1 **Data Bound List Controls, DetailsView and FormView**

CST272—ASP.NET

2 **The ASP:DropDownList Web Control (Page 1)**

- The asp:DropDownList Web control creates a Form field that allows users to select from drop-down lists of options
- Format:

  <asp:DropDownList id = "*ControlName*" runat = "server"> </asp:DropDownList>

3 **The ASP:DropDownList Web Control (Page 2)**

- The list (an Items collection) can be populated in one of the following two ways:
  - In the "Properties" window, click the Build button [...] for the Items collection to use the "ListItem Collection Editor" (*review*)

4 **The ASP:DropDownList Web Control (Page 3)**

- The list (an Items collection) can be populated in one of the following two ways (*con.*):
  - Assign the asp:DropDownList's DataSource property to a SqlDataSource object (or some other data source)
    - Easiest way is to use the "smart tag"
    - ... but it also can be set in "Properties" window
    - ... or typed manually in the Source code

5 **The ASP:DropDownList Web Control (Page 4)**

- Properties for the asp:DropDownList Web control with a data source:
  - DataSourceID—name of the SQLDataSource (or some other data source) that provides the items for the list
  - DataTextField—column from the SELECT list of the data source that is the *text displayed* for each item

6 **The ASP:DropDownList Web Control (Page 5)**

- Properties for the asp:DropDownList Web control with a data source (*con.*):
  - DataValueField—column from the SELECT list of the data source that is the value (text) passed to the server for processing when an item is selected
    - This text also is assigned to the value attribute in the rendered HTML code

8 **The ASP:CheckBoxList Web Control (Page 1)**

- The asp:CheckBoxList Web control provides a single-selection check box *group* that can be generated dynamically through data binding
  - Check boxes only grouped *physically*, not *logically*
  - Individual check boxes (asp:CheckBox) must each be created and configured separately

9 **The ASP:CheckBoxList Web Control (Page 2)**

- Contains an Items collection with members that correspond to individual items on the list

- Same as the asp:DropDownList Web control
- Format:
  <asp:CheckBoxList ID = "*ControlName*" runat = "server"> </asp:CheckBoxList>

10 **The ASP:CheckBoxList Web Control (Page 3)**

- The list (an Items collection) may be populated in one of the following two ways:
  - In the "Properties" window, click the Build button [...] for the Items collection to use the "ListItem Collection Editor"
  - Assign the asp:CheckBoxList's DataSource property to SqlDataSource object (or some other data source)

11 **The ASP:CheckBoxList Web Control (Page 4)**

- Properties for the asp:CheckBoxList Web control with a data source:
  - DataSourceID—name of the SQLDataSource (or some other data source) that provides the items for the list
  - DataTextField—column from the SELECT list of the data source that is the text displayed to the right of each item

12 **The ASP:CheckBoxList Web Control (Page 5)**

- Properties for the asp:CheckBoxList Web control with a data source (*con.*):
  - DataValueField—column from the SELECT list of the data source that is the value (text) passed to the server for processing when an item is selected
    - This text also is assigned to the value attribute in the rendered HTML code

14 **The ASP:RadioButtonList Web Control (Page 1)**

- The asp:RadioButtonList Web control provides a single-selection radio button *group* that can be generated dynamically through data binding
  - Individual radio buttons (asp:RadioButton) must each be created and configured separately

15 **The ASP:RadioButtonList Web Control (Page 2)**

- Contains an Items collection with members that correspond to individual items on the list
  - Same as the asp:DropDownList web control
- Format:
  <asp:RadioButtonList ID = "*ControlName*" runat = "server"> </asp:RadioButtonList>

16 **The ASP:RadioButtonList Web Control (Page 3)**

- The list (an Items collection) may be populated in one of the following two ways:
  - In the "Properties" window, click the Build button [...] for the Items collection to use the "ListItem Collection Editor"
  - Assign the asp:RadioButtonList's DataSource property to a SqlDataSource object (or some other data source)

17 **The ASP:RadioButtonList Web Control (Page 4)**

- Properties for the asp:RadioButtonList Web control with a data source:

- DataSourceID—name of the SQLDataSource (or some other data source) that provides the items for the list
- DataTextField—column from the SELECT list of the data source that is the text displayed to the right of each item

18 **The ASP:RadioButtonList Web Control (Page 5)**

- Properties for the asp:RadioButtonList Web control with a data source (*con.*):
  - DataValueField—column from the SELECT list of the data source that is the value (text) passed to the server for processing when an item is selected from the list
    - This text also is assigned to the value attribute in the rendered HTML code

20 **The RepeatColumns and RepeatDirection Properties**

- By default CheckBoxList and RadioButtonList controls are inserted into the cells of a <table>
  - Displayed vertically one row and one cell per item
- RepeatColumns property, if set to a value greater than 1, specifies that items span a number of columns top to bottom, or left to right
- RepeatDirection property specifies direction columns repeat, Vertical (default—spans top to bottom) or Horizontal (spans left to right)

21 **The SelectedIndexChanged Event for List Web Controls (Page 1)**

- Event that fires whenever a *new item is selected* for List Web controls (DropDownList, CheckBoxList, RadioButtonList, etc.)
- However it only will fire after a post-back occurs (by default only a <Submit> button can be clicked for event handler method to execute)

22 **The SelectedIndexChanged Event for List Web Controls (Page 2)**

- The onSelectedIndexChanged *property* is assigned for a List web control at design time when the control is double-clicked to create the event
- To automatically post-back at the occurrence of a SelectedIndexChanged event
  - Click smart tag for a data-bound list control
  - Then click the AutoPostBack check box "on"

24 **The SelectedItem Property for the Data-Bound List Controls**

- In Visual C# the SelectedItem property represents the Item currently selected from any of the data-bound list controls when the form is submitted
- Format:

  ListControlID.SelectedItem.*Property*
- A way of referencing the properties of the currently item within the "Code Editor" (C# code), e.g.:

  DropDownListProduct.SelectedItem.Text

  RadioButtonListProduct.SelectedItem.Value

25 **The SelectedValue Property for the Data-Bound List Controls**

- The SelectedValue property represents text assigned to the Value property for the

selected item for any of the data-bound list control when the form is submitted

- Format:
  ListControlID.SelectedValue
- Example:
  DropDownListProduct.SelectedValue

26 **The SelectedIndex Property for the Data Bound List Controls**

- A zero-based index (integer) representing the currently selected item in any of the data bound list controls when the form is submitted
  - If no item currently is selected, the value is set to -1
- Format:
  ListControlID.SelectedIndex
- Example:
  if (DropDownListProduct.SelectedIndex == -1)

28 **The foreach Loop (Page 1)**

- The foreach statement in Visual C# is used for iterating (looping) through a collection
- The collection could be an array or items in a data-bound list control

29 **The foreach Loop (Page 2)**

- Format:

```
foreach (ListItem listItemName in ListObject.Items)
{
    Statements that reference listItemName
}
```

  - *listItemName* is an object variable of type ListItem that holds the information about an individual collection items during each repetition of the foreach loop
  - *ListObject* in this case is name of the DropDownList or RadioButtonList or CheckBoxList (or any collection)

30 **The foreach Loop (Page 3)**

- Example:

```
foreach (ListItem ListItemProduct in CheckBoxListProduct.Items)
{
    LabelResults.Text += ListItemProduct.Value + "<br />";
}
```

31 **The ASP:DetailsView Web Control (Page 1)**

- Displays the values of a single record at a time from a data source in a table
  - Each data row represents one field within the record
  - The asp:DetailsView control allows users to edit, delete, and *insert* records

32 **The ASP:DetailsView Web Control (Page 2)**

- To assign the data source:
  - Click the "smart tag" and select a data source (e.g. a SQLDataSource object) from

the Choose Data Source: drop-down list
- ☐ The rows are formatted automatically to represent the fields of the selected query (data source)
- ☐ The source code is updated automatically to include a <Fields> block and <asp:BoundField> tags which represent the data

33 **The ASP:DetailsView Web Control (Page 3)**

- When a data source is selected for the DetailsView control, the following of its properties are updated automatically:
  - AutoGenerateColumns—how are fields generated? (set to False):
    - ☐ True—generated from the data source at run-time
    - ☐ False—permanently defined in the <Fields> block

34 **The ASP:DetailsView Web Control (Page 4)**

- When a data source is selected for the DetailsView control, the following of its properties are updated automatically (*con.*):
  - DataKeyNames—the DataField property of the BoundColumn(s) that uniquely identify each record (by default the *primary key*)
  - DataSourceID—the ID of its data source controls

35 **The ASP:DetailsView Web Control (Page 5)**

- Formatting for the DetailsView control is similar to that of the GridView with the exceptions:
  - There is no SelectedRowStyle property since the control cannot have a specific "selected row"
  - There is an InsertRowStyle property used when a new record is being inserted into the table

38 **The ASP:DetailsView Web Paging Property Settings (Page 1)**

- Controls whether or not users may page from one record to another
- The properties are:
  - AllowingPaging
    - ☐ True—paging is supported and the pager row (the paging controls) are displayed
    - ☐ False—paging is not supported (default)

39 **The ASP:DetailsView Web Paging Property Settings (Page 2)**

- The properties are:
  - PagerSettings—drill-down properties that are used to customize the pager row
    - ☐ Mode—options are NextPrevious, Numeric (displays record numbers—the default), NextPreviousFirstLast, and NumericFirstLast
    - ☐ The other properties determine the *text* and/or *image* for the Next, Previous, First and Last buttons

40 **The ASP:DetailsView Web Paging Property Settings (Page 3)**

- The PagerStyle property drills-down and allows for formatting (BackColor, ForeColor, Font, etc.) of the pager row (can be created with AutoFormat)

42 **Inserting Data with DetailsView (Page 1)**

- The GridView control does not provide a method for inserting new records
- In DetailsView records can be *inserted* by adding a New button or link at row following the fields

43 **Inserting Data with DetailsView (Page 2)**

- When the New button/link is clicked:
  - The page is posted back and all editable fields are converted to TextBoxes for inserting data
  - During inserting the New button converts to two buttons, Insert and Cancel

44 **Inserting Data with DetailsView (Page 3)**

- When the Insert button is clicked:
  - The page is posted back again
  - The data source's InsertCommand is executed
  - The DetailsView retrieves that updated table from the data source and redisplays the data

45 **Inserting Data with DetailsView (Page 4)**

- If the Cancel button is clicked:
  - The page is posted back, but DetailsView is updated from the unmodified data source
- To configure DetailsView for inserting, click smart tag and click "on" the Enable Inserting checkbox

46 **Inserting Data with DetailsView (Page 5)**

- Customizing is similar to GridView control:
  - ButtonType—either Button, Image or Link (default)
  - InsertImageUrl—an image if ButtonType is Image
  - InsertText—text if ButtonType is Button or Link

47 **Inserting Data with DetailsView (Page 6)**

- Customizing is similar to GridView control (*con.*):
  - InsertVisible—Boolean property which determines if the editing TextBox is visible (automatically set to False for Auto-Increment/Auto-Number fields)
  - Additional Font and other stylistic properties

48 **The DataFormatString Property (Page 1)**

- A property of a <BoundField> that sets a string which specifies the display format for value of the field (for numeric and dates fields)
- If not set, displays uses the default format for the value of the field
- Can be set by clicking "Smart Tag" for the GridView control, selecting "Edit Columns …" and entering DataFormatString value

- Or in "Properties" window

49 **The DataFormatString Property (Page 2)**

- Format:
  DataFormatString="{0:*formatCharacters*}">
  - Some *formatCharacters* examples:
    - {0:C}—*Currency* in which 123.456 displays as $123.46
    - {0:N}—*Number* in which 1234.456 displays as 1,234.46
    - {0:N1}—*Number with decimal positions* in which 123.456 displays as 123.5

50 **The DataFormatString Property (Page 3)**

- Format (*con.*):
  DataFormatString="{0:*formatCharacters*}">
  - Some *formatCharacters* examples:
    - {0:d}—*Short date* in which 6/15/2013 1:45:30 PM displays as 6/15/2013
    - {0:D}—*Long date* in which 6/15/2013 1:45:30 PM displays as Monday, June 15, 2009

51 **The DataFormatString Property (Page 4)**

- Format:
  DataFormatString="{0:*formatCharacters*}">
- Example:
  <asp:BoundField DataField="Subtotal"
  DataFormatString="{0:N}" />

53 **Query Strings (Page 1)**

- A query string is part of a URL that used to send the values from one webpage to other page (like passing arguments)
- Consists of a series of "key and value" pairs
  - The key is the "variable" name
  - The value is a value assigned (=) to the key
- Multiple keys and values may be joined by using the ampersand (&) character and passed together in the same URL, e.g.:
  - *key1=value1*[&*key2=value2* ...]

54 **Query Strings (Page 2)**

- In a URL the key and value pairs follow the Web address and a question mark (?) symbol:
  http://*WebAddress*?*key1=value1*[&*key2=value2* ...]
  - Each of the *key* and *value* pairs are separated by an ampersand (&) symbol
- Example:
  http://www.MyWebSite.com/VendorDetailsView.aspx?VendorNumber=20001
- Example when the file has the same path (location):
  VendorDetailsView.aspx?VendorNumber=20001

55 **QueryStringParameters in a SqlDataSource Select Command (Page 1)**

- A QueryStringParameter for a SELECT command automatically retrieves values from the query string
- When configuring the SqlDataSource:
  - Click the <WHERE...> button and select the Column (field) for the WHERE clause and the Operator
  - From the Source drop down list select "Query String"
  - Enter the name (*key* from the query string) into the QueryString field textbox

56 **QueryStringParameters in a SqlDataSource Select Command (Page 2)**

- Process inserts an asp:QueryStringParameter into the SqlDataSource's <SelectParameter> block and sets its QueryStringField property to query string *key*
- Example:

```
<SelectParameters>
   <asp:QueryStringParameter Name="VendorID"
      QueryStringField="VendorID"
      Type="String" />
</SelectParameters>
```

59 **Using Binding for a Lookup Table (Page 1)**

- A common scenario with data-bound controls is to enable users to update or insert a value by selecting it from a lookup table using a DropDownList control or other list control.
- In that case, the lookup control is bound to a *separate data source* that returns the list of possible values, and the lookup control's selected value is bound to a field in the parent data-bound row.

60 **Using Binding for a Lookup Table (Page 2)**

- Add this functionality as follows:
  1. First, for the lookup control, add a list control (either a DropDownList or ListBox control) to a template inside a data-bound control
     - E.g. a Grid View, DetailsView, or FormView control
  2. Bind the SelectedValue property of lookup control to the related field in the container control's data source
  3. Then set lookup control's DataSourceID property to a data source control that retrieves lookup values

61 **Using Binding for a Lookup Table (Page 3)**

- Add this functionality as follows (*con.*):
  4. Set the lookup control's DataTextField property to the field from the lookup table that contains the values to display (and set its DataValueField property to the field from the lookup table that contains the unique identifier for the lookup value, if applicable)

62 **Using Binding for a Lookup Table (Page 4)**

- The following example shows DropDownList control that is included in InsertItemTemplate of a DetailsView control:
  - The SelectedValue property of the DropDownList control uses the Bind method for two-way binding to the VendorNumber field of the current row for the DetailsView control
  - The DataSourceID property of the DropDownList control is set to a separate data source control that retrieves the list of possible VendorNames and VendorNumbers

65 **Using Binding for a Lookup Table (Page 5)**

- The following example shows DropDownList control that is included in InsertItemTemplate of a DetailsView control (*con.*):
  - The DataTextField property of the DropDownList control is set to the VendorName field from the lookup data source so that a list of possible VendorNames is displayed
  - The DataValueField property of the DropDownList control is set to the VendorNumber field from the lookup data source for the related VendorName

66 **Using Binding for a Lookup Table (Page 6)**

- The following example shows DropDownList control that is included in InsertItemTemplate of a DetailsView control (*con.*):
  - When a user selects a VendorName from the list, the SelectedValue property of the DropDownList control is set to the VendorNumber for the selected VendorName

68 **The ASP:FormView Web Control (Page 1)**

- The asp:FormView Web control displays one record at a time from its data source (similar to DetailsView)
- FormView is an entirely *template-based* data control
  - All elements are coded manually within templates

69 **The ASP:FormView Web Control (Page 2)**

- Format:

```
<asp:FormView ID = "ControlName"
   runat = "server"
   DataSourceID = "SqlDataSourceID">
      FormViewTemplates
</asp:FormView>
```

- Located in the Data group in the "Toolbox"

70 **The ASP:FormView Templates (Page 1)**

- The FormView templates include:
  - ❑EditItemTemplate—renders format of the editing interface (if the ListView is configured for editing)
  - ❑EmptyDataTemplate—renders if no records are returned from the data source
  - ❑FooterTemplate—rendered once after all other templates if present

- ❑HeaderTemplate—rendered once before all other templates if present

71 **The ASP:FormView Templates (Page 2)**

- The FormView templates include (*con.*):
  - ❑InsertItemTemplate—renders format of the inserting interface (if the FormView is configured for inserting)
  - ❑ItemTemplate—renders the layout for one record as returned from the data source (the only *required* template—there is no LayoutTemplate)
  - ❑PagerTemplate—used for rendering a paging template)