ExcelTemplate.BindData(System.Data.IDataReader, String, DataBindingProperties)

Description

Sets an ADO.NET IDataReader as a data source to bind to template data markers.

C#

```
 public void BindData(System.Data.IDataReader source, System.String dataSourceName,
DataBindingProperties property)
```

vb.net

```
Public Sub BindData(ByVal source As System.Data.IDataReader, ByVal dataSourceName As
String, ByVal [property] As DataBindingProperties)
```

Parameters

source

This parameter must be an instance of `System.Data.SqlClient.SqlDataReader`, `System.Data.OleDb.OleDbDataReader` or `Microsoft.AnalysisServices.AdomdClient.AdomdDataReader`, otherwise `BindData` will throw an exception.

dataSourceName

The name of the set of data markers at which to insert the values imported from the data source. `dataSourceName` must be specified, but can be left as null or an empty string if this is the first data source bound AND the data markers in the template use the short data marker syntax or refer to the datasource by number rather than name. Note: `dataSourceName` does not include a data marker's column name, for example, the `dataSourceName` for `%%=Products.ProductID` is "Products."

property

The DataBindingProperties object which contains information about how the data should be bound to the template.

property

The DataBindingProperties object which contains information about how the data should be bound to the template. `property` Must be specified, but the `DataBindingProperties` need not be set beforehand. To bind data to a template with the default `DataBindingProperties`, pass in `ExcelTemplate.CreateDataBindingProperties()` as the `property` value. Otherwise, use the `ExcelTemplate.CreateDataBindingProperties()` method to generate a new `DataBindingProperties` object and set the DataBindingProperties.MaxRows, DataBindingProperties.Transpose, and/or DataBindingProperties.WorksheetName properties for the workbook.

Exceptions

ArgumentNullException

`BindData` will throw this exception if `null` (C#) or `Nothing` (VB.NET) is passed to the method.

SARuntimeException

`BindData` will throw this exception if the data source contains more rows than the worksheet can hold.
If there is more than one data marker referring to a data source and the data source is forward only, the exception will be thrown only if the source is larger than all bindings can hold.

ArgumentException

Remarks

You can set several data sources for a single template. Use the following methods to set template data sources: BindCellData, BindColumnData, BindRowData, and BindData.

Examples

C#

```
OleDbDataReader CategoryRdr = GetCategoryReader();
try
{
     ExcelTemplate xlt = new ExcelTemplate();
     xlt.BindData(CategoryRdr,
          "Category",
          ExcelTemplate.CreateDataImportProperties());
} catch(System.Exception ex) {
} finally {
     //--- The OleDbDataReader must be open when it's passed
     //--- to ExcelTemplate. Remember to close it after use.
     if(CategoryRdr!=null)
          CategoryRdr.Close();
}

//--- Get an OleDbDataReader containing a list of product categories
private OleDbDataReader GetCategoryReader()
{
     OleDbConnection Conn = new OleDbConnection();
     Conn.ConnectionString = Application["connstring"].ToString();

     //--- SQL query for a list of categories
     string CategorySQL = "SELECT CategoryName FROM Categories";
     OleDbCommand Cmd = new OleDbCommand(CategorySQL, Conn);
     Conn.Open();
     return Cmd.ExecuteReader();
}
```

vb.net

```
Dim CategoryRdr As OleDbDataReader = GetCategoryReader()
Try
     Dim xlt As New ExcelTemplate()
     xlt.BindData(CategoryRdr, _
          "Category", _
          ExcelTemplate.CreateDataBindingProperties())
Catch ex As System.Exception
Finally
     '--- The OleDbDataReader must be open when it's passed
     '--- to ExcelTemplate. Remember to close it after use.
     If Not CategoryRdr = Nothing
          CategoryRdr.Close()
     End If
End Try

'--- Get an OleDbDataReader containing a list of product categories
Private Function GetCategoryReader() As OleDbDataReader
     Dim Conn As New OleDbConnection()
     Conn.ConnectionString = Application("connstring").ToString()

     '--- SQL query for a list of categories
     Dim CategorySQL As String = "SELECT CategoryName FROM Categories"
     Dim Cmd As New OleDbCommand(CategorySQL, Conn)
     Conn.Open()
     Return Cmd.ExecuteReader()
End Function
```