ASP.NET
State Management
lecture2

Madhuri Sawant

PostBack

- What is PostBack ?
- Postback is actually sending all the information from client to web server, then web server process all those contents and returns back to client.
- A postback originates from the client side browser. When the web page and its contents are sent to the web server for processing some information and then, the web server posts the same page back to the client browser.

PostBack

- What is AutoPostBack ?
- Then, what is AutoPostBack, Autopostback is the mechanism, by which the page will be posted back to the server automatically based on some events in the web controls. In some of the web controls, the property called auto post back, which if set to true, will send the request to the server when an event happens in the control
- Eg DropDown box (ComboBox) has the property autopostback .If we set the property to true ,when ever the user selects a different value in the combobox ,and event will be fired in the server , a request will be send to the server.

PostBack

- Whenever we set autopostback attribute to true in any of the controls, the .net framework will automatically insert few code in to the HTML generated to implement this functionality.
- a. A Java script method with name __doPostBack (eventtarget, eventargument)
  b. Two Hidden variables with name __EVENTTARGET and __EVENTARGUMENT
  c. OnChange JavaScript event to the control

PostBack

- a. __EVENTTARGET and __EVENTARGUMENT

  The __EVENTTARGET hidden variable will tell the server ,which control actually does the server side event firing so that the framework can fire the server side event for that control.

  The __ EVENTARGUMENT variable is used to provide additional event information if needed by the application, which can be accessed in the server.

PostBack

- b. __doPostBack (eventtarget, eventargument)
- This method is inserted to the HTML source to implement the autopostback functionality. This method will submit the form, when ever called. The two parameters in this method i.e. eventtarget and eventargument do the actual work of selecting the control to fire the event.
- This method will set the value of the __EVENTTARGET hidden variable with the eventtarget parameter and __ EVENTARGUMENT value with the eventargument parameter.
- The next activity is to submit the form, so that in the server side, the framework will check for the name of the control in the __EVENTTARGET hidden variable and will fire the appropriate event.

PostBack

- c. OnChange event.

  This event is added by the framework to any of the control where autopostback is set to true, this method will fire the client side OnChange event and calls the __doPostBack event with the name of the control where the OnChange event is happened

PostBack

- What is IsPostBack ?
- IsPostBack property is used by the Page to determine whether the page is posted back from the client. If IsPostBack property is false, then the page is loading for the first time, and if true, then the request is because of some event generated by web controls.
- IsPostBack is used when we want to load some information when the page loads,

State Management

- State Management :- How you store information over the lifetime of your application
- View State :- One of the most common ways to store information is in view state
- View State uses a hidden field that ASP.NET automatically inserts in the final, rendered HTML of a web page.
- It’s a perfect place to store information that’s used for multiple postbacks in a single web page.

View State

- View State is an ASP.NET feature that provides for retaining the values of page and control properties that change from one execution of a page to another.
- Before ASP.NET sends a page back to the client, it determines what changes the program has made to the properties of the page and its controls. These changes are encoded into a string that's assigned to the value of a hidden input field named _VIEWSTATE .
- When the page is posted back to the server ,the _VIEWSTATE field is sent back to the server along with the HTTP request. Then ,ASP.NET retrieves the property values from the _VIEWSTATE field and uses them to restore the page and control properties .
- View State is not used to restore data entered by a user into a textbox or any other input control unless the control responds to change events.

ViewState

- To add own data to view state ,store the data in a view state object that's created from the StateBag class

Property	Description
Item(name)	The value of the view state item with the specified name.
count	The number of items in the view state collection
Keys	A collection of keys for all of the items in the view state collection
values	A collection of values for all of the items in the view state collection

View State

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles Button1.Click
        Dim count As Integer
        If ViewState("count") Is Nothing Then
            count = 1
        Else
            count = CType(ViewState("count"), Integer) + 1
        End If
        ViewState("count") = count
        Label1.Text = "Counter:" & count.ToString

    End Sub
End Class
```

STEP 1

The ASP.NET Web page is visited by a user for the first time.

The rendered HTML displays the message "Hello, World!" in the Verdana font.

Instantiation Stage:
lblMessage.Text = "Hello, World!"

Load View State Stage:
Nothing happens - no postback

Save View State Stage:
Nothing happens - no state changes

Render Stage:
The HTML markup is generated. The Label reads "Hello, World!"

STEP 2

The user clicks the "Change Message" Button, causing a postback.

The "Change Message" Button's Click event handler fires, setting the Label's Text property to "Goodbye, Everyone!" This change is recorded in the view state in the save view state stage.

The rendered HTML displays the message "Goodbye, Everyone!" in the Verdana font.

Instantation Stage:
lblMessage.Text = "Hello, World!"

Load View State Stage:
Nothing happens - no state to reload from last postback

Raise Postback Event Stage:
The Button's Click event fires. The Label's `Text` property is set to "Goodbye, Everyone!"

Save View State Stage:
The Label's `Text` property value is persisted in ViewState.

Render Stage:
The HTML markup is generated. The Label reads "Goodbye, Everyone!"

STEP 3

The user clicks the "Empty Postback" Button, causing a postback. Upon postback in the Instantiation stage the Label's Text property is set to "Hello, World!"

In the load view state stage, the Label's Text property is assigned back to "Goodbye Everyone!", since this was the saved Label state from the previous visit (see step 2).

The rendered HTML, then, is the message "Goodbye, Everyone!"

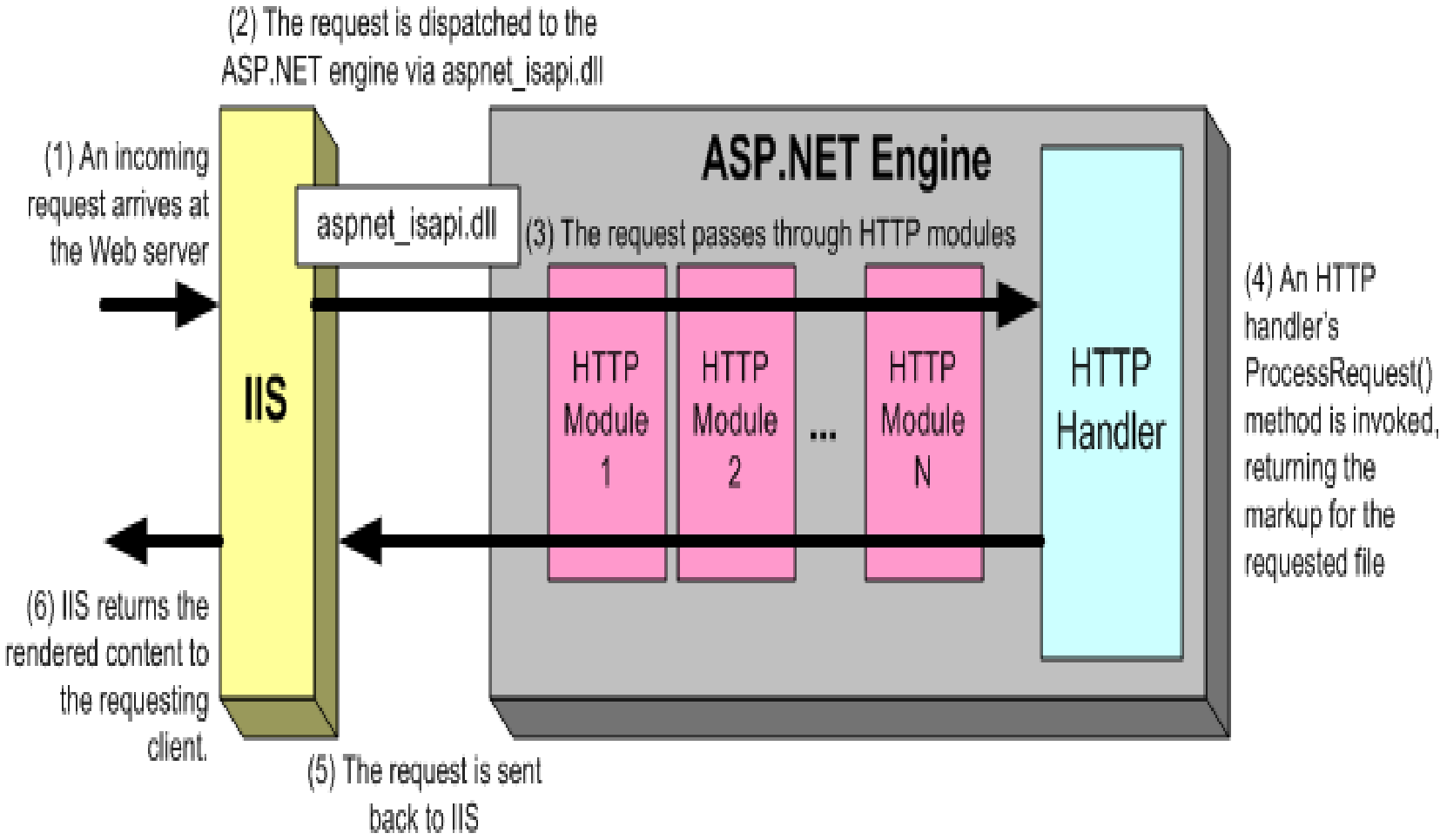
Instantiation Stage:
lblMessage.Text = "Hello, World!"

Load View State Stage:
The state from the previous postback is loaded. The Label's Text property is set to "Goodbye, Everyone!"

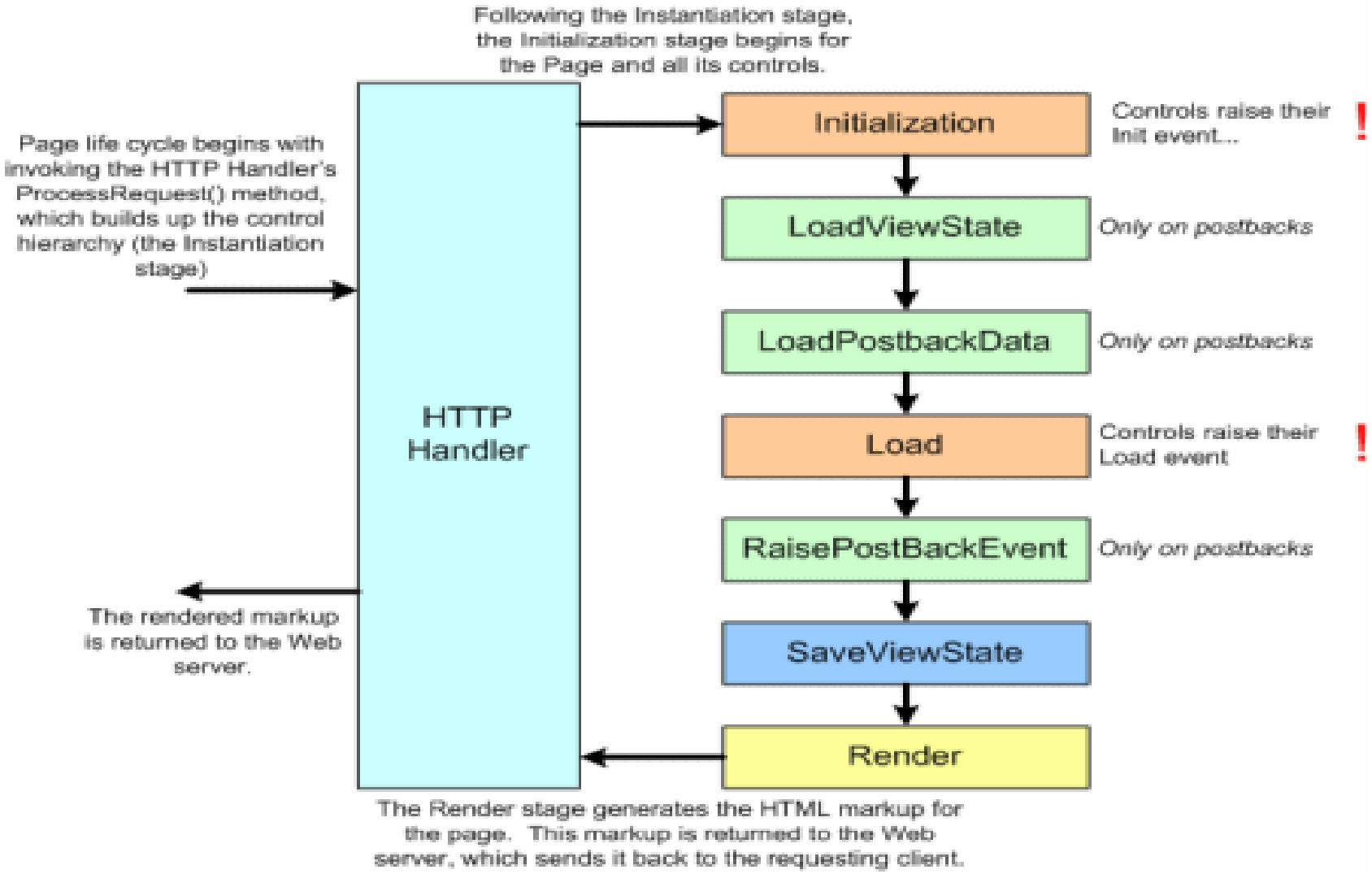
Save View State Stage:
The Label's `Text` property value is persisted in ViewState.

Render Stage:
The HTML markup is generated. The Label reads "Goodbye, Everyone!"

ASP.NET page life cycle



ASP.NET page life cycle



ASP.NET page life cycle

- Stage 0 - Instantiation
- The life cycle of the ASP.NET page begins with instantiation of the class that represents the requested ASP.NET Web page .
- The ASP.NET engine converts the HTML portion from its free-form text representation into a series of programmatically-created Web controls.
- When an ASP.NET Web page is visited for the first time after a change has been made to the HTML markup or Web control syntax in the .aspx page, the ASP.NET engine auto-generates a class.
- this autogenerated class, along with a compiled instance of the class, is stored
- The purpose of this autogenerated class is to programmatically create the page's *control hierarchy*.

ASP.NET page life cycle

- **Stage 1 - Initialization**
- After the control hierarchy has been built, the Page, along with all of the controls in its control hierarchy, enter the initialization stage. This stage is marked by having the Page and controls fire their Init events. At this point in the page life cycle, the control hierarchy has been constructed, and the Web control properties that are specified in the declarative syntax have been assigned.
- **Stage 2 - Load View State**
- The load view state stage only happens when the page has been posted back. During this stage, the view state data that had been saved from the previous page visit is loaded and recursively populated into the control hierarchy of the Page. It is during this stage that the view state is *validated*.

ASP.NET page life cycle

- **Stage 3 - Load Postback Data**
- The load postback data stage also only happens when the page has been posted back. A server control can indicate that it is interested in examining the posted back data by implementing the IPostBackDataHandler interface. In this stage in the page life cycle, the Page class enumerates the posted back form fields, and searches for the corresponding server control. If it finds the control, it checks to see if the control implements the IPostBackDataHandler interface. If it does, it hands off the appropriate postback data to the server control by calling the control's LoadPostData() method. The server control would then update its state based on this postback data.

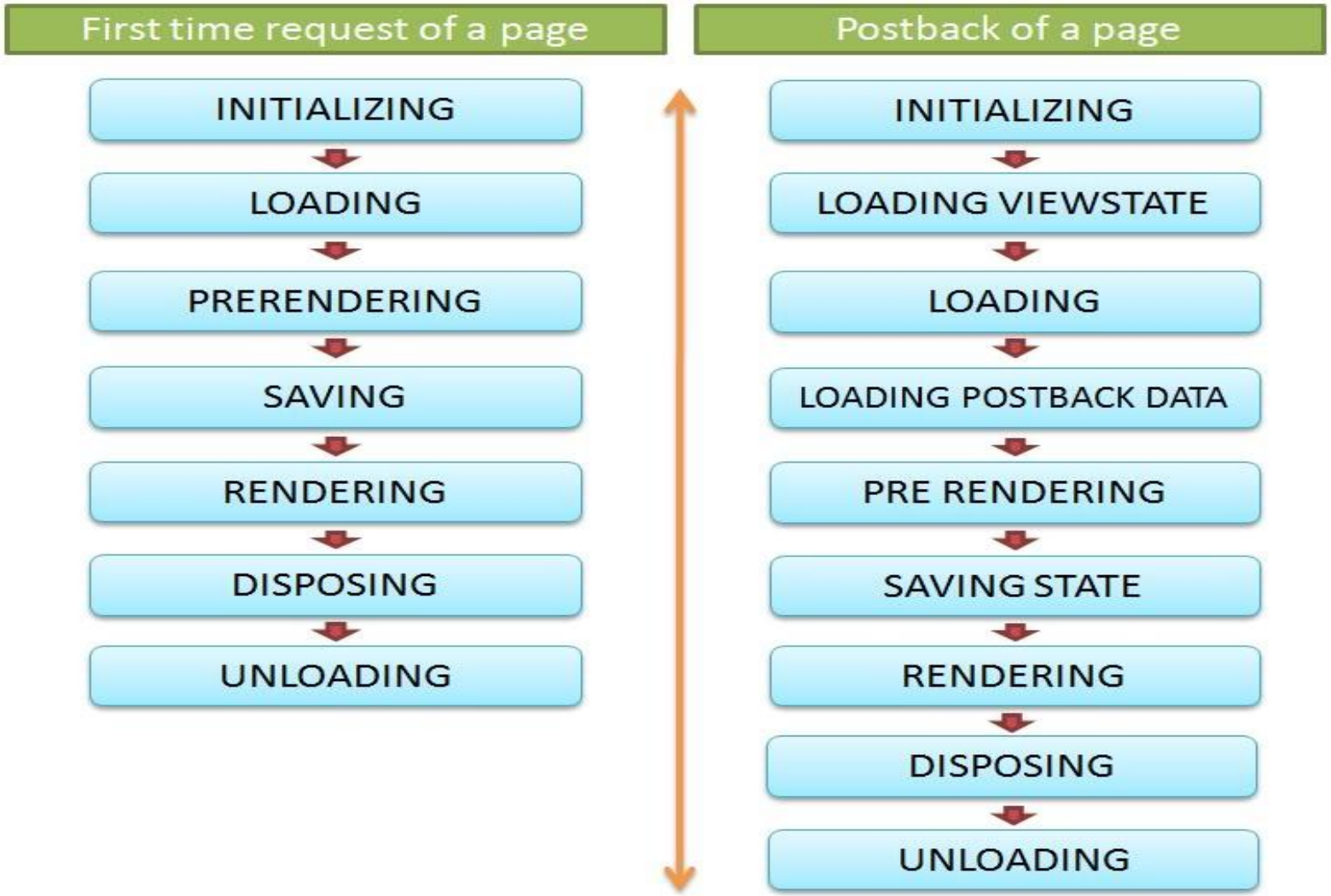
ASP.NET page life cycle

- **Stage 4 - Load**
- When the Load event fires, the view state has been loaded (from stage 2, Load View State), along with the postback data (from stage 3, Load Postback Data). If the page has been posted back, when the Load event fires we know that the page has been restored to its state from the previous page visit.
- **Stage 5 - Raise Postback Event**
- Certain server controls raise events with respect to changes that occurred between postbacks. Eg if the Web Form was posted back due to a Button Web control being clicked, the Button's Click event is fired during this stage.
- **Stage 6 :PreRendering**
- Associated value of the control is assigned. This is the last time changes of objects to save into the viewstate. After the execution of the method, controls value is locked for the viewstate.

ASP.NET page life cycle

- **Stage 7- Save View State**
- In the save view state stage, the Page class constructs the page's view state, which represents the state that must persist across postbacks. The page accomplishes this by recursively calling the SaveViewState() method of the controls in its control hierarchy. This combined, saved state is then serialized into a base-64 encoded string. In stage 7, when the page's Web Form is rendered, the view state is persisted in the page as a hidden form field.
- **Stage 8 - Render**
- In the render stage the HTML that is emitted to the client requesting the page is generated. The Page class accomplishes this by recursively invoking the RenderControl() method of each of the controls in its hierarchy.
- **Stage 9:Unload .**
- the rendered page is sent to the client and page properties

ASP.NET page life cycle



Cross Page Posting

- A cross-page postback is a technique that extends the postback mechanism so that one page can send the user to another page ,complete with all the information for that page.
- To use cross page posting ,you specify the URL of another page in the PostBackUrl property of a button control.
- Then when the user clicks the button,an HTTP Post message that contains the URL specified by the PostBackUrl property is sent back to the server.
- As a result ,the page with that URL is loaded and executed instead of the page that was orginally displayed.

Crosspage1.aspx

```
<%@ Page Language="VB" AutoEventWireup="false" CodeFile="CrossPage1.aspx.vb"
    Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
   <title>Cross Page1</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
<asp:Button ID="Button1" runat="server" PostBacKUrl="CrossPage2.aspx"
    Text="Cross-Page Postback" />
</div>
   </form>
</body>
</html>
```

Crosspage2.aspx.vb

```
Partial Class Default2
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
     System.EventArgs) Handles Me.Load
        If PreviousPage IsNot Nothing Then
            Label1.Text = "You came from a page titled::::" &
     PreviousPage.Title
        End If
    End Sub
End Class
```

Query String

- http://www.google.ca/search?q=organic+gardening
- The query string is the portion of the URL after the question mark.
- It defines a single variable named q,which contains the string

organic +gardening

- When using URL encoding, a query string that consists of attribute/value pairs is added to the end of a URL
- Query strings are frequently used in Anchor tags and hyperlinks to pass information from one page of an application to another or to display different information on the same page.
- Eq Two url with query string
- Order.aspx?cat=costumes
- Order.aspx?cat=props&prod=rat01

Query String

- Page1.aspx.vb

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
     System.EventArgs) Handles Button1.Click
        Dim url As String = "page2.aspx?no=" & TextBox1.Text &
     "&name=" & TextBox2.Text
        Response.Redirect(url)

    End Sub
End Class
```

Query String

Page2.aspx.vb

```
Partial Class page2
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
     System.EventArgs) Handles Me.Load
        Label1.Text = "no:" & Request.QueryString("no") & " name:" &
     Request.QueryString("name")
    End Sub
End Class
```

Query String

- The advantage of the query string is that its lightweight and doesn't exert any kind of burden on the server.
- Limitations
- Information is limited to simple strings
- Contain URL-legal characters
- Information is clearly visible to the user
- The user might decide to modify the query string and supply new values
- Cant place large amount of information in the query string
- Not compatible with most browsers

Cookies

- A cookie is a name/value pair that's stored in the user's browser or on the user's disk
- A web application sends a cookie to a browser via an HTTP response. Then each time the browser sends an HTTP request to the server, it attaches any cookies that are associated with that server.
- Retrieve cookies from the request object, and set cookies using the Response object.
- Advantage :- they work transparently without the user being aware that information needs to be stored.
- helping the Web site remember users.
- Can be easily used by any page in application
- Retained between visits , which allows for truly long –term storage.
- Disadvantage :- poor choice for complex or private information or large amounts of data

Cookies

- Two ways to create a cookie
- New HttpCookie(name)
- New HttpCookie(name,value)
- Common properties of the HttpCookie class

Property	Description
Expires	A DateTime value that indicates when the cookie should expire
Name	The cookie's name
Secure	A boolean value that indicates whether the cookie should be sent only when a secure connection is used.
value	The string value assigned to the cookie

- Code that creates a session cookie

```
Dim nameCookie as new HttpCookie(“UserName”,userName)
```

Code that creates a persistant cookie

```
Dim nameCookie as new HttpCookie(“UserName”)
nameCookie.Value=userName
nameCookie.Expires=Now.AddYears(1)
```

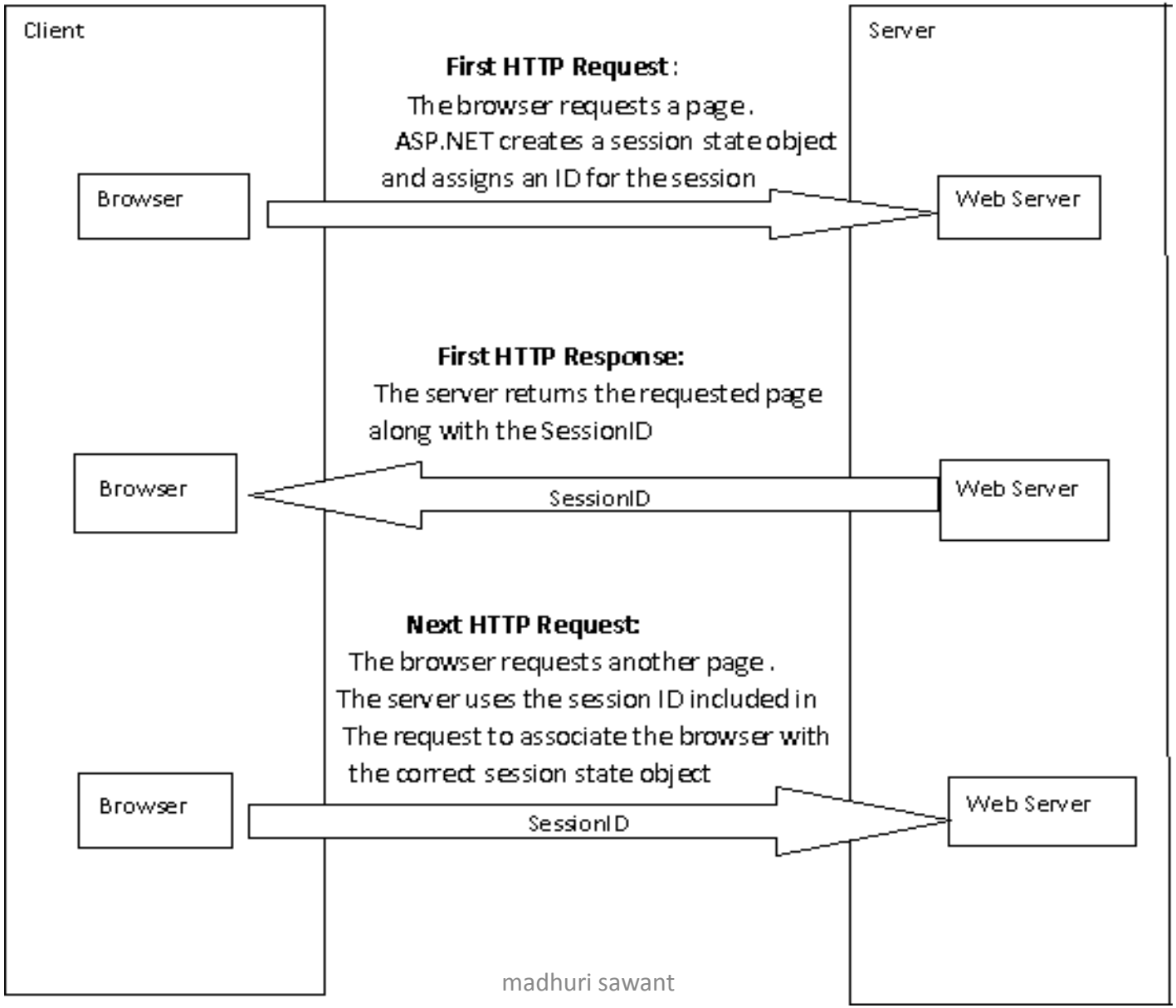
Cookies

```
Partial Class cookiedemo
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
      Me.Load
        Dim cook As HttpCookie = Request.Cookies("demo")
        If cook Is Nothing Then
            Label1.Text = "Cookie found"
        Else
            Label1.Text = "Welcome " & cook("Name")
        End If
    End Sub
    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
      Button1.Click
        Dim cook As HttpCookie = Request.Cookies("demo")
        If cook Is Nothing Then
            cook = New HttpCookie("demo")
        End If
        cook("Name") = TextBox1.Text
        cook.Expires = DateAndTime.Now.AddYears(1)
        Response.Cookies.Add(cook)
        Label1.Text = "Cookie created <br/><br/>"
        Label1.Text &= cook("name")
    End Sub
End Class
```

Session State

- ASP.NET uses **session state** to track the state of each user of an application To do that it creates a **session state object**
- The session state object includes a Session ID that's sent back to the browser as a cookie. Then the browser automatically returns the session ID cookie to the server with each request so the server can associate the browser with an existing session state object.
- Use the session state object to store and retrieve items across executions of an application.
- Because session state sends only the session ID to the browser, it doesn't slow response item.
- Session state objects are maintained in server memory.
- Every client that access the application has a different session and a distinct collection of information.

Session State



Session State

- Session Tracking :-
- ASP.NET tracks each session using a unique 120-bit identifier.
- This ID is the only piece of session relation information that is transmitted between the web server and the client.
- When the client presents the session ID,ASP.NET looks up the corresponding session ,retrieves the objects you stored previously ,and places them into a special collection so they can be accessed in your code. This process takes place automatically.
- The client must present the appropriate session ID with each request.
- This can be accomplished in two ways :
- → Using cookies
- →Using modified URLS

HttpSessionState class

Property	Description
SessionID	The uniqueID of the session
Item(name)	The value of the session state item with the specified name .
Count	The number of items in the session state collection

Method	Description
Add(name,value)	Adds an item to the session state collection
Clear()	Removes all items from the session state collection
Remove(name)	Removes the item with the specified name from the session state collection

- A statement that adds or updates a session state item

Session(“EMail”)=email

- A statement that retrieves the value of a session state item

Dim email as String=Session(“EMail”).ToString

- A statement that removes an item from session state

Session.Remove(“EMail”)

Session State

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Dim clickcount As Integer
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load
        If Session("clickcount") Is Nothing Then
            clickcount = 0
        Else
            clickcount = CInt(Session("clickcount"))
        End If
    End Sub

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
        Button1.Click
        clickcount += 1
        Label1.Text = "you have clicked the button   " & clickcount & "   times"
    End Sub

    Protected Sub Page_PreRender(ByVal sender As Object, ByVal e As System.EventArgs) Handles
        Me.PreRender
        Session("clickcount") = clickcount
    End Sub
End Class
```

Session State

Member	Description
Count	Provides the number of items in the current session collection
IsCookieless	Identifies whether the session is tracked with a cookie or modified URL
IsNewSession	Identifies whether the session was created only for the current request
Mode	Explains how ASP.NET stores session state information
SessionID	Provides a string with the unique session identifier for the current client
Timeout	Determines the number of minutes that will elapse before the current session is abandoned
Abandon()	Cancels the current session immediately and releases all the memory it occupied
Clear()	Removes all the session items but doesn't change the current session identifier

Application State

- An ASP.NET application is the collection of pages , code and other files within a single directory on a web server. An ASP.NET application corresponds to a single Visual Studio web project.
- The first time a user requests a page that resides in an application's directory , ASP.NET initializes the application.
- During that process , ASP.NET creates an application object from the HttpApplication class.
- This object is represented by a special class file named global.asax
- The application object can be accessed by any of the application's pages.
- Each time ASP.NET starts an application and creates an application object , it also creates an application state object from the HttpApplicationState class
- Use this object to store data in server memory that can be accessed by any page that's part of the application.

HttpApplicationState class

Property	Description
Item(name)	The value of the application state item with the specified name
Count	The number of items in the application state collection

Method	Description
Add(name,value)	Adds an item to the application state collection
Clear()	Removes all items from the application state collection
Remove(name)	Removes the item with the specified name from the application state collection
Lock()	Locks the application state collection so only the current user can access it.
UnLock()	Unlocks the application state collection so other users can acces it

Application State

```
Partial Class _Default
    Inherits System.Web.UI.Page
    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
      System.EventArgs) Handles Me.Load
        Application.Lock()
        Dim clickcount As Integer = CInt(Application("clickcount"))
        clickcount += 1
        Application("clickcount") = clickcount
        Application.UnLock()
        Label1.Text = clickcount.ToString
    End Sub
End Class
```