Arrays and Collections as Data Sources

Intro

Use a variety of .NET data types to populate ExcelTemplate including `string[]`, `ArrayList`, and `Hashtable`.

Methods used:

- BindData - For using 2-dimensional, grid-like data stored in an array. Arrays can be arranged in [row,col] (normal) or [col,row] (transposed) fashions.
- BindColumnData - For use with a 1-dimensional array, collection, or dictionary. Data is filled in the column.
- BindRowData - For use with a 1-dimensional array, collection, or dictionary. Populates with only a single row of the data source. There must be a data marker for every column you want to be shown.

Code

```
public void GenerateReport()
        {
            // Generate the new workbook with ExcelTemplate
            // Create an instance of SoftArtisans ExcelTemplate
            ExcelTemplate xlt = new ExcelTemplate();

            // Open the template workbook
           string templatePath =
@"..\..\ExcelTemplateFiles\ArrayDataSourceTemplate.xlsx";


            xlt.Open(templatePath);


            // Create arrays/IEnumerables and bind them to the template
             // data markers. The template data markers bound to each array
             // are shown below in comments.


            DataBindingProperties basicBindingProperties =
xlt.CreateDataBindingProperties();

            // %%=$SimpleArray
            string[] onedim = { "SoftArtisans", "OfficeWriter", "ExcelTemplate" };
            xlt.BindColumnData(onedim, "SimpleArray", basicBindingProperties);

            // %%=$ArrayList
            ArrayList arrayList = new ArrayList(onedim);
            xlt.BindColumnData(arrayList, "ArrayList", basicBindingProperties);


            // By default, ExcelTemplate expects an array orientation of
             // [rows][cols].  You can pass a transposed array by setting
             // the Transpose argument to true.  Notice how twoDimNormal
             // and twoDimTranspose differ.  This gives you flexibility
             // in how you arrange your data source arrays.

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

```
            // When binding to two-dimensional arrays, the second parameter
             // to BindData (dataMarkerName) must be a String[] array.
             // The values in that array correpond to the data marker names:
             // %%=TwoDimArray.FirstName
             // %%=TwoDimArray.LastName
             // %%=TwoDimArray.Position

            string[] names = { "FirstName", "LastName", "Position" };
            xlt.BindData(twoDimNormal, names, "TwoDimArray", basicBindingProperties);


            // %%=TwoDimTrans.FirstName
             // %%=TwoDimTrans.LastName
             // %%=TwoDimTrans.Position

            string[,] twoDimTranspose = {{"Nancy", "Michael", "Adrian"},
                                         {"Davolio", "Suyama", "King"},
                                         {"Sales Manager", "HR Representative", "IS
Support"}};
            DataBindingProperties bindTransposed = xlt.CreateDataBindingProperties();
            bindTransposed.Transpose = true;
            xlt.BindData(twoDimTranspose, names, "TwoDimTrans", bindTransposed);


            // When binding to an IDictionary collection, the "Key" is
             // the Data Marker Name.
             // The following entries bind to the data marker names:
             // %%=HashTable.FirstName
             // %%=HashTable.LastName
             // %%=HashTable.Position

            IDictionary hashTable = new Hashtable();
            hashTable.Add("FirstName", "Nancy");
            hashTable.Add("LastName", "Davolio");
            hashTable.Add("Position", "Sales Manager");

            xlt.BindRowData(hashTable, "HashTable", basicBindingProperties);


            // %%=Address.Street
            // %%=Address.City
            // %%=Address.State

            string[] addressvalues = { "3 Brook St.", "Watertown", "MA" };
            string[] addressnames = { "Street", "City", "State" };
            xlt.BindRowData(addressvalues, addressnames, "Address",
basicBindingProperties);


            // Process the template to populate it with the Data Source data
            xlt.Process();

            xlt.Save(@"..\..\ExcelOutputFiles\ArrayDataSource_output.xlsx");
xlt.Save(@"..\..\ExcelOutputFiles\ArrayDataSource_output.xlsx");
```

```
        }
    }
```

Downloads

- Template:ArrayDataSourceTemplate.xlsx
- Output:ArrayDataSource_output.xlsx