView1.CellFormatting
    Dim dgv As DataGridView = CType(sender, DataGridView)
    'セルの列を確認
    If dgv.Columns(e.ColumnIndex).Name = "Column1" AndAlso _
            TypeOf e.Value Is Integer Then
        Dim val As Integer = CInt(e.Value)
        'セルの値により、背景色を変更する
        If val < 0 Then
            e.CellStyle.BackColor = Color.Yellow
        Else If val = 0 Then
            e.CellStyle.BackColor = Color.Red
        End If
    End If
End Sub
```

[C#]

```
//CellFormatting イベントハンドラ
private void DataGridView1_CellFormatting(object sender,
    DataGridViewCellFormattingEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    //セルの列を確認
    if (dgv.Columns[e.ColumnIndex].Name == "Column1" && e.Value is int)
    {
        int val = (int)e.Value;
        //セルの値により、背景色を変更する
        if (val < 0)
        {
            e.CellStyle.BackColor = Color.Yellow;
        }
        else if (val == 0)
        {
            e.CellStyle.BackColor = Color.Red;
        }
    }
}
```

**DataGridView 控件用法合集(八)**

DataGridView Owner 描画

41. DataGridView 设置单元格背景颜色

42. DataGridView 行样式描画

43. DataGridView 显示行号

44. DataGridView 焦点所在单元格焦点框不显示的设定

**41. DataGridView 设置单元格背景颜色**

[VB.NET]

```
'CellPainting イベントハンドラ
Private Sub DataGridView1_CellPainting(ByVal sender As Object, _
```

```
        ByVal e As DataGridViewCellPaintingEventArgs) _
        Handles DataGridView1.CellPainting
    'ヘッダー以外のセルで、背景を描画する時
    If e.ColumnIndex >= 0 AndAlso e.RowIndex >= 0 AndAlso _
        (e.PaintParts And DataGridViewPaintParts.Background) = _
            DataGridViewPaintParts.Background Then
        '選択されているか調べ、色を決定する
        'bColor1 が開始色、bColor2 が終了色
        Dim bColor1, bColor2 As Color
        If (e.PaintParts And DataGridViewPaintParts.SelectionBackground) = _
                DataGridViewPaintParts.SelectionBackground AndAlso _
            (e.State And DataGridViewElementStates.Selected) = _
                DataGridViewElementStates.Selected Then
            bColor1 = e.CellStyle.SelectionBackColor
            bColor2 = Color.Black
        Else
            bColor1 = e.CellStyle.BackColor
            bColor2 = Color.LemonChiffon
        End If
        'グラデーションブラシを作成
        Dim b As New System.Drawing.Drawing2D.LinearGradientBrush( _
            e.CellBounds, bColor1, bColor2, _
            System.Drawing.Drawing2D.LinearGradientMode.Horizontal)
        Try
            'セルを塗りつぶす
            e.Graphics.FillRectangle(b, e.CellBounds)
        Finally
            b.Dispose()
        End Try
        '背景以外が描画されるようにする
        Dim paintParts As DataGridViewPaintParts = _
            e.PaintParts And Not DataGridViewPaintParts.Background
        'セルを描画する
        e.Paint(e.ClipBounds, paintParts)
       '描画が完了したことを知らせる
        e.Handled = True
    End If
End Sub
```

[C#]

```
//CellPainting イベントハンドラ
private void DataGridView1_CellPainting(object sender,
    DataGridViewCellPaintingEventArgs e)
{
    //ヘッダー以外のセルで、背景を描画する時
    if (e.ColumnIndex >= 0 && e.RowIndex >= 0 &&
        (e.PaintParts & DataGridViewPaintParts.Background) ==
            DataGridViewPaintParts.Background)
    {
        //選択されているか調べ、色を決定する
        //bColor1 が開始色、bColor2 が終了色
```

```
        Color bColor1, bColor2;
        if ((e.PaintParts & DataGridViewPaintParts.SelectionBackground) ==
                DataGridViewPaintParts.SelectionBackground &&
            (e.State & DataGridViewElementStates.Selected) ==
                DataGridViewElementStates.Selected)
        {
            bColor1 = e.CellStyle.SelectionBackColor;
            bColor2 = Color.Black;
        }
        else
        {
            bColor1 = e.CellStyle.BackColor;
            bColor2 = Color.LemonChiffon;
        }
        //グラデーションブラシを作成
        using (System.Drawing.Drawing2D.LinearGradientBrush b =
            new System.Drawing.Drawing2D.LinearGradientBrush(
            e.CellBounds, bColor1, bColor2,
            System.Drawing.Drawing2D.LinearGradientMode.Horizontal))
        {
            //セルを塗りつぶす
            e.Graphics.FillRectangle(b, e.CellBounds);
        }
        //背景以外が描画されるようにする
        DataGridViewPaintParts paintParts =
            e.PaintParts & ~DataGridViewPaintParts.Background;
        //セルを描画する
        e.Paint(e.ClipBounds, paintParts);
        //描画が完了したことを知らせる
        e.Handled = true;
    }
}
```

单元格背景显示图像

[VB.NET]

```
'セルの背景に表示する画像
Private cellBackImage As New Bitmap("C:\back.gif")
'CellPainting イベントハンドラ
Private Sub DataGridView1_CellPainting(ByVal sender As Object, _
        ByVal e As DataGridViewCellPaintingEventArgs) _
        Handles DataGridView1.CellPainting
    'ヘッダー以外のセルで、背景を描画する時
    If e.ColumnIndex >= 0 AndAlso e.RowIndex >= 0 AndAlso _
        (e.PaintParts And DataGridViewPaintParts.Background) = _
            DataGridViewPaintParts.Background Then
        '背景だけを描画する
        Dim backParts As DataGridViewPaintParts = _
            e.PaintParts And (DataGridViewPaintParts.Background Or _
                DataGridViewPaintParts.SelectionBackground)
        e.Paint(e.ClipBounds, backParts)
        '画像をセルの真ん中に描画する
```

```
            Dim x As Integer = e.CellBounds.X + _
                (e.CellBounds.Width - cellBackImage.Width) / 2
            Dim y As Integer = e.CellBounds.Y + _
                (e.CellBounds.Height - cellBackImage.Height) / 2
            e.Graphics.DrawImage(cellBackImage, x, y)
            '背景以外が描画されるようにする
            Dim paintParts As DataGridViewPaintParts = _
                e.PaintParts And Not backParts
            'セルを描画する
            e.Paint(e.ClipBounds, paintParts)
            '描画が完了したことを知らせる
           e.Handled = True
        End If
End Sub
```

[C#]

```
//セルの背景に表示する画像
private Bitmap cellBackImage = new Bitmap("C:\\back.gif");
//CellPainting イベントハンドラ
private void DataGridView1_CellPainting(object sender,
    DataGridViewCellPaintingEventArgs e)
{
    //ヘッダー以外のセルで、背景を描画する時
    if (e.ColumnIndex >= 0 && e.RowIndex >= 0 &&
        (e.PaintParts & DataGridViewPaintParts.Background) ==
            DataGridViewPaintParts.Background)
    {
        //背景だけを描画する
        DataGridViewPaintParts backParts = e.PaintParts &
            (DataGridViewPaintParts.Background |
            DataGridViewPaintParts.SelectionBackground);
        e.Paint(e.ClipBounds, backParts);
        //画像をセルの真ん中に描画する
        int x = e.CellBounds.X +
            (e.CellBounds.Width - cellBackImage.Width) / 2;
        int y = e.CellBounds.Y +
            (e.CellBounds.Height - cellBackImage.Height) / 2;
        e.Graphics.DrawImage(cellBackImage, x, y);
        //背景以外が描画されるようにする
        DataGridViewPaintParts paintParts =
            e.PaintParts & ~backParts;
        //セルを描画する
        e.Paint(e.ClipBounds, paintParts);
        //描画が完了したことを知らせる
        e.Handled = true;
    }
}
```

**42. DataGridView 行样式描画**

利用 RowPostPaint 事件描画

[VB.NET]

```
'RowPostPaint イベントハンドラ
```

```
Private Sub DataGridView1_RowPostPaint(ByVal sender As Object, _
        ByVal e As DataGridViewRowPostPaintEventArgs) _
        Handles DataGridView1.RowPostPaint
    Dim dgv As DataGridView = CType(sender, DataGridView)
    '線の色を決定する
    Dim linePen As Pen
    Select Case e.RowIndex Mod 3
        Case 0
            linePen = Pens.Blue
        Case 1
            linePen = Pens.Green
        Case Else
            linePen = Pens.Red
    End Select
    '線を引く位置を計算する
    Dim startX As Integer = IIf(dgv.RowHeadersVisible, dgv.RowHeadersWidth, 0)
    Dim startY As Integer = e.RowBounds.Top + e.RowBounds.Height - 1
    Dim endX As Integer = startX + _
        dgv.Columns.GetColumnsWidth(DataGridViewElementStates.Visible) - _
        dgv.HorizontalScrollingOffset
    '線を引く
    e.Graphics.DrawLine(linePen, startX, startY, endX, startY)
End Sub
```

[C#]

```
//RowPostPaint イベントハンドラ
private void DataGridView1_RowPostPaint(object sender,
    DataGridViewRowPostPaintEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    //線の色を決定する
    Pen linePen;
    switch (e.RowIndex % 3)
    {
        case 0:
            linePen = Pens.Blue;
            break;
        case 1:
            linePen = Pens.Green;
            break;
        default:
            linePen = Pens.Red;
            break;
    }
    //線を引く位置を計算する
    int startX = dgv.RowHeadersVisible ? dgv.RowHeadersWidth : 0;
    int startY = e.RowBounds.Top + e.RowBounds.Height - 1;
    int endX = startX + dgv.Columns.GetColumnsWidth(
        DataGridViewElementStates.Visible) -
        dgv.HorizontalScrollingOffset;
    //線を引く
```

```
        e.Graphics.DrawLine(linePen,
            startX, startY, endX, startY);
}
```

利用 RowPrePaint 事件描画

[VB.NET]

```
'RowPrePaint イベントハンドラ
Private Sub DataGridView1_RowPrePaint(ByVal sender As Object, _
        ByVal e As DataGridViewRowPrePaintEventArgs) _
        Handles DataGridView1.RowPrePaint
    '背景を描画するか
    If (e.PaintParts And DataGridViewPaintParts.Background) = _
            DataGridViewPaintParts.Background Then
        '選択されているか調べ、色を決定する
        'bColor1 が開始色、bColor2 が終了色
        Dim bColor1, bColor2 As Color
        If (e.PaintParts And DataGridViewPaintParts.SelectionBackground) = _
                DataGridViewPaintParts.SelectionBackground AndAlso _
            (e.State And DataGridViewElementStates.Selected) = _
                DataGridViewElementStates.Selected Then
            bColor1 = e.InheritedRowStyle.SelectionBackColor
            bColor2 = Color.Black
        Else
            bColor1 = e.InheritedRowStyle.BackColor
            bColor2 = Color.YellowGreen
        End If
        'グラデーションの範囲を計算する
        'ヘッダーを除くセルの部分だけ描画する
        Dim dgv As DataGridView = CType(sender, DataGridView)
        Dim rectLeft2 As Integer = _
            IIf(dgv.RowHeadersVisible, dgv.RowHeadersWidth, 0)
        Dim rectLeft As Integer = _
            rectLeft2 - dgv.HorizontalScrollingOffset
        Dim rectWidth As Integer = _
            dgv.Columns.GetColumnsWidth(DataGridViewElementStates.Visible)
        Dim rect As New Rectangle(rectLeft, e.RowBounds.Top, rectWidth, e.RowBounds.Height - 1)
        'グラデーションブラシを作成
        Using b As New System.Drawing.Drawing2D.LinearGradientBrush( _
            rect, bColor1, bColor2, System.Drawing.Drawing2D.LinearGradientMode.Horizontal)
            '描画する範囲を計算する
            rect.X = rectLeft2
            rect.Width -= dgv.HorizontalScrollingOffset
            'セルを塗りつぶす
            e.Graphics.FillRectangle(b, rect)
        End Using
        'ヘッダーを描画する
        e.PaintHeader(True)
        '背景を描画しないようにする
        e.PaintParts = _
            e.PaintParts And Not DataGridViewPaintParts.Background
    End If
```

```
End Sub
'ColumnWidthChanged イベントハンドラ
Private Sub DataGridView1_ColumnWidthChanged(ByVal sender As Object, _
        ByVal e As DataGridViewColumnEventArgs) _
        Handles DataGridView1.ColumnWidthChanged
    Dim dgv As DataGridView = CType(sender, DataGridView)
    dgv.Invalidate()
End Sub
[C#]
//RowPrePaint イベントハンドラ
private void DataGridView1_RowPrePaint(object sender,
    DataGridViewRowPrePaintEventArgs e)
{
    //背景を描画するか
    if ((e.PaintParts & DataGridViewPaintParts.Background) ==
        DataGridViewPaintParts.Background)
    {
        //選択されているか調べ、色を決定する
        //bColor1 が開始色、bColor2 が終了色
        Color bColor1, bColor2;
        if ((e.PaintParts & DataGridViewPaintParts.SelectionBackground) ==
                DataGridViewPaintParts.SelectionBackground &&
            (e.State & DataGridViewElementStates.Selected) ==
                DataGridViewElementStates.Selected)
        {
            bColor1 = e.InheritedRowStyle.SelectionBackColor;
            bColor2 = Color.Black;
        }
        else
        {
            bColor1 = e.InheritedRowStyle.BackColor;
            bColor2 = Color.YellowGreen;
        }
       //グラデーションの範囲を計算する
        //ヘッダーを除くセルの部分だけ描画する
        DataGridView dgv = (DataGridView)sender;
        int rectLeft2 = dgv.RowHeadersVisible ? dgv.RowHeadersWidth : 0;
        int rectLeft = rectLeft2 - dgv.HorizontalScrollingOffset;
        int rectWidth = dgv.Columns.GetColumnsWidth(
            DataGridViewElementStates.Visible);
        Rectangle rect = new Rectangle(rectLeft, e.RowBounds.Top,
            rectWidth, e.RowBounds.Height - 1);
        //グラデーションブラシを作成
        using (System.Drawing.Drawing2D.LinearGradientBrush b =
            new System.Drawing.Drawing2D.LinearGradientBrush(
            rect, bColor1, bColor2,
            System.Drawing.Drawing2D.LinearGradientMode.Horizontal))
        {
            //描画する範囲を計算する
            rect.X = rectLeft2;
```

```
                rect.Width -= dgv.HorizontalScrollingOffset;
                //セルを塗りつぶす
                e.Graphics.FillRectangle(b, rect);
            }
            //ヘッダーを描画する
            e.PaintHeader(true);
            //背景を描画しないようにする
            e.PaintParts &= ~DataGridViewPaintParts.Background;
        }
}
//ColumnWidthChanged イベントハンドラ
private void DataGridView1_ColumnWidthChanged(object sender,
    DataGridViewColumnEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    dgv.Invalidate();
}
```

**43. DataGridView 显示行号**

[VB.NET]

```
'CellPainting イベントハンドラ
Private Sub DataGridView1_CellPainting(ByVal sender As Object, _
        ByVal e As DataGridViewCellPaintingEventArgs) _
        Handles DataGridView1.CellPainting
    '列ヘッダーかどうか調べる
    If e.ColumnIndex < 0 And e.RowIndex >= 0 Then
        'セルを描画する
        e.Paint(e.ClipBounds, DataGridViewPaintParts.All)
        '行番号を描画する範囲を決定する
        'e.AdvancedBorderStyle や e.CellStyle.Padding は無視しています
        Dim indexRect As Rectangle = e.CellBounds
        indexRect.Inflate(-2, -2)
        '行番号を描画する
        TextRenderer.DrawText(e.Graphics, _
            (e.RowIndex + 1).ToString(), _
            e.CellStyle.Font, _
            indexRect, _
            e.CellStyle.ForeColor, _
            TextFormatFlags.Right Or TextFormatFlags.VerticalCenter)
        '描画が完了したことを知らせる
        e.Handled = True
    End If
End Sub
```

[C#]

```
//CellPainting イベントハンドラ
private void DataGridView1_CellPainting(object sender,
    DataGridViewCellPaintingEventArgs e)
{
    //列ヘッダーかどうか調べる
    if (e.ColumnIndex < 0 && e.RowIndex >= 0)
    {
```

```
            //セルを描画する
            e.Paint(e.ClipBounds, DataGridViewPaintParts.All);
            //行番号を描画する範囲を決定する
            //e.AdvancedBorderStyle や e.CellStyle.Padding は無視しています
            Rectangle indexRect = e.CellBounds;
            indexRect.Inflate(-2, -2);
            //行番号を描画する
            TextRenderer.DrawText(e.Graphics,
                (e.RowIndex + 1).ToString(),
                e.CellStyle.Font,
                indexRect,
               e.CellStyle.ForeColor,
                TextFormatFlags.Right | TextFormatFlags.VerticalCenter);
            //描画が完了したことを知らせる
            e.Handled = true;
        }
    }
```

利用 RowPostPaint 事件描画

[VB.NET]

```
'RowPostPaint イベントハンドラ
Private Sub DataGridView1_RowPostPaint(ByVal sender As Object, _
        ByVal e As DataGridViewRowPostPaintEventArgs) _
        Handles DataGridView1.RowPostPaint
    Dim dgv As DataGridView = CType(sender, DataGridView)
    If dgv.RowHeadersVisible Then
        '行番号を描画する範囲を決定する
        Dim rect As New Rectangle(e.RowBounds.Left, e.RowBounds.Top, _
            dgv.RowHeadersWidth, e.RowBounds.Height)
       rect.Inflate(-2, -2)
        '行番号を描画する
        TextRenderer.DrawText(e.Graphics, _
            (e.RowIndex + 1).ToString(), _
            e.InheritedRowStyle.Font, _
            rect, _
            e.InheritedRowStyle.ForeColor, _
            TextFormatFlags.Right Or TextFormatFlags.VerticalCenter)
    End If
End Sub
```

[C#]

```
//RowPostPaint イベントハンドラ
private void DataGridView1_RowPostPaint(object sender,
    DataGridViewRowPostPaintEventArgs e)
    DataGridView dgv = (DataGridView)sender;
    if (dgv.RowHeadersVisible)
    {
        //行番号を描画する範囲を決定する
        Rectangle rect = new Rectangle(
            e.RowBounds.Left, e.RowBounds.Top,
            dgv.RowHeadersWidth, e.RowBounds.Height);
        rect.Inflate(-2, -2);
```

```
            //行番号を描画する
            TextRenderer.DrawText(e.Graphics,
                (e.RowIndex + 1).ToString(),
                e.InheritedRowStyle.Font,
                rect,
                e.InheritedRowStyle.ForeColor,
                TextFormatFlags.Right | TextFormatFlags.VerticalCenter);
        }
}
```

**44. DataGridView 焦点所在单元格焦点框不显示的设定**

[VB.NET]

```
'CellPainting イベントハンドラ
Private Sub DataGridView1_CellPainting(ByVal sender As Object, _
            ByVal e As DataGridViewCellPaintingEventArgs) _
            Handles DataGridView1.CellPainting
        'ヘッダー以外のとき
        If e.ColumnIndex >= 0 And e.RowIndex >= 0 Then
            'フォーカス枠以外が描画されるようにする
            Dim paintParts As DataGridViewPaintParts = _
                e.PaintParts And Not DataGridViewPaintParts.Focus
            'セルを描画する
            e.Paint(e.ClipBounds, paintParts)
            '描画が完了したことを知らせる
           e.Handled = True
        End If
End Sub
```

[C#]

```
//CellPainting イベントハンドラ
private void DataGridView1_CellPainting(object sender,
        DataGridViewCellPaintingEventArgs e)
{
        //ヘッダー以外のとき
       if (e.ColumnIndex >= 0 && e.RowIndex >= 0)
        {
            //フォーカス枠以外が描画されるようにする
            DataGridViewPaintParts paintParts =
                e.PaintParts & ~DataGridViewPaintParts.Focus;
          //セルを描画する
            e.Paint(e.ClipBounds, paintParts);
            //描画が完了したことを知らせる
            e.Handled = true;
        }
}
```

利用 RowPrePaint 事件实现

[VB.NET]

```
'RowPrePaint イベントハンドラ
Private Sub DataGridView1_RowPrePaint(ByVal sender As Object, _
            ByVal e As DataGridViewRowPrePaintEventArgs) _
            Handles DataGridView1.RowPrePaint
        'フォーカス枠を描画しない
```

```
    e.PaintParts = e.PaintParts And Not DataGridViewPaintParts.Focus
End Sub
```
[C#]
```
//RowPrePaint イベントハンドラ
private void DataGridView1_RowPrePaint(object sender,
    DataGridViewRowPrePaintEventArgs e)
{
    //フォーカス枠を描画しない
    e.PaintParts &= ~DataGridViewPaintParts.Focus;
}
```
DataGridView 控件用.....

**DataGridView 控件用法合集(九)**

DataGridView 中添加控件

45. DataGridView 中显示选择框 CheckBox
46. DataGridView 中显示下拉框 ComboBox
47. DataGridView 单击打开下拉框
48. DataGridView 中显示按钮
49. DataGridView 中显示链接
50. DataGridView 中显示图像

**45. DataGridView 列中显示选择框 CheckBox**

[VB.NET]
```
'CheckBox 列を追加する
Dim column As New DataGridViewCheckBoxColumn
DataGridView1.Columns.Add(column)
```
[C#]
```
//CheckBox 列を追加する
DataGridViewCheckBoxColumn column = new DataGridViewCheckBoxColumn();
DataGridView1.Columns.Add(column);
```
中间状态在内的三种状态表示

[VB.NET]
```
'3 種類のチェック状態を表示できるようにする
Dim column As DataGridViewCheckBoxColumn = _
    CType(DataGridView1.Columns(0), DataGridViewCheckBoxColumn)
column.ThreeState = True
```
[C#]
```
//3 種類のチェック状態を表示できるようにする
DataGridViewCheckBoxColumn column =
    (DataGridViewCheckBoxColumn)DataGridView1.Columns[0];
column.ThreeState = true;
```
**46. DataGridView 中显示下拉框 ComboBox**

[VB.NET]
```
'DataGridViewComboBoxColumn を作成
Dim column As New DataGridViewComboBoxColumn()
'ComboBox のリストに表示する項目を指定する
column.Items.Add("日曜日")
column.Items.Add("月曜日")
column.Items.Add("火曜日")
column.Items.Add("水曜日")
column.Items.Add("木曜日")
```

```
column.Items.Add("金曜日")
column.Items.Add("土曜日")
'"Week"列にバインドされているデータを表示する
column.DataPropertyName = "Week"
'"Week"列の代わりに ComboBox 列を表示する
DataGridView1.Columns.Insert(DataGridView1.Columns("Week").Index, column)
DataGridView1.Columns.Remove("Week")
column.Name = "Week"
```

[C#]

```
//DataGridViewComboBoxColumn を作成
DataGridViewComboBoxColumn column = new DataGridViewComboBoxColumn();
//ComboBox のリストに表示する項目を指定する
column.Items.Add("日曜日");
column.Items.Add("月曜日");
column.Items.Add("火曜日");
column.Items.Add("水曜日");
column.Items.Add("木曜日");
column.Items.Add("金曜日");
column.Items.Add("土曜日");
//"Week"列にバインドされているデータを表示する
column.DataPropertyName = "Week";
//"Week"列の代わりに ComboBox 列を表示する
DataGridView1.Columns.Insert(DataGridView1.Columns["Week"].Index, column);
DataGridView1.Columns.Remove("Week");
column.Name = "Week";
```

通过列 Data 绑定设置 ComboBox

[VB.NET]

```
'ComboBox に表示するためのリストを作成する
Dim weekTable As New DataTable("WeekTable")
weekTable.Columns.Add("Display", GetType(String))
weekTable.Columns.Add("Value", GetType(Integer))
weekTable.Rows.Add("日曜日", 0)
weekTable.Rows.Add("月曜日", 1)
weekTable.Rows.Add("火曜日", 2)
weekTable.Rows.Add("水曜日", 3)
weekTable.Rows.Add("木曜日", 4)
weekTable.Rows.Add("金曜日", 5)
weekTable.Rows.Add("土曜日", 6)
'DataGridViewComboBoxColumn を作成
Dim column As New DataGridViewComboBoxColumn()
'"Week"列にバインドされているデータを表示する
column.DataPropertyName = "Week"
'DataGridViewComboBoxColumn の DataSource を設定
column.DataSource = weekTable
'実際の値が"Value"列、表示するテキストが"Display"列とする
column.ValueMember = "Value"
column.DisplayMember = "Display"
'DataGridView1 に追加する
DataGridView1.Columns.Add(column)
```

[C#]

```
//ComboBox に表示するためのリストを作成する
DataTable weekTable = new DataTable("WeekTable");
weekTable.Columns.Add("Display", typeof(string));
weekTable.Columns.Add("Value", typeof(int));
weekTable.Rows.Add("日曜日", 0);
weekTable.Rows.Add("月曜日", 1);
weekTable.Rows.Add("火曜日", 2);
weekTable.Rows.Add("水曜日", 3);
weekTable.Rows.Add("木曜日", 4);
weekTable.Rows.Add("金曜日", 5);
weekTable.Rows.Add("土曜日", 6);
//DataGridViewComboBoxColumn を作成
DataGridViewComboBoxColumn column = new DataGridViewComboBoxColumn();
//"Week"列にバインドされているデータを表示する
column.DataPropertyName = "Week";
//DataGridViewComboBoxColumn の DataSource を設定
column.DataSource = weekTable;
//実際の値が"Value"列、表示するテキストが"Display"列とする
column.ValueMember = "Value";
column.DisplayMember = "Display";
//DataGridView1 に追加する
DataGridView1.Columns.Add(column);
```

默认状态下，所有下拉框都显示；DisplayStyleForCurrentCellOnly＝True 的状态下，当前的单元格显示下拉框，其余不显示；还有一种就是光标移动时强调显示。如下图左中右三列。

**47. DataGridView 单击打开下拉框**

通常情况下要打开下拉框需要点击目标单元格三次，第一次选中单元格，第二次进入编辑状态，第三次才能打开下拉框

[VB.NET]

```
'CellEnter イベントハンドラ
Private Sub DataGridView1_CellEnter(ByVal sender As Object, _
        ByVal e As DataGridViewCellEventArgs) _
        Handles DataGridView1.CellEnter
    Dim dgv As DataGridView = CType(sender, DataGridView)
    If dgv.Columns(e.ColumnIndex).Name = "ComboBox" AndAlso _
        TypeOf dgv.Columns(e.ColumnIndex) Is DataGridViewComboBoxColumn Then
        SendKeys.Send("{F4}")
    End If
End Sub
```

[C#]

```
//CellEnter イベントハンドラ
private void DataGridView1_CellEnter(object sender,
    DataGridViewCellEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    if (dgv.Columns[e.ColumnIndex].Name == "ComboBox" &&
        dgv.Columns[e.ColumnIndex] is DataGridViewComboBoxColumn)
    {
        SendKeys.Send("{F4}");
    }
}
```

**48. DataGridView 中显示按钮**

[VB.NET]

```
'DataGridViewButtonColumn の作成
Dim column As New DataGridViewButtonColumn()
'列の名前を設定
column.Name = "Button"
'全てのボタンに"詳細閲覧"と表示する
column.UseColumnTextForButtonValue = True
column.Text = "詳細閲覧"
'DataGridView に追加する
DataGridView1.Columns.Add(column)
```

[C#]

```
//DataGridViewButtonColumn の作成
DataGridViewButtonColumn column = new DataGridViewButtonColumn();
//列の名前を設定
column.Name = "Button";
//全てのボタンに"詳細閲覧"と表示する
column.UseColumnTextForButtonValue = true;
column.Text = "詳細閲覧";
//DataGridView に追加する
DataGridView1.Columns.Add(column);
```

按钮按下事件取得

[VB.NET]

```
'CellContentClick イベントハンドラ
Private Sub DataGridView1_CellContentClick(ByVal sender As Object, _
        ByVal e As DataGridViewCellEventArgs) _
        Handles DataGridView1.CellContentClick
    Dim dgv As DataGridView = CType(sender, DataGridView)
    '"Button"列ならば、ボタンがクリックされた
    If dgv.Columns(e.ColumnIndex).Name = "Button" Then
        MessageBox.Show((e.RowIndex.ToString() + _
            "行のボタンがクリックされました。"))
    End If
End Sub
```

[C#]

```
//CellContentClick イベントハンドラ
private void DataGridView1_CellContentClick(object sender,
    DataGridViewCellEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    //"Button"列ならば、ボタンがクリックされた
    if (dgv.Columns[e.ColumnIndex].Name == "Button")
    {
        MessageBox.Show(e.RowIndex.ToString() +
            "行のボタンがクリックされました。");
    }
}
```

**49. DataGridView 中显示链接**

[VB.NET]

```
'DataGridViewLinkColumn の作成
```

```
Dim column As New DataGridViewLinkColumn()
'列の名前を設定
column.Name = "Link"
'全てのリンクに"詳細閲覧"と表示する
column.UseColumnTextForLinkValue = True
column.Text = "詳細閲覧"
'マウスポインタがリンク上にあるときだけ下線をつける
column.LinkBehavior = LinkBehavior.HoverUnderline
'自動的に訪問済みとならないようにする
'デフォルトで True
column.TrackVisitedState = True
'DataGridView に追加する
DataGridView1.Columns.Add(column)
[C#]
//DataGridViewLinkColumn の作成
DataGridViewLinkColumn column = new DataGridViewLinkColumn();
//列の名前を設定
column.Name = "Link";
//全てのリンクに"詳細閲覧"と表示する
column.UseColumnTextForLinkValue = true;
column.Text = "詳細閲覧";
//マウスポインタがリンク上にあるときだけ下線をつける
column.LinkBehavior = LinkBehavior.HoverUnderline;
//自動的に訪問済みになるようにする
//デフォルトで True
column.TrackVisitedState = true;
//DataGridView に追加する
DataGridView1.Columns.Add(column);
链接按下事件取得
[VB.NET]
'CellContentClick イベントハンドラ
Private Sub DataGridView1_CellContentClick(ByVal sender As Object, _
        ByVal e As DataGridViewCellEventArgs) _
        Handles DataGridView1.CellContentClick
    Dim dgv As DataGridView = CType(sender, DataGridView)
    '"Link"列ならば、ボタンがクリックされた
    If dgv.Columns(e.ColumnIndex).Name = "Link" Then
        MessageBox.Show((e.RowIndex.ToString() + _
            "行のリンクがクリックされました。"))
        '訪問済みにする
        Dim cell As DataGridViewLinkCell = _
            CType(dgv(e.ColumnIndex, e.RowIndex), DataGridViewLinkCell)
        cell.LinkVisited = True
    End If
End Sub
[C#]
//CellContentClick イベントハンドラ
private void DataGridView1_CellContentClick(object sender,
    DataGridViewCellEventArgs e)
{
```

```
    DataGridView dgv = (DataGridView)sender;
    //"Link"列ならば、ボタンがクリックされた
    if (dgv.Columns[e.ColumnIndex].Name == "Link")
    {
        MessageBox.Show(e.RowIndex.ToString() +
            "行のリンクがクリックされました。");
        //訪問済みにする
        DataGridViewLinkCell cell =
            (DataGridViewLinkCell)dgv[e.ColumnIndex, e.RowIndex];
        cell.LinkVisited = true;
    }
}
```

**50. DataGridView 中显示图像**

```
[VB.NET]
'DataGridViewImageColumn の作成
Dim column As New DataGridViewImageColumn()
'列の名前を設定
column.Name = "Image"
'Icon 型ではなく、Image 型のデータを表示する
'デフォルトで False なので、変更する必要はない
column.ValuesAreIcons = False
'値の設定されていないセルに表示するイメージを設定する
column.Image = New Bitmap("C:\null.gif")
'イメージを縦横の比率を維持して拡大、縮小表示する
column.ImageLayout = DataGridViewImageCellLayout.Zoom
'イメージの説明
'セルをクリップボードにコピーした時に使用される
column.Description = "イメージ"
'DataGridView に追加する
DataGridView1.Columns.Add(column)
'"Image"列の一番上のセルのイメージを変更する
DataGridView1("Image", 0).Value = New Bitmap("C:\top.gif") '
[C#]
//DataGridViewImageColumn の作成
DataGridViewImageColumn column = new DataGridViewImageColumn();
//列の名前を設定
column.Name = "Image";
//Icon 型ではなく、Image 型のデータを表示する
//デフォルトで False なので、変更する必要はない
column.ValuesAreIcons = false;
//値の設定されていないセルに表示するイメージを設定する
column.Image = new Bitmap("C:\\null.gif");
//イメージを縦横の比率を維持して拡大、縮小表示する
column.ImageLayout = DataGridViewImageCellLayout.Zoom;
//イメージの説明
//セルをクリップボードにコピーした時に使用される
column.Description = "イメージ";
//DataGridView に追加する
DataGridView1.Columns.Add(column);
//"Image"列の一番上のセルのイメージを変更する
```

```
DataGridView1["Image", 0].Value = new Bitmap("C:\\top.gif");
```

图片属性单元格未设值时红差不显示的设定

[VB.NET]

```
'イメージ列を取得
Dim imageColumn As DataGridViewImageColumn = _
    CType(DataGridView1.Columns("Image"), DataGridViewImageColumn)
'セルスタイルの NullValue を null にする
imageColumn.DefaultCellStyle.NullValue = Nothing
```

[C#]

```
//イメージ列を取得
DataGridViewImageColumn imageColumn =
    (DataGridViewImageColumn)DataGridView1.Columns["Image"];
//セルスタイルの NullValue を null にする
imageColumn.DefaultCellStyle.NullValue = null;
```

**DataGridView 控件用法合集(十)**

51. DataGridView 编辑中单元格控件取得
52. DataGridView 输入自动完成
53. DataGridView 单元格编辑时键盘 KEY 事件取得
54. DataGridView 下拉框（ComboBox）单元格编辑时事件取得
55. DataGridView 下拉框（ComboBox）单元格允许文字输入设定

**51. DataGridView 编辑中单元格控件取得**

[VB.NET]

```
'EditingControlShowing イベントハンドラ
Private Sub DataGridView1_EditingControlShowing(ByVal sender As Object, _
        ByVal e As DataGridViewEditingControlShowingEventArgs) _
        Handles DataGridView1.EditingControlShowing
    '表示されているコントロールが DataGridViewTextBoxEditingControl か調べる
    If TypeOf e.Control Is DataGridViewTextBoxEditingControl Then
        Dim dgv As DataGridView = CType(sender, DataGridView)
        '編集のために表示されているコントロールを取得
        Dim tb As DataGridViewTextBoxEditingControl = _
            CType(e.Control, DataGridViewTextBoxEditingControl)
        '次のようにしてもよい
        'Dim tb As TextBox = CType(e.Control, TextBox)
        '列によって IME のモードを変更する
        If dgv.CurrentCell.OwningColumn.Name = "Column1" Then
            tb.ImeMode = Windows.Forms.ImeMode.Disable
        Else
            tb.ImeMode = dgv.ImeMode
        End If
    End If
End Sub
```

[C#]

```
//EditingControlShowing イベントハンドラ
private void DataGridView1_EditingControlShowing(object sender,
    DataGridViewEditingControlShowingEventArgs e)
{
    //表示されているコントロールが DataGridViewTextBoxEditingControl か調べる
    if (e.Control is DataGridViewTextBoxEditingControl)
```

```
    {
        DataGridView dgv = (DataGridView)sender;
        //編集のために表示されているコントロールを取得
        DataGridViewTextBoxEditingControl tb =
            (DataGridViewTextBoxEditingControl)e.Control;
        //次のようにしてもよい
        //TextBox tb = (TextBox)e.Control;
        //列によって IME のモードを変更する
        if (dgv.CurrentCell.OwningColumn.Name == "Column1")
            tb.ImeMode = ImeMode.Disable;
        else
            tb.ImeMode = dgv.ImeMode;
    }
}
```

其他控件以此类推，比如 DataGridViewCheckBoxColumn 或者 DataGridViewButtonColumn 等等。

**52. DataGridView 输入自动完成**

[VB.NET]

```
Dim autoCompleteSource As New AutoCompleteStringCollection()
'EditingControlShowing イベントハンドラ
Private Sub DataGridView1_EditingControlShowing( _
        ByVal sender As Object, _
        ByVal e As DataGridViewEditingControlShowingEventArgs) _
        Handles DataGridView1.EditingControlShowing
    Dim dgv As DataGridView = CType(sender, DataGridView)
    If TypeOf e.Control Is TextBox Then
        '編集のために表示されているテキストボックスを取得
        Dim tb As TextBox = CType(e.Control, TextBox)
        '該当する列か調べる
        If dgv.CurrentCell.OwningColumn.Name = "Column1" Then
            'オートコンプリートを有効にする
            tb.AutoCompleteMode = AutoCompleteMode.SuggestAppend
            tb.AutoCompleteSource = _
                Windows.Forms.AutoCompleteSource.CustomSource
            tb.AutoCompleteCustomSource = Me.autoCompleteSource
        Else
            'オートコンプリートを無効にする
            tb.AutoCompleteMode = AutoCompleteMode.None
        End If
    End If
End Sub
'DataSourceChanged イベントハンドラ
Private Sub DataGridView1_DataSourceChanged( _
        ByVal sender As Object, ByVal e As EventArgs) _
        Handles DataGridView1.DataSourceChanged
    Dim dgv As DataGridView = CType(sender, DataGridView)
    'オートコンプリートのリストを初期化
    Me.autoCompleteSource.Clear()
    'DataGridView 内のデータをリストに追加
    Dim r As DataGridViewRow
    For Each r In dgv.Rows
```

```
            'セルの値を取得
            Dim val As String = r.Cells("Column1").Value
            If Not String.IsNullOrEmpty(val) AndAlso _
                    Not Me.autoCompleteSource.Contains(val) Then
                'オートコンプリートのリストに追加
                autoCompleteSource.Add(val)
            End If
        Next r
End Sub
'CellValueChanged イベントハンドラ
Private Sub DataGridView1_CellValueChanged(ByVal sender As Object, _
            ByVal e As DataGridViewCellEventArgs) _
            Handles DataGridView1.CellValueChanged
        Dim dgv As DataGridView = CType(sender, DataGridView)
        '該当する列か調べる
        If dgv.Columns(e.ColumnIndex).Name = "Column1" Then
            'セルの値を取得
            Dim val As String = dgv(e.ColumnIndex, e.RowIndex).Value
            If Not String.IsNullOrEmpty(val) AndAlso _
                    Not Me.autoCompleteSource.Contains(val) Then
                'オートコンプリートのリストに追加
                autoCompleteSource.Add(val)
            End If
        End If
End Sub
[C#]
AutoCompleteStringCollection autoCompleteSource =
        new AutoCompleteStringCollection();
//EditingControlShowing イベントハンドラ
private void DataGridView1_EditingControlShowing(object sender,
        DataGridViewEditingControlShowingEventArgs e)
{
        DataGridView dgv = (DataGridView)sender;
        if (e.Control is TextBox)
        {
            //編集のために表示されているテキストボックスを取得
            TextBox tb = (TextBox)e.Control;
            //該当する列か調べる
            if (dgv.CurrentCell.OwningColumn.Name == "Column1")
            {
                //オートコンプリートを有効にする
                tb.AutoCompleteMode = AutoCompleteMode.SuggestAppend;
                tb.AutoCompleteSource = AutoCompleteSource.CustomSource;
                tb.AutoCompleteCustomSource = this.autoCompleteSource;
            }
            else
            {
                //オートコンプリートを無効にする
                tb.AutoCompleteMode = AutoCompleteMode.None;
            }
```

```
    }
}
//DataSourceChanged イベントハンドラ
private void DataGridView1_DataSourceChanged(object sender, EventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    //オートコンプリートのリストを初期化
    this.autoCompleteSource.Clear();
    //DataGridView 内のデータをリストに追加
    foreach (DataGridViewRow r in dgv.Rows)
    {
        //セルの値を取得
        string val = r.Cells["Column1"].Value as string;
        if (!string.IsNullOrEmpty(val) &&
            !this.autoCompleteSource.Contains(val))
        {
            //オートコンプリートのリストに追加
            autoCompleteSource.Add(val);
        }
    }
}
//CellValueChanged イベントハンドラ
private void DataGridView1_CellValueChanged(object sender,
    DataGridViewCellEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    //該当する列か調べる
    if (dgv.Columns[e.ColumnIndex].Name == "Column1")
    {
        //セルの値を取得
        string val = dgv[e.ColumnIndex, e.RowIndex].Value as string;
        if (!string.IsNullOrEmpty(val) &&
            !this.autoCompleteSource.Contains(val))
        {
            //オートコンプリートのリストに追加
            autoCompleteSource.Add(val);
        }
    }
}
```

**53. DataGridView 单元格编辑时键盘 KEY 事件取得**

[VB.NET]

```
'EditingControlShowing イベントハンドラ
Private Sub DataGridView1_EditingControlShowing(ByVal sender As Object, _
        ByVal e As DataGridViewEditingControlShowingEventArgs) _
        Handles DataGridView1.EditingControlShowing
    '表示されているコントロールが DataGridViewTextBoxEditingControl か調べる
    If TypeOf e.Control Is DataGridViewTextBoxEditingControl Then
        Dim dgv As DataGridView = CType(sender, DataGridView)
        '編集のために表示されているコントロールを取得
        Dim tb As DataGridViewTextBoxEditingControl = _
```

```
            CType(e.Control, DataGridViewTextBoxEditingControl)
        'イベントハンドラを削除
        RemoveHandler tb.KeyPress, AddressOf dataGridViewTextBox_KeyPress
        '該当する列か調べる
        If dgv.CurrentCell.OwningColumn.Name = "Column1" Then
            'KeyPress イベントハンドラを追加
            AddHandler tb.KeyPress, AddressOf dataGridViewTextBox_KeyPress
        End If
    End If
End Sub
'DataGridView に表示されているテキストボックスの KeyPress イベントハンドラ
Private Sub dataGridViewTextBox_KeyPress(ByVal sender As Object, _
        ByVal e As KeyPressEventArgs) _
        Handles DataGridView1.KeyPress
    '数字しか入力できないようにする
    If e.KeyChar < "0"c Or e.KeyChar > "9"c Then
        e.Handled = True
    End If
End Sub
[C#]
//EditingControlShowing イベントハンドラ
private void DataGridView1_EditingControlShowing(object sender,
    DataGridViewEditingControlShowingEventArgs e)
{
    //表示されているコントロールが DataGridViewTextBoxEditingControl か調べる
    if (e.Control is DataGridViewTextBoxEditingControl)
    {
        DataGridView dgv = (DataGridView)sender;
        //編集のために表示されているコントロールを取得
        DataGridViewTextBoxEditingControl tb =
            (DataGridViewTextBoxEditingControl)e.Control;
        //イベントハンドラを削除
        tb.KeyPress -=
            new KeyPressEventHandler(dataGridViewTextBox_KeyPress);
        //該当する列か調べる
        if (dgv.CurrentCell.OwningColumn.Name == "Column1")
        {
            //KeyPress イベントハンドラを追加
            tb.KeyPress +=
                new KeyPressEventHandler(dataGridViewTextBox_KeyPress);
        }
    }
}
//DataGridView に表示されているテキストボックスの KeyPress イベントハンドラ
private void dataGridViewTextBox_KeyPress(object sender,
    KeyPressEventArgs e)
{
    //数字しか入力できないようにする
    if (e.KeyChar < '0' || e.KeyChar > '9')
    {
```

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

**54. DataGridView 下拉框（ComboBox）单元格编辑时事件取得**

[VB.NET]

```
Private dataGridViewComboBox As DataGridViewComboBoxEditingControl = Nothing
'EditingControlShowing イベントハンドラ
Private Sub DataGridView1_EditingControlShowing(ByVal sender As Object, _
        ByVal e As DataGridViewEditingControlShowingEventArgs) _
        Handles DataGridView1.EditingControlShowing
    '表示されているコントロールが DataGridViewComboBoxEditingControl か調べる
    If TypeOf e.Control Is DataGridViewComboBoxEditingControl Then
        Dim dgv As DataGridView = CType(sender, DataGridView)
        '該当する列か調べる
        If dgv.CurrentCell.OwningColumn.Name = "ComboBox" Then
            '編集のために表示されているコントロールを取得
            Me.dataGridViewComboBox = _
                CType(e.Control, DataGridViewComboBoxEditingControl)
            'SelectedIndexChanged イベントハンドラを追加
            AddHandler Me.dataGridViewComboBox.SelectedIndexChanged, _
                AddressOf dataGridViewComboBox_SelectedIndexChanged
        End If
    End If
End Sub
'CellEndEdit イベントハンドラ
Private Sub DataGridView1_CellEndEdit(ByVal sender As Object, _
        ByVal e As DataGridViewCellEventArgs) _
        Handles DataGridView1.CellEndEdit
    'SelectedIndexChanged イベントハンドラを削除
    If Not (Me.dataGridViewComboBox Is Nothing) Then
        RemoveHandler Me.dataGridViewComboBox.SelectedIndexChanged, _
            AddressOf dataGridViewComboBox_SelectedIndexChanged
        Me.dataGridViewComboBox = Nothing
    End If
End Sub
'DataGridView に表示されているコンボボックスの
'SelectedIndexChanged イベントハンドラ
Private Sub dataGridViewComboBox_SelectedIndexChanged(ByVal sender As Object,ByVal e As EventArgs)
    '選択されたアイテムを表示
    Dim cb As DataGridViewComboBoxEditingControl = _
        CType(sender, DataGridViewComboBoxEditingControl)
    Console.WriteLine(cb.SelectedItem)
End Sub
```

[C#]

```
private DataGridViewComboBoxEditingControl dataGridViewComboBox = null;
//EditingControlShowing イベントハンドラ
private void DataGridView1_EditingControlShowing(object sender,
    DataGridViewEditingControlShowingEventArgs e)
{
    //表示されているコントロールが DataGridViewComboBoxEditingControl か調べる
```

```
    if (e.Control is DataGridViewComboBoxEditingControl)
    {
        DataGridView dgv = (DataGridView)sender;
        //該当する列か調べる
        if (dgv.CurrentCell.OwningColumn.Name == "ComboBox")
        {
            //編集のために表示されているコントロールを取得
            this.dataGridViewComboBox =
                (DataGridViewComboBoxEditingControl)e.Control;
            //SelectedIndexChanged イベントハンドラを追加
            this.dataGridViewComboBox.SelectedIndexChanged +=
                new EventHandler(dataGridViewComboBox_SelectedIndexChanged);
        }
    }
}
//CellEndEdit イベントハンドラ
private void DataGridView1_CellEndEdit(object sender,
    DataGridViewCellEventArgs e)
{
    //SelectedIndexChanged イベントハンドラを削除
    if (this.dataGridViewComboBox != null)
    {
        this.dataGridViewComboBox.SelectedIndexChanged -=
            new EventHandler(dataGridViewComboBox_SelectedIndexChanged);
        this.dataGridViewComboBox = null;
    }
}
//DataGridView に表示されているコンボボックスの
//SelectedIndexChanged イベントハンドラ
private void dataGridViewComboBox_SelectedIndexChanged(object sender,
    EventArgs e)
{
    //選択されたアイテムを表示
    DataGridViewComboBoxEditingControl cb =
        (DataGridViewComboBoxEditingControl)sender;
    Console.WriteLine(cb.SelectedItem);
}
```

**55. DataGridView 下拉框（ComboBox）单元格允许文字输入设定**

```
[VB.NET]
'EditingControlShowing イベントハンドラ
Private Sub DataGridView1_EditingControlShowing(ByVal sender As Object, _
        ByVal e As DataGridViewEditingControlShowingEventArgs) _
        Handles DataGridView1.EditingControlShowing
    If TypeOf e.Control Is DataGridViewComboBoxEditingControl Then
        '該当する列か調べる
        Dim dgv As DataGridView = CType(sender, DataGridView)
        If dgv.CurrentCell.OwningColumn.Name = "ComboBox" Then
            '編集のために表示されているコントロールを取得
            Dim cb As DataGridViewComboBoxEditingControl = _
                CType(e.Control, DataGridViewComboBoxEditingControl)
```

```
            cb.DropDownStyle = ComboBoxStyle.DropDown
        End If
    End If
End Sub
'CellValidating イベントハンドラ
Private Sub DataGridView1_CellValidating(ByVal sender As Object, _
        ByVal e As DataGridViewCellValidatingEventArgs) _
        Handles DataGridView1.CellValidating
    Dim dgv As DataGridView = CType(sender, DataGridView)
    '該当する列か調べる
    If dgv.Columns(e.ColumnIndex).Name = "ComboBox" AndAlso _
        TypeOf dgv.Columns(e.ColumnIndex) Is DataGridViewComboBoxColumn Then
        Dim cbc As DataGridViewComboBoxColumn = _
            CType(dgv.Columns(e.ColumnIndex), DataGridViewComboBoxColumn)
        'コンボボックスの項目に追加する
        If Not cbc.Items.Contains(e.FormattedValue) Then
            cbc.Items.Add(e.FormattedValue)
        End If
    End If
End Sub
[C#]
//EditingControlShowing イベントハンドラ
private void DataGridView1_EditingControlShowing(object sender,
    DataGridViewEditingControlShowingEventArgs e)
{
    if (e.Control is DataGridViewComboBoxEditingControl)
    {
        //該当する列か調べる
        DataGridView dgv = (DataGridView)sender;
        if (dgv.CurrentCell.OwningColumn.Name == "ComboBox")
        {
            //編集のために表示されているコントロールを取得
            DataGridViewComboBoxEditingControl cb =
                (DataGridViewComboBoxEditingControl)e.Control;
            cb.DropDownStyle = ComboBoxStyle.DropDown;
        }
    }
}
//CellValidating イベントハンドラ
private void DataGridView1_CellValidating(object sender,
    DataGridViewCellValidatingEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    //該当する列か調べる
    if (dgv.Columns[e.ColumnIndex].Name == "ComboBox" &&
        dgv.Columns[e.ColumnIndex] is DataGridViewComboBoxColumn)
    {
        DataGridViewComboBoxColumn cbc =
            (DataGridViewComboBoxColumn)dgv.Columns[e.ColumnIndex];
        //コンボボックスの項目に追加する
```

```
            if (!cbc.Items.Contains(e.FormattedValue))
            {
                cbc.Items.Add(e.FormattedValue);
            }
        }
}
```

**DataGridView 控件用法合集(十一)**

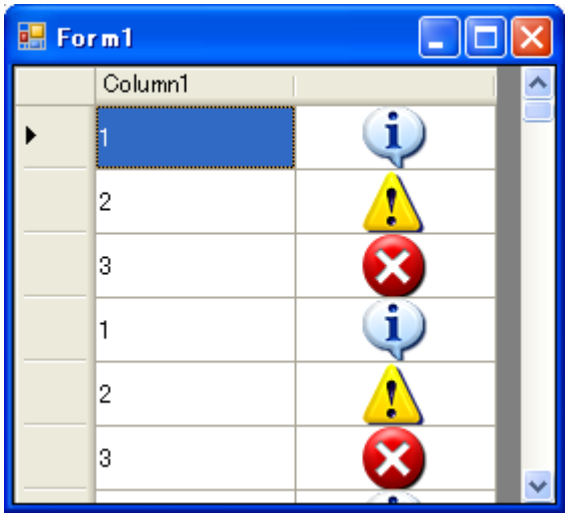
DataGridView 特殊控件

56. DataGridView 根据值不同在另一列中显示相应图片

57. DataGridView 中显示进度条（ProgressBar）

58. DataGridView 中添加 MaskedTextBox

**56. DataGridView 根据值不同在另一列中显示相应图片**




**57. DataGridView 中显示进度条（ProgressBar）**

[VB.NET]

```
Imports System
Imports System.Drawing
Imports System.Windows.Forms
''' <summary>
''' DataGridViewProgressBarCell オブジェクトの列
''' </summary>
Public Class DataGridViewProgressBarColumn
    Inherits DataGridViewTextBoxColumn
    'コンストラクタ
    Public Sub New()
        Me.CellTemplate = New DataGridViewProgressBarCell()
    End Sub
    'CellTemplate の取得と設定
    Public Overrides Property CellTemplate() As DataGridViewCell
        Get
            Return MyBase.CellTemplate
        End Get
        Set(ByVal value As DataGridViewCell)
            'DataGridViewProgressBarCell 以外はホストしない
            If Not TypeOf value Is DataGridViewProgressBarCell Then
                Throw New InvalidCastException( _
                    "DataGridViewProgressBarCell オブジェクトを" + _
```

```
                    "指定してください。")
            End If
            MyBase.CellTemplate = value
        End Set
    End Property
    ''' <summary>
    ''' ProgressBar の最大値
    ''' </summary>
    Public Property Maximum() As Integer
        Get
            Return CType(Me.CellTemplate, DataGridViewProgressBarCell).Maximum
        End Get
        Set(ByVal value As Integer)
            If Me.Maximum = value Then
                Return
            End If
            'セルテンプレートの値を変更する
            CType(Me.CellTemplate, DataGridViewProgressBarCell).Maximum = value
            'DataGridView にすでに追加されているセルの値を変更する
            If Me.DataGridView Is Nothing Then
                Return
            End If
            Dim rowCount As Integer = Me.DataGridView.RowCount
            Dim i As Integer
            For i = 0 To rowCount - 1
                Dim r As DataGridViewRow = Me.DataGridView.Rows.SharedRow(i)
                CType(r.Cells(Me.Index), DataGridViewProgressBarCell).Maximum = _
                    value
            Next i
        End Set
    End Property
    ''' <summary>
    ''' ProgressBar の最小値
    ''' </summary>
    Public Property Mimimum() As Integer
        Get
            Return CType(Me.CellTemplate, DataGridViewProgressBarCell).Mimimum
        End Get
        Set(ByVal value As Integer)
            If Me.Mimimum = value Then
                Return
            End If
            'セルテンプレートの値を変更する
            CType(Me.CellTemplate, DataGridViewProgressBarCell).Mimimum = value
            'DataGridView にすでに追加されているセルの値を変更する
            If Me.DataGridView Is Nothing Then
                Return
            End If
            Dim rowCount As Integer = Me.DataGridView.RowCount
            Dim i As Integer
```

```
                For i = 0 To rowCount - 1
                    Dim r As DataGridViewRow = Me.DataGridView.Rows.SharedRow(i)
                    CType(r.Cells(Me.Index), DataGridViewProgressBarCell).Mimimum = _
                        value
                Next i
            End Set
        End Property
End Class
''' <summary>
''' ProgressBar を DataGridView に表示する
''' </summary>
Public Class DataGridViewProgressBarCell
    Inherits DataGridViewTextBoxCell
    'コンストラクタ
    Public Sub New()
        Me.maximumValue = 100
        Me.mimimumValue = 0
    End Sub
    Private maximumValue As Integer
    Public Property Maximum() As Integer
        Get
            Return Me.maximumValue
        End Get
        Set(ByVal value As Integer)
            Me.maximumValue = value
        End Set
    End Property
    Private mimimumValue As Integer
    Public Property Mimimum() As Integer
        Get
            Return Me.mimimumValue
        End Get
        Set(ByVal value As Integer)
            Me.mimimumValue = value
        End Set
    End Property
    'セルの値のデータ型を指定する
    'ここでは、整数型とする
    Public Overrides ReadOnly Property ValueType() As Type
        Get
            Return GetType(Integer)
        End Get
    End Property
    '新しいレコード行のセルの既定値を指定する
    Public Overrides ReadOnly Property DefaultNewRowValue() As Object
        Get
            Return 0
        End Get
    End Property
    '新しいプロパティを追加しているため、
```

```
' Clone メソッドをオーバーライドする必要がある
Public Overrides Function Clone() As Object
    Dim cell As DataGridViewProgressBarCell = _
        CType(MyBase.Clone(), DataGridViewProgressBarCell)
    cell.Maximum = Me.Maximum
    cell.Mimimum = Me.Mimimum
    Return cell
End Function
Protected Overrides Sub Paint(ByVal graphics As Graphics, _
    ByVal clipBounds As Rectangle, _
    ByVal cellBounds As Rectangle, _
    ByVal rowIndex As Integer, _
    ByVal cellState As DataGridViewElementStates, _
    ByVal value As Object, _
    ByVal formattedValue As Object, _
    ByVal errorText As String, _
    ByVal cellStyle As DataGridViewCellStyle, _
    ByVal advancedBorderStyle As DataGridViewAdvancedBorderStyle, _
    ByVal paintParts As DataGridViewPaintParts)
    '値を決定する
    Dim intValue As Integer = 0
    If TypeOf value Is Integer Then
        intValue = CInt(value)
    End If
    If intValue < Me.mimimumValue Then
        intValue = Me.mimimumValue
    End If
    If intValue > Me.maximumValue Then
        intValue = Me.maximumValue
    End If
    '割合を計算する
    Dim rate As Double = CDbl(intValue - Me.mimimumValue) / _
        (Me.maximumValue - Me.mimimumValue)
    'セルの境界線（枠）を描画する
    If (paintParts And DataGridViewPaintParts.Border) = _
            DataGridViewPaintParts.Border Then
        Me.PaintBorder(graphics, clipBounds, cellBounds, _
            cellStyle, advancedBorderStyle)
    End If
    '境界線の内側に範囲を取得する
    Dim borderRect As Rectangle = Me.BorderWidths(advancedBorderStyle)
    Dim paintRect As New Rectangle(cellBounds.Left + borderRect.Left, _
        cellBounds.Top + borderRect.Top, _
        cellBounds.Width - borderRect.Right, _
        cellBounds.Height - borderRect.Bottom)
   '背景色を決定する
    '選択されている時とされていない時で色を変える
    Dim isSelected As Boolean = _
        ((cellState And DataGridViewElementStates.Selected) = _
            DataGridViewElementStates.Selected)
```

```
Dim bkColor As Color
If isSelected AndAlso _
    (paintParts And DataGridViewPaintParts.SelectionBackground) = _
        DataGridViewPaintParts.SelectionBackground Then
    bkColor = cellStyle.SelectionBackColor
Else
    bkColor = cellStyle.BackColor
End If
'背景を描画する
If (paintParts And DataGridViewPaintParts.Background) = _
    DataGridViewPaintParts.Background Then
    Dim backBrush As New SolidBrush(bkColor)
    Try
        graphics.FillRectangle(backBrush, paintRect)
    Finally
        backBrush.Dispose()
    End Try
End If
'Padding を差し引く
paintRect.Offset(cellStyle.Padding.Right, cellStyle.Padding.Top)
paintRect.Width -= cellStyle.Padding.Horizontal
paintRect.Height -= cellStyle.Padding.Vertical
'ProgressBar を描画する
If (paintParts And DataGridViewPaintParts.ContentForeground) = _
    DataGridViewPaintParts.ContentForeground Then
    If ProgressBarRenderer.IsSupported Then
        'visual スタイルで描画する
        'ProgressBar の枠を描画する
        ProgressBarRenderer.DrawHorizontalBar(graphics, paintRect)
        'ProgressBar のバーを描画する
        Dim barBounds As New Rectangle(paintRect.Left + 3, _
            paintRect.Top + 3, _
            paintRect.Width - 4, _
            paintRect.Height - 6)
        barBounds.Width = CInt(Math.Round((barBounds.Width * rate)))
        ProgressBarRenderer.DrawHorizontalChunks(graphics, barBounds)
    Else
        'visual スタイルで描画できない時
        graphics.FillRectangle(Brushes.White, paintRect)
        graphics.DrawRectangle(Pens.Black, paintRect)
        Dim barBounds As New Rectangle(paintRect.Left + 1, _
            paintRect.Top + 1, _
            paintRect.Width - 1, _
            paintRect.Height - 1)
        barBounds.Width = CInt(Math.Round((barBounds.Width * rate)))
        graphics.FillRectangle(Brushes.Blue, barBounds)
    End If
End If
'フォーカスの枠を表示する
If Me.DataGridView.CurrentCellAddress.X = Me.ColumnIndex AndAlso _
```

```
            Me.DataGridView.CurrentCellAddress.Y = Me.RowIndex AndAlso _
            (paintParts And DataGridViewPaintParts.Focus) = _
                DataGridViewPaintParts.Focus AndAlso _
            Me.DataGridView.Focused Then
            'フォーカス枠の大きさを適当に決める
            Dim focusRect As Rectangle = paintRect
            focusRect.Inflate(-3, -3)
            ControlPaint.DrawFocusRectangle(graphics, focusRect)
            '背景色を指定してフォーカス枠を描画する時
            'ControlPaint.DrawFocusRectangle(
            '    graphics, focusRect, Color.Empty, bkColor);
        End If
        'テキストを表示する
        If (paintParts And DataGridViewPaintParts.ContentForeground) = _
            DataGridViewPaintParts.ContentForeground Then
            '表示するテキストを決定
            Dim txt As String = String.Format("{0}%", Math.Round((rate * 100)))
            'string txt = formattedValue.ToString();
            '本来は、cellStyle により TextFormatFlags を決定すべき
            Dim flags As TextFormatFlags = _
                TextFormatFlags.HorizontalCenter Or _
                    TextFormatFlags.VerticalCenter
            '色を決定
            Dim fColor As Color = cellStyle.ForeColor
            'if (isSelected)
            '    fColor = cellStyle.SelectionForeColor;
            'else
            '    fColor = cellStyle.ForeColor;
            'テキストを描画する
            paintRect.Inflate(-2, -2)
            TextRenderer.DrawText( _
                graphics, txt, cellStyle.Font, paintRect, fColor, flags)
        End If
        'エラーアイコンの表示
        If (paintParts And DataGridViewPaintParts.ErrorIcon) = _
                DataGridViewPaintParts.ErrorIcon AndAlso _
            Me.DataGridView.ShowCellErrors AndAlso _
            Not String.IsNullOrEmpty(errorText) Then
            'エラーアイコンを表示させる領域を取得
            Dim iconBounds As Rectangle = _
                Me.GetErrorIconBounds(graphics, cellStyle, rowIndex)
            iconBounds.Offset(cellBounds.X, cellBounds.Y)
            'エラーアイコンを描画
            Me.PaintErrorIcon(graphics, iconBounds, cellBounds, errorText)
        End If
    End Sub
End Class
```

[C#]

```
using System;
using System.Drawing;
```

```
using System.Windows.Forms;
/// <summary>
/// DataGridViewProgressBarCell オブジェクトの列
/// </summary>
public class DataGridViewProgressBarColumn : DataGridViewTextBoxColumn
{
    //コンストラクタ
    public DataGridViewProgressBarColumn()
    {
        this.CellTemplate = new DataGridViewProgressBarCell();
    }
    //CellTemplate の取得と設定
    public override DataGridViewCell CellTemplate
    {
        get
        {
            return base.CellTemplate;
        }
        set
        {
            //DataGridViewProgressBarCell 以外はホストしない
            if (!(value is DataGridViewProgressBarCell))
            {
                throw new InvalidCastException(
                    "DataGridViewProgressBarCell オブジェクトを" +
                    "指定してください。");
            }
            base.CellTemplate = value;
        }
    }
    /// <summary>
    /// ProgressBar の最大値
    /// </summary>
    public int Maximum
    {
        get
        {
            return ((DataGridViewProgressBarCell)this.CellTemplate).Maximum;
        }
        set
        {
            if (this.Maximum == value)
                return;
            //セルテンプレートの値を変更する
            ((DataGridViewProgressBarCell)this.CellTemplate).Maximum =
                value;
            //DataGridView にすでに追加されているセルの値を変更する
            if (this.DataGridView == null)
                return;
            int rowCount = this.DataGridView.RowCount;
```

```
                for (int i = 0; i < rowCount; i++)
                {
                    DataGridViewRow r = this.DataGridView.Rows.SharedRow(i);
                    ((DataGridViewProgressBarCell)r.Cells[this.Index]).Maximum =
                        value;
                }
            }
        }
        /// <summary>
        /// ProgressBar の最小値
        /// </summary>
        public int Mimimum
        {
            get
            {
                return ((DataGridViewProgressBarCell)this.CellTemplate).Mimimum;
            }
            set
            {
                if (this.Mimimum == value)
                    return;
                //セルテンプレートの値を変更する
                ((DataGridViewProgressBarCell)this.CellTemplate).Mimimum =
                    value;
                //DataGridView にすでに追加されているセルの値を変更する
                if (this.DataGridView == null)
                    return;
                int rowCount = this.DataGridView.RowCount;
                for (int i = 0; i < rowCount; i++)
                {
                    DataGridViewRow r = this.DataGridView.Rows.SharedRow(i);
                    ((DataGridViewProgressBarCell)r.Cells[this.Index]).Mimimum =
                        value;
                }
            }
        }
    }
    /// <summary>
    /// ProgressBar を DataGridView に表示する
    /// </summary>
    public class DataGridViewProgressBarCell : DataGridViewTextBoxCell
    {
        //コンストラクタ
        public DataGridViewProgressBarCell()
        {
            this.maximumValue = 100;
            this.mimimumValue = 0;
        }
        private int maximumValue;
        public int Maximum
```

```
{
    get
    {
        return this.maximumValue;
    }
    set
    {
        this.maximumValue = value;
    }
}
private int mimimumValue;
public int Mimimum
{
    get
    {
        return this.mimimumValue;
    }
    set
    {
        this.mimimumValue = value;
    }
}
//セルの値のデータ型を指定する
//ここでは、整数型とする
public override Type ValueType
{
    get
    {
        return typeof(int);
    }
}
//新しいレコード行のセルの既定値を指定する
public override object DefaultNewRowValue
{
    get
    {
        return 0;
    }
}
//新しいプロパティを追加しているため、
// Clone メソッドをオーバーライドする必要がある
public override object Clone()
{
    DataGridViewProgressBarCell cell =
        (DataGridViewProgressBarCell)base.Clone();
    cell.Maximum = this.Maximum;
    cell.Mimimum = this.Mimimum;
    return cell;
}
protected override void Paint(Graphics graphics,
```

```
        Rectangle clipBounds, Rectangle cellBounds,
        int rowIndex, DataGridViewElementStates cellState,
        object value, object formattedValue, string errorText,
        DataGridViewCellStyle cellStyle,
        DataGridViewAdvancedBorderStyle advancedBorderStyle,
        DataGridViewPaintParts paintParts)
{
        //値を決定する
        int intValue = 0;
        if (value is int)
            intValue = (int)value;
        if (intValue < this.mimimumValue)
            intValue = this.mimimumValue;
        if (intValue > this.maximumValue)
            intValue = this.maximumValue;
        //割合を計算する
        double rate = (double)(intValue - this.mimimumValue) /
            (this.maximumValue - this.mimimumValue);
        //セルの境界線（枠）を描画する
        if ((paintParts & DataGridViewPaintParts.Border) ==
            DataGridViewPaintParts.Border)
        {
            this.PaintBorder(graphics, clipBounds, cellBounds,
                cellStyle, advancedBorderStyle);
        }
        //境界線の内側に範囲を取得する
        Rectangle borderRect = this.BorderWidths(advancedBorderStyle);
        Rectangle paintRect = new Rectangle(
            cellBounds.Left + borderRect.Left,
            cellBounds.Top + borderRect.Top,
            cellBounds.Width - borderRect.Right,
            cellBounds.Height - borderRect.Bottom);
        //背景色を決定する
       //選択されている時とされていない時で色を変える
        bool isSelected =
            (cellState & DataGridViewElementStates.Selected) ==
            DataGridViewElementStates.Selected;
      Color bkColor;
        if (isSelected &&
            (paintParts & DataGridViewPaintParts.SelectionBackground) ==
                DataGridViewPaintParts.SelectionBackground)
        {
            bkColor = cellStyle.SelectionBackColor;
        }
        else
        {
            bkColor = cellStyle.BackColor;
        }
        //背景を描画する
        if ((paintParts & DataGridViewPaintParts.Background) ==
```

```
        DataGridViewPaintParts.Background)
    {
        using (SolidBrush backBrush = new SolidBrush(bkColor))
        {
            graphics.FillRectangle(backBrush, paintRect);
        }
    }
    //Padding を差し引く
    paintRect.Offset(cellStyle.Padding.Right, cellStyle.Padding.Top);
    paintRect.Width -= cellStyle.Padding.Horizontal;
    paintRect.Height -= cellStyle.Padding.Vertical;
    //ProgressBar を描画する
    if ((paintParts & DataGridViewPaintParts.ContentForeground) ==
        DataGridViewPaintParts.ContentForeground)
    {
        if (ProgressBarRenderer.IsSupported)
        {
            //visual スタイルで描画する
            //ProgressBar の枠を描画する
            ProgressBarRenderer.DrawHorizontalBar(graphics, paintRect);
            //ProgressBar のバーを描画する
            Rectangle barBounds = new Rectangle(
                paintRect.Left + 3, paintRect.Top + 3,
                paintRect.Width - 4, paintRect.Height - 6);
            barBounds.Width = (int)Math.Round(barBounds.Width * rate);
            ProgressBarRenderer.DrawHorizontalChunks(graphics, barBounds);
        }
        else
        {
            //visual スタイルで描画できない時
            graphics.FillRectangle(Brushes.White, paintRect);
            graphics.DrawRectangle(Pens.Black, paintRect);
            Rectangle barBounds = new Rectangle(
                paintRect.Left + 1, paintRect.Top + 1,
                paintRect.Width - 1, paintRect.Height - 1);
            barBounds.Width = (int)Math.Round(barBounds.Width * rate);
            graphics.FillRectangle(Brushes.Blue, barBounds);
        }
    }
    //フォーカスの枠を表示する
    if (this.DataGridView.CurrentCellAddress.X == this.ColumnIndex &&
        this.DataGridView.CurrentCellAddress.Y == this.RowIndex &&
        (paintParts & DataGridViewPaintParts.Focus) ==
            DataGridViewPaintParts.Focus &&
        this.DataGridView.Focused)
    {
        //フォーカス枠の大きさを適当に決める
        Rectangle focusRect = paintRect;
        focusRect.Inflate(-3, -3);
        ControlPaint.DrawFocusRectangle(graphics, focusRect);
```

```
            //背景色を指定してフォーカス枠を描画する時
            //ControlPaint.DrawFocusRectangle(
            //        graphics, focusRect, Color.Empty, bkColor);
        }
        //テキストを表示する
        if ((paintParts & DataGridViewPaintParts.ContentForeground) ==
            DataGridViewPaintParts.ContentForeground)
        {
            //表示するテキストを決定
            string txt = string.Format("{0}%", Math.Round(rate * 100));
            //string txt = formattedValue.ToString();
           //本来は、cellStyle により TextFormatFlags を決定すべき
            TextFormatFlags flags = TextFormatFlags.HorizontalCenter |
                TextFormatFlags.VerticalCenter;
            //色を決定
            Color fColor = cellStyle.ForeColor;
            //if (isSelected)
            //    fColor = cellStyle.SelectionForeColor;
            //else
            //    fColor = cellStyle.ForeColor;
            //テキストを描画する
            paintRect.Inflate(-2, -2);
            TextRenderer.DrawText(graphics, txt, cellStyle.Font,
                paintRect, fColor, flags);
        }
        //エラーアイコンの表示
        if ((paintParts & DataGridViewPaintParts.ErrorIcon) ==
            DataGridViewPaintParts.ErrorIcon &&
            this.DataGridView.ShowCellErrors &&
            !string.IsNullOrEmpty(errorText))
        {
            //エラーアイコンを表示させる領域を取得
            Rectangle iconBounds = this.GetErrorIconBounds(
                graphics, cellStyle, rowIndex);
            iconBounds.Offset(cellBounds.X, cellBounds.Y);
            //エラーアイコンを描画
            this.PaintErrorIcon(graphics, iconBounds, cellBounds, errorText);
        }
    }
}
```

用法如下

[VB.NET]

```
'DataGridViewProgressBarColumn を作成する
Dim pbColumn As New DataGridViewProgressBarColumn()
'データソースの"Column1"をバインドする
pbColumn.DataPropertyName = "Column1"
'列を追加する
DataGridView1.Columns.Add(pbColumn)
```

[C#]

```
//DataGridViewProgressBarColumn を作成する
```

```
DataGridViewProgressBarColumn pbColumn =
    new DataGridViewProgressBarColumn();
//データソースの"Column1"をバインドする
pbColumn.DataPropertyName = "Column1";
//列を追加する
DataGridView1.Columns.Add(pbColumn);
```

**58. DataGridView 中添加 MaskedTextBox**

[VB.NET]

```
Imports System
Imports System.Windows.Forms
''' <summary>
''' DataGridViewMaskedTextBoxCell オブジェクトの列を表します。
''' </summary>
Public Class DataGridViewMaskedTextBoxColumn
    Inherits DataGridViewColumn
    'CellTemplate とする DataGridViewMaskedTextBoxCell オブジェクトを指定して
    '基本クラスのコンストラクタを呼び出す
    Public Sub New()
        MyBase.New(New DataGridViewMaskedTextBoxCell())
    End Sub
    Private maskValue As String = ""
    ''' <summary>
    ''' MaskedTextBox の Mask プロパティに適用する値
    ''' </summary>
    Public Property Mask() As String
        Get
            Return Me.maskValue
        End Get
        Set(ByVal value As String)
            Me.maskValue = value
        End Set
    End Property
    '新しいプロパティを追加しているため、
    ' Clone メソッドをオーバーライドする必要がある
    Public Overrides Function Clone() As Object
        Dim col As DataGridViewMaskedTextBoxColumn = _
            CType(MyBase.Clone(), DataGridViewMaskedTextBoxColumn)
        col.Mask = Me.Mask
        Return col
    End Function
    'CellTemplate の取得と設定
    Public Overrides Property CellTemplate() As DataGridViewCell
        Get
            Return MyBase.CellTemplate
        End Get
        Set(ByVal value As DataGridViewCell)
            'DataGridViewMaskedTextBoxCell しか
            ' CellTemplate に設定できないようにする
            If Not TypeOf value Is DataGridViewMaskedTextBoxCell Then
                Throw New InvalidCastException( _
```

```
                    "DataGridViewMaskedTextBoxCell オブジェクトを" + _
                    "指定してください。")
            End If
            MyBase.CellTemplate = value
        End Set
    End Property
End Class
''' <summary>
''' MaskedTextBox で編集できるテキスト情報を
''' DataGridView コントロールに表示します。
''' </summary>
Public Class DataGridViewMaskedTextBoxCell
    Inherits DataGridViewTextBoxCell
    'コンストラクタ
    Public Sub New()
    End Sub
   '編集コントロールを初期化する
    '編集コントロールは別のセルや列でも使いまわされるため、初期化の必要がある
    Public Overrides Sub InitializeEditingControl(ByVal rowIndex As Integer, _
        ByVal initialFormattedValue As Object, _
        ByVal dataGridViewCellStyle As DataGridViewCellStyle)
        MyBase.InitializeEditingControl(rowIndex, initialFormattedValue, _
          dataGridViewCellStyle)
        '編集コントロールの取得
        Dim maskedBox As DataGridViewMaskedTextBoxEditingControl = _
            Me.DataGridView.EditingControl
        If Not (maskedBox Is Nothing) Then
            'Text を設定
            maskedBox.Text = IIf(Me.Value Is Nothing, "", Me.Value.ToString())
            'カスタム列のプロパティを反映させる
            Dim column As DataGridViewMaskedTextBoxColumn = Me.OwningColumn
            If Not (column Is Nothing) Then
                maskedBox.Mask = column.Mask
            End If
        End If
    End Sub
    '編集コントロールの型を指定する
    Public Overrides ReadOnly Property EditType() As Type
        Get
            Return GetType(DataGridViewMaskedTextBoxEditingControl)
        End Get
    End Property
    'セルの値のデータ型を指定する
    'ここでは、Object 型とする
    '基本クラスと同じなので、オーバーライドの必要なし
    Public Overrides ReadOnly Property ValueType() As Type
        Get
            Return GetType(Object)
        End Get
    End Property
```

```
    '新しいレコード行のセルの既定値を指定する
    Public Overrides ReadOnly Property DefaultNewRowValue() As Object
        Get
            Return MyBase.DefaultNewRowValue
        End Get
    End Property
End Class
''' <summary>
''' DataGridViewMaskedTextBoxCell でホストされる
''' MaskedTextBox コントロールを表します。
''' </summary>
Public Class DataGridViewMaskedTextBoxEditingControl
    Inherits MaskedTextBox
    Implements IDataGridViewEditingControl
    '編集コントロールが表示されている DataGridView
    Private dataGridView As DataGridView
    '編集コントロールが表示されている行
    Private rowIndex As Integer
    '編集コントロールの値とセルの値が違うかどうか
    Private valueChanged As Boolean
    'コンストラクタ
    Public Sub New()
        Me.TabStop = False
    End Sub
    '編集コントロールで変更されたセルの値
    Public Function GetEditingControlFormattedValue( _
        ByVal context As DataGridViewDataErrorContexts) As Object _
        Implements IDataGridViewEditingControl.GetEditingControlFormattedValue
        Return Me.Text
    End Function
    '編集コントロールで変更されたセルの値
    Public Property EditingControlFormattedValue() As Object _
        Implements IDataGridViewEditingControl.EditingControlFormattedValue
        Get
            Return Me.GetEditingControlFormattedValue( _
                DataGridViewDataErrorContexts.Formatting)
        End Get
        Set(ByVal value As Object)
            Me.Text = CStr(value)
        End Set
    End Property
    'セルスタイルを編集コントロールに適用する
    '編集コントロールの前景色、背景色、フォントなどをセルスタイルに合わせる
    Public Sub ApplyCellStyleToEditingControl( _
        ByVal dataGridViewCellStyle As DataGridViewCellStyle) _
        Implements IDataGridViewEditingControl.ApplyCellStyleToEditingControl
        Me.Font = dataGridViewCellStyle.Font
        Me.ForeColor = dataGridViewCellStyle.ForeColor
        Me.BackColor = dataGridViewCellStyle.BackColor
        Select Case dataGridViewCellStyle.Alignment
```

```
            Case DataGridViewContentAlignment.BottomCenter, _
                    DataGridViewContentAlignment.MiddleCenter, _
                    DataGridViewContentAlignment.TopCenter
                Me.TextAlign = HorizontalAlignment.Center
            Case DataGridViewContentAlignment.BottomRight, _
                    DataGridViewContentAlignment.MiddleRight, _
                    DataGridViewContentAlignment.TopRight
                Me.TextAlign = HorizontalAlignment.Right
            Case Else
                Me.TextAlign = HorizontalAlignment.Left
        End Select
    End Sub
    '編集するセルがある DataGridView
    Public Property EditingControlDataGridView() As DataGridView _
        Implements IDataGridViewEditingControl.EditingControlDataGridView
        Get
            Return Me.dataGridView
        End Get
        Set(ByVal value As DataGridView)
            Me.dataGridView = value
        End Set
    End Property
    '編集している行のインデックス
    Public Property EditingControlRowIndex() As Integer _
        Implements IDataGridViewEditingControl.EditingControlRowIndex
        Get
            Return Me.rowIndex
        End Get
        Set(ByVal value As Integer)
            Me.rowIndex = value
        End Set
    End Property
    '値が変更されたかどうか
    '編集コントロールの値とセルの値が違うかどうか
    Public Property EditingControlValueChanged() As Boolean _
        Implements IDataGridViewEditingControl.EditingControlValueChanged
        Get
            Return Me.valueChanged
        End Get
        Set(ByVal value As Boolean)
            Me.valueChanged = value
        End Set
    End Property
    '指定されたキーを DataGridView が処理するか、編集コントロールが処理するか
    'True を返すと、編集コントロールが処理する
    'dataGridViewWantsInputKey が True の時は、DataGridView が処理できる
    Public Function EditingControlWantsInputKey(ByVal keyData As Keys, _
        ByVal dataGridViewWantsInputKey As Boolean) As Boolean _
        Implements IDataGridViewEditingControl.EditingControlWantsInputKey
        'Keys.Left、Right、Home、End の時は、True を返す
```

```
        'このようにしないと、これらのキーで別のセルにフォーカスが移ってしまう
        Select Case keyData And Keys.KeyCode
            Case Keys.Right, Keys.End, Keys.Left, Keys.Home
                Return True
            Case Else
                Return False
        End Select
    End Function
    'マウスカーソルが EditingPanel 上にあるときのカーソルを指定する
    'EditingPanel は編集コントロールをホストするパネルで、
    '編集コントロールがセルより小さいとコントロール以外の部分がパネルとなる
    Public ReadOnly Property EditingPanelCursor() As Cursor _
        Implements IDataGridViewEditingControl.EditingPanelCursor
        Get
            Return MyBase.Cursor
        End Get
    End Property
    'コントロールで編集する準備をする
    'テキストを選択状態にしたり、挿入ポインタを末尾にしたりする
    Public Sub PrepareEditingControlForEdit(ByVal selectAll As Boolean) _
        Implements IDataGridViewEditingControl.PrepareEditingControlForEdit
        If selectAll Then
            '選択状態にする
            Me.SelectAll()
        Else
            '挿入ポインタを末尾にする
            Me.SelectionStart = Me.TextLength
        End If
    End Sub
    '値が変更した時に、セルの位置を変更するかどうか
    '値が変更された時に編集コントロールの大きさが変更される時は True
    Public ReadOnly Property RepositionEditingControlOnValueChange() _
        As Boolean _
        Implements _
            IDataGridViewEditingControl.RepositionEditingControlOnValueChange
        Get
            Return False
        End Get
    End Property
    '値が変更された時
    Protected Overrides Sub OnTextChanged(ByVal e As EventArgs)
        MyBase.OnTextChanged(e)
        '値が変更されたことを DataGridView に通知する
        Me.valueChanged = True
        Me.dataGridView.NotifyCurrentCellDirty(True)
    End Sub
End Class
[C#]
using System;
using System.Windows.Forms;
```

```
/// <summary>
/// DataGridViewMaskedTextBoxCell オブジェクトの列を表します。
/// </summary>
public class DataGridViewMaskedTextBoxColumn :
    DataGridViewColumn
{
    //CellTemplate とする DataGridViewMaskedTextBoxCell オブジェクトを指定して
    //基本クラスのコンストラクタを呼び出す
    public DataGridViewMaskedTextBoxColumn()
        : base(new DataGridViewMaskedTextBoxCell())
    {
    }
    private string maskValue = "";
    /// <summary>
    /// MaskedTextBox の Mask プロパティに適用する値
    /// </summary>
    public string Mask
    {
        get
        {
            return this.maskValue;
        }
        set
        {
            this.maskValue = value;
        }
    }
    //新しいプロパティを追加しているため、
    // Clone メソッドをオーバーライドする必要がある
    public override object Clone()
    {
        DataGridViewMaskedTextBoxColumn col =
            (DataGridViewMaskedTextBoxColumn)base.Clone();
        col.Mask = this.Mask;
        return col;
    }
    //CellTemplate の取得と設定
    public override DataGridViewCell CellTemplate
    {
        get
        {
            return base.CellTemplate;
        }
        set
        {
            //DataGridViewMaskedTextBoxCell しか
            // CellTemplate に設定できないようにする
            if (!(value is DataGridViewMaskedTextBoxCell))
            {
                throw new InvalidCastException(
```

```
                    "DataGridViewMaskedTextBoxCell オブジェクトを" +
                    "指定してください。");
            }
            base.CellTemplate = value;
        }
    }
}
// <summary>
/// MaskedTextBox で編集できるテキスト情報を
/// DataGridView コントロールに表示します。
/// </summary>
public class DataGridViewMaskedTextBoxCell :
    DataGridViewTextBoxCell
{
    //コンストラクタ
    public DataGridViewMaskedTextBoxCell()
    {
    }
    //編集コントロールを初期化する
    //編集コントロールは別のセルや列でも使いまわされるため、初期化の必要がある
    public override void InitializeEditingControl(
        int rowIndex, object initialFormattedValue,
        DataGridViewCellStyle dataGridViewCellStyle)
    {
        base.InitializeEditingControl(rowIndex,
            initialFormattedValue, dataGridViewCellStyle);
        //編集コントロールの取得
        DataGridViewMaskedTextBoxEditingControl maskedBox =
            this.DataGridView.EditingControl as
            DataGridViewMaskedTextBoxEditingControl;
        if (maskedBox != null)
        {
            //Text を設定
            maskedBox.Text =
                this.Value != null ? this.Value.ToString() : "";
            //カスタム列のプロパティを反映させる
            DataGridViewMaskedTextBoxColumn column =
                this.OwningColumn as DataGridViewMaskedTextBoxColumn;
            if (column != null)
            {
                maskedBox.Mask = column.Mask;
            }
        }
    }
    //編集コントロールの型を指定する
    public override Type EditType
    {
        get
        {
            return typeof(DataGridViewMaskedTextBoxEditingControl);
```

```
        }
    }
    //セルの値のデータ型を指定する
    //ここでは、Object 型とする
    //基本クラスと同じなので、オーバーライドの必要なし
    public override Type ValueType
    {
        get
        {
            return typeof(object);
        }
    }
    //新しいレコード行のセルの既定値を指定する
    public override object DefaultNewRowValue
    {
        get
        {
            return base.DefaultNewRowValue;
        }
    }
}
/// <summary>
/// DataGridViewMaskedTextBoxCell でホストされる
/// MaskedTextBox コントロールを表します。
/// </summary>
public class DataGridViewMaskedTextBoxEditingControl :
    MaskedTextBox, IDataGridViewEditingControl
{
    //編集コントロールが表示されている DataGridView
    DataGridView dataGridView;
    //編集コントロールが表示されている行
    int rowIndex;
    //編集コントロールの値とセルの値が違うかどうか
    bool valueChanged;
    //コンストラクタ
    public DataGridViewMaskedTextBoxEditingControl()
    {
        this.TabStop = false;
    }
    #region IDataGridViewEditingControl メンバ
    //編集コントロールで変更されたセルの値
    public object GetEditingControlFormattedValue(
        DataGridViewDataErrorContexts context)
    {
        return this.Text;
    }
    //編集コントロールで変更されたセルの値
    public object EditingControlFormattedValue
    {
        get
```

```
        {
            return this.GetEditingControlFormattedValue(
                DataGridViewDataErrorContexts.Formatting);
        }
        set
        {
            this.Text = (string)value;
        }
    }
    //セルスタイルを編集コントロールに適用する
    //編集コントロールの前景色、背景色、フォントなどをセルスタイルに合わせる
    public void ApplyCellStyleToEditingControl(
        DataGridViewCellStyle dataGridViewCellStyle)
    {
        this.Font = dataGridViewCellStyle.Font;
        this.ForeColor = dataGridViewCellStyle.ForeColor;
        this.BackColor = dataGridViewCellStyle.BackColor;
        switch (dataGridViewCellStyle.Alignment)
        {
            case DataGridViewContentAlignment.BottomCenter:
            case DataGridViewContentAlignment.MiddleCenter:
            case DataGridViewContentAlignment.TopCenter:
                this.TextAlign = HorizontalAlignment.Center;
                break;
            case DataGridViewContentAlignment.BottomRight:
            case DataGridViewContentAlignment.MiddleRight:
            case DataGridViewContentAlignment.TopRight:
                this.TextAlign = HorizontalAlignment.Right;
                break;
            default:
                this.TextAlign = HorizontalAlignment.Left;
                break;
        }
    }
    //編集するセルがある DataGridView
    public DataGridView EditingControlDataGridView
    {
        get
        {
            return this.dataGridView;
        }
        set
        {
            this.dataGridView = value;
        }
    }
    //編集している行のインデックス
    public int EditingControlRowIndex
    {
        get
```

```
        {
            return this.rowIndex;
        }
        set
        {
            this.rowIndex = value;
        }
    }
    //値が変更されたかどうか
    //編集コントロールの値とセルの値が違うかどうか
    public bool EditingControlValueChanged
    {
        get
        {
            return this.valueChanged;
        }
        set
        {
            this.valueChanged = value;
        }
    }

    //指定されたキーを DataGridView が処理するか、編集コントロールが処理するか
    //True を返すと、編集コントロールが処理する
    //dataGridViewWantsInputKey が True の時は、DataGridView が処理できる
    public bool EditingControlWantsInputKey(
        Keys keyData, bool dataGridViewWantsInputKey)
    {
        //Keys.Left、Right、Home、End の時は、True を返す
        //このようにしないと、これらのキーで別のセルにフォーカスが移ってしまう
        switch (keyData & Keys.KeyCode)
        {
            case Keys.Right:
            case Keys.End:
            case Keys.Left:
            case Keys.Home:
                return true;
            default:
                return false;
        }
    }
//マウスカーソルが EditingPanel 上にあるときのカーソルを指定する
    //EditingPanel は編集コントロールをホストするパネルで、
    //編集コントロールがセルより小さいとコントロール以外の部分がパネルとなる
    public Cursor EditingPanelCursor
    {
        get
        {
            return base.Cursor;
        }
```

```
    }
    //コントロールで編集する準備をする
    //テキストを選択状態にしたり、挿入ポインタを末尾にしたりする
    public void PrepareEditingControlForEdit(bool selectAll)
    {
        if (selectAll)
        {
            //選択状態にする
            this.SelectAll();
        }
        else
        {
            //挿入ポインタを末尾にする
            this.SelectionStart = this.TextLength;
        }
    }
    //値が変更した時に、セルの位置を変更するかどうか
    //値が変更された時に編集コントロールの大きさが変更される時は True
    public bool RepositionEditingControlOnValueChange
    {
        get
        {
            return false;
        }
    }
    #endregion
    //値が変更された時
    protected override void OnTextChanged(EventArgs e)
    {
        base.OnTextChanged(e);
        //値が変更されたことを DataGridView に通知する
        this.valueChanged = true;
        this.dataGridView.NotifyCurrentCellDirty(true);
    }
}
```

用法如下

[VB.NET]

```
'DataGridViewMaskedTextBoxColumn を作成
Dim maskedColumn As New DataGridViewMaskedTextBoxColumn()
'データソースの"Column1"をバインドする
maskedColumn.DataPropertyName = "Column1"
'MaskedTextBox の Mask プロパティとなる値を設定する
maskedColumn.Mask = "000"
'DataGridView に列を追加する
DataGridView1.Columns.Add(maskedColumn)
```

[C#]

```
//DataGridViewMaskedTextBoxColumn を作成
DataGridViewMaskedTextBoxColumn maskedColumn =
    new DataGridViewMaskedTextBoxColumn();
//データソースの"Column1"をバインドする
```

```
maskedColumn.DataPropertyName = "Column1";
//MaskedTextBox の Mask プロパティとなる値を設定する
maskedColumn.Mask = "000";
//DataGridView に列を追加する
DataGridView1.Columns.Add(maskedColumn);
```
[VB.NET]
```
''' <summary>
''' セルの値により、適当なアイコンを表示する
''' </summary>
Public Class DataGridViewErrorIconColumn
    Inherits DataGridViewImageColumn
    Public Sub New()
        Me.CellTemplate = New DataGridViewErrorIconCell()
        Me.ValueType = Me.CellTemplate.ValueType
    End Sub
End Class
''' <summary>
''' セルの値により、適当なアイコンを表示する
''' </summary>
Public Class DataGridViewErrorIconCell
    Inherits DataGridViewImageCell
    Public Sub New()
        Me.ValueType = GetType(Integer)
    End Sub
    Protected Overrides Function GetFormattedValue( _
        ByVal value As Object, ByVal rowIndex As Integer, _
        ByRef cellStyle As DataGridViewCellStyle, _
        ByVal valueTypeConverter As System.ComponentModel.TypeConverter, _
        ByVal formattedValueTypeConverter As System.ComponentModel.TypeConverter, _
        ByVal context As DataGridViewDataErrorContexts) As Object
        '値が 0 の時は情報、1 の時は警告、2 の時はエラーアイコンを表示する
        Select Case CInt(value)
            Case 1
                Return SystemIcons.Information
            Case 2
                Return SystemIcons.Warning
            Case 3
                Return SystemIcons.Error
            Case Else
                Return Nothing
        End Select
    End Function
    Public Overrides ReadOnly Property DefaultNewRowValue() As Object
        Get
            Return 0
        End Get
    End Property
End Class
```
[C#]
```
using System;
```

```
using System.ComponentModel;
using System.Windows.Forms;
/// <summary>
/// セルの値により、適当なアイコンを表示する
/// </summary>
public class DataGridViewErrorIconColumn : DataGridViewImageColumn
{
    public DataGridViewErrorIconColumn()
    {
        this.CellTemplate = new DataGridViewErrorIconCell();
        this.ValueType = this.CellTemplate.ValueType;
    }
}
/// <summary>
/// セルの値により、適当なアイコンを表示する
/// </summary>
public class DataGridViewErrorIconCell : DataGridViewImageCell
{
    public DataGridViewErrorIconCell()
    {
        this.ValueType = typeof(int);
    }
    protected override object GetFormattedValue(
        object value, int rowIndex,
        ref DataGridViewCellStyle cellStyle,
        TypeConverter valueTypeConverter,
        TypeConverter formattedValueTypeConverter,
        DataGridViewDataErrorContexts context)
    {
        //値が 0 の時は情報、1 の時は警告、2 の時はエラーアイコンを表示する
        switch ((int)value)
        {
            case 1:
                return SystemIcons.Information;
            case 2:
                return SystemIcons.Warning;
            case 3:
                return SystemIcons.Error;
            default:
                return null;
        }
    }
    public override object DefaultNewRowValue
    {
        get
        {
            return 0;
        }
    }
}
```

用法如下
[VB.NET]
```
'DataGridViewErrorIconColumn を作成
Dim iconColumn As New DataGridViewErrorIconColumn()
'Column1 列（整数型）をバインドする
iconColumn.DataPropertyName = "Column1"
'DataGridView に追加
DataGridView1.Columns.Add(iconColumn)
```
[C#]
```
//DataGridViewErrorIconColumn を作成
DataGridViewErrorIconColumn iconColumn =
    new DataGridViewErrorIconColumn();
//Column1 列（整数型）をバインドする
iconColumn.DataPropertyName = "Column1";
//DataGridView に追加
DataGridView1.Columns.Add(iconColumn);
```
**DataGridView 控件用法合集(十二)**
59. DataGridView 中 Enter 键按下焦点移至旁边的单元格
60. DataGridView 行集合化（Group）

**59. DataGridView 中 Enter 键按下焦点移至旁边的单元格**
[VB.NET]
```
Imports System
Imports System.Windows.Forms
''' <summary>
''' Enter キーが押された時に、Tab キーが押されたのと同じ動作をする
'''  （現在のセルを隣のセルに移動する）DataGridView
''' </summary>
Public Class DataGridViewEx
    Inherits DataGridView
    Protected Overrides Function ProcessDialogKey( _
            ByVal keyData As Keys) As Boolean
        'Enter キーが押された時は、Tab キーが押されたようにする
        If (keyData And Keys.KeyCode) = Keys.Enter Then
            Return Me.ProcessTabKey(keyData)
        End If
        Return MyBase.ProcessDialogKey(keyData)
    End Function
    Protected Overrides Function ProcessDataGridViewKey( _
            ByVal e As KeyEventArgs) As Boolean
        'Enter キーが押された時は、Tab キーが押されたようにする
        If e.KeyCode = Keys.Enter Then
            Return Me.ProcessTabKey(e.KeyCode)
        End If
        Return MyBase.ProcessDataGridViewKey(e)
    End Function
End Class
```
[C#]
```
using System;
using System.Windows.Forms;
```

```
/// <summary>
/// Enterキーが押された時に、Tabキーが押されたのと同じ動作をする
/// （現在のセルを隣のセルに移動する）DataGridView
/// </summary>
public class DataGridViewEx : DataGridView
{
    protected override bool ProcessDialogKey(Keys keyData)
    {
        //Enterキーが押された時は、Tabキーが押されたようにする
        if ((keyData & Keys.KeyCode) == Keys.Enter)
        {
            return this.ProcessTabKey(keyData);
        }
        return base.ProcessDialogKey(keyData);
    }
    protected override bool ProcessDataGridViewKey(KeyEventArgs e)
    {
        //Enterキーが押された時は、Tabキーが押されたようにする
        if (e.KeyCode == Keys.Enter)
        {
            return this.ProcessTabKey(e.KeyCode);
        }
        return base.ProcessDataGridViewKey(e);
    }
}
```

**60. DataGridView 行集合化（Group）**

[VB.NET]

```
'デフォルトのセルスタイル
Private defaultCellStyle As DataGridViewCellStyle
'グループ化された一番上の行のセルスタイル
Private groupCellStyle As DataGridViewCellStyle
'フォームの Load イベントハンドラ
Private Sub Form1_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
    'セルスタイルを設定する
    Me.defaultCellStyle = New DataGridViewCellStyle()
    Me.groupCellStyle = New DataGridViewCellStyle()
    Me.groupCellStyle.ForeColor = Color.White
    Me.groupCellStyle.BackColor = Color.DarkGreen
    Me.groupCellStyle.SelectionBackColor = Color.DarkBlue
End Sub
'CellFormatting イベントハンドラ
Private Sub DataGridView1_CellFormatting(ByVal sender As Object, _
        ByVal e As DataGridViewCellFormattingEventArgs) _
        Handles DataGridView1.CellFormatting
    Dim dgv As DataGridView = CType(sender, DataGridView)
    'セルが 1 列目で、ヘッダーではなく、新しい行でもないとき
    If e.ColumnIndex = 0 AndAlso e.RowIndex >= 0 AndAlso _
        e.RowIndex <> dgv.NewRowIndex Then
        If e.RowIndex = 0 OrElse _
```

```
                Not dgv(e.ColumnIndex, e.RowIndex - 1).Value.Equals(e.Value) Then
                '1 行目か、上のセルと違う値の時は、背景色を変更する
                dgv.Rows(e.RowIndex).DefaultCellStyle = Me.groupCellStyle
            Else
                dgv.Rows(e.RowIndex).DefaultCellStyle = Me.defaultCellStyle
                e.Value = ""
                e.FormattingApplied = True
            End If
        End If
End Sub
```

[C#]

```
//デフォルトのセルスタイル
private DataGridViewCellStyle defaultCellStyle;
//グループ化された一番上の行のセルスタイル
private DataGridViewCellStyle groupCellStyle;
//フォームの Load イベントハンドラ
private void Form1_Load(object sender, EventArgs e)
{
    //セルスタイルを設定する
    this.defaultCellStyle = new DataGridViewCellStyle();
    this.groupCellStyle = new DataGridViewCellStyle();
    this.groupCellStyle.ForeColor = Color.White;
    this.groupCellStyle.BackColor = Color.DarkGreen;
    this.groupCellStyle.SelectionBackColor = Color.DarkBlue;
}
//CellFormatting イベントハンドラ
private void DataGridView1_CellFormatting(object sender,
    DataGridViewCellFormattingEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    //セルが 1 列目で、ヘッダーではなく、新しい行でもないとき
    if (e.ColumnIndex == 0 && e.RowIndex >= 0 &&
        e.RowIndex != dgv.NewRowIndex)
    {
        if (e.RowIndex == 0 ||
            !dgv[e.ColumnIndex, e.RowIndex - 1].Value.Equals(e.Value))
        {
            //1 行目か、上のセルと違う値の時は、背景色を変更する
            dgv.Rows[e.RowIndex].DefaultCellStyle = this.groupCellStyle;
        }
        else
        {
            dgv.Rows[e.RowIndex].DefaultCellStyle = this.defaultCellStyle;
            e.Value = "";
            e.FormattingApplied = true;
        }
    }
}
```

**DataGridView 控件用法合集(十二)**

59. DataGridView 中 Enter 键按下焦点移至旁边的单元格

60. DataGridView 行集合化（Group）

**59. DataGridView 中 Enter 键按下焦点移至旁边的单元格**

[VB.NET]

```
Imports System
Imports System.Windows.Forms
''' <summary>
''' Enter キーが押された時に、Tab キーが押されたのと同じ動作をする
'''  （現在のセルを隣のセルに移動する）DataGridView
''' </summary>
Public Class DataGridViewEx
    Inherits DataGridView
    Protected Overrides Function ProcessDialogKey( _
            ByVal keyData As Keys) As Boolean
        'Enter キーが押された時は、Tab キーが押されたようにする
        If (keyData And Keys.KeyCode) = Keys.Enter Then
            Return Me.ProcessTabKey(keyData)
        End If
        Return MyBase.ProcessDialogKey(keyData)
    End Function
    Protected Overrides Function ProcessDataGridViewKey( _
            ByVal e As KeyEventArgs) As Boolean
        'Enter キーが押された時は、Tab キーが押されたようにする
        If e.KeyCode = Keys.Enter Then
            Return Me.ProcessTabKey(e.KeyCode)
        End If
        Return MyBase.ProcessDataGridViewKey(e)
    End Function
End Class
```

[C#]

```
using System;
using System.Windows.Forms;
/// <summary>
/// Enter キーが押された時に、Tab キーが押されたのと同じ動作をする
///  （現在のセルを隣のセルに移動する）DataGridView
/// </summary>
public class DataGridViewEx : DataGridView
{
    protected override bool ProcessDialogKey(Keys keyData)
    {
        //Enter キーが押された時は、Tab キーが押されたようにする
        if ((keyData & Keys.KeyCode) == Keys.Enter)
        {
            return this.ProcessTabKey(keyData);
        }
        return base.ProcessDialogKey(keyData);
    }
    protected override bool ProcessDataGridViewKey(KeyEventArgs e)
    {
        //Enter キーが押された時は、Tab キーが押されたようにする
```

```
        if (e.KeyCode == Keys.Enter)
        {
            return this.ProcessTabKey(e.KeyCode);
        }
        return base.ProcessDataGridViewKey(e);
    }
}
```

**60. DataGridView 行集合化（Group）**

[VB.NET]

```
'デフォルトのセルスタイル
Private defaultCellStyle As DataGridViewCellStyle
'グループ化された一番上の行のセルスタイル
Private groupCellStyle As DataGridViewCellStyle
'フォームの Load イベントハンドラ
Private Sub Form1_Load(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles MyBase.Load
    'セルスタイルを設定する
    Me.defaultCellStyle = New DataGridViewCellStyle()
    Me.groupCellStyle = New DataGridViewCellStyle()
    Me.groupCellStyle.ForeColor = Color.White
    Me.groupCellStyle.BackColor = Color.DarkGreen
    Me.groupCellStyle.SelectionBackColor = Color.DarkBlue
End Sub
'CellFormatting イベントハンドラ
Private Sub DataGridView1_CellFormatting(ByVal sender As Object, _
        ByVal e As DataGridViewCellFormattingEventArgs) _
        Handles DataGridView1.CellFormatting
    Dim dgv As DataGridView = CType(sender, DataGridView)
    'セルが 1 列目で、ヘッダーではなく、新しい行でもないとき
    If e.ColumnIndex = 0 AndAlso e.RowIndex >= 0 AndAlso _
        e.RowIndex <> dgv.NewRowIndex Then
        If e.RowIndex = 0 OrElse _
            Not dgv(e.ColumnIndex, e.RowIndex - 1).Value.Equals(e.Value) Then
            '1 行目か、上のセルと違う値の時は、背景色を変更する
            dgv.Rows(e.RowIndex).DefaultCellStyle = Me.groupCellStyle
        Else
            dgv.Rows(e.RowIndex).DefaultCellStyle = Me.defaultCellStyle
            e.Value = ""
            e.FormattingApplied = True
        End If
    End If
End Sub
```

[C#]

```
//デフォルトのセルスタイル
private DataGridViewCellStyle defaultCellStyle;
//グループ化された一番上の行のセルスタイル
private DataGridViewCellStyle groupCellStyle;
//フォームの Load イベントハンドラ
private void Form1_Load(object sender, EventArgs e)
{
```

```
    //セルスタイルを設定する
    this.defaultCellStyle = new DataGridViewCellStyle();
    this.groupCellStyle = new DataGridViewCellStyle();
    this.groupCellStyle.ForeColor = Color.White;
    this.groupCellStyle.BackColor = Color.DarkGreen;
    this.groupCellStyle.SelectionBackColor = Color.DarkBlue;
}
//CellFormatting イベントハンドラ
private void DataGridView1_CellFormatting(object sender,
    DataGridViewCellFormattingEventArgs e)
{
    DataGridView dgv = (DataGridView)sender;
    //セルが 1 列目で、ヘッダーではなく、新しい行でもないとき
    if (e.ColumnIndex == 0 && e.RowIndex >= 0 &&
        e.RowIndex != dgv.NewRowIndex)
    {
        if (e.RowIndex == 0 ||
            !dgv[e.ColumnIndex, e.RowIndex - 1].Value.Equals(e.Value))
        {
            //1 行目か、上のセルと違う値の時は、背景色を変更する
            dgv.Rows[e.RowIndex].DefaultCellStyle = this.groupCellStyle;
        }
        else
        {
            dgv.Rows[e.RowIndex].DefaultCellStyle = this.defaultCellStyle;
            e.Value = "";
            e.FormattingApplied = true;
        }
    }
}
```