

Chapter -2 ASP.Net Web Forms

2.1 Adding controls to the web page

- **Inline Code and Code-behind file model**

Before learning to add controls to the web page, there are two different ways using which logic code can be used in web page. Inline code & code-behind file. Inline code refers to the design code & logic code to reside in the same page, while code behind file contains design code & logic code in two different file.

Inline Code: Inline code contains logic & design code in the same page. Both codes are in .aspx file. Logic code can be done the same way as in separated file, the only difference is logic code is in <head> section & is in <script> tag.

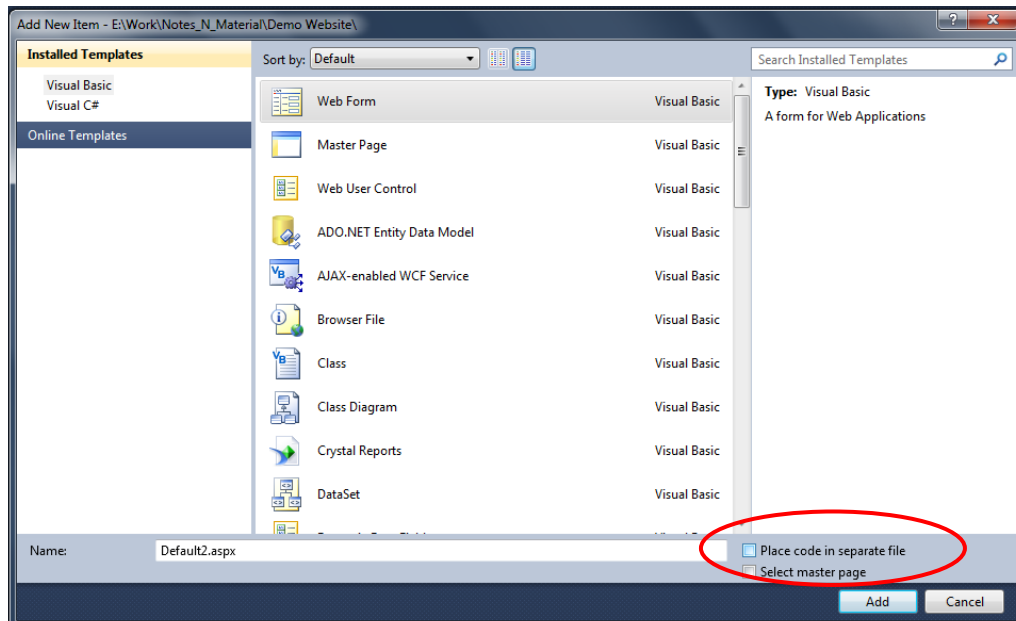


Figure: Not opting for separate files

If this checkbox is not checked it means the visual studio will not create separate files for design & logic code & single file will be created as illustrated below. Hence it is called inline code model.

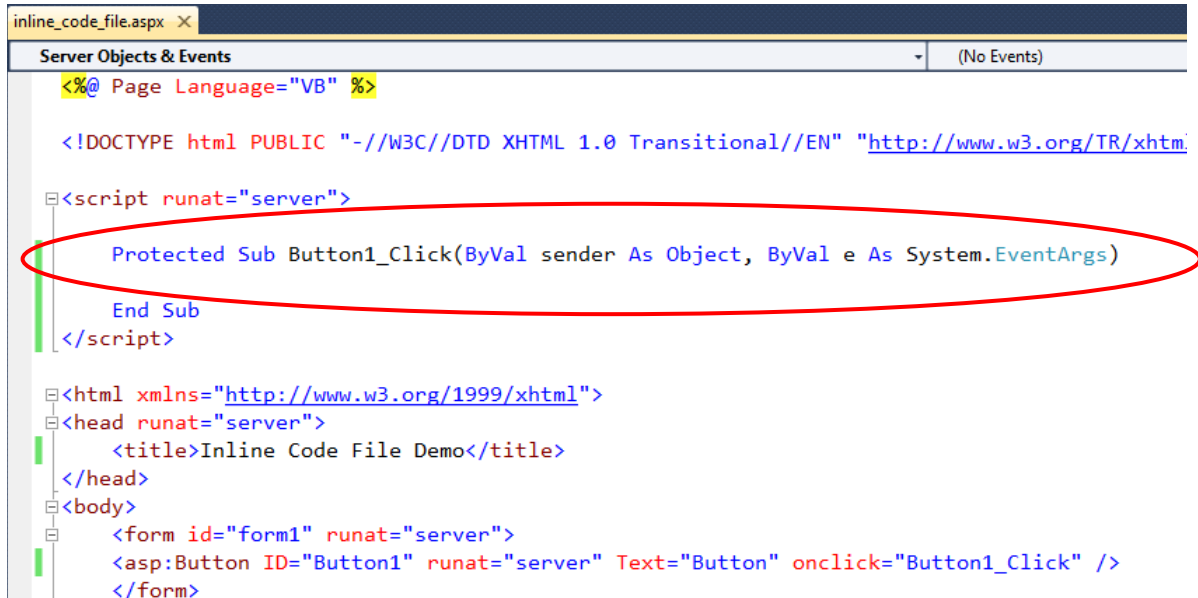


Figure: Inline code model with both codes in same file

As shown in the picture above, we have created a file named as inline_code.aspx which contains both code in the same file. No separate file for logic code is contained. The logic code of click event of button is created in the same page & is available in <script> tag.

Advantages:

- No separate file is needed for the logic code.
- Reduces the size of project/ web application.
- Changes can be done easily in file as it is in same file.
- Renaming files is easy.

Disadvantages:

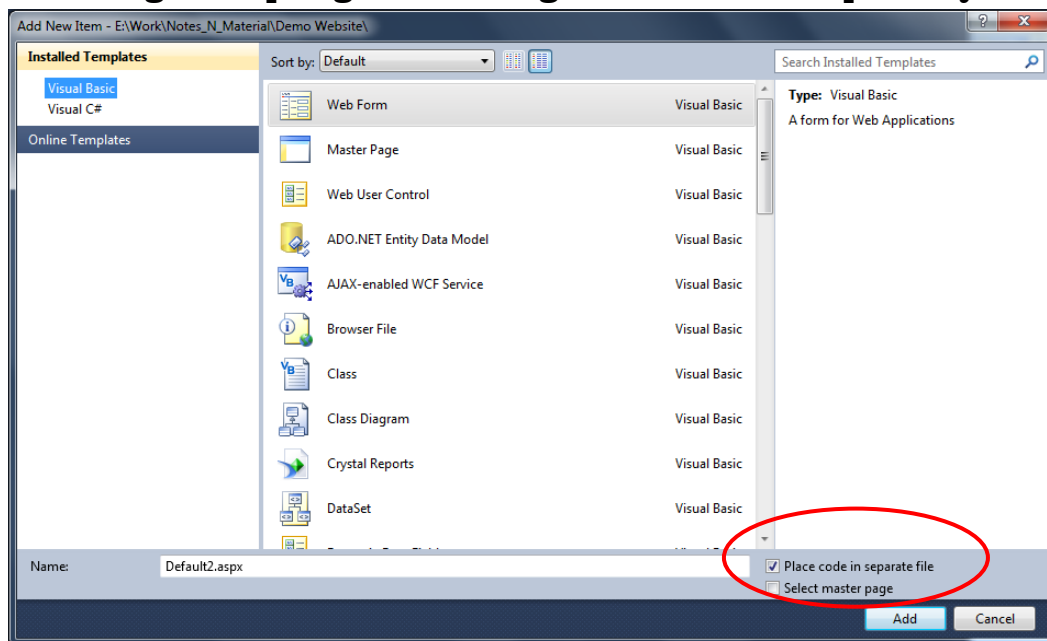
- Code readability is not easy as it may become a long file when it comes to complex logic.
- Logic & design code can be developed at the same time as it is in the same file/web page.

Code-behind file model:

ASP.Net facilitates the developers with a great option of putting logic code & design code in two separate files. This kind of file model is known as code-behind file model. As the name suggests, the core logic code is behind the design code in separated file. The code file is named with the extension .aspx.vb

This kind of file model can be opted while creating a particular page as illustrated below. Visual studio gives an option for opting this, by checking the checkbox in a dialogue. If checked, web application will have two separate files. One is for design code and other is for logic code.

Figure: Opting for creating both code file separately.



After selecting this option & clicking on **Add Button** will create a web page with the name chosen by developer. It will automatically create both files separately.

In the figure above, we have chosen the file name to be Default2.aspx. So after adding file it will have design file Default2.aspx & code file Default2.aspx.vb. Every design file contains information about its code file and from where to link it. In design file, **<%@ Page %>** contains this information.

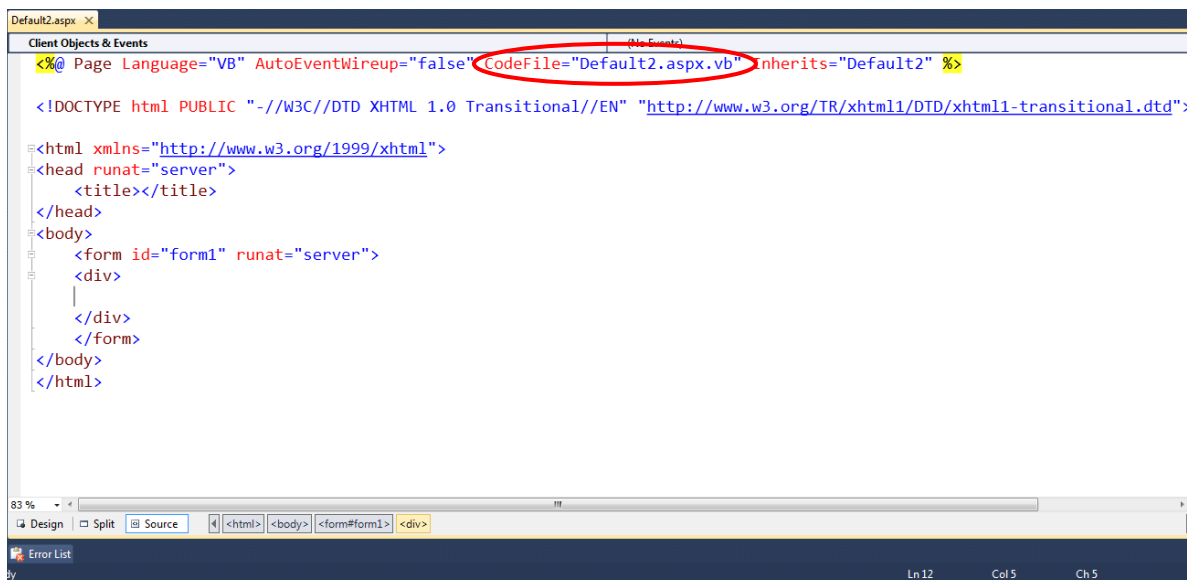


Figure: Design Code File Default2.aspx

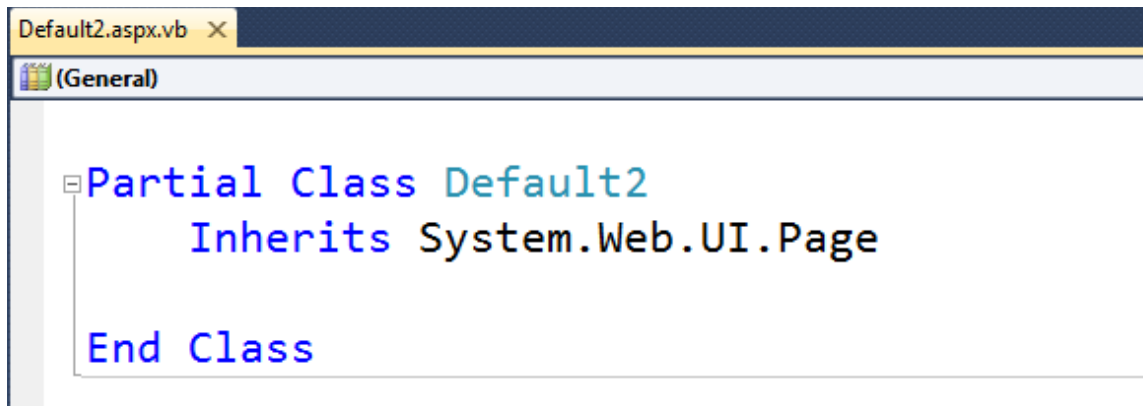


Figure: VB Code File Default2.aspx.vb

Advantages:

- Design and logic can be applied at the same time as both are in different files.
- Ease in accessing page and testing of code because of separate code file.
- Proper separation of codes enables easy management and reuse of code.

Disadvantages:

- Increases size of project/application as each page will have two files.
- Not suitable for simple pages that doesn't contain any logic code to be applied.

• **Inline Code and Code-behind file model**

To make a web page that performs particular task or job, ASP.Net web server controls are added to web page. Controls can be added to the web page using any of the following method:

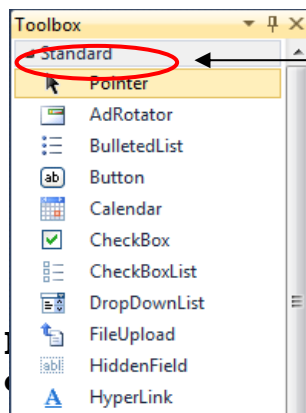
1. **At the time of designing a web page**
2. **At run time**

1. At the time of designing a web page [in design page - .aspx]

When you are designing a web page, controls can be added using any of the following method.

a. From Toolbox

Controls are added to the web page from Toolbox. All the standard ASP.Net web controls are available in the Standard Tab of Toolbox.



All the standard controls like Textbox, Button, Label, Calendar, HyperLink, Image, etc are available in the **Standard Tab** of Toolbox.

tab of Toolbox which contains ASP Web

To add a particular control we just either can drag the control from Toolbox & drop it in a web page where we want it to be placed or we can just double click on the control. In any of the case, we have to select that particular control first from Toolbox.

Example: Adding Control to Web Page

Web Page: adding_controls.aspx

In example illustrated below, an ASP.Net Button is added to the web page adding_controls.aspx

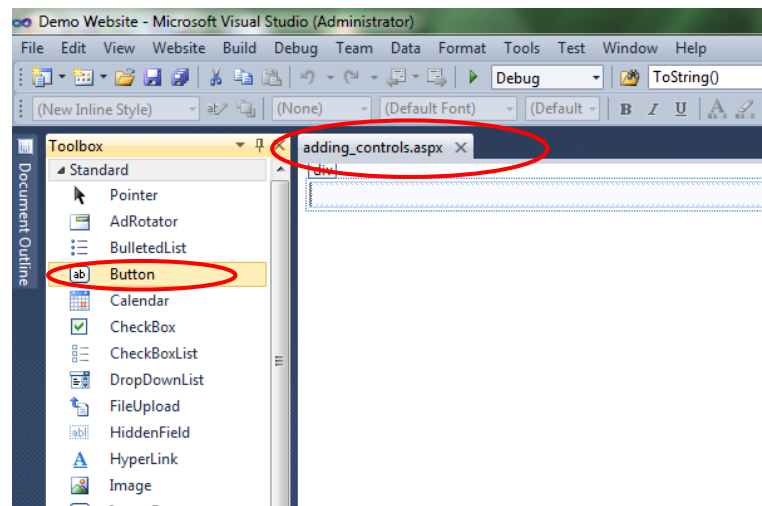


Figure: Adding Button control to web page [selecting control]

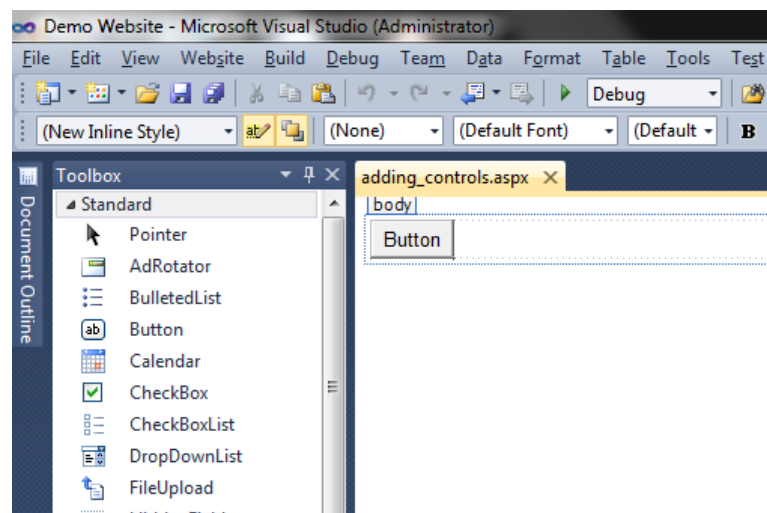


Figure: Adding Button control to web page [control added]

b. Form writing code in .aspx page

From design code you can also create any of the control. In our example, we have created ASP Button using following code.

```
<asp:Button ID="Button1" runat="server" Text="Button" />
```

The code above contains three attributes. **“ID”** represents the unique identity of the control. **“runat”** indicates .net engine that control has to be executed on server. **“Text”** indicates the value to be displayed as button text. The created button control is same as one created earlier from Toolbox.

2. At run time [in code page - .aspx.vb]

We can create any control from VB code as well. ASP.Net provides a large collection of in built classes & functions that helps us to create controls dynamically at run time. The example is illustrated below.

Example: Adding Control to Web Page

Design Code : adding_controls.aspx

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="adding_controls.aspx.vb" Inherits="adding_controls" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Adding Controls Using VB Control</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:Panel ID="Panel1" runat="server">    </asp:Panel>
    </form>
</body>
```



```
</html>
```

VB Code : adding_controls.aspx.vb

Partial Class adding_controls

Inherits System.Web.UI.Page

Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

'creating instance of the class button

Dim btn As New Button()

'assigning ID & Text.. as it is created by VB coding,

'runat=server is not needed as page will run on server

btn.ID = "btn1"

btn.Text = "Button1"

'adding control to panel...

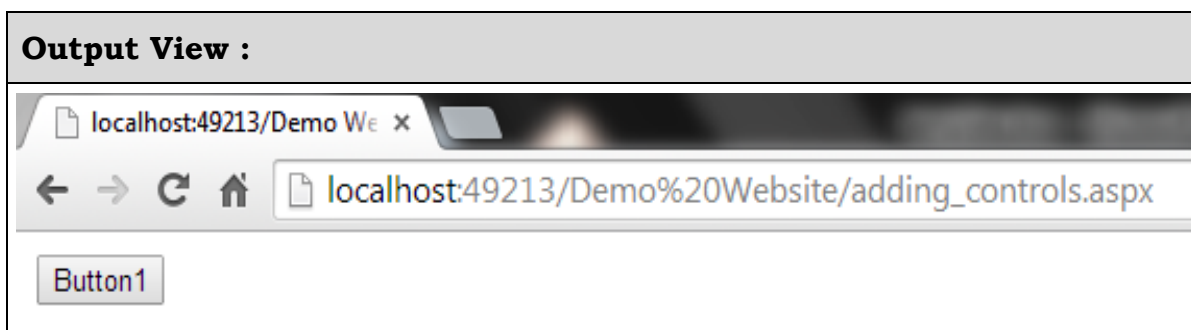
'panel is needed as controls using VB code can only be added to other

'ASP container control... hence it is required.

Panel1.Controls.Add(btn)

End Sub

End Class



2.2 Types of ASP.Net files

ASP.Net provides many types of files for different use. These file types are managed by ASP.Net & mapped in IIS.

File Type :	.asax
Location :	Application Root
Description :	Contains Application-Level & Session-Level event & their code. Extended version of Global.asa file of ASP. Code needed to be executed when a particular stage of page life cycle occurs can be added here.
File Type :	.ascx
Location :	Application Root or other sub directory
Description :	Web user control. May contain logic code & controls for any particular task. Control i.e. reusable & custom developed . Ex: Login Control for a web application. Can be used in other pages or application.
File Type :	.ashx
Location :	Application Root or other sub directory

Description :	An event handler file i.e. invoked when request for .aspx web is made. Different handlers are invoked depending upon the request. Ex: Most common is ASP.Net Page Handler for .aspx page request.
File Type :	.aspx
Location :	Application Root or other sub directory
Description :	ASP.Net web page that contains different web server controls & may contain styles or other scripting logic. Ex: Default.aspx i.e. default web page of any web application. Each .aspx page contains logic code separately or in same page, depending upon which option for a web page is chosen.
File Type :	.cs or .vb or .js
Location :	App_Code subdirectory, or in the case of a code-behind file for an ASP.NET page, in the same directory as the Web page.
Description :	Contains logic code to be executed at a particular stage of a web page life cycle or when a particular event is raised by user or application. Ex: The business logic for a login control of a web page is coded in this file depending upon which language is selected (C# or VB)
File Type :	.config
Location :	Application Root or other sub directory
Description :	Configuration setting files for a particular application or for whole machine/server. Ex: Web.config – for application level configuration settings Machine.config - for application level configuration

File Type :	.csproj, .vbproj
Location :	Visual Studio project directory.
Description :	Project files for a Visual Studio Web-application project.
File Type :	.sitemap
Location :	Application root.
Description :	A sitemap file that defines the logical structure of the Web application.
File Type :	.skin
Location :	App_Themes subdirectory
Description :	A skin file that contains property settings to apply to Web controls for consistent formatting. Used for universal formatting of web page controls such as height, width, etc.
File Type :	.master
Location :	Application Root or other sub directory
Description :	<p>A common master layout in whole application or in any particular module of web application for the other web pages.</p> <p>Ex: The Header, Logo, Menu Bar, Footer of a web application may be same throughout application for all web pages. It is designed and/or coded once in this file & then its reference is applied to the pages.</p>
File Type :	.mdf
Location :	App_Data subdirectory.

Description :	A SQL Server Express database file. Contains data in data tables & connected to the web pages using its in-built classes & methods. Ex: Student_details.mdf – SQL database for storing student details.
File Type :	.mdb
Location :	App_Data subdirectory.
Description :	An Access database file. Contains data in data tables & connected to the web pages using its in-built classes & methods. Ex: Student_details.mdb – MS Access database for storing student details.
File Type :	.sln
Location :	Visual Studio project directory.
Description :	Solution file for any web application. Used for quick loading of web application or project into IDE.

2.3 Web Page Life Cycle

A series of sequence in a particular order i.e. to be occurred from request of a web page to completion of web page task is known as web page life cycle.

Web page life cycle is divided into series of stages that are achieved throughout the life cycle of a web page. The span of web page life cycle is from its page request to page is destroyed.

Page Request

It is occurred before page life cycle starts. At this stage, it determined that whether a page needs to be compiled or parsed. A cached copy may be sent sometimes in response the request without running a page.

Example: user requests **www.studenthelper.com/register.aspx** from browser.

Start

Life cycle begins with this stage. At this stage, the page properties such as request or response are set. Whether the page is to be sent for the first time as a fresh copy of page or it is a posted back page with some data or code to be executed is determined in this stage.

Example: Server determines whether to send parsed/complied page or a fresh copy.

Page Initialization

Controls of the page are set at this stage with their Unique IDs. Also, any kinds of property settings or themes are applied in this stage. Server starts to send page & if it is a post back page, data is yet not available on client side.

Example: the page starts to be initialized on client browser as a fresh page. All themes & formatting is applied here.

Load

The requested web page arrives at the client browser & controls are converted into HTML. Loading of a web page completes & appropriate property is set as well in this stage.

Example: the page www.studenthelper.com/register.aspx is received on client browser.

Validation

In this stage, data is checked before sending page back to the server for processing or execution. Using various validation controls, validation process is carried out to make sure that the corrected data is posted back.

Example: using different validation controls, it is checked if the data entered by user is valid & proper or not, before allowing user to get registered into application.

Postback Handling

Appropriate event handlers are called while page is posted back at this stage. It may contain the execution of code or script on server side as well while event occurs.

Example: postback handlers are called & events occur accordingly.

Rendering

After the successful execution of logic code, output has to be sent to client browser. This process is carried over in this stage. View state is saved for the controls.

Example: The student is registered if valid data is given & output result such as login credentials or welcome message are stored to be rendered on client.

Unload

This is the last stage of page life cycle. In this stage, page is loaded on client browser & all processing is finished.

Example: www.studenthelper.com/register.aspx is completely arrived at client again after successful registration of user & is ready to be discarded.

Web page life cycle stages contains their events in ASP.Net and are as followed.

Page Initialization	PreInit
	Init
	InitComplete
Load	PreLoad
	Load
	ControlEvents
	LoadComplete
Rendering	PreRender
	SaveStateComplete
	Render
Unload	Unload

2.4 Web Form Processing Stages [Roundtrip]

Whenever user requires any web page it is requested & if it is found on sever, it 3333is sent back to the client in response of the request. Client/ user may require some execution of code in web page.

In ASP.Net all the logic execution is carried out on server. So, to execute any scripting core or any other business logic the web page has to be sent back to server with needed data.

At server side the logic execution is done & result is sent back to the client.

Executing code of a web page on server is known as web page processing.

Web pages of ASP.Net are needed to be processed on the server because the client browser only understands HTML code. Because of that after web page processing end result is converted into HTML code first & then it is sent to client browser. Client side scripting (like JavaScript) & CSS is applied then after on web page at client side.

To understand the web page processing, client server mechanism has to be understood.



In the diagram above,

Step 1 indicates that the client sends the web page with valid data to server for processing.

Step 2 indicates that the server side scripting & logic code execution is carried out.

Step 3 indicates that the result is sent back to client in form of HTML.

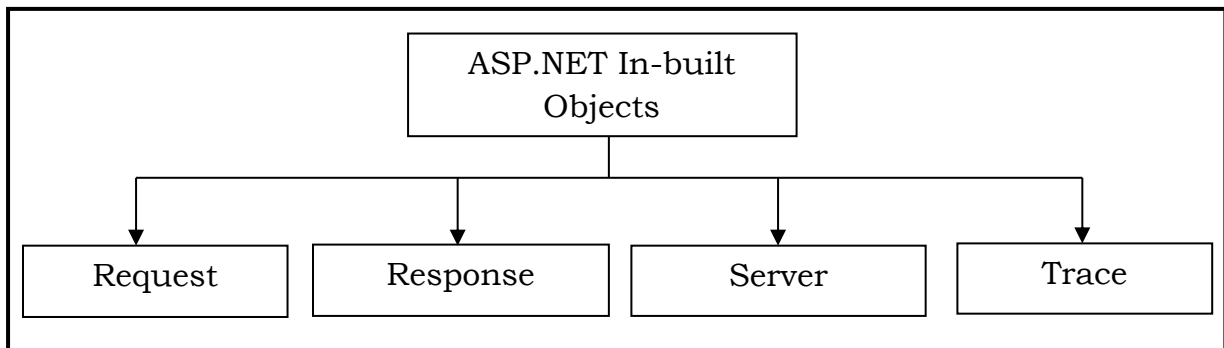
Step 4 indicates that the additional client side scripting and/or css are applied.

This whole process can be divided into stages of web page life cycle. Step 1 & 2 can be said the validation & postback stage. Step 3 can be said rendering & step 4 can be said the unload stage of life cycle of web page.

Client sends the web page with data for processing back to server for execution. Server executes the code & sends the result back to client. This whole process is known as Roundtrip.

2.5 ASP.Net In-built Objects

ASP.Net provides many in-built objects of classes that are useful for business logic & other tasks. ASP.Net provides following objects.



2.5.1 Request Object

The Request object is used to get information from a visitor. It enables ASP.NET to read the HTTP values sent by a client during a Web request. One common use of request object is with QueryString.

Example : `val = Request.QueryString("KeyName")`

Where **val** is any variable or appropriate property of any ASP server control.

2.5.2 Response Object

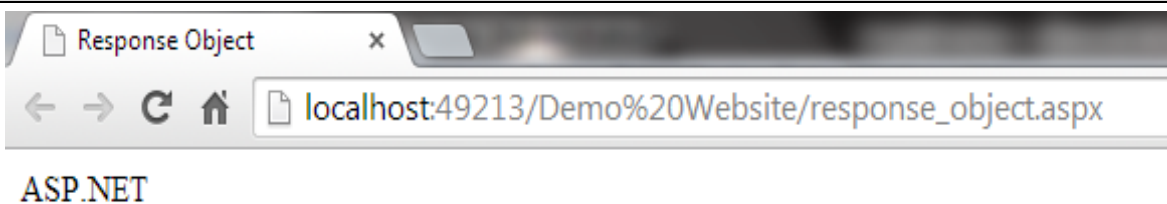
The ASP Response object is used to send output to the user from the server. It can be used to write text, data on web page & also navigation from one page to another is also possible using response object. We can also create cookies using response object.

Example I : Writing static text on web page

VB Code : response_object.aspx.vb

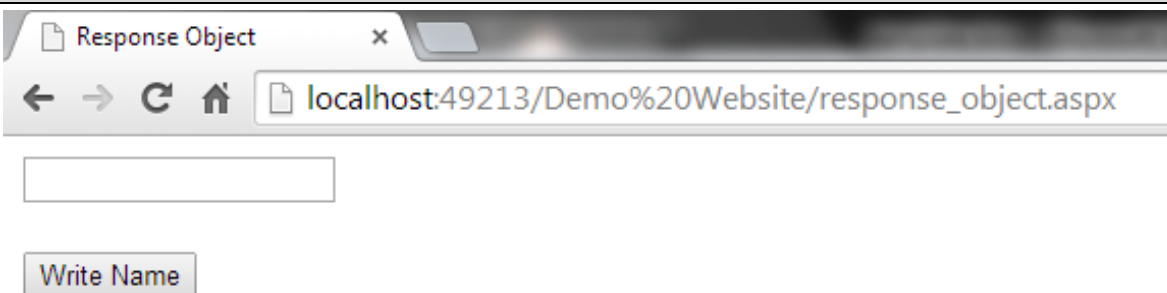
```
Partial Class response_object
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
        System.EventArgs) Handles Me.Load
        'Write() method is used to write data on web page
        Response.Write("ASP.NET")
        'data is always written from top-left corner of web page body.
    End Sub
End Class
```

Output View :



Example II : Writing any data on web page**Design Code :** response_object.aspx

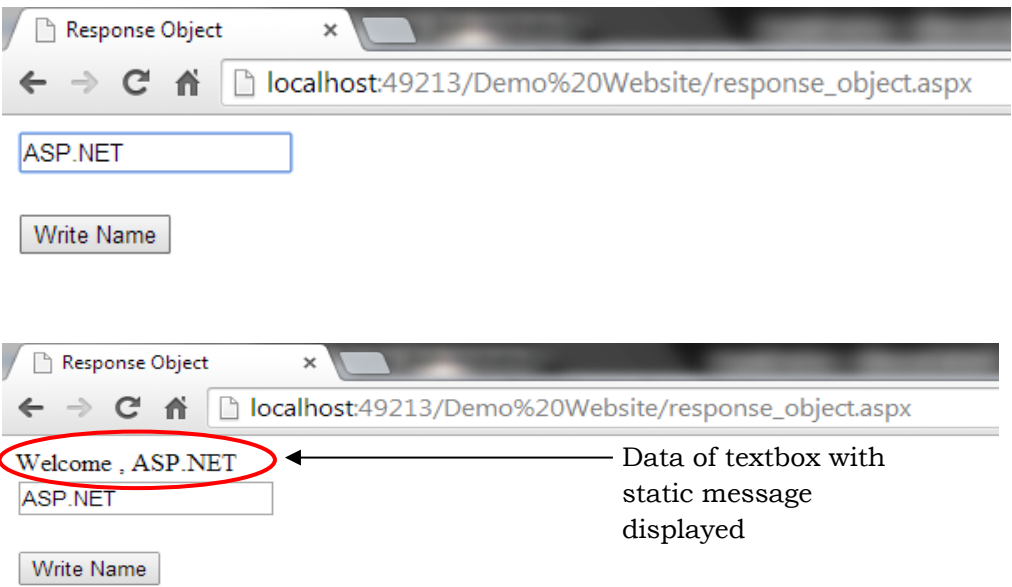
```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="response_object.aspx.vb" Inherits="response_object" %>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Response Object</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:TextBox ID="txt_name" runat="server"></asp:TextBox>
        <br /> <br />
        <asp:Button ID="btn_name" runat="server" Text="Write Name" />
    </form>
</body>
</html>
```

Design View : response_object.aspx

VB Code : response_object.aspx.vb

```
Partial Class response_object
    Inherits System.Web.UI.Page
    Protected Sub btn_name_Click(ByVal sender As Object, ByVal e As
        System.EventArgs) Handles btn_name.Click
        Dim name As String
        'Trim() removes extra white space before the first character
        'after the last character of data entered in textbox
        name = txt_name.Text.Trim()
        Response.Write("Welcome , " + name)
    End Sub
End Class
```

Output View :



The image displays two sequential screenshots of a web browser window titled 'Response Object'. The address bar shows the URL 'localhost:49213/Demo%20Website/response_object.aspx'. In the first screenshot, a text box contains 'ASP.NET' and a 'Write Name' button is below it. In the second screenshot, the text box now displays 'Welcome , ASP.NET', which is circled in red. An arrow points from the text 'Data of textbox with static message displayed' to this circled text. The 'Write Name' button remains below the text box.

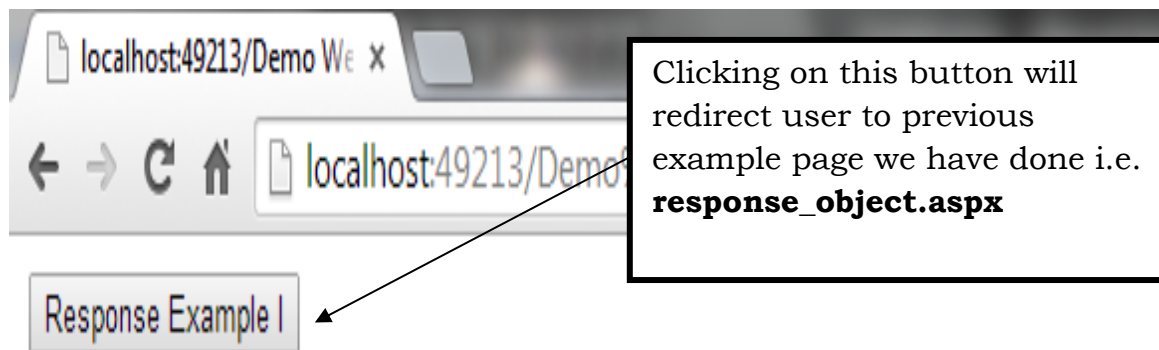
Example III : Navigating from one page to another

Design Code : response_redirect.aspx

```
<%@ Page Language="VB" AutoEventWireup="false"
CodeFile="response_object.aspx.vb" Inherits="response_object" %>

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>Response Object</title>
</head>
<body>
    <form id="form1" runat="server">
        <asp:Button ID="btn1" runat="server" Text="Response Example I" />
    </form>
</body>
</html>
```

Design View : response_redirect.aspx



VB Code : response_object.aspx.vb

```
Partial Class response_redirect
    Inherits System.Web.UI.Page
    Protected Sub btn1_Click(ByVal sender As Object, ByVal e As
        System.EventArgs) Handles btn1.Click
        Response.Redirect("response_object.aspx")
    End Sub
End Class
```

Output View :

2.5.3 Server Object

The ASP Response object is used to access properties and methods on the server. Some of the data of the server can be accessed using properties & methods given below.

Method :	Server. Transfer ("URL")
Description :	Used to transfer the control from one web page to another. Works same as Response.Redirect("url") but can be used for internal navigation in a web application. Faster than Response.Redirect("url") as it is pre decided that the page in the URL will be in the same application.
Example :	Server.Transfer("response_object.aspx"), Transfers control to page specified.
Method :	Server. Mappath ("location_directory")
Description :	Return the complete path of the directory specified.
Example :	Server.Mappath("../images") Will map complete path from root directory to images folder.
Property :	Server. MachineName
Description :	Returns machine name on which web application is running.
Example :	Response.Write("Server.MachineName")

2.5.4 Trace Object

Provides a way to display both system and custom trace diagnostic message in HTTP page output.

Exercise

1. Explain Inline Code Vs. Code Behind File.
2. List & explain ASP.NET page life cycle.
3. Explain Roundtrip in detail.
4. List & explain the page events available in ASP.Net with its sequence & purpose.
5. Explain the following ASP.Net objects with example & Syntax.
 - a. Response
 - b. Request
 - c. Server
 - d. Context
 - e. Application
 - f. Session

*******END OF CHAPTER*******