TechLift
KICKSTART YOUR CAREER

WEB ENGINEERING

.Net

P@SHA

PAKISTAN PSEB SOFTWARE EXPORT BOARD

Week 3 - Day 13

ASP.NET Web Application

Learning Objectives

By the end of this session, the students will have developed an understanding of:

- ASP.NET Web Applications
- ASP.NET Web Project Structure, Default Files, Folders
- Default References

ASP.NET Web Applications

Introduction

- ASP.NET is a free web framework for building great websites and web applications using HTML, CSS, and JavaScript. You can also create Web APIs and use real-time technologies like Web Sockets.
- Here, you will learn about three frameworks for creating web applications: Web Forms, ASP.NET MVC, and ASP.NET Web Pages.

ASP.NET WEB PAGES - STRUCTURE

Many websites have content that is displayed on every page (like headers and footers).

With Web Pages you can use the @RenderPage() method to import content from separate files.

Content block (from another file) can be imported anywhere in a web page, and can contain text, markup, and code, just like any regular web page.

Using common headers and footers as an example, this saves you a lot of work. You don't have to write the same content in every page, and when you change the header or footer files, the content is updated in all your pages.

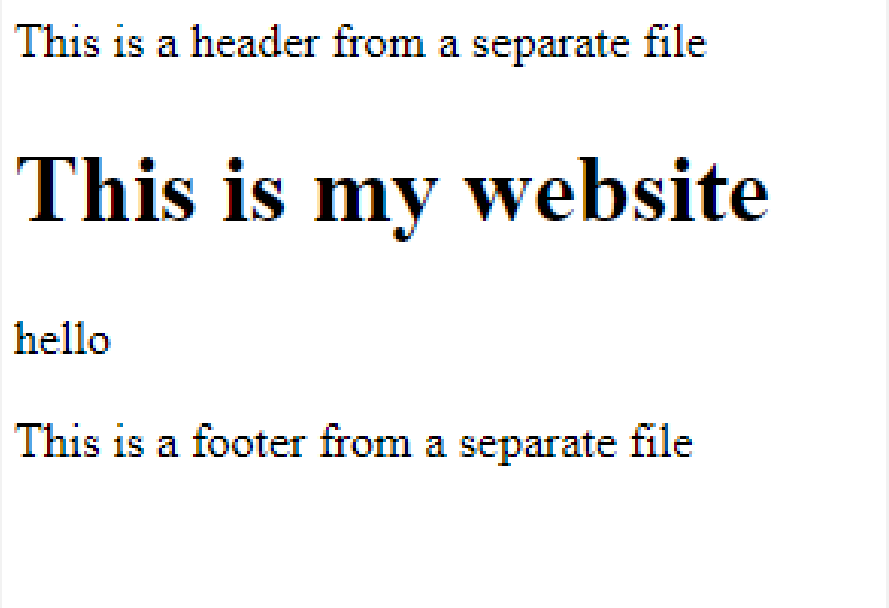
```
<html>
<body>
@RenderPage("header.cshtml")
<h1>Hello Web Pages</h1>
<p>This is a paragraph</p>
@RenderPage("footer.cshtml")
</body>
</html>
```

OUTPUT

Example

The trainer will show the output on their computer and encourage the students code the following output.

For help, click here.



ASP.NET WEB PAGES - STRUCTURE (Hiding Sensitive Information)

With ASP.NET, the common way to hide sensitive information (database passwords, email passwords, etc.) is to keep the information in a separate file named "_AppStart".

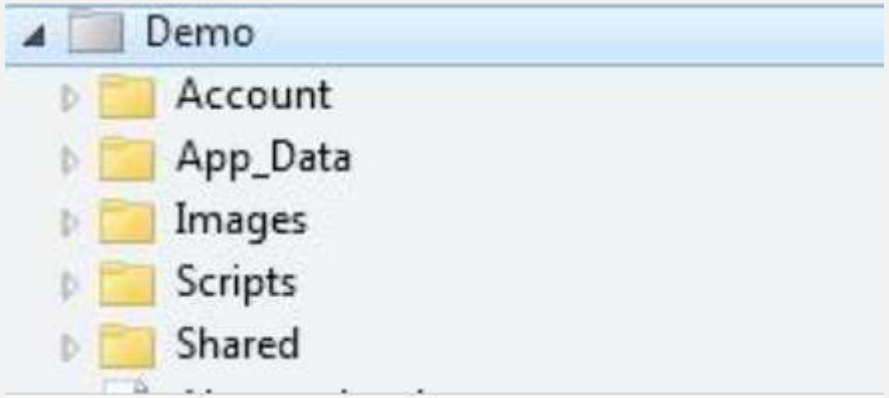
_AppStart.cshtml

```
@{
WebMail.SmtpServer = "mailserver.example.com";
WebMail.EnableSsl = true;
WebMail.UserName = "username@example.com";
WebMail.Password = "your-password";
WebMail.From = "your-name-here@example.com";
}
```

ASP.NET WEB PAGES - FOLDERS

Logical Folder Structure

Below is a typical folder structure for an ASP.NET web pages web site:



- The "Account" folder contains logon and security files
- The "App_Data" folder contains databases and data files
- The "Images" folder contains images
- The "Scripts" folder contains browser scripts
- The "Shared" folder contains common files (like layout and style files)

ASP.NET WEB PAGES - FOLDERS

Physical Folder Structure

The physical structure for the "Images" folder at the website above might look like this on a computer:

C:\Johnny\Documents\MyWebSites\Demo\Images

Virtual and Physical Names

From the example above:

The virtual name of a web picture might be "Images/pic31.jpg".

But the physical name is "C:\Johnny\Documents\MyWebSites\Demo\Images\pic31.jpg"

ASP.NET WEB PAGES - FOLDERS

URL and PATHS

URLs are used to access files from the web:
https://www.w3schools.com/html/html5_intro.asp

The URL corresponds to a physical file on a server:
C:\MyWebSites\w3schools\html\html5_intro.asp

A virtual path is shorthand to represent physical paths. If you use virtual paths, you can move your pages to a different domain (or server) without having to update the paths.

The ~ Operator

To specify the virtual root in programming code, use the ~ operator.

If you use the ~ operator, instead of a path, you can move your website to a different folder or location without changing any code:

```
var myImagesFolder = "~/images";
var myStyleSheet = "~/styles/StyleSheet.css";
```

ASP.NET WEB PAGES - FOLDERS

The Server.MapPath Method

The Server.MapPath method converts a virtual path (/default.cshtml) to a physical path that the server can understand (C:\Johnny\MyWebSited\Demo\default.cshtml).

You will use this method when you need to open data files located on the server (data files can only be accessed with a full physical path):

The Href Method

The Href method converts a path used in the code to a path that the browser can understand (the browser cannot understand the ~ operator).

You use the Href method to create paths to resources like image files, and CSS files.

You will often use this method in HTML <a>, <img>, and <link> elements:

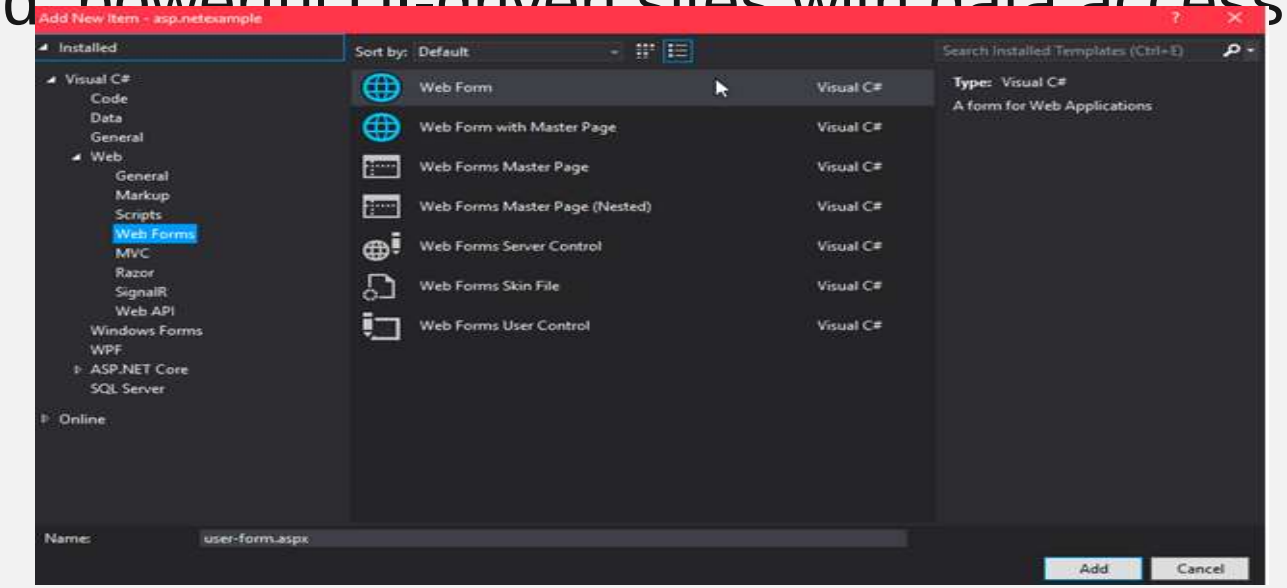
```
@{var myStyleSheet = "~/Shared/Site.css";}

<!-- This creates a link to the CSS file. -->
<link rel="stylesheet" type="text/css" href="@Href(myStyleSheet)" />

<!-- Same as : -->
<link rel="stylesheet" type="text/css" href="/Shared/Site.css" />
```

Web Forms

With ASP.NET Web Forms, you can build dynamic websites using a familiar drag-and-drop, event-driven model. A design surface and hundreds of controls and components let you rapidly build sophisticated, powerful UI-driven sites with data access.

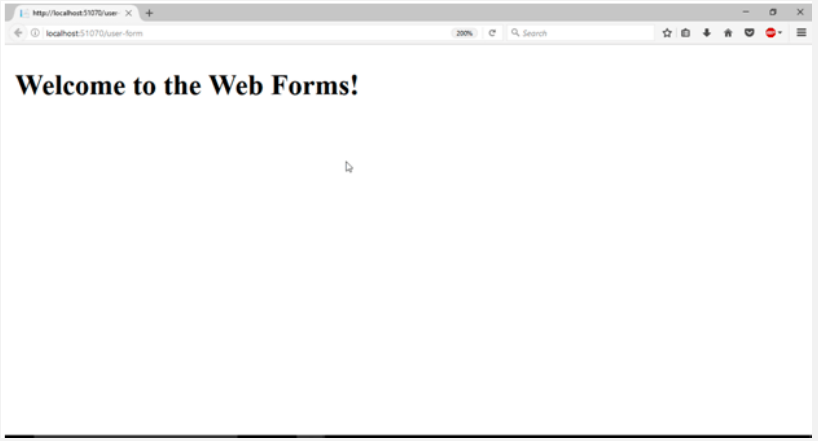


Example Web Forms

The trainer will demonstrate this code to provide an idea of web forms

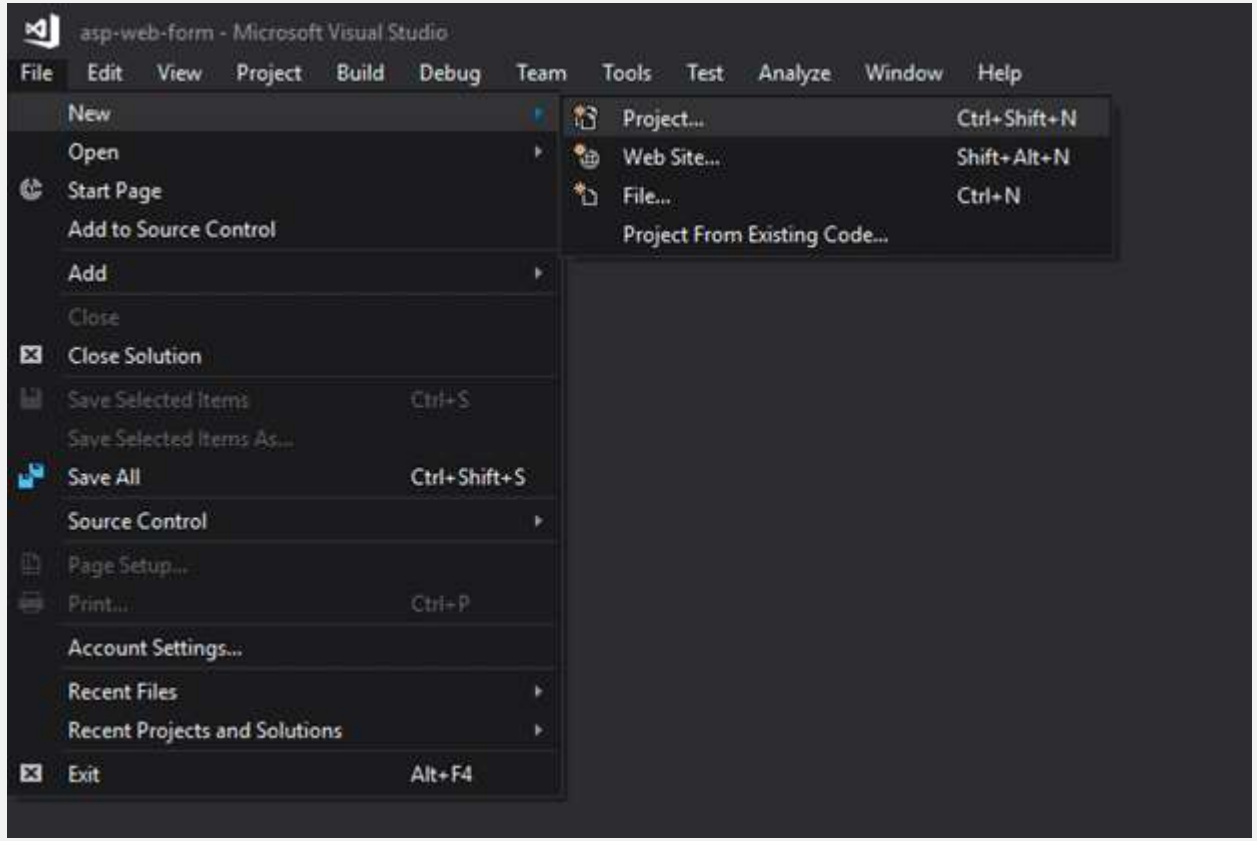
```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="user-form.aspx.cs"
Inherits="asp.netexample.user_form" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
<h2>Welcome to the Web Forms!</h2>
</div>
</form>
</body>
</html>
```

OUTPUT



Task: Create ASP.NET Web Forms

Create a web form on your computer



ASP.NET Web Forms Server Controls

ASP.NET provides web forms controls that are used to create HTML components. These controls are categories as server and client based.

Examples of Web form server controls include:

1. Label
2. TextBox
3. Button
4. LinkButton
5. ImageButton
6. Hyperlink
7. DropDownList
8. ListBox
9. DataGrid
10. DataList
11. CheckBox, etc

Example: Web form server controls

The trainer will demonstrate this code on their computer and encourage the students to guess its output.

Once completed, click here to check the output.

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="WebControls.aspx.cs"
Inherits="WebFormsControlls.WebControls" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            <asp:Label ID="labelId" runat="server">User Name</asp:Label>
<asp:TextBox ID="UserName" runat="server" ToolTip="Enter User Name"></asp:TextBox>
        </div>
        <p>
        <asp:Button ID="SubmitButton" runat="server" Text="Submit" OnClick="SubmitButton_Click" />
        </p>
        <br />
    </form>
     <asp:Label ID="userInput" runat="server"></asp:Label>
</body>
</html>
```

ASP.NET Validation

- To perform validation, ASP.NET provides controls that automatically check user input and require no code. We can also create custom validation for our application.
- Following are the validation controls:

Validator	Description
CompareValidator	It is used to compare the value of an input control against a value of another input control.
RangeValidator	It evaluates the value of an input control to check the specified range.
RegularExpressionValidator	It evaluates the value of an input control to determine whether it matches a pattern defined by a regular expression.
RequiredFieldValidator	It is used to make a control required.
ValidationSummary	It displays a list of all validation errors on the Web page.

Compare Validator

This validator evaluates the value of an input control against another input control on the basis of specified operator.

We can use comparison operators like: less than, equal to, greater than etc.

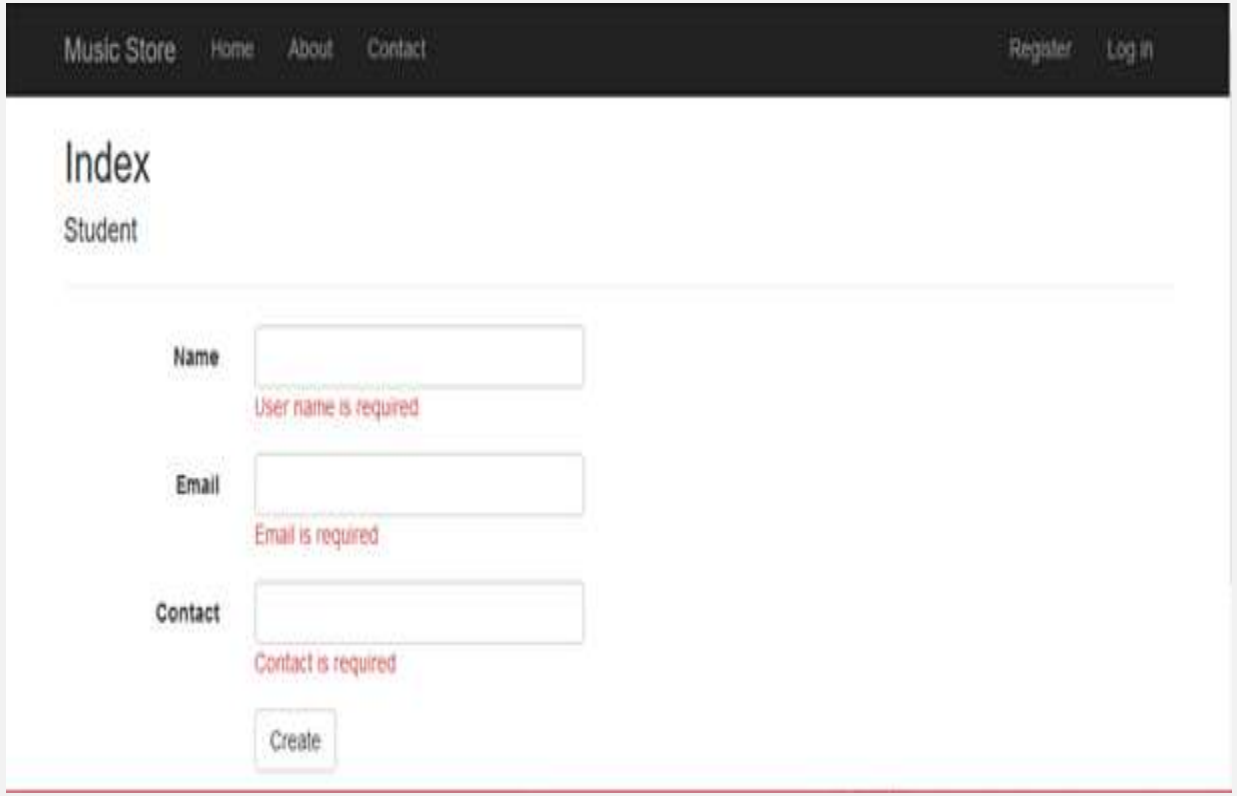
Click here for full explanation

```
<table border="0" cellpadding="0" cellspacing="0">
        <tbody><tr>
            <td>
                <b>Password</b>
                <asp:textbox runat="server" textmode="Password" id="txtPassword">
            </asp:textbox></td>
        </tr>
        <tr>
            <td>
                <b>Confirm Password</b>
                <asp:textbox runat="server" textmode="Password" id="txtConfirmPassword">
                <br>
                <asp:comparevalidator errormessage="Password and confirm password mismatch
            </asp:comparevalidator></asp:textbox></td>
        </tr>
        <tr>
            <td>
                <asp:button id="Button1" text="Save" runat="server">
            </asp:button></td>
        </tr>
    </tbody></table>
```

Example: Compare Validator

The trainer will show the output on their computer and encourage the students code the following output.

Once completed, click here to check the output.



Range Validator

This validator evaluates the value of an input control to check that the value lies between specified ranges.

It allows us to check whether the user input is between a specified upper and lower boundary. This range can be numbers, alphabetic characters and dates.

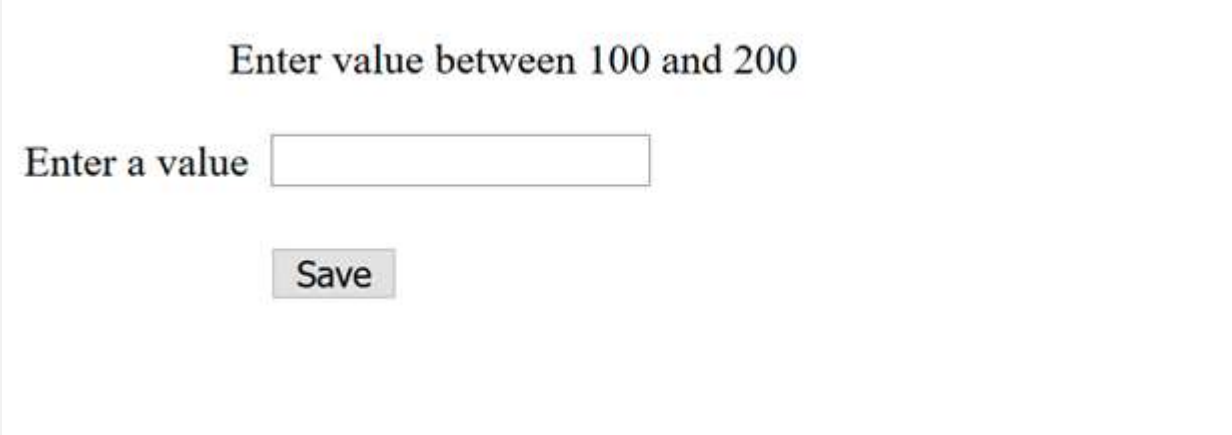
Click here for full explanation

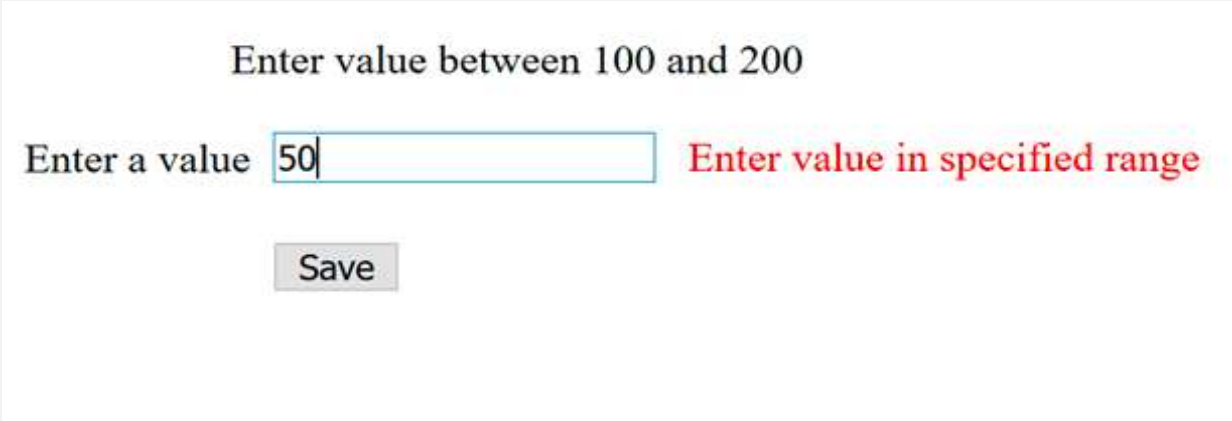
```
<table border="0" cellpadding="0" cellspacing="0">
        <tbody><tr>
            <td>
                <b>Enter Value</b>
                <asp:textbox runat="server" id="txtRangeValidator">
            </asp:textbox></td>
        </tr>
        <tr>
            <td>
                <asp:rangevalidator errormessage="Please enter value between 10-20."
                forecolor="Red" controltovalidate="txtRangeValidator"
                minimumvalue="10" maximumvalue="20" runat="server">
            </asp:rangevalidator></td>
        </tr>
    </tbody></table>
```

Example: Range Validator

The trainer will show the output on their computer and encourage the students code the following output.

Once completed, click here to check the output.





RegularExpressionValidator Control

This validator is used to validate the value of an input control against the pattern defined by a regular expression.

It allows us to check and validate predictable sequences of characters like: e-mail address, telephone number etc.

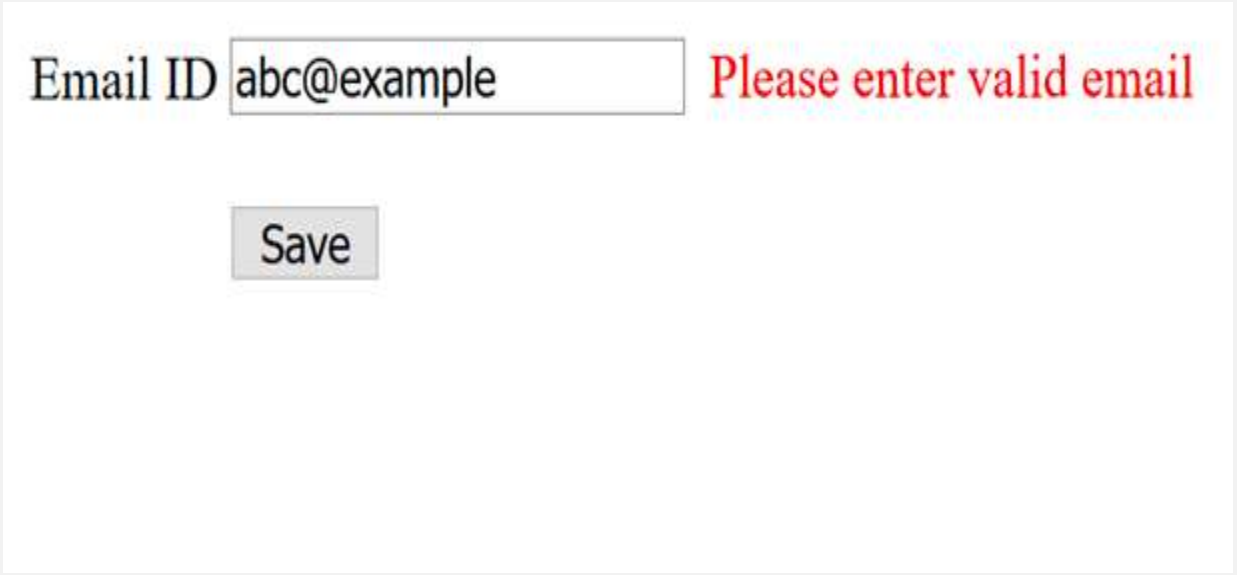
The **ValidationExpression** property is used to specify the regular expression, this expression is used to validate input control.

Click here for full explanation

```
<table border="0" cellpadding="0" cellspacing="0">
 <tbody><tr>
     <td>
         <b>Enter Your Email </b>
         <asp:textbox runat="server" id="txtRegularExpressionValidator">
     </asp:textbox></td>
 </tr>
 <tr>
     <td>
         <asp:regularexpressionvalidator validationexpression="\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*'
         errormessage="Invalid Email Format." forecolor="Red"
         controltovalidate="txtRegularExpressionValidator"
           runat="server">
     </asp:regularexpressionvalidator></td>
 </tr>
 <tr>
     <td>
         <asp:button id="Button1" text="Save" runat="server">
     </asp:button></td>
 </tr>
</tbody></table>
```

Example: RegularExpressionValidator Control

The trainer will show the output on their computer and encourage the students code the following output.



Once completed, click here to check the output.

ASP.NET MVC

The MVC (Model-View-Controller) is an application development pattern or design pattern which separates an application into three main components:

1. Model
2. View
3. Controller

Advantages of ASP.NET MVC Framework

This approach provides the following advantages.

- It manages application complexity by dividing an application into the model, view and controller.
- It does not use view state or server-based forms. This makes the MVC framework ideal for developers who want full control over the behavior of an application.
- It provides better support for test-driven development.
- It is suitable for large scale developer team and web applications.
- It provides high degree of control to the developer over the application behavior.

ASP.NET Razor

Razor is a markup syntax that lets you embed server-based code (Visual Basic and C#) into web pages.

Server-based code can create dynamic web content on the fly, while a web page is written to the browser. When a web page is called, the server executes the server-based code inside the page before it returns the page to the browser. By running on the server, the code can perform complex tasks, like accessing databases.

Razor is based on ASP.NET, and designed for creating web applications. It has the power of traditional ASP.NET markup, but it is easier to use, and easier to learn.

Razor Syntax

```
<ul>
<%for i = 0 to 10%>
<li><%=i%></li>
<%next%>
</ul>
```

Razor Loops

If you need to run the same statements repeatedly, you can program a loop.

If you know how many times you want to loop, you can use a for loop. This kind of loop is especially useful for counting up or counting down:

```
<html>
<body>
@for(var i = 10; i < 21; i++)
{
<p>Line @i</p>
}
</body>
</html>
```

Output

Line 10

Line 11

Line 12

Line 13

Line 14

Line 15

Line 16

Line 17

Line 18

Line 19

Line 20

IMPLEMENTATION

The trainer will demonstrate this code on their computer and encourage the students to guess its output.

Once completed, click here to check the output.

```
@{
string[] members = {"Jani", "Hege", "Kai", "Jim"};
int i = Array.IndexOf(members, "Kai")+1;
int len = members.Length;
string x = members[2-1];
}
<html>
<body>
<h3>Members</h3>
@foreach (var person in members)
{
<p>@person</p>
}
<p>The number of names in Members are @len</p>
<p>The person at position 2 is @x</p>
<p>Kai is now in position @i</p>

</body>
</html>
```

ASP.NET Default References

These are few of the default references in ASP.NET

Class	Description
AcceptVerbsAttribute	Represents an attribute that specifies which HTTP verbs an action method will respond to.
ActionDescriptor	Provides information about an action method, such as its name, controller, parameters, attributes, and filters.
ActionExecutedContext	Provides the context for the ActionExecuted method of the ActionFilterAttribute class.
ActionExecutingContext	Provides the context for the ActionExecuting method of the ActionFilterAttribute class.
ActionFilterAttribute	Represents the base class for filter attributes.
ActionMethodSelectorAttribute	Represents an attribute that is used to influence the selection of an action method.
ActionNameAttribute	Represents an attribute that is used for the name of an action.
ActionNameSelectorAttribute	Represents an attribute that affects the selection of an action method.
ActionResult	Encapsulates the result of an action method and is used to perform a framework-level operation on behalf of the action method.
AdditionalMetadataAttribute	Provides a class that implements the IMetadataAware interface in order to support additional metadata.

Homework

1. Design a Login form Using the concepts of ASP.NET Validations

Learning Objectives

By the end of this session, the students have practised

- ASP.NET Web Applications
- ASP.NET Web Pages - Folders
- Default References

- Top Features Of ASP.NET MVC

Conclusion & Q/A

See you tomorrow!