Worksheet

Description

The `Worksheet` class represents a single Excel worksheet.

C#

```
[DefaultMember("Item")]
 public class Worksheet
```

vb.net

```
<DefaultMember("Item")> _
 Public Class Worksheet
```

Remarks

To add a Worksheet to a Workbook, call Workbook.Worksheets.CreateWorksheet(). To get an existing Worksheet call Workbook.Worksheets[index or name].

Examples

C#

```
//--- Create a new Worksheet
          ExcelApplication xla = new ExcelApplication();
          Workbook wb = xla.Create();
          Worksheet ws = wb.Worksheets.CreateWorksheet("Sheet2");

          //--- Get an existing Worksheet by index
          ExcelApplication xla = new ExcelApplication();
          Workbook wb = xla.Create();
          Worksheet ws = wb.Worksheets[0];

          //--- Get an existing Worksheet by name
          ExcelApplication xla = new ExcelApplication();
          Workbook wb = xla.Create();
          Worksheet ws = wb.Worksheets["Sheet1"];
```

vb.net

```
'--- Create a new Worksheet
          Dim xla As New ExcelApplication()
          Dim wb As Workbook = xla.Create()
          Dim ws As Worksheet = wb.Worksheets.CreateWorksheet("Sheet2")

          '--- Get an existing Worksheet by index
          Dim xla As New ExcelApplication()
          Dim wb As Workbook = xla.Create()
          Dim ws As Worksheet = wb.Worksheets(0)

          '--- Get an existing Worksheet by name
          Dim xla As New ExcelApplication()
          Dim wb As Workbook = xla.Create()
          Dim ws As Worksheet = wb.Worksheets("Sheet2")
```

Properties

Name	Description
Cells	Returns a Cells collection that contains all cells in the worksheet.
Charts	Returns a Charts collection containing all the charts contained as objects within this worksheet. Use this collection to add, access, and remove charts from the worksheet.
Comments	Returns a Comments collection that contains all comments in the worksheet.
FirstShownColumn	Sets or returns the 0-based index of the first column that is shown in the worksheet.
FirstShownRow	Sets or returns the 0-based index of the first row that is shown in the worksheet.
FreezePanes	Freezes panes in a worksheet or returns the cell at which the panes are split. Set the property to a Cell object. The cell specified in `Worksheet.FreezePanes` is the first unfrozen cell; freeze panes will be applied to all rows above the cell and all columns to the left of the cell.
GridlinesColor	Sets or returns a `Color` object representing the color of the gridlines in the worksheet.
Hyperlinks	Returns the collection of hyperlinks in the worksheet.
IsProtected	Returns whether or not the worksheet is write-protected.
IsSelected	Returns whether this worksheet is selected.
Name	Sets or returns the name of the worksheet. Sheet names are limited to 255 characters and must not contain the following characters: \ / ? [ ] : '
NamedRanges	Returns the named ranges of the Worksheet as an array of NamedObject objects as an `IEnumerable` collection. Though the named ranges are not returned in any particular order, they iterable.
PageSetup	Returns a PageSetup object representing page layout properties of a printed worksheet.

Pictures	Returns a Pictures collection containing all pictures in the worksheet. Use this collection to add, access, and remove pictures from the worksheet.
PopulatedCells	Returns a rectanglular area of cells from the first (top, left) populated cell to the last (bottom, right) populated cell in the worksheet. The area returned will include cells that contain no data but have been formatted.
Position	Returns the 0-based index of the worksheet in the collection of all sheets in the workbook.
ProtectPasswordHash	Sets or returns the password hash that is used to write-protect (not encrypt) the worksheet.
ShapeGroups	Returns a Groups collection containing all shape groups within the worksheet.
Shapes	Returns a Shapes collection containing all shapes within the worksheet.
SheetProtection	Returns a SheetProtection object containing all protection properties on the worksheet.
ShowFormulas	Sets or returns whether formulas or their calculated values will be displayed in the worksheet. Calculated values are displayed by default.
ShowGridlines	Sets or returns whether gridlines should be shown in the worksheet. Gridlines are shown by default.
ShowRowColHeaders	Sets or returns whether column and row headers will be displayed in the worksheet. The headers are displayed by default.
ShowZeroValues	Sets or returns whether zero values in cells will be displayed in the worksheet. Zero values are displayed by default.
StandardHeight	Sets or returns the default row height in points (1/72 of an inch). This must be between 0 and 409.
StandardWidth	Sets or returns the default column width, in points (1/72 of an inch).
StandardWidthInChars	Sets or returns the default column width as a number of character widths in the 'Normal' font. This must be a value between 0 and 255.
SummaryColumns	When grouping columns, SummaryColumns determines where the summary columns for the groups are located. By default the summary rows are located to the right of the groupings.
SummaryRows	When grouping rows, SummaryRows determines where the summary rows for the groups are located. By default the summary rows are located below the groupings.
TabColor	Sets or returns the color of the worksheet's sheet name tab. Setting the color to an automatic color will set the color to no color.
Workbook	Returns the Workbook object in which this worksheet resides.
ViewState	Sets or returns the worksheet's viewing mode.
Visibility	Sets or returns the visibility setting of the worksheet.
ZoomPercentage	Sets or returns the percentage of standard size by which the worksheet will be magnified or reduced. Set this property to a value between 10 and 400.

Indexers

Name	Description
Item(Int32, Int32)	Returns the cell at a specified row and column position. This property is an indexer for the Worksheet class.

Item(String)	Returns the cell at the specified Excel-style reference. This property is an indexer for the Worksheet class.

Methods

Name	Description
CopyPaste(Cell, Area)	Copies an area of cells from another Worksheet to cells in this worksheet.
CopyPaste(Cell, Area, CopyPasteProperties)	Copies an area of cells from another Worksheet to cells in this worksheet.
CopyPaste(Int32, Int32, Area)	Copies an area of cells from another Worksheet to cells in this worksheet.
CopyPaste(Int32, Int32, Area, CopyPasteProperties)	Copies an area of cells from a Worksheet in either the same workbook or another workbook. You may specify the types of data copied (cell value, formulas, comments, formatting, etc.) using the CopyPasteProperties object. The CopyPaste method supports copying column width, comments, formulas, merged cells, row height, values, and cell and number formatting.The CopyPaste method does not currently support copying anything not listed above, including charts, images, or autofilters.
CopyPaste(String, Area, CopyPasteProperties)	Copies an area of cells from another Worksheet to cells in this worksheet.
CopyPaste(String, Area)	Copies an area of cells from another Worksheet to cells in this worksheet.
CreateAnchor(Int32, Int32, Double, Double)	Creates an anchor at a specified position in the worksheet.
CreateArea(Int32, Int32, Int32, Int32)	Defines a rectangular Area of cells in the worksheet.
CreateArea(String)	Defines a rectangular Area of cells in the worksheet.
CreateAreaOfColumns(Int32, Int32)	Returns an Area object representing all the cells in a specified group of columns. An area is a rectangular collection of cells.
CreateAreaOfRows(Int32, Int32)	Returns an Area object representing all the cells in a specified group of rows. An area is a rectangular collection of cells.
CreateNamedRange(Int32, Int32, Int32, Int32, String)	Creates a named range, containing one area. A range is a collection of areas; an area is a rectangular collection of cells.
CreateNamedRange(String, String)	Creates a named range from a specified formula. A range is a collection of areas; an area is a rectangular collection of cells. This method can be used to create a non-rectangular range containing multiple rectangular areas.
CreateRange(String)	Creates a range from a specified formula. A range is a collection of areas; an area is a rectangular collection of cells. This method can be used to create a non-rectangular range containing multiple rectangular areas.
DeleteColumn(Int32)	Deletes a column and its contents from the worksheet. Columns after the deleted column will be moved to the left.
DeleteColumns(Int32, Int32)	Deletes a number of columns and its contents from the worksheet. Columns after the deleted columns will be moved to the left.
DeleteRow(Int32)	Deletes a specified row and its contents from the worksheet. Rows below the deleted row will be moved up.
DeleteRows(Int32, Int32)	Deletes a specified number of rows and its contents from the worksheet. Rows below the deleted rows will be moved up.
GetColumnProperties(Int32)	Returns a ColumnProperties object representing the column specified by index.

GetNamedObject(String)	Returns the NamedObject object that represents the name of a specified array, number, picture, or range. If the named object does not exist, the method returns null.
GetNamedRange(String)	Returns the named Range object that specified by name. If the named range does not exist, the method returns null.
GetRowProperties(Int32)	Returns a RowProperties object representing the row specified by index.
GroupColumns(Int32, Int32, Boolean)	Groups or outlines a contiguous set of columns.
GroupRows(Int32, Int32, Boolean)	Groups or outlines a contiguous set of rows.
ImportData(Object()(), Cell)	Imports data from a two-dimensional array of objects to cells in the worksheet.
ImportData(Object()(), String(), Cell, DataImportProperties)	Imports data from a two-dimensional array of objects to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
ImportData(System.Data.DataTable, Cell)	Imports data from an ADO.NET `DataTable` to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
ImportData(System.Data.DataTable, Cell, DataImportProperties)	Imports data from an ADO.NET `DataTable` to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
ImportData(System.Data.DataView, Cell)	Imports data from an ADO.NET `DataView` to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
ImportData(System.Data.DataView, Cell, DataImportProperties)	Imports data from an ADO.NET `DataView` to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
ImportData(Object(,), Cell)	Imports data from a rectangular array of objects to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
ImportData(Object(,), String(), Cell, DataImportProperties)	Imports data from a rectangular array of objects to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
ImportData(System.Data.IDataReader, Cell)	Imports data from an `IDataReader` to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
ImportData(System.Data.IDataReader, Cell, DataImportProperties)	Imports data from an `IDataReader` to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
InsertColumn(Int32)	Inserts a column in the worksheet to the left of the specified column.
InsertColumn(Int32, ColumnInsertBehavior)	Inserts a column in the worksheet to the left of the specified column, copying the style based on the `copyBehavior` parameter.
InsertColumn(Int32, ColumnInsertBehavior, ColumnInsertBehavior)	Inserts a column in the worksheet to the left of the specified column, copying the style based on the `copyBehavior` parameter.
InsertColumns(Int32, Int32)	Inserts a block of columns in the worksheet to the left of the specified column.
InsertColumns(Int32, Int32, ColumnInsertBehavior)	Inserts a block of columns in the worksheet to the left of the specified column, copying the style based on the `copyBehavior` parameter.
InsertColumns(Int32, Int32, ColumnInsertBehavior, ColumnInsertBehavior)	Inserts a block of columns in the worksheet to the left of the specified column, copying the style based on the `copyBehavior` parameter.
InsertHorizontalPageBreak(Cell)	Inserts a horizontal page break in the worksheet after the specified cell.
InsertRow(Int32)	Inserts a row in the worksheet above the specified row.

InsertRow(Int32, RowInsertBehavior)	Inserts a row in the worksheet above the specified row.
InsertRow(Int32, RowInsertBehavior, RowInsertBehavior)	Inserts a row in the worksheet above the specified row.
InsertRows(Int32, Int32)	Inserts a number of rows in the worksheet above the specified row.
InsertRows(Int32, Int32, RowInsertBehavior)	Inserts a number of rows in the worksheet above the specified row.
InsertRows(Int32, Int32, RowInsertBehavior, RowInsertBehavior)	Inserts a number of rows in the worksheet above the specified row.
InsertVerticalPageBreak(Cell)	Inserts a vertical page break in the worksheet to right of the specified cell.
Protect(String)	Write-protects the worksheet. A user will not be able to modify the worksheet in Excel without entering the specified password. This method does not encrypt the worksheet.
Select()	Selects the current worksheet and deselects all others. To select multiple worksheets, use Worksheets.Select().
UngroupColumns(Int32, Int32)	Ungroups a contiguous set of columns.
UngroupRows(Int32, Int32)	Ungroups a contiguous set of rows.
Unprotect()	Removes the write-protection from the worksheet.

Extension Methods

Overload	Description
ImportData(Cell, Microsoft.SharePoint.SPList, DataImportProperties)	Imports data from a SharePoint List to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
ImportData(Cell, Microsoft.SharePoint.SPList)	Imports data from a SharePoint List to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
ImportData(Cell, Microsoft.SharePoint.SPView, Microsoft.SharePoint.SPList, DataImportProperties)	Imports data from a SharePoint View to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.
ImportData(Cell, Microsoft.SharePoint.SPView, Microsoft.SharePoint.SPList)	Imports data from a SharePoint View to cells in the worksheet. The new data will overwrite values and formulas in the target worksheet cells, but existing formatting will be preserved.

Nested Classes

Name	Description
Constants	Defines a set of constants related to worksheets.
SheetViewState	A SheetViewState value specifies the view state of a worksheet in Excel.
SheetVisibility	A SheetVisibility value specifies the visibility level of a worksheet.
SummaryColumnsLocation	A SummaryColumnsLocation value specifies the location of the summary column for a set of grouped columns.
SummaryRowsLocation	A SummaryColumnsLocation value specifies the location of the summary row for a set of grouped rows.