1 **DataList and Repeater**

CST272—ASP.NET

2 **The asp:DataList Control (Page 1)**

- DataList is a fully template driven control that acts as a container of *repeated* data items (a "list")
  - Initially displays as a gray box until its DataSourceID property is set
- Repeated items are defined in the <ItemTemplate> element which repeats (loops) one time for each record from the data source

3 **The asp:DataList Control (Page 2)**

- Use *data-binding syntax* with the Eval method in an <ItemTemplate> to repeat through and display the records
  - Coding may use Label Web controls with Text property is assigned the value of the data-binding syntax
  - Alternatively Label controls can be omitted completely by just typing the data-binding syntax

4 **The asp:DataList Control (Page 3)**

- Associating the DataList with a SqlDataSource through its smart tag automatically will create an <ItemTemplate> block
  - Displays the name and value of each data field returned by the data source
  - Identical to the template created when binding a data source to a FormView control through the "Designer"

5 **The asp:DataList Control (Page 4)**

- Associating a SqlDataSource with the DataList does *not* create an <EditItemTemplate> which must be added manually
  - DataList does contain "edit-related" as well as "delete-related" events (but *not* "insert-related" events) so include the Bind method for fields that may be updated
  - Editing and deleting must be implemented with additional manual coding

7 **Templates for DataList**

- The templates for the DataList control are:
  - <ItemTemplate>
  - <AlternatingItemTemplate>
  - <EditItemTemplate>
  - <HeaderTemplate>
  - <FooterTemplate>
  - <SelectedItemTemplate>
  - <SeparatorTemplate>

8 **Formatting the DataList (Page 1)**

- The style elements for the DataList control are:
  - <AlternatingItemStyle>

- <EditItemStyle>
- <FooterStyle>
- <HeaderStyle>
- <ItemStyle>
- <SelectedItemStyle>
- <SeparatorStyle>

9 Formatting the DataList (Page 2)

- Any of the DataList "style elements" specified on the previous page may use formatting properties from the "Properties" window
  - E.g. Font (and all the different font elements), Fore-Color, Back-Color, Gridlines, etc.
- Formatting also can be applied to other individual HTML elements and ASP.NET controls placed within any of the templates

10 Customizing DataList Using AutoFormat

- Formats the DataList by allowing the user to select from a series of *predefined styles*
- Click the "smart tag" for the DataList and click the AutoFormat… command
- Select one of the predefined styles to *preview* it
- Click the <OK> button to implement the style
  - The appropriate properties are set in the source code

11 The RepeatLayout Property (Page 1)

- A feature of DataList control, the RepeatLayout property which is used to specify *customized layouts*
- By default DataList uses "Table" layout to render as an HTML <table> which allows *multiple records* to be displayed per table row
- Layouts include:
  - Table
  - Flow
  - Horizontal
  - Vertical

12 The RepeatLayout Property (Page 2)

- Format:
  DataListControl.<u>RepeatLayout</u> = "Table|Flow|Horizontal|Vertical";
- Example:
  DataListBackOrdered.RepeatLayout = "Table";

13 The RepeatDirection Property

- For the "Table" layout option of the RepeatLayout property, the RepeatDirection property sets which direction records will flow (default is vertical)
- Format:
  DataListControl.<u>RepeatDirection</u> = "Vertical|Horizontal";
- Example:

DataListBackOrdered.RepeatDirection = "Horizontal";

14 **The RepeatColumns Property**

- For the “Table” layout option of the RepeatLayout property, the RepeatColumns property sets the number of columns for the table
  - Default is 0 in which case there will be 1 column
- Format:
  DataListControl.RepeatColumns = *columns*;
- Example:
  DataListBackOrdered.RepeatColumns = 3;

16 **The asp:Repeater Control (Page 1)**

- The Repeater control is a fully template driven control that gives developers total control over the layout of *repeated* data items (a “list”)
  - Repeated items are defined in the <ItemTemplate> element which repeats (loops) one time for each record from the data source
- Serves as sort of a “catch-all” control
  - If there is not an existing control for the desired layout, the Repeater control may be used

17 **The asp:Repeater Control (Page 2)**

- Unlike the DataList, <ItemTemplate> for a Repeater is not created after binding it to a data source
  - Initially displays as a gray box in the “Designer” window until the <ItemTemplate> is defined manually
- Its templates cannot be configured in the “Designer” but must be coded directly in the “Source” window
  - Properties can be assigned in “Properties” window

18 **The asp:Repeater Control (Page 3)**

- Use method Eval in the <ItemTemplate> to repeat through and display the records
  - Method Bind is never used for any of the Repeater templates since there is no record inserting or updating

20 **Templates for the Repeater Control**

- Since there is no inserting or editing of records, the only templates are:
  - <ItemTemplate>
  - <AlternatingItemTemplate>
  - <HeaderTemplate>
  - <FooterTemplate>
  - <SeparatorTemplate>

21 **Formatting Repeater Control**

- Since all coding is manual, there no style-related properties for the Repeater control
- The developer must add manually the needed HTML or CSS content to the Repeater

templates

23 **The DefaultValue Property (Page 1)**

- The DefaultValue property programmatically assigns a value to a SqlDataSource parameter before a SQL statement executes
- Assigned value may come from almost any source, for example:
  - The property of a control on the web form
  - A global variable
  - A system variable such as system date or time

24 **The DefaultValue Property (Page 2)**

- Format:
  SqlDataSourceID. *ParametersType*["*ParameterName*"]. <u>DefaultValue</u> = *value*;
  - Always takes a string type (might need to convert *value*)
- Examples:
  SqlDataSourceInsertVendor. InsertParameters["Vendor"]. DefaultValue =
  TextBoxVendor.Text;
  SqlDataSourceInsertVendor. InsertParameters["Date"].DefaultValue =
  DateTime.Now.ToString();

25 **The SqlDataSource Insert() Method**

- Inserts a new record (or records) into the table represented by a SqlDataSource control
  - An InsertCommand must have been configured
  - Values must have been previously assigned to all InsertParameters for fields that require a value
- Returns the number of records inserted
- Format:
  [[int] *value*] = *SqlDataSource*.<u>Insert</u>();
- Example:
  SqlDataSourceInsertVendor.Insert();

26 **The SqlDataSource Update() Method**

- Updates one or more records in the table represented by a SqlDataSource control
  - An UpdateCommand must have been configured
  - Values must have been previously assigned to all UpdateParameters for fields that require a value
- Returns the number of records updated
- Format:
  [[int] *value*] = *SqlDataSource*.<u>Update</u>();
- Example:
  SqlDataSourceUpdateVendor.Update();

27 **The SqlDataSource Delete() Method**

- Deletes one or more records in the table represented by a SqlDataSource control
  - A DeleteCommand must have been configured
  - Values must have been previously assigned to all DeleteParameters for fields that require a value
- Returns the number of records deleted
- Format:
  [[int] *value*] = *SqlDataSource*.Delete();
- Example:
  SqlDataSourceDeleteVendor.Delete();

28 **SQL I/O Exceptions (Page 1)**

- Since the SQL Insert, Update, and Delete methods all are *file* operations, always use them within a try…catch exception handling block

29 **SQL I/O Exceptions (Page 2)**

- Example:

```
try
{
   SqlDataSourceInsertVendor.Insert();
}
catch (Exception ex)
{
   LabelError.Text = ex.Message;
}
```

31 **Control Parameters (Page 1)**

- An asp:ControlParameter is an alternate type of parameter that may be used for parameterized SQL statements within a SqlDataSource control
- Values are assigned *automatically* from controls on the web form to the parameters in the SQL statement when that statement executes
  - The assigned value cannot come from another source besides as ASP.NET web form control

32 **Control Parameters (Page 2)**

- Properties in addition to those that are specified in a standard asp:Parameter:
  - ControlID—ID of a control (such as a TextBox) from which the value comes
  - PropertyName—specifies the property from that control which provides the value, e.g. Text
- Format:

```
<asp:ControlParameter Name="parameterName"
   Type="type"
   ControlID="controlID"
   PropertyName="property" />
```

33 **Control Parameters** **(Page 3)**

- Example:

```
<asp:SqlDataSource … >

  <InsertParameters>
    <asp:ControlParameter Name="Vendor"
      Type="String"
      ControlID="TextBoxVendor"
      PropertyName="Text" />
    …
  </InsertParameters>

</ asp:SqlDataSource >
```