UNIT - I

Introduction to Dot Net

NET is a software framework that is designed and developed by Microsoft. The first version of the .Net framework was 1.0 came in the year 2002. In easy words, it is a virtual machine for compiling and executing programs written in different languages like C#, VB.Net, etc.

Developed by Microsoft, Dot NET (also known as .NET) is a framework that makes **application development** a bit easy task for developers. The framework supports the development as well as maintenance of modern-days applications and **XML (Extensible Markup Language)** web services. .NET offers a highly consistent object-oriented programming environment to **dedicated developers** and is used to build applications that can run on multiple platforms.

It is used to develop Form-based applications, Web-based applications, and Web services. There is a variety of programming languages available on the .Net platform, VB.Net and C# being the most common ones. It is used to build applications for Windows, phones, web, etc. It provides a lot of functionalities and also supports industry standards.

.NET Framework supports more than 60 programming languages in which 11 programming languages are designed and developed by Microsoft. The remaining Non-Microsoft Languages are supported by .NET Framework but not designed and developed by Microsoft.

One of the interesting about .NET framework is that it supports the creation of cross-platform applications that can smoothly run across server platforms such as Windows, Linux etc.

This framework can be used to design, develop, compile, **build and deploy a mobile application** with its comprehensive range of compilers, code libraries, support programs and APIs. The various components of .NET aids in the development of the customized project.

.NET is widely used by developers for the creation of services and apps on multiple devices and operating systems. One of the aspects that make .NET to stand is the range of features offered by the application.

Features of .NET

- It is a platform neutral framework.
- It is a layer between the operating system and the programming language.
- It supports many programming languages, including VB.NET, C# etc.
- .NET provides a common set of class libraries, which can be accessed from any .NET based programming language. There will not be separate set of classes and libraries for each language. If you know any one .NET language, you can write code in any .NET language.

- In future versions of Windows, .NET will be freely distributed as part of operating system and users will never have to install .NET separately.

Features of .Net Framework

- Platform builds for app developers.
- Framework supports several languages and cross-language combinations.
- It also includes an IDE for writing programs.
- It is a set of utilities or in other words, building blocks of the user application system.
- The framework provides a Graphical User Interface in the form of GUI.
- .NET is an individual platform but it uses Mono Compilation System (MCS) where it is a mid-level interface.

Advantages of .NET

- It allows the use of multiple languages.
- It has horizontal scalability.
- .NET creates a unified environment that allows developers to create programs in C++, Java or Visual Basic.
- Interfaces easily with Windows or Microsoft.
- All tools and IDEs have been pre-tested and are easily available in the Microsoft Developer Network.
- Language integration is simple, as we can call methods from C# to VB.NET.

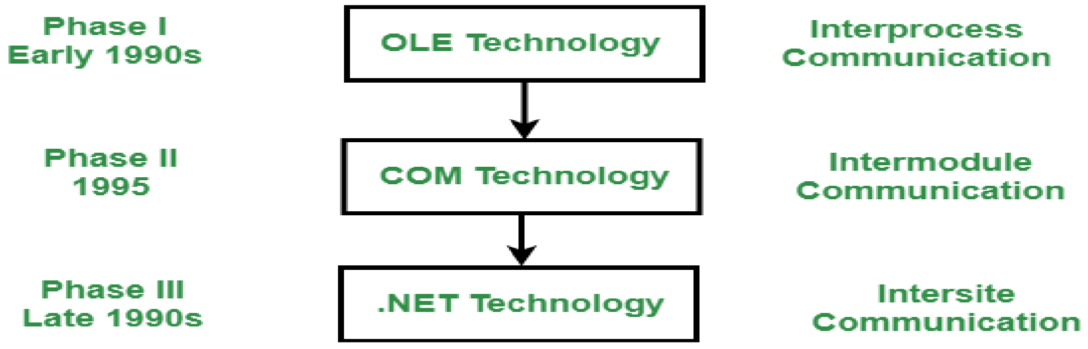
Disadvantages of .NET

- Limited object-relational support as it comes only with Entity Framework.
- Does not come with multi platform support from Microsoft, and is not available right
- after installing Visual Studio.
- The managed code can be slower than native code.
- Involves a vendor lock-in, and future development is solely dependent on Microsoft.
- Migrating applications to .NET can be expensive.

Phases of .Net

There are three significant phases of the development of .NET technology.

- **OLE Technology**
- **COM Technology**
- **.NET Technology**



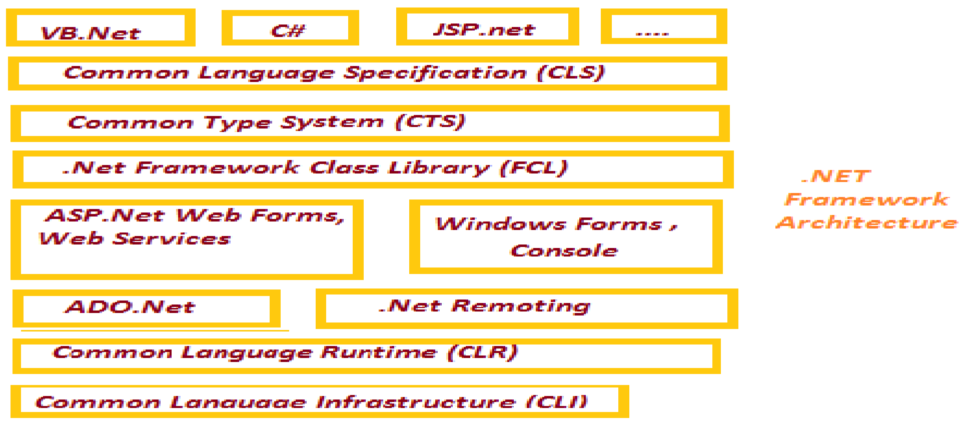
OLE Technology: OLE (Object Linking and Embedding) is one of the technologies of Microsoft's component document. Basically, its main purpose is to link elements from different applications within each other.

COM Technology: The technology of the Microsoft Windows family of the operating system, Microsoft COM (Common Object Model) enables various software components to communicate. COM is mostly used by developers for various purposes like creating reusable software components, linking components together to build applications, and also taking advantage of Windows services. The objects of COM can be created with a wide range of programming languages.

.NET Technology: .NET technology of collection or set of technologies to develop and windows and web applications. The technology of .Net is developed by Microsoft and was launched in Feb. 2002, by basic definition, Microsoft's new Internet Strategy. It was originally called NGWS (Next Generation Web Services). It is considered to be as one of the powerful, popular and very much useful Internet Technology available today.

.Net Framework Architecture

.Net Framework Architecture is a programming model for the .Net platform that provides an execution environment and integration with various programming languages for simple development and deployment of various Windows and desktop applications. It consists of class libraries and reusable components.


VB.Net
C#
JSP.net
....
Common Language Specification (CLS)
Common Type System (CTS)
.Net Framework Class Library (FCL)
ASP.Net Web Forms, Web Services
Windows Forms , Console
ADO.Net
.Net Remoting
Common Language Runtime (CLR)
Common Language Infrastructure (CLI)
.NET Framework Architecture

Components of .Net Framework

.Net framework architecture is based on some key components that include;

- CLS- Common Language Specification
- CTS- Common Type Specification
- FCL- .Net Framework Class Library
- CLR- Common Language Runtime
- CLI- Common Language Infrastructure

I. CLS- Common Language Specification

Common Language Specification or CLS includes a set of procedures that specifies a .NET language. Apps build using several coding languages like C# and VB.NET, are shorten down to CLS or Common Language Specification.

II. CLI- Common Language Infrastructure

The CLI or Common Language Infrastructure provides a language-independent platform for app development and its performance. It also includes the function for exception handling, waste collection, security, etc.

The CLI has the following key features:

Handling Anomaly- Anomalies are errors/bugs that occur while the application is executed.

Examples of anomalies or exceptions:

In case an application tries to open a file on the local system, but the file is not present.

In case the application tries to raise some records from a database, but the connection to the database becomes invalid.

Waste Collection – Collecting waste or garbage is the process of deleting unwanted resources when these are no longer needed.

Examples of the waste collection:

A file handle that is no longer useful. In case the application has completed all operations on a file, then the file handle may no longer be useful.

Besides, the database connection that doesn't have longer use. If the application has completed all its operations on a database, then the database connection may no longer be used.

Working with different languages

Language – The first level includes the programming language where the most common ones are VB.Net and C# languages.

Compiler – The compiler that exists here may be separate for each coding language. So basically the VB.Net language will have an individual VB.Net compiler. Likely, for C# also user will have another compiler.

CLI– the CLI is the last layer in the net framework. This would be useful to run a .net program built in any coding language. So the afterward compiler will send the coding to the CLI layer to run the .Net application.

III. CTS- Common Type Specification

The Common Type System or CTS includes all types of data that are supported by multiple languages. For Example, .NET types.System.Int32, System. Decimal, etc.

IV. CLR- Common Language Runtime

CLR/Common Language Runtime is the fundamental and VM component of the .Net framework. It provides the run-time environment within the .NET framework that runs the programs. By this, it helps the users in making the development process easier. Besides, it also provides different types of services like remote service, type-safety, waste management, durability, etc. Generally, it is responsible for managing the performance of .NET codes nevertheless of any .NET coding language.

V. FCL- .Net Framework Class Library

The FCL is the collection of all kinds of class libraries, functions, and methods that can be combined with CLR. These can be reusable also. They are also known as the "Assemblies". It is similar to the header files within C or C++ and packages in the java. The FCL provides different types for strings, dates, numbers, etc. Moreover, the Class Library consists of APIs for reading and writing files, database connections, drawing, and more.

For example, there is a class library having various methods to handle all file-level processes. It means there is a function useful to read the text from a file. In the same way, there is a method to write a text to file.

Generally, the .NET apps are written in the C# or VB coding languages. Besides, the code is joined into a language-skeptic CLI or Common Intermediate Language.

Code Manager

In the .Net framework, the code manager induces class loader for performing operations.

.NET platform supports two types of coding;

- Managed Code
- Unmanaged Code

Managed Code

This (managed code) is the resource that exists within the user's application domain. The resources that are inside the domain are much faster. The code/program that is developed within the
.NET framework is a managed code. This code is directly performed by CLR- Common Language runtime using the managed code execution. Any coding language written in this framework is a managed code. The code uses CLR that takes care of user apps by managing memory, managing security, allows multi-language debugging, etc.

Unmanaged Code

This type of code is that which is developed outside the software framework. Unmanaged applications do not run under the direct control of the CLR. There are certain languages such as C++ are useful to write such type of applications. For instance, accessing low-level functions of the OS. There are a few examples of the unmanaged code such as background affinity with the code of VB, ASP, and COM.

This code can be source code & compile code. Moreover, this type of code is performed and used with help of wrapper classes.

What is ASP.NET? and it's ARCHITECTURE

What is ASP.Net?

ASP.Net is a web development platform provided by Microsoft. It is used for creating web-based applications. ASP.Net was first released in the year 2002.

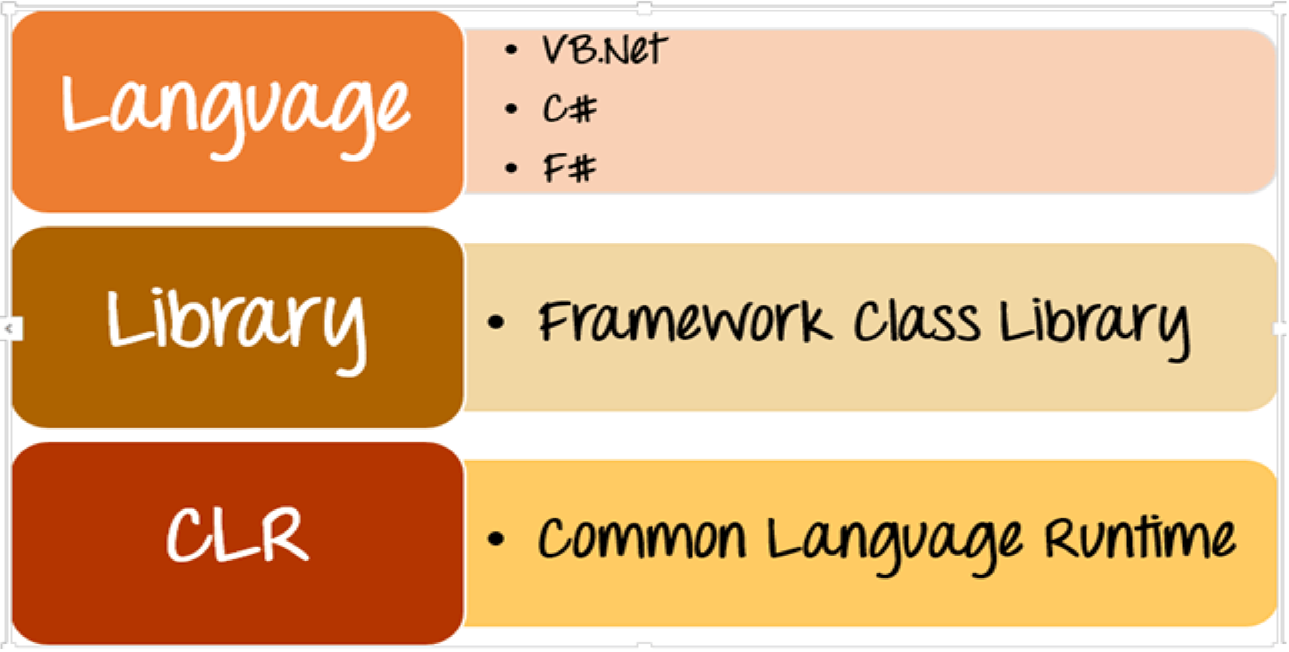
The first version of ASP.Net deployed was 1.0. The most recent version of ASP.Net is version 4.6. ASP.Net is designed to work with the HTTP protocol. This is the standard protocol used across all web applications.

ASP.Net applications can also be written in a variety of .Net languages. These include C#, VB.Net, and J#. In this chapter, you will see some basic fundamental of the .Net framework.

The full form of ASP is Active Server Pages, and .NET is Network Enabled Technologies.

ASP.NET Architecture and its Components

ASP.Net is a framework which is used to develop a Web-based application. The basic architecture of the ASP.Net framework is as shown below.



ASP.NET Architecture Diagram

The architecture of the.Net framework is based on the following key components

1. **Language** – A variety of languages exists for .net framework. They are VB.net and C#. These can be used to develop web applications.
2. **Library** – The .NET Framework includes a set of standard class libraries. The most common library used for web applications in .net is the Web library. The web library has all the necessary components used to develop.Net web-based applications.
3. **Common Language Runtime** – The Common Language Infrastructure or CLI is a platform. .Net programs are executed on this platform. The CLR is used for performing key activities. Activities include Exception handling and Garbage collection.

Below are some of the key characteristics of the ASP.Net framework

1. **Code Behind Mode** – This is the concept of separation of design and code. By making this separation, it becomes easier to maintain the ASP.Net application. The general file type of an ASP.Net file is aspx. Assume we have a web page called MyPage.aspx. There will be another file called MyPage.aspx.cs which would denote the code part of the page. So Visual Studio creates separate files for each web page, one for the design part and the other for the code.
2. **State Management** – ASP.Net has the facility to control state management. HTTP is known as a stateless protocol. Let's take an example of a shopping cart application. Now, when a user decides what he wants to buy from the site, he will press the submit button. The application needs to remember the items the user choose for the purchase. This is known as remembering the state of an application at a current point in time. HTTP is a stateless protocol. When the user goes to the purchase page, HTTP will not store the information on the cart items. Additional coding needs to be done to ensure that the cart

items can be carried forward to the purchase page. Such an implementation can become complex at times. But ASP.Net can do state management on your behalf. So ASP.Net can remember the cart items and pass it over to the purchase page.

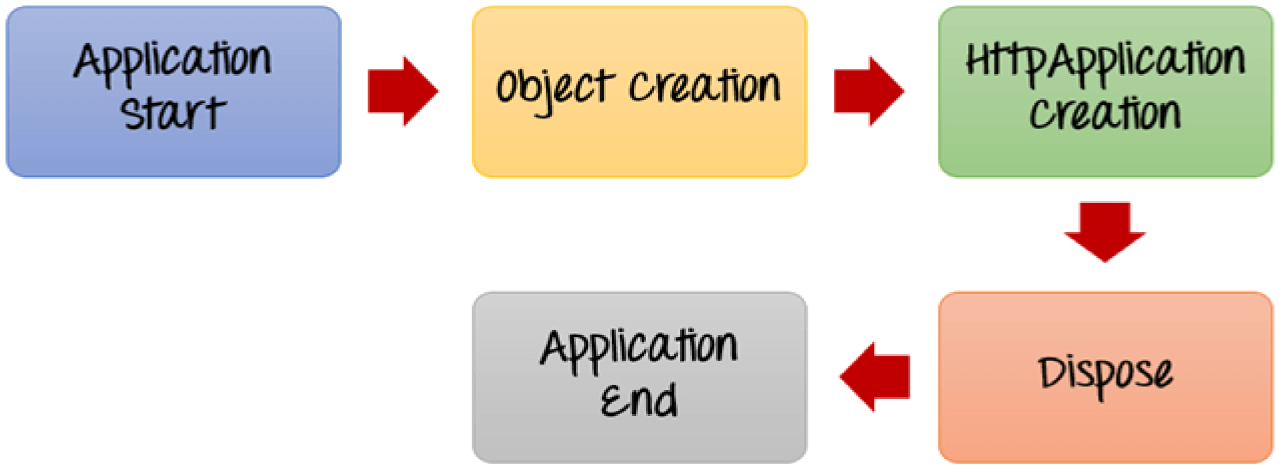
3. **Caching** – ASP.Net can implement the concept of Caching. This improve's the performance of the application. By caching those pages which are often requested by the user can be stored in a temporary location. These pages can be retrieved faster and better responses can be sent to the user. So caching can significantly improve the performance of an application.

ASP.Net is a development language used for constructing web-based applications. ASP.Net is designed to work with the standard HTTP protocol.

What is ASP.Net Lifecycle?

When an ASP.Net application is launched, there are series of steps which are carried out. These series of steps make up the lifecycle of the application.

Let's look at the various stages of a typical page lifecycle of an ASP.Net Web Application.



ASP.Net Lifecycle

1) Application Start – The life cycle of an ASP.NET application starts when a request is made by a user. This request is to the Web server for the ASP.Net Application. This happens when the first user normally goes to the home page for the application for the first time. During this time, there is a method called Application_start which is executed by the web server. Usually, in this method, all global variables are set to their default values.

2) Object creation – The next stage is the creation of the HttpContext, HttpRequest & HttpResponse by the web server. The HttpContext is just the container for the HttpRequest and HttpResponse objects. The HttpRequest object contains information about the current request, including cookies and browser information. The HttpResponse object contains the response that is sent to the client.

3) HttpApplication creation – This object is created by the web server. It is this object that is used to process each subsequent request sent to the application. For example, let's assume we have 2 web applications. One is a shopping cart application, and the other is a news website. For each application, we would have 2 HttpApplication objects created. Any further requests to each website would be processed by each HttpApplication respectively.

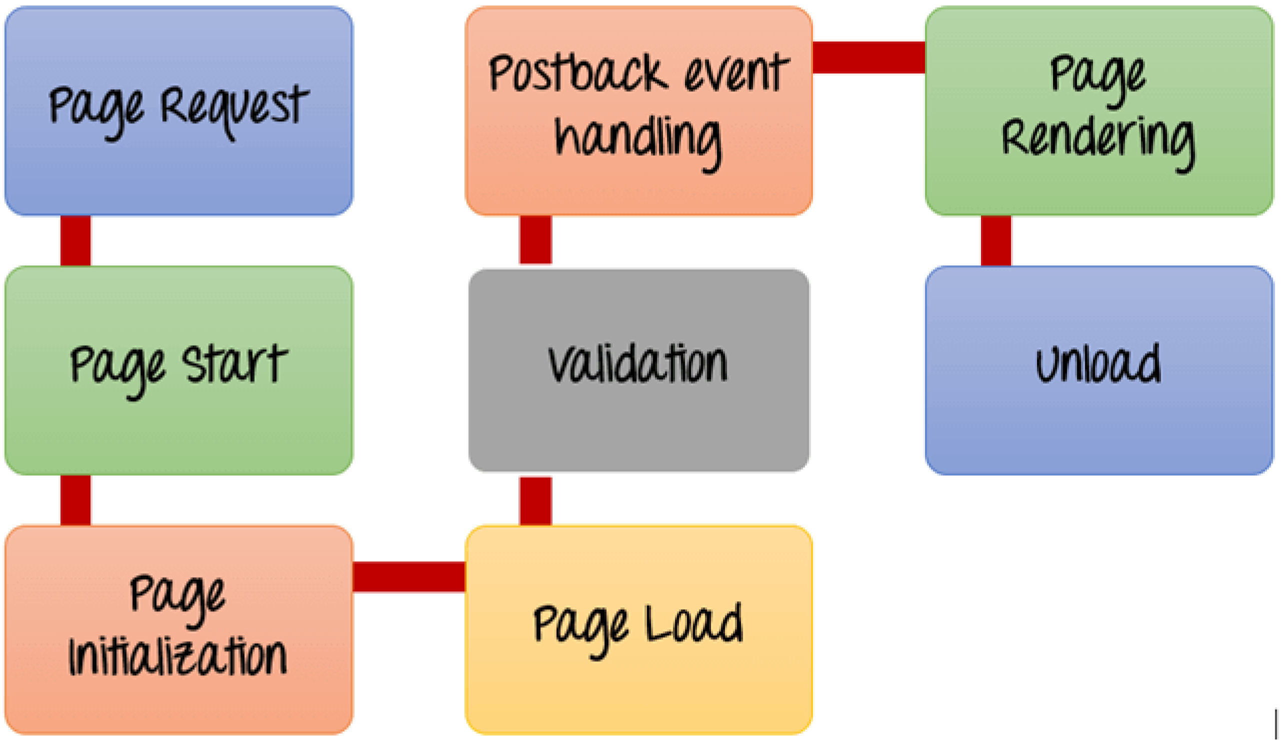
4) Dispose – This event is called before the application instance is destroyed. During this time, one can use this method to manually release any unmanaged resources.

5) Application End – This is the final part of the application. In this part, the application is finally unloaded from memory.

What is ASP.Net Page Lifecycle?

When an ASP.Net page is called, it goes through a particular lifecycle. This is done before the response is sent to the user. There are series of steps which are followed for the processing of an ASP.Net page.

Let's look at the various stages of the lifecycle of an ASP.Net web page.



ASP.Net Page Lifecycle

1. **Page Request**– This is when the page is first requested from the server. When the page is requested, the server checks if it is requested for the first time. If so, then it needs to compile the page, parse the response and send it across to the user. If it is not the first time the page is requested, the cache is checked to see if the page output exists. If so, that response is sent to the user.
2. **Page Start** – During this time, 2 objects, known as the Request and Response object are created. The Request object is used to hold all the information which was sent when the

page was requested. The Response object is used to hold the information which is sent back to the user.

3. **Page Initialization** – During this time, all the controls on a web page is initialized. So if you have any label, textbox or any other controls on the web form, they are all initialized.
4. **Page Load** – This is when the page is actually loaded with all the default values. So if a textbox is supposed to have a default value, that value is loaded during the page load time.
5. **Validation** – Sometimes there can be some validation set on the form. For example, there can be a validation which says that a list box should have a certain set of values. If the condition is false, then there should be an error in loading the page.
6. **Post back event handling** – This event is triggered if the same page is being loaded again. This happens in response to an earlier event. Sometimes there can be a situation that a user clicks on a submit button on the page. In this case, the same page is displayed again. In such a case, the Postback event handler is called.
7. **Page Rendering** – This happens just before all the response information is sent to the user. All the information on the form is saved, and the result is sent to the user as a complete web page.
8. **Unload** – Once the page output is sent to the user, there is no need to keep the ASP.net web form objects in memory. So the unloading process involves removing all unwanted objects from memory.

ASP.NET Controls: Checkbox, Radio Button, List Box, Textbox, Label

Adding ASP.Net Controls to Web Forms

ASP.Net has the ability to add controls to a form such as textboxes and labels.

Let's look at the other controls available for Web forms and see some of their common properties.

In our example, we will create one form which will have the following functionality.

1. The ability for the user to enter his name.
2. An option to choose the city in which the user resides in
3. The ability for the user to enter an option for the gender.
4. An option to choose a course which the user wants to learn. There will be choices for both C# and ASP.Net

Table of Content:

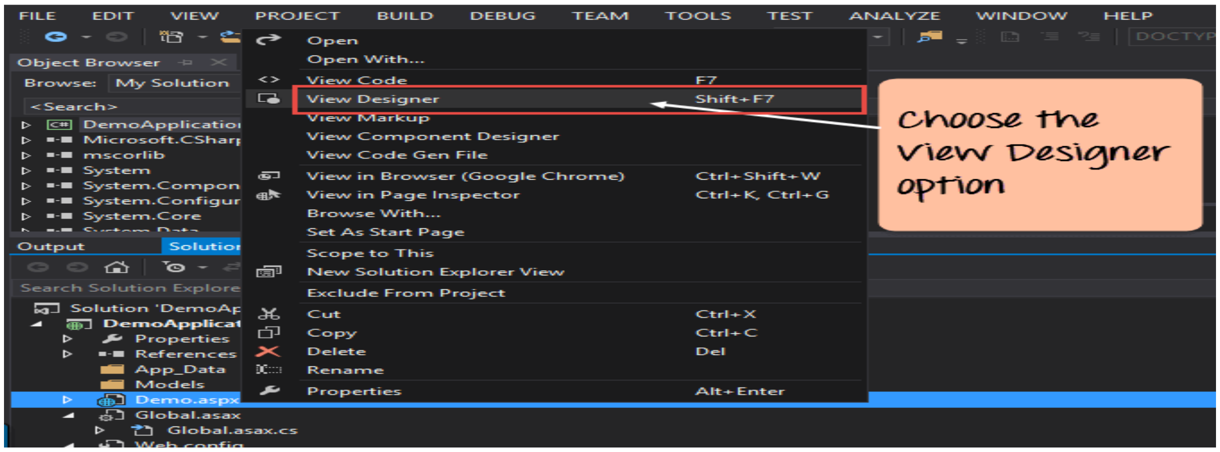
- Button
- Event Handler in ASP.Net

Let's look at each control in detail. Let's add them to build the form with the above-mentioned functionality.

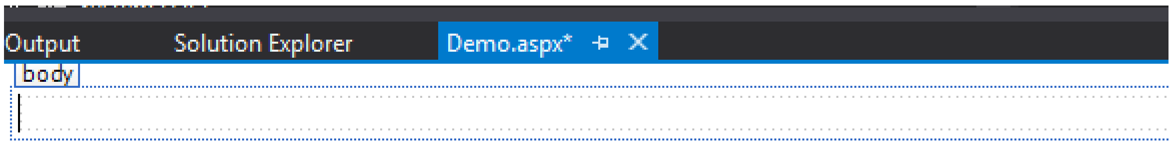
Step 1) The first step is to open the Forms Designer for the Demo web form. Once you do this, you will be able to drag controls from the toolbox to the Web form.

To open the Designer web form,

- Right-click the Demo.aspx file in the Solution Explorer and
- Choose the menu option View Designer.



Once you perform the above step, you will be able to see your Form Designer as shown below.

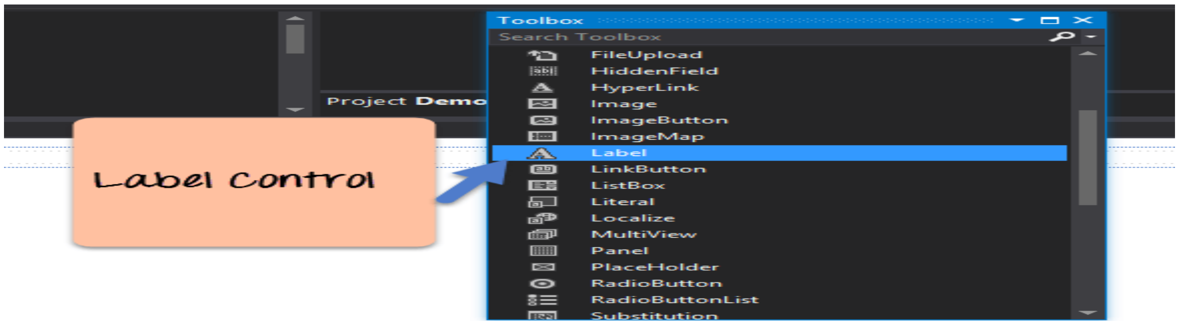


Now let's start adding our controls one by one

Label Control

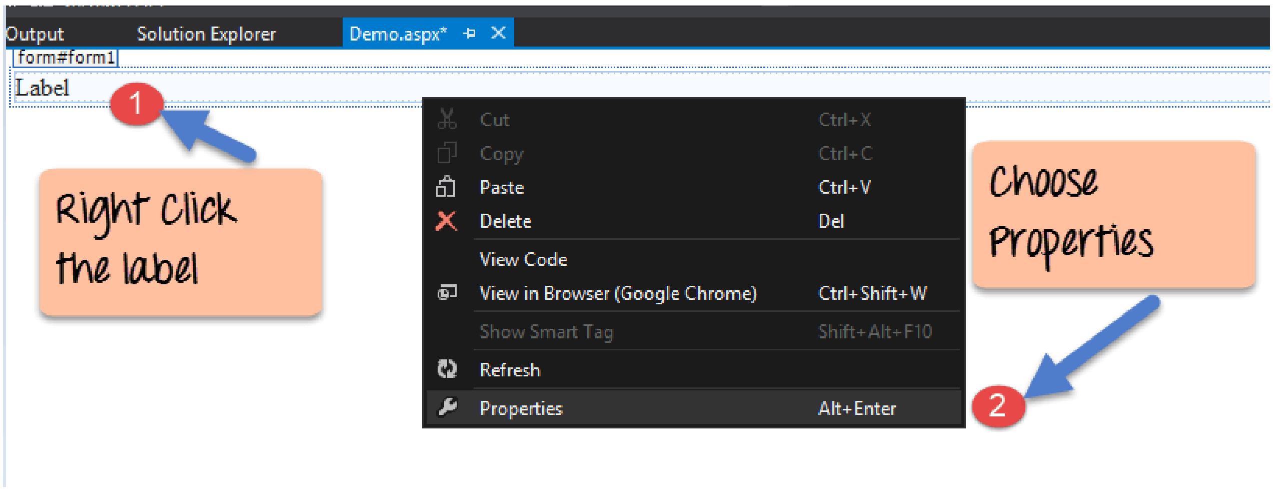
The label control is used to display a text or a message to the user on the form. The label control is normally used along with other controls. Common examples is wherein a label is added along with the textbox control. The label gives an indication to the user on what is expected to fill up in the textbox. Let's see how we can implement this with an example shown below. We will use a label called 'name.' This will be used in conjunction with the textbox controls, which will be added in the later section.

Step 1) The first step is to drag the ‘label’ control on to the Web Form from the toolbox as shown below.

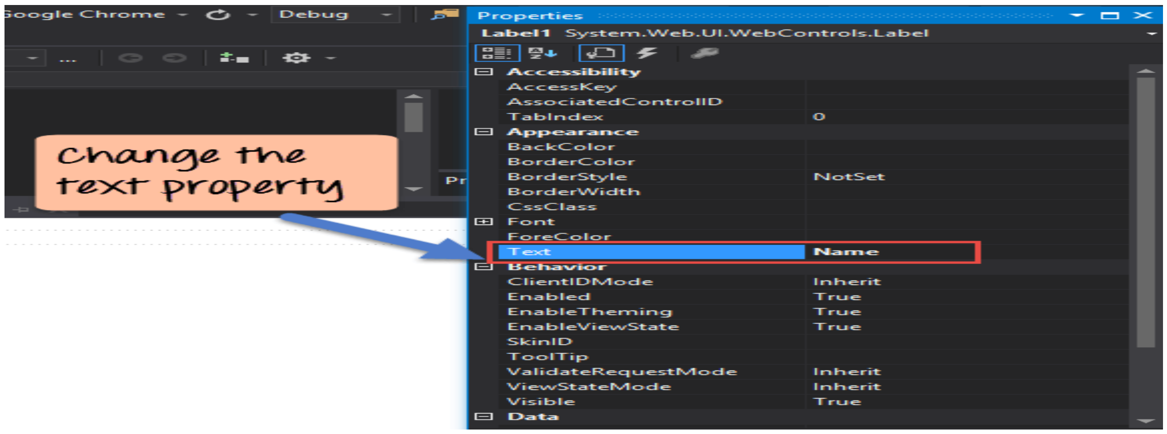


Step 2) Once the label has been added, follow the following steps.

1. Go to the properties window by right-clicking on the label control
2. Choose the Properties menu option

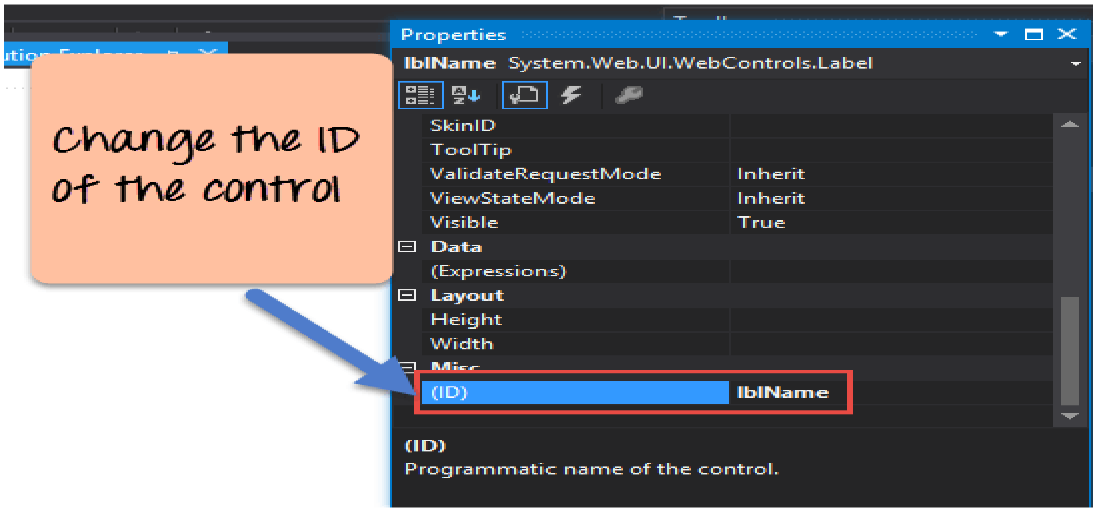


Step 3) from the properties window, change the name of the Text property to Name



Accessibility	
AccessKey	
AssociatedControlID	
TabIndex	0
Appearance	
BackColor	
BorderColor	
BorderStyle	NotSet
BorderWidth	
CssClass	
Font	
ForeColor	
Text	Name
Behavior	
ClientIDMode	Inherit
Enabled	True
EnableTheming	True
EnableViewState	True
SkinID	
ToolTip	
ValidateRequestMode	Inherit
ViewStateMode	Inherit
Visible	True
Data	

Similarly, also change the ID property value of the control to lblName. By specifying a meaningful ID to controls, it becomes easier to access them during the coding phase. This is shown below.

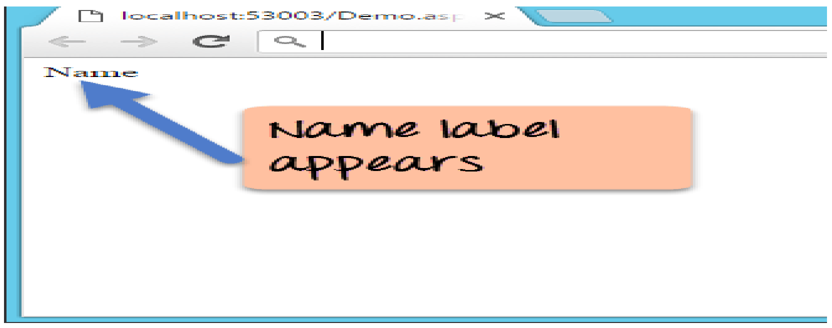

Change the ID of the control
Properties
lblName System.Web.UI.WebControls.Label

SkinID	
ToolTip	
ValidateRequestMode	Inherit
ViewStateMode	Inherit
Visible	True
Data	
(Expressions)	
Layout	
Height	
Width	
Misc	
(ID)	lblName

(ID)
Programmatic name of the control.

Once you make the above changes, you will see the following output

Output:-

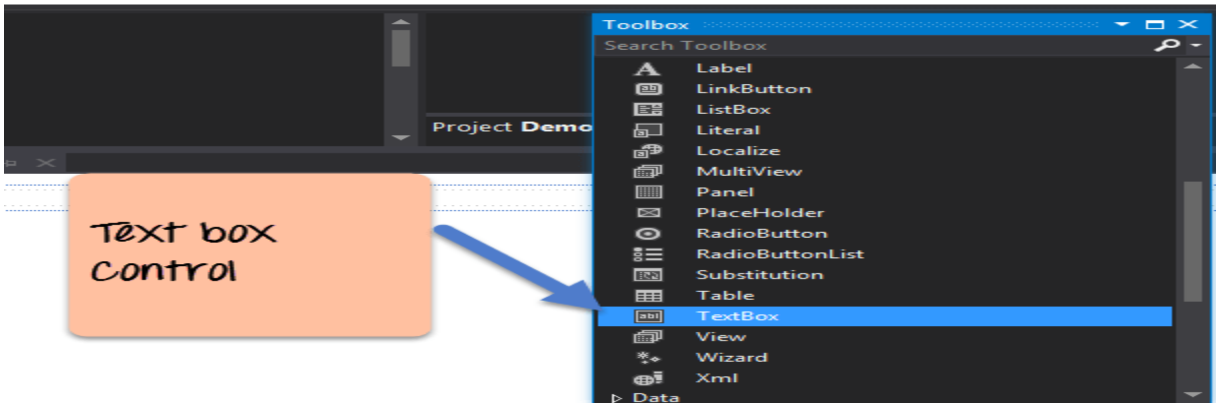


You will see that the Name label appears on the Web Form.

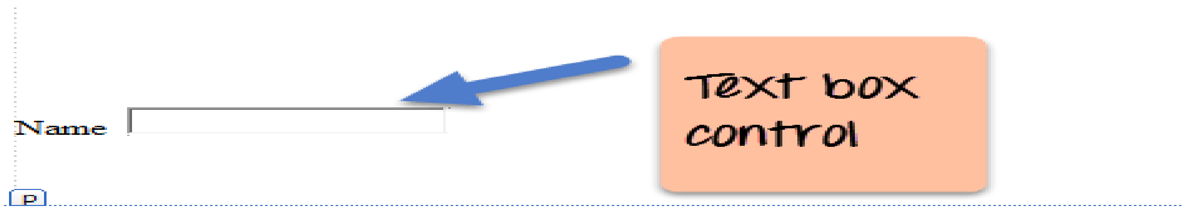
Textbox

A text box is used for allowing a user to enter some text on the Web form application. Let's see how we can implement this with an example shown below. We will add one textbox to the form in which the user can enter his name.

Step 1) The first step is to drag the textbox control onto the Web Form from the toolbox as shown below

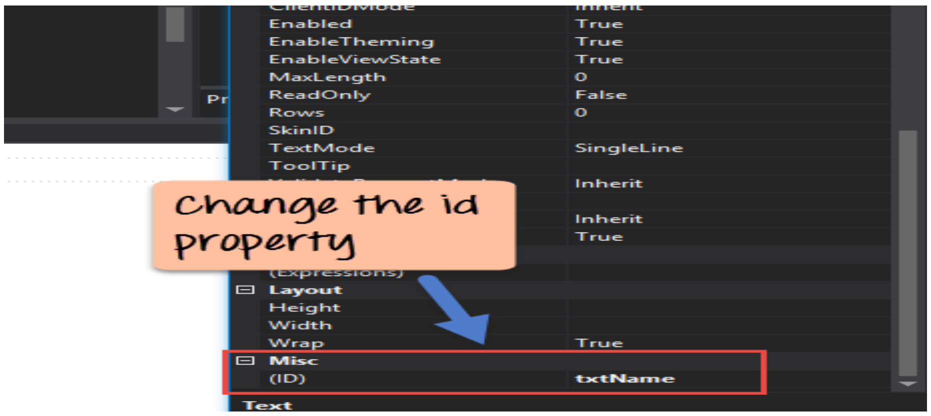


Below is how this would look in the forms designer once the Textbox control is on the form



Step 2) Once the Textbox has been added, you have to change the ID property.

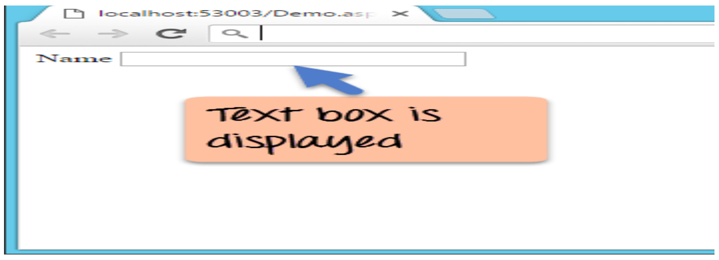
- Go to the properties window by right-clicking on the Textbox control and
- Choose properties then
- Change the id property of the textbox to txtName.



```
ClientIDMode            Inherit
Enabled                 True
EnableTheming           True
EnableViewState         True
MaxLength               0
ReadOnly                False
Rows                    0
SkinID
TextMode                SingleLine
ToolTip
                        Inherit

                        Inherit
                        True
(Expressions)
⊟ Layout
Height
Width
Wrap                    True
⊟ Misc
(ID)                    txtName
Text
```

Once you make the above changes, you see the following output.

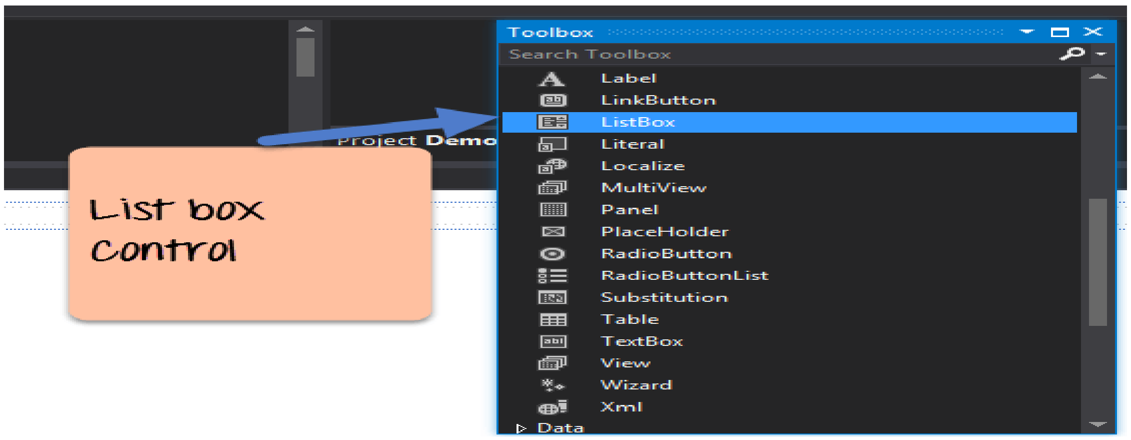
Output:-



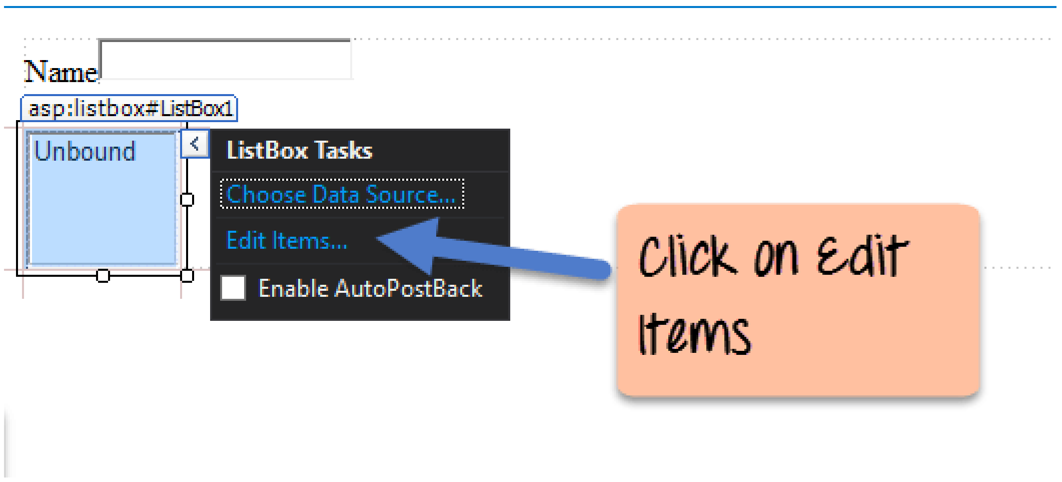
List box

A Listbox is used to showcase a list of items on the Web form. Let's see how we can implement this with an example shown below. We will add a list box to the form to store some city locations.

Step 1) The first step is to drag the list box control on to the Web Form from the toolbox as shown below

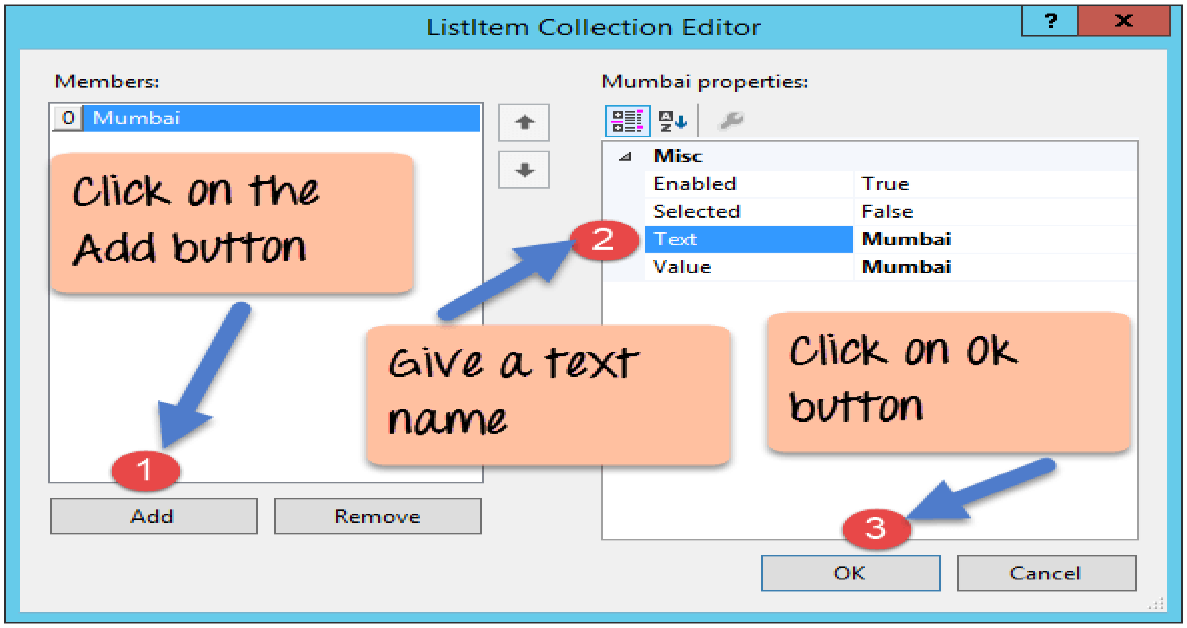


Step 2) Once you drag the listbox to the form, a separate side menu will appear. In this menu choose the 'Edit Items' menu.



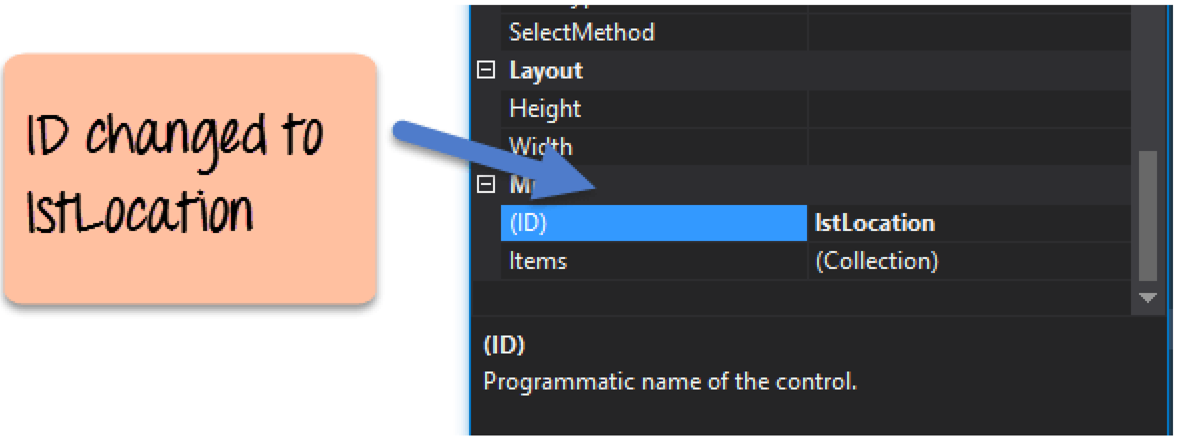
Step 3) You will now be presented with a dialog box in which you can add the list items to the listbox.

1. Click on the Add button to add a list item.
2. Give a name for the text value of the list item – In our case Mumbai. Repeat steps 1 and 2 to add list items for Mangalore and Hyderabad.
3. Click on the OK button



Enabled	True
Selected	False
Text	Mumbai
Value	Mumbai

Step 4) Go to the properties window and change the ID property value of the control to lstLocation.

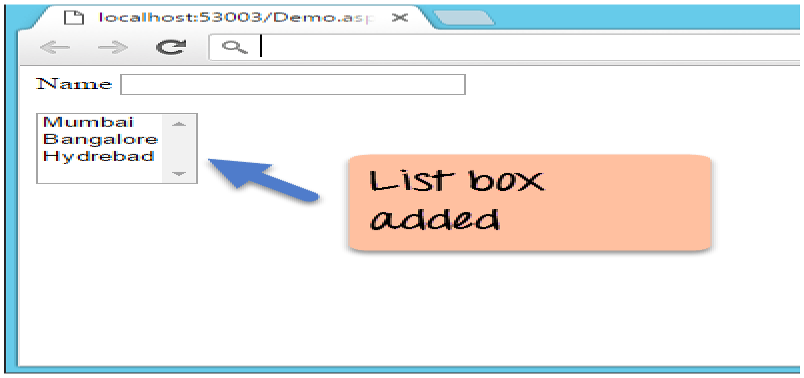


(ID)	lstLocation
Items	(Collection)

(ID)
Programmatic name of the control.

Once you make the above changes, you will see the following output

Output:-

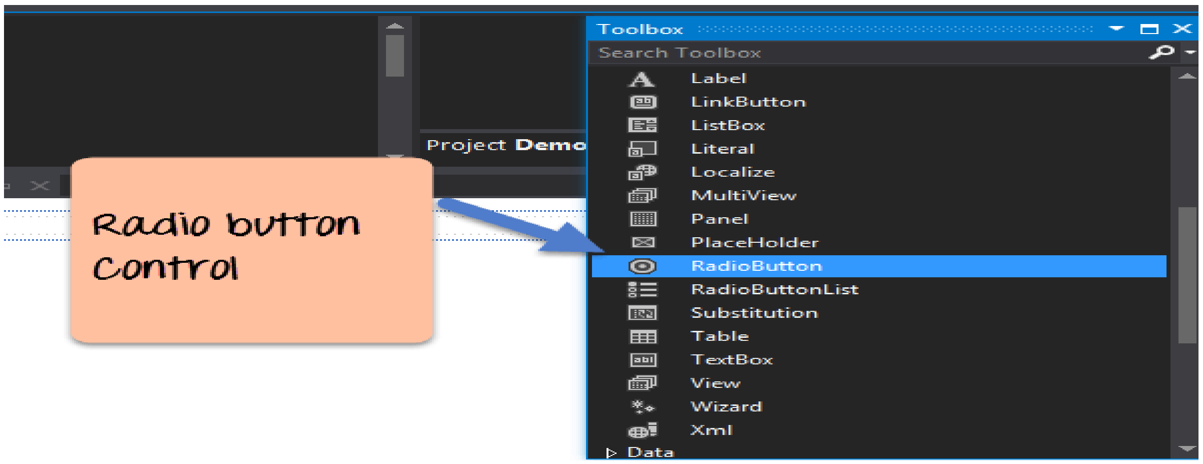


From the output, you can clearly see that the Listboxes was added to the form.

RadioButton

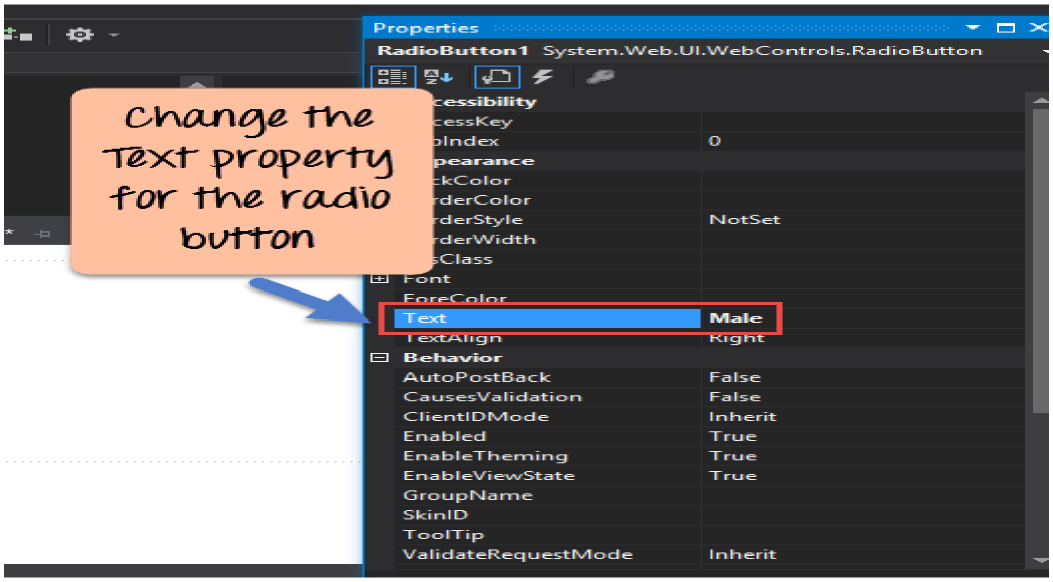
A Radio button is used to showcase a list of items out of which the user can choose one. Let's see how we can implement this with an example shown below. We will add a radio button for a male/female option.

Step 1) The first step is to drag the 'radio button' control onto the Web Form from the toolbox. (see image below). Make sure to add 2 radio buttons, one for the option of 'Male' and the other for 'Female.'



Step 2) Once the Radio button has been added, change the 'text' property.

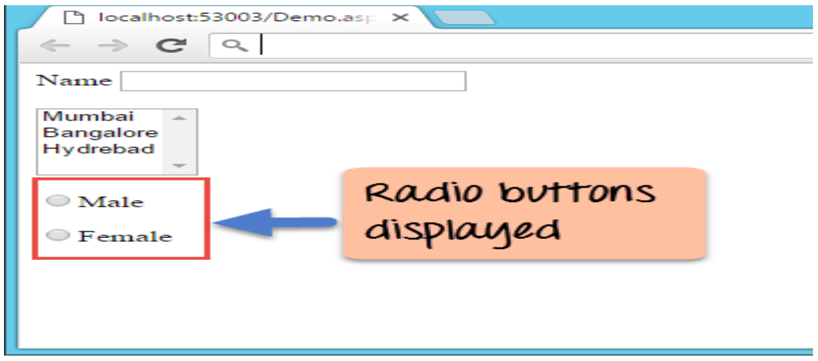
- Go to the properties window by clicking on the 'Radio button control'.
- Change the text property of the Radio button to 'Male'.
- Repeat the same step to change it to 'Female.'
- Also, change the ID properties of the respective controls to rdMale and rdFemale.



```
Properties
RadioButton1  System.Web.UI.WebControls.RadioButton
  cessibility
  cessKey
  bIndex              0
  pearance
  kColor
  rderColor
  rderStyle           NotSet
  rderWidth
  sClass
⊞ Font
  ForeColor
  Text                Male
  TextAlign           Right
⊟ Behavior
  AutoPostBack        False
  CausesValidation    False
  ClientIDMode        Inherit
  Enabled             True
  EnableTheming       True
  EnableViewState     True
  GroupName
  SkinID
  ToolTip
  ValidateRequestMode Inherit
```

Once you make the above changes, you will see the following output

Output:-

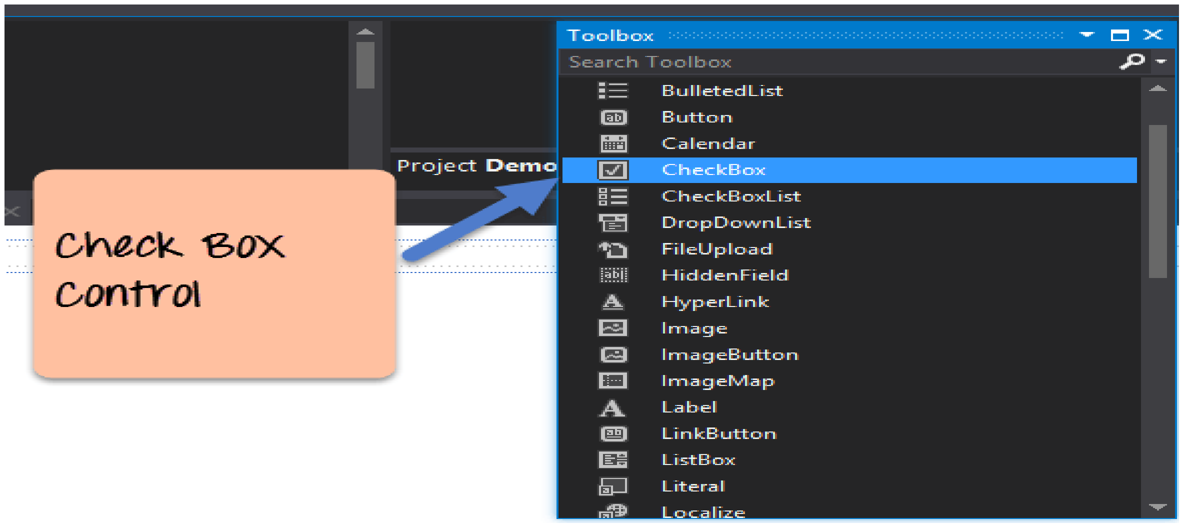


From the output, you can clearly see that the radio button was added to the form

Checkbox

A checkbox is used to provide a list of options in which the user can choose multiple choices. Let's see how we can implement this with an example shown below. We will add 2 checkboxes to our Web forms. These checkboxes will provide an option to the user on whether they want to learn C# or ASP.Net.

Step 1) The first step is to drag the checkbox control onto the Web Form from the toolbox as shown below

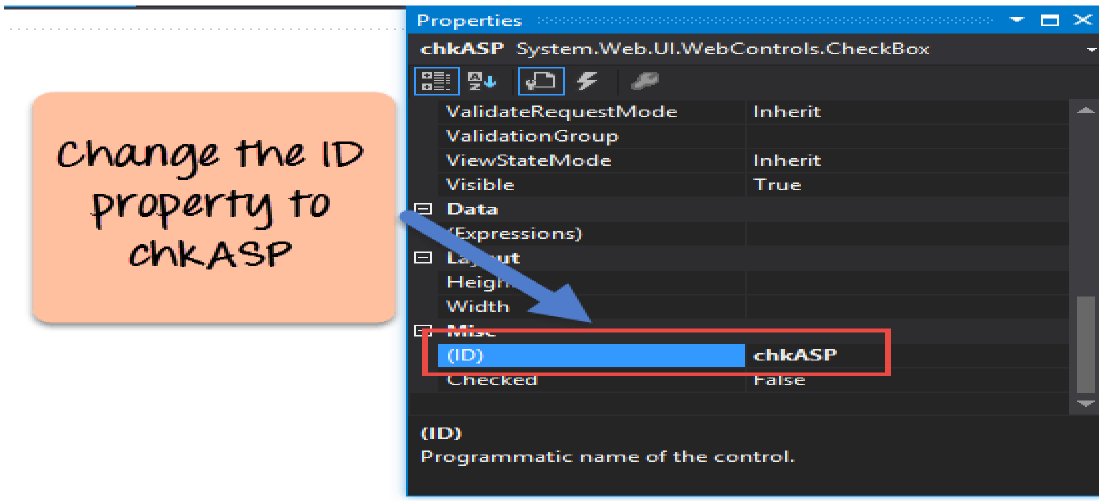


Step 2) Once the Checkboxes have been added, change the checkbox id property to 'chkASP'.

- Go to the properties window by clicking on the Checkbox control.
- Change the ID properties of the respective controls to 'chkC' and 'chkASP'.

Also, change the text property of the Checkbox control to 'C#'. Do the same for the other Checkbox control and change it to 'ASP.Net'.

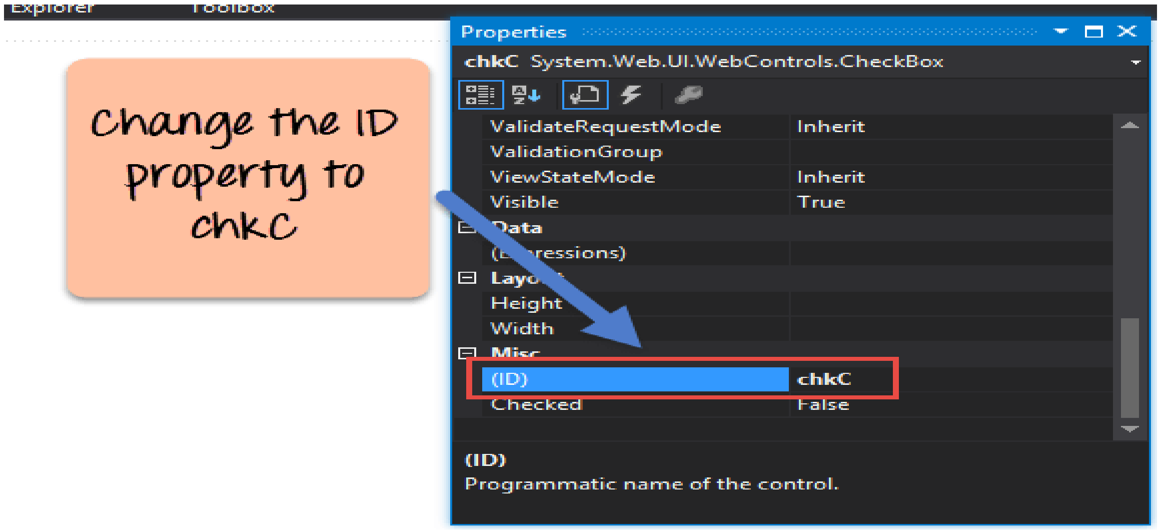
1. Change the ID property of the checkbox to 'chkASP'


Change the ID property to chkASP
Properties
chkASP System.Web.UI.WebControls.CheckBox

ValidateRequestMode	Inherit
ValidationGroup	
ViewStateMode	Inherit
Visible	True
Data	
(Expressions)	
Layout	
Height	
Width	
Misc	
(ID)	chkASP
Checked	False

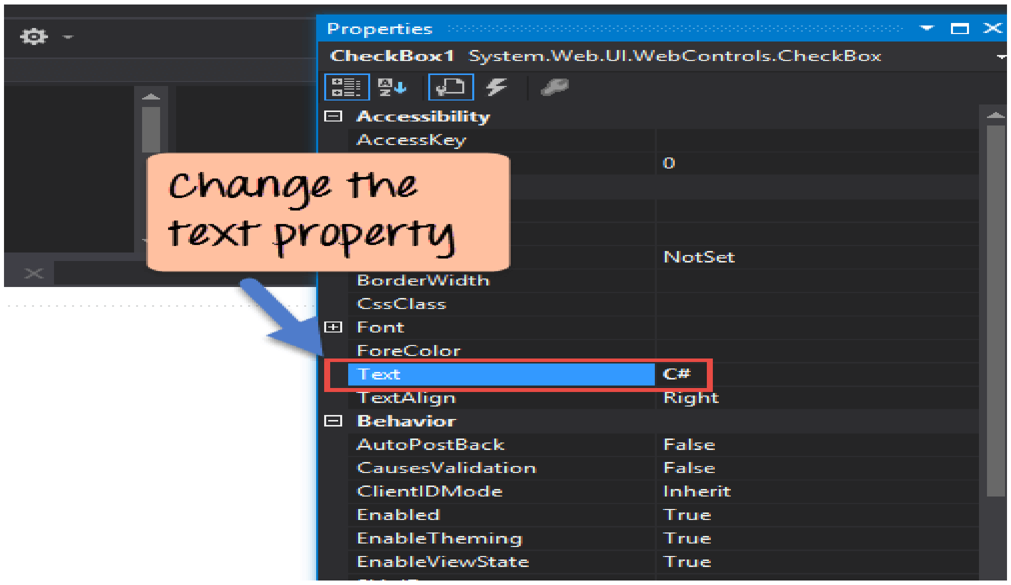
(ID)
Programmatic name of the control.

2. Change the ID property of the checkbox to chkC


Change the ID property to chkC
Properties
chkC System.Web.UI.WebControls.CheckBox

ValidateRequestMode	Inherit
ValidationGroup	
ViewStateMode	Inherit
Visible	True
Data	
(Expressions)	
Layout	
Height	
Width	
Misc	
(ID)	chkC
Checked	False

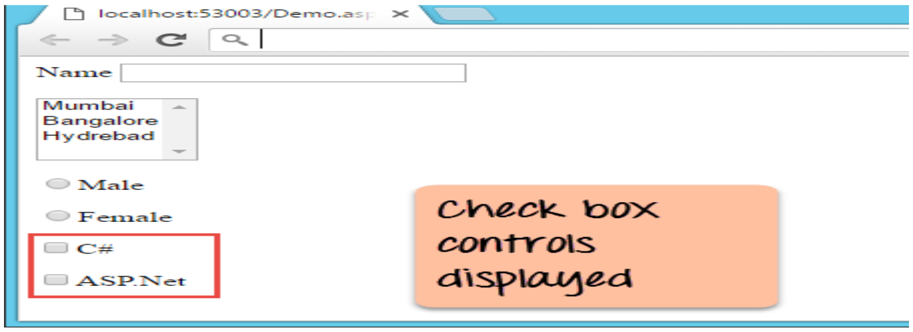
(ID)
Programmatic name of the control.



Accessibility	
AccessKey	
	0
	NotSet
BorderWidth	
CssClass	
Font	
ForeColor	
Text	C#
TextAlign	Right
Behavior	
AutoPostBack	False
CausesValidation	False
ClientIDMode	Inherit
Enabled	True
EnableTheming	True
EnableViewState	True

Once you make the above changes, you will see the following output

Output:-

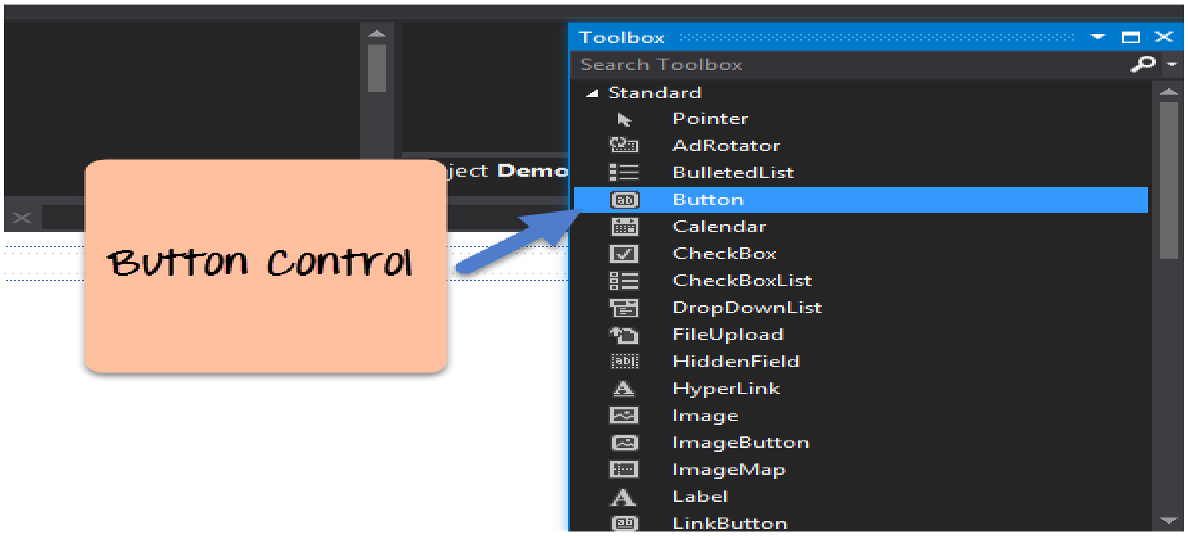


From the output, you can clearly see that the Checkboxes was added to the form.

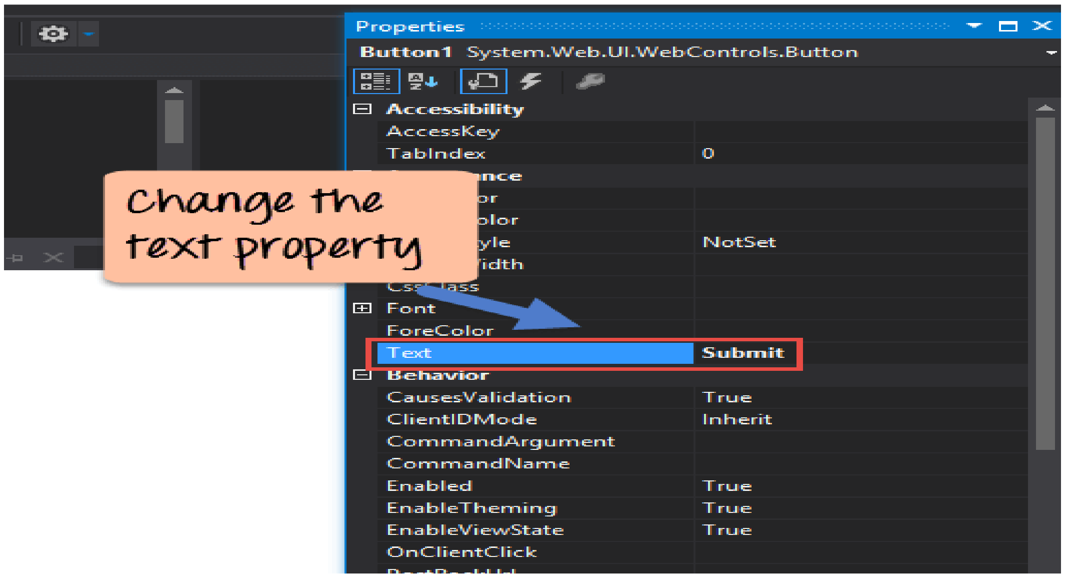
Button

A button is used to allow the user to click on a button which would then start the processing of the form. Let's see how we can implement this with our current example as shown below. We will add a simple button called 'Submit' button. This will be used to submit all the information on the form.

Step 1) The first step is to drag the button control onto the Web Form from the toolbox as shown below



Step 2) Once the button has been added, go to the properties window by clicking on the button control. Change the text property of the button control to Submit. Also, change the ID property of the button to 'btnSubmit'.

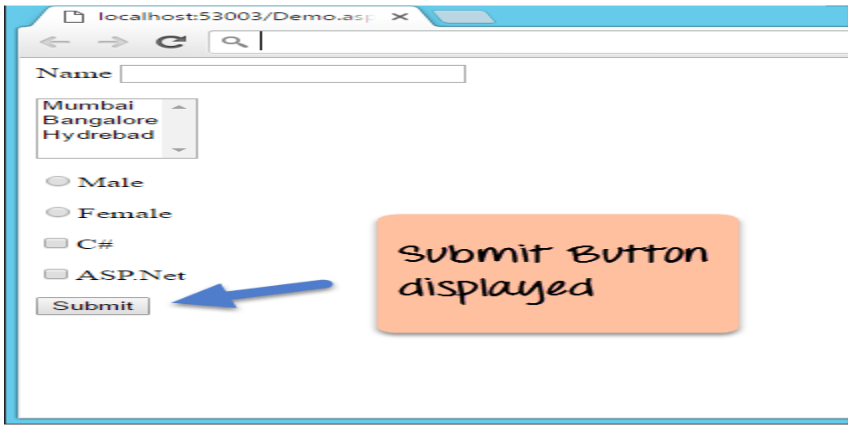


Button1 System.Web.UI.WebControls.Button

⊟ Accessibility	
AccessKey	
TabIndex	0
nce	
or	
olor	
yle	NotSet
idth	
Css lass	
⊞ Font	
ForeColor	
Text	Submit
⊟ Behavior	
CausesValidation	True
ClientIDMode	Inherit
CommandArgument	
CommandName	
Enabled	True
EnableTheming	True
EnableViewState	True
OnClientClick	

Once you make the above changes, you will see the following output

Output:-



ASP.NET Web Forms Tutorial: User Controls Examples

In ASP.Net, it is possible to create re-usable code. The re-usable code can be used in many places without having the need to write the code again.

The re-usable code helps in reducing the amount of time spent by the developer after writing the code. It can be done once and reused at multiple places.

Create User Control in ASP.Net

ASP.Net has the ability to create Web controls. These controls contain code which can be re-used. It can be used across application as per the requirement.

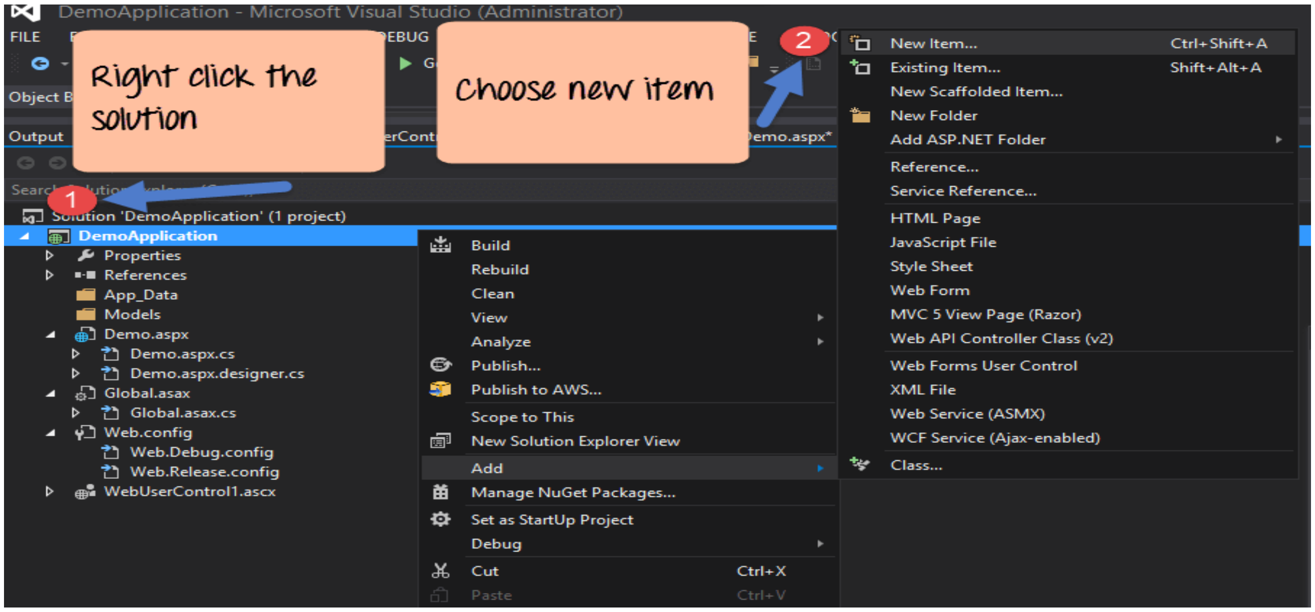
Let’s take a look at an example of how we can create a web user control in ASP.Net

In our example,

- We are going to create a web control.
- It will be used to create a header component.
- It will contain the below mentioned text.”Guru99 Tutorials”This Tutorial is for ASP.Net”

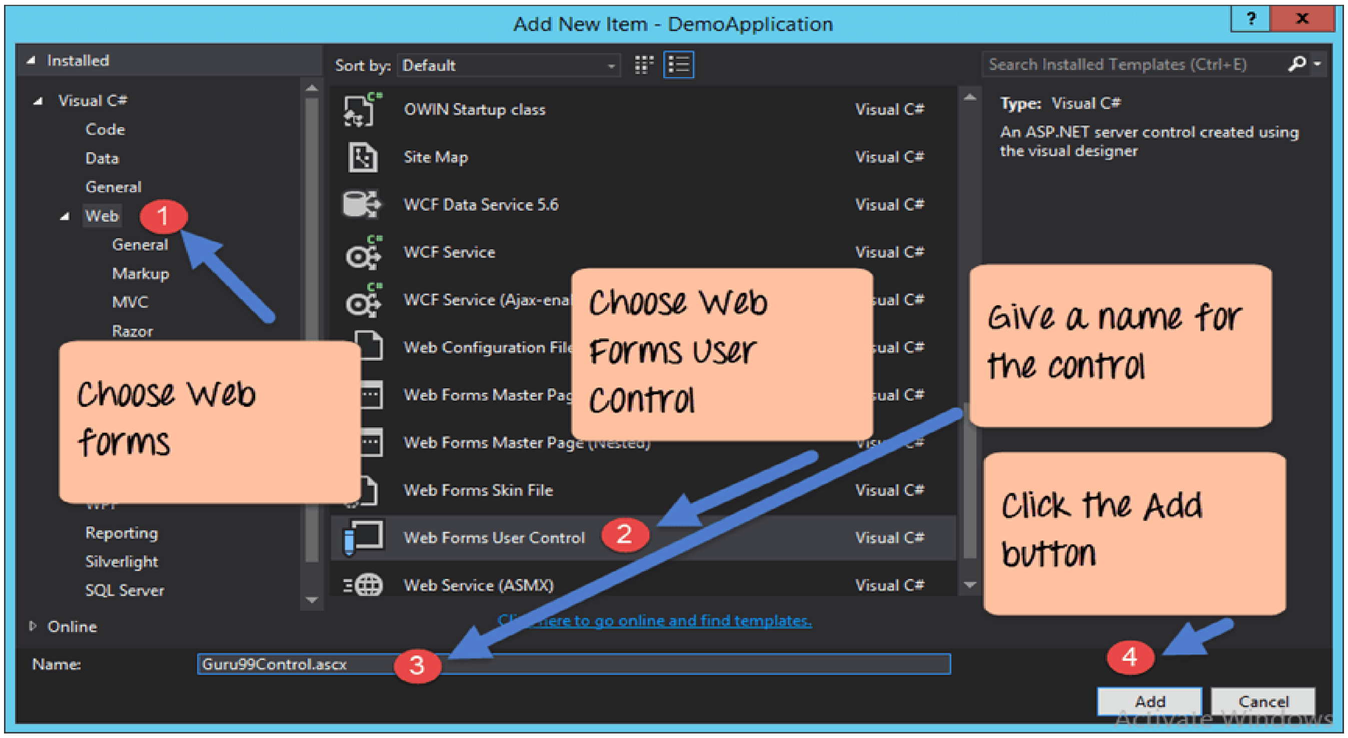
Let’s work with our current web application created in the earlier sections. Let’s follow the below steps to create a Web user control.

Step 1) The first step is to create a web user control and add it to our Visual Studio Solution.



1. Go to the Solution Explorer in Visual Studio and right click the DemoApplication Solution
2. Choose the menu item Add->New Item

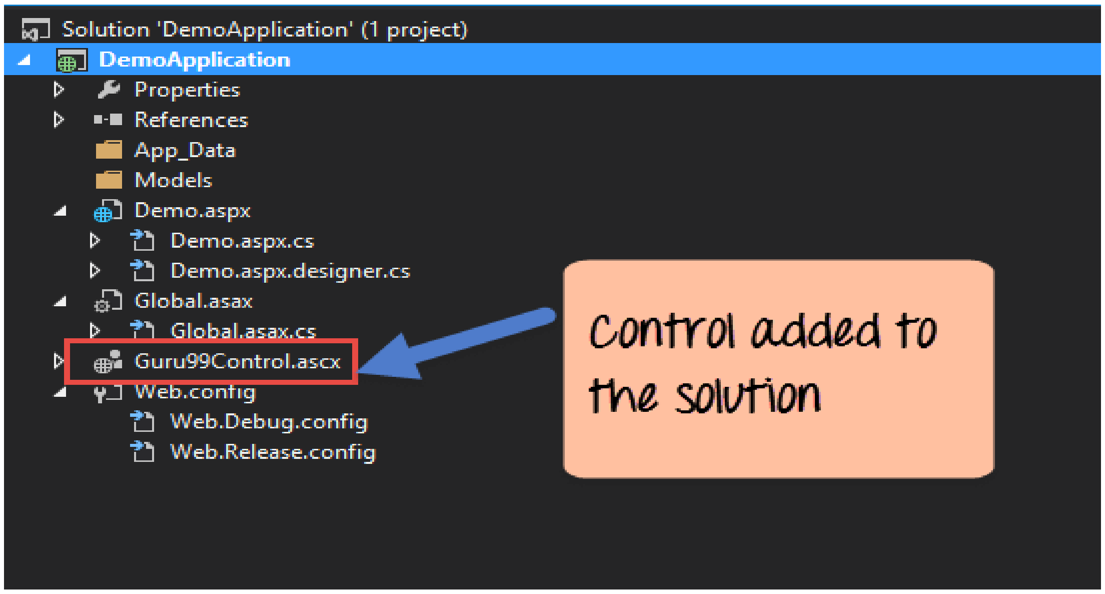
Step 2) In the next step, we need to choose the option of creating a web user control



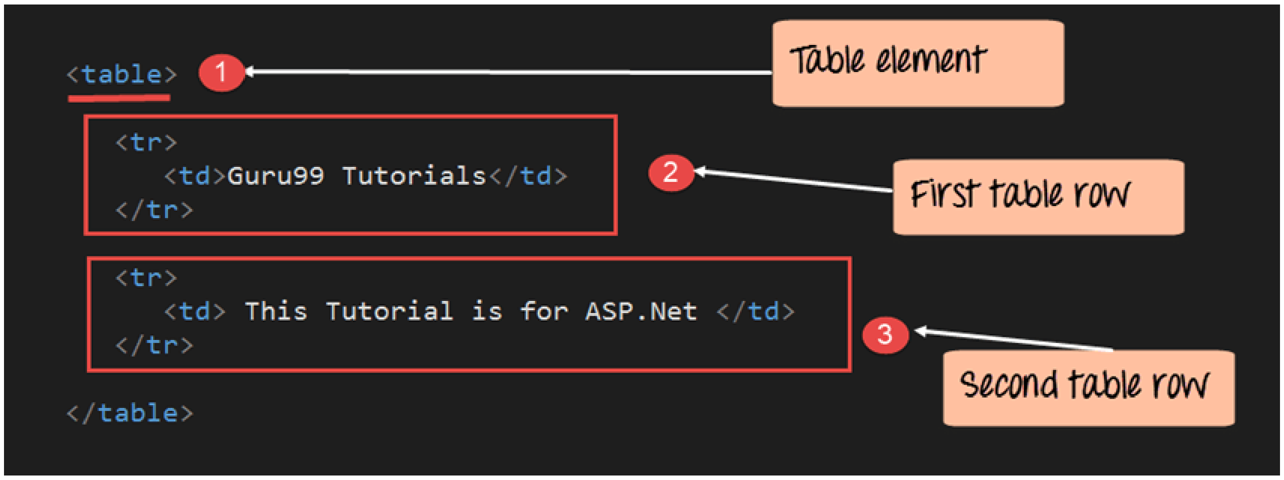
1. In the project dialog box, we can see various options for creating different types of components. Click the "Web" option on the left-hand side.
2. When we click the "Web" option, you see an option for "Web Forms User control." Click this option.
3. We then give a name for the Web Control "Guru99Control".

4. Finally, click the 'Add' button to let Visual Studio add the web user control to our solution.

You will the see the "Guru99Control" added to the solution.



Step 4) Now it's time to add the custom code to the Web user control. Our code will be based on pure HTML syntax. Add the following code to the 'Guru99Control.ascx' file



```
<table>
  <tr>
    <td>Guru99 Tutorials</td>
  </tr>

  <tr>
    <td> This Tutorial is for ASP.Net </td>
  </tr>

</table>
```

```
<table>
      <tr>
        <td>Guru99 Tutorials</td>
      </tr>

      <tr>
        <td> This Tutorial is for</td>
      </tr>
</table>
```

Code Explanation:-

1. In our Web Control file, we are first creating a table element. This will be used to hold 2 rows of text which will be used to display

- "Guru99 Tutorials" and
- "This Tutorial is for ASP.Net."

2. Next, we define our first table row and put the text as "Guru99 Tutorials."
3. We then define our second table row and put the text as "This Tutorial is for ASP.Net."

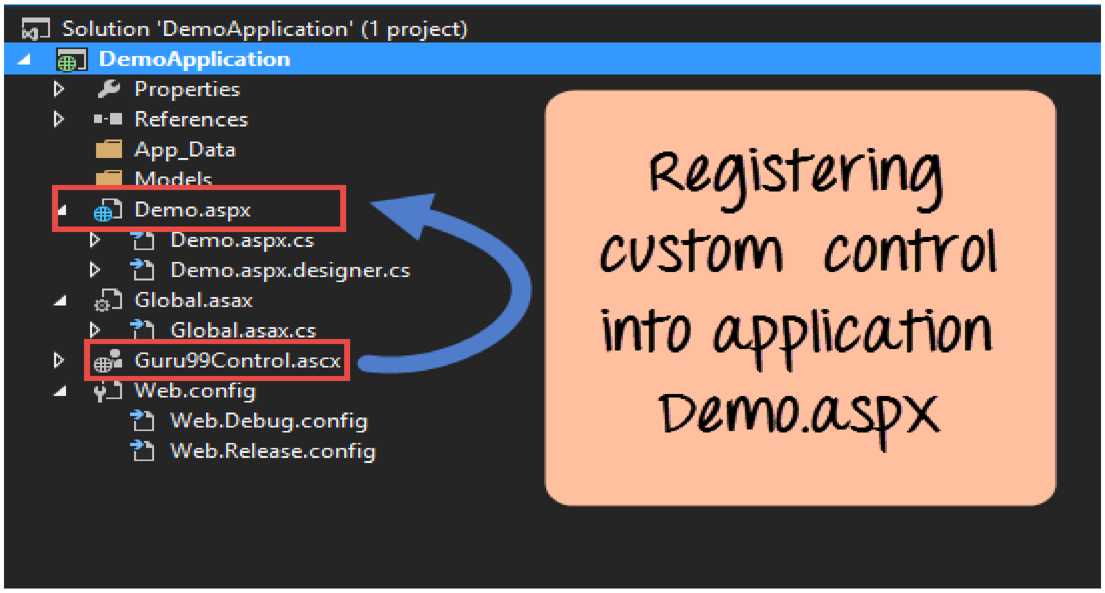
NOTE: Now we cannot execute this code and show the output. The only way to see if this works is to include it in our application (aspx file). We will see this in the sub-sequent topic.

Registering User Controls on a ASP.NET web forms

In the earlier section, we saw how we can create a custom web control. This can be used to display the following two lines in a web form

- "Guru99 Tutorials"
- "This Tutorial is for ASP.Net."

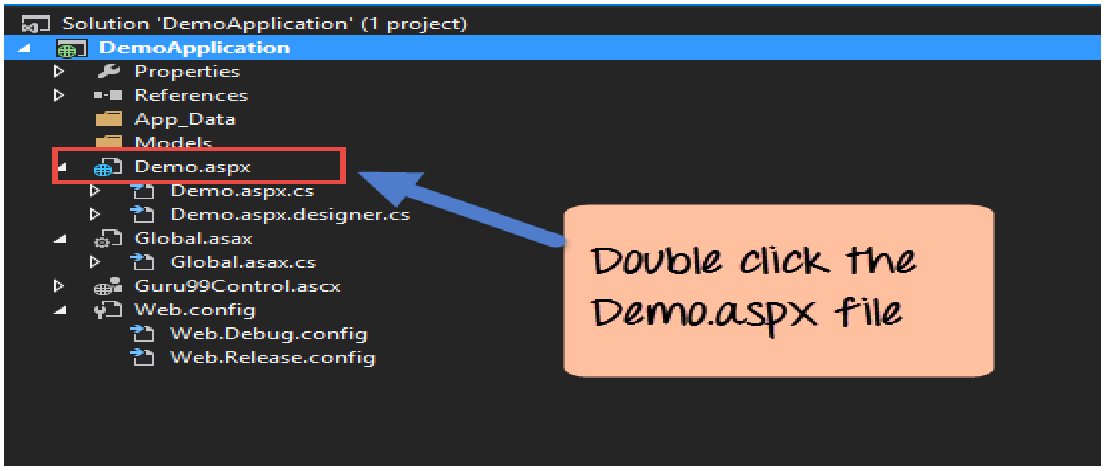
Once the custom 'control' is created, we need to use it in our web application. The first step is to register the component in our application (Demo.aspx). This is the pre-requisite to use in any custom web control in an ASP.Net application.



Let's look at how we can achieve this. The below steps are a continuation to the previous section. In the previous section, we have created our custom control. In this section, we will use the control in our **Demo.aspx** web form.

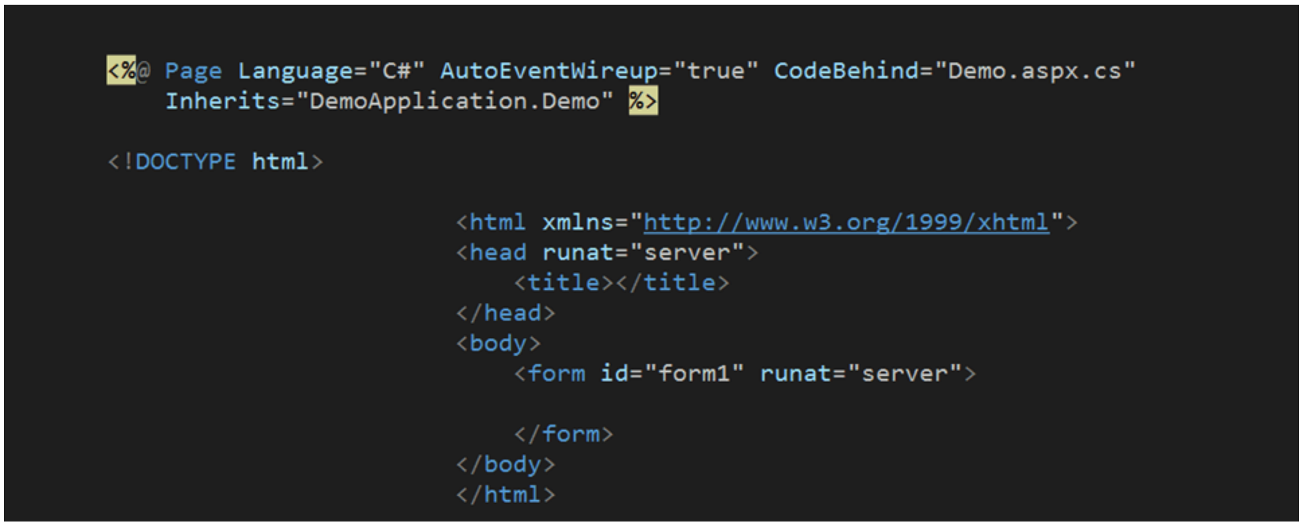
First, we will register our custom 'control' into the Demo.aspx file.

Step 1) Ensure that you are working on the demo.aspx file. It is in this file that the web user control will be registered. This can be done by double-clicking the demo.aspx file in the Solution explorer of your .Net solution.



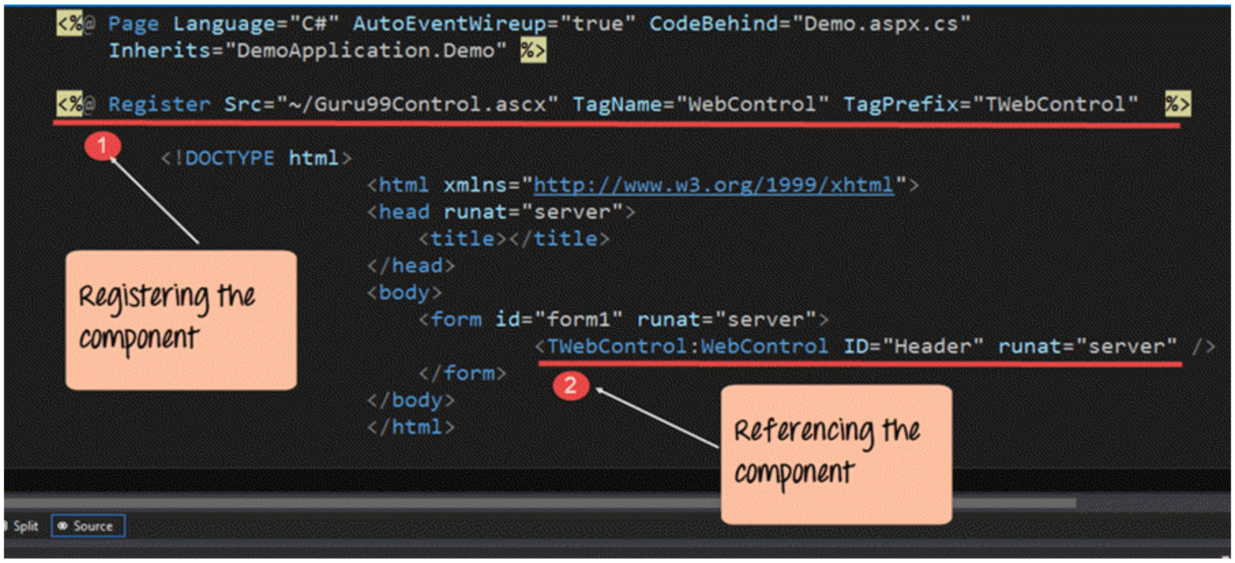
Once you double click the form, you will probably see the below code in the form. This is the default code added by Visual Studio when a web form is added to an ASP.Net project.

The default code consists of steps, which are required to ensure that the form can run as an ASP.Net web form in the browser.



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Demo.aspx.cs"
    Inherits="DemoApplication.Demo" %>

<!DOCTYPE html>

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

                            </form>
                        </body>
                        </html>
```

Step 2) Now let’s add our code to register the user control. The screenshot below shows registration of the user control to the above basic code.



```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Demo.aspx.cs"
    Inherits="DemoApplication.Demo" %>

<%@ Register Src="~/Guru99Control.ascx" TagName="WebControl" TagPrefix="TWebControl"  %>
        <!DOCTYPE html>
                    <html xmlns="http://www.w3.org/1999/xhtml">
                    <head runat="server">
                        <title></title>
                    </head>
                    <body>
                        <form id="form1" runat="server">
                            <TWebControl:WebControl ID="Header" runat="server" />
                        </form>
                    </body>
                    </html>
```

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Demo.aspx.cs" Inherits="DemoApplication.Demo" %>
<%@ Register Src="~/Guru99Control.ascx" TagName="WebControl"
TagPrefix="TWebControl"%>

<!DOCTYPE html>
      <html xmlns="http://www.w3.ore/1999/xhtml">
      <head runat="server">
            <title></title>
      </head>
<body>
      <form id="forml" runat="server”>
            <TWebControl:WebControl ID="Header" runat="server"
/>
      </form>
</body>
</html>
```

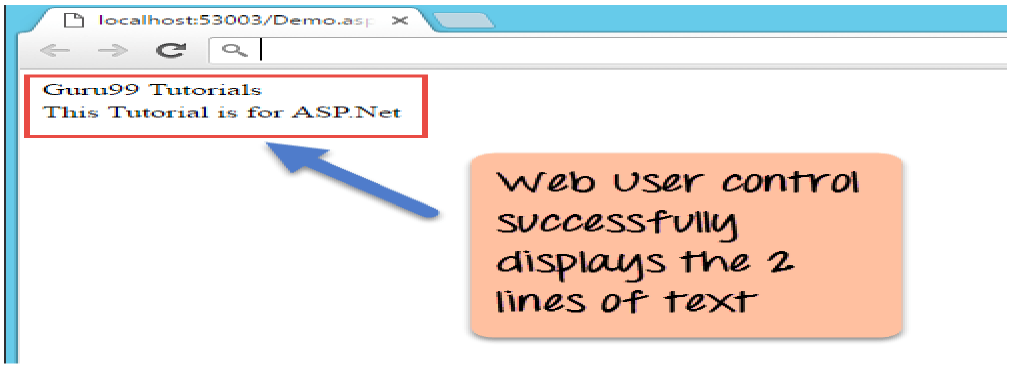
Code Explanation:-

1. The first step is to register the web user control. This comprises of the below basic parameters
1. The ‘Register’ keyword is used to register the web user control.
2. The src parameter is used to define the name of the control, which in our case is Guru99Control.ascx.
3. The tagname and Tagprefix are individual names given to the control. This is done so that they can references in HTML pages as a normal HTML control.
2. Next, we reference our Web user control via the TagPrefix:TagName which was assigned earlier. The TagPrefix:TagName is an indicator that we want to use our custom web control. When the page is processed by the web server, you can see we have used the TWebControl:WebControl tag. It will then process the ‘Guru99Control’ accordingly.In our example, it is TWebControl:WebControl.

1. An optional ID is given to the control of “Header”. It’s generally a good practice to give an ID to an HTML control.
2. Finally, the runat=server attribute so that the control will run on the web server. For all ASP.Net controls, this is the default attribute. All ASP.Net controls (including custom controls) have to be run on the server. Their output is then sent from the server to the client and displayed in the browser accordingly.

When the above code is set, and the project is executed using Visual Studio. You will get the below output.

Output:-



```
Guru99 Tutorials
This Tutorial is for ASP.Net
```

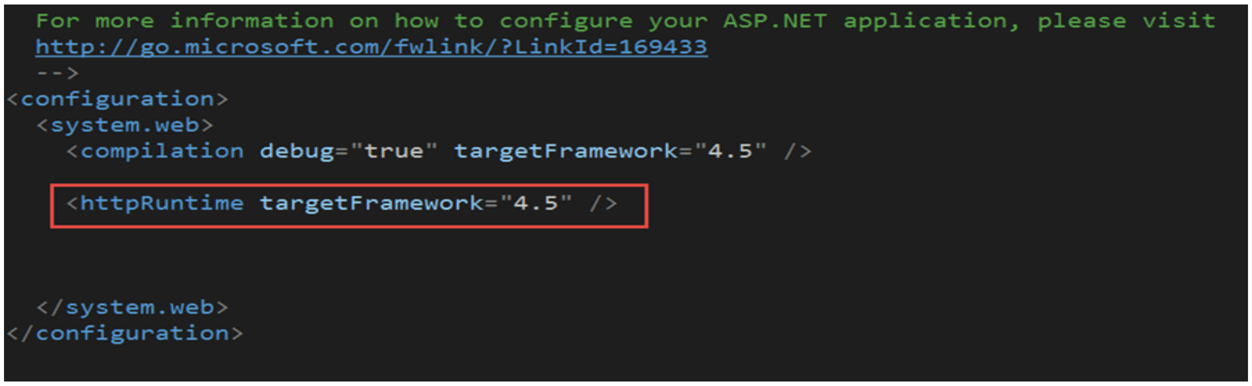
The output message displayed in the browser shows that the web user control was successfully executed.

Registering asp.net controls globally in the web config configuration file asp

Sometimes one might want to use user controls in multiple pages in a .Net application. At this point, you don’t want to keep on registering user controls on each and every ASP.Net page.

- In .Net you can carry out the registration in the ‘web.config’ file.
- The web.config file is a common configuration file used by all web pages in .Net project.
- It contains necessary configuration details for the ASP.Net web project. For example, one common configuration in the web.config file is the **target framework parameter**.
- This parameter is used to identify the .Net framework version used by the application.

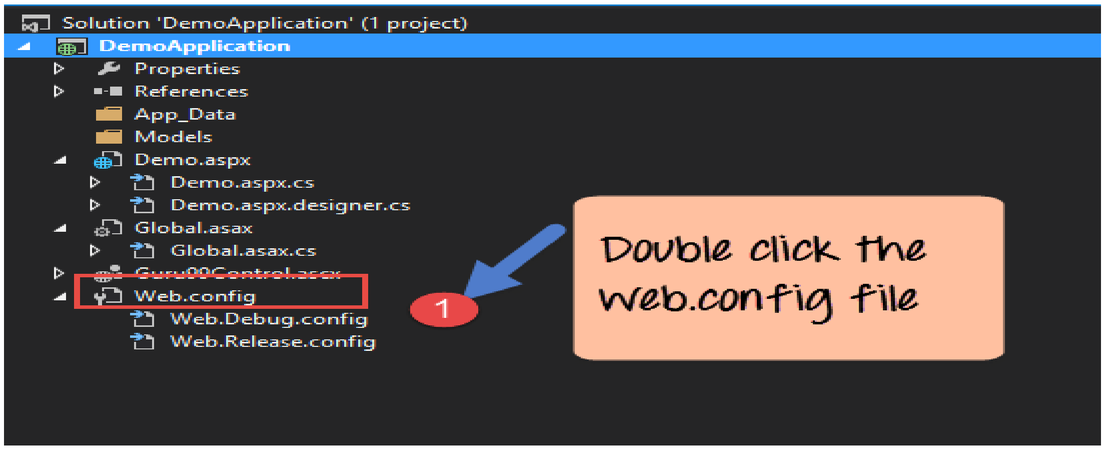
Below is a snapshot of the default code in the web.config file. The highlighted part is the target framework part.



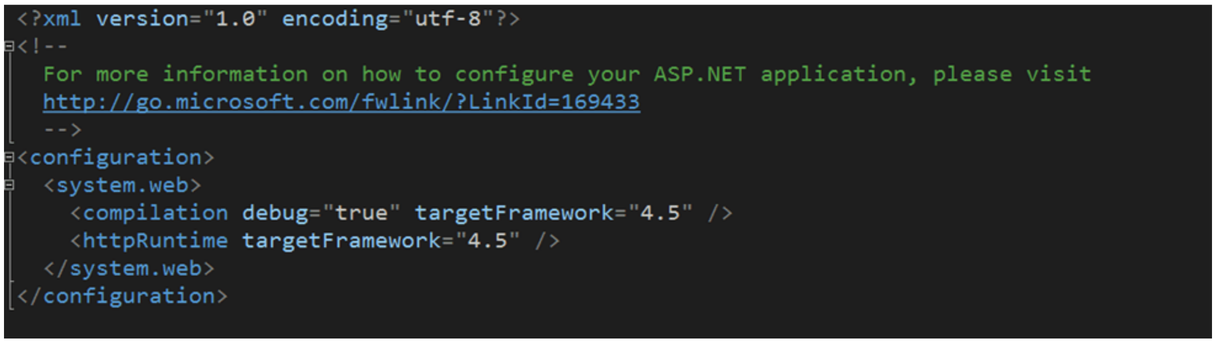
```
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.5" />

    <httpRuntime targetFramework="4.5" />


  </system.web>
</configuration>
```

Let’s see how we can register our Guru99Control in the web.config file.

Step 1) Open the web.config file from solution explorer by double-clicking the file.

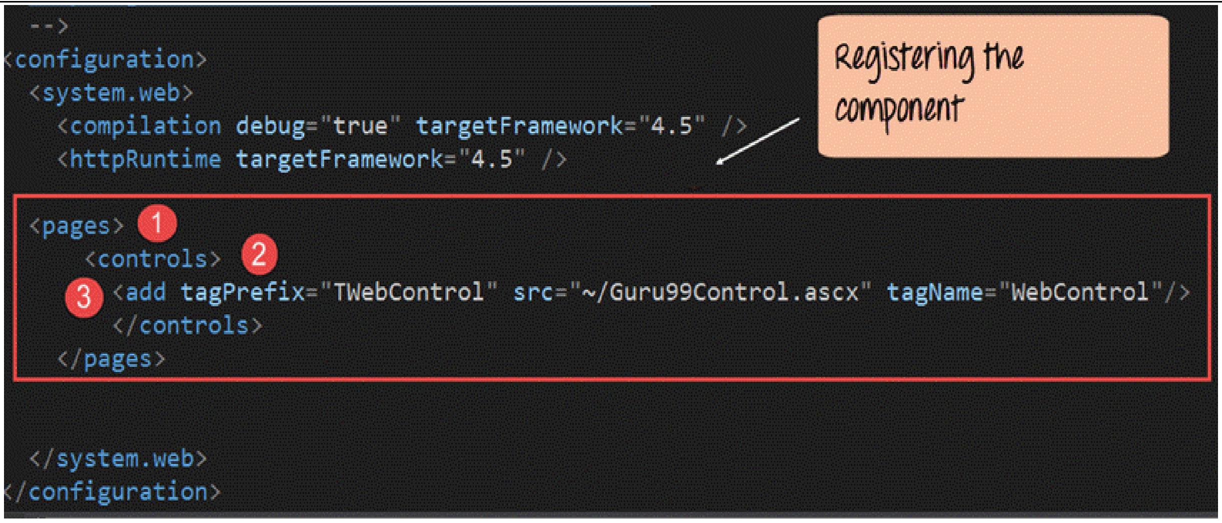


When you open the web.config file, you might see the below configuration. The ‘web.config’ is added automatically by Visual Studio when the project is created. This is the basic configuration required to make the ASP.Net project work properly.



```
<?xml version="1.0" encoding="utf-8"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5" />
  </system.web>
</configuration>
```

Step 2) Now let’s register our component in the web.config file. We need to add the below lines for that.



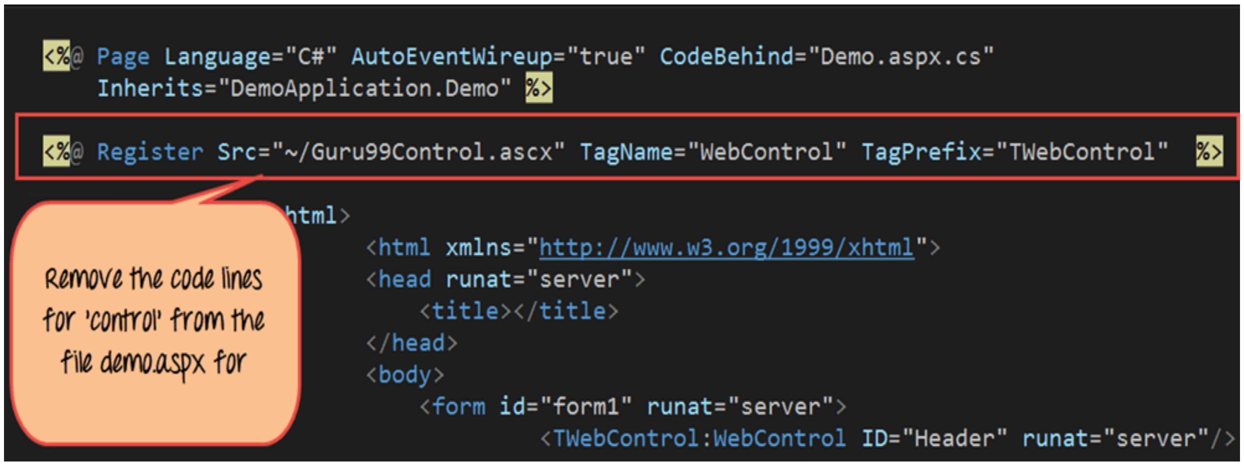
```
  -->
<configuration>
  <system.web>
    <compilation debug="true" targetFramework="4.5" />
    <httpRuntime targetFramework="4.5" />

  <pages> 1
      <controls> 2
   3 <add tagPrefix="TWebControl" src="~/Guru99Control.ascx" tagName="WebControl"/>
      </controls>
    </pages>


  </system.web>
</configuration>
```

```
<configuration>
      <system.web>
       <compilation debug="true" targetFramework="4.5" />
      <pages>
             <controls>
              <add tagPrefix="TWebControl" src
="~/Guru99Control.ascx" tagName="WebControl"/>
             </controls>
      </pages>
      </system.web>
</configuration>
```

The registration comprises of the below substeps

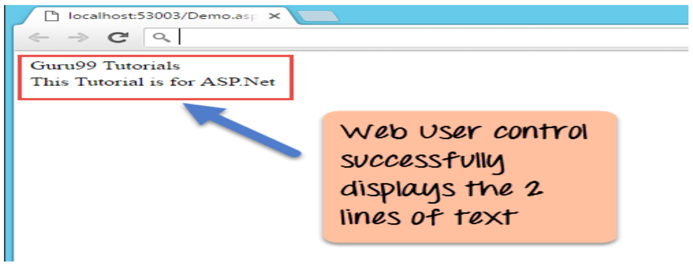
1. Add a tag called <pages>. It means all the configuration for the controls will be applicable to all the ASP.Net pages in the solution.
2. The <controls> tag means that you are adding a configuration for the user control.
3. Then we register the user control with the additional tag. The remaining parameters of tagPrefix, tagName and src remain the same as before.

Step 3) Remember to go the 'demo.aspx' page and remove the lines for control, which had the registration of the Guru99 component. If you don't perform this step, then the 'Guru99Control.ascx' a file will be executed from the 'demo.aspx' file instead of 'web.config' file.


```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Demo.aspx.cs"
    Inherits="DemoApplication.Demo" %>

<%@ Register Src="~/Guru99Control.ascx" TagName="WebControl" TagPrefix="TWebControl"  %>

                   html>
                       <html xmlns="http://www.w3.org/1999/xhtml">
                       <head runat="server">
                           <title></title>
                       </head>
                       <body>
                           <form id="form1" runat="server">
                                    <TWebControl:WebControl ID="Header" runat="server"/>
```
Remove the code lines for 'control' from the file demo.aspx for

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Demo.aspx.cs" Inherits="DemoApplication.Demo" %>
<%@ Register Src="~/Guru99Control.ascx" TagName="WebControl"
TagPrefix="TWebControl"%>

<!DOCTYPE html>
<html xmlns="http://www.w3.ore/1999/xhtml">
      <head runat="server">
        <title></title>
      </head>
<body>
      <form id="form1" runat="server”>
        <TWebControl:WebControl ID="Header" runat="server" />
      </form>
</body>
</html>
```
The above code is set, and the project is executed using Visual Studio. You will get the below output.

Output:-

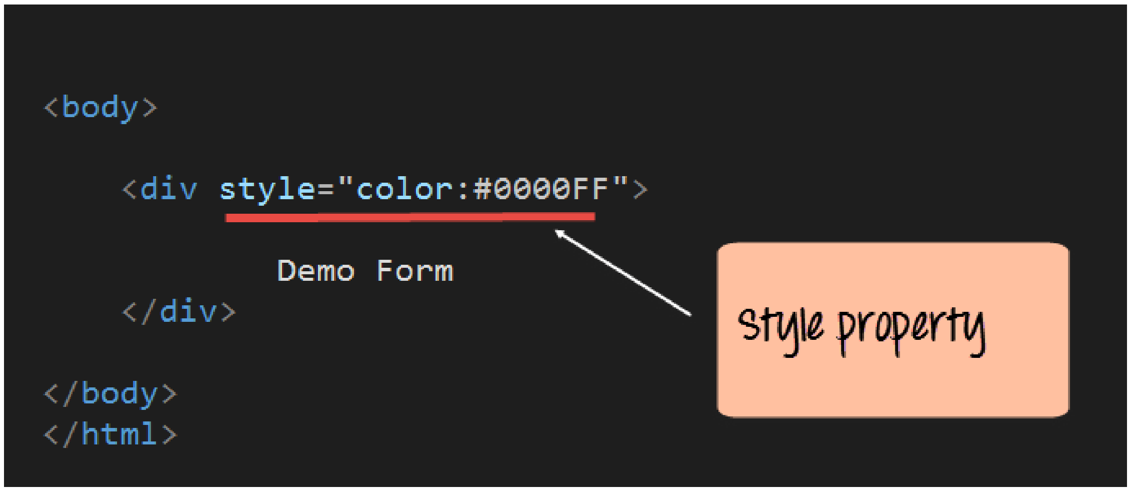


```
Guru99 Tutorials
This Tutorial is for ASP.Net
```

The output message shows that the web user control was successfully executed.

Adding public properties to a web control

A property is a key-value pair associated with any control. Let’s take an example of the simple <div> HTML tag. A screenshot of how the tag looks like is shown below.



```
<body>

    <div style="color:#0000FF">

            Demo Form
    </div>

</body>
</html>
```

```
<html>
<body>
      <div style="color:#0000FF">

            Demo Form

      </div>

<body>
</html>
```

The ‘div’ tag is used to create a section in an HTML document. The ‘div’ tag has a property called a style property. This can be used to give a different style to the text displayed in the div tag. Normally you would see the code for the div tag as shown below.

<div style="color:#0000FF">
So the color attribute is nothing but a key-value pair which gives more information on the tag itself. In the above case, the key name is ‘style’ and the key value is ‘color:#0000FF’.

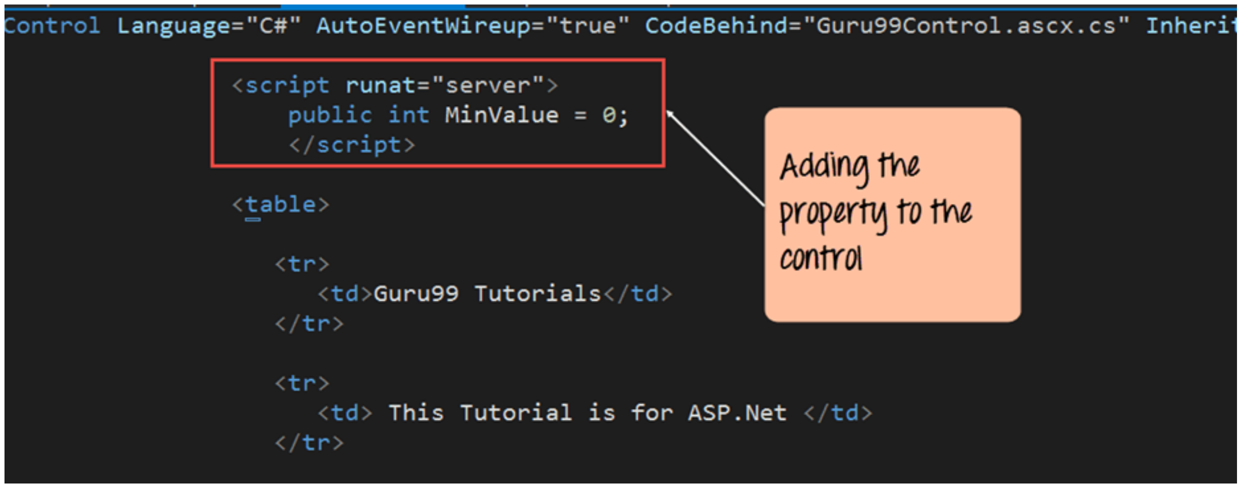
Similarly, for user controls, you can create your own properties that describe the control.

Let’s take a simple example and build upon our ‘Guru99Control’ created in the earlier sections.

In our example, we are going to add a simple integer property called MinValue. This value would represent the minimum number of characters in the text displayed in the user control.

Let’s carry out the below-mentioned steps to get this in place.

Step 1) Open the Guru99Control.ascx file. Add the code for adding the MinValue property.



```
Control Language="C#" AutoEventWireup="true" CodeBehind="Guru99Control.ascx.cs" Inheri
            <script runat="server">
                public int MinValue = 0;
                </script>

            <table>

                <tr>
                    <td>Guru99 Tutorials</td>
                </tr>

                <tr>
                    <td> This Tutorial is for ASP.Net </td>
                </tr>
```

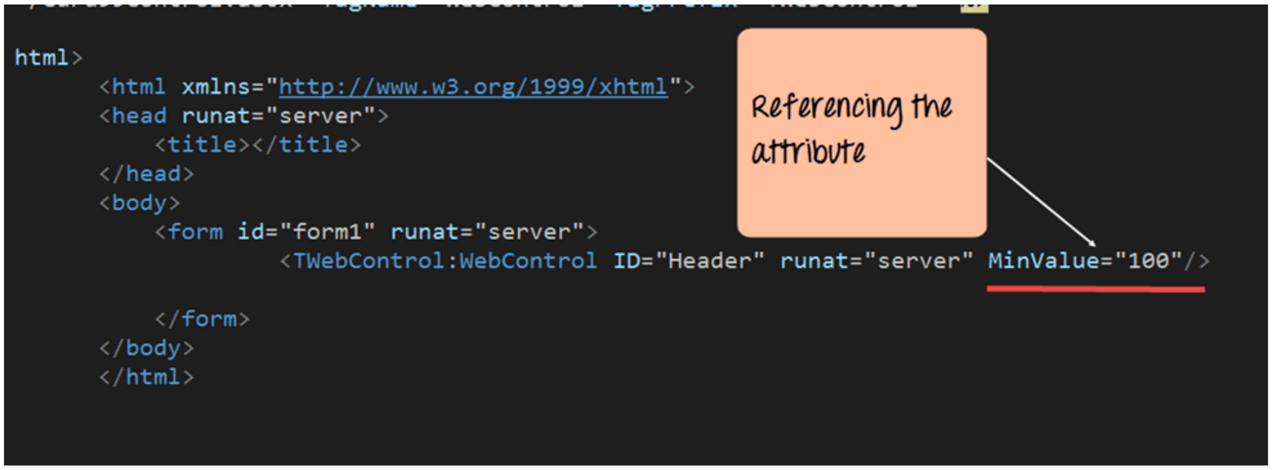
```
<%@ Page Language="C#" AutoEventWireup="true"
CodeBehind="Demo.aspx.cs" Inherits="DemoApplication.Demo" %>

      <script runat="server">
        public int MinValue = 0;
      </script>

<table>
      <tr>
        <td>Guru99 Tutorials</td>
      </tr>

      <tr>
        <td> This Tutorial is for
      </tr>
</table>
```

Code Explanation:-

The script runat=server attribute is used to indicate that we are adding some.Net specific code and that it needs to be run on the web server.

This is required for processing any property added to the user control. We then add our property MinValue and give it a default value of 0.

Step 2) Now let’s reference this property in our demo.aspx file. All we are doing now is just referencing the MinValue property and assigning a new value of 100.



```
html>
      <html xmlns="http://www.w3.org/1999/xhtml">
      <head runat="server">
         <title></title>
      </head>
      <body>
         <form id="form1" runat="server">
                  <TWebControl:WebControl ID="Header" runat="server" MinValue="100"/>

         </form>
      </body>
      </html>
```

```
!DOCTYPE html>
<html xmlns="http://www.w3.ore/1999/xhtml">
      <head runat="server">
        <title></title>
      </head>
<body>
      <form id="form1" runat="server”>
        <TWebControl:WebControl ID="Header" runat="server"
MinValue="100"/>
      </form>
</body>
</html>
```

NOTE: – When you run this code, it will not show any output. This is because the output falls under 100 character limit.

Types of Validation Controls in ASP.NET!

1. Introduction

ASP.NET is a popular, open-source app development framework that enables developers to create dynamic web pages and design robust websites. This framework comes with various validation controls that enable the developers to set properties that can validate the user data to ensure that the entered data satisfies the condition. Some of the popular ASP.NET validators are RequiredFieldValidator, CompareValidator, RangeValidator, RegularExpressionValidator, ValidationSummary, and CustomValidator. These validations make business websites look more professional and also enable valid users to get access. This is the only reason any business owner would prefer to hire a .NET developer from a top .NET software development company. To know more about the types of validation controls of ASP.NET and see how developers can use them for web development, let’s go through this blog.

2. Why Do We Use Validation Controls?

Validation controls are generally used to validate the user’s input data as per the specified range before it is sent to different layers of the web application. And this is why they are known as the most essential part of web apps. The top reasons for using validation controls are-

- To validate user input value.
- Helps in implementing presentation logic.
- It enables the use of data format, data range, and data type for validation.

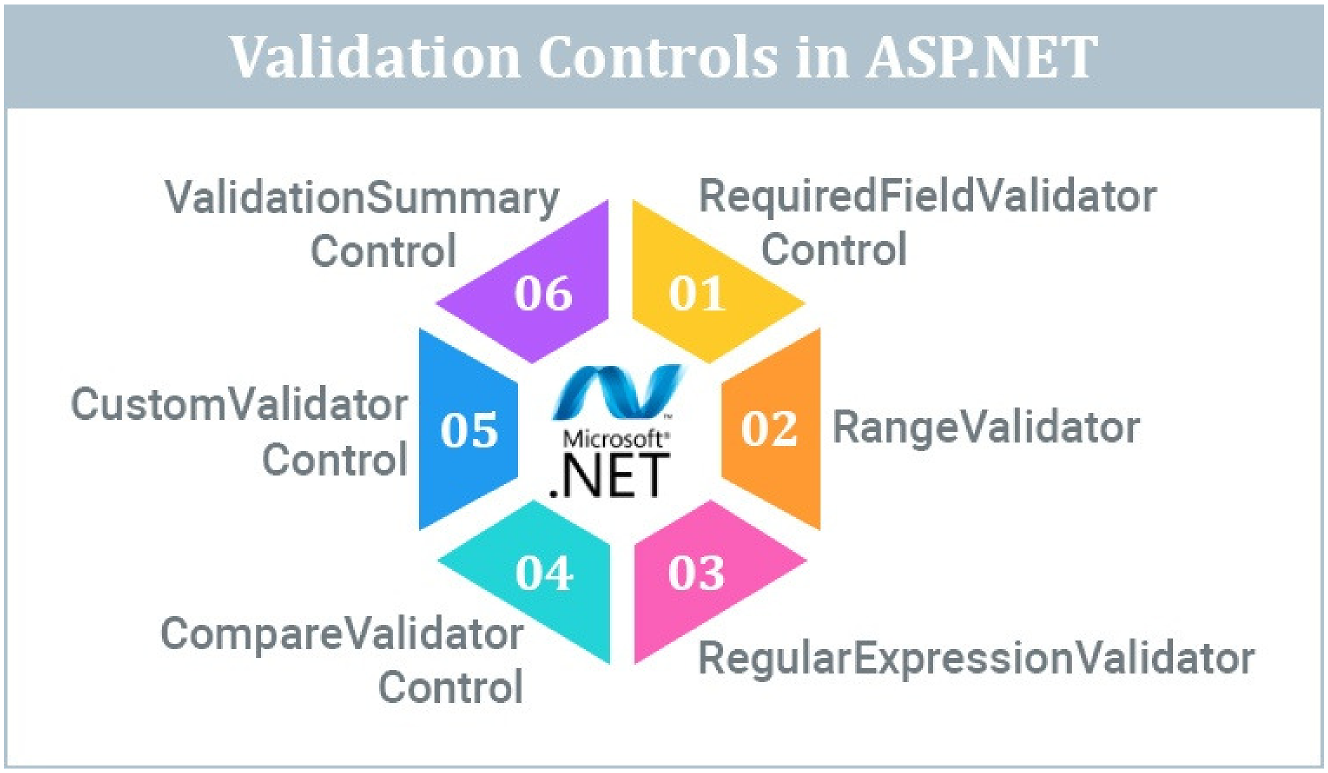
3. How Do They Work?

When it comes to understanding the working of ASP.NET Validation Controls, having knowledge of the class that inherits all the controls is necessary. This class is called 'The BaseValidator Class'. All the validation controls in .NET inherit the methods and properties of the BaseValidator class. This enables the developers to make a generic suite of validation controls.

BaseValidator class

- ControlToValidate – This is a property that indicates the input control to validate. Basically, the input value throughout the form must be unique. And this is why it is a mandatory attribute as it helps in associating the input control with a data validation control.
- ErrorMessage – This property holds the message that is to be displayed in the event when the validation fails.
- Text – The value in this property is going to be displayed when ValidationSummary control is used or there is a missing Text property.
- Enabled – It is a property that enables or disables the validator.
- Validate() – It helps in revalidating the control and updating the IsValid.
- IsValid – It is an attribute that specifies whether the input control is valid or not.

4. Validation Controls in ASP.NET


Validation Controls in ASP.NET
ValidationSummary Control
06
RequiredFieldValidator Control
01
CustomValidator Control
05
Microsoft .NET
02
RangeValidator
04
03
CompareValidator Control
RegularExpressionValidator

Some of the most popular .NET validation controls that developers use are –

4.1 RequiredFieldValidator Control

RequiredFieldValidator is known as an elementary validation control. There is no form that doesn't consist of fields that are mandatory to be filled. If the users want to proceed with the form, they will have to mandatorily fill these fields. To ensure that such fields are not left empty, RequiredFieldValidator is used. Basically, this validation checks that there must be some value-added within the control. Some important properties of RequiredFieldValidator are-

- Initial value:- Initial value is displayed by default to guide the users on how the value must be added. This property is also used by the developers for a drop-down list.
- ControlToValidate:- This control is used to set the field of the text box for validation.
- Text:- The text value is set for the validation under this property.

Syntax of RequiredFieldValidator

```
<asp:requiredfieldvalidator id="UniqueId" runat="server"
controltovalidate="UniqueControlId"
errormessage="ErrorMessageForFailure"
initialvalue="aPlaceholderValue">

</asp:requiredfieldvalidator>
```

4.2 RangeValidator

The RangeValidator is a validation control that is used by the .NET developers to check whether the value of the input control is inside some specific range or not. This type of control is used when it comes to getting inputs like Age, Date of Birth, or mobile numbers from the user on the websites. Some of the major properties of RangeValidator control are-

- Minimum Value:- This property is used by the developers to hold the valid range's minimum value.
- Maximum value:- This property is used to hold a valid range's maximum value.
- ControlToValidate:- ControlToValidate enables the developers to set the specific control that needs to be validated.
- Type:- Here, Type properties are set after the above properties if necessary. RangeValidator can compare the following data types:

1. String
2. Integer
3. Double
4. Date
5. Currency

Syntax of RangeValidator

```
<asp:rangevalidator id="UniqueId" runat="server"
controltovalidate="UniqueControlId"
errormessage="ErrorMessageForFailure" type="Integer"
minimumvalue="”10”" maximumvalue="”500”">

</asp:rangevalidator>
```

4.3 RegularExpressionValidator

RegularExpressionValidator or Regex is a control that holds various patterns which clearly define the format of the text. This means that if the text that has been added is in the same format then the Regex control will return true or else false. This type of validation control is generally used to validate input fields like email, phone, Zip code etc. Some of the most important elements used to make the regular expression in this control are-

- \d:- [0->5] (for value 0 to 5, matches decimal characters)
- \D:- Other than [0->9] (matches non-decimal characters)
- {Length}:-{min ,max}
- \s:- space
- \S:- other than space
- |:- OR
- \w:- [a->z][A->Z][0->9]

- ( ):- a Validation group
- [ ]:- choice of given character

Syntax of RegularExpressionValidator

```
<asp:regularexpressionvalidator id="someUniqueId" runat="server"
controltovalidate="someUniqueControlId"
errormessage="ErrorToDisplayOnValidationFailure"
validationexpression=""”aRegexPattern”"">

</asp:regularexpressionvalidator>
```

4.4 CompareValidator Control

CompareValidator control compares the input control of one field to fixed value or another control. This checks whether the entered value is the same or not. So, if in any case, both entered values are not the same, it will give validation errors to the users. This type of validation control is generally used in change password functionality to confirm new password and date range fields like start date and end date pair. Some of the most specific properties used under this control for different types of comparisons are –

- Equal:- This property of CompareValidator is used by developers to check if the user input data is equal.
- Not Equal:- It is a property that is used to check if the controls are not equal.
- LessThan:- It checks out for less than a relationship.
- LessThanEqual:- It figures out for less than equal relationships.
- Greater than:- This property is used to check for a greater than relationship.
- GreaterThanEqual:- It is a property that is used by the developers to check for the GreaterThanEqual relationship.

Syntax of CompareValidator

```
<asp:comparevalidator id="UniqueId" runat="server"
controltovalidate="UniqueControlId"
errormessage="ErrorMessageForFailure" type="string"
valuetocompare=""”anyFixedValue”"" operator=""”Equal”"">

</asp:comparevalidator>
```

4.5 CustomValidator Control

CustomValidator is a control that is used by the .NET software development companies to customize and implement data validation as per the requirements and conditions that occur. For instance, if the developer wants to check whether the entered number is even or odd, then the existing controls cannot be used and for this, the developers use

CustomValidation controls. Some of the major properties of custom validator controls in .NET are-

ClientValidationFunction:- This property is used to set the name of the function in the custom client-side script that is used for validation. In simple words, this property is used to do client-side validation as well as server-side validation and must be written in a scripting language, such as JavaScript or VBScript.

ValidateEmptyText:- It is a property that is used by the .NET developers to set a Boolean value that indicates whether the empty text is valid or not.

Syntax of CustimValidator

```
<asp:customvalidator id="UniqueId" runat="server"
controltovalidate="UniqueControlId"
errormessage="ErrorMessageForFailure"
clientvalidationfunction="”functionName”">

</asp:customvalidator>
```

4.6 ValidationSummary Control

ValidationSummary is a control that is used to display error messages. It is a control that collects every type of validation control error message and is used by the other validation controls on a web page. It then displays the error message on the screen. Some of the major properties of validation summary control are –

- Forecolor:- This is a property that is used to set the foreground color.
- DisplayMode:- It is a property that is used by the developers to set the display mode of the control.
- HeaderText:- Header text is set at the top of the summary.

Syntax of ValidationSummary

```
<asp:ValidationSummary ID="ValidationSummaryControl"

runat="server" DisplayMode=”BulletList” ShowSummary=”true”

HeaderText=”List of Errors” />
```

5. Validation Groups

In ASP.NET, complex pages come with various different groups of data that are present in different panels. And in this case, a requirement for performing validation might arise which is divided separately into separate groups. This type of situation is managed by the .NET developers with the use of validation groups. For creating a validation group, the developer must put the validation controls and input controls into the same logical group which can be done by setting their ValidationGroup property.