

# ExcelTemplate.BindData(Object>(), String(), String, DataBindingProperties)

## Description

Sets a two-dimensional - possibly jagged - array of objects as a template data source.

### C#

```
public virtual void BindData(System.Object[][] arrayData, System.String[] columnNames, System.String dataSourceName, DataBindingProperties property)
```

### vb.net

```
Public Overridable Sub BindData(ByVal arrayData As Object>(), ByVal columnNames As String(), ByVal dataSourceName As String, ByVal [property] As DataBindingProperties)
```

## Parameters

### *arrayData*

A two-dimensional array of objects to use as the data source. By default, the first dimension corresponds to row and the second to column (that is, `Object[row][column]`).

### *columnNames*

The names of the columns to get from the data source. If the `columnNames` parameter is null, field binding can only be performed by ordinal (for example, `%%=DSN.#1` or `%%=$DSN`). If `columnNames` is specified, both ordinal field binding and named field binding can be used.

### *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` 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#**

```
ExcelTemplate xlt = new ExcelTemplate();
xlt.Open(@"C:\DataBinding\ArrayBindingTemplate.xls");
string[][] twodim = {
    new string[]{"Watertown", "MA", "02472"},
    new string[]{"Washington", "DC", "20500"}
};
string[] names = {"City", "State", "Zip"};
xlt.BindData(twodim,
    names,
    "TwoDimArray",
    xlt.CreateDataBindingProperties());
xlt.PreserveStrings = true;
xlt.Process();
xlt.Save(Page.Response,
    "ArrayBinding.xls",
    false);
```

### **vb.net**

```
Dim xlt As New ExcelTemplate()
xlt.Open("C:\DataBinding\ArrayBindingTemplate.xls")
Dim twodim()() As String = New String()() { _
    New String[]{"Watertown", "MA", "02472"}, _
    New String[]{"Washington", "DC", "20500"}, _
}
Dim names As String() = {"City", "State", "Zip"}
xlt.BindData(twodim, _
    names, _
    "TwoDimArray", _
    xlt.CreateDataBindingProperties())
xlt.PreserveStrings = True
xlt.Process()
xlt.Save(Page.Response, _
    "ArrayBinding.xls", _
    False)
```

