Code Repository Exercises

Data Binding

Purpose: The purpose of this exercise is to familiarize you with data binding, an ASP.NET feature that allows you to declaratively link various types of data display controls to a data source. You will see the difference between single-value data binding and repeated-value data binding and learn how to implement both. You will also learn to use two different types of bindable data source controls: The SqlDataSource and the ObjectDataSource

Directory Name: DataBinding

Instructions:

1. Create a new .aspx page called *SingleDataBinding.aspx* in the directory created for this exercise. Use single-value data binding to do the following:
    a. Display the current time in the text of a Label control
    b. Display an image URL in the text of a Label, the caption of a Checkbox, the NavigateUrl of a Hyperlink, and the ImageUrl of an Image as shown on page 478 of the MacDonald book (download your own image or copy one from another directory).
2. Create a new .aspx page called *RepeatedDataBinding.aspx* in the directory created for this exercise. Using repeated-value data binding, do the following (see pages 480-486 for examples):
    a. Create an integer array of 10 integers and bind it to a ListBox control
    b. Create an ArrayList of 10 of your favorite foods (as strings) and bind it to a CheckBoxList control
    c. Create a generic list collection of strings that contains your favorite music bands (as strings). Bind it to a RadioButtonList control.
    d. Create a dictionary collection that stores month names as values and month numbers as keys (e.g. 1=January, 2=February, etc.). Bind this collection to a DropDownList control. The Month names should be displayed as the text for each item in the in the DropDownList and the month numbers should be stored as the values.
3. In this step, you will utilize data binding to bind a DropDownList to a DataTable with data from a database. Copy the *AuthorManager3.aspx* page from the ADO.NET Exercise into the directory created for this exercise. Rename it to *AuthorManager_DataBound.aspx*. Modify the FillAuthorList method to bind the *lstAuthor* DropDownList to the DataTable using repeated-value data binding instead of creating the ListItem objects in a loop. (You will need to modify the query to concatenate the author first and last names in a calculated field, and then set the *DataTextField* property to this field.)
4. In this step, you will modify your page to use SQLDataSource controls rather than calling the data access methods by hand. Copy the *AuthorManager_DataBound.aspx* file and paste it in again as a new page. Rename this page to *AuthorManager_SqlDataSource.aspx*. Do the following:

a. Delete all the methods from the code-behind file except for *Page_Load*. Remove the contents of *Page_Load* so that it is an empty method.
b. Remove all of the buttons on the page.
c. Remove all of the Label and TextBox controls from the lower part of the page (those inside the gray)
d. Remove the *onselectedIndexChanged* property from the *lstAuthors* DropDownList
e. Add a SqlDataSource control to the page. Use the Smart tag of this SqlDataSource to configure its properties. Set its *SelectCommand* to the same select query that you wrote for *AuthorManager_DataBound* in step 3 above (this SqlDataSource will have only a SelectCommand).
f. Set the *DataSourceID* property of the *lstAuthors* DropDownList to the ID of the SqlDataSource you just created. Set the *DataTextField* and *DataValueField* properties of the DropDownList to the author's full name and ID, respectively.
g. Add another SqlDataSource to the page. This SqlDataSource should select all columns from the Authors Table where the author ID is equal to the author ID selected in the *lstAuthors* DropDownList (again, this should all be configured using the configuration wizard accessed from the SqlDataSource's smart tag). Click the 'Advanced' button and select the option to auto-generate update, insert, and delete statements. When you have finished with the configuration wizard, look at the source code for the SqlDataSource to see how it is built.
h. Add a DetailsView control to the where you deleted the Labels and Textboxes. (We have not yet covered this control, but we will be covering it soon.) Set the *DataSourceID* property of the DetailsView to the ID of the SqlDataSource you created in the previous step. Set the *AutoGenerateDeleteButton*, *AutoGenerateEditButton*, and *AutoGenerateInsertButton* properties to true.
i. Run the page to verify that it works. You should have full select, update, insert, and delete capabilities (excepting any foreign key constraint problems). Notice how much code you wrote in the code-behind file to enable this!
j. Fine-tune your page a bit by doing the following:
    i. Apply a nice-looking formatting template to the DetailsView
    ii. The labels of each of the fields in the DetailsView contain the database column names, which are not very user friendly. Change these labels to more appropriate names.
    iii. When you insert a new author, the new author does not show up in the lstAuthors DropDownList. Similarly, if you delete an author, the deleted author still shows up in the lstAuthors DropDownList. Why? Fix this so that newly inserted authors show up and deleted authors disappear.

5. In this step, you will replace the SQLDataSource controls with ObjectDataSource controls in order to utilize a business object in a tiered architecture. Copy the *AuthorManager_SqlDataSource.aspx* file and paste it in again as a new page. Rename this page to *AuthorManager_ObjectDataSource.aspx*. Do the following:

a. Create a new stored procedure called *NewGetAllAuthorNames* in the *Pubs* database that performs the modified *select* query you created in step 3 (concatenates the author first and last names in a calculated field and returns this field along with the author id).
b. Create a new method in the *AuthorAccess.cs* class called *NewGetAllAuthorNames* that will call the new stored procedure and return the results in a DataTable object.
c. Remove the first SqlDataSource control from the page and replace it with an ObjectDataSource control. In the configuration wizard of this control, select the *AuthorAccess.NewGetAllAuthorNames* method as the select statement for this control.
d. Bind the *lstAuthor* DropDownList to the new ObjectDataSource control. Show the author full name as the text of each item in the list and store the author ID as the value.
e. Remove the second SqlDataSource control from the page and replace it with a second ObjectDataSource control. This control will be bound to the DetailsView in the lower portion of the page. In the configuration wizard, select the *AuthorAccess* class as the business object, and assign the appropriate methods for select, update, insert, and delete. Be sure to define the source of the ObjectDataSource's select parameter as the selected value of the *lstAuthor* DropDownList control.
f. Bind the DetailsView control to the new ObjectDataSource control.
g. Run the page to verify that everything works.

Header on Default.aspx: Data Binding

Pages linked from Default.aspx: *SingleDataBinding.aspx, RepeatedDataBinding.aspx, AuthorManager_Databound.aspx, AuthorManager_SqlDataSource.aspx*, *AuthorManager_ObjectDataSource.aspx*