

EjunGrid 用户手册

版本:V4.0

版本号	修改日期	修改人	说明
1.0	2010-6-17	xx	创建文档

目录

1 概述.....	6
1.1 关于 EjunGrid.....	6
1.2 主要特点.....	6
2 快速入门.....	8
2.1.1 常见问题解答 (FQA).....	8
3 使用指南.....	9
3.1 设置外观.....	9
3.1.1 整体外观.....	9
3.1.2 行列外观.....	13
3.1.3 表头外观.....	15
3.1.4 单元格外观.....	17
3.1.5 滚动条.....	18
3.2 操作表格.....	18
3.2.1 行/ 列操作.....	18

3.2.2 单元格操作.....	22
3.2.3 选择框操作.....	25
3.2.4 冻结行列.....	31
3.2.5 表格树操作.....	31
3.2.6 工作表操作.....	32
3.2.7 文件操作.....	34
3.3 单元格格式.....	37
3.3.1 数字格式.....	37
3.3.2 对齐方式.....	40
3.3.3 字体.....	41
3.3.4 边框.....	42
3.3.5 填充颜色.....	45
3.3.6 保护.....	45
3.3.7 其他.....	45
3.4 单元格类型.....	47
3.4.1 常规.....	47
3.4.2 文本框型.....	47
3.4.3 复选框型.....	48
3.4.4 单选框型.....	48
3.4.5 组合下拉框型.....	49
3.4.6 下拉列表框型.....	49
3.4.7 数字旋钮型.....	50
3.4.8 按钮型.....	51
3.4.9 图表型.....	52

3.4.10 图片型.....	52
3.4.11 日期选择型.....	53
3.4.12 财务金额型.....	54
3.5 公式计算.....	54
3.5.1 单元格公式.....	55
3.5.2 列公式.....	55
3.5.3 内置函数.....	56
3.5.4 用户自定义函数.....	59
3.6 撤销与重做.....	59
3.6.1 操作历史记录.....	59
3.7 打印.....	62
3.7.1 打印.....	63
3.7.2 打印时自动调整行高.....	65
3.8 树形表格.....	68
3.9 工作簿控件.....	68
4 EJunGrid 参考.....	69
4.1 对象模型表.....	69
4.2 TZcCustomGrid.....	69
4.3 TZcCustomDataGrid.....	97
4.3.1 Properties.....	98
4.3.2 Functions.....	105
4.3.3 Events.....	110
4.4 TEJunDataGrid.....	110
4.4.1 Properties.....	110

4.4.2 Functions.....	111
4.4.3 Events.....	111
4.5 TEjunTreeGrid.....	111
4.6 TEjunSheetControl.....	111
4.7 TEjunDBGrid.....	111
4.8 TEjunDBSheet.....	111
4.9 TEjunGridPrinter.....	111
4.10 TEjunGridCommandHistory.....	111
4.11 TEjunGridPreviewPanel.....	111
4.12 表格枚举列表.....	111
4.12.1 TZcCellType.....	111
4.12.2 TZcColumnSortType.....	112
4.12.3 TZcGridRowOption.....	112
4.12.4 TSelectionMode.....	113
4.12.5 TZcGridHorzAlign.....	113
4.12.6 TZcGridVertAlign.....	114
4.12.7 TGridDir.....	115
4.12.8 TZcGridBorderSide.....	115
4.12.9 TZcCellOption.....	116
4.12.10 TZcStyleUsedAttrib.....	117
4.12.11 TGridState.....	118
4.13 Objects.....	118
4.13.1 TZcCell.....	118
4.13.2 TZcDataGridColumn.....	143

4.13.3	TZcGridRow.....	146
4.13.4	TZcGridRange.....	151
4.13.5	IZcCellRange.....	162
4.13.6	TZcGridSelection.....	163
4.13.7	TZcGridStyle.....	185
4.13.8	TZcTree.....	195
4.13.9	TZcTreeNode.....	196
4.13.10	TZcGridExcelRW.....	197
5	应用文章.....	198

1 概述

1.1 关于 EjunGrid

EjunGrid 是一款类似 Excel 风格的高品质表格控件,我们设计的目标是让广大软件开发者能够轻松快速开发出专业、高水准的软件产品,使 您的软件具备方便快捷的录入界面、清晰漂亮的数据显示界面、完美强大的打印预览功能、可以让您的用户在打印预览时实时方便的调整页面布局,所见即所得,操 作方式与 Excel 完全兼容,输出的报表精美典雅。

众多优质的功能, 让 EjunGrid 跻身于高端表格控件之列, EjunGrid 是纯 Delphi 表格控件, 同时提供 Web 插件版, 用于开发 Web 报表, 我们设计的目标是让广大软件开发者能够轻松快速开发出专业、高水准的软件产品,, 使您 的软件具备方便快捷的录入界面、清晰漂亮的数据显示界面、完美强大的打印预览功能、可以让您的用户在打印预览时实时方便的调整页面布局,所见即所得,操作 方式与 Excel 完全兼容,输出的报表精美典雅。

你还在为10多个 OCX, DLL 而烦恼吗, 那不是 Delphi 所需要的, Ejun 是100%纯 VCL 控件, 不依赖任何第三方控件和 DLL;

Ejun 是2000多位正版客户的选择, 为您提供了信心保证, 我们还为您提供最新的金丝雀版, 请赶快[下载试用](#)吧。

1.2 主要特点

- 强大的单元格合并功能, 客户区、 表头、列头, 都可以随意合并单元格, 能够制作出任意复杂的表格
- 支持行锁定和列锁定, 拖动滚动条时固定行和固定列不随滚动条滚动而改变位置, 适合显示商品名称、编号等固定信息
- 单元格可以插入任何类型的对象
- 兼容 Excel 操作方式, 使您的软件用户能够轻易上手, 减少培训费用
- 支持 Excel 方式的拖动选择, 拖动复制, 行选, 列选
- 能够和 Excel 一样, 拖动选择框右下角的小方框进行行填充和列填充
- 可以和 Excel 之间相互复制粘贴内容
- 可以灵活地控制选择框的运行轨迹, 例如用户在第一列输入完数据后按回车键, 您可以根据需要让选

择框掠过第二列直接跳转到第三列，或者您需要的任何地方

- 丰富鼠标事件和键盘事件，完善的开发接口
- 很好地支持鼠标滚轮
- 随意设置单元格字体颜色，可以通过设计器设计，也可以通过属性设置，还可以通过事件根据不同的运行情况动态设定
- 可以继承 TZjCell 实现更多风格的单元格，支持多达9种的对齐方式
- 树表结合，将树和表格完美地集成在一起，可以很方便地实现树节点的插入、删除、升级、降级、上移、下移
- 丰富的事件，可以方便地扩展表格功能，通过事件代码可以向单元格中放入任何类型的控件
- 支持统计曲线
- 可以插入图片
- 如果需要特殊类型的单元格，提供定制服务
- **强大的打印支持**
- 可以指定自定义纸张大小
- 根据纸张大小自动分页
- 可根据页面宽度按比例自动拉伸列宽
- 可根据页面高度自动插入空白行充满整个页面高度
- 可随意选择打印范围，打印表格中指定的区域
- 更强大的是：可以指定表格中的某些行和列为标题行和标题列，打印时每页都出现。这样可以轻松打印出每页都需要的表头或列头
- 可以在打印预览时拖动鼠标调整页边距、行高、列宽，调整时以虚线提示调整的位置，所有操作完全适应 Excel
- 可以选择预览调整的结果是否实时同步到表格中
- 可以设置多行页眉页脚，自动选择打印页码、总页数、日期、事件等等，可以设定字体颜色
- 可以设置多行标题，实现主大标题、副标题等效果

2 快速入门

怎样入门?

- 1.查看我们的[常见问题解答\(FQA\)](#)
- 2.浏览我们的[使用指南](#)
- 3.进行论坛进行学习
- 4.进<<乐图 EjunGrid 开发者总部>>QQ 群: 5408540

2.1.1 常见问题解答(FQA)

2.1.1.1 为什么会出现加载数据很慢的情况

在给单元格赋值的时候，会触发 OnCellValueChanged 事件，以及刷新单元格，在大量数据的加载时，加载的时间就会重复的调用这些动作,导致加载速度很慢. 表格提供了一对方法（BeginUpdate, EndUpdate），进行加载数据控制。开发者应把加载数据的代码放在这对方法中。

使用了加载控制方法，给单元格赋值的时候将不会触发相应的事件，也不会刷新，表格会在 EndUpdate 时进行控制刷新

注意:BeginUpdate 和 EndUpdate 需要成对使用，可以嵌套使用。

代码示例

```
[Delphi]
EjunGrid.BeginUpdate;
try
    // 加载数据代码区
Finally
    EjunGrid.EndUpdate;
end;
```

3 使用指南

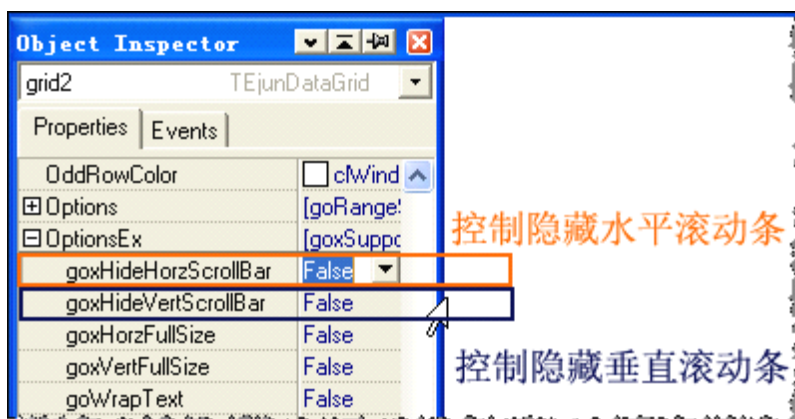
3.1 设置外观

3.1.1 整体外观

3.1.1.1 隐藏滚动轴

EjunGrid 通过设置表格的 OptionsEx 中的 goxHideHorzScrollBar 项来设置隐藏水平滚动轴，goxHideVertScrollBar 项来设置隐藏垂直滚动轴。

表格默认是不隐藏水平和垂直滚动轴的



代码示例

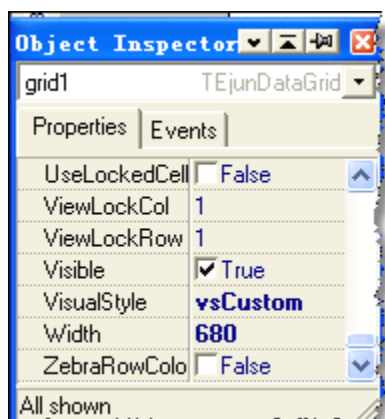
[Delphi]

```
EjunGrid.OptionsEx := grid1.OptionsEx - [goxHideHorzScrollBar]; // 设置隐藏水平滚动条
EjunGrid.OptionsEx := grid1.OptionsEx - [goxHideVertScrollBar]; // 设置隐藏垂直滚动条
```

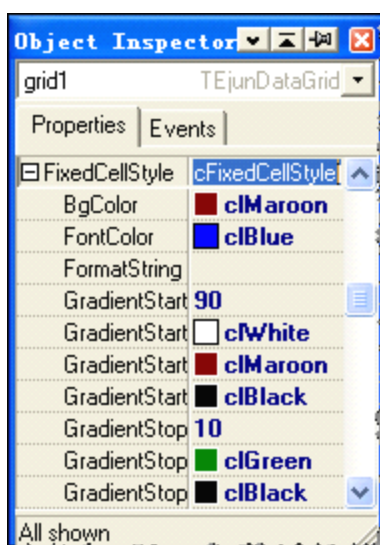
3.1.1.2 设置表格控件外观

表格的显示风格

EjunGrid 提供了多种显示风格, 通过设置 VisualStyle 属性, 可以设置表格的显示风格,

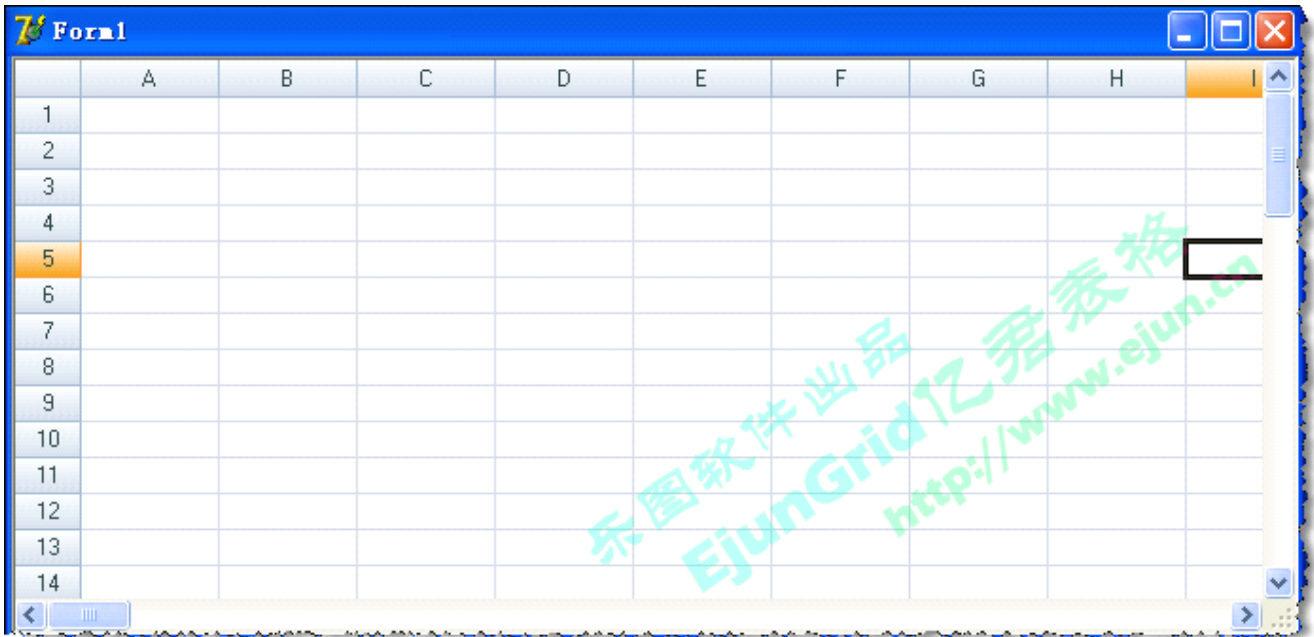


设置 VisualStyle 属性为 vsCustom 时，表格显示为自定义显示样式，再通过配置

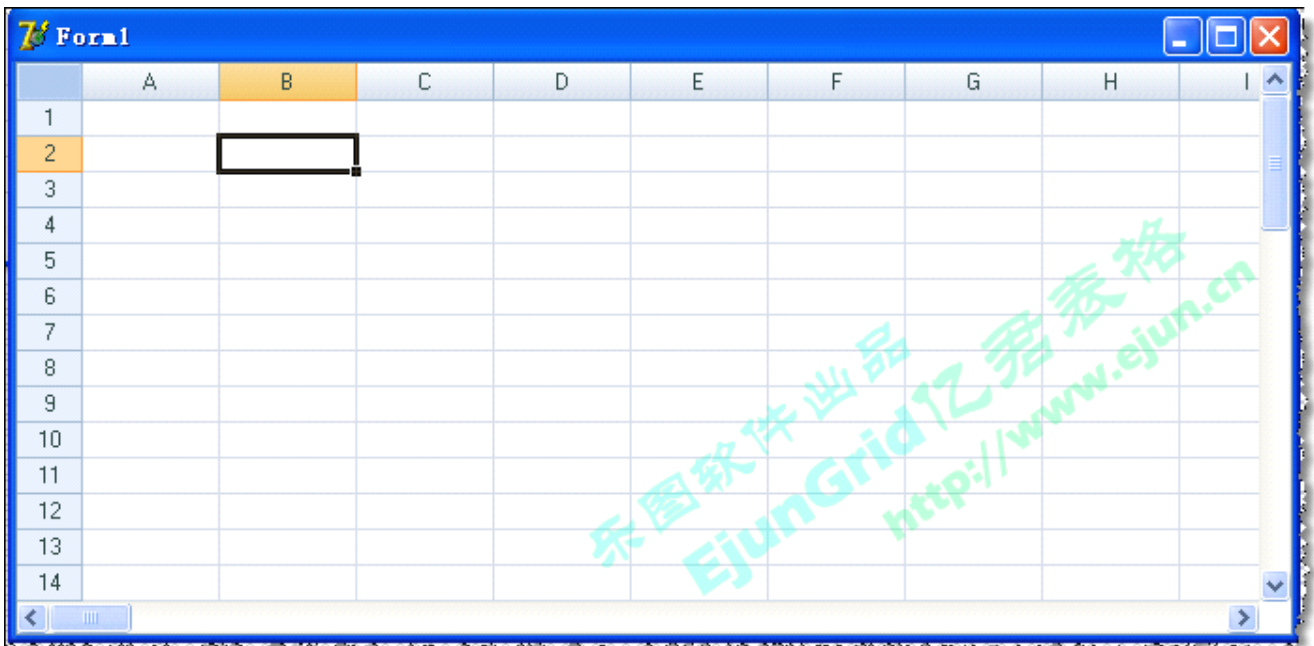


可以设置自己想要的风格.

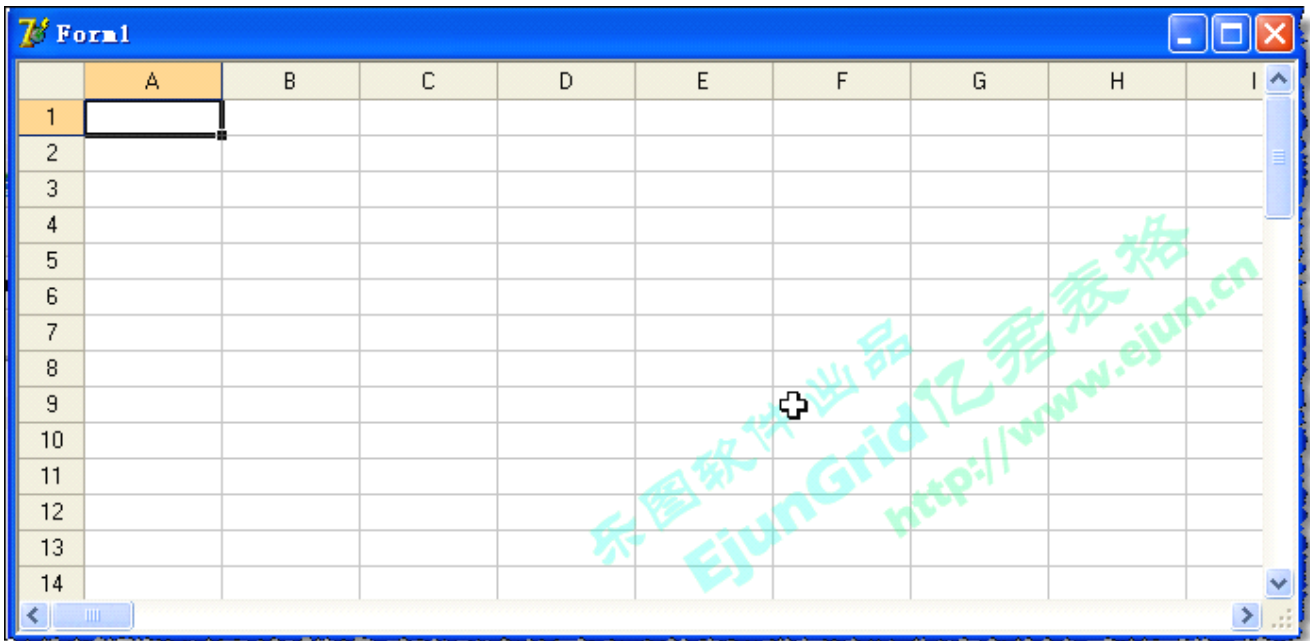
以下是表格显示风格.



(Excel2003 效果)



(Excel2007 效果)



(经典效果)



(自定义效果)

表格的背景及边框

表格通过 `BorderStyle` 属性设置有无边框。

表格的背景可以设置颜色，也可以设置背景图片

代码示例

```
[Delphi]
EjunGrid.BorderStyle := bsSingle;    // 设置有边框
EjunGrid.BorderStyle := bsNone;      // 设置无边框

EjunGrid.BgColor := ClRed;           // 设置背景为红色
EjunGrid.BackgroundImage.LoadFromFile('Bg.jpg'); // 设置背景图片为 Bg.jpg
```

3.1.2 行列外观

表格行列分为固定行和非固定行，固定列和非固定列。

固定行列是永远显示在表格上的，非固定行列则会随表格滚动而滚动。



3.1.2.1 隐藏行列

设置行列隐藏，只需要设置对应的行列对象的 Visible 属性。用户不能通过拖拽调整行高列宽。

调整行高或者列宽为 0，也可以达到隐藏行列的作用。但是设置 0 行高或者列宽，用户可以通过[拖拽调整行高列宽](#)。

注意：行列索引是基于 0 的，如果要隐藏固定行列，只需要隐藏固定行列的索引号的行列就行。

代码示例

```
[Delphi]
EjunGrid.Columns[1].Visible := False; // 隐藏第二列
EjunGrid.Columns[1].Width := 0;      // 设置第二列宽为 0，达到隐藏
EjunGrid.Rows[2].Visible := False;  // 隐藏第三行
EjunGrid.Rows[2].Height := 0;       // 设置第三行高为 0 达到隐藏
```

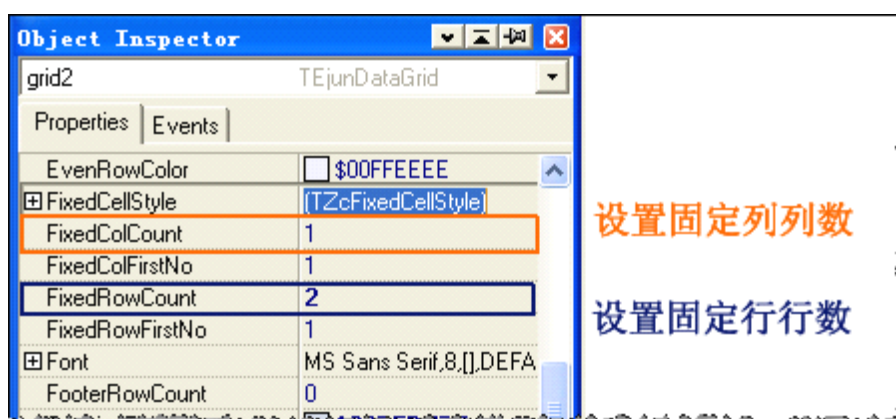
注意：表格行列索引都是基于 0 的

3.1.2.2 设置固定行列

表格通过 FixedRowCount 属性设置固定行数

表格通过 FixedColCount 属性设置固定列数

注意：固定行列数，不能设置为 0，如果想隐藏固定行列，请使用正常的隐藏固定行列方法



代码示例

[Delphi]

```
EjunGrid.FixedRowCount := 2;           // 设置固定行为 2 行
EjunGrid.FixedColCount := 2;          // 设置没有固定列

EjunGrid.Rows[0].Visible := False;    // 隐藏固定行 1
EjunGrid.Columns[0].Visible := False; // 隐藏固定列 1
```

3.1.2.3 设置不滚动行列

设置不滚动行/列可以方便查看。EjunGrid 通过冻结行列来控制不滚动行列。请参见[冻结行列](#)

3.1.2.4 设置行高/列宽

默认行高列宽

表格通过 DefaultFixedRowHeight 属性设置默认固定行高

表格通过 DefaultRowHeight 属性设置默认非固定行高

表格通过 DefaultFixedColWidth 属性设置固定列宽

表格通过 DefaultColWidth 属性设置默认非固定列宽



自定义行高列宽

用户可以通过[拖拽调整行高列宽](#)。也可以通过设置指定行的行高，指定列的列宽

代码示例

[Delphi]

```
EjunGrid.Columns[1].Width := 0;           // 设置第二列宽为
EjunGrid.Rows[2].Height := 0;           // 设置第三行高为
```

注意：表格行列索引都是基于 0 的

3.1.2.5 设置分级显示

3.1.3 表头外观

3.1.3.1 设置行标/列标

默认情况下行标是从 1 开始的数字，列标是从 A 开始的字母。表现方式和 Excel 相同。

	A	B	C	D
1				
2				
3				
4				

如果想控制行标显示的开始数字可以通过设置 FixedRowFirstNo, 设置 FixedColFirstNo 控制列标显示的开始字母

	B	C	D	E
2				
3				
4				
5				

Object Inspector

grid1 TEjunDataGrid

Properties | Events

EvenRowColor \$00FFEEEE

FixedCellStyle (TZcFixedCellStyle)

FixedColCount 1

FixedColFirstNo 2

FixedRowCount 1

FixedRowFirstNo 2

控制列标起始字母, 1为A, 2为B...

控制行标起始数字

代码示例

[Delphi]

```
EjunGrid.FixedColFirstNo := 2; // 设置表格列表从 B 开始
EjunGrid.FixedRowFirstNo := 3; // 设置表格行标从 3 开始
```

用户可以自定义行列, 列标文字, 效果如下图

	列一	列二	列三	E
行一				
3				
4				
5				

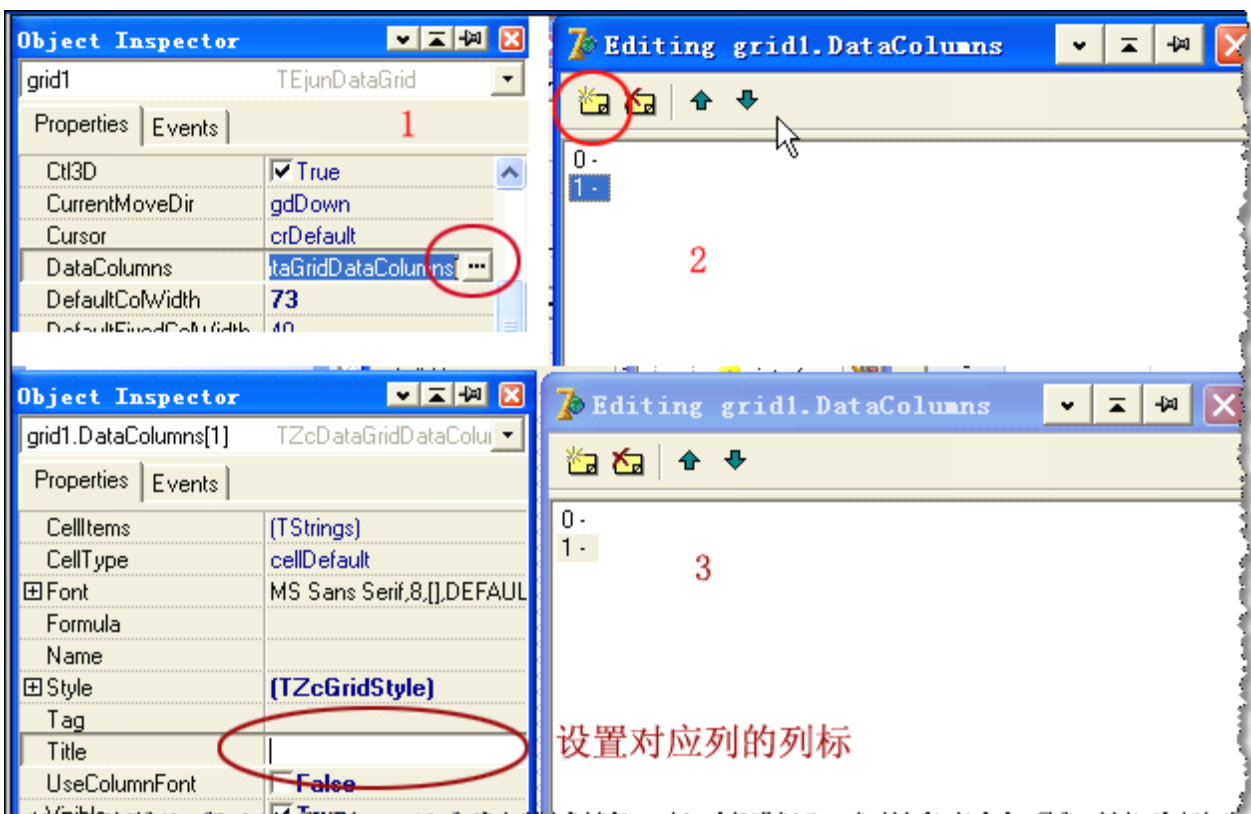
自定义行标,列标文字可以通过设置对应的单元格文字实现, 行标, 列标对应的也是一个一个的单元格

代码示例

[Delphi]

```
EjunGrid.Cells[1, 0].AsString := '列一'; // 设置第一列的列标文字
EjunGrid.Cells[0, 1].AsString := '行一'; // 设置第一行的行标文字
```

EjunGrid 表格还可以通过修改数据列对象(DataColumns 属性)的 Item 项的标题(Title)来自定义列标文字。在这里我们还可以设置列的样式, 宽度, 可视化等



关于隐藏行标列标请参见[设置固定行列](#)

3.1.4 单元格外观

3.1.4.1 设置单元格颜色

单元格的边框请参见[单元格格式](#)的填充色

3.1.4.2 设置单元格对齐

单元格的对齐请参见[单元格格式](#)的[单元格对齐方式](#)

3.1.4.3 设置单元格边框

单元格的边框请参见[单元格格式](#)的[边框](#)

3.1.5 滚动条

3.2 操作表格

3.2.1 行/列操作

用户可以设置表格的行列,也可以对表格的行列进行添加删除.可以进行移动行列操作,也可以设置行高,列宽.

3.2.1.1 增加行/列

删除行用 InsertRow 方法

删除列用 InsertCol 方法

代码示例

[Delphi]

```
// 在第二行的位置新插入 5 行  
EjunDataGrid1.InsertRow(2, 5);  
// 在第二列的位置新插入 5 列  
EjunDataGrid1.InsertCol(2, 5);
```

3.2.1.2 删除行/列

删除行用 DeleteRow 方法

删除列用 DeleteCol 方法

代码示例

[Delphi]

```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
    // 删除选择框框选住的几行  
    // Selection 属性的 Top 属性返回选择框的顶行序号, RowCount 属性返回选择框选择的行数。
```

```
EjunDataGrid1.DeleteRow(EjunDataGrid1.Selection.Top, EjunDataGrid1.Selection.RowCount);
end;
d
```

3.2.1.3 设置行/列数

- RowCount 属性，设置和访问表格总行数
- ColCount 属性，设置和访问表格总列数

这里的总行数包含固定行的行数 FixedRowCount 的，忆君表格行序号从0开始计算，固定行和客户区域行的序号是连续排下来的；Ejun 表格中固定行必须至少有一行，所以序号为0的行肯定是固定行，固定行的行数可以通过 FixedRowCount 属性可以进行设置和访问，可以推得，最后一行固定行的序号是 FixedRowCount-1，客户区域第一行的序号是 FixedRowCount，记住这些规律，对于我们编写程序时特别有帮助。

列也是同样的情况，FixedColCount 属性可以设置和访问固定列的数量，固定列的数量也必须大于0，第一个固定列的序号是0，最后一个固定列的序号是 FixedColCount-1，客户区域第一列的序号是 FixedColCount。

用户可以通过表格对象的属性编辑器进行设置行列数：

设置行数 设置列

下面的例子演示了，计算客户区域第一行到最后一行，第一列所有单元格值的和。因为 Ejun 表格行序号从0开始，所以最后一行的序号是 RowCount - 1；

代码示例

[Delphi]

```
procedure TForm1.Button1Click(Sender: TObject);
var
  iRow: Integer;
  dTotal: Double;
begin
  dTotal := 0;
  // 遍历客户区域第一列所有单元格，并统计合计
  for iRow := EjunDataGrid1.FixedRowCount to EjunDataGrid1.RowCount - 1 do
    dTotal := dTotal + EjunDataGrid1.Cells[1, iRow].AsFloat;
  ShowMessage('第一列合计是:' + FloatToStr(dTotal));
end;
```

3.2.1.4 移动行/列



- 移动行的方法是 MoveRow(ARow, ANewIndex)，第一个参数是要移动的行的序号，第二个参数是目标位置序号
- 移动列的方法是 MoveCol(ACol, ANewIndex)，第一个参数是要移动的列的序号，第二个参数是目标位置序号

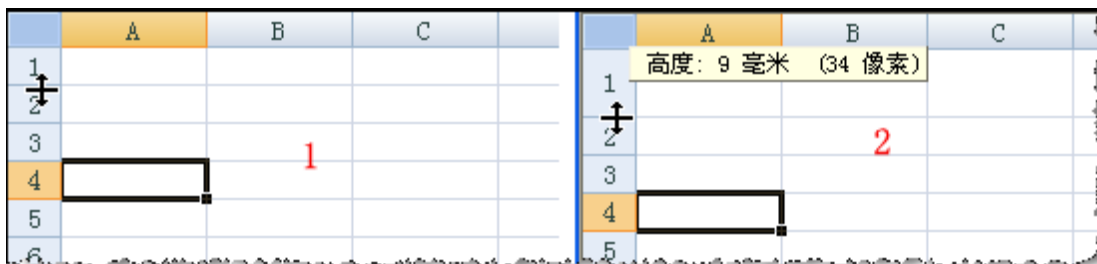
```
// 把第五行移到第一行的位置，第一行到第四行都被向下挪动一个位置
EjunDataGrid1.MoveRow(5, 1);
```

也可以调用行对象的 MoveTo(ANewIndex) 方法来移动行，该方法内部调用的是表格对象的 MoveRow 方法。

```
EjunDataGrid1.Rows[5].MoveTo(1); // 将第五行移动到第一行的位置
```

3.2.1.5 拖拽调整行高/列宽

当鼠标移动到固定行或列的边线时时，鼠标会自动变成(行) , (列) , 用户可以单击鼠标左键不放，拖拽调整行高列宽



拖拽调整行高



拖拽调整列宽

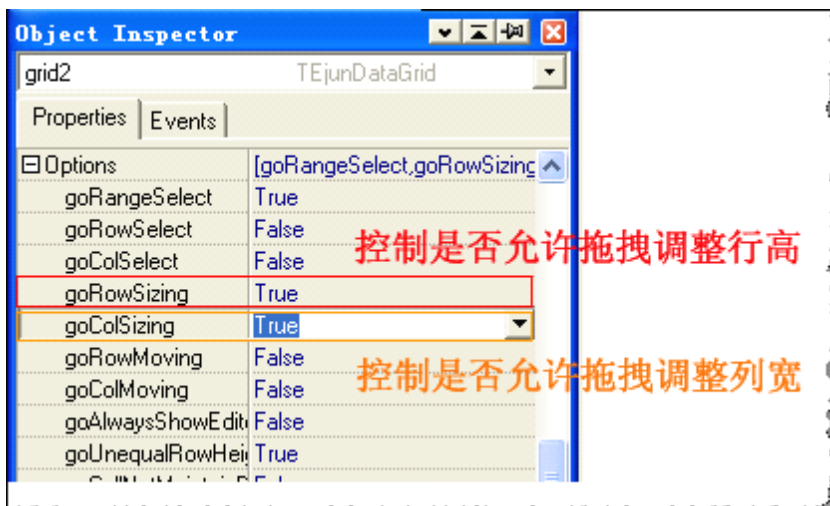
鼠标移动到固定行的边线时，鼠标左键双击表格，那么选择的行将自动变化至默认行高

注意：

表格默认是可以拖拽调整行高列宽。

要控制拖拽调整行高，请确定表格 Options 属性中 goRowSizing 为 True，

要控制拖拽调整列宽，请确定表格 Options 属性中 goColSizing 为 True，



代码示例

[Delphi]

```
// 默认表格是允许拖拽调整行高列宽，我们代码禁止此功能
EjunGrid.Options := grid1.Options - [goColSizing]; // 禁止拽调整行高
EjunGrid.Options := grid1.Options - [goRowSizing]; // 禁止拽调整列宽
```

3.2.1.6 自动调整行高/列宽

通过设置表格对象的 AutoColWidth 来自动调整列宽，AutoRowHeight 来自动调整行高

代码示例

[Delphi]

```
EjunDataGrid1.AutoColWidth := true; // 设置为自动调整列宽
EjunDataGrid1.AutoRowHeight:= true; // 设置为自动调整行高
```

3.2.2 单元格操作

3.2.2.1 合并单元格

Ejun 表格提供了二种合并方式：

1. 针对用户所选区域合并操作
2. 指定区域的合并操作

- a. 针对用户所选的区域合并操作，Ejun 表格提供了三种合并的方式：

把所选合并成一个单元格

```
EjunDataGrid1.Selection.Merge;
```

横向合并

```
EjunDataGrid1.Selection.HorzMerge;
```

纵向合并

```
EjunDataGrid1.Selection.VertMerge;
```

- b. 针对指定区域合并操作

代码示例

[Delphi]

```
// 合并 1, 2 行 1, 2 列
var
  MergeRect: TRect;
begin
  MergeRect.Left := 1;
  MergeRect.Right := 2;
  MergeRect.Top := 1;
  MergeRect.Bottom := 2;

  EjunDataGrid1.Merge(MergeRect);
  // 上下二个方法的效果是一样的
  EjunDataGrid1.Merge(1, 2, 1, 2);
end;
```

3.2.2.2 单元格名字

EjunGrid 可以设置每个单元格的名称，在查找指定名称的单元格时，只需要访问表格的 NameCells 属性的 CellByName 方法。

如果保存表格文件为 ejun 文件时，设置的单元格名称，会被保存进文件中，这样在做报表模板时，这样就可以让用户在单元格格式变化时，能不修改查找表格代码

代码示例 1: 赋值单元格名称，通过字符串找到单元格

[Delphi]

```
EjunGrid.Cells[2, 3].Name := 'ejun'; // 设置单元格 2X3 的名称为 'ejun'
EjunGrid.NameCells.CellByName('ejun').Value; // 定位到名称为 'ejun' 的单元格，并得到其值
```

代码示例 2: 加载数据集的数据到绑定了数据字段的单元格中

[Delphi]

```
// 加载数据集的数据到绑定了数据字段的单元格中
// 注意数据集数据的创建释放，及访问数据库代码省略
procedure TForm1.LoadDataSetValueByCellName();
var
  i : Integer;
  CellName: string;
  qry1: TADOQuery;
begin
  // ... 创建数据集对象，访问数据库，准备数据库

  // 把数据库内容赋值到设置到对应单元格
  if qry1.RecordCount > 0 then
  begin
    for i := 0 to grid1.NameCells.Count - 1 do
    begin
      // 得到单元格的 Name，然后从数据集中访问此字段名的值
      CellName := grid1.NameCells.Names[i];
      grid1.NameCells.CellFromIndex[i].Value := qry1.FieldByName(CellName).AsVariant;
    end;
  end;

  // .. 释放数据集对象
end;
```

3.2.2.3 设置批注

设置批注，请参阅[单元格批注](#)

3.2.2.4 锁定单元格

通过设置表格对象的 Style 对象的 Locked 属性，即可设置单元格的锁定，锁定后的单元格不能被编辑。

可以通过表格的 OptionsEx 集合属性中 goxCanSelectLocked 项的来设置是否让用户可以选择锁定单元格，对应的的属性还有 OptionEx 集合属性中的 goxCanSelectUnLocked 项来设置是否让用户选中未锁定的单元格

```
EjunDataGrCurCell.Style.Locked := true; // 锁定选中的单元格
```

3.2.2.5 隐藏单元格公式

通过设置表格对象的 Style 对象的 Hidden 属性，即可设置单元格公式的隐藏。

```
EjunDataGrCurCell.Style.Hidden := True; // 设置表格单元选中的单元格公式隐藏
```

3.2.2.6 判断单元格对象是否存在

Ejun 表格中，单元格对象只有需要用到时才会被创建，这样的设计可以有效节省内存和提交创建速度。可以通过 SeekCell 函数判断单元格对象是否存在，不同于 Cells 属性，当通过 Cells 属性访问单元格对象时，如果单元格对象不存在，Ejun 表格就会创建一个，而 SeekCell 方法不会创建单元格对象，他会返回 False。

代码示例

```
[Delphi]

procedure TForm1.Button2Click(Sender: TObject);
var
  CellObj: TZcCell;
begin
  if not EjunDataGrid1.SeekCell(3, 1000, CellObj) then
    ShowMessage('单元格对象还没有被创建');
end;
```

这个特性在我们需要遍历表格大量单元格时特别有用，使用 Cells[ColIndex,RowIndex]属性遍历时，会把所有单元格对象都创建起来，导致速度很慢，相比较而言，SeekCell 方法就非常快了。

3.2.2.7 判断列是否有空单元格

判断一列中是否有单元格为空，注意，单元格为空并不代表单元格对象不存在，可能单元格对象存在，

但对象中没有保存任何值，单元格也为空，通过单元格对象的 IsEmpty() 方法判断是否为空。

代码示例

[Delphi]

```
// 判断指定列中是否有空单元格，如果有就返回 True
function TForm1.HasEmptyCellInColumn(AColIndex: Integer): Boolean;
var
  I: Integer;
  Cell: TZcCell;
  bHasEmptyCell: Boolean;
begin
  bHasEmptyCell := False;

  // 循环遍历所有行
  for I := EjunDataGrid1.FixedRowCount to EjunDataGrid1.RowCount - 1 do
    // 如果单元格对象不存在，或者单元格值为空，说明存在空单元格，就退出循环
    if not EjunDataGrid1.SeekCell(AColIndex, I, Cell) or Cell.IsEmpty then
      begin
        bHasEmptyCell := True;
        Break;
      end;

  Result := bHasEmptyCell;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
  if not HasEmptyCellInColumn(EjunDataGrid1.CurCol) then
    ShowMessage(Format('第%d 列的单元格全部都有数据', [EjunDataGrid1.CurCol]));
end;
```

3.2.3 选择框操作

EjunGrid 加强了客用户选择操作上的支持,用户可以设置选择区域,选择框的移动,限制选择,区域选择,行选,列选,拖动复制/剪切等功能.

3.2.3.1 选定区域

选定指定区域使用方法 SelectRange

代码示例

[Delphi]

```
EjunGrid.SelectRange(1, 1, 3, 3); // 选中 1 行 1 列至 3 行 3 列
EjunGrid.SelectRange(Rect(1, 1, 3, 3)); // 同上
```

3.2.3.2 键盘控制选择框移动方向

选中表格，用户可以通过方向键(上下左右)控制选择框的移动，也可以按[Enter]回车键来移动选择框，默认的是向下移动，如果想控制移动方式，只需设置表格的 CurrentMoveDir 属性。如果要自定义选择框更复杂的运行轨迹，可以响应 OnMoveCurrent 事件。

当用户按回车时，选择框自动向右移动一列，但跳过最后一列，当选择框处于最后一列是，自动跳转到下一行的第一列

代码示例

[Delphi]

```
// 设置按回车键时，选择框左移
EJunGrid.CurrentMoveDir := gdLeft;

// 响应表格的 OnMoveCurrent 事件，控制移动方式
procedure TForm1.OnEJunGridMoveCurrent(Sender: TObject;
  var NewCurrent: TPoint; ADir: TGridDir);
begin
  // 注意要判断 ADir = gdAuto 才处理，因为 gdAuto 才一定是按回车键触发的，
  // 其他几种移动方式有可能是上下左右方向键触发的
  if ADir = gdAuto then
  begin
    case ZJGrid1.CurCol of
      // 如果当前单元格是第一列就直接跳到第三列
      1: NewCurrent := Point(ZJGrid1.CurCol + 2, ZJGrid1.CurRow)
    else
      if ZJGrid1.CurCol < ZJGrid1.ColCount - 1 then
        // 如果不是最后一列就向右移动一列
        NewCurrent := Point(ZJGrid1.CurCol + 1, ZJGrid1.CurRow)
      else
        // 跳到下一行的第一列
        NewCurrent := Point(1, ZJGrid1.CurRow + 1);
    end;
  end;
end;
```

3.2.3.3 限制选择

表格里可以设置用户是否能选中表格

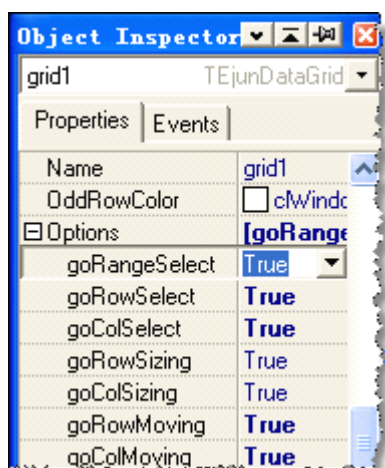
可以通过表格的 OptionsEx 集合属性中 goxCanSelectLocked 项的来设置是否让用户可以选择锁定单元格，也可以通过 OptionEx 集合属性中的 goxCanSelectUnLocked 项来设置是否让用户选中未锁定的单元格

如果要控制某些单元格不能被选中，需要进行对应的编码. 参见[控制某些单元格不能被选中](#)

3.2.3.4 区域选择模式

区域选择模式，也就是让用户可以选择表格的一块区域, 可以是一个单元格，一行单元格，多行，多列.

通过表格对象的属性编辑器可以设置区域选择模式



代码示例

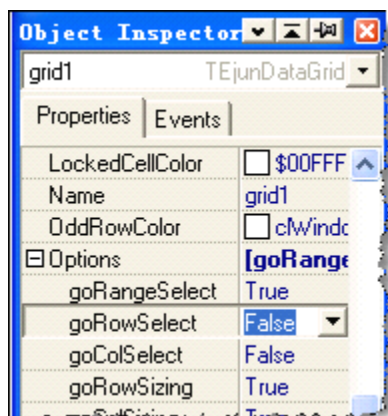
[Delphi]

```
EjunGrid.Options := EjunGrid.Options + [goRangeSelect]; // 设置区域选择  
EjunGrid.Options := EjunGrid.Options - [goRangeSelect]; // 取消区域选择
```

3.2.3.5 行选模式

行选模式，也就是用户选择单元格的时候，会自动的把整行都选中；

通过表格对象的属性编辑器可以设置行选模式



代码示例

[Delphi]

```
EjunGrid.Options := EjunGrid.Options + [goRowSelect]; // 设置行选
EjunGrid.Options := EjunGrid.Options - [goRowSelect]; // 取消行选
```

注意：如果设置了行选的模式同时也设置了列选模式，那表格系统会只支持行选

3.2.3.6 单选模式

关闭区域选择，行选，列选模式就是单选模式

代码示例

[Delphi]

```
EjunGrid.Options := EjunGrid.Options - [goRowSelect, goColSelect, goRangeSelect];
```

3.2.3.7 多选

多选功能是指用户按住[Shift]键，选择多个区域. 如果要支持多选，请打开区域选择模式，关闭行选列选模式. 也就是说多选情况下，就不能设置成为行选或者列选模式.


表格默认为多选.

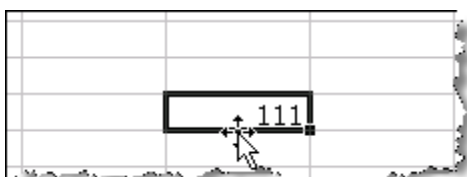
代码示例

[Delphi]

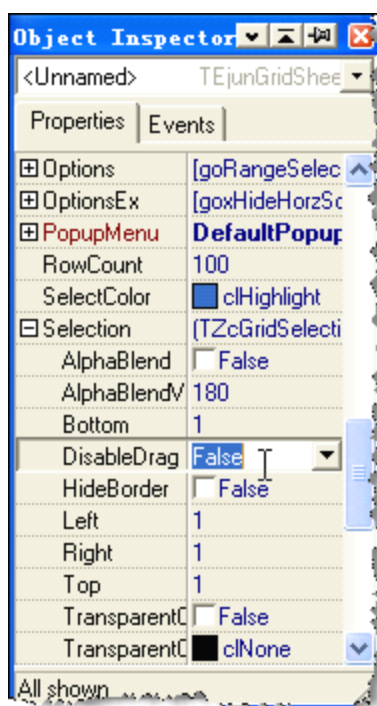
```
EjunGrid.Options := EjunGrid.Options - [goRowSelect, goColSelect];
EjunGrid.Options := EjunGrid.Options + [goRangeSelect]; // 设置多选功能
```

3.2.3.8 拖动复制/剪切


当鼠标移动到表格的边框上时,鼠标图标会变成如 , 这时可以按住鼠标左键不放, 进行剪切选中单元格内容移动到指定位置, 如果按住鼠标左键的同时, 按住 Ctrl 键, 则进行复制粘贴选中单元格操作。

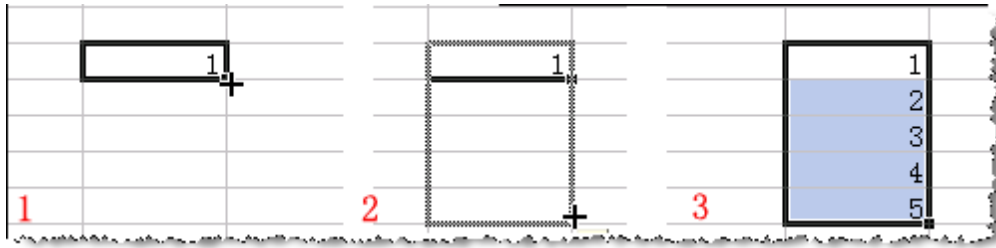


如果想禁止拖动复制/剪切功能, 请设置 Selection 对象的 DisableDrag 属性为 True;



3.2.3.9 自动填充

当鼠标移动到表格的边框右下角时, 鼠标图标会变成 , 这时可以按住鼠标左键不放, 向下(上左右)移动, 表格将自动填充值, 效果如下图:



3.2.3.10 如何控制某些单元格不能选中

如果控制所有被锁定的单元格不能被选中，可以通过表格的 OptionsEx 集合属性中 goxCanSelectLocked 项的设置是否让用户可以选择锁定单元格，对应的属性还有 OptionEx 集合属性中 goxCanSelectUnLocked 项来设置是否让用户选中未锁定的单元格

下图通过表格的属性编辑器进行设置是否能选中锁定的单元格

OptionsEx	[goxHideH...
goxHideHorzScrollBar	True
goxHideVertScrollBar	False
goxHorzFullSize	False
goxVertFullSize	False
goWrapText	False
goIgnoreMouseWheel	False
goTransparent	False
goStringGrid	False
goxSupportFormula	True
goxAutoCalculate	True
goxCanSelectLocked	True
goxCanSelectUnLocked	True

如果要控制某些单元格不能被选中，需要进行对应的编码。

下面演示的是先锁定某些单元格，锁定单元格不可被编辑，然后控制锁定的单元格不能被选中，即选择框不能移动到这些单元格上面。

代码示例

[Delphi]

```

procedure TForm1.FormCreate(Sender: TObject);
Begin
    // 设置第 2 列所有单元格锁定
    EjunDataGrid1.Columns[2].Style.Locked := True;
    // 设置单个单元格锁定
    EjunDataGrid1.Cells[2, 3].Style.Locked := True;
    // 设置第 5 行所有单元格锁定
    EjunDataGrid1.Rows[5].Style.Locked := True;
end;

```

// 当用户移动选择框时，表格控件会触发该事件，询问程序员是否允许移动到指定的单元格上面

```
// 我们可以在这里控制，锁定单元格不允许被选中
procedure TForm1.EjunDataGrid1CurrentChanging(Sender: TObject;
  const ACoord: TPoint; var Allow: Boolean);
begin
  // 在这里我们判断了单元格、行、列的锁定情况，都不锁定的时候才允许选中
  Allow := not (EjunDataGrid1.Items[ACoord].Style.Locked or
    EjunDataGrid1.Columns[ACoord.X].Style.Locked or
    EjunDataGrid1.Rows[ACoord.Y].Style.Locked);
end;
```

3.2.3.11 选择模式

3.2.3.12 选择控制

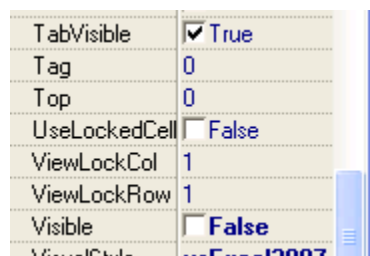
3.2.4 冻结行列

冻结行列也就是设置不滚动行/列，EjunGrid 可以轻松的设置冻结行列功能。

通过表格对象的 ViewLockCol 属性就可以控件冻结的列标题列数, 默认值为 1

通过表格对象的 ViewLockRow 属性就可以控件冻结的行标题行数, 默认值为 1

下图为通过表格对象的属性编辑进行设置



代码示例

[Delphi]

```
EjunDataGrid1.ViewLockCol := 2; // 设置冻结列数为 2
EjunDataGrid1.ViewLockRow := 3; // 设置冻结列数为 2
```

3.2.5 表格树操作

EjunGrid 提供了一款树表完美结合的表格控件-TEjunTreeGrid。表格的节点对应着一个行号，给表格节点的单元格赋值和 EjunDataGrid 是一样的，通过 Cells 属性进行赋值。

	编号	名称	单价	数量	金额	备注
1	笔					
2	钢笔	钢笔	12	2	24	
3	铅笔	铅笔	3	5	72	
4	圆珠笔	圆珠笔	4	66		
5	本子					
6	作业本	作业本	1	2	2	
7	笔记本	笔记本	3	2	6	
8						

3.2.5.1 添加树节点

通过 AddNode 方法添加树节点

代码示例

[Delphi]

```
// 添加一个根节点
EjunTreeGrid.Tree.AddNode(nil);
// 给指定行对应的节点插入子节点
EjunTreeGrid.Tree.AddNode(EjunTreeGrid.RowNode[EjunTreeGrid.CurRow]);
```

3.2.5.2 删除树节点

通过表格树的 Delete 方法删除树节点

代码示例

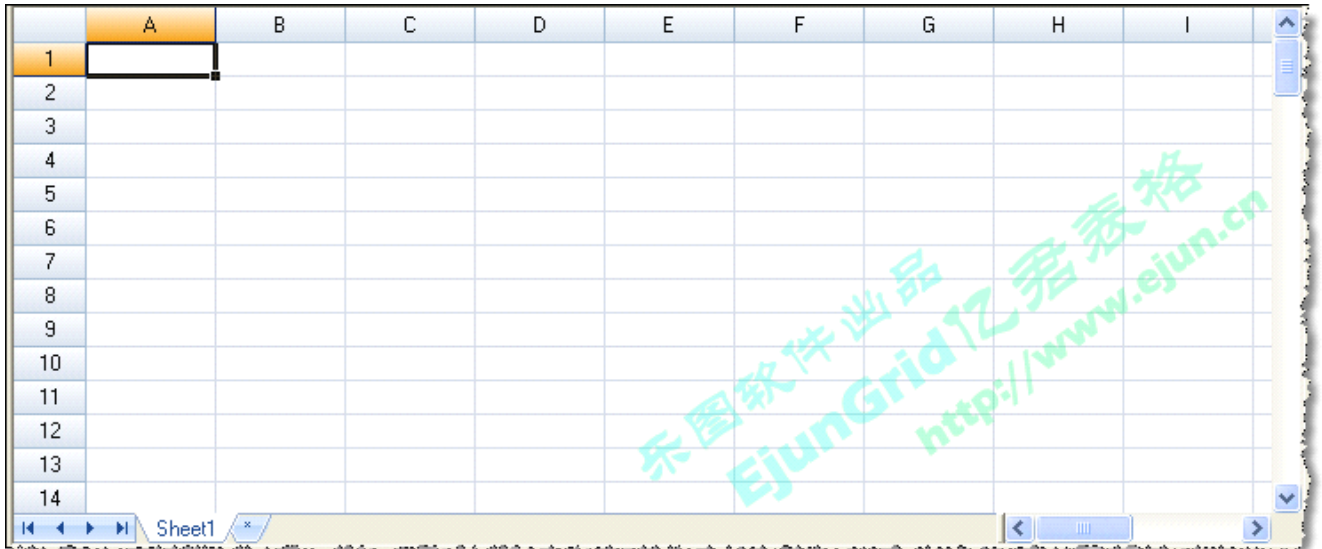
[Delphi]

```
// 添加一个根节点
EjunTreeGrid.Delete(EjunTreeGrid.CurNode.MajorIndex);
```

注意：如果我们要在一个循环中删除树节点时，一定要采用倒序来删除，否则会出错，实际上不光 Tree 对象会这样，其他所有的列表都有这个要求，例如 TList, TListBox。

3.2.6 工作表操作

工作表操作是针对 TEjunSheetControl 控件, TEjunSheetControl 控件默认是没有工作表的, 用户可以自定义添加, 删除工作表。工作表号是从 0 开始计数的。每个工作表都是对应着一个表格。



3.2.6.1 增加工作表

通过单击 TEjunSheetControl 控件界面上的  可以增加一个工作表,也可以通过工作表对象的 NewSheet 方法添加一个新的工作表。

默认创建的工作表名字为 Sheet+工作表页号,如果必想改工作表的显示名称,可以通地双击工作标签头,进行改名输入,或者通过设置工作表的 Caption 属性进行修改

代码示例

[Delphi]

```
// 新增一个工作表,并设置为当前工作表
EjunSheetControl.ActiveSheetIndex := EjunSheetControl.NewSheet;
// 修改活动的工作表的表名
EjunSheetControl.ActiveSheet.Caption := '自定义工作表名';
```

3.2.6.2 删除工作表

如果要删除工作表,用户可以通过调用工作表的 Delete 方法。

代码示例

[Delphi]

```
// 删除当前的工作表
If EjunSheetControl.ActiveSheetIndex <> -1 then
```

```
EjunSheetControl.Delete(EjunSheetControl.ActiveSheetIndex);
```

注意：工作表号是从 0 开始计数的

3.2.7 文件操作

3.2.7.1 导出 Excel 文件

使用亿君表格控件导出 Excel 文件是一件非常简单事情，大部分情况下，只需一行代码

```
EjunDataGrid1.SaveToExcel('d:\myexcel.xls', 'sheet1');
```

第一个参数是带有路径的全文件名，第二个参数是工作表的名称。默认情况下只导出表格控件的客户区域，不包含表头(固定行)和行头(固定列)的内容，如果想导出这两部分的内容，只需在上面方法的后面再添加两个参数，分别控制是否导出表头和行头

```
EjunDataGrid1.SaveToExcel('d:\test.xls', 'sheet1', True, True);
```

因为 Excel 没有多表头的功能，当把 EjunGrid 的表头(固定行)或行头(固定列)导出到 Excel 工作表中时，Excel 工作表会被设置窗口冻结，冻结的行数对应 EjunGrid 中固定行的数量，冻结列对应 EjunGrid 的固定列的数量。

3.2.7.2 导入 Excel 文件

使用 LoadFromExcel 方法，可以方便的把 Excel 文件导入到表格控件中，第一个参数指定 Excel 文件名，第二个参数指定工作表的名称；

代码示例

[Delphi]

```
// 读取 test.xls 文件中的名称叫 Sheet1 的工作表
EjunDataGrid1.LoadFromExcel('d:\test.xls', 'Sheet1');
```

也可以在第二个参数中指定工作表的序号，序号从0开始，第一个工作表序号为0，第二个工作表序号为1，以此类推；

代码示例

[Delphi]

```
// 读取 test.xls 文件中的第一个工作表
```

```
EjunDataGrid1.LoadFromExcel('d:\test.xls', 0);
```

Excel 工作表的内容被读取到 EjunGrid 表格中的位置，默认是从的客户区域的第一行和第一列开始，我们也可以设置从其他位置开始，例如，从 EjunGrid 表格的第3行开始显示 Excel 工作表的内容。

[Delphi]

```
EjunDataGrid1.LoadFromExcel('d:\test.xls', 0, 3, 1);
```

第三个参数指明目标行，设为-1表示从客户区域的第一行开始，第四个参数指明目标列，设为-1表示从客户区域的第一列开始

Excel 读写辅助类:

忆君表格提供了专门的 Excel 读写辅助类，可以实现更加高级的效果；在下面的例子中，Excel 文件中的工作表设置了窗口冻结，我们想在读取数据后，把 Excel 中的窗口冻结转化为忆君表格的固定行列，因为忆君表格的多表头功能非常强大和漂亮。

代码示例

[Delphi]

```
procedure TForm1.btn1Click(Sender: TObject);
var
  ExcelRW: TZcGridExcelRW;
  ExcelLockRow, ExcelLockCol: Integer;
  DestRow, DestCol: Integer;
begin
  // 创建 Excel 读写辅助对象
  ExcelRW := TZcGridExcelRW.Create;
  try
    // 读取 Excel 文件
    ExcelRW.Open(ExtractFilePath(Application.ExeName) + '111.xls');
    DestRow := EjunDataGrid1.FixedRowCount;
    DestCol := EjunDataGrid1.FixedColCount;
    // 判断 Excel 中的 Sheet 是否设置了窗口冻结，如果冻结了，就把表格的固定行列数设为 Excel 冻结的行列数
    if ExcelRW.IsSheetViewLock(0, ExcelLockRow, ExcelLockCol) then
    begin
      if ExcelLockRow > 0 then
      begin
        EjunDataGrid1.FixedRowCount := ExcelLockRow;
        DestRow := 0;
      end;

      if ExcelLockCol > 0 then
      begin
        EjunDataGrid1.FixedColCount := ExcelLockCol;
        DestCol := 0;
      end;
    end;
  end;
```

```

// 从工作表中把内容读到表格控件中
ExcelRW.ReadSheet(EjunDataGrid1, 0, DestRow, DestCol);
finally
    ExcelRW.Free;
end;
end;

```

注 :TZcGridExcelRW 类声明在 ZcDataGrid 单元。

工作簿控件:

忆君表格提供工作簿控件(TEjunSheetControl), 一个工作簿控件包含多个工作表(TEjunGridSheet), 工作簿控件也提供了一个方法, 可以整个工作簿(包含所有的工作表)保存到一个 Excel 文件中,

```
EjunSheetControl1.SaveToExcel('d:\test.xls');
```

从 Excel 文件中将所有工作表读入到工作簿控件中

代码示例

[Delphi]

```

// 如果工作簿控件中已经存在工作表, 需要先全部删除
EjunSheetControl1.DeleteAll;
// 再从 Excel 文件中加载工作表
EjunSheetControl1.LoadFromExcel('d:\test.xls');

```

混合保存:

可以将任意多个表格的内容到一个 Excel 文件中:

代码示例

[Delphi]

```

var
    ExcelRW: TZcGridExcelRW;
begin
    ExcelRW := TZcGridExcelRW.Create;
    try
        ExcelRW.WriteSheet(EjunSheetControl1.Sheets[1], 'sheet1');
        ExcelRW.WriteSheet(EjunDataGrid2, 'sheet2');
        ExcelRW.Save('d:\c.xls');
    finally
        ExcelRW.Free;
    end;
end;

```

提醒: TZcGridExcelRW 类声明在 ZcDataGrid 单元

3.3 单元格格式

单元格格式包括文本对齐方式、背景颜色、边框线性、颜色、文本自动换行、只读、打印等。

EjunGrid 通过 Style 对象用来控制单元格的外观风格。

注意：EjunGrid 采用级联样式表的方式来控制一个具体单元格的格式，**表格对象，列对象，行对象，单元格对象四个层次的格式信息逐层控制**，首先是取单元格的 Style 属性，如果是默认值，则取列对象的 Style 属性设置，如果还是默认值，则取行对象的 Style 设置，最后才取表格对象的 Style 设置。

代码示例

[Delphi]

```
with EjunGrid.Cells[x, y] do
begin
  Style.BgColor := clRed; // 设置背景颜色
  Style.WrapText := True; // 设置文本换行
  Style.HorzAlign := haLeft; //设置单元格显示内容居左对齐
  Style.VertAlign := vaTop; //设置单元格显示内容顶端对齐
  Style.SetBorder([gbLeft,gbTop,gbRight,gbBottom], gbsThin, clBlue); // 设置边框线
  Style.Locked := True; // 设置单元格锁定，不能编辑
  Style.NotPrint := True; // 不打印该单元格，可以通过这种设置实现套打。
  Style.Hidden := False; //隐藏单元格的内容
end;

// 设置列对象的风格
EjunGrid.Columns[1].Style.FormatString := '#,###.00';
// 设置行对象的风格
EjunGrid.Rows[1].Style.FormatString := '#,###.00';
```

注：使用该属性时需要引用 ZcGridStyle 单元

3.3.1 数字格式

EjunGrid 在对单元格内容显示的支持上，做到了几乎与 Excel 一样. 用户通过设置单元格的显示格式字符串，就可以得到特殊的效果。

开发人员可以通过单元格(TZcCell)对象的样式属性(TZcGridStyle)的 formateString 属性来设置显示格式。

代码示例

[Delphi]

```
// 定义有一位小数点的格式
```

```
EjunGrid.CurCell.Style.FormatString := '0.0';
```

3.3.1.1 [Excel 标准字符串代码示例]

常规

代码示例

[Delphi]

```
EjunGrid.CurCell.Style.FormatString := 'G/通用格式'; // 通用格式,默认为字符串
```

数值: 数值格式用于一般数字的表示。货币和会计格式则提供货币值计算的专用格式。

代码示例

[Delphi]

```
EjunGrid.CurCell.Style.FormatString := '0.00_'; // 保留二位小数点的数字
EjunGrid.CurCell.Style.FormatString := '#,##0.00_'; // 带有千分号,并保留二位小数点的数字
```

货币: 货币格式用于表示一般货币数值。会计格式可以对一系列数值进行小数点对齐

代码示例

[Delphi]

```
// 货币数值,二位小数位,如果负数,为红色
EjunGrid.CurCell.Style.FormatString := '¥#,##0.00;[红色]¥-#,##0.00';
```

会计专用: 会计格式可对一系列数值进行货币符号和小数点对齐。

代码示例

[Delphi]

```
// 会计专用格式,人民币,二位小数位。
EjunGrid.CurCell.Style.FormatString := '_ ¥* #,##0.00_ ;_ ¥* -#,##0.00_ ;_ ¥* "-"??_ ;_ @_ ;
```

日期: 日期格式把日期和时间系列数显示为日期值。

代码示例

[Delphi]

```
EjunGrid.CurCell.Style.FormatString := 'yyyy-m-d; // 2010-7-8 这种日期格式
```

时间: 时间格式把日期和时间系列数显示为时间值。

代码示例

[Delphi]

```
EjunGrid.CurCell.Style.FormatString := 'h:mm;@'; // 13:50 时分时间格式。
EjunGrid.CurCell.Style.FormatString := 'h:mm:ss;@'; // 13:50:30 时分秒时间格式
```

百分比： 以百分比格式将单元格中数值乘以 100, 并以百分数形式显示。

代码示例

[Delphi]

```
// 保留二位小数点的百分数, 如 1.00%。
EjunGrid.CurCell.Style.FormatString := '0.00%';
```

分数：

代码示例

[Delphi]

```
// 百分之几, 如 30/100。
EjunGrid.CurCell.Style.FormatString := '# ?/?';
```

科学记数：

代码示例

[Delphi]

```
EjunGrid.CurCell.Style.FormatString := '0.00E+00'; // 保留二位小数的科学记数。
```

文本：文本字符，即时输入的是小数，也会当做字符串看待

代码示例

[Delphi]

```
EjunGrid.CurCell.Style.FormatString := '@';
```

特殊：特殊格式可用于跟踪数据列表及数据库的值。

代码示例

[Delphi]

```
邮政编码:      EjunGrid.CurCell.Style.FormatString := '000000';
中文小写数字:  EjunGrid.CurCell.Style.FormatString := '[DBNum1] [$-804]G/通用格式';
中文大写数字:  EjunGrid.CurCell.Style.FormatString := '[DBNum2] [$-804]G/通用格式';
```

用户可以编排自己想要的格式

3.3.2 对齐方式

单元格的文本的对齐分为水平对齐，垂直对齐。

3.3.2.1 水平对齐

水平对齐有:常规对齐，靠左对齐，居中对齐，靠右对齐，填充对齐。

通过设置单元格，或者选择区的 `HorzAlign` 属性可以设置水平对齐。单元格的默认对齐方式为常规对齐，其效果和靠左对齐是一样的

```
THorzAlign = (haGeneral, haLeft, haCenter, haRight, haFill);
```

代码示例

[Delphi]

```
EjJunGrid.Cells[1, 1].HorzAlign := haLeft; // 设置单元格 1x1 的水平左对齐  
EjJunGrid.Selection.HorzAlign := haLeft; // 设置单元格所选择区域水平左对齐
```

3.3.2.2 垂直对齐

垂直对齐有:常规对齐，靠顶对齐，居中对齐，靠底对齐，

通过设置单元格，或者选择区的 `VertAlign` 属性可以设置垂直对齐。单元格的默认垂直对齐方式为常规对齐，其效果和靠顶对齐是一样的

```
TVertAlign = (vaGeneral, vaTop, vaCenter, vaBottom);
```

代码示例

```
[Delphi]
```

```
EjunGrid.Cells[1, 1].VertAlign := vaTop; // 设置单元格 1x1 的垂直顶对齐
EjunGrid.Selection.VertAlign := vaTop; // 设置单元格所选择区域垂直顶对齐
```

3.3.3 字体

单元格提供了丰富的字体功能，用户可以设置字体的类型，大小，颜色，形状效果。

字体的类型: 设置字体的 Name 属性,其值为系统字体的名字

代码示例

```
[Delphi]
```

```
EjunGrid.Cells[1, 1].Font.Name := '宋体'; // 设置单元格 1x1 为宋体字
EjunGrid.Selection.Font.Name := '宋体'; // 设置单元格所选择区域为宋体字
EjunGrid.Selection.FontName := '宋体'; // 等价上面
```

字体大小: 设置字体的 Size 属性

代码示例

```
[Delphi]
```

```
EjunGrid.Cells[1, 1].Font.Size:= 12; // 设置单元格 1x1 的字体大小为 12
EjunGrid.Selection.Font.Size:= 12; // 设置单元格所选择区域的字体大小为 12
EjunGrid.Selection.FontSize:= 12; // 等价上面
```

字体颜色: 设置字体的 Color 属性

代码示例

```
[Delphi]
```

```
EjunGrid.Cells[1, 1].Font.Color:= clRed; // 设置单元格 1x1 的字体颜色为红色
EjunGrid.Selection.Font.Size:= clRed; // 设置单元格所选择区域的字体颜色为红色
EjunGrid.Selection.FontSize:= clRed; // 等价上面
```

形状: 设置字体的 Style 属性(TFontStyles 集合类型). 字体的形状有,常规, 粗体, 斜体, 下划线.

Delphi 标准定义 Graphics 单元

```
TFontStyle = (fsBold, fsItalic, fsUnderline, fsStrikeOut);
TFontStyles = set of TFontStyle;
```

代码示例

[Delphi]

```

EjunGrid.Cells[1, 1].Font.Stype:= []; // 设置单元格 1x1 的字体形状为常规
EjunGrid.Cells[1, 1].Font.Stype:= [fsBold, fsItalic]; // 设置单元格 1x1 的字体形状
为粗斜体

// 设置单元格所选择区域的字体形状为粗斜体+下划线, 加删除线
EjunGrid.Selection.Font.Stype:= [fsBold, fsItalic, fsUnderline, fsStrikeOut];
// 等价上面
EjunGrid.Selection.FontStype:= [fsBold, fsItalic, fsUnderline, fsStrikeOut];

```

3.3.4 边框

用户可以自定义设置表格边框的线条样式(决定了粗细, 线条样式), 边框颜色, 及边框的框线

用户可以对选区设置表格的框线,方法定义如下:

EjunGrid 代码

ZcGrid 单元

```

IZcCellRange = interface(IInterface)
  ['{8C778A92-9232-45C2-8427-D020F8D053F7}']
  procedure SetBorder(ASides: TZcGridBorderSides; AStyle: TZcGridBorderStyle;
    AColor: TColor; AInnerRow: Boolean = False; AInnerCol: Boolean = False);
end;

```

参数说明:

ASides: TZcGridBorderSides 集合,用来指定设置边框线的位置,如果要指定内十字线,请使用 AInnerRow, AInnerCol 参数

AStyle: 边框线样式, 决定线的样式及粗细

AColor: 边框线颜色

AInnerRow: 当选择多个单元格时, 是否设置所选区内部的行边线框, 如果只有一个单元, 此参数无意义

AInnerCol: 当选择多个单元格时, 是否设置所选区内部的列边线框, 如果只有一个单元, 此参数无意义

3.3.4.1 边框线的位置:

表格提供了多方位的边框线位置, 如下面表格:

3.3.4.3 边框线条样式:

线条样式决定了样条的粗细，及线条的样式,亿君表格通过一个枚举(TZcGridBorderStyle)

EjunGrid 代码

ZcGridStyle 单元

```
TZcGridBorderStyle = (gbsNone, gbsThin, gbsMedium, gbsDashed, gbsDotted,
gbsThick, gbsDouble, gbsHair, gbsMedium dashed, gbsDash dot, gbsMedium dash dot,
gbsDash dot dot, gbsMedium dash dot dot, gbsSlanted dash dot);
```

效果图如下面在表格:

gbsNone	无框线
gbsThin	—————
gbsMedium	—————
gbsDashed	-----
gbsDotted
gbsThick	—————
gbsDouble	—————
gbsHair	—————
gbsMedium_dashed	-----
gbsDash_dot	-----
gbsMedium_dash_dot	-----
gbsDash_dot_dot	-----
gbsMedium_dash_dot_dot	-----
gbsSlanted_dash_dot	-----

代码示例

[Delphi]

```
var
  BorderStyle: TZcGridBorderStyle;
  BorderColor: TColor;
  BorderSides: TZcGridBorderSides;
begin
  BorderStyle := gbsThin;
  BorderColor := clRed;
  BorderSides := [gbLeft, gbTop, gbRight, gbBottom];
  // 设置四边框为红色的细线
  grid1.Selection.SetBorder(BorderSides, BorderStyle, BorderColor);
```

```

BorderColor := clBlue;
BorderStyle := gbsThick;
// 设置四边框加内部行线为兰色粗线
grid1.Selection.SetBorder(BorderSides, BorderStyle, BorderColor, True, False);
// 设置四边框加内部列线为兰色粗线
grid1.Selection.SetBorder(BorderSides, BorderStyle, BorderColor, False, True);
// 设置四边框(包含行列交叉)的为兰色粗线
grid1.Selection.SetBorder(BorderSides, BorderStyle, BorderColor, True, True);
end;

```

3.3.5 填充颜色

单元格的填充色, 设置所选区域的背景颜色.

代码示例

[Delphi]

```

EjunGrid.Selection.BgColor := clRed; // 设置所选区域的背景颜色
EjunGrid.CurCell.BgColor := clRed; // 设置选中的当前表格为红色背景

```

3.3.6 保护

单元格的锁定与隐藏:单元格(TZcCell)对象的 Style 对象的的 Locked 属性设置锁定, Hidden 属性设置隐藏

代码示例

[Delphi]

```

EjunGrid.Cells[1, 1].Style.Locked := true; // 设置锁定
EjunGrid.Cells[1, 1].Style.Hidden:= true; // 设置隐藏

```

3.3.7 其他

3.3.7.1 单元格名称

设置单元格 Name 属性可以设置单元格的名称。

单元格的名字通常用法是，在乐表设计器中定义报表模板，把指定数据库字段到单元格名称中，然后通过加载报表模板，把查询的数据集通过模板的单元格名称(绑定了数据库字段名)，把值显示到单元格中。

代码示例

[Delphi]

```
EjunDataGrid1.CurCell.Name := 'm_Code';           // 设置单元格名称为字段名 m_code
```


3.3.7.2 单元格批注

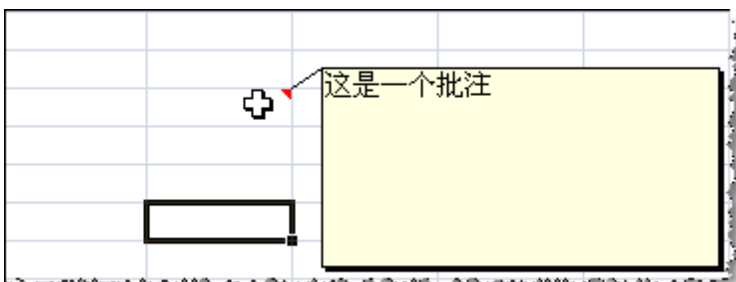
EjunGrid 提供了丰富的批注属性接口，用户可以对指针单元格设置批注(Postil 属性)，提示(Hint 属性)，备注(Note 属性)；

代码示例

[Delphi]

```
EjunDataGrid1.CurCell.Postil := '这是一个批注';           // 设置单元格批注
EjunDataGrid1.CurCell.Hint   := '这是一个浮动提示';       // 设置单元格提示
EjunDataGrCurCell.id1.Note  := '备注';                   // 设置备注
```

设置单元格的批注属性，单元格的右上角会有  红色标记，当鼠标移动到单元格上时，单元格会弹出一个批注层，显示批注内容，效果如下图



设置了单元格的提示属性，当鼠标移到单元格上时，单元格会弹出一个 Delphi 标准的提示框出来，显示提示内容，效果如下图



设置了单元格的备注，在表格外观上没有什么变化，这是一个只供开发人员使用的属性

3.4 单元格类型

Ejun 表格的单元格类型现在有:常规, 文本框型, 复选框, 单选框, 组合下拉框, 下拉列表框, 数字旋钮, 按钮型, 图表型, 图片型, 日期选择型, 财务金额型.

表格给用户二种方式设置单元格的类型:

1. 针对单个单元格: 用户可以通过设置表格的 `CellType` 属性 (`TZcCellType` 枚举性) 来设置对应的单元格类型
2. 针对整列: 设置表格的 `Columns` 对象的 `CellType` 属性, 来设置整列的所有单元格类型

EjunGrid 代码

ZJGrid 单元

```
TZcCellType = (cellDefault, cellTextBox, cellCheckBox, cellRadioBox, cellComboBox,
cellDropDownList, cellNumberSpin, cellChart, cellEllipsisBtn, cellImageBox,
cellCalendar, cellCurrency);
```

3.4.1 常规

cellDefault: 表格单元格默认类型, 和文本框显示类型是同一种效果

代码示例

[Delphi]

```
EjunGrid1.CellType[1, 1] := cellDefault; // 设置单个单元格的类型为默认类型
EjunGrid1.Columns[1].CellType := cellDefault; // 设置第一列的所有单元格的类型为默认类型
```

3.4.2 文本框型

cellTextBox: 文本框显示类型, 表格单元格默认类型, 和常规类型是同一种效果

文本框可以设置自动换行

代码示例

[Delphi]

```
EjunGrild1.CellType[1, 1] := cellTextBox; // 设置单个单元格的类型为复选框
EjunGrild1.Columns[1].CellType := cellTextBox; // 设置第一列的所有单元格的类型为复选框

EjunGrild1.Cells[1, 1].Style.WrapText := true; // 第一行一列的单元格自动换行
```

3.4.3 复选框型

cellCheckBox: 复选框类型，单个单元格内能有一个复选框

复选框类型还增加了 Caption 属性，通过设置 CheckCells 对象的 Caption 属性可以在单元格内的复选框右边显示字符串



代码示例

[Delphi]

```
EjunGrild1.CellType[1, 1] := cellCheckBox; // 设置单个单元格的类型为复选框
EjunGrild1.Columns[1].CellType := cellCheckBox; // 设置第一列的所有单元格的类型为复选框

EjunGrild1.CheckCells[1, 1].Caption := edtCheckBoxTitle.Text;
```

3.4.4 单选框型

cellRadioButton: 单选框类型，与复选框类型不同，单个单元格内能有多个单选按钮

通过表格的 RadioCells 对象的 AddItem 方法设置单元格单选框的按钮项



代码示例

[Delphi]

```
EjunGrild1.CellType[1, 1] := cellRadioButton; // 设置单个单元格的类型为单选框
EjunGrild1.Columns[1].CellType := cellRadioButton; // 设置第一列的所有单元格的类型为单选框
```

```
EjunGrild1. RadioCells[1, 1]. ClearItems(); // 清空单选框内容
EjunGrild1. RadioCells[1, 1]. AddItem(' 学生'); // 添加学生项
EjunGrild1. RadioCells[1, 1]. AddItem(' 教师'); // 添加教师项
EjunGrild1. RadioCells[1, 1]. AddItem(' 校长'); // 添加校长项
```

3.4.5 组合下拉框型

cellComboBox: 组合下拉框型, 出现下拉菜单, 与下拉列表框型不同的时, 组合下拉框型的单元格是可以输入字符串的, 而下拉列表框型不能输入字符串, 通过表格的 ComboCells 对象的 AddItem 方法设置单元格单选框的安钮项.



效果图:

代码示例

[Delphi]

```
EjunGrild1. CellType[1, 1] := cellComboBox; // 设置单个单元格的类型为组合下拉框表框形
EjunGrild1. Columns[1]. CellType := cellComboBox; // 设置第一列的所有单元格的类型为组合下拉框表框形

EjunGrild1. ComboCells[1, 1]. ClearItems(); // 清空下拉框内容
EjunGrild1. ComboCells[1, 1]. AddItem(' 学生'); // 添加学生项
EjunGrild1. ComboCells[1, 1]. AddItem(' 教师'); // 添加教师项
EjunGrild1. ComboCells[1, 1]. AddItem(' 校长'); // 添加校长项
```

3.4.6 下拉列表框型

cellDropDownList: 下拉列表框型, 出现下拉菜单, 与组合下拉框型不同的时, 下拉列表框型的单元格是不可以输入字符串的. 通过表格的 ComboCells 对象的 AddItem 方法设置单元格单选框的安钮项



效果图:

代码示例

[Delphi]

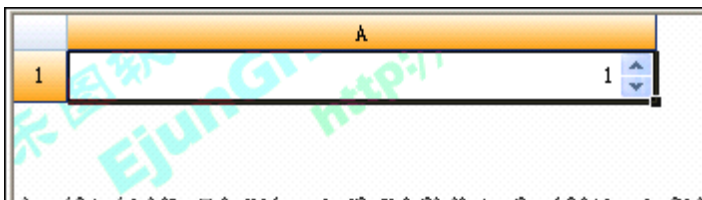
```
EjunGrild1.CellType[1, 1] := cellcellDropDownList; // 设置单个单元格的类型为下拉列表框形
EjunGrild1.Columns[1].CellType := cellcellDropDownList; // 设置第一列的所有单元格的类型为下拉列表框形
```

```
EjunGrild1.ComboCells[1, 1].ClearItems(); // 清空下拉框内容
EjunGrild1.ComboCells[1, 1].AddItem('学生'); // 添加学生项
EjunGrild1.ComboCells[1, 1].AddItem('教师'); // 添加教师项
EjunGrild1.ComboCells[1, 1].AddItem('校长'); // 添加校长项
```

3.4.7 数字旋钮型

cellNumberSpin: 数字旋钮型，在单元格右边出现一个上下按钮，通过点击来步进(退)值，

通过表格的 SpinCells 对象的 Step 属性设置步进值



效果图:

代码示例

[Delphi]

```
EjunGrild1.Columns[1].CellType := cellNumberSpin; // 设置第一列的所有单元格的类型为数字旋钮形
EjunGrild1.CellType[1, 1] := cellNumberSpin; // // 设置单个单元格的类型为数字旋钮形
```

```
EjunGrild1.SpinCells[1, 1].Setp := 1; // 设置步进值为 1
```

我们把数字旋钮类型的单元格改造一下就可以用来当做**日期旋钮**用，只需要两步：

1. 把格式化字符串设成日期类型，例如'yyyy-mm-dd'
2. 设置步进值为1

代码示例

[Delphi]

```
// 设置单元格类型
EjunDataGrid1.CellType[2, 2] := cellNumberSpin;
// 设置显示格式为显示时间类型
EjunDataGrid1.SpinCells[2, 2].Style.FormatString := 'yyyy-mm-dd';
// Delphi 中的日期类型, 1 表示一天
EjunDataGrid1.SpinCells[2, 2].Step := 1;
```

同样我们也可以按照上面的原理制作出**时间旋钮**:

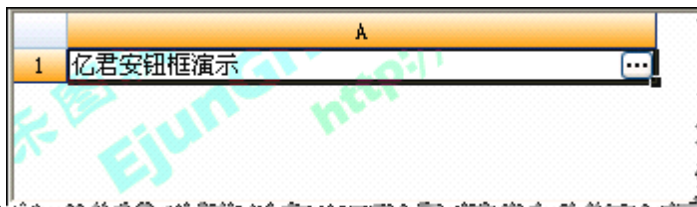
代码示例

[Delphi]

```
// 设置单元格类型
EjunDataGrid1.CellType[2, 2] := cellNumberSpin;
// 设置显示格式为显示时间类型
EjunDataGrid1.SpinCells[2, 2].Style.FormatString := 'hh:mm:ss';
// 设置步进值, 为一分钟, Delphi 中的时间, 1 表示一天, 1/24 表示一个小时,
// 推理 1/24/60 是一分钟, 再除 60 就是一秒种
EjunDataGrid1.SpinCells[2, 2].Step := 1/24/60;
// 设置步进值, 为一秒钟, Delphi 中的时间, 1 表示一天, 1/24 表示一个小时, 推理 1/24/60 是一分钟, 再除 60 就是一秒种
// EjunDataGrid1.SpinCells[2, 1].Step := 1/24/60/60;
```

3.4.8 按钮型

cellEllipsisBtn: 按钮型, 在单元格右边出现一个省略号按钮, 用户点击按钮时, 触发表格的 OnCellButtonClick 事件, 开发人员可以在此事件中进行编码。



效果图:

代码示例

[Delphi]

```
EjunGrid1.CellType[1, 1] := cellEllipsisBtn; // 设置单个单元格的类型为按钮形
EjunGrid1.Columns[1].CellType := cellEllipsisBtn; // 设置第一列的所有单元格的类型为按钮形
```

3.4.9 图表型

cellChart: 图表型。

代码示例

[Delphi]

```
EjunGrid1.CellType[1, 1] := cellChart; // 设置单个单元格的类型为图表形  
EjunGrid1.Columns[1].CellType := cellChart; // 设置第一列的所有单元格的类型为图表形
```

3.4.10 图片型

cellImageBox: 图片型。

设置类型:

代码示例

[Delphi]

```
// 设置一个单元格为图片型  
EjunDataGrid1.CellType[2, 3] := cellImageBox;  
// 设置第二整列单元格的类型为图片型  
EjunDataGrid1.Columns[2].CellType := cellImageBox;
```

指定图片: 单元格本身并不保存图片，表格控件内部有一个图片列表，负责统一管理所有用到的图片对象，要显示图片，只需将图片对象在图片列表中的序号赋给单元格即可，

代码示例

[Delphi]

```
EjunDataGrid1.Cells[2, 3].AsInteger := EjunDataGrid1.LoadImage('d:\2.jpg');
```

EjunGrid 提供了两个方法用来把图片对象添加到内部列表中,他们都返回新图片在列表中的序号,方便使用,就像上面的代码。

- LoadImage(AImageFileName: string): Integer;
- AddImage(AImage: TGraphic): Integer;

在这种机制下，如果多个单元格要显示同一张图片，程序不需要重复加载图片多次，只需给这些单元格指定相同的图片序号即可，这样可以有效节省内存空间。

访问图片列表: EJunGrid 提供了两个属性用来遍历图片列表中的图片对象:

- ImageCount 返回图片数量
- Images[I: Integer]: TGraphic 返回指定序号的图片对象

代码示例

[Delphi]

```
procedure TForm1.btn1Click(Sender: TObject);
var
  I: Integer;
begin
  for I := 0 to EJunDataGrid1.ImageCount - 1 do
    EJunDataGrid1.Images[I].SaveToFile(Format('d:\ejun%d.jpg', [I + 1]));
end;
```

3.4.11 日期选择型

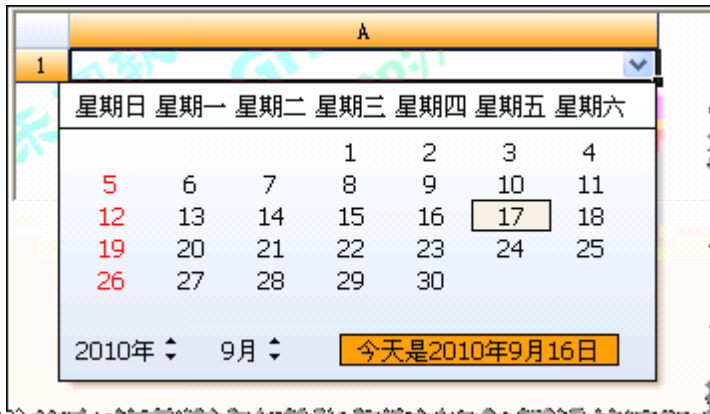
cellChart: 日期选择型。选中这种类型的单元格时，单元格的右边会显示一个下拉按钮，点击该按钮会弹出下拉日期选择框，如果单元格中没有值，日期选择框会默认选中当天的日期，否则会选中单元格中的日期。

编辑日期值可以通过单元格对象的 AsDateTime 属性

代码示例

[Delphi]

```
EJunDataGrid1.Cells[2, 3].AsDateTime := Now;
```



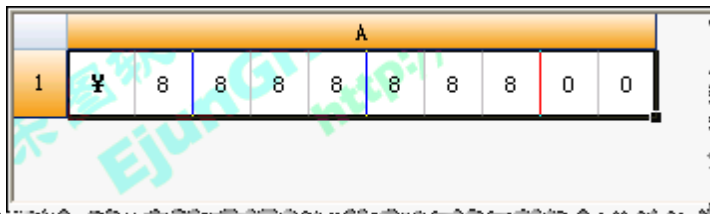
效果图:

代码示例

[Delphi]

```
EjunGrid1.CellType[1, 1] := cellCalendar; // 设置单个单元格的类型为日期选择形
EjunGrid1.Columns[1].CellType := cellCalendar; // 设置第一列所有单元格的类型为日期选择形
```

3.4.12 财务金额型



效果图:

代码示例

[Delphi]

```
EjunGrid1.CellType[1, 1] := cellCurrency; // 设置单个单元格的类型为财务金额形
EjunGrid1.Columns[1].CellType := cellCurrency; // 设置单个单元格的类型为财务金额形
```

3.5 公式计算

EjunGrid 支持公式计算功能, 内部丰富的内置函数让你制作出强大的设计, 用法与 Excel 中操作一样.

EjunGrid 提供了两种公式计算模型: [列公式](#), [单元格公式](#)

列公式以行为计算单位, 一次计算一整行, 当我们给每个列对象设置了 Formula 公式属性之后, 调用

PrepareColumnsFormula 方法与编译列公式。在 4.3.0.3 之后的版本，不再需要开发人员显示的调用 PrepareColumnsFormula 方法了，EjunGrid 做了智能处理，能够判断何时需要编译列公式。

3.5.1 单元格公式

单元格公式，针对单个单元进行公式计算。开发人员只需要对单元格对象的 formula 属性进行赋值即可

代码示例

[Delphi]

```
EjunDataGrid1.Cells[1,1].Formula := '=C6+D6'; // 设置第一个单元的值为，C6 的单元格与 D6 单元格的值之和
```

3.5.1.1 单元格引用

3.5.2 列公式

列公式不同于单元格公式，列公式以行为计算单位，一次计算一整行，当我们给每个列对象设置了 Formula 公式属性之后，调用 PrepareColumnsFormula 方法与编译列公式。在 4.3.0.3 之后的版本，不再需要开发人员显示的调用 PrepareColumnsFormula 方法了，EjunGrid 做了智能处理，能够判断何时需要编译列公式。

基于效率的考虑，EjunGrid 不会自动计算列公式，何时计算开发人员最清楚了，所以我们把这个工作留给了开发人员，要计算某一行只需调用 CalcRow (RowIndex) 方法，参数指定行序号。

代码示例 1

[Delphi]

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    // 设置列公式，注意公式前面等号的作用，如果列公式前面没有等号，相当于给
    // 该列起了一个名称，这个名称可以在其他列的公式中引用
    EjunDataGrid1.Columns[1].Name := 'price';
    EjunDataGrid1.Columns[2].Name := 'qty';
    // 在公式中可以使用函数，这里演示的是，保证最低价为 10 元
    EjunDataGrid1.Columns[3].Formula := '=IF(price<10, 10, price) *qty';
    // 设置合计行的行数
    EjunDataGrid1.FooterRowCount := 1;
    // 设置合计行单元格的计算公式
    EjunDataGrid1.FooterCells[2, 0].Formula := 'SUM(B:B)';
    EjunDataGrid1.FooterCells[3, 0].Formula := 'SUM(C:C)';
end;

// 通常情况下，大家会在单元格的值改变事件中调用，
procedure TForm1.EjunDataGrid1CellValueChanged2(Sender: TObject; Col, Row: Integer);
begin
```

```
TEjunDataGrid(Sender).CalcRow(Row);
end;
```

代码示例 2

[Delphi]

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    // 给 1, 2, 3 列命名, 这样在列公式中就可以使用这些名称
    EjunDataGrid1.Columns[1].Name := 'A';
    EjunDataGrid1.Columns[2].Name := 'B';
    EjunDataGrid1.Columns[3].Name := 'C';
    EjunDataGrid1.Columns[3].Formula := '=A*B'; // 设置 C 列公式
    EjunDataGrid1.Columns[4].Formula := '=C*10'; // 设置 D 列公式
end;
procedure TForm1.EjunDataGrid1CellValueChanged2(Sender: TObject; Col,
    Row: Integer);
begin
    EjunDataGrid1.CalcRow(Row);
end;
```

3.5.3 内置函数

EjunGrid 提供了丰富的内置公式。

POWER

返回某数的乘幂 POWER(number, power)

COUNT

返回包含数字的单元格的个数以及返回参数列表中的数字个数 COUNT(value1, [value2], ...)

IF

判断是否满足某个条件, 如果满足返回一个值, 如果不满足返回另外一个值 IF(logic_test, value_if_true, value_if_false)

SUM

计算单元格区域中所有数值的和 SUM(value1, [value2], ...)

AVERAGE

返回其参数的算术平均值 AVERAGE(number1, [number2], ...)

MIN

返回一组数值中的最小值，忽略逻辑值及文本 MIN(number1, [number2], ...)

MAX

返回一组数值中的最大值，忽略逻辑值及文本 MAX(number1, [number2], ...)

ROW

返回一个引用的行号 ROW(reference)

COLUMN

返回一个引用的列号 COLUMN(reference)

NPV

(未实现) 基于一系列将来的收(正值)支(负值)现金流和一贴现率，返回一项投资的净现值 NPV(rate, value1, [value2], ...)

STDEV

(未实现) 估算基于给定样本的标准偏差(忽略样本中的逻辑值和文本) STDEV(number1, [number2], ...)

DOLLAR

按照货币格式及给定的小数位数，将数字转换成文本 DOLLAR(number, [decimals]);

FIXED

用定点小数格式将数值舍入成特定位数并返回带或不带千分位符的文本 FIXED(number, [decimals], [no_commas])

SIN

返回给定角度的正弦值 SIN(number)

COS

返回给定角度的余弦值 COS(number)

TAN

返回给定角度的正切值 TAN(number)

ATAN

返回反正切值，以弧度表示，大小在 $-\pi/2$ 到 $\pi/2$ 之间 ATAN(number)

PI

返回圆周率 Pi 的值，3.14159265358979，精确到15位 PI()

SQRT

返回数值的平方根 SQTR(number)

EXP

返回 e 的 n 次方 EXP(number)

LN

返回给定数值的自然对数 LN(number)

LOG10

返回给定数值以10为底数的对数 LOG10(number)

ABS

求绝对值 ABS(number)

INT

将数值向下取整为最接近的整数 INT(number)

SIGN

返回数字的正负号：为正时，返回1；为0时，返回0；为负时，返回-1 SIGN(number)

ROUND

按指定的位数对数值进行四舍五入 ROUND(nuber, num_digits)

REPT

根据指定次数重复文本。可用 REPT 在一个单元格中重复填写一个文本字符串 REPT(text, number_times)

MID

返回文本字符串中从指定位置开始的特定数目的字符，该数目由用户指定 MID(text, start_num, num_chars)

LEN

返回文本字符串中的字符个数 `LEN(text)`

SINH

返回双曲正弦值 `SINH(number)`

COSH

返回双曲余弦值 `COSH(number)`

TANH

返回双曲正切值 `TANH(number)`

ASINH

返回反双曲正弦值 `ASINH(number)`

ACOSH

返回反双曲余弦值 `ACOSH(number)`

ATANH

返回反双曲正切值 `ATANH(number)`

3.5.4 用户自定义函数

3.6 撤销与重做

操作历史记录是软件开发一个重要的环节，EjunGrid 当然不会忽略这个环节，EjunGrid 在对操作历史上，设计了良好的接口，让用户可以很方便的实现此功能。

3.6.1 操作历史记录

EjunGrid 框架中，TEjunGridCommandHistory 实现了对表格标准操作历史记录的操作。表格类中聚合了 TEjunGridCommandHistory 于 CommandHistory 属性。用户可以操作此属性，轻松实现撤销与重做。

撤销与重做一次操作：

```
EjunGrid1.CommandHistory.Undo; // 撤销一步
```

```
EjunGrid1.CommandHistory.Redo; // 重做撤销的动作
```

精确撤销与重做操作:

表格的历史记录中记录了撤销与重作历史,

撤销历史表中通过 `EjunGrid1.CommandHistory.GetUndoList` 获取

重作历史表中通过 `EjunGrid1.CommandHistory.GetRdoList` 获取

撤销步骤命令为:`EjunGrid1.CommandHistory.UndoStep(index);`

重作步骤命令为:`EjunGrid1.CommandHistory.RedoStep(index);`

代码示例

[Delphi]

// 加载撤销历史列表

```
var
  UndoList: TStringList;
begin
  UndoList := TStringList.Create;
  Try
    EjunGrid1.CommandHistory.GetUndoList(UndoList);
  // 显示操作
  finally
    UndoList.free;
  end;
end;
```

// 加载重作历史列表

```
var
  RedoList: TStringList;
begin
  RedoList:= TStringList.Create;
  Try
    EjunGrid1.CommandHistory.GetUndoList(RedoList);
  // 显示操作
  finally
    RedoList.free;
  end;
end;
```

// 重做 1 动作

```
EjunGrid1.CommandHistory.RedoStep(1);
```

// 撤销已操作的第 2 动作

```
EjunGrid1.CommandHistory.UndoStep(2);
```

注意: `index` 索引值是基于 1 的

上面的历史记录都是 `EjunGrid` 架框内部已经封装好的标准动作, 如果用户需要添加其他动作的撤销与重做

3.6.1.1 3.6.2 自定义命令

EjunGrid 实现了很多标准的操作命令，如果用户需要添加其他命令动作来进行撤销与重做. 那就要使用到 EjunGrid 的自定义操作命令.

EjunGrid 支持自定义操作命令。只要从命令基类继承下来，然后重载几个方法即可，非常方便。记得要引用 ZcGridCommands 单元.

流程为:

- 1 编写 TZcGridBaseCommand 的派生类, 实现 Exec, Undo 等基类方法
- 2 实例化一个命令对象, 调用表格控件的 ExecuteCommand() 方法, 并将命令对象作为参数传入。

定义命令

代码示例

[Delphi]

```
interfaceuses ZcGridCommands;
```

```
type
```

// 我们定义一个新的命令，用来执行插入行和撤销插入行的操作，首先从基类 TZcGridBaseCommand 继承下来，并重载三个重要的方法。

```
TZcGridInsertRowCommand = class(TZcGridBaseCommand)
```

```
private
```

```
    FRow: Integer;
```

```
    FCount: Integer;
```

```
protected
```

```
    // 返回描述信息，该描述一般显示在撤销下拉菜单中，用来告诉用户我们执行了什么操作
```

```
function GetDescription: EjunString; override;
```

```
    // 执行命令
```

```
procedure Exec; override;
```

```
    // 反向执行命令
```

```
procedure Undo; override;
```

```
public
```

```
    constructor Create(AGrid: TZcCustomGrid; ARow, ACount: Integer);
```

```
end;
```

```
implementationtype
```

```
TZcGridAccess = class(TZcCustomGrid);
```

```
{ TZcGridInsertRowCommand }
```

```
constructor TZcGridInsertRowCommand.Create(AGrid: TZcCustomGrid; ARow, ACount: Integer);
```

```
begin
```

```

    inherited Create(AGrid);
    FRow := ARow;
    FCount := ACount;
end;

procedure TZcGridInsertRowCommand.Exec;
begin
    // 调用 EjunGrid 的内部插入行的方法
    TZcGridAccess(FGrid).InnerInsertRow(FRow, FCount);
end;

function TZcGridInsertRowCommand.GetDescription: EjunString;
begin
    Result := '插入行';
end;

procedure TZcGridInsertRowCommand.Undo;
begin
    // 调用 EjunGrid 的内部删除行的方法
    TZcGridAccess(FGrid).InnerDeleteRow(FRow, FCount);
end;

end.

```

执行命令

调用刚刚定义好的命令，调用步骤是：

- 首先实例化一个命令对象；
- 调用表格控件的 ExecuteCommand() 方法，并将命令对象作为参数传入；

我们并不需要显示的释放命令对象，EjunGrid 内部会负责管理命令对象的生命周期

代码示例

[Delphi]

```

procedure TForm1.btn1Click(Sender: TObject);
begin
    EjunDataGrid1.ExecuteCommand(TZcGridInsertRowCommand.Create(EjunDataGrid1, 2, 2));
end;

```

开放架构

EjunGrid 的命令框架是一个开放的架构，例如，你可以为你的增删改数据库的操作写一个命令，然后让 EjunGrid 来执行，这样你的数据库的操作也可以支持撤销与重做了

3.7 打印

EjunGrid 在对于打印功能上也异常强大，丰富的报表打印接口，让用户可以设置出完美的打印设置

3.7.1 打印

3.7.1.1 如何只打印表格某一范围的内容

EjunGridPrinter 可以只打印表格中某一指定范围内的内容，通过设置 PrintBounds 属性实现

PrintBounds.TopRow 指定打印范围的起始行

.LeftCol 指定打印范围的起始列

.RowCount 指定打印的行数 默认值是-1，表示打印所有行

.ColCount 指定打印的列数 默认值是-1，表示打印所有列

3.7.1.2 如何打印每页都出现的标题行和标题列

可以指定表格的某些行和某些列每页都打印，适合打印标题行和标题列

- PrintBounds.TitleRows 指定从打印范围起始行开始的数行为标题行，每页都打印
- PrintBounds.TitleCols 指定从打印范围起始行开始的数列为标题列，每页都打印

3.7.1.3 如何自动按比例调整列款，使表格刚好占满页面宽度

EjunGridPrinter 默认情况下是根据页面宽度自动横向分页，当指定 FitGridToPageWidth := True 时将不会自动横向分页，而是调整每一列的宽度，使表格刚好适应页面宽度

3.7.1.4 是否可以插入分页符

可以，Grid.Rows[xx].PageBreak := True

3.7.1.5 是否能指定每页打印的行数或者列数

可以，指定 PageColCount, PageRowCount 即可

3.7.1.6 如何设置页眉页脚

EjunGrid 打印控件可以指定页眉页脚每一个条目的位置，页眉页脚区域可以被划分成虚拟的表格，可以指定每一条目所属的行列坐标，还可以指定对齐方式，要实现如上效果的页眉，以下代码可以实现

例如实现下面格式的页眉

xx 报表

编号: xxx

建设单位: xxxx

日期:

工程名称: xxxx

第 1 页 共 5 页

...

在这里为了演示方便，使用代码来设置这些属性，也可以在设计器期在属性编辑器中设置。

代码示例

[Delphi]

```
// 设置报表标题
GridPrinter.ReportTitle.Caption := ' xx 报表' ;
GridPrinter.ReportTitle.Font.Size := 16;
GridPrinter.ReportTitle.Font.Style := [fsBold];
// 设置纸张为横向打印
GridPrinter.Orientation := poLandscape;
with GridPrinter.PageHeader.Add do
begin
  Caption := ' 编号:' ;
  Align := caRight;
  // 这里的行列值和 Grid 的行列值没有关系，而是将页眉划分成虚拟的表格的行列坐标
  Row := 0;
  Col := 2;
  Font := Grid.Font;
end;
with GridPrinter.PageHeader.Add do
begin
  Caption := ' 日期&[date]' ;
  Align := caRight;
  Row := 1;
  Col := 2;
  Font := Grid.Font;
end;
with GridPrinter.PageHeader.Add do
begin
  Caption := ' 第&[Page]页 共&[Pages]页' ;
  Align := caRight;
  Row := 2;
  Col := 2;
  Font := Grid.Font;
end;
with GridPrinter.PageHeader.Add do
begin
  Caption := ' 建设单位: ' ;
  Align := caLeft;
  Row := 1;
  Col := 0;
  Font := Grid.Font;
```

```
end;  
with GridPrinter.PageHeader.Add do  
begin  
  Caption := ' 工程名称: '  ;  
  Align := caLeft;  
  Row := 2;  
  Col := 0;  
  Font := Grid.Font;  
end;
```

3.7.2 打印时自动调整行高

之所以要自动调整行高，是因为我们想自动缩小表格的宽度适应打印纸张的页面宽度，当表格宽度缩小后，原本能显示的单元格内容可能就显示不下了，需要换行显示，这就需要自动调整行高，才能将文本整个显示出来。

EjunGrid 在生成打印页面之前会触发 OnBuildReport 事件，在该事件中，我们可以做一些处理，基本思想是：

- 1 先得到页面的宽度
- 2 统计表格所有需要打印的列的总宽度
- 3 两个宽度相除得到缩放系数
- 4 对每一个需要打印的列的宽度乘以缩放系数得到新的列宽
- 5 调用表格的自动调整行高的函数，重新调整行高

具体代码如下：

代码示例

[Delphi]

```
procedure TFrmMain.EGridPrinterBeforeBuildReport(Sender: TObject;
  AReport: TZcPrintReport);
var
  PageClientWidth: Integer;
  GridClientWidth: Integer;
  I: Integer;
  iTotalWidth: Integer;
  LastVisibleCol: Integer;

  PrintRange: TZcPrintGridRange;
  DataRange: TRect;

  // 计算要打印的数据区域
  procedure CalcPrintDataRange(Grid: TZcCustomGrid);
  begin
    if PrintRange.IsPrintAll then
    begin
      DataRange.Left := Grid.FixedColCount;
      DataRange.Top := Grid.FixedRowCount;
      DataRange.Right := Grid.ColCount - 1;
      DataRange.Bottom := Grid.RowCount - 1;
    end
    else
    begin
      DataRange.Left := PrintRange.LeftCol;
      DataRange.Top := PrintRange.TopRow;
      if PrintRange.RowCount <= 0 then
        DataRange.Bottom := Grid.RowCount - 1
      else
        DataRange.Bottom := DataRange.Top + PrintRange.RowCount - 1;

      if PrintRange.ColCount <= 0 then
        DataRange.Right := Grid.ColCount - 1
      else
        DataRange.Right := DataRange.Left + PrintRange.ColCount - 1;
    end;
  end;

  // 缩放指定列的宽度
  procedure ScaleCols(ACol, ACount: Integer);
  var
    I: Integer;
  begin
    for I := ACol to ACol + ACount - 1 do
    begin
      // 计算缩放后的列的宽度
      GridPrint.Columns[I].Width := MulDiv(GridPrint.Columns[I].Width,
        PageClientWidth, GridClientWidth);
    end;
  end;
end;
```

```

    // 统计实际尺寸
    Inc(iTotalWidth, GridPrint.Columns[I].Width);
    // 记录最后一个可见列
    if GridPrint.Columns[I].Width > 0 then
        LastVisibleCol := I;
    end;
end;
begin
    // 页面客户区宽度
    PageClientWidth := EGridPrinter.Report.PageLayout.ClientWidth;
    // 获取有效的打印范围
    if EGridPrinter.IsUseGridPageSetup then
        PrintRange := GridPrint.PageSetup.PrintRange
    else
        PrintRange := EGridPrinter.PrintRange;
    // 计算要打印的数据范围
    CalcPrintDataRange(GridPrint);

    // 计算表格总宽度
    GridClientWidth := 0;
    // 统计左标题列宽度
    for I := PrintRange.LeftTitleCol to PrintRange.LeftTitleCol + PrintRange.LeftTitleColCount -
1 do
        Inc(GridClientWidth, GridPrint.Columns[I].Width);
    // 统计数据列宽度
    for I := DataRange.Left to DataRange.Right do
        Inc(GridClientWidth, GridPrint.Columns[I].Width);
    // 统计右标题列宽度
    for I := PrintRange.RightTitleCol to PrintRange.RightTitleCol + PrintRange.RightTitleColCount
- 1 do
        Inc(GridClientWidth, GridPrint.Columns[I].Width);

    // 对表格进行缩放
    iTotWidth := 0;
    LastVisibleCol := 0;
    // 缩放左标题列宽度
    ScaleCols(PrintRange.LeftTitleCol, PrintRange.LeftTitleColCount);
    // 缩放数据列宽度
    ScaleCols(DataRange.Left, DataRange.Right - DataRange.Left + 1);
    // 缩放右标题列宽度
    ScaleCols(PrintRange.RightTitleCol, PrintRange.RightTitleColCount);
    // 修正最后一个可见列, 将误差放到他上面
    GridPrint.Columns[LastVisibleCol].Width := PageClientWidth - iTotWidth +
        GridPrint.Columns[LastVisibleCol].Width;
    // 自动调整行高
    GridPrint.AllowAutoAdjustCustomRowHeight := True;
    GridPrint.AdjustRowHeightAuto(DataRange.Top, DataRange.Bottom);
end;

```

3.8 树形表格

3.9 工作簿控件

4 EjunGrid 参考

4.1 对象模型表

DataGrid 数据表格对象	
TreeGrid 树表格对象	
SheetControl 工作表对象	
DBGrid 对象	
DBSheet 对象	
GridPrinter 打印对象	
CommandHistory 历史命令对象	
PreviewPanel 预览面板对象	
Tree 表格树对象	
Cell 单元格对象	
GridColumn 列对象	
GridRow 行对象	
TreeNode 表格树节点对象	
GridSelection 选择区对象	
GridStyle 表格样式对象	
CellGroups 表格线对象	
CellNames 对象	

4.2 **TZcCustomGrid**

4.2.1.1 Properties (属性)

4.2.1.1.1 AllowAutoAdjustCustomRowHeight 属性

todo

Delphi 语法

```
property AllowAutoAdjustCustomRowHeight: Boolean;
```

描述

todo.

4.2.1.1.2 BackgroundImage 属性

todo

Delphi 语法

```
property BackgroundImage: TPicture;
```

描述

todo.

4.2.1.1.3 BorderStyle 属性

todo

Delphi 语法

```
property BorderStyle: TBorderStyle;
```

描述

Todo.

4.2.1.1.4 ColWidths 属性

指定或获取表格列的宽度，单位是像素

Delphi 语法

```
property ColWidths[AIndex: Integer]: Integer;
```

描述

可读写属性,指定列的宽度，列的宽度被改变时会触发 `OnColWidthChange` 事件，要改变行高请参见 [RowHeights](#) 属性.

4.2.1.1.5 RowHeights 属性

可读写属性，表格行的高度，单位是像素

Delphi 语法

```
property RowHeights[AIndex: Integer]: Integer;
```

描述

可读写属性, 表格行的高度, 单位是像素.

参数 `AIndex` 指定要改变或者读取哪一行的高度, 行的高度被改变时会触发 `OnRowHeightChange` 事件, 要改变列的宽度请参见 [ColWidths](#) 属性

4.2.1.1.6 TopLeftCoord 属性

可读写属性, 表格当前滚动的位置, 该属性代表滚动区域最左上角单元格的坐标

Delphi 语法

```
property TopLeftCoord: TZcGridCoord;
```

描述

通过设置该属性, 可以将指定的单元格滚动到滚动区左上角的位置. `TZcGridCoord = TPoint;`

4.2.1.1.7 ViewLockCoord 属性

可读写属性, 指定锁定区域的坐标点, 其效果相当于同时设置了 `ViewLockCol` 和 `ViewLockRow`。

Delphi 语法

```
property ViewLockCoord: TZcGridCoord;
```

描述

`EjunGrid` 支持滚动锁定功能, 被锁定的区域不随滚动条滚动而滚动, 该属性指定锁定区域的坐标点, 其效果相当于同时设置了 `ViewLockCol` 和 `ViewLockRow`.

4.2.1.1.8 Items 属性

只读属性, 返回指定行列坐标上的单元格对象。

Delphi 语法

```
property Items[ACoord: TPoint]: TZcCell;
```

描述

返回参数 `ACoord` 指定行列坐标上的单元格对象, `ACoord.X` 指定列坐标, `ACoord.Y` 指定行坐标。

调用结果指定行列的 [Cells](#) 属性相同

4.2.1.1.9 Cells 属性

返回指定位置的单元格对象

Delphi 语法

```
property Cells[ACol, ARow: Integer]: TZcCell;
```

描述

返回指定位置的单元格对象, ACol 指定单元格所在的列, ARow 指定单元格所在的行。

4.2.1.1.10 Texts 属性

返回指定行列单元格的文本字符串

Delphi 语法

```
property Texts[ACol, ARow: Integer]: EJunString;
```

描述

返回指定行列单元格的文本字符串, 该属性是为了方便使用而对 Cells[ACol, ARow].Text 属性的封装。

4.2.1.1.11 CellClass 属性

用来指定单元格对象类型

Delphi 语法

```
property CellClass: TCellClassManager;
```

描述

EJunGrid 每一个单元格都是一个独立的对象, 单元格对象可以属于不同的类,但他们都从 TZjCell 继承而来, TZjCell 是所有单元格的基类,提供一些核心的基本管理功能, 为了实现某一特定功能的单元格,可以从 TZjCell 继承, 比如 TZjEditCell 实现了编辑框功能的单元格, TZjCheckBoxCell 实现了复选框功能的单元格, TZjEllipsisBtnCell 实现了带省略号按钮风格的单元格

为每一单元格指定不同类型的 Cell 类， 可以通过以下代码实现：

```
Grid.CellClass[Col, Row] := TZjCheckBox;
```

如果需要将表格的某一行的所有单元格都设为 CheckBox 类型， 可以写如下代码：

```
Grid.CellClass.Rows[Row] := TZjCheckBox;
```

同理设置表格的某一列的所有单元格都设为 CheckBox 类型， 代码如下：

```
Grid.CellClass.Cols[Col] := TZjCheckBox;
```

单元格的默认类型是 TZjEditCell, 即文本框类型， 可以通过 DefaultCellClass 属性改变默认单元格类型

```
Grid.CellClass.DefaultCellClass := TZjEllipsisBtnCell;
```

也可以为固定行或者列指定默认的单元格类型

```
Grid.DefaultFixedCellClass := TZjFixedCellXP; // 将表头单元格指定为具有 XP 风格的 Cell 类型
```

注

以上提到的单元格类型都是系统提供的, 在 ZjCells 单元中.

4.2.1.1.12 ClientCells 属性

todo

Delphi 语法

```
property ClientCells[ACol, ARow: Integer]: TZcCell;
```

描述

todo.

4.2.1.1.13 ClientColCount 属性

todo

Delphi 语法

```
property ClientColCount: Integer;
```

描述

todo.

4.2.1.1.14 Style 属性

该属性返回 [TZcGridStyle](#) 对象，用来设置单元格格式信息，包括对齐方式、颜色、边框等

Delphi 语法

```
property Style: TZcGridStyle;
```

描述

[TZcGridStyle](#) 对象用来控制单元格的外观风格，包括文本对齐方式、背景颜色、边框线性、颜色、文本自动换行、只读、打印等。

EJunGrid 采用级联样式表的方式来控制一个具体单元格的格式，EJunGrid.Style EJunGrid.Columns[x].Style, EJunGrid.Cells[x, y].Style 三个层次的格式信息逐层控制，首先是单元格的 [Style](#) 属性，如果是默认值就采用表格列对象的 [Style](#) 属性中的设置，如果也是默认值就采用表格 [Style](#) 属性中设置。

注：使用该属性时需要引用 ZcGridStyle 单元

4.2.1.1.15 Editing 属性

可读写属性，指定表格是否处于编辑状态

Delphi 语法

```
property Editing: Boolean;
```

描述

指定表格是否处于编辑状态，除了在表格上通过双击鼠标左键使单元格进入编辑状态外，还可以通过代码 `Grid.Editing := True` 使当前活动单元格进入编辑状态，同样要退出编辑状态将该属性设为 `False` 即可。

4.2.1.1.16 IsEditing 属性

todo

Delphi 语法

```
property IsEditing: Boolean;
```

描述

todo.

4.2.1.1.17 Editor 属性

todo

Delphi 语法

```
property Editor: TZcEditor;
```

描述

todo.

4.2.1.1.18 DefaultEditor 属性

todo

Delphi 语法

```
property DefaultEditor: TWinControl;
```

描述

todo.

4.2.1.1.19 State 属性

只读属性，返回表格的当前状态

Delphi 语法

```
property State: TGridState;
```

描述

通过该属性可以了解到表格的当前状态，该属性返回 TGridState 类型的枚举值。

4.2.1.1.20 CurCol 属性

返回或者指定选择框所在的行

Delphi 语法

```
property CurCol: Integer;
```

描述

返回或者指定选择框所在的行, 可读写属性. 注意: 行的序号从 0 开始, 包括固定行

4.2.1.1.21 CurRow 属性

返回或者指定选择框所在的列的序号

Delphi 语法

```
property CurRow: Integer;
```

描述

返回或者指定选择框所在列的序号, 可读写属性. 注意, 列的序号从 0 开始, 包括固定列。

4.2.1.1.22 TopRow 属性

可读写属性, 当前视图中最上边的可滚动行的序号, 设置该属性可以在垂直方向滚动表格。

Delphi 语法

```
property TopRow: Integer;
```

描述

通过设置该属性, 可以将指定的行滚动到表格的最上边。

4.2.1.1.23 LeftCol 属性

可读写属性, 当前视图中最左边的可滚动列的序号。

Delphi 语法

```
property LeftCol: Integer;
```

描述

通过设置该属性，可以将指定的列滚动到表格的最左边，不可以指定固定列的序号给该属性。

4.2.1.1.24 UsedRows 属性

todo

Delphi 语法

```
property UsedRows: Integer;
```

描述

todo.

4.2.1.1.25 UsedCols 属性

todo

Delphi 语法

```
property UsedCols: Integer;
```

描述

todo.

4.2.1.1.26 RowVisible 属性

指定表格行的显示隐藏属性

Delphi 语法

```
property RowVisible[ARow: Integer]: Boolean;
```

描述

可读写属性，指定表格行的显示隐藏属性。

显示或者隐藏参数 ACol 指定的行，隐藏行实际上就是将行的高度设为 0，再显示行时，系统会将用内部保存的该行隐藏前的高度恢复。

4.2.1.1.27 ColVisible 属性

指定表格列的显示隐藏属性

Delphi 语法

```
property ColVisible[ACol: Integer]: Boolean;
```

描述

可读写属性, 指定表格列的显示隐藏属性.

显示或者隐藏指定的列, 隐藏列实际上就是将列的宽度设为 0, 再显示列时, 系统会将用内部保存的该列隐藏前的宽度恢复.

4.2.1.1.28 Rows 属性

返回指定行序号的行对象

Delphi 语法

```
property Rows[Index: Integer]: TZcGridRow;
```

描述

返回指定行序号的行对象.要返回指定列序号的列对象, 请调用 [Columns](#) 属性

4.2.1.1.29 ClientRows 属性

todo

Delphi 语法

```
property ClientRows[Index: Integer]: TZcGridRow;
```

描述

todo.

4.2.1.1.30 Model 属性

todo

Delphi 语法

```
property Model: IZjGridModel;
```

描述

todo.

4.2.1.1.31 NewColRowVisible 属性

todo

Delphi 语法

```
property NewColRowVisible: Boolean;
```

描述

todo.

4.2.1.1.32 ClientRange 属性

todo

Delphi 语法

```
property ClientRange: TRect;
```

描述

todo.

4.2.1.1.33 ClientRowCount 属性

todo

Delphi 语法

```
property ClientRowCount: Integer;
```

描述

todo.

4.2.1.1.34 CurCell 属性

返回表格中当前选中的单元格对象

Delphi 语法

```
property CurCell: TZcCell;
```

描述

只读属性, 返回表格中当前选中的单元格对象.

4.2.1.1.35 CurColRow 属性

todo

Delphi 语法

```
property CurColRow: TPoint;
```

描述

todo.

4.2.1.1.36 ShapeCount 属性

返回图表对象的数量

Delphi 语法

```
property ShapeCount: Integer;
```

描述

有关图表对象的描述请参见 对象模型 TZjGridShape 对象.

4.2.1.1.37 Shapes 属性

图表对象数组属性, 返回指定索引号的图表对象

Delphi 语法

```
property Shapes[I: Integer]: TZjGridShape;
```

描述

表格中包含的图表对象的数量可以通过 ShapeCount 获取, 有关图表对象的描述请参见 TZjGridShape

对象.

4.2.1.1.38 HelpmateCount 属性

todo

Delphi 语法

```
property HelpmateCount: Integer;
```

描述

todo.

4.2.1.1.39 Helpmates 属性

todo

Delphi 语法

```
property Helpmates[I: Integer]: TObject;
```

描述

todo.

4.2.1.1.40 Columns 属性

返回指定列序号的列对象

Delphi 语法

```
property Columns[AIndex: Integer]: TZcGridColumn;
```

描述

返回指定列序号的列对象.行列序号都是基于 0 的。访问行序号的行对象请参阅 [Rows](#) 属性

4.2.1.1.41 CommandHistory 属性

todo

Delphi 语法

```
property CommandHistory: TEjunGridCommandHistory;
```

描述

todo.

4.2.1.1.42 CopyRange 属性

todo

Delphi 语法

```
property CopyRange: TRect;
```

描述

todo.

4.2.1.1.43 DontShowZero 属性

todo

Delphi 语法

```
property DontShowZero: Boolean;
```

描述

todo.

4.2.1.1.44 PageOrientation 属性

todo

Delphi 语法

```
property PageOrientation: TPrinterOrientation;
```

描述

todo.

4.2.1.1.45 EditorText 属性

可读写属性，如果单元格处于编辑状态，该属性引用了编辑框内的文本。

Delphi 语法

```
property EditorText: EJunString;
```

描述

如果单元格处于编辑状态，可以通过该属性返回或者设置编辑框内的文本，如果不是处于编辑状态，则返回空字符串。

4.2.1.1.46 EvenRowColor 属性

todo

Delphi 语法

```
property EvenRowColor: TColor;
```

描述

todo.

4.2.1.1.47 FixedCellStyle 属性

todo

Delphi 语法

```
property FixedCellStyle: TZcFixedCellStyle;
```

描述

todo.

4.2.1.1.48 GridLineColor 属性

todo

Delphi 语法

```
property GridLineColor: TColor;
```

描述

todo.

4.2.1.1.49 MergedCellRanges 属性

todo

Delphi 语法

```
property MergedCellRanges[AIndex: Integer]: TRect;
```

描述

todo.

4.2.1.1.50 MergedCellCount 属性

todo

Delphi 语法

```
property MergedCellCount: Integer;
```

描述

todo.

4.2.1.1.51 ImageCount 属性

todo

Delphi 语法

```
property ImageCount: Integer;
```

描述

todo.

4.2.1.1.52 Images 属性

todo

Delphi 语法

```
property Images[AIndex: Integer]: TGraphic;
```

描述

todo.

4.2.1.1.53 ImageKey 属性

todo

Delphi 语法

```
property ImageKey[Aindex: Integer]: string;
```

描述

todo.

4.2.1.1.54 IsCopySouce 属性

todo

Delphi 语法

```
property IsCopySouce: Boolean;
```

描述

todo.

4.2.1.1.55 NameObjects 属性

todo

Delphi 语法

```
property NameObjects: TZcNameObjects;
```

描述

todo.

4.2.1.1.56 TransparentBackground 属性

todo

Delphi 语法

```
property TransparentBackground: Boolean;
```

描述

todo.

4.2.1.1.57 VisibleRowCount 属性

todo

Delphi 语法

```
property VisibleRowCount: Integer;
```

描述

todo.

4.2.1.1.58 KeepEditWhenExit 属性

控制焦点离开表格控件时，不退出编辑框。

Delphi 语法

```
property KeepEditWhenExit: Boolean;
```

描述

控制焦点离开表格控件时，不退出编辑框。

4.2.1.1.59 IsProtected 属性

todo

Delphi 语法

```
property IsProtected: Boolean;
```

描述

todo.

4.2.1.1.60 OddRowColor 属性

todo

Delphi 语法

```
property OddRowColor: TColor;
```

描述

todo.

4.2.1.1.61 NameCells 属性

todo

Delphi 语法

```
property NameCells: TZcNameCells;
```

描述

todo.

4.2.1.1.62 DefaultPasteFormat 属性

todo

Delphi 语法

```
property DefaultPasteFormat: TZcPasteFormat;
```

描述

todo.

4.2.1.1.63 VisualStyle 属性

todo

Delphi 语法

```
property VisualStyle: TZcGridVisualStyle;
```

描述

todo.

4.2.1.1.64 ZebraRowColor 属性

todo

Delphi 语法

```
property ZebraRowColor: Boolean;
```

描述

todo.

4.2.1.1.65 MergeRanges 属性

todo

Delphi 语法

```
property MergeRanges: TUnions;
```

描述

todo.

4.2.1.1.66 Options 属性

指定表格的一些属性选项

Delphi 语法

```
property Options: TGridOptions;
```

描述

todo.

4.2.1.1.67 OptionsEx 属性

Option 属性的扩展选项

Delphi 语法

```
property OptionsEx: TGridOptionsEx;
```

描述

Option 属性的扩展选项.指定表格的一些特性

4.2.1.1.68 FixedColFirstNo 属性

固定列上显示的序号的起始值。

Delphi 语法

```
property FixedColFirstNo: Integer;
```

描述

可读写属性， 固定列上显示的序号的起始值。

指定表格固定列的每一个单元格上会显示从小到大递增的序号，该属性指定序号的起始值

4.2.1.1.69 FixedRowFirstNo 属性

固定行上显示的序号的起始值.

Delphi 语法

```
property FixedRowFirstNo: Integer;
```

描述

可读写属性， 固定行上显示的序号的起始值。

指定表格固定行的每一个单元格上会显示从小到大递增的序号，该属性指定序号的起始值.

4.2.1.1.70 ColCount 属性

可读写属性，指定或者返回表格列的数量

Delphi 语法

```
property ColCount: Integer;
```

描述

该属性返回的列的数量包括固定列，固定列的数量可以通过 `FixedColCount` 属性获取。

4.2.1.1.71 RowCount 属性

可读写属性，指定或者返回表格行的数量

Delphi 语法

```
property RowCount: Integer;
```

描述

该属性返回的行数包括固定行的数量，固定行的数量可以通过 `FixedRowCount` 属性获取。

4.2.1.1.72 FixedRowCount 属性

可读写属性， 固定行的数量

Delphi 语法

```
property FixedRowCount: Integer;
```

描述

指定表格固定行的数量。

4.2.1.1.73 FixedColCount 属性

可读写属性， 固定列的数量。

Delphi 语法

```
property FixedColCount: Integer;
```

描述

指定表格固定列的数量。

4.2.1.1.74 ShowGridLine 属性

是否显示网格线

Delphi 语法

```
property ShowGridLine: Boolean;
```

描述

指定是否显示表格的网格线。

4.2.1.1.75 DefaultColWidth 属性

可读写属性，指定表格列的默认宽度，单位是像素

Delphi 语法

```
property DefaultColWidth: Integer;
```

描述

指定表格列的默认宽度，单位是像素。

4.2.1.1.76 DefaultRowHeight 属性

可读写属性，指定表格行的默认高度，单位是像素

Delphi 语法

```
property DefaultRowHeight: Integer;
```

描述

指定表格行的默认高度。

4.2.1.1.77 DefaultFixedColWidth 属性

可读写属性，指定固定列的默认宽度，单位是像素

Delphi 语法

```
property DefaultFixedColWidth: Integer;
```

描述

指定固定列的默认宽度，单位是像素

4.2.1.1.78 DefaultFixedRowHeight 属性

可读写属性，指定固定行的默认高度，单位是像素

Delphi 语法

```
property DefaultFixedRowHeight: Integer;
```

描述

指定固定行的默认高度，单位是像素。

4.2.1.1.79 SelectColor 属性

可读写属性，指定表格选中区域的颜色

Delphi 语法

```
property SelectColor: TColor;
```

描述

指定选择框所在区域的颜色，默认值是和菜单高亮选择时的颜色一致，用户也可以通过该属性设定自己喜欢的颜色，但为了实现一定的界面效果，用户指定的颜色在内部会被重新计算得出一个合适的值。

选择框还可以实现漂亮的半透明效果，请参见 [Selection](#) 属性

4.2.1.1.80 CurrentMoveDir 属性

指定当用户按回车键时选择框移动的方式

Delphi 语法

```
property CurrentMoveDir: TGridDir;
```

描述

可读写属性，指定当用户按回车键时选择框移动的方式。移动方式详情请参阅 [TGridDir](#)

如果要自定义选择框更复杂的运行轨迹，可以响应 [OnMoveCurrent](#) 事件，例如如下代码实现的是：

当用户按回车时，选择框自动向右移动一列，但跳过第二列，当选择框处于最后一列是，自动跳转到下一行的第一列

```

procedure TForm1.ZJGrid1MoveCurrent(Sender: TObject;
  var NewCurrent: TPoint; ADir: TGridDir);
begin
  // 注意要判断 ADir = gdAuto 才处理，因为 gdAuto 才一定是按回车键触发的，
  // 其他几种移动方式有可能是上下左右方向键触发的
  if ADir = gdAuto then
  begin
    case ZJGrid1.CurCol of
      // 如果当前单元格是第一列就直接跳到第三列
      1: NewCurrent := Point(ZJGrid1.CurCol + 2, ZJGrid1.CurRow)
    else
      if ZJGrid1.CurCol < ZJGrid1.ColCount - 1 then
        // 如果不是最后一列就向右移动一列
        NewCurrent := Point(ZJGrid1.CurCol + 1, ZJGrid1.CurRow)
      else
        // 跳到下一行的第一列
        NewCurrent := Point(1, ZJGrid1.CurRow + 1);
    end;
  end;
end;

```

4.2.1.1.81 HintPause 属性

todo

Delphi 语法

```
property HintPause: Integer;
```

描述

todo.

4.2.1.1.82 ViewLockCol 属性

可读写属性， 指定锁定列的数量。

Delphi 语法

```
property ViewLockCol: Integer;
```

描述

可读写属性， 指定锁定列的数量。

EjunGrid 支持滚动锁定功能，被锁定的区域不随滚动条滚动而滚动，该属性指定被锁定的列的数量，要设置锁定行的数量请参见 [ViewLockRow](#) 属性

4.2.1.1.83 ViewLockRow 属性

可读写属性，指定锁定行的数量。

Delphi 语法

```
property ViewLockRow: Integer;
```

描述

可读写属性，指定锁定行的数量。

EjunGrid 支持滚动锁定功能，被锁定的区域不随滚动条滚动而滚动，该属性指定被锁定的行的数量，要指定锁定列的数量请参见 [ViewLockCol](#) 属性

4.2.1.1.84 AllowEdit 属性

返回或设置表格所有单元格是否可编辑

Delphi 语法

```
property AllowEdit: Boolean;
```

描述

指定表格中所有单元格是否可编辑,也可以通过事件 OnCellCanEdit 单独控制某一个单元格是否可编辑.

4.2.1.1.85 BgColor 属性

可读写属性,用来指定表格的背景颜色,即非表格区域的颜色。表格所在区域的背景颜色有 [Color](#) 属性指定

Delphi 语法

```
property BgColor: TColor;
```

描述

指定非表格区域的颜色,如果该颜色值与表格区域的颜色之相同,系统将会自动计算出一个颜色值,以示区分表格区和非表格区.

4.2.1.1.86 AutoColWidth 属性

todo

Delphi 语法

```
property AutoColWidth: Boolean;
```

描述

todo.

4.2.1.1.87 AutoRowHeight 属性

可读写属性, 用来指定是否根据单元格中文本的高度自动调整行高

Delphi 语法

```
property AutoRowHeight: Boolean;
```

描述

指定是否根据文本高度自动调整行高, 如果自动调整, 行高将等于该行中所有单元格文本高度的最大值.

4.2.1.1.88 LockedCellColor 属性

todo

Delphi 语法

```
property LockedCellColor: TColor;
```

描述

todo.

4.2.1.1.89 Selection 属性

返回 Selection 对象

Delphi 语法

property Selection: [TZcGridSelection](#);

描述

该属性返回 [Selection](#) 对象，[Selection](#) 对象代表表格的选择框，提供了一系列的属性和方法来控制选择框的外观和行为，详细请参见 [Selection](#) 对象。

4.2.1.1.90 UseLockedCellColor 属性

todo

Delphi 语法

```
property UseLockedCellColor: Boolean;
```

描述

todo.

4.2.1.2 Functions(函数)

4.2.1.2.1 SortRow 函数

对表格的行进行排序

Delphi 语法

```
procedure SortRow(ACol: Integer; Ascending: Boolean = True); overload;
```

```
procedure SortRow(ACol1, ACol2: Integer; Ascending: Boolean = True); overload; virtual;
procedure SortRow(ACol1, ACol2, ARow1, ARow2: Integer; Ascending: Boolean = True); overload; virtual;
procedure SortRow(AColumns: array of Integer; Ascending: Boolean = True); overload; virtual;
procedure SortRow(AColumns: array of Integer; ARow1, ARow2: Integer; Ascending: Boolean = True); overload; virtual;
```

描述

表格重载了多种方式对表格的行进行排序

4.2.1.2.2 函数

Delphi 语法

描述

4.2.1.2.3 函数

Delphi 语法

描述

4.3 **TZcCustomDataGrid**

继承于 TZcCustomGrid 类

4.3.1 Properties

4.3.1.1.1 ShowHeaderMenu 属性

是否显示表头菜单，可读写属性。这里的菜单实际上是指排序标识（小三角形）

Delphi 语法

```
property ShowHeaderMenu: Boolean;
```

描述

如果该属性为 `True`，用户点击列头单元格靠近右边线部分，表格控件会对按照所点击的列进行排序，点击一次升序排序，再点击一次就降序排序，如此切换。如果是升序排序会在标头单元格显示正三角形，如果是降序排序会显示倒三角形。

如果该属性为 `False`，用户点击列头单元格不会执行排序操作，也不会显示排序标识三角形。

要判断当前是升序排序还是降序排序，可以访问[列对象](#)的 `SortType` 属性

4.3.1.1.2 Columns 属性

todo

Delphi 语法

```
property Columns[ACol: Integer]: TZcDataGridColumn;
```

描述

todo.

4.3.1.1.3 CellType 属性

指定单元格类型

Delphi 语法

```
property CellType[ACol, ARow: Integer]: TZcCellType;
```

描述

指定单元格类型，`TZcCellType` 是枚举类型，定义了 `EJunGrid` 支持的单元格类型。详细请参阅 `TZcCellType` 类型。

4.3.1.1.4 CellTypes 属性

todo

Delphi 语法

```
property CellTypes[ACol, ARow: Integer]: TZcCellType;
```

描述

todo.

4.3.1.1.5 ColumnCellType 属性

todo

Delphi 语法

```
property ColumnCellType[ACol: Integer]: TZcCellType;
```

描述

todo.

4.3.1.1.6 Charts 属性

返回图表类型的单元格对象.

Delphi 语法

```
property Charts[ACol, ARow: Integer]: TZcChart;
```

描述

返回图表类型的单元格对象, 只读属性.

4.3.1.1.7 CheckCells 属性

返回复选框类型的单元格对象

Delphi 语法

```
property CheckCells[ACol, ARow: Integer]: TZcCheckBoxCell;
```

描述

返回复选框类型的单元格对象.只读属性

4.3.1.1.8 ComboCells 属性

返回下拉框类型的单元格对象

Delphi 语法

```
property ComboCells[ACol, ARow: Integer]: TZcComboBoxCell;
```

描述

返回下拉框类型的单元格对象:

参数 ACol 指定单元格的列号, ARow 指定单元格行号, 如果这个单元格本身不是下拉框类型, 访问该属性时会产生异常。那如何指定单元格的类型为下拉框呢, 我们可以用 [CellType](#) 属性, 例如:

```
EjunDataGrid.CellType[3, 2] := cellComboBox;
```

下面代码演示了如何给下拉框的列项添加内容:

```
EjunDataGrid1.ComboCells[3, 2].AddItem('item1');  
EjunDataGrid1.ComboCells[3, 2].AddItem('item2');  
EjunDataGrid1.ComboCells[3, 2].AddItem('item3');
```

4.3.1.1.9 ImageCells 属性

todo

Delphi 语法

```
property ImageCells[ACol, ARow: Integer]: TZcImageCell;
```

描述

todo.

4.3.1.1.10 CalendarCells 属性

todo

Delphi 语法

```
property CalendarCells[ACol, ARow: Integer]: TZcCalendarCell;
```

描述

todo.

4.3.1.1.11 RadioCells 属性

返回单选框类型的单元格对象

Delphi 语法

```
property RadioCells[ACol, ARow: Integer]: TZcRadioBoxCell;
```

描述

返回单选框类型的单元格对象，两个参数分别为单元格的列序号和行序号。可以通过 [CellType](#) 属性指定单元格类型为单选框。

```
EjunDataGrid1.CellType[4, 2] := cellRadioBox;
EjunDataGrid1.RadioCells[4, 2].AddItem('item1');
EjunDataGrid1.RadioCells[4, 2].AddItem('item2');
```

	A	B	C	D
1				
2				<input checked="" type="radio"/> item1 <input type="radio"/> item2
3				
4				

4.3.1.1.12 SpinCells 属性

返回数字旋钮类型的单元格对象

Delphi 语法

```
property SpinCells[ACol, ARow: Integer]: TZcNumberSpinCell;
```

描述

返回数字旋钮类型的单元格对象，通过 [CellType](#) 属性设置单元格类型，例如：.

```
EjunDataGrid1.CellType[3, 3] := cellNumberSpin; // 设置单个单元格的类型
EjunDataGrid1.Columns[3].CellType := cellNumberSpin; // 设置整列单元格的类型
```

本表格由忆A软件赞助 h:8p://www.ejuC.cn			
1			
2			
3		3	
4			

4.3.1.1.13 DataColumns 属性

todo

Delphi 语法

```
property DataColumns: TZcDataGridDataColumns;
```

描述

todo.

4.3.1.1.14 FooterRowCount 属性

设置或者获取表脚的行数

Delphi 语法

```
property FooterRowCount: Integer;
```

描述

设置或者获取表脚的行数，可读写属性。

表脚也是一个子表格，固定在表格控件的底部，不随垂直滚动条滚动，但会随着水平滚动条滚动，我们一般用表脚表格来显示统计数据。该属性用来指定表脚的行数，默认是 0，即没有表脚。要访问表脚单元格可以通过 [FooterCells](#) 属性。

4.3.1.1.15 FooterCells 属性

返回表脚单元格对象

Delphi 语法

```
property FooterCells[ACol, ARow: Integer]: TZcCell;
```

描述

返回表脚单元格对象，只读属性。

表脚是固定在忆君表格控件底部的子表格，一般用来显示统计数据，该属性是用来访问表脚表格的单元格，表脚表格的第一行序号为0，第二行序号为1，依次类推。设置表脚表格的行数可以用 [FooterRowCount](#) 属性。

应用示例：

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  EjunDataGrid1.FooterRowCount := 1;
end;

procedure TForm1.EjunDataGrid1CellValueChanged(Sender: TObject);
begin
  // 如果发生变化的是第3列的单元格，就重新对第三列求和，并显示在表脚单元格中
  if TZcCell(Sender).Col = 3 then
    EjunDataGrid1.FooterCells[3, 0].AsFloat := EjunDataGrid1.SumCol(3)
  else if TZcCell(Sender).Col = 4 then
    // 计算平均值
    EjunDataGrid1.FooterCells[4, 0].AsFloat := EjunDataGrid1.AvgCol(4);
end;
```

上面的例子并没充分利用 EjunGrid 的特性，我们完全可以用 EjunGrid 的公式计算功能来实现列的求和：

```
EjunDataGrid1.FooterCells[3, 0].Formula := 'SUM(C:C)';
EjunDataGrid1.FooterCells[4, 0].Formula := 'SUM(D:D)';
```

4.3.1.16 Formulas 属性

todo

Delphi 语法

```
property Formulas: TZcGridFormulas;
```

描述

todo.

4.3.1.1.17 ColumnFormulas 属性

todo

Delphi 语法

```
property ColumnFormulas: TZcDataGridColumnFormulas;
```

描述

todo.

4.3.1.1.18 ShowDefaultPopupMenu 属性

控制是否显示默认右键菜单，

Delphi 语法

```
property ShowDefaultPopupMenu: Boolean;
```

描述

控制是否显示默认右键菜单，可读写属性。

当该属性设为 `True`，用户在忆君表格控件上点击鼠标右键时，会显示默认的右键菜单。程序员可以给表格控件指定自定义右键菜单，方法是在窗体上放置一个 `PopupMenu` 菜单控件，然后把表格控件的 `PopupMenu` 属性指向该菜单控件

4.3.1.1.19 PageSetup 属性

todo

Delphi 语法

```
property PageSetup: TEjunSheetPageSetup;
```

描述

todo.

4.3.1.1.20 TextDoc 属性

todo

Delphi 语法

```
property TextDoc: TZcGridTextDocument;
```

描述

todo.

4.3.2 Functions

4.3.2.1 Functions (函数)

4.3.2.1.1 PrepareColumnsFormula 函数

预编译列公式

Delphi 语法

```
procedure PrepareColumnsFormula;
```

描述

当我们在调用 CalcRow 方法计算列公式之前，必须调用该方法预编译列公式。每次改变列公式之后只需要编译一次，不需要每次计算之前都编译一遍。

关于列公式的详细说明请参见 [TZcDataGridColumn.Formula](#) 属性

4.3.2.1.2 CalcRow 函数

计算指定行的列公式。

Delphi 语法

```
procedure CalcRow (ARow: Integer);
```

描述

参数 ARow 指定要计算的行的序号。注意与 [Calculate](#) 方法的区别，CalcRow 计算的是列公式，[Calculate](#) 计算的是单元格公式。可以通过列公式对象的 [Formula](#) 属性指定列公式。通过单元格对象的 [Formula](#) 属性指定单元格公式。

```
// 指定列公式
```

```
EjunDataGrid1.Columns[1].Formula := 'price';  
EjunDataGrid1.Columns[2].Formula := 'count';  
EjunDataGrid1.Columns[3].Formula := '=price * count';  
  
// 指定单元格公式  
EjunDataGrid1.Cells[1, 1].Formula := 'B1 + C2 * 3';
```

4.3.2.1.3 Calculate 函数

计算单元格公式

Delphi 语法

```
procedure Calculate; override;
```

描述

计算单元格公式

4.3.2.1.4 AddPopupMenu 函数

向默认右键菜单中添加弹出菜单项

Delphi 语法

```
function AddPopupMenu(const APath: string; AExecuter: TNotifyEvent; ATag:  
    Integer = 0; AChecked: Boolean = False): TMenuItem;
```

描述

4.3.2.1.5 RemovePopupMenu 函数

从默认右键菜单中删除菜单项

Delphi 语法

```
procedure RemovePopupMenu(const APath: string);
```

描述

4.3.2.1.6 SumCol 函数

对指定列的所有单元格求和

Delphi 语法

```
function SumCol (ACol: Integer): Double;
```

描述

参数 ACol 指定要求和的列序号。

该方法的实现非常简单，把源代码贴在这里，供大家参考。对列求平均值忆君表格控件提供了 AvgCol 方法方便大家使用。

```
function TZcCustomDataGrid.SumCol (ACol: Integer): Extended;  
var  
  I: Integer;  
  V: Extended;  
begin  
  Result := 0;  
  for I := FixedRowCount to RowCount - 1 do  
    if TryCellValue (ACol, I, V) then Result := Result + V;  
end;
```

其中用到了 TryCellValue 方法，该方法尝试判断单元格的值是否是数值型，如果是就返回 True, 否则就返回 False;

4.3.2.1.7 AvgCol 函数

求列平均值

Delphi 语法

```
function AvgCol (ACol: Integer): Double;
```

描述

对指定的列求平均值，参数 ACol 指定列序号。对列求和可以使用 [SumCol](#) 方法

4.3.2.1.8 AddChart 函数

Delphi 语法

```
function AddChart (ACol, ARow, AColSpan, ARowSpan: Integer): TZcChart;
```

描述

4.3.2.1.9 GetRange 函数

Delphi 语法

```
function GetRange(ALeft, ATop, ARight, ABottom: Integer): IZcCellRange;
```

描述

4.3.2.1.10 CreateExcelRW 函数

创建并返回 Excel 读写对象

Delphi 语法

```
function CreateExcelRW: TZcGridExcelRW;
```

描述

这是一个工厂方法，旨在提供一个简便的方法给程序员创建 Excel 读写辅助对象，方便的实现将表格控件的内容保存到 Excel 文件中。

实际上表格控件本身提供了 [SaveToExcel](#) 和 [LoadFromExcel](#) 两个方法，用来方便读写 Excel 文件。但这两个方法一次只能读写一个工作表。如果我们想把多个表格控件保存到一个 Excel 文件的不同工作表中，怎么办呢，有了 [TZcGridExcelRW](#) 辅助对象就可以很容易做到了。

```
// 保存多个表格控件到 Excel 文件中
procedure TForm1.btnSaveExcelClick(Sender: TObject);
var
  ExcelRW: TZcGridExcelRW;
begin
  ExcelRW := EjunDataGrid1.CreateExcelRW;
  try
    ExcelRW.WriteSheet(EjunDataGrid1, 'EjunSheet1');
    ExcelRW.WriteSheet(EjunDataGrid2, 'EjunSheet2');
    ExcelRW.Save('D:\Demo.xls');
  finally
    ExcelRW.Free;
  end;
end;

// 读取 Excel 文件中的多个工作表到不同的表格控件中
procedure TForm1.btnLoadExcelClick(Sender: TObject);
var
  ExcelRW: TZcGridExcelRW;
```

```
begin
  ExcelRW := EjunDataGrid1.CreateExcelRW;
  try
    ExcelRW.Open('D:\Demo.xls');
    ExcelRW.ReadSheet(EjunDataGrid1, 'EjunSheet1');
    ExcelRW.ReadSheet(EjunDataGrid2, 'EjunSheet2');
  finally
    ExcelRW.Free;
  end;
end;
```

记得最后要释放 Excel 读写辅助对象

4.3.2.1.11 ShowPageSetupDialog 函数

Delphi 语法

```
function ShowPageSetupDialog: Boolean;
```

描述

4.3.2.1.12 ColumnByName 函数

查找并返回指定列名的列对象

Delphi 语法

```
function ColumnByName (AColName: EjunString): TZcDataGridColumn;
```

描述

查找并返回指定列名的列对象

EjunGrid 的每一列都有对应的列对象，我们可以给每一个列指定一个名称，列对象的 [ColName](#) 属性表示列的名称，这个方法用来查找指定名称的列对象，如果没有找，就返回 nil。

我们也可以根据列序号通过 [Columns\[AColIndex\]](#) 属性来访问列对象，运行期用户可以通过按住 Alt 键的同时用鼠标左键拖动列头来改变列的顺序，也可以通过方法 [MoveCol](#) 来把列移到其他位置，当列的顺序被改变后，我们在也不能用原来的序号访问列了，这时列名就派上用场了，因为列名不会变，我们可以通过 [ColumnByName](#) 方法来访问列对象。

4.3.2.1.13 ExecuteCommand 函数

Delphi 语法

```
procedure ExecuteCommand(ACommand: TZcGridCommand); override;
```

描述

4.3.3 Events

4.3.3.1.1 OnFormulaGetNameValue

Delphi 语法

```
property OnFormulaGetNameValue: TZcFormulaGetNameValueEvent;
```

描述

4.3.3.1.2 OnCellGetComboItems

Delphi 语法

```
property OnCellGetComboItems: TZcCellGetItemsEvent;
```

描述

4.4 TEjunDataGrid

4.4.1 Properties

4.4.2 Functions

4.4.3 Events

4.5 TEjunTreeGrid

4.6 TEjunSheetControl

4.7 TEjunDBGrid

4.8 TEjunDBSheet

4.9 TEjunGridPrinter

4.10 TEjunGridCommandHistory

4.11 TEjunGridPreviewPanel

4.12 表格枚举列表

4.12.1 TZcCellType

单元格类型枚举

单元

Delphi 语法

```
TZcCellType = (ctNormal, ctCheckBox, ctRadioBox, ctComboBox, ctNumberSpin);
```

描述

单元格类型枚举，下面的表格为 TZcCellType 的可能值

枚举值	含义
ctNormal	
ctCheckBox	
ctRadioBox	
ctRadioBox	

ctComboBox	
ctNumberSpin	

4.12.2 TZcColumnSortType

列排序类型枚举

单元

Delphi 语法

```
TZcColumnSortType = (cstNone, cstAscending, cstDescending);
```

描述

列排序类型枚举，下面的表格为 TZcColumnSortType 的可能值

枚举值	含义
cstNone	不排序
cstAscending	升序排列
cstDescending	降序排列

4.12.3 TZcGridRowOption

表格行选项枚举

单元

Delphi 语法

```
TZcGridRowOption = (roCustomHeight, roPageBreak, roHidden);
TZcGridRowOptions = set of TZcGridRowOption;
```

描述

表格行选项枚举，下面的表格为 TZcGridRowOption 的可能值

枚举值	含义
roCustomHeight	用户自定义行高
roPageBreak	
roHidden	

4.12.4 TSelectionState

表格选择框对象 [Selection](#) 的选择状态枚举

单元

Delphi 语法

```
TSelectionState = (ssNormal, ssFullRow, ssFullCol, ssAll);
```

描述

表格选择框对象 [Selection](#) 的选择状态枚举，下面的表格为 TSelectionState 的可能值

枚举值	含义
ssNormal	表格普通选择
ssFullRow	表格选择区域占满了整行
ssFullCol	表格选择区域占满了整列
ssAll	表示选择区域占满整个客户区域

4.12.5 TZcGridHorzAlign

表格单元格水平对齐方式枚举

单元

ZcGridStyle

Delphi 语法

```
TZcGridHorzAlign = (haGeneral, haLeft, haCenter, haRight, haFill, haJustify, haCenter_across_selection);
```

描述

表格单元格水平对齐方式枚举，除了提供左，右，居中对齐，EjunGrid 还提供了调整对齐，填充对齐等，

下面的表格为 TZcGridHorzAlign 的可能值

枚举值	含义
haGeneral	常规对齐，效果和左对齐效果相同
haLeft	左对齐
haCenter	居中对齐
haRight	右对齐
haFill	填充对齐
haJustify	调整对齐
haCenter_across_selection	

4.12.6 TZcGridVertAlign

表格单元格垂直对齐方式枚举

单元

ZcGridStyle

Delphi 语法

```
TZcGridVertAlign = (vaTop, vaCenter, vaBottom, vaJustify);
```

描述

表格单元格垂直对齐方式枚举，提供置顶对齐，置底对齐，居中对齐，调整对齐。

下面的表格为 TZcGridHorzAlign 的可能值

枚举值	含义
vaTop	置顶对齐
vaCenter	居中对齐
vaBottom	置底对齐
vaJustify	调整对齐 s

4.12.7 TGridDir

表格单元格移动方向枚举

单元

ZJGrid

Delphi 语法

```
TGridDir = (gdAuto, gdLeft, gdRight, gdUp, gdDown, gdNone);
```

描述

表格单元格移动方向枚举，提供置顶对齐，置底对齐，居中对齐，调整对齐。

下面的表格为 TGridDir 的可能值

枚举值	含义
gdAuto	具体的移动方向依赖于表格的 <code>todo:CurrentMoveDir</code> 属性中指定的移动方向。
gdLeft	向左移动一列
gdRight	向右移动一列
gdUp	向上移动一行
gdDown	向下移动一行
gdNone	不移动

4.12.8 TZcGridBorderSide

边框线位置枚举

单元

ZcGridStyle

Delphi 语法

```
TZcGridBorderSide = (gbLeft, gbTop, gbRight, gbBottom, gbDiagonalDown, gbDiagonalDown1, gbDiagonalDown2, gbDiagonalUp, gbDiagonalUp1, gbDiagonalUp2);
```

描述

边框线位置枚举，定义了单元格不同位置的边框线。下面的表格为 TGridDir 的可能值

枚举值	含义
gbLeft	单元格左边的边框线
gbTop	单元格顶边的边框线
gbRight	单元格右边的边框线
gbBottom	单元格底边的边框线
gbDiagonalDown	单元格左上右下的边框线 1
gbDiagonalDown1	单元格左上右下的边框线 2
gbDiagonalDown2	单元格左上右下的边框线 3
gbDiagonalUp	单元格左下右上的边框线 1
gbDiagonalUp1	单元格左下右上的边框线 2
gbDiagonalUp2	单元格左下右上的边框线 3

4.12.9 TZcCellOption

设置单元格的一些特定属性,如锁定单元格，自动折行，设置打印内容等

单元

ZcGridStyle

Delphi 语法

```
TZcCellOption = (gcoLocked, gcoHidden, gcoWrapText, gcoShrinkToFit,
  gcoNotPrint);
TZcCellOptions = set of TZcCellOption;
```

描述

设置单元格的一些特定属性。下面的表格为 TZcCellOption 的可能值

枚举值	含义
gcoLocked	锁定单元格，设置单元格为只读的
gcoHidden	隐藏单元格中的内容

gcoWrapText	当单元格中的文本长度超出单元格宽度时自动折行显示
gcoShrinkToFit	自动调整单元格中文本的字体以适应单元格的宽度(目前没有实现这个功能)
gcoNotPrint	不打印单元格中的内容，主要是实现套打功能

4.12.10 TZcStyleUsedAttrib

设置单元格的 todo

单元

ZcGridStyle

Delphi 语法

```
TZcStyleUsedAttrib = (uaFormat, uaFont, uaAlignment, uaBorder, uaBackground,
    uaProtection, uaOptions);
TZcStyleUsedAttribs = set of TZcStyleUsedAttrib;
```

描述

下面的表格为 TZcStyleUsedAttrib 的可能值

枚举值	含义
uaFormat	
uaFont	
uaAlignment	
uaBorder	
uaBackground	
uaProtection	
uaOptions	

4.12.11 TGridState

4.13 Objects

4.13.1 TZcCell

4.13.1.1 Properties(属性)

4.13.1.1.1 AsBoolean 属性

读取或者设置单元格的布尔类型的值

Delphi 语法

```
property AsBoolean: Boolean read write;
```

描述

EjunGrid 的单元格对象能够保存数值、字符串、布尔三种类型的值。数值类型又分为整型、浮点型、日期型。如果单元格内部保存的值类型和要获取的值类型不一致时会存在一个转换，例如，如果单元格保存的数字类型，那么调用 Cell.AsString 属性获取文本类型的值时，会首先把数值转换成文本类型，当然有时不同类型之间的转换会失败，这时将会触发异常。

通过这些属性读取单元格的值时不会改变单元格中保存的值的类型，而给这些属性赋值时将会改变单元格中实际保存的值的类型。

布尔类型的值在 EjunGrid 内部是通过数值型的值来存储的，0 表示 False，非 0 表示 True；

所以当我们用代码 EjunGrid.Cells[1, 1].AsInteger := 1; 给单元格赋值后，通过 EjunGrid.Cells[1, 1].AsBoolean 属性取出来的值是 True。

参见

[AsString](#)、[AsBoolean](#)、[AsDatetime](#)、[AsInteger](#);

4.13.1.1.2 AsFloat 属性

读取或者设置单元格的浮点类型的值

Delphi 语法

```
property AsFloat:Extended;
```

描述

EJunGrid 的单元格对象能够保存数值、字符串、布尔三种类型的值。数值类型又分为整型、浮点型、日期型。如果单元格内部保存的值类型和要获取的值类型不一致时会存在一个转换，例如，如果单元格保存的数字类型，那么调用 Cell.AsString 属性获取文本类型的值时，会首先把数值转换成文本类型，当然有时不同类型之间的转换会失败，这时将会触发异常。

通过这些属性读取单元格的值时不会改变单元格中保存的值的类型，而给这些属性赋值时将会改变单元格中实际保存的值的类型。

参见

[AsString](#)、[AsBoolean](#)、[AsDatetime](#)、[AsInteger](#);

4.13.1.1.3 AsInteger 属性

以整数类型返回单元格的值，或者给单元格指定整数类型的值，可读写属性

Delphi 语法

```
property AsInteger: Integer;
```

描述

详细说明请参见 [AsFloat](#) 属性

4.13.1.1.4 AsString 属性

可读写属性，按照文本类型返回单元格中的值，或者给单元格指定文本类型的值

Delphi 语法

```
property AsString: string;
```

描述

按照文本类型返回单元格中的值，如果单元格中的值不是文本类型，那么表格将会先把单元格的值转换成文本类型返回。详情请参见 [AsFloat](#) 属性

4.13.1.1.5 AsDateTime 属性

4.13.1.1.6 ColIndex 属性

返回单元格所在的列的序号

Delphi 语法

```
property ColIndex: Integer;
```

描述

返回单元格所在的列的序号，关于单元格行列序号的详细说明请参见 [RowIndex](#) 属性的介绍

4.13.1.1.7 Color 属性

获取或者设置单元格的顏色

Delphi 语法

```
property Color: TColor
```

描述

当通过 Color 属性获取单元格的顏色值时，表格控件会依次检查单元格的 Style 对象，[Column](#) 对象的 Style，[Row](#) 对象的 Style，以及表格的 Style 对象，直到某一个 Style 对象的 BgColor 属性不是缺省值。最后还会触发 [OnCellGetColor](#) 事件，给开发人员更多的控制。

关于单元格级联样式的详细说明请参见 [Style 对象](#)。

对 Color 属性赋值，会直接给单元格对象的 Style.[BgColor](#) 属性赋值。

参见

[Column 属性](#)，[Row 属性](#)，[Style 对象](#)，[BgColor 属性](#)

4.13.1.1.8 ColSpan 属性

单元格横向合并的列数，可读写属性

Delphi 语法

```
property ColSpan: Integer;
```

描述

EJunGrid 支持合并单元格，要设置一个单元格横向合并三列可以用如下代码，ColSpan 等于 1 表示没有横行合并，

```
EJunDataGrid.Cells[1, 1].ColSpan := 3;
```

实现同样的效果可以调用表格对象的 [Merge](#) 方法

```
EJunDataGrid.Merge(1, 1, 3, 1);
```

[Selection](#) 对象也提供了 Merge 方法，用来把选中的多个单元格合并

```
EJunDataGrid1.SelectRange(1, 1, 3, 1);  
EJunDataGrid1.Selection.Merge;
```

RowSpan 属性用来设置一个单元格纵向合并的列数；要判断一个单元格是否是合并单元格，可以访问单元格对象的 [United](#) 属性，True 表示是合并的单元格。关于合并单元格更多的介绍，请参见有关合并单元格的说明

参见

[RowSpan](#), [Merge](#), [Selection](#), [SelectRange](#), [United](#)

4.13.1.1.9 EditText 属性

获取或者设置单元格的编辑文本

Delphi 语法

```
property EditText: WideString;
```

描述

获取或者设置单元格的编辑文本.

4.13.1.1.10 Value 属性

获取或者设置单元格的值

Delphi 语法

```
property Value: Variant;
```

描述

获取或者设置单元格的值.值为 Variant 类型

4.13.1.1.11 Text 属性

获取或者设置单元格的文本字符串值

Delphi 语法

```
property Text: EjunString;
```

描述

获取或者设置单元格的文本字符串值

4.13.1.1.12 Formula 属性

设置或者获取单元格的公式字符串

Delphi 语法

```
property Formula: WideString
```

描述

忆君表格控件可以给单元格设置计算公式，表格控件在运行期根据公式自动计算出单元格的值。

例如：

```
EjunGrid.Cells[3, 1].Formula := '3 * 2 + 4';  
EjunGrid.Cells[3, 2].Formula := 'A1 + A2 * 3';  
EjunGrid.Cells[3, 3].Formula := 'SUM(A1:A5)';
```

也可以直接通过 [Text 属性](#)给单元格设置公式，但必须在公式字符串前面加上“=”，否则 EjunGrid 会把公式当作普通字符串处理

```
EjunGrid.Cells[3, 3].Text := '=SUM(A1:A5)';
```

读取 [Text 属性](#)获得的是计算之后的值，而且是经过 [FormatString](#) 格式化之后的文本。

[EditText 属性](#)返回单元格的编辑文本，如果是公式，会返回带等号的公式字符串。

4.13.1.1.13 IsCurrent 属性

只读属性，用来判断单元格是否是表格控件的当前单元格

Delphi 语法

```
property IsCurrent: Boolean
```

描述

所谓当前单元格，就是 Selection. [Current 属性](#)指定的行列坐标所在的单元格。即如果单元格的 [RowIndex](#) = Selection.Current.Y 并且 [ColIndex](#) = Selection.Current.X 即表示该单元格是当前单元格。

表格对象的 [CurCol](#) 属性也可以用来访问当前列的序号，[CurRow](#) 用来访问当前行的序号，[CurCell 属性](#)用来获得当前单元格对象。

4.13.1.1.14 Range 属性

返回单元格所在的行列坐标区域

Delphi 语法

```
property Range: TRect;
```

描述

TRect 结构有四个字段，分别表示区域的左、上、右、下坐标值，这里的坐标是表格的行列坐标，对于没有合并的单元格，Range 属性返回的区域矩形的 Left, Right 字段相等，等于单元格对象所在的列序号，Top, Bottom 字段也相等，等于单元格对象所在的行序号；对于合并的单元格，Range 返回的区域矩形 Left 字段等于 Cell.ColIndex, Top 字段等于 Cell.RowIndex, Right 字段等于 Cell.ColIndex + Cell.ColSpan - 1, Bottom 字段等于 Cell.RowIndex + Cell.RowSpan - 1。

[ViewRect](#) 返回的是单元格的屏幕像素区域。

4.13.1.1.15 RowIndex 属性

返回单元格所在的行的序号，只读属性

Delphi 语法

```
property RowIndex: Integer;
```

描述

返回单元格所在的行的序号，忆君表格控件的行序号从 0 开始，第一个固定行的行序号等于 0，第二个固定行的行序号等于 1，如果表格有 2 行固定行，那么客户区的第一行的序号等于 2，一次类推。

单元格所在的列的序号可以通过 [ColIndex](#) 获取。

为了使用方便，单元格对象提供了另外一组属性，用来获取一客户区左上角为坐标原点的行列序号：ClientCol 属性，ClientRow 属性。

对于合并的单元格，可以通过 [Range](#) 属性返回整个合并返回

4.13.1.1.16 RowSpan 属性

设置或者获取单元格纵向合并的行数

Delphi 语法

```
property RowSpan: Integer;
```

描述

没有合并的单元格的 RowSpan 属性等于 1，合并两行的等于 2，依次类推，设置单元格纵向合并 3 行可以用如下代码

```
EJunGrid.Cells[1, 1].RowSpan := 3;
```

设置单元格横向合并可以用 [ColSpan](#) 属性。单元格既可以纵向合并也可以横向合并，我们可以调用 EJunGrid 的 [Merge](#) 方法来方便的执行多行多列单元格的合并。要取消行合并，可以把 RowSpan 属性重新设为 1。

参见

[ColSpan](#), [Merge](#)

4.13.1.1.17 Style 属性

返回单元格的 Style 对象，通过 Style 对象可以设置单元格的格式，只读属性。

Delphi 语法

```
property Style: TZcGridStyle;
```

描述

Style 对象维护了单元格的格式信息，包括单元格背景颜色、对齐方式、边框颜色、边框线性、换行显示、锁定、打印等，使用 Style 对象需要引用 ZcGridStyle 单元。

参见

[Style 对象](#)

4.13.1.1.18 Name 属性

单元格名称

Delphi 语法

```
property Name: string;
```

描述

单元格的名字

4.13.1.1.19 RefName 属性

单元格的显示的引用名称。

Delphi 语法

```
property RefName: EjunString;
```

描述

单元格的显示的引用名称。比第 1 行 1 列的单元格为 A1, 2 行 3 列为 C2

4.13.1.1.20 ColName 属性

todo

Delphi 语法

```
property ColName: string;
```

描述

todo.

4.13.1.1.21 Postil 属性

单元格批注

Delphi 语法

```
property Postil: EjunString;
```

描述

单元格的批注默认值为空, 设置批注, 会有一个显示批注效果的对话框浮动在单元格旁边

4.13.1.1.22 Note 属性

单元格备注

Delphi 语法

property Note: EjunString;

描述

单元格的批注默认值为空，设置设置备注属性不会影响界面显示

4.13.1.1.23 Hint 属性

可读写属性，用来指定单元格的提示信息

Delphi 语法

property Hint: WideString

描述

Hint 属性默认为空，当给该属性指定一个文本串之后，当用户把鼠标移动到单元格范围内时，表格控件会显示一个浮动提示窗口。

关于提示窗口更详细的信息请参见表格控件的 [ShowHint 属性](#)

4.13.1.1.24 Data 属性

指定特定应用的指针数据

Delphi 语法

property Data: Pointer;

描述

通过单元格的 Data 属性访问用户绑定的特定数据.用法和常见的 Data 属性（TListItem）一样.行列对象也拥有 Data 属性，用户可以选择绑定指定数据到 Data 属性上.

4.13.1.1.25 Tag 属性

指定特定字符标记

Delphi 语法

property Tag: EjunString;

描述

Tag 没有任何预定义意义，默认值为空，和户可以标记特定的字符.

4.13.1.1.26 Union 属性

todo

Delphi 语法

```
property Union: TUnion;
```

描述

Todo.

4.13.1.1.27 ItemCount 属性

todo

Delphi 语法

```
property ItemCount: Integer;
```

描述

Todo.

4.13.1.1.28 AsItems 属性

todo

Delphi 语法

```
property AsItems[Index: Integer]: EjunString;
```

描述

Todo.

4.13.1.1.29 Col 属性

todo

Delphi 语法

```
property Col: Integer;
```

描述

todo.

4.13.1.1.30 Row 属性

todo

Delphi 语法

```
property Row: Integer;
```

描述

todo.

4.13.1.1.31 ViewRect 属性

单元格的屏幕像素区域

Delphi 语法

```
property ViewRect: TRect;
```

描述

单元格的屏幕像素区域.

4.13.1.1.32 ViewWidth 属性

todo

Delphi 语法

```
property ViewWidth: Integer;
```

描述

todo.

4.13.1.1.33 ViewHeight 属性

todo

Delphi 语法

```
property ViewHeight: Integer;
```

描述

todo.

4.13.1.1.34 ViewTop 属性

todo

Delphi 语法

```
property ViewTop: Integer;
```

描述

todo.

4.13.1.1.35 ViewLeft 属性

todo

Delphi 语法

```
property ViewLeft: Integer;
```

描述

todo.

4.13.1.1.36 ClientRect 属性

todo

Delphi 语法

```
property ClientRect: TRect;
```

描述

todo.

4.13.1.1.37 Canvas 属性

提供访问单元格绘图界面

Delphi 语法

```
property Canvas: TCanvas;
```

描述

Canvas 允许在单元格上进行绘制。

Canvas 对象会自动创建，这个属性为只读

4.13.1.1.38 CellType 属性

单元格类型

Delphi 语法

```
property CellType: TZcCellType;
```

描述

单元格默认的类型为文本类型，单元格还提供了单选按钮类型，复选框类型等，详情请看 [TZcCellType](#) 枚举

4.13.1.1.39 United 属性

todo

Delphi 语法

```
property United: Boolean;
```

描述

todo.

4.13.1.1.40 IsUnionMainCell 属性

判断合并区域的主单元格

Delphi 语法

```
property IsUnionMainCell: Boolean;
```

描述

使用 `IsUnionMainCell` 属性判断是否是合并区域的主单元格，也就是合并区域的左上角的那个单元格。只读属性。

4.13.1.1.41 `IsActive` 属性

判断单元格是否活动状态

Delphi 语法

```
property IsActive: Boolean;
```

描述

判断单元格是否活动状态,只读属性

4.13.1.1.42 `IsCurrent` 属性

todo

Delphi 语法

```
property IsCurrent: Boolean;
```

描述

todo.

4.13.1.1.43 `Grid` 属性

得到当前单元格所在的表格对象

Delphi 语法

```
property Grid: TZCustomGrid;
```

描述

得到当前单元格所在的表格对象,只读属性

4.13.1.1.44 `LeftCell` 属性

当前单元格左边的单元格

Delphi 语法

```
property LeftCell: TZCell;
```

描述

返回当前单元格左边的单元格对象，如果左边没有单元格，返回 nil，只读属性。

4.13.1.1.45 UpsideCell 属性

当前单元格上边的单元格

Delphi 语法

```
property UpsideCell: TZcCell;
```

描述

返回当前单元格上边的单元格对象，如果上边没有单元格，返回 nil，只读属性。

4.13.1.1.46 RightCell 属性

当前单元格右边的单元格对象

Delphi 语法

```
property RightCell: TZcCell;
```

描述

返回当前单元格右边的单元格对象，如果右边没有单元格，返回 nil，只读属性。

4.13.1.1.47 UndersideCell 属性

当前单元格底边的单元格对象

Delphi 语法

```
property UndersideCell: TZcCell;
```

描述

返回当前单元格底边的单元格对象，如果底边没有单元格，返回 nil，只读属性。

4.13.1.1.48 TextWidth 属性

单元格显示文本的宽度

Delphi 语法

```
property TextWidth: Integer;
```

描述

得到单元格显示文本的宽度,只读属性

4.13.1.1.49 TextHeight 属性

单元格显示文本的高度

Delphi 语法

```
property TextHeight: Integer;
```

描述

得到单元格显示文本的高度

4.13.1.1.50 Width 属性

单元格宽度

Delphi 语法

```
property Width: Integer;
```

描述

获取或者设置单元格的宽度

4.13.1.1.51 Height 属性

单元格高度

Delphi 语法

```
property Height: Integer;
```

描述

获取或者设置单元格的高度

4.13.1.1.52 Font 属性

单元格显示字体对象

Delphi 语法

```
property Font: TFont;
```

描述

单元格显示字体对象,控制显示字体, 字体大小, 形状, 颜色等, 可以 new 一个 TFont 对象修改字体对象, 也可以直接修改字体对象的属性.

4.13.1.1.53 属性

todo

Delphi 语法

```
property : ;
```

描述

todo.

4.13.1.1.54 属性

todo

Delphi 语法

```
property : ;
```

描述

todo.

4.13.1.1.55 属性

todo

Delphi 语法

```
property : ;
```

描述

todo.

4.13.1.2 Functions (函数)

4.13.1.2.1 Invalidate 函数

使单元格所在的屏幕区域无效，促使 EJunGrid 重新绘制该单元格

Delphi 语法

```
procedure Invalidate; virtual;
```

描述

使单元格所在的屏幕区域无效，促使 EJunGrid 重新绘制该单元格，[ViewRect](#) 可以用来获取单元格所占的屏幕区域，该属性返回 TRect 结构。

4.13.1.2.2 IsEmpty 函数

单元格是否为空

Delphi 语法

```
function IsEmpty: Boolean; virtual;
```

描述

判断单元格是否为空,这里为空的标准为字符串为空 todo.

4.13.1.2.3 Perform 函数

当接受到特定窗体单元格消息时，进行回应

Delphi 语法

```
function Perform(Msg: Cardinal; WParam, LParam: Longint): Longint;
```

描述

接受到特定窗体单元格消息时的回应, todo

4.13.1.2.4 IsFixedRowCell 函数

判断是否是固定行中的单元格

Delphi 语法

```
function IsFixedRowCell: Boolean;
```

描述

判断是否是固定行中的单元格。

如果要判断是否是固定列的单元格，请调用 [IsFixedColCell](#)，判断是否是固定行列相交的单元格，请调用

[IsFixedCornerCell](#)

4.13.1.2.5 IsFixedColCell 函数

判断是否是固定列中的单元格

Delphi 语法

```
function IsFixedColCell: Boolean;
```

描述

判断是否是固定列中的单元格。

如果要判断是否是固定行的单元格，请调用 [IsFixedRowCell](#)，判断是否是固定行列相交的单元格，请调用

[IsFixedCornerCell](#)

4.13.1.2.6 IsFixedCornerCell 函数

判断是否是固定行列相交的单元格

Delphi 语法

```
function IsFixedCornerCell: Boolean;
```

描述

判断是否是固定行列相交的单元格，

如果要判断是否是固定行的单元格，请调用 [IsFixedRowCell](#)，判断是否是固定列的单元，请调用 [IsFixedRowCell](#)

4.13.1.2.7 IsClientCell 函数

判断单元格是否是客户区单元格

Delphi 语法

```
function IsClientCell: Boolean;
```

描述

判断单元格是否是客户区单元格

4.13.1.2.8 Assign 函数

拷贝目标 Source 的内容及样式到单元格

Delphi 语法

```
procedure Assign(Source: TPersistent); override;
```

描述

拷贝目标 Source 的值及样式到单元格, Source 虽然标明为 TPersistent 类型, 如果调用,参数应该传入单元格 TZcCell 类型.

如果只想复制指定单元格的值请使用 [CopyValue](#), 复制样式请使用 [CopyStyle](#)

4.13.1.2.9 CopyStyle 函数

拷贝指定单元格的样式

Delphi 语法

```
procedure CopyStyle(Source: TZcCell);
```

描述

复制单元格样式, 包括合并状态、字体、对其方式、背景颜色、换行、锁定、打印等样式属性, 即复制单元格的 Style 对象.

如果想复制单元格值可以用 [CopyValue](#) 方法, [Assign](#) 方法是两者都复制。

4.13.1.2.10 CopyValue 函数

复制指定单元格对象的值

Delphi 语法

```
procedure CopyValue(ACell: TZcCell);
```

描述

复制指定单元格对象的值，参数 ACell 指定源单元格对象。复制单元格样式可以用 [CopyStyle](#) 方法，[Assign](#) 方法是两者都复制。

4.13.1.2.11 Clear 函数

清除单元格

Delphi 语法

```
procedure Clear(AEditMode, AClearStyle: Boolean); overload; virtual;  
procedure Clear; overload; virtual;
```

描述

清除单元格。表格提供了二个重载方法。

procedure Clear; overload; virtual;

清空样式和值。

procedure Clear(AEditMode, AClearStyle: Boolean); overload; virtual;

清空单元格中的值，该方法有二个参数，

AEditMode:Boolean 类型。指明是否是编辑模式，所谓编辑模式，是指终端客户在表格控件中按 Del 键来删除单元格内容时触发该方法调用时的状态。在编辑模式下，被锁定的单元格是不能被清空的，非编辑模式下，所有的单元格都可以被清空，这种模式一般是给程序员写代码删除单元格时使用。

AClearStyle:Boolean 类型。指明是否是清空表格样式

如果用户自己写了派生类，可以重写清除方法。设置单元格锁定可以设置 Style 对象的 [Locked 属性](#)

4.13.1.2.12 Compare;函数

与指定单元格进行比较

Delphi 语法

```
function Compare(ADest: TZcCell): Integer; virtual;
```

描述

与指定单元格进行比较,

4.13.1.2.13 Calculate 函数

计算单元格中设置的公式

Delphi 语法

```
procedure Calculate;
```

描述

如果单元格中设置的有公式，就计算该公式

4.13.1.2.14 GetRealStyle 函数

Delphi 语法

```
function GetRealStyle(Attrib: TZcStyleUsedAttrib): TZcGridStyle;
```

描述

4.13.1.2.15 InsertItem 函数

Delphi 语法

```
procedure InsertItem(Index: Integer; const S: EjunString); virtual;
```

描述

4.13.1.2.16 AddItem 函数

Delphi 语法

```
function AddItem(const S: EjunString): Integer; virtual;
```

描述

4.13.1.2.17 ClearItems 函数

Delphi 语法

```
procedure ClearItems; virtual;
```

描述

4.13.1.2.18 IsFormula 函数

是否是公式单元格

Delphi 语法

```
function IsFormula: Boolean; override;
```

描述

判断单元格内是否有公式，如果给单元格设置了公式，返回为 true

4.13.1.2.19 Print 函数

打印

Delphi 语法

```
procedure Print(ACanvas: TCanvas; const ARect: TRect); virtual;
```

描述

4.13.1.2.20 Select 函数

选中单元格

Delphi 语法

```
procedure Select;
```

描述

选中单元格

4.13.1.3 Events(事件)

4.13.2 TZcDataGridColumn

表格数据列类，继承与 TZcGridColumn.

单元

4.13.2.1 Properties(属性)

4.13.2.1.1 CellType 属性

数据列的单元格类型

Delphi 语法

```
property CellType: TZcCellType;
```

描述

数据列的单元格类型.

4.13.2.1.2 CellTypeName 属性

单元的类型名字

Delphi 语法

```
property CellTypeName: string;
```

描述

单元的类型名字.

4.13.2.1.3 Title 属性

单元列的列标题

Delphi 语法

```
property Title: EJunString;
```

描述

单元列的列标题.

4.13.2.1.4 Formula 属性

todo

Delphi 语法

```
property Formula: EJunString;
```

描述

todo.

4.13.2.1.5 SortType 属性

表格列数据的排序方式,

Delphi 语法

```
property SortType: TZcColumnSortType;
```

描述

表格列数据的排序方式,默认为不排序,设置不同的 [TZcColumnSortType](#) 值,可以设置不排序,升序,降序。可读可写属性.

指定该属性并不能让忆君表格控件执行真正的排序操作,而仅仅是告诉表格控件如何在列头(固定列)单元格上显示排序标志,指定 `cstAscending` 表示升序显示正三角形, `cstDescending` 表示降序显示倒三角形, `cstNone` 不显示排序标志。

当用户在列头单元格的靠近右边线的区域点击鼠标时，表格控件会对点击的列进行排序，表格内部会设置该列的 `SortType` 属性，告诉固定单元格绘制正确的排序标志。

如果被点击列的 `SortType` 属性为 `cstNone`，表格会对该列正序排序，然后给 `SortType` 属性设为 `cstAscending`，同时会把其他所有列的 `SortType` 属性设为 `cstNone`；

如果被点击列的 `SortType` 属性为 `cstAscending`，表格会对该列倒序排列，然后给 `SortType` 属性设为 `cstDescending`，同时把其他所有列设为 `cstNone`；

如果被点击列的 `SortType` 属性为 `cstDescending`，表格会对该列再执行正序排列，然后给 `SortType` 属性设为 `cstAscending`，同时把其他所有列设为 `cstNone`；

如此反复，列被采用正序和倒序切换排序。

如果表格控件的 `ShowHeaderMenu` 属性设为 `False`，点击列头单元格时不会执行排序操作的，但直接调用 `SortRow` 方法仍会排序。

表格内部调用 `SortRow` 方法执行真正的排序操作，如果程序员用代码直接调用 `SortRow` 方法进行排序，忆君表格不会自动帮您设置 `SortType` 属性。

4.13.2.2 Functions(函数)

4.13.2.2.1 HasCellItems 函数

判断是否拥有 `CellItems`

Delphi 语法

```
function HasCellItems: Boolean;
```

描述

判断是否拥有 `CellItems`

4.13.2.2.2 函数

Delphi 语法

描述

4.13.2.3 Events(事件)

4.13.3 TZcGridRow

表格行对项，控制整行的单元格信息，样式

Properties(属性)

4.13.3.1.1 Data 属性

todo

Delphi 语法

```
property Data: Pointer;
```

描述

todo.

4.13.3.1.2 Grid 属性

todo

Delphi 语法

```
property Grid: TZcCustomGrid;
```

描述

得到当前单元格所在的表格对象,只读属性

4.13.3.1.3 Options 属性

表格行选项

Delphi 语法

property Options: [TZcGridRowOption](#);

描述

表格行选项.

4.13.3.1.4 Range 属性

行所在的区域

Delphi 语法

property Range: TRect;

描述

行所在的区域

4.13.3.1.5 Cells 属性

得到表格行中指定列的单元格

Delphi 语法

property Cells[ACol: Integer]: [TZcCell](#);

描述

得到表格行中指定列的单元格

4.13.3.1.6 Caption 属性

todo

Delphi 语法

property Caption: string;

描述

todo.

4.13.3.1.7 Tag 属性

todo

Delphi 语法

```
property Tag: EJunString;
```

描述

todo.

4.13.3.1.8 PageBreak 属性

todo

Delphi 语法

```
property PageBreak: Boolean;
```

描述

todo.

4.13.3.1.9 Style 属性

todo

Delphi 语法

```
property Style: TZcGridStyle;
```

描述

todo.

4.13.3.1.10 Visible 属性

todo

Delphi 语法

```
property Visible: Boolean;
```

描述

todo.

4.13.3.1.11 RowIndex 属性

todo

Delphi 语法

```
property RowIndex: Integer;
```

描述

todo.

4.13.3.1.12 Height 属性

todo

Delphi 语法

```
property Height: Integer;
```

描述

todo.

4.13.3.1.13 CustomHeight 属性

todo

Delphi 语法

```
property CustomHeight: Boolean;
```

描述

Todo.

4.13.3.1.14 Level 属性

todo

Delphi 语法

```
property Level: Integer;
```

描述

todo.

4.13.3.2 Functions (函数)

4.13.3.2.1 Exchange 函数

Delphi 语法

```
procedure Exchange (ADestIndex: Integer);
```

描述

4.13.3.2.2 MoveTo 函数

Delphi 语法

```
procedure MoveTo (ADestIndex: Integer);
```

描述

4.13.3.2.3 Delete 函数

删除行

Delphi 语法

```
procedure Delete;
```

描述

删除行

4.13.3.2.4 Clear 函数

清空行所有单元格的内容

Delphi 语法

```
procedure Clear(AEditMode: Boolean = True);
```

描述

清空行所有单元格的内容

4.13.3.2.5 Invalidate 函数

Delphi 语法

```
procedure Invalidate;
```

描述

4.13.3.3 Events(事件)

4.13.4 TZcGridRange

表格区域类

单元

ZjGrid

4.13.4.1 Properties(属性)

4.13.4.1.1 Grid 属性

todo

Delphi 语法

```
property Grid: ;
```

描述

todo.

4.13.4.1.2 ColCount 属性

todo

Delphi 语法

```
property ColCount: Integer;
```

描述

todo.

4.13.4.1.3 RowCount 属性

todo

Delphi 语法

```
property RowCount: Integer;
```

描述

todo.

4.13.4.1.4 Bottom 属性

todo

Delphi 语法

```
property Bottom: Integer;
```

描述

4.13.4.1.5 Left 属性

todo

Delphi 语法

```
property Left: Integer;
```

描述

todo.

4.13.4.1.6 Right 属性

todo

Delphi 语法

```
property Right: Integer;
```

描述

todo.

4.13.4.1.7 Top 属性

todo

Delphi 语法

```
property Top: Integer;
```

描述

todo.

4.13.4.2 Functions(函数)

4.13.4.2.1 ForAll 函数

遍历所有的单元格,执行用户传入的回调函数

Delphi 语法

```
TExecCellProc = procedure (ACell: TZcCell);  
procedure ForAll(Proc: Pointer{TExecCellProc}); virtual;
```

描述

遍历所有的单元格,执行用户传入的回调函数。回调函数指针原形为 TExecCellProc。当用户想对所选的单
元格进行特殊操作, 就可以调用此方法

4.13.4.2 Invalidate 函数

刷新区域。

Delphi 语法

```
procedure Invalidate; virtual;
```

描述

刷新区域。

4.13.4.3 SetBorder 函数

设置区域边框线

Delphi 语法

```
procedure SetBorder(ASides: TZcGridBorderSides; AStyle: TZcGridBorderStyle;  
    AColor: TColor; AInerRow: Boolean = False; AInnerCol: Boolean = False);
```

描述

设置区域边框线,可以设置边框线精细, 颜色 , 边框线的位置

4.13.4.2.4 IsContainCoord 函数

Delphi 语法

```
function IsContainCoord(const ACoord: TPoint): Boolean;
```

描述

4.13.4.2.5 IsFullCol 函数

Delphi 语法

```
function IsFullCol: Boolean; override;
```

描述

4.13.4.2.6 IsFullRow 函数

Delphi 语法

```
function IsFullRow: Boolean; override;
```

描述

4.13.4.2.7 SetWrapText 函数

设置单元格自动换行。

Delphi 语法

```
procedure SetWrapText(const Value: Boolean);
```

描述

将选择框范围内的所有单元格设置为自动换行。

```
procedure TForm1.tbWrapTextClick(Sender: TObject);
begin
  // 设置选择范围
  EjunDataGrid1.SelectRange(2, 2, 5, 5);
  // 设置单元格自动换行
  EjunDataGrid1.Selection.SetWrapText(True);
  // 自动调整行高以完全显示换行后的文本
  EjunDataGrid1.Selection.AdjustRowHeightAuto;
end;
```

	A	B	C	D	E	F
1						
2						
3						
4						
5						
6						

本表格由忆A软件赞助 h t t p : / / w w w . e j u n . c o m

EjunGrid是一款优秀的报表控件

EjunGrid是一款优秀的报表控件

4.13.4.2.8 SetFontItalic 函数

设置字体风格是否为斜体

Delphi 语法

```
procedure SetFontItalic(const Value: Boolean);
```

描述

设置选择范围内的单元格的字体是否采用斜体

```
// 设置选择范围  
EjunDataGrid1.SelectRangle(2, 2, 5, 5);  
// 设置单元格字体为斜体  
EjunDataGrid1.Selection.SetFontItalic(True);
```

4.13.4.2.9 SetFontBold 函数

设置单元格字体是否为粗体

Delphi 语法

```
procedure SetFontBold(const Value: Boolean);
```

描述

设置选择范围内的所有单元格为粗体或取消粗体，参数 Value 为 True 表示设置为粗体，False 表示设置为正常字体。

```
// 设置选择范围  
EjunDataGrid1.SelectRangle(2, 2, 5, 5);  
// 设置单元格字体为粗体  
EjunDataGrid1.Selection.SetFontBold(True);
```

4.13.4.2.10 SetFontUnderline 函数

设置单元格字体是否显示下划线。

Delphi 语法

```
procedure SetFontUnderline(const Value: Boolean);
```

描述

设置选择范围内的所有单元格字体是否显示下划线，参数 **Value** 等于 **true** 时表示显示下划线，否则表示不显示下划线

```
// 设置选择范围  
EjunDataGrid1.SelectRangle(2, 2, 5, 5);  
// 设置单元格字体显示下划线  
EjunDataGrid1.Selection.SetFontUnderline(True);
```

4.13.4.2.11 SetFontStrikeOut 函数

设置单元格字体是否显示删除线

Delphi 语法

```
procedure SetFontStrikeOut(const Value: Boolean);
```

描述

设置选择范围内的所有单元格字体是否显示删除线，参数 **Value** 为 **True** 时表示显示删除线，**False** 表示不显示删除线

```
// 设置选择范围  
EjunDataGrid1.SelectRangle(2, 2, 5, 5);  
// 设置单元格字体显示删除线  
EjunDataGrid1.Selection.SetFontStrikeOut(True);
```

4.13.4.2.12 SetFontSize 函数

设置单元格字体大小。

Delphi 语法

```
procedure SetFontSize(const ASize: Integer);
```

描述

设置选择范围内的所有单元格的字体大小为参数 **ASize** 指定的大小。

```
// 设置选择范围  
EjunDataGrid1.SelectRange(2, 2, 5, 5);  
// 设置单元格字体大小  
EjunDataGrid1.Selection.SetFontSize(10);
```

4.13.4.2.13 SetFontName 函数

设置单元格字体名称

Delphi 语法

```
procedure SetFontName(const AName: TFontName);
```

描述

设置选择范围内的所有单元格使用同一种字体，参数 AName 指定字体名称。

```
// 设置选择范围  
EjunDataGrid1.SelectRange(2, 2, 5, 5);  
// 设置单元格字体名称  
EjunDataGrid1.Selection.SetFontName('宋体');
```

4.13.4.2.14 SetFontColor 函数

设置单元格字体颜色

Delphi 语法

```
procedure SetFontColor(const Value: TColor);
```

描述

设置选择范围内的所有单元格使用同一种字体颜色，参数 Value 指定颜色值。

```
// 设置选择范围  
EjunDataGrid1.SelectRange(2, 2, 5, 5);  
// 设置单元格字体颜色  
EjunDataGrid1.Selection.SetFontColor(clRed);
```

4.13.4.2.15 SetLocked 函数

设置单元格是否锁定

Delphi 语法

```
procedure SetLocked(const Value: Boolean);
```

描述

设置单元格是否锁定，锁定的单元格是不允许编辑的。

该方法实际上是批量设置选择框范围内所有单元格的 `Style.Locked` 属性

4.13.4.2.16 SetBgColor 函数

设置单元格背景颜色

Delphi 语法

```
procedure SetBgColor(const Value: TColor);
```

描述

设置选择范围内所有单元格的背景颜色。

该方法实际上是批量设置选择框范围内所有单元格的 `Style.BgColor` 属性。

4.13.4.2.17 SetFontStyle 函数

Delphi 语法

```
procedure SetFontStyle(const Value: TFontStyles);
```

描述

4.13.4.2.18 SetHorzAlign 函数

设置单元格文本水平对齐方式

Delphi 语法

```
procedure SetHorzAlign(const Value: TZcGridHorzAlign);
```

描述

参数 Value 是 TZcGridHorzAlign 枚举类型，指定对齐方式。

该方法实际上是批量设置选择框范围内所有单元格的 Style.HorzAlign 属性。

关于 TZcGridHorzAlign 类型的详细介绍请参见 [HorzAlign](#)。

```
// 设置选择范围  
EjunDataGrid1.SelectRangle(2, 2, 5, 5);  
// 设置居中对齐  
EjunDataGrid1.Selection.SetHorzAlign(haCenter);
```

设置竖直对齐方式 Selection 对象提供了 [SetVertAlign](#) 方法

4.13.4.2.19 SetVertAlign 函数

设置单元格竖直对齐方式。

Delphi 语法

```
procedure SetVertAlign(const Value: TZcGridVertAlign);
```

描述

统一设置选择框范围内的所有单元格的竖直对齐方式。

该方法实际上是批量设置选择框范围内所有单元格的 Style.VertAlign 属性。

TZcGridVertAlign 枚举变量声明在 TZcGridStyle 单元，所以使用时要记得引用 TZcGridStyle 单元。

关于 TZcGridVertAlign 类型的详细介绍请参见 [HorzAlign](#)。

```
// 设置选择范围  
EjunDataGrid1.SelectRangle(2, 2, 5, 5);  
// 设置底端对齐  
EjunDataGrid1.Selection.SetVertAlign(vaBottom);
```

设置水平对齐方式 Selection 对象提供了 [SetVertAlign](#) 方法

4.13.4.2.20 SetFormatString 函数

设置控制数字显示格式的格式字符串。

Delphi 语法

```
procedure SetFormatString(const Value: string);
```

描述

该方法实际上是批量设置选择框范围内所有单元格的 [Style.FormatString](#) 属性。

请参见 [FormatString 属性](#) 来详细了解格式字符串的用法

```
// 设置选择范围  
EjunDataGrid1.SelectRange(2, 2, 5, 5);  
// 设置格式字符串  
EjunDataGrid1.Selection.FormatString('#,###.00');
```

4.13.4.2.21 SetNotPrint 函数

设置单元格是否打印，主要用来实现套打

Delphi 语法

```
procedure SetNotPrint(const Value: Boolean);
```

描述

EjunGrid 可以用来控制某些单元格打印，某些单元格不打印，这样可以实现票据的套打。

关于套打的详细介绍请参见 [NotPrint](#) 属性

4.13.4.2.22 函数

Delphi 语法

描述

4.13.4.3 Events(事件)

4.13.5 IZcCellRange

4.13.5.1 Properties(属性)

4.13.5.1.1 RefCount 属性

todo

Delphi 语法

```
property RefCount: Integer;
```

描述

todo.

4.13.5.2 Functions(函数)

4.13.5.2.1 AfterConstruction 函数

Delphi 语法

```
procedure AfterConstruction; override;
```

描述

4.13.5.2.2 BeforeDestruction 函数

Delphi 语法

```
procedure BeforeDestruction; override;
```

描述

4.13.6 TZcGridSelection

每个表格都会有一个 Selection 对象, 表格所选区域类, 对应的表现就是表格的选择框, 继承于 [TZcGridRange](#).

单元

ZjGrid

描述

每个表格都会有一个 Selection 对象, 对应的表现就是表格的选择框, 每次改变选区, Selection 对象内部会自动调整, 对应用户的所选. Selection 对象提供了一系列属性函数接口, 可以对选区进行设置字体, 单元格边框线, 单元格数字显示, 背景色, 文本对齐方式, 合并单元格, 拆分单元格等功能

4.13.6.1 Properties(属性)

4.13.6.1.1 Height 属性

todo

Delphi 语法

```
property Height: Integer;
```

描述

todo.

4.13.6.1.2 Current 属性

读取或者设置表格的当前单元格的行列坐标值

Delphi 语法

```
property Current: TPoint;
```

描述

指定 `Current` 属性也就改变了选择框的位置，选择框是个矩形区域，由两个对角坐标指定，`Current` 属性指定选择框的起点行列坐标，`EndSel` 指定单元格的结束行列坐标。起点行列坐标所在的单元格成为当前单元格。当我们用鼠标拖动框选一个选择范围时，鼠标左键点下的位置就是 `Current`，然后按住左键拖动选择，选择完毕，释放鼠标左键时，鼠标指针所在的位置就是 `EndSel`;

`EJunGrid` 会给一些常用的属性提供简便的访问途径，例如下面一组属性：

`EJunGrid.CurRow` 属性等于 `EJunGrid.Selection.Current.Y`;

`EJunGrid.CurCol` 属性等于 `EJunGrid.Selection.Current.X`;

`EJunGrid.CurCell` 属性返回值返回 `Current` 所在的单元格对象。

4.13.6.1.3 EndSel 属性

设置或者获取选择框的终止坐标，可读写属性

Delphi 语法

```
property EndSel: TPoint;
```

描述

`EJunGrid` 的选择框可以包含多个连续单元格，或者说选择的不只是单个单元格，而是一块矩形区域，那我们如何来定义这个矩形区域呢，`EJunGrid` 采用了两个属性 `Current` 和 `EndSel` 来表示，它们都是 `TPoint` 类型，代表表格的行列坐标，`Current` 表示选择框的开始坐标，`EndSel` 表示选择框的终止坐标，值得一提的是，开始坐标并不一定是矩形区域的左上角，终止坐标也不一定是矩形的右下角，而可以是任何一个角。

当我们用鼠标做框选动作时，鼠标左键点下的位置就是 `Current`，然后按住左键拖动选择，选择完毕，释放鼠标左键时，鼠标指针所在的位置就是 `EndSel`；大家可以注意到，在框选的过程中 `Current` 是不变的，`EndSel` 随鼠标的移动而改变。

开发人员也可以通过 `Selection` 对象的 `Move` 方法来模拟键盘方向键改变选择框位置的动作。

4.13.6.1.4 OldCurrent 属性

返回 Current 变化之前的值，只读属性。

Delphi 语法

```
property OldCurrent: TPoint;
```

描述

当我们改变了选择框的 Current 属性时，EjunGrid 会把旧的 [Current](#) 属性值保存到 OldCurrent 属性中。

```
procedure TForm1.EjunDataGrid1CurrentChanged(Sender: TObject; Col,
  Row: Integer);
begin
  StatusBar1.Panels[0].Text := Format('OldCurrent:%d,%d',
    [EjunDataGrid1.Selection.OldCurrent.X,
    EjunDataGrid1.Selection.OldCurrent.Y]);
end;
```

4.13.6.1.5 BgColor 属性

设置或获取所选区域的背景色

Delphi 语法

```
property BgColor: TColor;
```

描述

设置或得到所选区域的背景色

4.13.6.1.6 Font 属性

设置或获取所选区域的字体设置

Delphi 语法

```
property Font: TFont;
```

描述

设置或获取所选区域的字体设置,通过 Font 对象,可以设置字体的颜色,形状,大小等

4.13.6.1.7 Visible 属性

设置是否显示选择框，可读写属性。

Delphi 语法

```
property Visible: Boolean;
```

描述

设置是否显示选择框，可读写属性。

我们可以通过这个属性来模拟 Excel 的效果，当 Excel 失去焦点时选择框消失，当获得焦点时选择框又显示出来。

```
procedure TForm1.EjunDataGrid1Exit(Sender: TObject);
begin
  EjunDataGrid1.Selection.Visible := False;
end;

procedure TForm1.EjunDataGrid1Enter(Sender: TObject);
begin
  EjunDataGrid1.Selection.Visible := True;
end;
```

该属性是 **Published** 属性，可以在设计期通过属性编辑器设置

4.13.6.1.8 State 属性

返回当前选择区域的选择状态: 全选、行选、列选、普通。

Delphi 语法

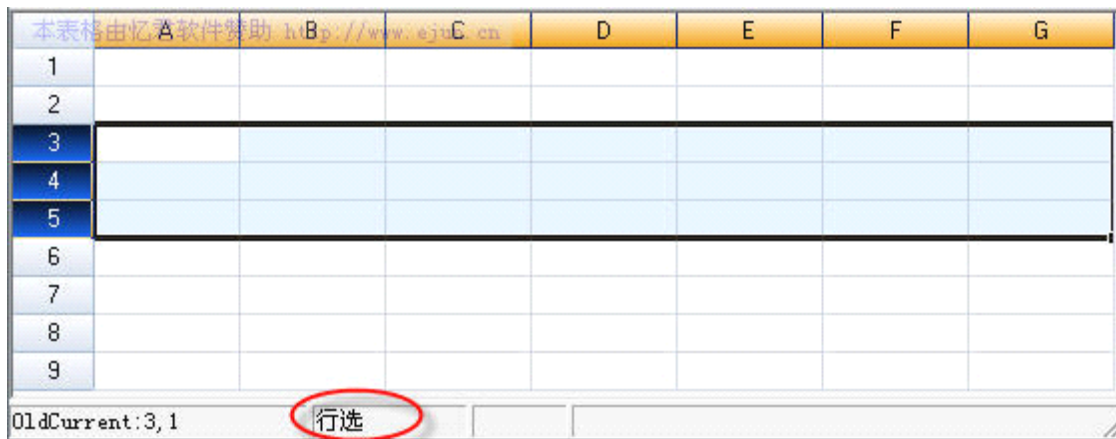
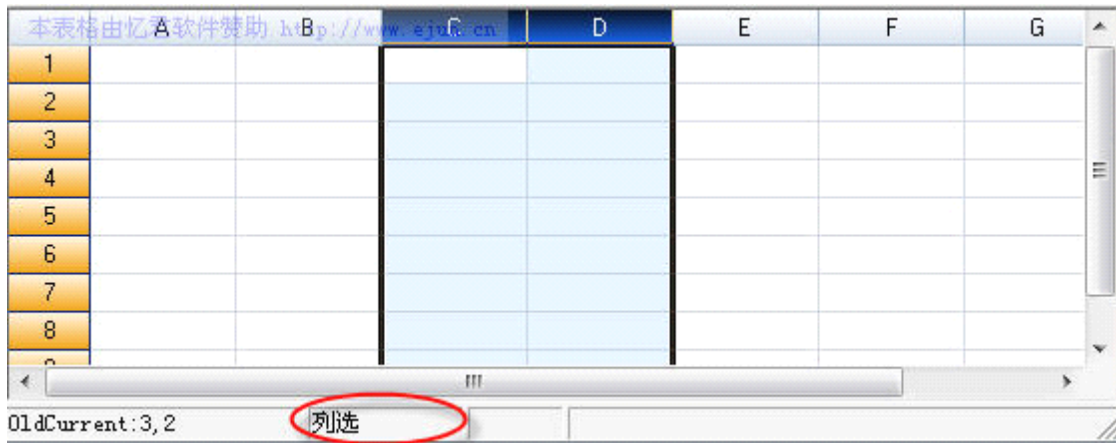
```
property State: TSelectionMode;
```

描述

返回当前选择区域的选择状态: 全选(ssAll)、行选(ssFullRow)、列选(ssFullCol)、普通(ssNormal)。 只读属性

```
procedure TForm1.EjunDataGrid1SelectionChange(Sender: TObject;
  const ARange: TRect);
begin
  // 显示选择框状态
  case EjunDataGrid1.Selection.State of
    ssNormal: StatusBar1.Panels[1].Text := '普通选择';
    ssFullCol: StatusBar1.Panels[1].Text := '列选';
    ssFullRow: StatusBar1.Panels[1].Text := '行选';
    ssAll:     StatusBar1.Panels[1].Text := '全选';
```

```
end;  
end;
```



4.13.6.1.9 Selecting 属性

todo

Delphi 语法

property Selecting: Boolean;

描述

todo.

4.13.6.1.10 Merged 属性

获取选择区域是否被合并过

Delphi 语法

```
property Merged: Boolean;
```

描述

获取选择区域是否被合并过。

4.13.6.1.11 FontColor 属性

设置或获取选择区域的单元格显示文本颜色

Delphi 语法

```
property FontColor: TColor;
```

描述

设置或获取选择区域的单元格显示文本颜色。

4.13.6.1.12 FontSize 属性

设置或获取选择区域的单元格显示文本尺寸

Delphi 语法

```
property FontSize: Integer;
```

描述

设置或获取选择区域的单元格显示文本尺寸。

4.13.6.1.13 FontName 属性

设置或获取选择区域的单元格显示文本的字体类型

Delphi 语法

```
property FontName: TFontName;
```

描述

设置或获取选择区域的单元格显示文本的字体类型。

4.13.6.1.14 **FontStyle** 属性

设置或获取选择区域的单元格显示文本的样式

Delphi 语法

```
property FontStyle: TFontStyles;
```

描述

设置或获取选择区域的单元格显示文本的样式。

4.13.6.1.15 **HorzAlign** 属性

设置或获取选择区域的单元格显示文本水平对齐方式。

Delphi 语法

```
property HorzAlign: TGridHorzAlign;
```

描述

设置或获取选择区域的单元格显示文本水平对齐方式。

4.13.6.1.16 **VertAlign** 属性

设置或获取选择区域的单元格显示文本垂直对齐方式。

Delphi 语法

```
property VertAlign: TGridVertAlign;
```

描述

设置或获取选择区域的单元格显示文本垂直对齐方式。

4.13.6.1.17 **SelectedRowCount** 属性

返回所选区域中行数

Delphi 语法

property SelectedRowCount: Integer;

描述

返回所选区域中行数.配合 [SelectedRows](#) 可以得到指定对应行的行索引

4.13.6.1.18 SelectedRows 属性

返回所选区域中行对应的行索引

Delphi 语法

property SelectedRows[AIndex: Integer]: Integer;

描述

返回所选区域中行对应的行索引, AIndex 参数为区域所选行数的对应的索引.

$0 < AIndex < \text{SelectedRowCount}$

4.13.6.1.19 RangeCount 属性

返回选择区域的个数

Delphi 语法

property RangeCount: Integer;

描述

返回选择区域的个数.EjuiGrid 允许同时选中多个区域.配合 [Ranges](#) 属性, 可以遍历选中的第一个区域.

当设置了支持多选区功能, 用户按着[Ctrl].键, 就可以选择多个区域了

4.13.6.1.20 Ranges 属性

返回选中区域中指定索引的区域对象

Delphi 语法

property Ranges[AIndex: Integer]: TZcGridRange;

描述

返回选中区域中指定索引的区域对象.

索引参数值: $0 < AIndex < \text{RangeCount}$

4.13.6.1.21 FlashSelectRange 属性

todo

Delphi 语法

```
property FlashSelectRange: TRect;
```

描述

todo.

4.13.6.1.22 FlashSelect 属性

todo

Delphi 语法

```
property FlashSelect: Boolean;
```

描述

todo.

4.13.6.1.23 AlphaBlend 属性

todo

Delphi 语法

```
property AlphaBlend: Boolean;
```

描述

todo.

4.13.6.1.24 AlphaBlendValue 属性

todo

Delphi 语法

```
property AlphaBlendValue: Byte;
```

描述

todo.

4.13.6.1.25 TransparentColor 属性

todo

Delphi 语法

```
property TransparentColor: Boolean;
```

描述

todo.

4.13.6.1.26 TransparentColorValue 属性

设置或获取透明颜色值

Delphi 语法

```
property TransparentColorValue: TColor;
```

描述

设置或获取透明颜色值.

4.13.6.1.27 DisableDrag 属性

可读写属性，设置是否禁用拖动选择框的边框

Delphi 语法

```
property DisableDrag: Boolean;
```

描述

EJunGrid 可以设置是否禁用拖动选择框的边框，默认是不禁用选择边框.

拖拽选择框的边框来执行[复制剪切](#)，[自动填充](#)等功能是 EJunGrid 的一大特色。.

4.13.6.1.28 HideBorder 属性

设置是否隐藏边框线

Delphi 语法

```
property HideBorder: Boolean;
```

描述

选择区域的时候设置是否隐藏边框线,默认值为 False, 即不隐藏边框线.

4.13.6.1.29 CurCol 属性

todo

Delphi 语法

```
property CurCol: Integer;
```

描述

todo.

4.13.6.1.30 CurRow 属性

todo

Delphi 语法

```
property CurRow: Integer;
```

描述

todo.

4.13.6.1.31 OldCurCol 属性

todo

Delphi 语法

```
property OldCurCol: Integer;
```

描述

todo.

4.13.6.1.32 OldCurRow 属性

todo

Delphi 语法

```
property OldCurRow: Integer;
```

描述

todo.

4.13.6.2 Functions(函数)

4.13.6.2.1 Move 函数

调用该方法用来移动选择框。

Delphi 语法

```
procedure Move(ADir: TGridDir; IsAnchor: Boolean = False);
```

描述

移动选择框,可以指定方向进行移动.

参数说明:

ADir: 指明移动的方向, [TGridDir](#) 枚举类型, 具体定义如下: TGridDir = (gdAuto, gdLeft, gdRight, gdUp, gdDown, gdNone);

- gdAuto: 具体的移动方向依赖于表格的 [CurrentMoveDir](#) 属性中指定的移动方向。
- gdLeft: 向左移动一列
- gdRight: 向右移动一列
- gdUp: 向上移动一行
- gdDown: 向下移动一行
- gdNone: 不移动

IsAnchor:Boolean 类型: 指定选择框移动模式

选择框移动有两种模式。

- 1 参数 IsAnchor=False 时, 选择框是整个移动, 这时的选择框只会选择一个单元格, 即 [Current](#) 和 EndSel 是同时移动且相同, 这是模拟用键盘方向键和回车移动选择框时的效果;

- 2 参数 `IsAnchor=True` 时，是框选模式，在这种模式下选择框的 [Current](#) 不会改变，移动的是 [EndSel](#)，模拟同时按住 `Shift` 键和方向键移动选择框时的效果；

实际上 **EjunGrid** 内部正式调用这个方法实现选择框的移动的。当用户按回车移动选择框时，**EjunGrid** 内部是这样调用的 `Selection.Move(gdAuto,False)`

4.13.6.2.2 Invalidate 函数

通知表格刷新选择框所在区域的屏幕

Delphi 语法

```
procedure Invalidate; override;
```

描述

调用该方法会促使 `EjunGrid` 刷新屏幕，但只会刷新选择框所在区域内的屏幕

4.13.6.2.3 Merge 函数

合并所选区域

Delphi 语法

```
procedure Merge;
```

描述

合并所选区域，把选择框选中范围内的单元格合并成一个单元格，如果选择多个区域,将对每个区域进单独合并

横向合并请调用 [HorzMerge](#) 函数，竖向合并请调用 [VertMerge](#) 函数，取消选择框范围内的所有的合并单元格，请调用 [Split](#) 函数

4.13.6.2.4 HorzMerge 函数

将选择框范围内的单元格横向合并

Delphi 语法

```
procedure HorzMerge;
```

描述

水平合并所选区域,即把所选区域每一行的单元格合并为一个单元格。

```
// 设置选择框范围  
EjunDataGrid1.SelectRangle(2, 2, 4, 4);  
// 横向和并单元格  
EjunDataGrid1.Selection.HorzMerge;
```

竖向合并请调用 [VertMerge](#) 函数, 合并所选区域单元格请调用 [Merge](#) 函数, 取消选择框范围内的所有的合并单元格, 请调用 [Split](#) 函数。

4.13.6.2.5 VertMerge 函数

竖向合并选择框范围内的单元格

Delphi 语法

```
procedure VertMerge;
```

描述

竖向合并选择框范围内的单元格,即把所选区域每一列的单元格合并为一个单元格

横向合并请调用 [HorzMerge](#), 合并所选区域单元格请调用 [Merge](#) 函数, 取消选择框范围内的所有的合并单元格, 请调用 [Split](#) 函数。

4.13.6.2.6 Split 函数

取消选择框范围内的所有的合并单元格

Delphi 语法

```
procedure Split;
```

描述

取消选择框范围内的所有的合并单元格, 合并操作的反操作. 如果所选单元格没有被合并, 则保持原状.

合并所选区域单元格请调用 [Merge](#) 函数, 横向合并请调用 [HorzMerge](#) 函数, 竖向合并请调用 [VertMerge](#) 函数.

4.13.6.2.7 SelectRow 函数

选择指定行间的区域, 整行选择

Delphi 语法

```
procedure SelectRow(AFrom, ATo: Integer);
```

描述

选择从 AFrom 到 ATo 的行, 整行选择

参数介绍:

AForm:Integer 类型,指定选中区域的开始行索引

ATo:Integer 类型,指定选中区域的结束行索引

```
// 整行选择演示  
procedure TForm1.tbSelectRowClick(Sender: TObject);  
begin  
    // 选择第 2 行到第 4 行  
    EjunDataGrid1.Selection.SelectRow(2, 4);  
end;
```

选择列使用 [SelectCol](#) 方法, 选择部分区域使用 [SelectionRange](#) 方法

4.13.6.2.8 SelectCol 函数

选择指定列间的区域,整列选择

Delphi 语法

```
procedure SelectCol(AFrom, ATo: Integer);
```

描述

选择从 AFrom 到 ATo 的列, 整列选择

参数介绍:

AForm:Integer 类型,指定选中区域的开始列索引

ATo:Integer 类型,指定选中区域的结束列索引

```
// 整列选择演示
```

```
procedure TForm1.tbSelectColumnClick(Sender: TObject);
begin
    // 选择第 2 行到第 4 行
    EjunDataGrid1.Selection.SelectCol(2, 4);
end;
```

选择行使用 [SelectRow](#) 方法，选择部分区域使用 [SelectionRange](#) 方法

4.13.6.2.9 Select 函数

选择指定区域

Delphi 语法

```
procedure Select(ALeft, ATop, ARight, ABottom: Integer);
```

描述

选择指定区域

4.13.6.2.10 SelectAll 函数

全选表格

Delphi 语法

```
procedure SelectAll;
```

描述

全选的范围是指整个客户区表格，因为 EjunGrid 的行序号是包括固定行的，所以客户区的第一行的序号等于固定行的行数，EjunGrid 的 [FixedRowCount](#) 属性表示固定行的行数，同样列的序号也是从固定列的第一列算起的，固定列的第一列序号为0，客户区的第一列序号等于 EjunGrid.[FixedColCount](#) 固定列的列数。

全选的范围可以用 `Rect(Grid.FixedColCount, Grid.FixedRowCount, Grid.ColCount-1, Grid.RowCount-1)` 来表示。

```
EjunDataGrid1.Selection.SelectAll;
// 也可以用下面的方法实现全选，效果是一样的
EjunDataGrid1.SelectRangle(EjunDataGrid1.FixedColCount,
                           EjunDataGrid1.FixedRowCount,
                           EjunDataGrid1.ColCount - 1,
```

```
EjunDataGrid1.RowCount - 1);
```

在运行期用户点击固定行和固定列交叉地方的单元格时，EjunGrid 会在内部调用这个方法实现全选。

4.13.6.2.11 Clear 函数

清除选择框范围内单元格中的内容。

Delphi 语法

```
procedure Clear(AEditMode: Boolean = False);
```

描述

该方法用来清除选择框范围内所有单元格的内容，当我们改变单元格中的内容时，EjunGrid 提供了两种模式，一种是编辑模式，一种是代码模式。编辑模式是指用户在运行期操作表格，例如敲键盘在单元格中输入字符，或者按 Delete 键删除单元格中的内容。代码模式是指程序员通过代码改变单元格的内容，例如 EjunGrid.Cells[1,1].AsString := 'abc'。

区分编辑模式和代码模式可以给程序员灵活的控制，编辑模式改变单元格内容时会触发 [OnSetCellText](#) 事件、[OnCellTextChangingW](#) 事件、[OnCellTextChanged](#) 事件和 [OnCellValueChanged](#) 事件，而代码模式只会触发 [OnCellValueChanged](#) 事件。

运行期当用户编辑单元格内容时，EjunGrid 内部是通过 Cell 对象的 EditText 属性给单元格赋值的，值的类型是字符串型，这属于编辑模式，EjunGrid 会分析字符串，判断字符串是数值型、还是布尔型、或者是日期型，并由此调用对应的属性 [AsString](#)、[AsInteger](#)、[AsFloat](#)、[AsBoolean](#)、[AsDateTime](#) 来给单元格赋值。由此看来，通过 [EditText](#) 属性给单元格赋值没有直接通过 [AsString](#)、[AsInteger](#)、[AsFloat](#)、[AsBoolean](#)、[AsDateTime](#) 属性赋值效率高。

参数 AEditMode=True 表示编辑模式，当用户在运行期按 Delete 键时，EjunGrid 内部会这样调用该方法：

```
EjunGrid.Selection.Clear(True);
```

编辑模式调用 Clear 方法可以支持撤销操作。

4.13.6.2.12 Cut 函数

剪切

Delphi 语法

```
procedure Cut;
```

描述

把选择框范围内单元格的内容复制到剪贴板上，当调用 [Paste](#) 方法时，EjunGrid 把剪贴板上的内容复制到目标单元格中，并清除源单元格中的内容。[Copy](#) 方法和 [Cut](#) 方法的区别就是 [Copy](#) 方法不会清除源单元格的内容。

当用户在运行期按 Ctrl+X 时，EjunGrid 内部调用下面方法：

```
EjunGrid.Selection.Cut;
```

按 Ctrl+V 时，EjunGrid 内部调用下面方法执行粘贴操作，把源单元格中的内容复制到目标单元格中。

```
EjunGrid.Selection.Paste
```

4.13.6.2.13 Copy 函数

复制

Delphi 语法

```
procedure Copy;
```

描述

将选择框范围内的单元格内容放到剪贴板上。

4.13.6.2.14 Paste 函数

粘贴

Delphi 语法

```
procedure Paste(AMode: TZcGridPasteMode = pmPasteNormal);
```

描述

把剪贴板中的内容复制到选择框所在的目标区域。

4.13.6.2.15 CorrectRange 函数

Delphi 语法

```
procedure CorrectRange;
```

描述

4.13.6.2.16 InsertRow 函数

Delphi 语法

```
procedure InsertRow(ACount: Integer);
```

描述

4.13.6.2.17 InsertCol 函数

Delphi 语法

```
procedure InsertCol(ACount: Integer);
```

描述

4.13.6.2.18 DeleteRow 函数

删除选择框所在的所有行

Delphi 语法

```
procedure DeleteRow;
```

描述

删除选择框所在的所有行，即从 Selection.Top 到 Selection.Bottom 之间的所有行。

该方法内部调用 TEJunGrid 对象的 [DeleteRow](#) 方法。

4.13.6.2.19 DeleteCol 函数

删除选择框所在的所有列

Delphi 语法

```
procedure DeleteCol;
```

描述

删除选择框所在的所有列，即从 Selection.Left 到 Selection.Right 之间的所有列。

该方法内部调用 TEJunGrid 对象的 DeleteCol 方法

4.13.6.2.20 AddRange 函数

添加区域到选择框

Delphi 语法

```
procedure AddRange(const ARange: TRect);
```

描述

添加区域到选择框

4.13.6.2.21 CancelMode 函数

Delphi 语法

```
procedure CancelMode;
```

描述

4.13.6.2.22 ValidSelectedRows 函数

Delphi 语法

```
procedure ValidSelectedRows;
```

描述

4.13.6.2.23 AdjustRowHeightAuto 函数

自动调整所选区域的行高

Delphi 语法

```
procedure AdjustRowHeightAuto;
```

描述

自动调整所选区域的行高

4.13.6.2.24 AdjustColWidthAuto 函数

自动调整所选区域的列宽

Delphi 语法

```
procedure AdjustColWidthAuto;
```

描述

自动调整所选区域的列宽

4.13.6.2.25 ForAll 函数

Delphi 语法

```
procedure ForAll(Proc: Pointer); override;
```

描述

4.13.6.2.26 InsertCopy 函数

Delphi 语法

```
procedure InsertCopy(AMode: TZcGridPasteMode = pmPasteNormal);
```

描述

4.13.6.2.27 函数

Delphi 语法

描述

4.13.6.2.28 函数

Delphi 语法

描述

4.13.6.2.29 函数

Delphi 语法

描述

4.13.6.2.30 函数

Delphi 语法

描述

4.13.6.2.31 函数

Delphi 语法

描述

4.13.6.2.32 函数

Delphi 语法

描述

4.13.6.3 Events(事件)

4.13.7 TZcGridStyle

GridStyle 维护了单元格的格式信息，包括单元格背景颜色、对齐方式、边框颜色、边框线性、换行显示、锁定、打印等

单元

ZcGridStyle

4.13.7.1 Properties(属性)

4.13.7.1.1 BorderStyle 属性

todo

Delphi 语法

```
property BorderStyle[ASide: TZcGridBorderSide]: TZcGridBorderStyle;
```

描述

todo.

4.13.7.1.2 BorderColor 属性

设置单元格的边框颜色

Delphi 语法

```
property BorderColor[ASide: TZcGridBorderSide]: TColor;
```

描述

设置单元格的边框颜色, 可读写属性. 该属性是数组属性, ASide 是 [TZcGridBorderSide](#) 类型的枚举值, 定义了单元格的每一个上下左右边框以及单元格内部的斜线

4.13.7.1.3 IsDefault 属性

todo

Delphi 语法

```
property IsDefault: Boolean;
```

描述

todo.

4.13.7.1.4 UseAttributes 属性

todo

Delphi 语法

```
property UseAttributes: TZcStyleUsedAttribs;
```

描述

todo.

4.13.7.1.5 IsDateTimeFormat 属性

判断文本是否为日期时间数据格式

Delphi 语法

```
property IsDateTimeFormat: Boolean;
```

描述

判断文本是否为日期时间数据格式.判断的根据是看 [FormatString](#) 属性是否是日期时间数据字符串格式.

4.13.7.1.6 IsScientificFormat 属性

判断文本是否为科学记数数据格式

Delphi 语法

```
property IsScientificFormat: Boolean;
```

描述

判断文本是否为科学记数数据格式,判断的根据是看 [FormatString](#) 属性是否是科学记数字符串格式.

4.13.7.1.7 IsThousandFormat 属性

判断文本是千分位数据格式

Delphi 语法

```
property IsThousandFormat: Boolean;
```

描述

判断文本是千分位数据格式,判断的根据是看 [FormatString](#) 属性是否是千分位数据字符串格式

4.13.7.1.8 IsPercentFormat 属性

判断文本是否为百分比数据格式

Delphi 语法

property **IsPercentFormat**: Boolean;

描述

判断文本是否为百分比数据格式.判断的根据是看 [FormatString](#) 属性是否是百分比数据字符串格式

4.13.7.1.9 Pattern 属性

todo

Delphi 语法

property **Pattern**: Integer;

描述

todo.

4.13.7.1.10 WrapText 属性

设置或获取当单元格中的文本长度超出单元格宽度时, 是否自动折行显示

Delphi 语法

property **WrapText**: Boolean;

描述

当单元格中的文本长度超出单元格宽度时自动折行显示.

此属性与操作集合 [Options](#) 属性中的 [gcoWrapText](#) 是同样的效果

4.13.7.1.11 Locked 属性

设置或获取单元格是否锁定

Delphi 语法

property **Locked**: Boolean;

描述

设置或获取单元格是否锁定, 此属性与操作集合 [Options](#) 属性中的 [gcoLocked](#) 是同样的效果

4.13.7.1.12 Hidden 属性

设置或获取是否隐藏单元格中的内容

Delphi 语法

```
property Hidden: Boolean;
```

描述

设置或获取是否隐藏单元格中的内容.此属性与操作集合 [Options](#) 属性中的 [gcoHidden](#) 是同样的效果

4.13.7.1.13 Indent 属性

todo

Delphi 语法

```
property Indent: Byte;
```

描述

todo.

4.13.7.1.14 NotPrint 属性

设置或获取是否不打印单元格中的内容

Delphi 语法

```
property NotPrint: Boolean;
```

描述

设置或获取是否不打印单元格中的内容.不打印单元格中的内容，主要是实现套打功能.

此属性与操作集合 [Options](#) 属性中的 [gcoNotPrint](#) 是同样的效果

4.13.7.1.15 ShrinkToFit 属性

设置或获取是否自动调整单元格中文本的字体以适应单元格的宽度

Delphi 语法

```
property ShrinkToFit: Boolean;
```

描述

自动调整单元格中文本的字体以适应单元格的宽度(目前没有实现这个功能).

此属性与操作集合 [Options](#) 属性中的 [gcoShrinkToFit](#) 是同样的效果

4.13.7.1.16 BgColor 属性

设置单元格背景颜色

Delphi 语法

```
property BgColor: TColor;
```

描述

设置单元格背景颜色.

4.13.7.1.17 HorzAlign 属性

获取或设置单元格文本的水平对齐方式

Delphi 语法

```
property HorzAlign: TZcGridHorzAlign;
```

描述

获取或设置单元格文本的水平对齐方式, 参阅 [TZcGridHorzAlign](#) 设置水平对方式

4.13.7.1.18 VertAlign 属性

获取或设置单元格中文本的垂直对齐方式

Delphi 语法

```
property VertAlign: TZcGridVertAlign;
```

描述

获取或设置单元格中文本的垂直对齐方式. 参阅 [TZcGridVertAlign](#) 设置水平对方式

4.13.7.1.19 Options 属性

设置单元格的一些属性选项

Delphi 语法

```
property Options: TZcCellOptions;
```

描述

设置单元格的一些属性选项，例如是否锁定单元格、是否隐藏单元格文本、是否打印单元格、文本自动换行等。

[TZcCellOptions](#) 是一个集合属性，可以组合以下选项：

- `gcoLocked` 锁定单元格，设置单元格为只读的
- `gcoHidden` 隐藏单元格中的内容
- `gcoWrapText` 当单元格中的文本长度超出单元格宽度时自动折行显示
- `gcoShrinkToFit` 自动调整单元格中文本的字体以适应单元格的宽度(目前没有实现这个功能)
- `gcoNotPrint` 不打印单元格中的内容，主要是实现套打功能

```
// 设置单元格中的文本自动换行  
EjunGrid.Cells[x, y].Style.Options := EjunGrid.Cells[x, y].Style.Options +  
[gcoWrapText];
```

为了方便使用这些集合选项，`EjunGrid` 也提供了一些快捷属性。参阅 [Locked](#)、[Hidden](#)、[NotPrint](#)、[ShrinkToFit](#)、[WrapText](#) 属性。

4.13.7.1.20 FormatIndex 属性

设置对数字的格式化方式，**过时的属性**，新版本请使用 `FormatString` 属性

Delphi 语法

```
property FormatIndex: Word;
```

描述

控制数字类型的单元格显示的格式。

4.13.7.1.21 FormatString 属性

设置单元格中数字的格式字符串

Delphi 语法

```
property FormatString: WideString;
```

描述

EjJunGrid 使用格式化字符串来控制数字的显示格式，例如“#.00”可以让数字“123”显示成“123.00”，即显示两位小数，而格式串“#,###.000”可以让浮点数“1234.1”显示成“1,234.100”，即显示千分位符，且显示3位小数。

忆君表格控件还可以控制日期时间的显示格式，例如“yyyy"年"mm"月"dd"日”可以让数值1显示成“1899年12月31日”，表格中列出了常用格式串即其作用效果，可供大家参考：

FormatString	效果	说明
空串	不格式化	
0	123.5 => 124	取整数
0.00	123.5 => 123.50	保留两位小数
#,###0	1234 => 1,234	显示千分符
#,###0.00	1234.5 => 1,234.50	显示千分符并保留两位小数
"¥"#,###0	1234.5 => ¥1,235	显示货币符号，显示千分符，四舍五入取整
"¥"#,###0.00	1234.5 => ¥1,234.50	显示货币符号，显示千分符，保留两位小数
0%	0.12 => 12%	显示百分数形式
0.00%	0.1234 => 12.34%	显示百分数形式,保留两位小数
0.00E+00	0.012345 => 1.23E-02	显示科学记数的形式
yyyy"年"m"月"	2008 年 8 月	显示年月
yyyy"年"mm"月"	2008 年 08 月	显示年月，月份用两位数字显示
yy"年"mm"月"dd"日"	08 年 08 月 01 日	显示年月日，年份用缩略的形式，只显示两位
m"月"d"日"	8 月 1 日	显示月日
yyyy-m-d	2008-8-1	
h:mm AM/PM	8:08 PM	
h:mm:ss AM/PM	8:08:08 PM	
h"时"mm"分"	20 时 08 分	
h"时"mm"分"ss"秒"	20 时 08 分 08 秒	

上午/下午 h"时"mm"分" 下午 8 时 01 分
 上午/下午 h"时"mm"分
 "ss"秒" 下午 8 时 08 分 08 秒
 yyyy-m-d h:mm 2008-1-1 20:08

```
EjunGrid1.Cells[2, 1].AsFloat := 3;
EjunGrid1.Cells[2, 2].Style.FormatString := '#.00'; // 设置一个单元格的数字格式
EjunGrid.Columns[2].Style.FromatString := '#.00'; // 设置整列单元格的数字格式
EjunGrid.Rows[2].Stye.FormatString := '#.00'; // 设置正行单元格的数字格式
```

4.13.7.2 Functions(函数)

4.13.7.2.1 Clear 函数

Delphi 语法

```
procedure Clear;
```

描述

4.13.7.2.2 FormatNumber 函数

Delphi 语法

```
function FormatNumber(const AValue: Double; var AColor: TColor): EjunString;
    overload;
function FormatNumber(const AValue: Double): EjunString; overload;
```

描述

4.13.7.2.3 SetBorder 函数

设置边框线

Delphi 语法

```
procedure SetBorder(ASides: TZcGridBorderSides; AStyle: TZcGridBorderStyle;
  AColor: TColor); overload;
procedure SetBorder(ASide: TZcGridBorderSide; AStyle: TZcGridBorderStyle;
  AColor: TColor); overload;
procedure SetBorder(ASides: TZcGridBorderSides; AStyle: TZcGridBorderStyle);
  overload;
procedure SetBorder(ASides: TZcGridBorderSides; AColor: TColor); overload;
```

描述

4.13.7.2.4 IncludeOptions 函数

添加单元格的一些 Options 属性选项,如锁定单元格, 自动换行等

Delphi 语法

```
procedure IncludeOptions(const AOptions: TZcCellOptions); overload;
procedure IncludeOptions(const AOption: TZcCellOption); overload;
```

描述

添加单元格的一些 [Options](#) 属性选项,如锁定单元格, 自动换行等.实际上就是 [Options](#) 集合属性的添加操作.

EjunGrid 重载了二个方法进行 [Options](#) 属性添加操作.

```
procedure IncludeOptions(const AOptions: TZcCellOptions); overload;
```

参数为 [TZcCellOptions](#) 集合类型,可以一次性增加几个属性到 [Options](#) 集合属性中

```
procedure IncludeOptions(const AOption: TZcCellOption); overload;
```

参数为 [TZcCellOption](#) 类型,可以一次添加一个属性选项到 [Options](#) 集合属性中

4.13.7.2.5 ExcludeOptions 函数

移出单元格的一些 Options 属性选项,如锁定单元格, 自动换行等

Delphi 语法

```
procedure ExcludeOptions(const AOptions: TZcCellOptions); overload;
procedure ExcludeOptions(const AOption: TZcCellOption); overload;
```

描述

移除单元格的一些 Options 属性选项,如锁定单元格,自动换行等.实际上就是 Options 集合移除操作.

EjuncGrid 重载了二个方法进行 Options 属性移除操作.

```
procedure ExcludeOptions(const AOptions: TZcCellOptions); overload;
```

参数为 [TZcCellOptions](#) 集合类型,可以一次性移除 [Options](#) 集合属性中的几个属性

```
procedure ExcludeOptions(const AOption: TZcCellOption); overload;
```

参数为 [TZcCellOption](#) 类型,可以一次移除 [Options](#) 集合属性中的一个属性

4.13.7.3 Events(事件)

4.13.8 TZcTree

4.13.8.1 Properties(属性)

4.13.8.2 Functions(函数)

4.13.8.2.1 函数

Delphi 语法

描述

4.13.8.2.2 函数

Delphi 语法

描述

4.13.8.3 Events(事件)

4.13.9 TZcTreeNode

4.13.9.1 Properties(属性)

4.13.9.2 Functions(函数)

4.13.9.2.1 函数

Delphi 语法

描述

4.13.9.2.2 函数

Delphi 语法

描述

4.13.9.3 Events(事件)

4.13.10 TZcGridExcelRW

4.13.10.1 Properties(属性)

4.13.10.2 Functions(函数)

4.13.10.2.1 函数

Delphi 语法

描述

4.13.10.2.2 函数

Delphi 语法

描述

4.13.10.3 Events(事件)

5 应用文章