Importing Data

ExcelApplication's ImportData method allows you to import blocks of data to a worksheet from a database or a rectangular array. ImportData is a method of both Worksheet and Area. The method returns a Area object representing a the set of cells that contain the imported data.

Table of Contents

ImportData does not insert values into the worksheet and push existing values down. The imported data will overwrite existing values in the target cells.

Importing from a Database

You can import values from a database to a worksheet by passing the ImportData method a DataTable, DataView, SqlDataReader, OleDbDataReader, or AdomdDataReader.

The DataTable and DataView classes are in the System.Data namespace. Use an Import directive to import the namespace to the aspx page:

```
<%@ Import namespace="System.Data" %>
```

To import System.Data to a C# code-behind page (.aspx.cs), use:

```
using System.Data;
```

To import database values using OleDb, import the System.Data.OleDb namespace to your page. To import database values using SqlClient, import the System.Data.SqlClient namespace to your page.

To import values from a database to your worksheet:

1. Connect to the database and execute a query to return a DataTable, DataView, SqlDataReader, OleDbDataReader, or AdomdDataReader for example:

```
private DataTable GetEmployeeDataTable()
{
     string EmployeeSQL = "SELECT TOP 10 FirstName + ' ' + LastName As Name, " +
          "Title FROM Employee";

     DataTable dt = new DataTable();

     //--- "connString" is a SQL connection string
     using(SqlConnection conn = new SqlConnection(connString))
          new SqlDataAdapter(EmployeeSQL, conn).Fill(dt);

     return dt;
}
```

2. Pass the ADO.NET DataTable to ImportData:

```
//--- Import to a worksheet and pass ImportData
//--- the cell at which to start entering the data.
Cell CellB5 = Sheet1.Cells[4, 1];
Area importedValues = Sheet1.ImportData(dt, CellB5);

//--- Or: Import to a defined area.
Area targetArea = Sheet1.CreateArea(4, 4, 15, 6);
Area importedValues = targetArea.ImportData(dt);
```

Importing from an Array

To import values from a two-dimensional array to your worksheet:

1. Create a rectangular array, for example:

```
string[,] arrayData = {{"Nancy", "Davolio", "Sales Manager"},
     {"Michael", "Suyama", "HR Representative"},
     {"Adrian", "King", "IS Support"}};
```

2. Pass the array to ImportData:

```
//--- Import to a worksheet and pass ImportData
//--- the cell at which to start entering the data.
Cell CellB5 = Sheet1.Cells[4, 1];
Area importedValues = Sheet1.ImportData(arrayData, CellB5);

//--- Or: Import to a defined area.
Area targetArea = Sheet1.CreateArea(4, 4, 15, 6);
Area importedValues = targetArea.ImportData(arrayData);
```

Customizing Your Data Import

The DataImportProperties class contains a set of properties that are used when importing data to cells in a worksheet. The settings of a DataImportProperties object will be applied to a data import if the object is passed to ImportData (with the set of values to import). You can create several DataImportProperties objects and assign a different one to each data import, or re-use one object in multipleImportData calls.

To customize a data import using a DataImportProperties object:

1. Create a DataImportProperties object with the Workbook.CreateDataImportProperties() method:

```
ExcelApplication xla = new ExcelApplication();
Workbook wb = xla.Create();
DataImportProperties importProps = wb.CreateDataImportProperties();
```

2. Set one or more data import propeties:

```
//--- Truncate imported data rows and columns that do
//--- not fit within the target rows and columns in the spreadsheet.
importProps.Truncate = true;

//--- By default, when data is imported from a two-dimensional
//--- array, the data will be entered as [row][column]. If
//--- Transpose is set to true, the data will be entered as
//--- [column][row].
importProps.Transpose = true;
```

3. Define a DataTable, DataView, SqlDataReader, OleDbDataReader, AdomdDataReader or rectangular array, for example:

```
string[,] arrayData = {{"Nancy", "Davolio", "Sales Manager"},
     {"Michael", "Suyama", "HR Representative"},
     {"Adrian", "King", "IS Support"}};
```

4. Pass the data and the DataImportProperties object to ImportData:

```
ExcelApplication xla = new ExcelApplication();
Workbook wb = xla.Create();
Worksheet ws = wb.Worksheets[0];
DataImportProperties importProps = wb.CreateDataImportProperties();

//--- UseColumnNames will import column names in addition to data
importProps.UseColumnNames = true;

//--- Import to a worksheet and pass ImportData
//--- the cell at which to start entering the data.
Cell CellB5 = ws.Cells[4, 1];
String[] colNames = {"Name", "LastName", "Title"};
Area importedValues = ws.ImportData(arrayData, colNames,
     CellB5, importProps);

//--- Or: Import to a defined area.
Area targetArea = ws.CreateArea(4, 4, 15, 6);
String[] colNames = {"Name", "LastName", "Title"};
Area importedValues = targetArea.ImportData(arrayData, colNames,
     importProps);
```