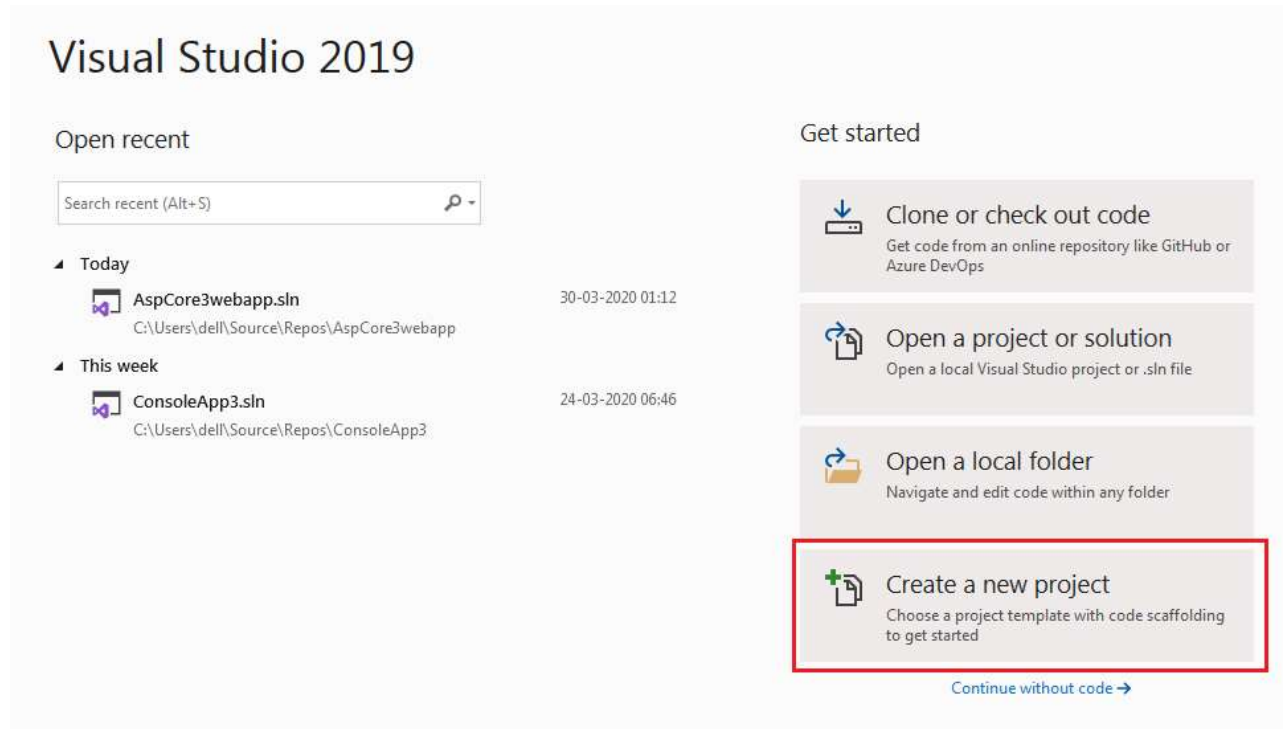


2PGDCA4 (B)-Programming With ASP. Net

UNIT-III

1. Create ASP.NET Core Application

Open Visual Studio 2019 and click on **Create a new project**, as shown below.



1.1 Create a New ASP.NET Core 3.0 Project

The "Create a new project" dialog box includes different .NET Core 3.0 application templates. Each will create predefined project files and folders depend on the application type. Here we will create a simple web application, so select **ASP.NET Core Web Application** template and click **next**, as shown below.

Create a new project

Recent project templates

Console App (.NET Core) C#

ASP.NET Core Web Application C#

All languages

All platforms

All project types

Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
C# Linux macOS Windows Console

Console App (.NET Core)
A project for creating a command-line application that can run on .NET Core on Windows, Linux and MacOS.
Visual Basic Windows Linux macOS Console

ASP.NET Core Web Application
Project templates for creating ASP.NET Core web apps and web APIs for Windows, Linux and macOS using .NET Core or .NET Framework. Create web apps with Razor Pages, MVC, or Single Page Apps (SPA) using Angular, React, or React + Redux.
C# Linux macOS Windows Cloud Service Web

Blazor App
Project templates for creating Blazor apps that run on the server in an ASP.NET Core app or in the browser on WebAssembly. These templates can be used to build web apps with rich dynamic user interfaces (UIs).
C# Linux macOS Windows Cloud Web

ASP.NET Web Application (.NET Framework)

Back

Next

1.2 Select Application Template

Next, give the appropriate name, location, and the solution name for the ASP.NET Core application. In this example, we will give the name "MyFirstCoreWebApp" and click on the Create button, as shown below.

Configure your new project

ASP.NET Core Web Application C# Linux macOS Windows Cloud Service Web

Project name

Location

D:\TestProjects\core

...

Solution name ⓘ

☐ Place solution and project in the same directory

Back

Create

1.3 Configure Project

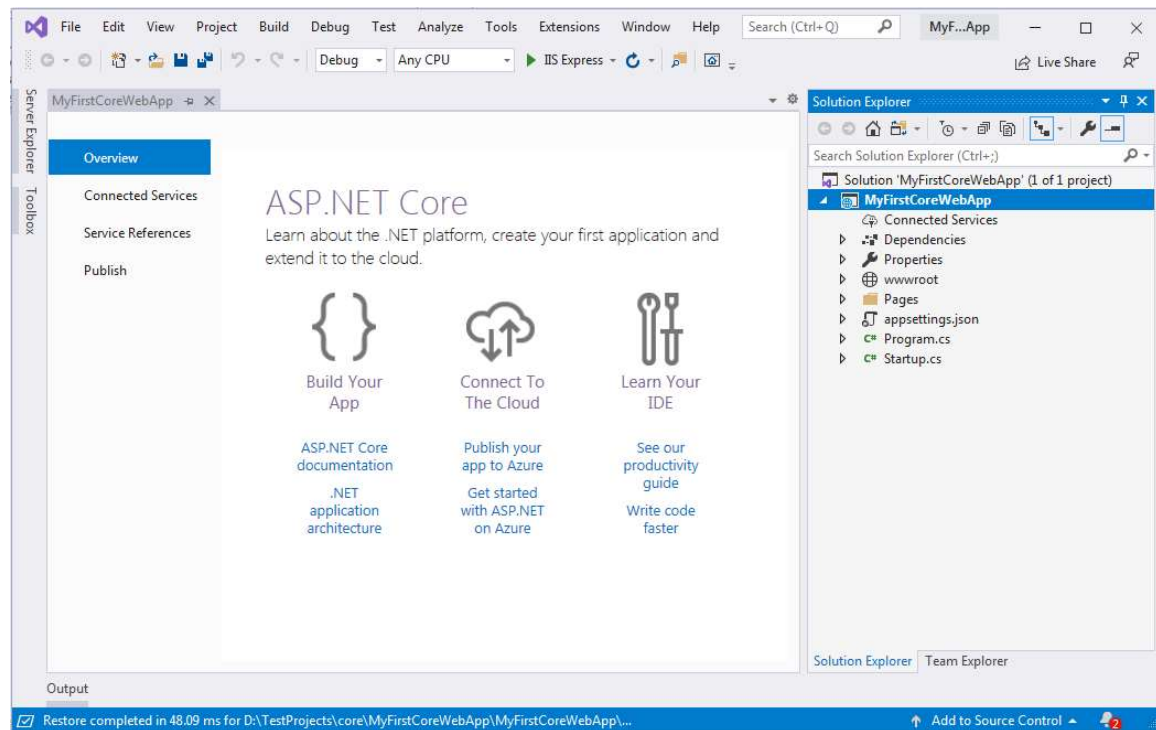
Next, select appropriate ASP.NET Core Web application template such as Empty, API, Web Application, MVC, etc. Here, we want to create a web application, so select the Web Application template. We don't want HTTPS at this point, so uncheck Configure for HTTPS checkbox, as shown below. Also, make sure you have selected the appropriate .NET Core and ASP.NET Core versions. Click on the Create button to create a project.

Create a new ASP.NET Core web application

The screenshot shows the 'Create a new ASP.NET Core web application' dialog. At the top, there are two dropdown menus: '.NET Core' and 'ASP.NET Core 3.1'. Below these, a list of project templates is displayed. The 'Web Application' template is selected and highlighted in blue. To the right of the templates, there are configuration options. Under 'Authentication', 'No Authentication' is selected. Under 'Advanced', the 'Configure for HTTPS' checkbox is unchecked. At the bottom right, there are 'Back' and 'Create' buttons. The 'Create' button is highlighted.

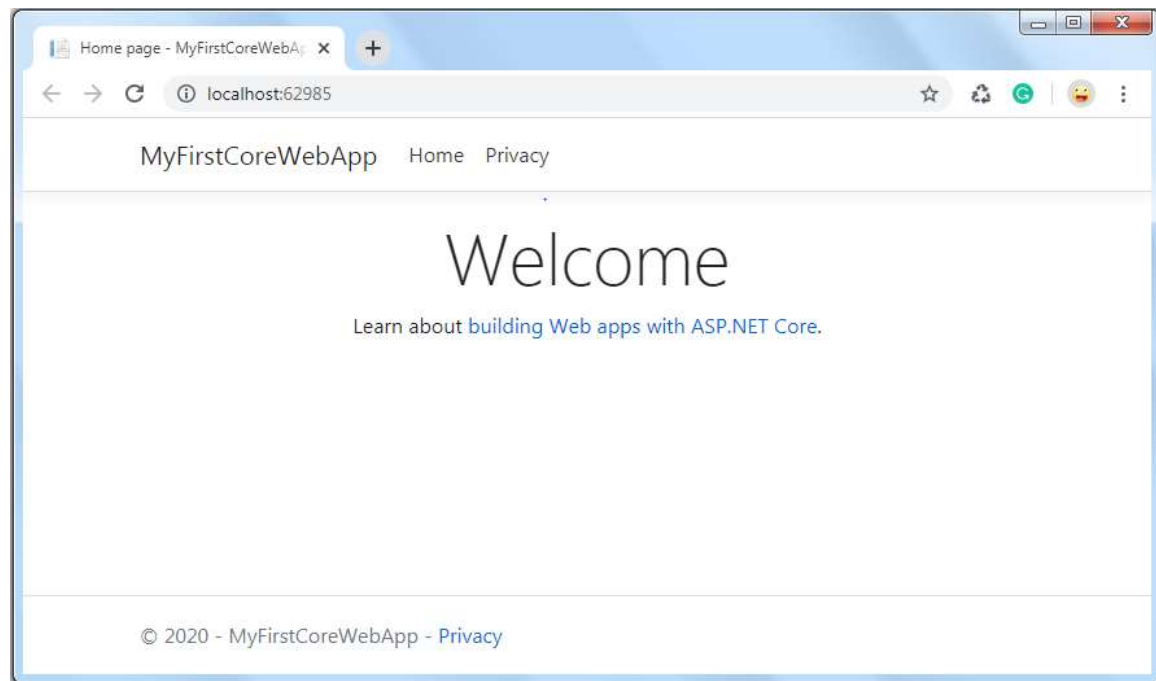
1.4 Select Web Application Template

This will create a new ASP.NET Core web project in Visual Studio 2019, as shown below. Wait for some time till Visual Studio restores the packages in the project. Restoring process means Visual Studio will automatically add, update or delete configured dependencies as NuGet packages in the project.



1.5 Create ASP.NET Core 3 Application

To run this web application, click on **IIS Express** or press Ctrl + F5. This will open the browser and display the following result.



2. Create a Multiform Web Project

2.1 Single Project Method

Microsoft recommends the single project method for small-sized to medium-sized Web applications. Visual Studio .NET directly supports this method. The whole Web application is built as a single ASP.NET Web application project. Each team member downloads a copy of the entire project to a development computer, where he or she develops part of the application. You should use source control software to coordinate the work of the team members on the files that make up the project.

The advantages of the single project method are as follows:

- This method is easy to implement. You create a single Web application project in Visual Studio .NET, and then you add application items to it.
- For smaller Web applications, a single Visual Studio .NET project is easy to manage.
- Visual Studio .NET directly supports this method. You do not have to take special steps to make this method work.
- Because the whole project is built into a single assembly, you do not need references between multiple assemblies.

The disadvantages of the single project method are as follows:

- Large Web applications are difficult to manage as a single unit.
- Every time you want to build your work, even for small code changes, you must build the whole project. For large projects, this can be very time consuming and can make development less efficient.

2.2 Multiple Project Method

If you separate the Web application into multiple Visual Studio .NET projects, you may find it easier to manage the development of large Web applications. You can separate a large development project into smaller projects, which you can manage and build separately. Your team can work on separate parts of a Web application by working on separate projects. You should still use source control software to coordinate work on project files. Visual Studio .NET does not directly support this method because Visual Studio .NET Web projects are always created in their own IIS application root directories with their own assemblies. IIS Web applications cannot span multiple IIS application root directories.

Additionally, you cannot use resources in one root directory from other root directories in Visual Studio .NET. It is difficult to create a single application from multiple projects if

these applications cannot share resources. To resolve these problems, you can use the procedure in the Make Multiple Visual Studio .NET Projects Participate in the Same Web Application section so that multiple Visual Studio projects share the same IIS application root directory.

The advantages of the multiple project method are as follows:

- It is easier to manage smaller units of a large Web application.
- You can build each project separately from the other projects, which shortens the build times during development.
- You can divide a large Web application into logical units and have these units share common resources, such as controls.

The disadvantages of the multiple project method are as follows:

- Visual Studio .NET does not directly support this method. You must perform additional steps so that separate Visual Studio .NET projects participate in the same Web application.
- Assemblies that access each other's resources must set references to each other. Visual Studio .NET does not allow circular references.
- This method is not ideal for small Web applications. For small Web applications, it is more complex to manage multiple projects than to manage a single project.

2.3 Make Multiple Visual Studio .NET Projects Participate in the Same Web Application

To make the Visual Studio .NET projects participate in the same Web application, the projects must share the same IIS application root directory. Visual Studio .NET creates Web projects in their own application root directories. Therefore, you must configure this separately.

To make the Visual Studio .NET projects participate in the same Web application, you must complete four main steps:

1. Create the main project in a directory that is the root directory for the whole application.
2. Create the child projects in subdirectories of the root directory in the *same* Visual Studio .NET solution.
3. Remove the Web applications that Visual Studio .NET creates for the child projects through IIS.

4. NOTE: Because Visual Studio .NET does not allow you to create or to work with multiple Web applications in the same physical directory, you must create the various projects that make up your application in separate directories.
5. To deploy the projects to individual development computers, copy the application directory structure to those computers, and then create an IIS application root directory for the main project directory.

Create the IIS Application Root Project

To create the IIS application root project for the Web application, follow these steps:

1. Start Visual Studio .NET.
2. On the File menu, point to New, and then click Project.
3. In the New Project dialog box, click the language that you want to use under Project Types, and then click ASP.NET Web Application under Templates.
4. In the Location text box, replace the WebApplication# default name with MainWeb. If you are using the local server, you can leave the server name as http://localhost. The Location box should then appear as follows:

http://localhost/MainWeb

Create the Child Projects

To create the child projects, follow these steps:

1. For each child project, right-click the solution in the Solution Explorer window, point to Add, and then click New Project.
2. In the Add New Project dialog box, click ASP.NET Web Application under Templates.
3. In the Location text box, type http://localhost/MainWeb/<subwebname>. This creates a child project named <subwebname> in a subdirectory of the root application directory.
4. Click OK.
5. Delete any files that are not needed for a non-application root directory from the project. Specifically, delete the Global.asax and the Web.config files. This child project will rely on the Global.asax and the Web.config files from the main project.
6. If the child project will only contain shared resources, such as user controls, delete the WebForm1.aspx file.
7. Build the solution.

Remove the IIS Applications That Correspond to the Child Projects

To remove the IIS applications that correspond to the child projects, follow these steps:

1. Click Start, point to Programs (or All Programs in Windows XP), point to Administrative Tools, and then click Internet Services Manager.
2. Locate your main Web application and the child Web projects that you want to remove.
3. For each child project, right-click the Web application node, and then click Properties.
4. On the Directory tab, click Remove, and then click OK.

IMPORTANT: Do *not* click Delete. This can permanently delete the corresponding file directory and your project files.

After you configure the child projects to share a common IIS application root directory, you can share resources between projects in the solution. For example, you can drag a user control from a shared resources project into an .aspx file in another project. Note that you can only do this after you configure the projects to share a common IIS application root. Visual Studio .NET does not allow you to share resources if the projects are still in separate IIS application root directories.

Set References to the Child Projects

To add references to the main Web application for all of the child projects, follow these steps.

1. In Solution Explorer of the main Web application, right-click References, and then click Add Reference.
2. In the Add Reference dialog box, click the Projects tab.
3. Select the child projects, and then click OK.

After you set references to the child projects, when you build the solution, the child project assemblies are copied to the Bin directory of the main application. You can then debug and use the components that are defined in the child projects.

Deploy Multi-Project Applications to Development Computers

To deploy the multi-project Web application to a development computer, you must copy the main (root) project that represents the root of the Web application, as well as any

child applications that you want to work on. You must make the main project directory the root of an IIS Web application.

There are many options for deployment, and it is beyond the scope of this article to describe each option. For example, you can deploy built versions of some of the source files in large Web applications to avoid bringing in many source files. Any references to local projects must be project references. If you want to drag ASP.NET user controls from another project onto a Web Form, the project that contains the user control must exist on your computer and in your solution.

For smaller ASP.NET Web applications, you can combine all of the projects. A good way to do this is to click Copy Project on the Project menu in Visual Studio .NET. You can also copy the project files to a computer.

To deploy the multi-project Web application to a development computer, follow these steps:

1. Copy the whole directory structure that you created in the previous steps to a development computer.
2. Make the directory that contains the root project an IIS application root. The easiest way to do this is to use the File System Explorer (if you are working with the NTFS file system) as follows:
 1. In the File System Explorer, locate the root directory of the Web application.
 2. Right-click the directory, and then click Properties.
 3. In the Properties dialog box, click the Web Sharing tab, and then click **Share this folder**.
 4. In the Alias box, type the name of the Web application, and then click OK.
 5. Click OK to close the Properties dialog box.
3. Make sure that all of the user identities who will be using the Web application (such as the IUSR account) have access to the directories.
4. Build the solution.

3. Form Validation

3.1 What is Validation?

Validation is basically the act of comparing something to a given set of rules and determining if it satisfies the criteria those rules represent. In this case, the something that we are trying to validate is the input that a visitor to our site has entered into a web form.

There are a number of reasons why we'd want to do this. Some basic examples are:

- No data or incomplete data was entered
- The value of the data entered is not within the appropriate range
- The format of the entered data is not as expected

There are any number of explanations why one of the above might occur. Perhaps a spider or web-bot has come across our form and tried to submit it without entering any data, maybe the user entered a item that doesn't really exist (ie. Feb. 29, 2002), or maybe they simply made a typo (ie. forgot to enter one of the digits in their phone number). Whatever the reason, the best course of action is usually easiest to determine if you know there is a problem while the user is still available to fix it.

3.2 What is Form Validation

That's why the concept of form validation became so popular so fast. If the type of data you're expecting to receive is relatively specific, there's no reason you can't set up a set of rules by which you can validate the data you receive right as the user is giving it to you. Assuming you can't make the correction automatically, still having the user available allows you to ask them to review their submission. After all, computers are very precise, but they're not all that smart. It's often easier for the user to fix whatever's causing the error then it is for the computer to do so.

Validations can be performed on the server side or on the client side (web browser). The user input validation take place on the Server Side during a post back session is called Server Side Validation and the user input validation take place on the Client Side (web browser) is called Client Side Validation. Client Side Validation does not require a postback. If the user request requires server resources to validate the user input, you should use Server Side Validation. If the user request does not require any server resources to validate the input, you can use Client Side Validation.

3.3 Server Side Validation

In the Server Side Validation, the input submitted by the user is being sent to the server and validated using one of server side scripting languages such as ASP.Net, PHP etc. After the validation process on the Server Side, the feedback is sent back to the client by a new dynamically generated web page. It is better to validate user input on Server Side because you can protect against the malicious users, who can easily bypass your Client Side scripting language and submit dangerous input to the server.

3.4 Client Side Validation

In the Client Side Validation you can provide a better user experience by responding quickly at the browser level. When you perform a Client Side Validation, all the user inputs validated in the user's browser itself. Client Side validation does not require a

round trip to the server, so the network traffic which will help your server perform better. This type of validation is done on the browser side using script languages such as JavaScript, VBScript or HTML5 attributes.

For example, if the user enters an invalid email format, you can show an error message immediately before the user move to the next field, so the user can correct every field before they submit the form.

Mostly the Client Side Validation depends on the JavaScript Language, so if users turn JavaScript off, it can easily bypass and submit dangerous input to the server. So the Client Side Validation cannot protect your application from malicious attacks on your server resources and databases.

4. ASP.NET validation controls

Data entered by the user is assumed to be correct and appropriate. But it is impossible and user must validate the data before it is processed. The validation can be applied to the code using various scripting languages. It is necessary that the user is capable of using scripting languages. To overcome this the validation controls are provided by ASP.NET. Each control has its unique functionality to be performed. The detail description and working of these controls is explained in this chapter.

A validation control is used to validate data from the input control. The error message will be displayed if the data is incorrect. The syntax for creating validation control in ASP.NET control is as shown below:

```
<asp:control_name id="some_id" runat="server"/>
```

4.1 Required Field Validator

Required Field Validator control is used to make the input field a required field in the application. The control raises an error if the input value is not entered by the user. By default, the initial value is an empty string.

The properties associated with the control are as follows:

Property	Description
id	It is the unique id for the control
runat	It specifies that the control is a server control

Text	It is the message to be displayed when the validation fails
IsValid	It is a Boolean value indicating that ControlToValidate is valid
InitialValue	It is the starting value of the control
ForeColor	It is the foreground color of the control
ErrorMessage	It is the text displayed when the property is not set for the control
Display	<p>It is the display behavior of the control. The values are as follows:</p> <p>None: The control is not displayed</p> <p>Static: The control displays an error message when the validation fails</p> <p>Dynamic: The control displays an error message if the validation fails</p>
BackColor	It is the back color for the control
ControlToValidate	It is the id for the control
EnabledClientScript	It is a Boolean value specifying the client side validation is enabled
Enabled	It is a Boolean value that specifies the client side validation is enabled or not

A sample code for the required field validator is:

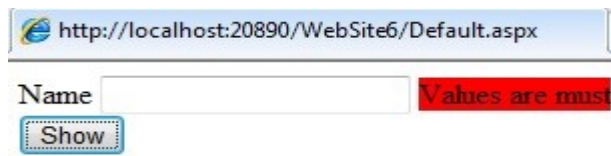
```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title></title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
        <asp:Label ID="Label1" runat="server" Text="Name" ></asp:Label>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      </div>
    </form>
  </body>
</html>
```

```

        <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ControlToValidate="TextBox1" ErrorMessage="Values are
must" BackColor="Red">
        </asp:RequiredFieldValidator>
        <br/>
        <asp:Button ID="Button1" runat="server" Text="Show" />
    </div>
</form>
</body>
</html>

```

Output



4.2 Range Validator

The Range Validator control is used to check that the user enters an input between two values. User can check between numbers, dates and characters. The control will not fail if the input value is not converted to the specific type.

The properties of the Range Validator are as follows:

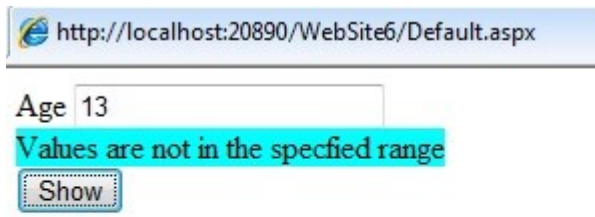
Property	Description
Text	It is the message to be displayed when the validation fails
runat	It specifies that the control is a server control
MaximumValue	It specifies the maximum value of the control
MinimumValue	It specifies the minimum value of the control
IsValid	It is a Boolean value that indicates that control specified by the ControlToValidate is valid
ControlToValidate	It is the id of the control to be validated
BackColor	It is the background color of the control

id	It is the unique id for the control
----	-------------------------------------

A sample code for the range validator control is:

```
<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
    <title></title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
        <asp:Label ID="Label1" runat="server" Text="Age" ></asp:Label>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br/>
        <asp:RangeValidator ID="RangeValidator1" runat="server"
ErrorMessgae="Values are not in the specified range"
ControlToValidate="Textbox1" Type="Integer" MinimumValue="18"
MaximumValue="25" BackColor="Aqua" >
        </asp:RangeValidator>
        <br/>
        <asp:Button ID="Button1" runat="server" Text="Show" />
      </div>
    </form>
  </body>
</html>
```

Output



4.3 Compare Validator

The Compare Validator is used to compare the values of one input control to the another input control or to a fixed value. If the input control value is empty then no validation occurs. The properties of the control are as follows:

Property	Description
id	It is the unique id for the control
runat	It specifies that the control is a server control
Text	The message to be displayed when the validation fails
Operator	It is the type of comparison to be performed. The operators used are Equal, GreaterThan, LessThan, LessThanEqual, NotEqual, DataTypeCheck, GreaterThanEqual
ControlToCompare	The name of the control to be compared with
BackColor	It is the background color of the control
ControlToValidate	The id of the control to be validated
EnableClientScript	It is a Boolean value to specify the client side validation
Type	It specifies the data type of the values to compare. The types are currency, Date, Double, Integer, String
Display	<p>It is the display behavior of the control. The values are as follows:</p> <p>None: The control is not displayed</p> <p>Static: The control displays an error message when the validation fails</p> <p>Dynamic: The control displays an error message if the validation fails</p>

A sample code for the control is as shown below:

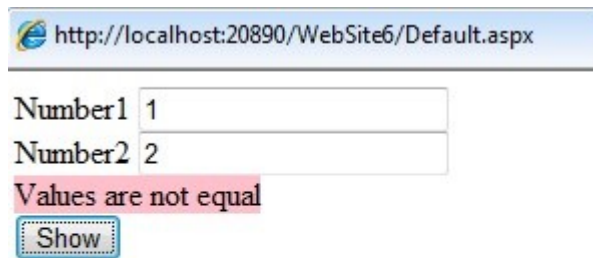
```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title></title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
        <asp:Label ID="Label1" runat="server" Text="Number1"
      ></asp:Label>
```

```

        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br/>
        <asp:Label ID="Label2" runat="server" Text="Number2"
    ></asp:Label>
        <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox><br/>
        <asp:CompareValidator ID="CompareValidator1" runat="server"
    ErrorMessage="Values
    are not equal" ControlToCompare="TextBox2" ControlToValidate="TextBox1"
    Type="Integer" Operator="Equal" BackColor="Pink" >
        </asp:CompareValidator>
        <br/>
        <asp:Button ID="Button1" runat="server" Text="Show" />
    </div>
</form>
</body>
</html>

```

Output



http://localhost:20890/WebSite6/Default.aspx

Number1 1

Number2 2

Values are not equal

Show

4.4 Regular Expression

The Regular Expression validator control is used to ensure that an input value matches a specifies pattern. It is useful for checking email address, phone numbers, zip codes, etc. The validation will not fail is the input control is empty. The properties of the control are as mentioned below:

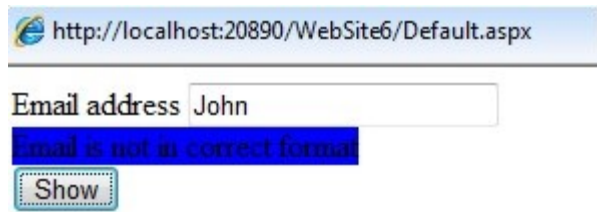
Property	Description
id	It is the unique id for the control
runat	It specifies that the control is a server control
ValidationExpression	It specifies that the expression used to validate the input control
ControlToValidate	It is the id of the control to validate

EnableClientScript	It is a Boolean value specifying the client side validation is enabled
ForeColor	It is the foreground color of the control
BackColor	It is the background color of the control
ErrorMessage	It is the text to display the ValidationSummary control when the validation fails
IsValid	It is a Boolean value that indicates that control specified by the ControlToValidate is valid

A sample code of the control is as shown below:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title></title>
  </head>
  <body>
    <form id="form1" runat="server" >
      <div>
        <asp:Label ID="Label1" runat="server" Text="Email
address"></asp:Label>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server"
        ControlToValidate="TextBox1" ErrorMessage="Email is not in
correct format" BackColor="Blue" ValidationExpression="\w+([-
+.' ]\w+)*@\w+([-.] )\w+([-.] \w+)*" >
        </asp:RegularExpressionValidator>
        <br/>
        <asp:Button ID="Button1" runat="server" Text="Show" />
      </div>
    </form>
  </body>
</html>
```

Output



The screenshot shows a web browser window with the address bar displaying 'http://localhost:20890/WebSite6/Default.aspx'. Below the address bar, there is a text input field labeled 'Email address' containing the text 'John'. A red error message, 'Email is not in correct format', is displayed below the input field. A 'Show' button is located below the error message.

4.5 Custom Validator

The custom validator allows the user to write a method to handle the validation of the value entered. It is used to compare the users input to a value in the database or the arithmetic validations for the controls. The properties for the control are as mentioned below:

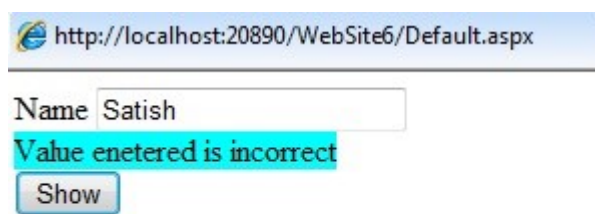
Property	Description
ClientValidationFunction	It specifies the name of the client side validation script function to be executed
ControlToValidate	It is the id of the control to validate
Display	It is the display behavior of the control. The values are as follows: None: The control is not displayed Static: The control displays an error message when the validation fails Dynamic: The control displays an error message if the validation fails
id	It is the unique id for the control
runat	It specifies that the control is a server control
EnableClientScript	It is a Boolean value specifying the client side validation is enabled

ForeColor	It is the foreground color of the control
BackColor	It is the background color of the control
OnServerValidate	It specifies the name of the server side validation script function to be executed
ErrorMessage	It is the text to display the ValidationSummary control when the validation fails

A sample code of the control is as shown below:

```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title></title>
  </head>
  <form id="form1" runat="server">
    <div>
      <asp:Label ID="Label1" runat="server" Text="Name" ></asp:Label>
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br/>
      <asp:CustomValidator ID="CustomValidator1" runat="server"
ErrorMessage="Value entered is incorrect" ControlToValidate="TextBox1"
BackColor="Aqua" OnServerValidate="CustomValidate_value" >
      </asp:CustomValidator>
      <asp:Button ID="btn1" runat="server" Text="Show" />
    </div> </form></html>
```

Output



4.6 Validation Summary

The Validation Summary control is used to display the summary of all validation errors in a web page. The error message displayed in this control is specified by the ErrorMessage property for every validation control. The summary can be displayed as a list, bulleted list, single paragraph depending on the **DisplayMode** property. The properties for the control are as follows:

Property	Description
DisplayMode	It defines how the summary will be displayed. The values are: BulletList, List, paragraph
id	It is the unique id for the control
runat	It specifies that the control is a server control
EnableClientScript	It is a Boolean value specifying the client side validation is enabled
ForeColor	It is the foreground color of the control
ShowSummary	It is a Boolean value that specifies whether ValidationSummary control should be displayed
HeaderText	It is a header in the ValidationSummary Control
ShowMessageBox	It is a Boolean value that specifies the summary should be displayed in the MessageBox or not
Enabled	It is a Boolean value that specifies whether the validation control is enabled or not

A code sample for the control is as shown below:

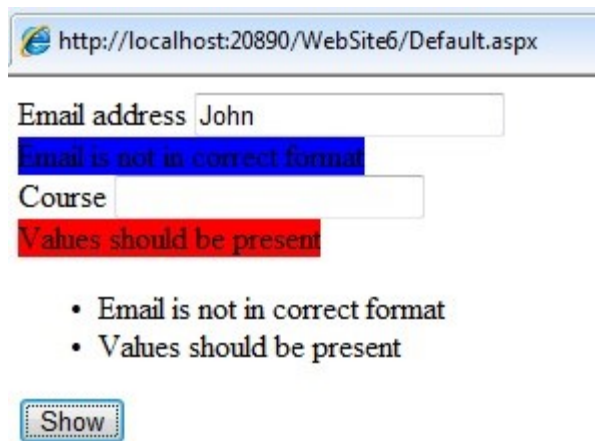
```
<html xmlns="http://www.w3.org/1999/xhtml">
  <head runat="server">
    <title></title>
  </head>
  <body>
    <form id="form1" runat="server">
      <div>
        <asp:Label ID="Label1" runat="server" Text="Email
address"></asp:Label>
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <asp:RegularExpressionValidator ID="RegularExpressionValidator1"
runat="server" ControlToValidate="TextBox1" ErrorMessage="Email is not
in correct format" BackColor="Blue" ValidationExpression="\w+([-+.'
]\w+)*\.\w+([- . ] \w+)*" >
        </asp:RegularExpressionValidator>
        <br/>
        <asp:Label ID="Label2" runat="server" Text="Course" ></asp:Label>
        <asp:TextBox ID="TextBox2" runat="server" ></asp:TextBox>
        <br/>
      </div>
    </form>
  </body>
</html>
```

```

        <asp:RequiredFieldValidator ID="RequiredFieldValidator1"
runat="server" ControlToValidate="TextBox2" BackColor="Red"
ErrorMessage="Values should be present" >
        </asp:RequiredFieldValidator>
        <br/>
        <asp:ValidationSummary ID="ValidationSummary1" runat="server" />
        <asp:Button ID="Button1" runat="server" Text="Show" />
    </div>
</form>
</body>
</html>

```

Output



The screenshot shows a web browser window with the address bar displaying "http://localhost:20890/WebSite6/Default.aspx". The form contains two input fields: "Email address" with the value "John" and "Course" which is empty. Below the "Email address" field, a blue error message box states "Email is not in correct format". Below the "Course" field, a red error message box states "Values should be present". At the bottom of the form, there is a list of error messages: "Email is not in correct format" and "Values should be present". A "Show" button is located at the bottom of the form.

5. AdRotator Control

The AdRotator control is used to display a sequence of advertisement images. It uses an external XML file to store the ad information. The XML file consists of several tags for designing the content in an application. The XML file consists of <Advertisements> tag at the start and end of the tags added. There are several <Ad> tags defined inside the <Advertisements> tag.

The list of elements that can be added in the <Ad> tag are as mentioned below:

Element	Description
<ImageUrl>	It defines the path of the image file
<NavigateUrl>	It is the URL to be linked when the user clicks the ad
<AlternateText>	It is an alternate text for the image
<Keyword>	It is a category for the ad

<Impressions>	It is the display rate for the number of hits on an ad
<Height>	It is used to define the height of the image
<Width>	It is used to define the width of the image

Some of the properties of the AdRotator control are mentioned below:

Property	Description
AdvertisementFile	It gets or sets the path to an XML file containing the advertisement information
Attributes	It gets the collection of arbitrary attributes that do not correspond to the control
BackColor	It gets or sets the background color of the Web Server control
BorderColor	It gets or sets the border color of the Web Server control
ClientID	It gets the controlID for the HTML Markup generated by ASP.NET
Context	It gets the HttpContext object associated with the server control
CssClass	It gets or sets the CSS class rendered by the Web Server control
DataSource	It gets or sets the object from which the data bound control retrieves the list of data items
Enabled	It gets or sets the value indicating the Web Server control is enabled
HasAttributes	It gets a value indicating the control has attributes set
ImageUrlField	It gets or sets the custom data field to use instead of the ImageUrl attribute
ItemType	It gets or sets the name of the data type for data binding

NavigateUrlField	It gets or sets the custom data field to use instead of the NavigateUrl attribute
Page	It gets a reference to the Page instance containing the server control
RenderingCompatibility	It gets a value specifying the version of ASP.NET for rendering
SelectMethod	It is the name of the method to call in order to read data
TabIndex	It gets or sets the tab index of the Web Server control
Target	It gets or sets the name of the browser window or frame
ToolTip	It gets or sets the text to be displayed when the mouse pointer hovers over the control
Visible	It gets or sets the value indicating the server control is rendered as UI on the page

The list of some AdRotator events is mentioned below:

Events	Description
AdCreated	It occurs once during the round trip to the server after the creation of the control
CallingDataMethods	It occurs when data methods are being called by the user
CreatingModelDataSource	It occurs when the ModelDataSource object is being created
DataBinding	It occurs when the server control binds to a data source
DataBound	It occurs after the server control binds to a data source
Disposed	It occurs when the server control is released from memory
Init	It occurs when the server control is initialized

Load	It occurs when the server control is loaded into Page object
PreRender	It occurs when the control object is loaded prior to rendering
Unload	It occurs when the server control is unloaded from the memory

An example to demonstrate the AdRotator control is as shown below:

The XML file for the AdRotator control is as mentioned below:

```
<Advertisements>
  <Ad>
    <ImageUrl>Tulips.jpg</ImageUrl>
    <NavigateUrl>http://www.gmail.com</NavigateUrl>
    <AlternateText>Gmail</AlternateText>
    <Keyword>Site</Keyword>
  </Ad>

  <Ad>
    <ImageUrl>LightHouse.jpg</ImageUrl>
    <NavigateUrl>http://www.google.com</NavigateUrl>
    <AlternateText>Google</AlternateText>
    <Keyword>Site</Keyword>
  </Ad>
</Advertisements>
```

The source code file for AdRotator file is as shown below:

```
<form id="form1" runat="server">
  <div>
    <asp:AdRotator ID="AdRotator1" runat="server"
    AdvertisementFile="~/XMLFile.xml" />
  </div>
</form>
```


The output is as shown below:



6. Calendar control

Calendar control is used to display a calendar in the browser. It displays a one month calendar allowing user to select dates and move to the different months. Some of the properties of the calendar control are as mentioned below:

Property	Description
AccessKey	It gets or sets the access key allowing to navigate to the Web Server control
Attributes	It gets the collection of arbitrary attributes not corresponding to control properties
BindingContainer	It gets the control that contains the controls data binding
BorderWidth	It gets or sets the border width of the Web Server control
Caption	It gets or sets the text value rendered as caption for the calendar
CellPadding	It gets or sets the space between the cell contents and cell border
CellSpacing	It gets or sets the space between cells
ClientID	It gets the ControlID for HTML markup generated by the ASP.NET
Context	It gets the HttpContext object associated with the server control
DataItemContainer	It gets a reference to the naming container if the container implements IDataItemContainer
Enabled	It gets or sets the value indicating the control is enabled
Font	It gets the font properties associated with the Web Server control
HasAttributes	It gets a value indicating the control has attributes set
Height	It gets or sets the height of the control
ID	It gets or sets the identifier assigned to the control
NamingContainer	It gets a reference to the server controls naming container
NextMonthText	It gets or sets the text displayed for next month navigation control
Page	It gets the reference to the Page instance of the server

	control
PrevMonthText	It gets or sets the text displayed for the previous month navigation control
SelectedDate	It gets or sets the selected date
SelectionMode	It gets or sets the date selection mode on the control
SelectMonthText	It gets or sets the text displayed for the month selection element
TabIndex	It gets or sets the tab index of the Web Server control
TemplateControl	It gets or sets the reference to the template that contains the control
Today'sDate	It gets or sets the value for today date
WeekendDayStyle	It gets or sets the style properties for the weekend date on the control

Some of the events related to the calendar control are as mentioned below:

Events	Description
DataBinding	It occurs when the server control binds to a data source
DayRender	It occurs when the day is created in the control hierarchy
Disposed	It occurs when the server control is released in the memory
Init	It occurs when the server control is initialized
Load	It occurs when the server control is loaded in the Page object
PreRender	It occurs after the control object is loaded but before rendering of the control
SelectionChanged	It occurs when the user selects a day, a week, or month by clicking the selector control
Unload	It occurs when the server control is unloaded from the memory

VisibleMonthChanged	It occurs when the user clicks the next and previous month navigation controls on the title heading
---------------------	---

The sample code for calendar control is as shown below:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form is="form1" runat="server">
        <div>
            <asp:Calendar ID="Calendar1" runat="server"
DayNameFormat="Full" BorderColor="Black"
onselectionchanged="Calendar1_SelectionChanged">
                <WeekendDayStyle BackColor="Brown" ForeColor="Red" />
                <DayHeaderStyle ForeColor="Aqua" />
                <TodayDayStyle BackColor="PaleGreen" />
            </asp:Calendar><br/>
            <asp:TextBox ID="TextBox1" runat="server"
Width="200px"></asp:TextBox>
        </div>
    </form>
</body>
</html>
```

The code behind file is as shown below:

```
public partial class Default2: System.Web.UI.Page
{
    protected void Page_Load( object sender, EventArgs e)
    {
        Calendar1.SelectedDate = DateTime.Now;
    }
    protected void Calendar1_SelectionChanged(object sender, EventArgs
e)
    {
        TextBox1.Text="selected date is"+Calendar1.SelectedDate;
    }
}
```

The output is:

February 2014						
Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
27	28	29	30	31	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	1	2
3	4	5	6	7	8	9

selected date is04/03/2014 00:00:00

7. Internet Explorer control

7.1 Introduction

Many web applications such as proxy web servers and multiple search engines are required to access the HTML pages of other websites. The classes WebClient, WebRequest, and WebResponse are usually used to perform these requirements in ASP.NET.

On the other hand, a Web Browser control is used in a Windows Forms application to browse web pages and other browser-enabled documents. Web Browser provides many events to track data processes, and many properties and methods to access and create new contents on the HTML element level.

Five advantages to using a Web Browser control in ASP.NET:

- Browses HTML pages while debugging;
- Handles cookies automatically without related coding;
- Allows scripts running and changing HTML contents. Conversely, it is impossible for the WebClient, WebRequest, and WebResponse classes to support script running;
- Accesses HTML elements and script variables programmatically;
- The HTML contents in a Web Browser control are always flat, but that in WebClient or WebResponse may be compressed.

Internet explorer control means control on web browser. It can be Internet Explorer or any other. We can understand this mechanism by Enable/Disable Java script in Microsoft Internet Explore

1. Click on the Tools Menu
 - a. Select “Internet Options”
2. Select the Security Tab
3. Click on “Custom Level”
4. Browse down to the “Scripting” section
5. Click on “Active Scripting” to enable / disable
6. Click “Yes” when asked “Are you sure you want to change the settings for this zone?”
7. Click on the “OK” button to close.
8. Press F5 or just reload the webpage.

8. ADO.NET Data Set

It is a collection of data tables that contain the data. It is used to fetch data without interacting with a Data Source that's why, it also known as **disconnected** data access method. It is an in-memory data store that can hold more than one table at the same time. We can use DataRelation object to relate these tables. The Data Set can also be used to read and write data as XML document.

ADO.NET provides a Data Set class that can be used to create Data Set object. It contains constructors and methods to perform data related operations.

8.1 DataSet Class Signature

```
public class DataSet : System.ComponentModel.MarshalByValueComponent, System.ComponentModel.IListSource, System.ComponentModel.ISupportInitializeNotification, System.Runtime.Serialization.ISerializable, System.Xml.Serialization.IXmlSerializable
```

8.2 DataSet Constructors

Constructor	Description
DataSet()	It is used to initialize a new instance of the DataSet class.
DataSet(String)	It is used to initialize a new instance of a DataSet class with the given name.
DataSet(SerializationInfo, StreamingContext)	It is used to initialize a new instance of a DataSet class that has the given serialization information and context.
DataSet(SerializationInfo, StreamingContext, Boolean)	It is used to initialize a new instance of the DataSet class.

8.3 Data Set Properties

Properties	Description
CaseSensitive	It is used to check whether DataTable objects are case-sensitive or not.
DataSetName	It is used to get or set name of the current DataSet.
DefaultViewManager	It is used to get a custom view of the data contained in the DataSet to allow filtering and searching.
HasErrors	It is used to check whether there are errors in any of the DataTable objects within this DataSet.
IsInitialized	It is used to check whether the DataSet is initialized or not.
Locale	It is used to get or set the locale information used to compare strings within the table.
Namespace	It is used to get or set the namespace of the DataSet.
Site	It is used to get or set an ISite for the DataSet.
Tables	It is used to get the collection of tables contained in the DataSet.

8.4 Data Set Methods

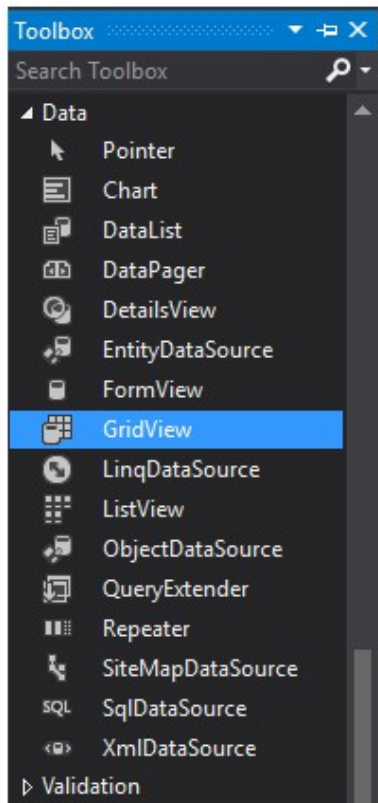
The following table contains some commonly used methods of DataSet.

Method	Description
BeginInit()	It is used to begin the initialization of a DataSet that is used on a form.
Clear()	It is used to clear the DataSet of any data by removing all rows in all tables.
Clone()	It is used to copy the structure of the DataSet.
Copy()	It is used to copy both the structure and data for this DataSet.

CreateDataReader(DataTable[])	It returns a DataTableReader with one result set per DataTable.
CreateDataReader()	It returns a DataTableReader with one result set per DataTable.
EndInit()	It ends the initialization of a DataSet that is used on a form.
GetXml()	It returns the XML representation of the data stored in the DataSet.
GetXmlSchema()	It returns the XML Schema for the XML representation of the data stored in the DataSet.
Load(IDataReader, LoadOption, DataTable[])	It is used to fill a DataSet with values from a data source using the supplied IDataReader.
Merge(DataSet)	It is used to merge a specified DataSet and its schema into the current DataSet.
Merge(DataTable)	It is used to merge a specified DataTable and its schema into the current DataSet.
ReadXml(XmlReader, XmlReadMode)	It is used to read XML schema and data into the DataSet using the specified XmlReader and XmlReadMode.
Reset()	It is used to clear all tables and removes all relations, foreign constraints, and tables from the DataSet.
WriteXml(XmlWriter, XmlWriteMode)	It is used to write the current data and optionally the schema for the DataSet using the specified XmlWriter and XmlWriteMode.

Example:

Here, in this example, we are implementing **DataSet** and displaying data into a gridview. Create a web form and drag a gridview from the **toolbox** to the form. We can find it inside the data category.



// DataSetDemo.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="DataSetDemo.aspx.
cs"
Inherits="DataSetExample.DataSetDemo" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title></title>
</head>
<body>
<form id="form1" runat="server">
<div>
</div>
<asp:GridView ID="GridView1" runat="server" CellPadding="4" ForeColor="#33
3333" GridLines="None">
<AlternatingRowStyle BackColor="White" />
<EditRowStyle BackColor="#2461BF" />
<FooterStyle BackColor="#507CD1" Font-
Bold="True" ForeColor="White" />
```

```

        <HeaderStyle BackColor="#507CD1" Font-
Bold="True" ForeColor="White" />
        <PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Cente
r" />
        <RowStyle BackColor="#EFF3FB" />
        <SelectedRowStyle BackColor="#D1DDF1" Font-
Bold="True" ForeColor="#333333" />
        <SortedAscendingCellStyle BackColor="#F5F7FB" />
        <SortedAscendingHeaderStyle BackColor="#6D95E1" />
        <SortedDescendingCellStyle BackColor="#E9EBEF" />
        <SortedDescendingHeaderStyle BackColor="#4870BE" />
    </asp:GridView>
</form>
</body>
</html>

```

CodeBehind

```

// DataSetDemo.aspx.cs
using System;
using System.Data.SqlClient;
using System.Data;
namespace DataSetExample
{
    public partial class DataSetDemo : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            using (SqlConnection con = new SqlConnection("data source=.; database=stu
dent; integrated security=SSPI"))
            {
                SqlDataAdapter sde = new SqlDataAdapter("Select * from student", con);
                DataSet ds = new DataSet();
                sde.Fill(ds);
                GridView1.DataSource = ds;
                GridView1.DataBind();
            }
        }
    }
}

```

9. ADO.NET Data Adapter

The Data Adapter works as a bridge between a Data Set and a data source to retrieve data. Data Adapter is a class that represents a set of SQL commands and a database connection. It can be used to fill the Data Set and update the data source.

9.1 Data Adapter Class Signature

```
public class DataAdapter : System.ComponentModel.Component, System.Data.IDataAdapter
```

9.2 Data Adapter Constructors

Constructors	Description
Data Adapter()	It is used to initialize a new instance of a Data Adapter class.
Data Adapter(Data Adapter)	It is used to initialize a new instance of a Data Adapter class from an existing object of the same type.

9.3 Methods

Method	Description
CloneInternals()	It is used to create a copy of this instance of Data Adapter.
Dispose(Boolean)	It is used to release the unmanaged resources used by the Data Adapter.
Fill(DataSet)	It is used to add rows in the Data Set to match those in the data source.
FillSchema(DataSet, SchemaType, String, IDataReader)	It is used to add a Data Table to the specified Data Set.
GetFillParameters()	It is used to get the parameters set by the user when executing an SQL SELECT statement.

ResetFillLoadOption()	It is used to reset FillLoadOption to its default state.
ShouldSerializeAcceptChangesDuringFill()	It determines whether the AcceptChangesDuringFill property should be persisted or not.
ShouldSerializeFillLoadOption()	It determines whether the FillLoadOption property should be persisted or not.
ShouldSerializeTableMappings()	It determines whether one or more DataTableMapping objects exist or not.
Update(DataSet)	It is used to call the respective INSERT, UPDATE, or DELETE statements.

Example

// DataSetDemo.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="DataSetDemo.aspx.cs"
Inherits="DataSetExample.DataSetDemo" %>
<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
    <div>

    </div>
    <asp:GridView ID="GridView1" runat="server" CellPadding="3" BackColor="#DE
BA84"
      BorderColor="#DEBA84" BorderStyle="None" BorderWidth="1px" CellSpacing="2"
>
      <FooterStyle BackColor="#F7DFB5" ForeColor="#8C4510" />
      <HeaderStyle BackColor="#A55129" Font-
Bold="True" ForeColor="White" />
      <PagerStyle ForeColor="#8C4510" HorizontalAlign="Center" />
      <RowStyle BackColor="#FFF7E7" ForeColor="#8C4510" />
```

```

        <SelectedRowStyle BackColor="#738A9C" Font-
Bold="True" ForeColor="White" />
        <SortedAscendingCellStyle BackColor="#FFF1D4" />
        <SortedAscendingHeaderStyle BackColor="#B95C30" />
        <SortedDescendingCellStyle BackColor="#F1E5CE" />
        <SortedDescendingHeaderStyle BackColor="#93451F" />
    </asp:GridView>
</form>
</body> </html>

```

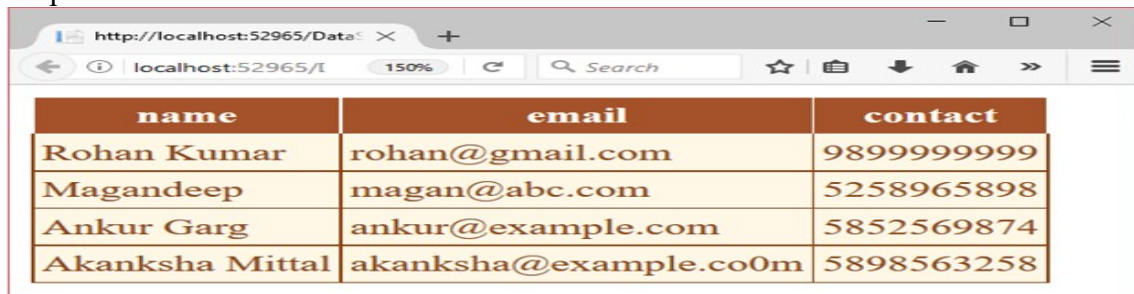
Code Behind

```

using System;
using System.Data.SqlClient;
using System.Data;
namespace DataSetExample
{
    public partial class DataSetDemo : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            using (SqlConnection con = new SqlConnection("data source=.; database=stud
ent; integrated security=SSPI"))
            {
                SqlDataAdapter sde = new SqlDataAdapter("Select * from student", con);
                DataSet ds = new DataSet();
                sde.Fill(ds);
                GridView1.DataSource = ds;
                GridView1.DataBind();
            }
        }
    }
}

```

Output:



The screenshot shows a web browser window with the address bar displaying 'http://localhost:52965/DataSetDemo.aspx'. The browser's address bar also shows 'localhost:52965/' and a search bar. The main content area displays a table with three columns: 'name', 'email', and 'contact'. The table contains four rows of data:

name	email	contact
Rohan Kumar	rohan@gmail.com	9899999999
Magandeep	magan@abc.com	5258965898
Ankur Garg	ankur@example.com	5852569874
Akanksha Mittal	akanksha@example.co0m	5898563258

References:

- <https://support.microsoft.com/en-in/help/307467/how-to-create-an-asp-net-application-from-multiple-projects-for-team-d>
- <http://www.wideskills.com/aspnet/validation-controls>
- <https://www.javatpoint.com/ado-net-dataadapter>