Data Binding

Today you will learn

- Repeated-Value Data Binding
- Data Source Controls

Repeated-Value Data Binding

- ASP.NET List controls and Rich controls support for Data Binding are:
- ListBox, DropDownList, CheckBoxList, and RadioButtonList
- HtmlSelect
- GridView, DetailsView, FormView, and ListView

Data Binding with Simple List Controls

This can be achieved by three steps:

Data Binding with Simple List Controls

- Create and fill some kind of data object.(DataSet,DataTable,ArrayList,HashTable)
- Link the object to appropriate control.(DataSource and DataMember)
- Activate the binding.(DataBind())

A Simple List Binding Example (Demo)

System.Colletions : ArrayList and HashTable

Strongly Typed Collections

- System.Collections.Generic:
  - You can Specify Single type of Data Object (Catching errors and type conversion is not needed)

Example:

```
using System.Collections.Generic
List<string> fruit = new List<string>();
fruit.Add("Kiwi")
fruit.Add("Pear")
```

Multiple Binding (Demo)

Data Binding with a Dictionary Collection

- A dictionary collection is a special kind of collection in which every item is indexed with a specific key.
- Two basic dictionary-style collections in .NET:
  - The Hashtable collection (in the System.Collections)
  - The Dictionary collection (in the System.Collections.Generic namespace)
- You create a Dictionary object in much the same way you create an ArrayList or List collection. The only difference is that you need to supply a unique key for every item. (Demo)

Data Binding Using ADO.NET

- Steps using DataSet object:
  - First, create the DataSet.
  - Next, create a new DataTable, and add it to the DataSet.Tables collection.
  - Next, define the structure of the table by adding DataColumn objects (one for each field) to the DataTable.Colums collection.
  - Finally, supply the data. You can get a new, blank row that has the same structure as your DataTable by calling the DataTable.NewRow() method. You must then set the data in all its fields, and add the DataRow to the DataTable.Rows col lection.

(Demo)

Creating a Record Editor (Demo)

Data Source Controls

- Data source controls allow you to create data-bound pages without writing any data access code at all.
- The data source controls include any control that implements the IDataSource interface.
- The .NET Framework includes the following data source controls:
  - SqlDataSource: This data source allows you to connect to any data source that has an ADO.NET data provider.
  - AccessDataSource: This data source allows you to read and write the data in an Access database file (.mdb).

Data Source Controls

- ObjectDataSource: This data source allows you to connect to a custom data access class.
- XmlDataSource: This data source allows you to connect to an XML file.
- SiteMapDataSource: This data source allows you to connect to a .sitemap file that describes the navigational structure of your website.
- EntityDataSource: This data source allows you to query a database using the LINQ to Entities feature.
- LinqDataSource: This data source allows you to query a database using the LINQ to SQL feature, which is a similar (but somewhat less powerful) predecessor to LINQ to Entities.

The Page Life Cycle with Data Binding

- Data source controls can perform two key tasks:
  - They can retrieve data from a data source and supply it to bound controls.
  - They can update the data source when edits take place. (GridView or DetailsView)
- Page Life Cycle Steps:
  - The page object is created (based on the .aspx file).
  - The page life cycle begins, and the Page.Init and Page.Load events fire.
  - All other control events fire.
  - **If the user is applying a change, the data source controls perform their update operations now. If a row is being updated, the Updating and Updated events fire. If a row is being inserted, the Inserting and Inserted events fire. If a row is being deleted, the Deleting and Deleted events fire.**
  - The Page.PreRender event fires.

The Page Life Cycle with Data Binding

- **The data source controls perform their queries and insert the data they retrieve into the bound controls.This step happens the first time your page is requested and every time the page is posted back, ensuring you always have the most up-to-date data.The Selecting and Selected events fire at this point.**
- The page is rendered and disposed.

The SqlDataSource

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server" … />
```

- The **Data Provider factory** has the responsibility of creating the provider-specific objects that the SqlDataSource needs to access the data source.

The Page Life Cycle with Data Binding

- .NET includes a data provider factory for each of its four data providers:
  - System.Data.SqlClient
  - System.Data.OracleClient
  - System.Data.OleDb
  - System.Data.Odbc

```
<asp:SqlDataSource ProviderName="System.Data.SqlClient" … />
```

Put the connection string in web.config and access it in SqlDataSource by using:

```
<%$ ConnectionStrings:[NameOfConnectionString] %>
```

Example: `<asp:SqlDataSource ConnectionString=<%$ ConnectionStrings:[NameOfConnectionString] %> ../>`

Selecting Records

```
<configuration>
        <connectionStrings>
            <add name="Northwind" connectionString=
"Data Source=(localdb)\v11.0;Initial Catalog=Northwind;Integrated Security=SSPI" />
  </connectionStrings>
  …
  </configuration>
```

SqlDataSource supports four properties:

- SelectCommand
- InsertCommand
- UpdateCommand
- DeleteCommand

(Demo)

Parameterized Commands

Uses the @ Symbol for parameters to select from the table.

```
<asp:SqlDataSource ID="sourceProductDetails" runat="server" ProviderName="System.Data.SqlClient" ConnectionString="<%$ ConnectionStrings:Northwind %>"
 SelectCommand="SELECT * FROM Products WHERE ProductID=@ProductID">
 <SelectParameters>
                <asp:ControlParameter ControlID="lstProduct" Name="ProductID"  PropertyName="SelectedValue" />
 </SelectParameters>
</asp:SqlDataSource>
```

Other Types of Parameters:

Source	Control Tag	Description
Control property	<asp:ControlParameter>	A property from another control on the page.
Query string value	<asp:QueryStringParameter>	A value from the current query string.
Session state value	<asp:SessionParameter>	A value stored in the current user's session.
Cookie value	<asp:CookieParameter>	A value from any cookie attached to the current request.
Profile value	<asp:ProfileParameter>	A value from the current user's profile (see Chapter 21 for more about profiles).

Other Types of Parameters

Routed URL value	<asp:RouteParameter>	A value from a routed URL. Routed URLs are an advanced technique that lets you map any URL to a different page (so a request such as `http://www.mysite.com/products/112` redirects to the page `www.mysite.com/productdetails.aspx?id=112`, for example). To learn more about URL routing, refer to the Visual Studio Help or *Pro ASP.NET 4.5 in C#* (Apress).
A form variable	<asp:FormParameter>	A value posted to the page from an input control. Usually, you'll use a control property instead, but you might need to grab a value straight from the Forms collection if you've disabled view state for the corresponding control.

(Demo)

Setting Parameter Values in Code

```
<SelectParameters>
   <asp:ControlParameter Name="CustomerID"
    ControlID="lstCustomers" PropertyName="SelectedValue" />
   <asp:Parameter Name="EarliestOrderDate" DefaultValue="1900/01/01" />
 </SelectParameters>
```

Handling Errors

SqlDataSourceStatusEventArgs.Exception is the property, through which you can access the exception object. (Just made this True)

Example: Code-behind class (Event)

```
protected void sourceProducts_Selected(object sender, SqlDataSourceStatusEventArgs e)
{
        if (e.Exception != null)
        {
                lblError.Text = "An exception occurred performing the query.";
                // Consider the error handled.
                e.ExceptionHandled = true;
        }
}
```

Updating Records

- Just add AutoGenerateEditButton = "True" in Detailsview control (demo)

Same way you can Delete and Insert Data to database.

END OF LECTURE