Nasosoft Spreadsheets for .NET

Nasosoft

Advanced .NET Solution

Table of Contents

Overview of Nasosoft Spreadsheets for .NET

Welcome to Nasosoft® Spreadsheets for .NET, which is included in the Nasosoft .NET Components Suite.

Nasosoft Spreadsheets for .NET is a powerful spreadsheet component that gives you the ability to read, write and manipulate Microsoft Excel spreadsheets into your .NET applications without requiring any Microsoft® Office installation.

Nasosoft Spreadsheets for .NET helps .NET programmers to create professional spreadsheet applications more quickly than ever before. With a host of new tools, you can quickly read and write Excel spreadsheets in your .NET applications with an easy-to-use programming interface.

Nasosoft Spreadsheets also fully supports to import data into spreadsheets from variable data sources, manipulate charts, pictures, and formulas etc.

Nasosoft Spreadsheets for .NET is purely built on .NET frameworks. Your application doesn’t depend on Microsoft Excel COM+ interfaces or other native code. Microsoft® Office isn’t required to be installed on the server.

This document provides a programming guide of Nasosoft Spreadsheets for .NET, with an emphasis on new and improved features.

Nasosoft Spreadsheets for .NET Features

Platforms

Feature	Description
.NET Framework v2.0	Yes, Nasosoft Spreadsheets for .NET supports .NET Framework 2.0.
.NET Framework v3.0	Yes, Nasosoft Spreadsheets for .NET supports .NET Framework 3.0.
.NET Framework v3.5	Yes, Nasosoft Spreadsheets for .NET supports .NET Framework 3.5.
.NET Framework v4.0	Yes, Nasosoft Spreadsheets for .NET supports .NET Framework 4.0.

Mono	Yes, Nasosoft Spreadsheets for .NET supports Mono on Windows and Linux.

Spreadsheet Content Manipulation

Feature	Feature
Insert Images	Yes
Create Charts	Yes
Add /Remove Comments	Yes
Auto-Filters	Yes
Data Validations	Yes
Implement Data Sorting	Yes
Find and replace text	Yes
Merge / Unmerge Cells	Yes
Group / Ungroup Rows and Columns	Yes
Create custom Page Breaks	Yes
Insert / Delete or Hide Rows, Columns and Sheets	Yes
Formula	Yes
Conditional Formatting	Yes
Freeze / Unfreeze Panes	Yes
Insert Hyperlinks	Yes
Specify Document Properties Settings	Yes
Protect / Unprotect Worksheets	Yes
Manipulate Named Ranges	Yes
Encrypt and Decrypt Files	Yes

Formatting

Feature	Feature
Formatting in cells like fonts, colors, effects, borders and shading.	Yes
Number format for cells.	Yes
Text Alignment.	Yes
Font Settings for cells.	Yes
Colors to Cells, Fonts, Gridlines, Graphic objects etc.	Yes
Rich Text formatting.	Yes
Border Settings.	Yes

Printing and Page Setup

Feature	Feature
Adjust page orientation, scaling, paper size.	Yes
Specify margins and page centering.	Yes
Create or edit header and/or footer.	Yes
Set print area, print titles, or turn on gridlines etc.	Yes

Install Nasosoft Spreadsheets for .NET

System Requirements

Operating Systems

Operating System	support
Microsoft Windows Server 2003 (x86 and x64)	Yes
Microsoft Windows Server 2003 R2 (x86 and x64)	Yes
Microsoft Windows Server 2008 (x86 and x64)	Yes
Microsoft Windows Server 2008 R2 (x86 and x64)	Yes
Microsoft Windows Vista (x86 and x64)	Yes

Microsoft Windows 7 (x86 and x64)	Yes
Microsoft Windows 8/8.1 (x86 and x64)	Yes
Microsoft Windows Server 2012	Yes
Microsoft Windows Server 2012 R2	Yes

Platforms

Platforms	**Support**
Microsoft .NET Framework 2.0	Yes
Microsoft .NET Framework 3.0	Yes
Microsoft .NET Framework 3.5	Yes
Microsoft .NET Framework 4.0	Yes
Mono	Yes

Development Tools / IDE

Development Tools /IDE	**Support**
Microsoft Visual Studio 2005	Yes
Microsoft Visual Studio 2008	Yes
Microsoft Visual Studio 2010	Yes
Microsoft Visual Studio 2012	Yes
Microsoft Visual Studio 2013	Yes
Borland C# Builder	Yes
Mono Develop	Yes
Sharp Develop	Yes

Install and Uninstall Nasosoft Spreadsheets for .NET

Nasosoft Spreadsheets for .NET is distributed as a MSI installer. You can use the installer to install and uninstall the components. The installer performs following tasks:

- Copies the Nasosoft Spreadsheets component assemblies
- Copies the sample projects with source code
- Install the API documentation and programming guide.

Deploy Nasosoft Spreadsheets for .NET

Nasosoft Spreadsheets for .NET supports XCOPY deployment like other typical .NET components. You can simply copy the Nasosoft Spreadsheets assemblies to your application bin folder.

Upgrade from Nasosoft Excel for .NET

This section provides information about upgrading from Nasosoft Excel for .NET v5.x.

Move to Next Generation Spreadsheet Component

Nasosoft Spreadsheets for .NET is the next generation spreadsheet component for Microsoft Excel formats.

- More spreadsheet formats
- Better Open XML file format support
- Big Performance Improvement
- Better Tracing and Diagnostics

Class Mappings

The class mappings from Nasosoft Excel to Nasosoft Spreadsheet are quite straightforward. Here are the class mappings for major classes.

Nasosoft.Document.Xls	**Nasosoft.Documents.Spreadsheets**
XlsWorkbook	Workbook
XlsWorksheet	Worksheet
XlsRange	Range
XlsCell	Cell
XlsColumn	Column
XlsRow	Row
XlsComment	Comment

Getting Started with Nasosoft Spreadsheets for .NET

This section contains basic background information that will help you understand and build your first spreadsheet application with Nasosoft Spreadsheets for .NET.

Download and Evaluate Nasosoft Spreadsheets for .NET

Nasosoft Spreadsheets for .NET is provided as part of the Nasosoft .NET Components Suite, which is a collection of all the .NET components package from Nasosoft.

Download and install the Nasosoft .NET Components Suite.

> **Note** Before you install Nasosoft .NET Components Suite, you need to install .NET Framework 2.0 or later version, which you can download at Microsoft Download Center.

The evaluation copy of the Nasosoft Spreadsheets for .NET provides fully functions with watermarks injection.

Build your First App with Nasosoft Spreadsheets for .NET

Below tutorial walks-through how to create, build and run your first spreadsheet application using C#.

Please follow the step by step tutorial to create the HelloWorld application:

1. Create a new instance of **Workbook** class
2. Create a new Excel document
3. Insert the "Hello, World!" string into the cell in the Excel document
4. Save the Excel document

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0];

//Insert a row into the worksheet
worksheet["A1"].Text = "Hello, World!";

//Save the Excel file
workbook.Save("output.xlsx");
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

'Insert a row into the worksheet
worksheet("A1").Text = "Hello, World!"

'Save the Excel file
workbook.Save("output.xlsx")
```

Explore and Study Code Samples

The code samples for Nasosoft Spreadsheets for .NET are included in MSI setup package.

You can explore and build code samples easily.

How to Get Answers and Technical Support

FREE Technical Support by Email

We also provide FREE Email support service to our customers. If you have any questions about Nasosoft Spreadsheets for .NET or having problems to implement your scenario, please feel free to email your questions or concerns to sales@nasosoft.com. We are more than glad to provide any help.

This service is totally **FREE**. It is available to all our product users and customers. You don’t need to pay for it. You don’t need to purchase any of our products. If you are using Nasosoft products, feel free to submit your problems in emails to sales@nasosoft.com.

FREE Customized Demo and Samples

We understand that people have different and variable product scenarios for spreadsheet programming. Our code samples and demo might not cover all of possible product scenarios. We are more than glad to help our customers to create customized demos and code samples.

Please feel free to submit your product scenarios and requirements on our forums or by email (sales@nasosoft.com). It is FREE. Our developers and support engineers will develop demo and code samples for your reference.

Ask Your Questions on Forum

We have product forums for technical support and discussion. Our developers and technical support engineers are responsive to reply the posts on the forums. If you have any questions about Nasosoft Spreadsheets for .NET, please feel free to post your questions or concerns on our forums. We will get back to you as soon as possible.

> **Note** You need to register an account on our website in order to get the permission to submit your questions.

FREE 30 Days Trial License

Nasosoft .NET Components come with advanced, professional, feature-rich controls for C# and VB.NET. Start to get FREE evaluation in this section.

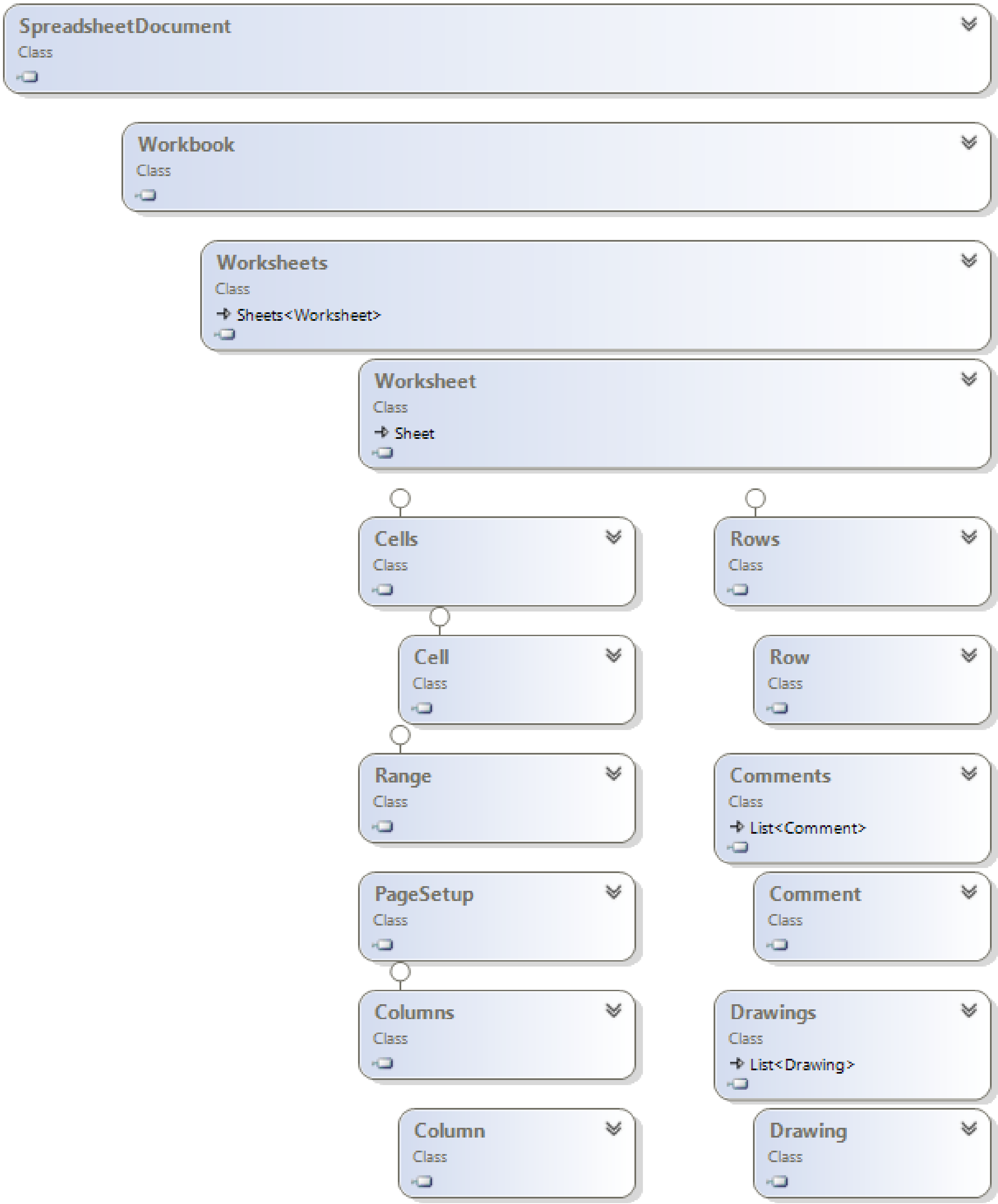
You can get 30 days Trial License for free by submitting your request by email (sales@nasosoft.com). Our sales will get back to you within 2 business days.

Spreadsheet Object Model Overview

To develop spreadsheet solutions that uses Nasosoft Spreadsheets for .NET, you can easily interact with Excel document model programmatically. This topic introduces the most important objects:

- Nasosoft.Documents.Spreadsheets.SpreadsheetDocment
- Nasosoft.Documents.Spreadsheets.Workbook
- Nasosoft.Documents.Spreadsheets.Worksheet
- Nasosoft.Documents.Spreadsheets.Range
- Nasosoft.Documents.Spreadsheets.Cell

The Spreadsheet Object Model closely follows the user interface of Microsoft Excel. The SpreadsheetDocument object represents the entire document, and Workbook object contains a collection of Worksheet objects. From there, the major abstraction that represents cells is the Range object and Cell object, which enables you to work with a group of cells and individual cell.


SpreadsheetDocument
Class
Workbook
Class
Worksheets
Class
Sheets<Worksheet>
Worksheet
Class
Sheet
Cells
Class
Rows
Class
Cell
Class
Row
Class
Range
Class
Comments
Class
List<Comment>
PageSetup
Class
Comment
Class
Columns
Class
Drawings
Class
List<Drawing>
Column
Class
Drawing
Class

SpreadsheetDocument Object

The Nasosoft.Documents.Spreadsheets.SpreadsheetDocument object represents the entire Excel document. It is the root of the spreadsheet object model. You can create and edit an Excel file using SpreadsheetDocument object.

[Syntax Sugar]

You can just create Workbook object directly without creating a SpreadsheetDocument object explicitly. We will create a SpreadsheetDocument object for you silently in the constructor of Workbook object.

Workbook Object

The Nasosoft.Documents.Spreadsheets.Workbook object represents a single workbook within an Excel document. When you open an Excel document or create a new document, you create a new Workbook object.

Worksheet Object

The Nasosoft.Documents.Spreadsheets.Worksheet object represents a single worksheet in an Excel workbook. The Workbook object contains a collection of Worksheet objects.

Range Object

The Nasosoft.Documents.Spreadsheets.Range object represents a cell, a row, a column, a selection of cells containing one or several blocks of cells. Before you can manipulate any region in the Excel worksheet, you need to get a Range object from the Worksheet object.

Range object is one of the most popular objects you would use in your spreadsheet applications.

There are a couple ways to get a Range object from the Worksheet object:

1. Use Worksheet.GetRange function

```
 [C#]

var range = worksheet.GetRange(1,1);
```

```
[VB.NET]

Dim range As Range = worksheet.GetRange(1, 1)
```

2. Use indexers of the Worksheet object

```
[C#]

var range = worksheet[1,1,2,2]
```

```
[VB.NET]

Dim range As Range = worksheet(1,1,2,2)
```

Cell Object

The Nasosoft.Documents.Spreadsheets.Cell object presents a single cell within the Excel worksheet.

Working with Workbooks

This section contains several topics about how to open, create and manipulate an Excel Files.

Open an Excel File

Nasosoft Spreadsheets for .NET provides variable ways to open an existing Excel file. With specified file location and file format type, you can load Excel file formats or other data file formats easily.

[Syntax Sugar]

You don’t need to specify the file format type of the excel file that you want to open. Workbook object would automatically detect what is the file format of the excel file for you.

Open an Excel File with Specified Path

Code Example:

```
[C#]

//Create a Workbook object, and open an Excel file with specified path
Workbook workbook = new Workbook("book1.xlsx");
```

```
[VB.NET]

'Create a Workbook object, and open an Excel file with specified path
Dim workbook As New Workbook("book1.xlsx")
```

Open an Excel File with Specified Stream

Code Example:

```
[C#]

FileStream fstream = new FileStream("book1.xlsx", FileMode.Open);

//Create a Workbook object, and open an Excel file with specified stream object
Workbook workbook = new Workbook(fstream);
```

```
[VB.NET]
Dim fstream As FileStream = New FileStream("C:\book1.xlsx", FileMode.Open)

//Create a Workbook object, and open an Excel file with specified stream object
Dim workbook As Workbook = New Workbook(fstream)
```

Open an Encrypted Excel File with Password

Code Example:

```
[C#]
//Create a Workbook object, and open an Excel file with specified path
Workbook workbook = new Workbook("book1.xlsx", “password1”);
```

```
[VB.NET]
'Create a Workbook object, and open an Excel file with specified path
Dim workbook As New Workbook("book1.xlsx", “password1”)
```

Save an Excel File

Create and Save an Excel File to Specified Path

```
[C#]
//Create a Workbook object
Workbook workbook = new Workbook();

//Save the workbook to specified path
workbook.Save("book1.xlsx");
```

```
[VB.NET]
'Create a Workbook object
Dim workbook As Workbook = New Workbook()

'Save the workbook to specified path
workbook.Save("book1.xlsx")
```

Save an Excel File to Specified Stream

```
[C#]

FileStream fstream = new FileStream("book1.xlsx", FileMode.CreateNew);

//Create a Workbook object
Workbook workbook = new Workbook();

//Save the workbook to the specified stream object
workbook.Save(fstream);
```

```
[VB.NET]

Dim fstream As FileStream = New FileStream("C:\book1.xlsx", FileMode.CreateNew)

'Create a Workbook object
Dim workbook As Workbook = New Workbook()

'Save the workbook to the specified stream object
workbook.Save(fstream)
```

Save an Excel File as Excel 97-2003 Format

```
[C#]


//Create a Workbook object
Workbook workbook = new Workbook();

//Save the workbook to the specified stream object
workbook.Save(“book1.xls”, SpreadsheetFormatType.Excel2003);
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook()

'Save the workbook to the specified stream object
workbook.Save(“book1.xls”, SpreadsheetFormatType.Excel2003)
```

Protect and Unprotect an Excel File

Protect Content

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

//Protect the workbook content with password
Workbook.ProtectContent(“password”);

workbook.Save("C:\book1.xlsx");
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook()

'Protect the workbook content with password
Workbook.ProtectContent( “password”)

workbook.Save("C:\book1.xlsx")
```

Unprotect

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook("C:\book1.xlsx");

//UnProtect the workbook content with password
Workbook.Unprotect(“password”);

workbook.Save("C:\book1.xls");
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook("C:\book1.xlsx")

'UnProtect the workbook content with password
Workbook.Unprotect(“password”)

workbook.Save("C:\book2.xlsx")
```

Working with Worksheets

This section contains several topics about how to add, delete and manage worksheets in an Excel files.

Add and Remove a Worksheet

Add a Worksheet

Example:

```
[C#]

//Create a new workbook
Workbook workbook = new Workbook();

//Add a new worksheet
workbook.Worksheets.Add("newsheet");

//Save the Excel file
workbook.Save("C:\book1.xls");
```

```
[VB.NET]

'Create a new workbook
Dim workbook As Workbook = New Workbook()

'Adding a new worksheet to the Workbook object
workbook.Worksheets.Add("newsheet")

'Save the Excel file
workbook.Save("C:\book1.xlsx")
```

Remove a Worksheet

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook("C:\book1.xlsx");

//Remove the worksheet with specified sheet name
workbook.Worksheets.Remove("sheet1");

//Save the Excel file
workbook.Save("C:\book1.xlsx");
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook("C:\book1.xlsx")

'Remove the worksheet with specified sheet name
workbook.Worksheets.Remove("sheet1")

'Save the Excel file
workbook.Save("C:\book1.xlsx")
```

Hide and Show a Worksheet

Hide a Worksheet

Example:

```
[C#]

// Create a Workbook object
Workbook workbook = new Workbook ();

//Add a new worksheet
workbook.Worksheets.Add("newsheet");

//Get the first worksheet
Worksheet worksheet = workbook.Worksheets[0];
```

```
//Hide the sheet
worksheet.HideSheet();

//Save the Excel file
workbook.Save("C:\book1.xlsx");
```

```
[VB.NET]

'Instantiating a Workbook object
Dim workbook As Workbook = New Workbook()

'Add a new worksheet
workbook.Worksheets.Add("newsheet")

'Get the first worksheet
Dim worksheet As Worksheet = workbook.Worksheets(0)

'Hide the sheet
worksheet.HideSheet();

'Save the Excel file
workbook.Save("C:\book1.xlsx")
```

Print a Worksheet

Change Page Setup

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

//Get the page setup of the worksheet
PageSetup pageSetup = workbook.Worksheets[0].PageSetup;

//change page setup
pageSetup.FitToHeight = true;

pageSetup.FitToWidth = true;

pageSetup.CellComments = CellCommentsType.None;
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the page setup of the worksheet
Dim pageSetup As PageSetup = workbook.Worksheets(0).PageSetup

'Set print title columns
pageSetup.FitToHeight = True

pageSetup.FitToWidth = True

pageSetup.CellComments = CellCommentsType.None
```

Insert and Delete a Row

Insert a Row

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets.Add();

//Insert a row into the worksheet
worksheet.InsertRow(2);

//Save the Excel file
workbook.Save("output.xlsx");
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

'Insert a row into the worksheet
worksheet.InsertRow(2);

'Save the Excel file
workbook.Save("output.xlsx");
```

Delete a Row

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0];

//Delete a row in the worksheet
worksheet.DeleteRow(2);

//Save the Excel file
workbook.Save("book1.xlsx");
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets.Add();

'Delete a row in the worksheet
worksheet.DeleteRow(2)

'Save the Excel file
workbook.Save("book1.xlsx")
```

Insert and Delete a Column

Insert a Column

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets.Add()

//Insert a column into the worksheet
worksheet.InsertColumn(2);

//Save the Excel file
workbook.Save("output.xlsx");
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets.Add();

'Insert a column into the worksheet
worksheet.InsertColumn (2)

'Save the Excel file
workbook.Save("book1.xlsx")
```

Delete a Column

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets.Add();

//Delete a column in the worksheet
worksheet.DeleteColumn(2);

//Save the Excel file
workbook.Save("book1.xlsx");
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

'Delete a column in the worksheet
worksheet.DeleteColumn (2)

'Save the Excel file
workbook.Save("book1.xlsx")
```

Working with Ranges

This section contains several topics about how to use Range object to edit the content in the worksheet.

Get and Set Cell Value

Get and Set String Value

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets.Add();

worksheet[“A1”].Text = “Hello, World!”;

//Get cell value
string cellValue = worksheet[“A1”].Text;
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets.Add()

'Get cell value
Dim celValue As String = worksheet(“A1”).Text

worksheet(“A1”).Text = “Hello, World!”
```

Get and Set Number Value

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0];

//Set a Number value
worksheet[“A1”].Number = 1000;

//Get cell value
double cellValue = worksheet[“A1”].Number;
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

‘Set a Number value
Worksheet(“A1”).Number = 1000

'Get cell value
Dim celValue As Double = worksheet(“A1”).Text
```

Get and Set Boolean Value

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0];

//Set a Boolean value
worksheet[“A1”].Boolean = true;

//Get cell value
bool cellValue = worksheet[“A1”].Boolean;
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

'Set a Boolean value
Worksheet(“A1”).Boolean = True

'Get cell value
Dim celValue As Boolean = worksheet(“A1”).Text
```

Change Cell Format of a Range

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0];

worksheet[“A1”].Text = “Bold”;
worksheet[“A1”].CellFormat.Font.Bold = true;
worksheet[“A1”].CellFormat.Font.Color = System.Drawing.Color.Red;
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0);

worksheet[“A1”].Text = “Bold”
worksheet[“A1”].CellFormat.Font.Bold = True
worksheet[“A1”].CellFormat.Font.Color = System.Drawing.Color.Red
```

Working with Cells and Columns

This section contains several topics about how to use Cell and Column in worksheets.

Get and Visit Cells

[Important]
Be careful to choose GetCells and VisitCells function of Worksheet. GetCells function would have a side effect on the worksheet object, which would create a cell if the specified cell is not existing in this worksheet. However, VisitCells function doesn't create a cell and return Null for this case.

Get Cells by Row and Column

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook(“book1.xlsx”);

Worksheet worksheet = workbook.Worksheets[0];

//Get all cells in [0,0,20,20]
foreach(var cell in worksheet.GetCells(0, 0, 20, 20))
{
   cell.Text = “Hello, World!”;
}
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

‘Get all cells in [0,0,20,20]
For Each cell As Cell In worksheet.GetCells(0, 0, 20, 20)
   cell.Text = “Hello, World!”
NEXT
```

Get Cells by Reference

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();
'Get the first worksheet
Worksheet worksheet = workbook.Worksheets[0];

//Get all cells in [A1:B20], if the cell is not existing, create one
foreach(var cell in worksheet.GetCells(“A1:B20”))
{
   cell.Text = “Hello, World!”;
}
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

‘Get all cells in [A1:B20], if the cell is not existing, create one
For Each cell As Cell In worksheet.GetCells(“A1:B20”)
   cell.Text = “Hello, World!”
NEXT
```

Visit Cells by Row and Column

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook(“book1.xlsx”);

Worksheet worksheet = workbook.Worksheets.Add();

//visit all cells in [0,0,20,20]
foreach(var cell in worksheet.VisitCells(0, 0, 20, 20))
{
   cell.Text = “Hello, World!”;
}
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook(“book1.xlsx”)

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

‘Visit all cells in [0,0,20,20]
For Each cell as Cell in worksheet.VisitCells(0, 0, 20, 20)
   cell.Text = “Hello, World!”
NEXT
```

Visit Cells by Reference

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook(“book1.xlsx”);

Worksheet worksheet = workbook.Worksheets.Add();

//visit all cells in [A1:B20]
foreach(var cell in worksheet.VisitCells(“A1:B20”))
{
   cell.Text = “Hello, World!”;
}
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets.Add();

‘Visit all cells in A1:B20
For Each cell as Cell in worksheet.VisitCells(“A1:B20”)
   cell.Text = “Hello, World!”
NEXT
```

Get and Set Cell Value

Get and Set String Value

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0];

var cell = Worksheet.GetCell(0,0);

cell.Text = “Hello, World!”;
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

'Get cell value
Dim cell As Cell = worksheet.GetCell(0,0)

cell.Text = “Hello, World!”

//Get cell value
Dim cellValue As String = cell.Text;
```

Get and Set Number Value

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0];

var cell = Worksheet.GetCell(0,0);

//Set a Number value
cell.Number = 1000;

//Get cell value
double cellValue = cell.Number;
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

Dim cell As Cell = worksheet.GetCell(0,0)

‘Set a Number value
cell.Number = 1000

'Get cell value
Dim celValue As Double = cell.Number
```

Get and Set Boolean Value

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0];

var cell = Worksheet.GetCell(0,0);

//Set a Boolean value
cell.Boolean = true;

//Get cell value
bool cellValue = cell.Boolean;
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

Dim cell As Cell = worksheet.GetCell(0,0)

‘Set a Boolean value
cell.Boolean = 1000

'Get cell value
Dim celValue As Boolean = cell.Boolean
```

Change Cell Format

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0];

var cell = worksheet.GetCell(0,0);
cell.Text = “Bold”;
cell.CellFormat.Font.Bold = true;
cell.CellFormat.Font.Color = System.Drawing.Color.Red;
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0);

Dim cell As Cell = worksheet.GetCell(0,0)
cell.Text = “Bold”
cell.CellFormat.Font.Bold = True
cell.CellFormat.Font.Color = System.Drawing.Color.Red
```

Working with Formulas

This section contains several topics about how to use formulas in worksheets.

Use Built-In Formulas

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0]

worksheet[“A1”].Number = 10;
worksheet[“B1”].Number = 20;

worksheet[“C1”].Formula=”SUM(A1,B1)”;
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

worksheet[“A1”].Number = 10
worksheet[“B1”].Number = 20

worksheet[“C1”].Formula=”SUM(A1,B1)”
```

Working with Comments

This section contains several topics about how to insert and delete comments in worksheets.

Insert Comments

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0];

worksheet[“A1”].Text = “hello, world!”;
worksheet[“A1”].InsertComment(“This is a comment.”);
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0);

worksheet[“A1”].Text = “Bold”
worksheet[“A1”].InsertComment(“This is a comment.”)
```

Delete Comments

Example:

```
[C#]

//Create a Workbook object
Workbook workbook = new Workbook();

Worksheet worksheet = workbook.Worksheets[0];

worksheet[“A1”].DeleteComment();
```

```
[VB.NET]

'Create a Workbook object
Dim workbook As Workbook = New Workbook ()

'Get the first worksheet
Dim worksheet As Worksheet= workbook.Worksheets(0)

worksheet[“A1”].DeleteComment()
```

Licensing

This section introduces how to set license file.

Set License File

Example:

```
[C#]

//Create a license object
LicenseManager license = new LicenseManager();

//Set the license
license.SetLicense("nasosoft.lic");
```

```
[VB.NET]

'Create a license object
Dim license As New LicenseManager()

'Set the license
license.SetLicense("nasosoft.lic")
```

Resources

For more information about Nasosoft Spreadsheets for .NET, please visit www.nasosoft.com.