

ISTANBUL KÜLTÜR UNIVERSITY, DEPARTMENT OF COMPUTER ENGINEERING
CSE5001 WEB PROGRAMMING LAB MANUAL

Lab 05: Introduction to ASP.NET

Objective: To Acquire the basics of building an ASP.NET Web application.

Summary: ASP.NET is a server side framework/language for developing web application. ASP.NET web forms are event driven pages with server controls and server events.

ASP.NET Web Forms lets you build dynamic websites using a drag-and-drop, event driven model. (event-driven programming is a programming paradigm in which the flow of the program is determined by events such as user actions (*mouse clicks, key presses*), sensor outputs, or messages from other programs/threads.)

ASP.NET Page Life Cycle:

When a page is requested, it is loaded into the server memory, processed, and sent to the browser. Then it is unloaded from the memory. At each of these steps, methods and events are available, which could be overridden according to the need of the application.

The page life cycle phases are:

- Initialization
- Instantiation of the controls on the page
- Restoration and maintenance of the state
- Execution of the event handler codes
- Page rendering

ASP.NET Page Life Cycle Events

At each stage of the page life cycle, the page raises some events, which could be coded. An event handler is basically a function or subroutine, bound to the event, using declarative attributes such as OnClick or handle.

Following are the page life cycle events:

PreInit is the first event in page life cycle. It checks the IsPostBack property and determines whether the page is a postback. It sets the themes and master pages, creates dynamic controls, and gets and sets profile property values. This event can be handled by overloading the OnPreInit method or creating a Page_PreInit handler.

Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page_Init handler.

InitComplete event allows tracking of view state. All the controls turn on view-state tracking.

LoadViewState event allows loading view state information into the controls.

LoadPostData During this phase, the contents of all the input fields are defined with the <form> tag are processed.

PreLoad occurs before the post back data is loaded in the controls. This event can be handled by overloading the OnPreLoad method or creating a Page_PreLoad handler.

The Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page_Load handler.

LoadComplete, the loading process is completed, control event handlers are run, and page validation takes place. This event can be handled by overloading the OnLoadComplete method or creating a Page_LoadComplete handler.

The PreRender event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.

PreRenderComplete, As the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.

SaveStateComplete, State of control on the page is saved. Personalization, control state and view state information is saved. The HTML markup is generated. This stage can be handled by overriding the Render method or creating a Page_Render handler.

The UnLoad phase is the last phase of the page life cycle. It raises the UnLoad event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as database connections, are freed. This event can be handled by modifying the OnUnLoad method or creating a Page_UnLoad handler.

Basic Server Controls:

There are 3 different types of Server Controls in ASPNET webforms.

- Html server controls
- Web server controls
- Validations server controls.

The syntax for using server controls is:

```
<asp:controlType ID ="ControlID" runat="server" Property1=value1 [Property2=value2] />
```

In order to be a server control regardless of type the tag must contain the **runat="Server"** attribute.

Html server controls: The form runat server attribute in the aspx page is an HTML server control. This allows you to interact html controls in a server side code. Or you can add runat attribute to any elements like **div**

Web server controls: these controls are named things like **buttons, textboxes, labels, dropdowns** and allow you to interact with create events for these controls. You can also bind data to these controls for dynamic content.

Validations server controls: allow you to validate the user input data to ensure that useless, unauthenticated, or contradictory data don't get stored.

Procedure 01: Building the first ASP.NET web page

1. Open Visual Studio for Web.
2. In the **File** menu, click **New Web Site**.
The **New Web Site** dialog box is displayed.
3. Under **Installed**, click **Visual C#** and then select **ASP.NET Empty Web Site**.
4. In the **Web location** box, select **File System**, and then enter the name of the folder where you want to keep the pages for your website.
For example, type the folder name **C:\Lab05_FirstASPNET**. Click **OK**.
5. In **Solution Explorer**, right-click the root of your Web site, and then click **Add New Item**.
6. Select **Web Form**, name the file **HelloWold.aspx**, and then click **Add**.
7. We will add a **Label** control to display our message in the page.
 - A Label control is somewhat simple, since it's just used to hold a piece of text. Add the following piece of HTML-looking code somewhere between the set of <form> tags

```
<asp:Label runat="server" id="lblHelloWorld"></asp:Label>
```

11. You can write the page logic in this file. This file is the code behind file. Switch to code behind and write the following code block on **Page Load**
 - Every control has specific properties and methods. You can set control properties to modify the appearance or behavior of controls. For example, you can set the **font**, **color**, and **size** of a control. You can use the control methods to perform a specific task, such as moving a control. You can set control properties at design times by using the **Properties window** or at run time by using the code. Every control has a property called **ID** that is used for the unique identification of the control (eg. *lblHelloWorld*). You can set the property of a control at run time by using the following syntax:

```
lblHelloWorld.Text = "Hello, world!"; //Text property represents the value assigned to control
```

*You use the **Label** control to display static text in a Web page that users cannot edit. When you add a Label control, the text "Label" appears as its caption. However, you can use the **Text** property to modify the caption.*

8. Run the project (F6), and have a look.

Procedure 02: Getting User Input and Setting it.

*(open a new blank web page with the name **BasicControls.aspx**)*

8. We will add a **TextBox**, **Button**, **Label** and **Literal** control to get an input from user and displaying it. Drag and drop these controls to the page.

```
<asp:Label ID="Label1" runat="server" Text="Enter your name"></asp:Label>
<asp:TextBox ID="txtInput" runat="server"></asp:TextBox><br />
<asp:Button ID="btnDisplayMessage" runat="server" Text="Display"/>
<asp:Literal ID="LiteralText" runat="server" Text=" "></asp:Literal>
```

9. If you are in Source view, click the **Design** view button in the bottom of VS. Now you will see a visual representation of our page. We wish to add a Click event to the button, doubleclick the **Display**, and you will be taken to the *CodeBehind* file of our page. As you can see, a new method has been added, called **btnDisplayMessage_Click**. Now let's add some code to our event. We wish to get the text from the TextBox, to display it on Literal control.

```
protected void btnDisplayMessage_Click(object sender, EventArgs e)
{
    LiteralText.Text = "Hello, " + txtInput.Text;
}
```

10. Run the project (F6), and you will see TextBox control, try entering a name in the textbox, and press the button.

Procedure 03: Usage of CheckBox and CheckBoxList controls

*(open a new blank web page with the name **CheckBoxControl.aspx**)*

8. Drag and Drop a **CheckBox** and **CheckBoxList** on the page.
9. Click the ellipsis button for the Items property of the CheckBoxList control.
10. In the ListItem Collection Editor dialog box, click Add to create a new item. A new item is created and its properties are displayed in the Properties pane of the dialog box.

11. Verify that the item is selected in the Members list, and then set the item properties. Each item is a separate object and has following properties:
- **Text**: Represents the text to be displayed for the item in the list.
 - **Value**: Represents the value associated with the item without displaying it.
For example, you can set the Text property of an item as the city name and the Value property to the postal code of the city. Thus, you can keep the Text and Value properties different when you do not want the actual value to be displayed to the user.
 - **Selected**: A Boolean value that indicates whether or not the item is selected.
12. Html view;

```
<!-- This is a CheckBoxList -->
<asp:CheckBoxList ID="chk1stCountry" runat="server">
    <asp:ListItem Text="Austria" Value=" Austria"></asp:ListItem>
    <asp:ListItem Text="Belgium" Value=" Belgium"></asp:ListItem>
    <asp:ListItem Text="China" Value=" China"></asp:ListItem>
    <asp:ListItem Text="England" Value=" England"></asp:ListItem>
    <asp:ListItem Text="Turkey" Value="Turkey"></asp:ListItem>
</asp:CheckBoxList>
<br />
<asp:Button ID="btnSubmit" runat="server" Text="Submit" />
<br />
<!-- This is a CheckBox -->
<asp:CheckBox ID="chkAgreement" runat="server" Text="Do you accept the user
agreement." AutoPostBack="true" />

<h3>User Agreement:</h3>
<div>
    <asp:Label ID="lblAggrement" runat="server"></asp:Label>
</div>
<h3>Selected Countries:</h3>
<div>
    <asp:Label ID="lblCountry" runat="server"></asp:Label>
</div>
```

12. Double Click the Submit button, and you will be taken to the *CodeBehind* file of our page. Add the following code to the button click event.

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    if (chkAgreement.Checked)
        lblAggrement.Text = "User accepted the agreement";
    else lblAggrement.Text = " User did not accept the agreement ";

    foreach (ListItem item in chk1stCountry.Items)
    {
        if (item.Selected)
            lblCountry.Text += item.Text + "<br>";
    }
}
```

Procedure 04: Usage of DropDownList control (*The DropDownList control allows users to select an item from a set of predefined items each item is a separate object with its own properties, such as Text, Value, and Selected.*)
(open a new blank web page with the name **DropDownList.aspx**)

8. Drag and Drop a **DropDownList** and a **Label** to the page.
9. Add some professions like Artist, Lawyer....
10. Html view;

```
<asp:DropDownList runat="server" ID="ddlProfession" AutoPostBack="true">
  <asp:ListItem Value="1"> Artist </asp:ListItem>
  <asp:ListItem Value="2"> Builder </asp:ListItem>
  <asp:ListItem Value="3"> Computer programmer </asp:ListItem>
  <asp:ListItem Value="4"> Dancer</asp:ListItem>
  <asp:ListItem Value="5"> Engineer</asp:ListItem>
  <asp:ListItem Value="6"> Lawyer</asp:ListItem>
</asp:DropDownList>

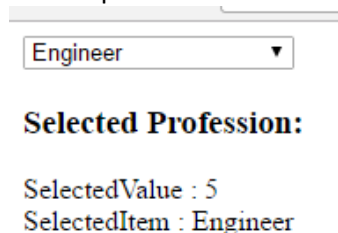
<br />

<h3>Selected Profession:</h3>
<div>
  <asp:Label ID="lblProfession" runat="server"></asp:Label>
</div>
```

13. Double Click the DropDownList control to create **OnSelectedIndexChanged** event, and you will be taken to the *CodeBehind* file of our page. Add the following code to your page.
 - The *SelectedIndexChanged* event of will work / fire / trigger only when the *AutoPostBack* property of the ASP.Net DropDownList is set to *True*.

```
protected void ddlProfession_SelectedIndexChanged(object sender, EventArgs e)
{
    lblProfession.Text = "SelectedValue : " + ddlProfession.SelectedValue
        + "<br> SelectedItem : " + ddlProfession.SelectedItem;
}
```

13. So, when you run the Page, it would show you the DropDownList on the Page with the Options defined.
14. When you select an Option from the DropDownList, it hits the SelectedIndexChanged Event in the Code Behind and display the selected profession.



Engineer ▼

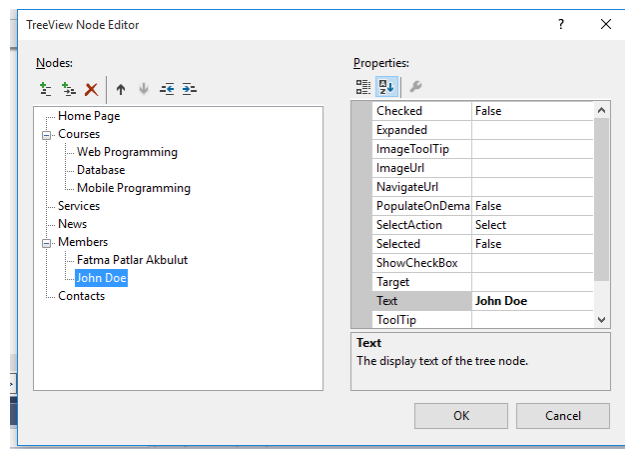
Selected Profession:

SelectedValue : 5
SelectedItem : Engineer

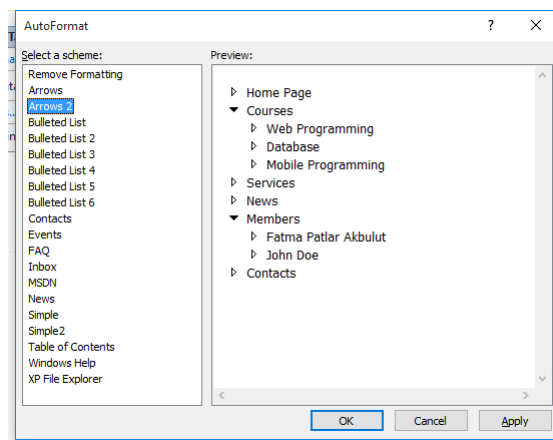
Procedure 04: Add a tree view control on the page

(open a new blank web page with the name **TreeView.aspx**)

8. Add a tree view control with the name treeMenu to the page. Select Edit Nodes... from the tasks. Edit each of the nodes using the Tree view node editor as shown:



9. The AutoFormat... task allows you to format the tree view as shown:



10. Add a label control and a text box control on the page and name them lblMessage and txtMessage respectively. Write a few lines of code to ensure that when a particular node is selected, the label control displays the node text and the text box displays all child nodes under it, if any. The code behind the file should look like this:

```
protected void treeMenu_SelectedNodeChanged(object sender, EventArgs e)
{
    txtMessage.Text = "";
    lblMessage.Text = "Selected node changed to: " +
    treeMenu.SelectedNode.Text;
    TreeNodeCollection childnodes = treeMenu.SelectedNode.ChildNodes;
    if (childnodes != null)
    {
        foreach (TreeNode tnode in childnodes)
        {
            txtMessage.Text += tnode.Value + "\n";
        }
    }
}
```

11. Execute the page to see the effects. You will be able to expand and collapse the nodes.

- ▷ Home Page
- ▼ Courses
 - ▷ Web Programming
 - ▷ Database
 - ▷ Mobile Programming
- ▷ Services
- ▷ News
- ▼ [Members](#)
 - ▷ Fatma Patlar Akbulut
 - ▷ John Doe
- ▷ Contacts

Selected node changed to: Members

Fatma Patlar Akbulut
John Doe

Now you can start your lab assignment!

Take Home Exercises

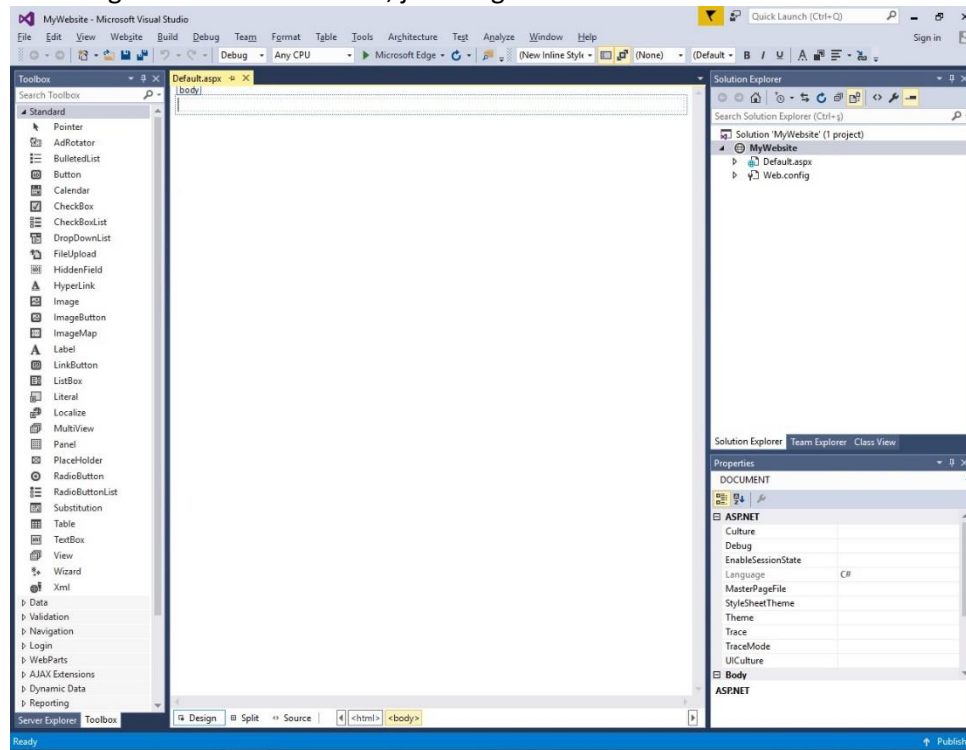
1. Write a Program to insert data in a multiline textbox by querying in another textbox.
2. Write a Program to get selected date from Calendar control.
3. Write a Program to insert an image to your page
4. Write a Program to generate the Login control (username=test, password=1234).
5. Write a Program to give users a list of color choices in List box, when a user selects an item from list display it in label control: "selected color is "+
6. Write a Program to display gender, use two radio buttons (male, female), display selected ones in a label control.
7. Try to use Radio button list and HyperLink.

USEFUL INFORMATION

Working with Views and Windows

You can work with windows in the following ways:

- To change the Web Forms Designer from one view to another, click on the Design or source button.
- To close a window, click on the close button on the upper right corner and to redisplay, select it from the View menu.
- To hide a window, click on its Auto Hide button. The window then changes into a tab. To display again, click the Auto Hide button again.
- To change the size of a window, just drag it.



Adding Folders and Files to your Website

When a new web form is created, Visual Studio automatically generates the starting HTML for the form and displays it in Source view of the web forms designer. The Solution Explorer is used to add any other files, folders or any existing item on the web site.

- To add a standard folder, right-click on the project or folder under which you are going to add the folder in the Solution Explorer and choose New Folder.
- To add an ASP.NET folder, right-click on the project in the Solution Explorer and select the folder from the list.
- To add an existing item to the site, right-click on the project or folder under which you are going to add the item in the Solution Explorer and select from the dialog box.

Projects and Solutions

Typical ASP.NET application consists of many items: the web content files (.aspx), source files (.cs files), assemblies (.dll and .exe files), data source files (.mdb files), references, icons, user controls and miscellaneous other files and folders. All these files that make up the website are contained in a Solution.

When a new website is created, automatically creates the solution and displays it in the solution explorer.

Solutions may contain one or more projects. A project contains content files, source files, and other files like data sources and image files. Generally, the contents of a project are compiled into an assembly as an executable file (.exe) or a dynamic link library (.dll) file.

Typically a project contains the following content files:

- Page file (.aspx)
- User control (.ascx)
- Web service (.asmx)
- Master page (.master)
- Razor page(.cshtml)
- Site map (.sitemap)
- Website configuration file (.config)
-

Building and Running a Project

You can execute an application by:

- Selecting Start
- Selecting Start Without Debugging from the Debug menu,
- pressing F5
- Ctrl-F5

The program is built meaning, the .exe or the .dll files are generated by selecting a command from the Build menu.

References

1. Learn ASP.NET Web Application Framework, Tutorials Point, 2014
2. Michaelis, Mark, and Eric Lippert. Essential C# 6.0. Addison-Wesley Professional, 2015.