1 **Introduction to ASP.NET and C#**

CST272—ASP.NET

2 **ASP.NET Server Controls (Page 1)**

- Server controls can be Buttons, TextBoxes, etc.
- In the source code, ASP.NET controls are identified with the prefix asp: followed by the control name, plus a runat="server" attribute and value, e.g.
  <asp:Button runat="server">
- Server controls use a *tag* formatting similar to that of HTML tags

3 **ASP.NET Server Controls (Page 2)**

- Some types of ASP.NET Server Controls
  – ASP.NET Form Controls (Web controls)
  – Data Validation Controls
  – Data Controls
  – Mobile Controls (run on mobile devices)

5 **Browser Source Code**

- What does the Web server send to the browser? (Look at the browser's source code for an ASP.NET page):
  – ASP.NET code is *never* sent to the browser
  – Instead the ASP.NET code runs on the Web server and builds an HTML document
  – Only HTML tags, along with client-side scripts (e.g. JavaScript statements) are sent to the browser

6 **Adding an Existing Item**

- Existing items might include Web Forms, HTML Pages, Web User Controls, image files, etc.
- Before starting, be certain folder that the item(s) are to be inserted into is selected in "Solution Explorer"
  1. Select Website from the menu bar
  2. Select Add Existing Item... from the Website menu
  3. Browse to Look in: folder
  4. Select Files of type: from drop-down list
  5. Select files and click <Add> button; the item(s) is/are added to specified folder in Solution Explorer

8 **Dragging Image from Solution Explorer**

- In either Source or Design view, drag an image from the Solution Explorer window into the document
- Creates an HTML <img> tag in the document

10 **Properties Window**

- Sets the properties for objects, controls, classes and other project components
- Properties can be updated:

- At design time by changing their values in the Properties Window
- By writing Visual C# assignment statements in the program code behind the page, e.g.

```
TextBoxAge.Text = "30";
```

11 **Setting HTML Attributes in the Properties Window**

- In either Source or Design view, select the element to be updated
- Enter the new values in the "Properties Window"
- The values are assigned as attributes and values, or created as a "style" element

12 **Using the Style Attribute for Formatting**

- In either Source or Design view, select the element to be formatted
- For the "Style" property in the "Properties" window, click the Build [...] button which opens the "Modify Style" dialog window
- Select "Category" and set the attributes and values
- Creates a style attribute for the selected tag, e.g.

```
<hr style="position: absolute; top: 89; left: 0" />
```

14 **Creating a Table from the Menu Bar**

- In Design view select Table from the menu bar and Insert Table from the "Table" menu
- In the "Insert Table" dialog window select number of Rows and Columns, and any other attributes
- Adds <table>, <tr> and <td> tags to the document

17 **Formatting with the Formatting Toolbar**

- In Design view select the text to be formatted ...
  - The "Styles" dropdown list can be use to insert heading styles (<h1> to <h6>) and a wide variety of other styles into the document
  - The "Font" dropdown list can be used to implement any available typeface to the document (inserts "style rule" into the <head> section

19 **The Toolbox**

- Provides access to commonly used controls, e.g. the ASP.NET Server controls
- Can be hidden and made to slide out (true of several other windows as well)—the Auto Hide feature
  - Hover over the text "Toolbox" in the left side of the IDE window and it slides into view (the default mode for the Toolbox is hidden)
  - Click the Auto Hide icon (push pin) to turn the feature on and off

20 **The asp:Label Control (Page 1)**

- The asp:Label control displays "stand-alone" text that may not be manually updated by the user
- It has a series of properties used for *formatting* the contents of its Text property
- These properties all may be updated:
  - By modifying them in the "Properties" window

- By assigning (keying) the properties and values in the "Source" code window
- Programmatically using an assignment statement in the "Code Editor"

21 **The asp:Label Control (Page 2)**

- Format:
  <asp:Label ID="*LabelID*" runat="server" Text="*Label text*" [*formatting properties*] ></asp:Label>
- Example:
  <asp:Label ID="LabelTHR" runat="server" Text=""></asp:Label>

22 **The Text Property**

- The Text property means different things for different Web controls:
  - Button—the text that appears on the Button
  - TextBox—a String that represents the *current value* of the TextBox which is the value that is displayed inside the box
  - Label—the value that is displayed for the Label, often the result processing

23 **Dragging ASP.NET Web Controls from the ToolBox**

- In Source or Design view, from the ToolBox drag an ASP.NET Web control into the document
- Places the tag into the document

25 **The asp:TextBox Control (Page 1)**

- The asp:TextBox control displays a box into which the user may type input which effectively modifies the control's Text property
- This Text property as well as other properties of the TextBox may be updated:
  - By modifying them in the "Properties" window
  - By assigning (keying) the properties and values in the "Source" code window
  - Programmatically using an assignment statement in the "Code Editor"

26 **The asp:TextBox Control (Page 2)**

- Format:
  <asp:TextBox ID="*TextBoxID*" runat="server" Text="*TextBox text*" [*formatting properties*] ></asp:TextBox>
- Example:
  <asp:TextBox ID="TextBoxAge" runat="server" Text=""></asp:TextBox>

28 **Dragging HTML Controls from the ToolBox**

- In Source or Design view, open HTML group in the ToolBox and drag an HTML control into the document
- Places the tag into the document

31 **The asp:Button Control (Page 1)**

- The Button control is used to display a push button
  - May be a submit button or a command button (by default it is a submit button)

- It posts the page back to the server when it is clicked
- It is possible to write an event handler (a code block that executes) to control the actions performed when the submit button is clicked

32 **The asp:Button Control (Page 2)**

- Format:
  <asp:Button id="*ButtonID*" runat="server" Text="*Button Text*" />
- Example:
  <asp:Button id="ButtonSubmit" runat="server" Text="Submit" />

34 **Creating Bulleted Lists from the Formatting Toolbar**

- In Design view click the <Bullets> button on the "Formatting Toolbar" and begin typing the text for each bulleted item
- Press the <Enter> key after each line to create each subsequent bulleted item
- Creates the <ul> and <li> tags in the document
- Click the <Numbering> button on the "Formatting Toolbar" to create a numbered list
  – Inserts <ol> and <li> tags into document

36 **Creating Hyperlinks from the Formatting Toolbar**

- In Design view select text (or object, e.g. an image) to be formatted as a hyperlink
- Click the <Hyperlink> button on the "Formatting Toolbar"
- In the "Hyperlink" dialog window:
  – Select the hyperlink Type:, e.g. "mailto" or "http"
  – Type the Text: which is the rest of the URL
- Adds the <a> tag with href attribute to the document

38 **Creating Formatting Styles**

- In Design view, from the Format menu select the New Style command
- In the "New Style" dialog window either:
  – Type a CssClass attribute value that matches one that has been assigned to an element in the document
  – From the dropdown menu, select an HTML tag, e.g. a Selector
- Select "Category" and set the attributes and values
- Inserts a "style rule" into the <head> section for that CssClass or selector

41 **The Code Behind the Page (Page 1)**

- Code associated with the Form (and which gives it dynamic capability) is written in an ASP.NET-compatible language, e.g., Visual C#
- This code generally is stored in a separate file from the Form with the same name as the Web Form but with the extension ".aspx.cs"
  – E.g. "WebForm2.aspx.cs"

42 **The Code Behind the Page (Page 2)**

- Compiled code behind the page is the class definition for the page
- When the application is built, the code is compiled into an executable file stored in the

bin directory

47 **Events and Event Handlers (Page 1)**

- For almost every object on the Form, the browser can respond to many *mouse* and *keyboard* actions
- Some examples:
  - Click, MouseDown, MouseMove, KeyPress, Validating, Validated

48 **Events and Event Handlers (Page 2)**

- To *automatically* create template including a header and {braces} for an event handler method template for any control, *double-click* on the object
  - Default event for an asp:Button is Click
  - Default event for the Web Form is Load

49 **Events and Event Handlers (Page 3)**

- For each event handler method, double-clicking and creating the method also "registers" a property within the control that links it to the method, e.g.:
  <asp:Button runat="server" Text="Submit" OnClick="ButtonTHR_Click">

50 **Syntax of Event Handler Method Header (Page 1)**

- Format:
  protected void *MethodName*(object sender, *EventClassType* e)
  - *methodName* by default consists of the concatenation of the *object name (ID)* and *event type*
- Event handler methods have two parameters:
  - *sender*: a reference which identifies the object (control) which initiated the call to the method
  - *e*: a reference that stores property values and methods related to the event, e.g. the "Click" event

51 **Syntax of Event Handler Method Header (Page 2)**

- Format:
  protected void *MethodName*(object sender, *EventClassType* e)
- Example:

```
protected void ButtonTHR_Click(object sender, EventArgs e)
{
    ...
}
```

53 **The Convert Class (Page 1)**

- The C# Convert class contains a rich set of static methods for converting from one type to another
- Format:
  Convert.To*Type*(*expression*)
- Some (of dozens of) examples:

Convert.ToInt32(*expression*) // an 'int'
Convert.ToUInt32(*expression*) // U = unsigned
Convert.ToDouble(*expression*)
Convert.ToDateTime(*expression*)
Convert.ToString(*expression*)
(etc.)

54 **The Convert Class (Page 2)**

- Example 1:
  int age = Convert.ToInt32(TextBoxAge.Text);
  – Converts a String from a TextBox to a 32-bit integer
  – Possible Java (JavaFX) equivalent might be:
    String stringAge = textFieldAge.getText();
    int age = Integer.parseInt(stringAge);

55 **The Convert Class (Page 3)**

- Example 2:
  int age = Convert.ToInt32(TextBoxAge.Text);
  – Converts a String from a TextBox to a 32-bit integer
  LabelAge.Text = Convert.ToString(ageInteger);
  – Converts an integer to a String
  – There is no simple Java equivalent