

(ASP.NET REDIRECTION)

What is Post Back ?

PostBack is actually sending all the information from client to web server, then web server process all those contents and returns back to the client. Most of the time ASP control will cause a post back (e. g. buttonclick) but some don't unless you tell them to do In certain events (**Listbox Index Changed,RadioButton Checked etc..**) in an ASP.NET page upon which a PostBack might be needed.

PostBack is the name given to the process of submitting an ASP.NET page to the server for processing. PostBack is done if certain credentials of the page are to be checked against some sources (such as verification of username and password using database). This is something that a client machine is not able to accomplish and thus these details have to be 'posted back' to the server.

In ASP.NET , when we design the web page , we do not provide any method and action property in the <form> tag ... let us understand post back working :

see the code of **login.aspx** page

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="login.aspx.cs" Inherits="login" %>

<!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></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>

            User ID :
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <br />
            <br />
            Password : <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
            <br />
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Submit" Width="97px" />

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

In the above code , we have taken two textboxes and one button for accepting user id and password and button to submit the details , originally we did not provide action and method property in form tag .

After you will run it on any browser and see the page source you will get certain changes in <form> tag like:

```
<!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><title>
</title></head>
<body>
<form method="post" action="login.aspx" id="form1">
<div class="aspNetHidden">
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
value="/wEPDwUKLTg0OTIzODQwNWRkrWp9NdqfxWM2rEq+tObpmx0vnxh3Ivisi5QAh
OHm0KA=" />
</div>
<div class="aspNetHidden">
<input type="hidden" name="__VIEWSTATEGENERATOR"
id="__VIEWSTATEGENERATOR" value="27777991" />
<input type="hidden" name="__EVENTVALIDATION" id="__EVENTVALIDATION"
value="/wEdAARDOCxtk8AISjLViHABUvADESCFkFW/RuhzY1oLb/NUVB2nXP6dhZn6m
KtmTGNHd3PN+DvxnwFeFeJ9MIBWR693jM/Ib5tjUhpp/p3b5s8GRmOkVOnLvVq2MYrJX
dseytc=" />
</div>
<div>
User ID :
<input name="TextBox1" type="text" id="TextBox1" />
<br />
<br />
Password :<input name="TextBox2" type="text" id="TextBox2" />
<br />
<br />
<br />
<input type="submit" name="Button1" value="Submit" id="Button1" style="width:97px;" />
</div>
</form>
</body>
</html>

```

You can see , that ASP.NET engine automatically adding two properties to the form tag while rendering the web page to client ..

<form method="post" action="login.aspx" id="form1">

method : either get or post , asp by default provide post method for security reasons.

action : name of the page to which data is to be submitted and called by server as response . By default in

ASP.NET , we get the name of the same web page in the action property .. like
action = "login.aspx" , the reason behind giving the same page reference in action is to provide the event driven approach in asp . It means when user click on the button or submit the page , all the values of the server controls and web page is sent to the server to the page mentioned in the action property , after getting information , server load the web page mentioned in the action property , means it will follow the ASP Page life cycle , during following the page life cycle it will call the event handlers which in directly a methodology for making asp event driven approach . It will bind the information from view state and render the same web page with the changes.

Let us know the method options in Deep :

The Get/Post methods are used to send client information to web server. The methods are specified in inside form element using method attribute.

Syntax:

```
<form method="get|post">
```

The method attribute uses either get or post. Default is GET method.

GET method:

This method appends form data to page request URL with key value pairs called as query string.

E.g.: [http://www.dotnetmirror.com/home.htm?key1=value1&key2=value2 \](http://www.dotnetmirror.com/home.htm?key1=value1&key2=value2)

- Page request URL and the form data information (query string) are separated by the ? Character.
- Get method is restricted to send up to 1024 characters only.
- Never use Get method to send sensitive information like password, login information to web server.
- Form data support only ASCII characters.Cannot used to send binary information like word document, text file and image data to server.
- We can access query string in ASP.NET using `Request.QueryString["key1"]`

Example: Usage of Get method In ASP.NET/HTML:

Below is the code of the page Login.aspx

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="login.aspx.cs" Inherits="login" %>

<!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></title>
</head>
<body>
    <form id="form1" method="get" action="successbyget.aspx" runat="server">
        <div>

            User ID :
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
            <br />
            <br />
            Password : <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
            <br />
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Submit" Width="97px" />

        </div>
    </form>
</body>
</html>

```

As you can see the form tag

```

<form id="form1" method="get" action="successbyget.aspx" runat="server">

```

```

method = "get"
action = "successbyget.aspx"

```

here action is not the reference of the same page , as we have seen by default in asp.
As user will submit the page , the information will be submitted to the successbyget web page
and it will be invoked after that .

Let us see the successbyget.aspx.cs web page code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class successbypost : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string id = Request.QueryString["TextBox1"].ToString();
        string password = Request.QueryString["TextBox2"].ToString();

        Response.Write("Id : " + id + "<br/>");
        Response.Write("Password : " + password + "<br/>");
    }
}

```

```
}
```

We have not done any changes to the successbyget.aspx page but made some changes in.aspx.cs of it..When using get method , we need to use Request.QueryString because data will be append into the url of the web page .

```
string id = Request.QueryString["TextBox1"].ToString();  
string password = Request.QueryString["TextBox2"].ToString();  
  
Request.QueryString["name of the control"].ToString();
```

It is very important to understand that QueryString takes the reference of name of the control not id ..

But we never provide the name property in any asp web server control but ASP engine by default adds the name property to the rendered html element with the same value as of id..

See the example below :

Asp textbox control is written in.aspx page :

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

After user request the web page then ASP engine will render the below element to the client:

```
<input name="TextBox1" type="text" id="TextBox1" />
```

It is very interesting concept in ASP , that ASP engine generates the html element from the asp element and render accordingly .

You can see the name ="TextBox1" property added automatically by ASP with the same value as of ID property which was not added in source.aspx page .

Post method:

- POST method transfers information over HTTP headers.
- Data transfer through HTTP headers so using secure HTTP protocol we can make sure that data is secured.
- No restriction on sending data size via Post and also we can send binary data or ASCII information using POST method.
- We can access form data in ASP.NET using Request.Form["key1"]

Usage of Post method:

- If the form data is large then use POST method because GET method cannot handle long URL's
- Form data contains Non-ASCII characters use POST because GET doesn't support it.

Example: Usage of Get method In ASP.NET/HTML:

We will use the same Login.aspx web page but with certain changes in the form tag like :

```

<%@ Page Language="C#" AutoEventWireup="true" CodeFile="login.aspx.cs" Inherits="login" %>

<!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></title>
</head>
<body>
    <form id="form1" method="post" action="successbypost.aspx" runat="server">
    <div>

        User ID :
        <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
        <br />
        <br />
        Password : <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
        <br />
        <br />
        <br />
        <asp:Button ID="Button1" runat="server" Text="Submit" Width="97px" />

    </div>
    </form>
</body>
</html>

```

here we have changed the two properties like
method="post"
action="successbypost.aspx"

let us see the code of successbypost.aspx.cs page code , you need not to do any changes in aspx page

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class successbypost : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string id = Request.Form["TextBox1"].ToString();
        string password = Request.Form["TextBox2"].ToString();

        Response.Write("Id : " + id + "<br/>");
        Response.Write("Password : " + password + "<br/>");
    }
}

```

Here we have used Request.Form["name of the control"] irrespective of Request.QueryString[.. this time the information will not be append on the url while sending to the server .

Difference between GET and Post:

GET	POST
Data will be arranged in HTTP header by appending to the URL as query string	Data will be arranged in HTTP message body.
Data is in query string so user can view the data	Not visible to user
Less secured compared to POST method because data is in query string so it will be saved in browser history and web server logs	Bit safer than GET method because data is not saved in history or web server logs
As data is saved in URL so its saves only 2048 bytes data	Can be used for any amount of data
Can be bookmarked	Can't bookmarked
Hacking will be easy	Hacking is difficult
Only ASCII character data type allowed	No restrictions. Allows binary data also
Caching is possible	No caching

So most of things are cleared by now regarding the actual code and rendered code of aspx page.

What is AutoPostBack Property in ASP.NET:

If we create a web Page, which consists of one or more Web Controls that are configured to use AutoPostBack (Every Web controls will have their own AutoPostBack property), the ASP.Net adds a special JavaScript function to the rendered HTML Page. This function is named `_doPostBack()` . When Called, it triggers a PostBack, sending data back to the web Server.

ASP.NET also adds two additional hidden input fields that are used to pass information back to the server. This information consists of ID of the Control that raised the event and any additional information if needed. These fields will empty initially as shown below,

```
<input type="hidden" name="__EVENTTARGET" id="__EVENTTARGET" value="" />
<input type="hidden" name="__EVENTARGUMENT" id="__EVENTARGUMENT" value="" />
```

The `_doPostBack()` function has the responsibility for setting these values with the appropriate information about the event and the submitting the form. The `_doPostBack()` function is shown below:

```
<script language="text/javascript">
function __doPostBack(eventTarget, eventArgument) {
if (!theForm.onsubmit || (theForm.onsubmit() != false)) {
theForm.__EVENTTARGET.value = eventTarget;
```

```

theForm.__EVENTARGUMENT.value = eventArgument;
theForm.submit();
}
</script>

```

ASP.NET generates the `_doPostBack()` function automatically, provided at least one control on the page uses automatic postbacks.

Any Control that has its `AutoPostBack` Property set to true is connected to the `_doPostBack()` function using the `onclick` or `onchange` attributes. These attributes indicate what action should Browser take in response to the Client-Side javascript events `onclick` and `onchange`.

In other words, ASP.Net automatically changes a client-side javascript event into a server-side ASP.Net event, using the `_doPostBack()` function as an intermediary.

Life Cycle of a Web Page:

To work with the ASP.Net Web Controls events, we need a solid understanding of the web page life cycle. The following actions will be taken place when a user changes a control that has the `AutoPostBack` property set to true :

1. On the client side, the JavaScript `_doPostBack` function is invoked, and the page is resubmitted to the server.
2. ASP.NET re-creates the Page object using the .aspx file.
3. ASP.NET retrieves state information from the hidden view state field and updates the controls accordingly.
4. The `Page.Load` event is fired.
5. The appropriate change event is fired for the control. (If more than one control has been changed, the order of change events is undetermined.)
6. The `Page.PreRender` event fires, and the page is rendered (transformed from a set of objects to an HTML page).
7. Finally, the `Page.Unload` event is fired.
8. The new page is sent to the client.

To watch these events in action, we can create a simple event tracker application. All this application does is write a new entry to a list control every time one of the events it's monitoring occurs. This allows you to see the order in which events are triggered.

Event Tracker Web Page:

I have shown Markup codes and C# Codes below to make this works,

```

<%@Page
Language="C#"AutoEventWireup="true"CodeFile="EventTracker.aspx.cs"Inherits="EventTracker"
%>

<!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>Untitled Page</title>
</head>
<body>
<form id="form1" runat="server">
<div>
<h1>Controls being monitored for change events:</h1>
<asp:TextBox ID="txt" runat="server" AutoPostBack="true"
OnTextChanged="CtrlChanged"/>
<br/><br/>
<asp:CheckBox ID="chk" runat="server" AutoPostBack="true"
OnCheckedChanged="CtrlChanged"/>
<br/><br/>
<asp:RadioButton ID="opt1" runat="server" GroupName="Sample"
AutoPostBack="true" OnCheckedChanged="CtrlChanged"/>
<asp:RadioButton ID="opt2" runat="server" GroupName="Sample"
AutoPostBack="true" OnCheckedChanged="CtrlChanged"/>
<h1>List of events:</h1>
<asp:ListBox ID="lstEvents" runat="server" Width="355px"
Height="150px"/><br/>
<br/><br/><br/>
</div>
</form>
</body>
</html>

```

Below is the aspx.cs page code

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Default3 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Log("<< Page_Load >>");
    }

    protected void Page_PreRender(object sender, EventArgs e)
    {
        // When the Page.PreRender event occurs, it is too late
        // to change the list.
        Log("Page_PreRender");
    }

    protected void CtrlChanged(Object sender, EventArgs e)
    {

```

```

        // Find the control ID of the sender.
        // This requires converting the Object type into a Control class.
        string ctrlName = ((Control)sender).ID;
        Log(ctrlName + " Changed");
    }
    private void Log(string entry)
    {
        lstEvents.Items.Add(entry);
        // Select the last item to scroll the list so the most recent
        // entries are visible.
        lstEvents.SelectedIndex = lstEvents.Items.Count - 1;
    }
}

```

Above is the good example of explaining the Autopostback feature in ASP.NET with the implementation of Page Life Cycle.

What is IsPostBack ?


IsPostBack is a property of the Asp.Net page that tells whether or not the page is on its initial load or if a user has perform a button on your web page that has caused the page to post back to itself. The value of the Page.IsPostBack property will be set to true when the page is executing after a postback, and false otherwise. We can check the value of this property based on the value and we can populate the controls on the page.

Is Postback is normally used on page _load event to detect if the web page is getting generated due to postback requested by a control on the page or if the page is getting loaded for the first time.

Let us see the example below :

Let us take the example where we need to fill the values into the **DropDownList** control on page load event :

see the details.aspx web page



The screenshot shows a web form with the following elements:

- A text input field labeled "Name:".
- A dropdown menu labeled "Exam Centers:" with the value "Unbound" selected.
- A "Submit" button.

see the code below :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default3.aspx.cs" Inherits="Default3"%>

<!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></title>
</head>
<body>
    <form id="form1" runat="server">
        <div>
            Name : &nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&~
            <asp:TextBox ID="TextBox1" runat="server" Width="129px"></asp:TextBox>
            <br />
            Exam Centers :
            <asp:DropDownList ID="DropDownList2" runat="server" Height="22px" Width="134px">
            </asp:DropDownList>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Submit" />
        </div>
    </form>
</body>
</html>
```

Now you want that the name of centers will be added automatically in dropdownlist while page gets load ..

To implement it below is the code :

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class Default3 : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        DropDownList2.Items.Add(new ListItem("Agra"));
        DropDownList2.Items.Add(new ListItem("Bangalore"));
        DropDownList2.Items.Add(new ListItem("Delhi"));
        DropDownList2.Items.Add(new ListItem("Chandigarh"));
        DropDownList2.Items.Add(new ListItem("Kolkatta"));
        DropDownList2.Items.Add(new ListItem("Ahmedaba"));
        DropDownList2.Items.Add(new ListItem("Mumbai"));
    }
}
```

this code works fine and works also ..

But there is a very big issue , means if user click the button then page will get submitted to the server and all the values in dropdownlist and textbox will also be moved along with it , as we know it will post back itself so the same page will be rendered with same values as before submitted , but as we know the page will be recreated again as per page life cycle so again page load event will get fired due to which the same center list will be added again which will result in duplicate values.

Let us See the output when the fresh page is loaded from server .

Name :

Exam Centers :

Agra

Agra

Bangalore

Delhi

Chandigarh

Kolkatta

Ahmedaba

Mumbai

All the centers as mentioned in page load is added .

But As user click on the submit button the page post back itself then see the output below :

Name :

Exam Centers :

Agra

Agra

Bangalore

Delhi

Chandigarh

Kolkatta

Ahmedaba

Mumbai

Agra

Bangalore

Delhi

Chandigarh

Kolkatta

Ahmedaba

Mumbai

This time we are getting duplicate values of centers , because page will be created again , so as per page life cycle load event will be called again which will append the values to the existing list.

To avoid such circumstances , we can check that page is post back or not by property called `Page.IsPostBack`

So we need to change the code as given below :

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Page.IsPostBack == false)
    {
        DropDownList2.Items.Add(new ListItem("Agra"));
        DropDownList2.Items.Add(new ListItem("Bangalore"));
        DropDownList2.Items.Add(new ListItem("Delhi"));
        DropDownList2.Items.Add(new ListItem("Chandigarh"));
        DropDownList2.Items.Add(new ListItem("Kolkatta"));
        DropDownList2.Items.Add(new ListItem("Ahmedaba"));
        DropDownList2.Items.Add(new ListItem("Mumbai"));
    }
}
```

this time , items will be added only on fresh request means when page is not post back .

We can also replace the code

```
if (Page.IsPostBack == false)
{
    }
}
```

with

```
if (!Page.IsPostBack)
{
    }
}
```

Redirecting Users to Another Page with values

You will often want to redirect users to other pages as part of your Web application. ASP.NET provides the following ways for you to build redirection into your Web pages:

- Using hyperlinks on pages.
- Configuring cross-page posting, which enables you to specify an alternate target page when the current page is submitted.
- Redirecting pro grammatically by forcing the browser to request a different page.
- Redirecting pro grammatically by transferring control to a different page in the same Web application.

Each of these options is described below. A table at the end of the topic summarizes the options and provides guidelines to help you decide when to use each option.

A. Hyperlinks

You can use an HTML anchor tag (<a>) on an ASP.NET Web page to create static links, or you can pro grammatically control the link text and target URL of hyperlinks using the Hyper Link control. In this scenario, the user explicitly clicks a link and the browser transfers to the target page. **The target page is invoked using an HTTP GET command.** Therefore, no information about the source page is passed to the target page unless you specify a query

string on the URL of the target page. If the source and target page are in the same Web application, they can share information using session state or application state.

Characteristics

Performs new request on the target page.

- Does not pass current page information to the target page.
- Requires user initiation.
- Redirects to any page, not just pages in the same Web application.
- Enables you to share information between pages using query string or session state. (The HyperLink control enables you to create URL and query strings programmatically)

Usage

- When navigation should be under user control.
- For navigation without any additional processing, as in menus or lists of links.

Let us see the example :

Default.aspx pag:

```
<body>
  <form id="form1" runat="server">
    <div>

      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <br />
      <br />
      <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/Default2.aspx">Second
Page</asp:HyperLink>

    </div>
  </form>
</body>
```

here we have used HyperLink control with NavigationUrl , at run time it will render <a> tag of html as shown below

```
<a id="HyperLink1" href="Default2.aspx">Second Page</a>
```

As user will click on the link , it will send request to server for fresh call of Default2.aspx without sending any information of Default.aspx page .

Below is the code behind page of default2.aspx.cs below where I have done coding In page load event with the request to display the HttpMethod whether it is get or post , here it will be get .

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write(Request.HttpMethod);
}
```

Further if you want to transfer any information to the Default2.aspx page then can pass as QueryString as shown below :

```
<asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/Default2.aspx?id=admin">Second
Page</asp:HyperLink>
```

and do the code on.aspx.cs file as given below:

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write(Request.HttpMethod);

    Response.Write(Request.QueryString["id"].ToString());
}
```

Here we have retrieved the value by using QueryString["id"] where id is the identity name appended with url like above code

```
NavigateUrl="~/Default2.aspx?id=admin"
```

We have also seen that hyperlink internally rendered to <a> tag to client . There is a big difference and advantage of HyperLink over <a> tag , that is you can manipulate or change the NavigateUrl at run time by user end because HyperLink is server side .

Let us see the below example :

I have provided the textbox to user for entering the page name and one HyperLink control so that user can navigate to that page by clicking it ..

```
<body>
    <form id="form1" runat="server">
        <div>

            <asp:TextBox ID="TextBox1" runat="server"
                ontextchanged="TextBox1_TextChanged" AutoPostBack="True" ></asp:TextBox>
            <br />
            <br />

            <asp:HyperLink ID="HyperLink1" runat="server">Second Page</asp:HyperLink>

        </div>
    </form>
</body>
```

We have used ontextchanged event of textbox , which will be fired when user will leave it and it loses the focus from user when user will press tab or any area except textbox which is equivalent to onblur event in javascript.

But for the proper working of `onTextChanged` event handler at server side , we need to make the `autopostback` property true which we have discussed before.

B. Cross Page Posting:

By default, server control buttons in an ASP.NET Web Forms page post the page to itself. Cross-page posting enables you to configure a button on an ASP.NET Web Forms page to post the current page to a different page. A typical example is when creating a multi-page form. You can configure buttons on the page to move to the next and previous pages of the form.

Cross-page posting is similar to hyperlinks in that the transfer is initiated by a user action. However, in cross-page posting, **the target page is invoked using an HTTP POST command**, which sends the values of controls on the source page to the target page. In addition, if the source and target page are in the same Web application, the target page can access public properties of the source page. As always, all of the pages in the application can share information stored in session state or application state.

Under some circumstances, you might want to post one page to another page. For example, you might be creating a multi-page form that collects different information on each page. In that case, you can configure certain controls (those that implement the `IButtonControl` interface, such as the `Button` control) on the page to post to a different target page. This is referred to as cross-page posting. `Button` control is provided with `PostBackUrl` property . We can give the reference of the page to which page is to be posted on click of button.

Let us take the example of it:

Page : `login.aspx`



The image shows a simple login form within a dashed blue border. It contains three elements: a text label 'User ID :', a text input field, a text label 'Password :', another text input field, and a 'Submit' button at the bottom. The text labels are in a blue, serif font, and the button is a standard Windows-style button.


```

<body>
  <form id="form1" method="post" action="Default2.aspx" runat="server">
    <div>
      User ID : <br />
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
      <br />
      <br />
      Password : <br />
      <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox>
      <br />
      <br />
      <asp:Button ID="Button1" runat="server" Text="Submit" PostBackUrl="~/result.aspx"/>
    </div>
  </form>
</body>

```

Now if user will click on submit button then the whole page will be submitted to result.aspx page means all the values and control state will be posted to result.aspx.

Now what activities we can perform after page is posted to result.aspx page :

A.We can access the values of all the controls from login.aspx.

B.We can access the control properties like bgcolor,forecolor , font-style etc .

Accessing the input values of All controls of login.aspx into result.aspx page :

Do the code on page load event of result.aspx page .

```

protected void Page_Load(object sender, EventArgs e)
{
    string id = Request.Form["TextBox1"].ToString();
    string password = Request.Form["TextBox2"].ToString();

    Response.Write("ID : " + id + "<br/>");
    Response.Write("Password : " + password + "<br/>");
}

```

It will give you the output as below :

```

ID : abc1234
Password : mypassword

```

This is good approach for reading the posted values of controls but not up to the mark . Means if I want to read all list items of DropDownList , List Box control then I would not be able to do it by using the above approach , so for that we will apply second approach .
Do the code below in the Page Load event of result.aspx page

```
protected void Page_Load(object sender, EventArgs e)
{
    TextBox tx1 = (TextBox)PreviousPage.FindControl("TextBox1");
    TextBox tx2 = (TextBox)PreviousPage.FindControl("TextBox2");

    Response.Write("ID : " + tx1.Text + "<br/>");
    Response.Write("Password : " + tx2.Text + "<br/>");

}
```

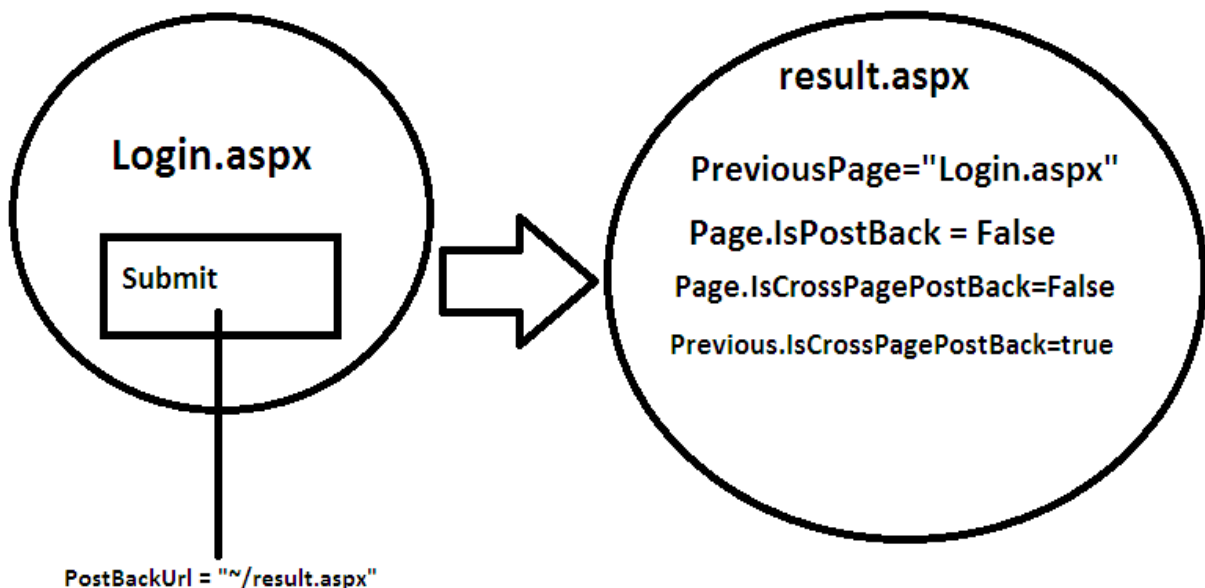
It will also give the same output as above.
Here something different is going on :

Let us discuss every line in deep

```
TextBox tx1 = (TextBox)PreviousPage.FindControl("TextBox1");
```

In cross-page posting , Previous page property of target page holds the reference of Previous page or source page from where target page is posted by button with PostBackUrl property.

See the image below:



We know that result.aspx page is requested as Http Post but still it's page property called IsPostBack is still false because result.aspx page is not posted back . So it's previous page is cross-page postback on it. Therefore Previous.IsCrossPagePostBack is true.

So it is clear that PreviousPage property of result.aspx page holds the reference of Login.aspx page so we can easily access it and can also access it's control by FindControl() method.

We need to pass the Id of previous page controls as parameter into FindControl() method.

Activity :

Create two web pages

One for allowing user to accept the details as given below :

Background Color	ActiveBorder
TextBox Fore Color	ActiveBorder
TextBox Back Color	ActiveBorder
Button Back Color	ActiveBorder

Submit

Let us see it's source code for details.aspx page:

[illegible]

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

Button 's PostBackUrl property is “~/login.aspx” page .

Now we need to display the colors name in all DropDownList , but problem is how many colors and manually writing them for all DropDownList controls ?

.Net also simplifies this problem by allowing you to access .net **Enums** for colors under namespace “**System.Drawing**”

Let us see the below details.aspx.cs code

```
protected void Page_Load(object sender, EventArgs e)
{
    string[] colors= Enum.GetNames(typeof(KnownColor));
    DropDownList1.DataSource = colors;
    DropDownList2.DataSource = colors;
    DropDownList3.DataSource = colors;
    DropDownList4.DataSource = colors;

    DropDownList1.DataBind();
    DropDownList2.DataBind();
    DropDownList3.DataBind();
    DropDownList4.DataBind();
}
```

It seems very easy and in very less code we achieved it . Let us discuss the code in deep :

```
string[] colors= Enum.GetNames(typeof(KnownColor));
```

here we are retrieving all the enum values by Enum.GetNames() with has parameter of typeof for enum name .KnownColor will fetch all the available color definition in .Net under namespace System.Drawing .All the retrieved colors will be stored in string based array colors.

Now as user will submit details.aspx page then all controls with values will be posted to Login.aspx.

Note : But there is one problem with sending the details like selected value of bound DropDownList1 from source to target page.

So to avoid such problems , we will not bind the DropDownList to data source array . We will add items programmatically by adding items from array to DropDownList.

Let us replace the above code of page Load event by below code :

```
protected void Page_Load(object sender, EventArgs e)
{
    string[] colors= Enum.GetNames(typeof(KnownColor));

    foreach (string s in colors)
        DropDownList1.Items.Add(s);

    foreach (string s in colors)
        DropDownList2.Items.Add(s);

    foreach (string s in colors)
        DropDownList3.Items.Add(s);

    foreach (string s in colors)
        DropDownList4.Items.Add(s);
}
```

Now our next task it to apply the selected color in dropdownlist into the login.aspx page respectively .

For E.G If user has selected background color as green in dropDownList then it will be applied to the Login page background .

Let us see the code of Login.aspx page , where I have done little changes :

```
protected void Page_Load(object sender, EventArgs e)
{
    if (PreviousPage != null)
    {
        DropDownList d1 = (DropDownList)(PreviousPage.FindControl("DropDownList1"));
        DropDownList d2 = (DropDownList)(PreviousPage.FindControl("DropDownList2"));
        DropDownList d3 = (DropDownList)(PreviousPage.FindControl("DropDownList3"));
        DropDownList d4 = (DropDownList)(PreviousPage.FindControl("DropDownList4"));

        div1.Style.Add(HtmlTextWriterStyle.BackgroundColor, d1.SelectedValue);

        TextBox1.ForeColor = System.Drawing.Color.FromName(d2.SelectedValue);
        TextBox2.ForeColor = System.Drawing.Color.FromName(d2.SelectedValue);

        TextBox1.BackColor = System.Drawing.Color.FromName(d3.SelectedValue);
        TextBox2.BackColor = System.Drawing.Color.FromName(d3.SelectedValue);

        Button1.BackColor = System.Drawing.Color.FromName(d4.SelectedValue);
    }
}
```

We have first checked whether its previous page is available or not means Login page is redirected by previous page or directly from user . So these changes will be applied only when Login page will be redirected by details.aspx page else no changes will be done .

C. Redirecting Programmatically Using the Browser :

You can redirect users to another page using the capabilities of the user's browser. In a browser redirect, the browser issues a new request to the target server in the form of an HTTP GET request.

You can trigger the redirect programmatically in client script or server code.

In client script, you can call the form.submit method, provided the <form> element's method attribute value is get. In that case, if the current page contains form data, it is passed to the target server by appending it as a query string onto the requested URL. By default when we click on the submit button (input type="submit") then page is posted as per the method used (E.G. Get or Post) in form tag to the target page on server . Otherwise you can also submit it by using java script by using submit() function

For E.G we need to allow user to accept id and password , and want to validate data first before it's submission to server , so we will use button control like <input type="button"> which does not have submitting capability so , we can call any java script function which will validate the input fields and then submit the page .

Let us see the code for Login page which takes asp text boxes and one HTML button .

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
    <script type="text/javascript">

        function submit_by_id()
        {

            if (validation()) // Calling validation function
            {
                form1.submit();
            }
        }

        function validation()
        {
            var name = document.getElementById("TextBox1").value;
            var email = document.getElementById("TextBox2").value;
            var emailReg = /^[a-zA-Z0-9_-]+@[a-zA-Z0-9_-]+\.[a-zA-Z]{2,4}$/;
            if (name === '' || email === '')
            {
                alert("Please fill all fields...!!!!!!");
                return false;
            }
            else if
            (!(email).match(emailReg))
            {
                alert("Invalid Email...!!!!!!");
                return false;
            }
        }
    }
</script>
</head>
<body>
    <div>
        <input type="text" id="TextBox1" value="Name" />
        <input type="text" id="TextBox2" value="Email" />
        <input type="button" value="Submit" />
    </div>
</body>
</html>
```

```

        else
        {
            return true;
        }
    }

</script>
</head>
<body>
    <form id="form1" runat="server" action="result.aspx" method="get">
    <div id="div1" runat="server">
        User ID : <br />
        <asp:TextBox ID="TextBox1" runat="server" EnableViewState="False"></asp:TextBox>
        <br />
        <br />
        Email ID : <br />
        <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox>
        <br />
        <br />
        <input type="button" value="Submit" onclick="submit_by_id()" />

        <br />
        <br />
        <br />

    </div>
    </form>
</body>
</html>

```

In the above code we have used javascript functions validation() and submit_by_id() for validation and submission based on the validation return of true or false, if validation() function will return true that page will be submitted else not . Validation() function will return false when either fields are blank or email id is not right . If validation() function will return true then page will be submitted and the form details will be posted to the action="page" E.G. result.aspx with method as given in <form > tag above.

Rest see the below code for displaying the submitted data to result.aspx from Login page .

```

protected void Page_Load(object sender, EventArgs e)
{
    //In case of GET
    string eid = Request.QueryString["TextBox1"].ToString();

    string passwd = Request.QueryString["TextBox2"].ToString();

    //In case of POST
    //string eid = Request.Form["id"].ToString();

    //string passwd = Request.Form["password"].ToString();

    Response.Write("<h1>Welcome User , Your Registration is Done below are the details</h1>
    <br/>");
    Response.Write(" <br/>");
    Response.Write(" <b> ID : " + eid + " </b> <br/>");
    Response.Write(" <b> Email ID : " + passwd + " </b> <br/><br/>");
    Response.Write("It is <b>" + Request.HttpMethod + " </b> Request from user ");
}

```

In server code, you can programmatic-ally redirect by calling the Redirect method. The method sends a command to the user's browser that **causes the browser to issue an HTTP GET command** for the target page. Calling the server Redirect method is the programmatic equivalent of clicking a hyperlink, in that it results in a new request for the target page. Because you are calling the methods from your own code, you can dynamically define the target URL, including any query-string information. If the source and target pages are in the same Web application, you can share data between the source and target pages by adding server code to store the data in session state.

This is somewhere same as we have done with hyperlink above , but provides you an opportunity to add the query-string dynamically by taking values from input fields.

Let us we know in deep

Below is the Login.aspx page code :

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
        <div id="div1" runat="server">
            User ID : <br />
            <asp:TextBox ID="TextBox1" runat="server" EnableViewState="False"></asp:TextBox>
            <br />
            <br />
            Password : <br />
            <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Submit" onclick="Button1_Click" />

            <br />
            <br />
            <br />
        </div>
    </form>
</body>
</html>
```

Now we need to pass the above given information from Login page to result page , so that we can display to user as confirmation .

We need to use Response Class with it's method Redirect as shown below:

```
protected void Button1_Click(object sender, EventArgs e)
{
    string url = "~/result.aspx?id=" + TextBox1.Text + "&password=" + TextBox2.Text;
    Response.Redirect(url);
}
```

Be clear about making the URL for redirection , which will consist of destination page address and also the query string after “?” followed by unique name like id and password with values taken from the

input fields.

Let us check whether the target page is requested by GET or POST by adding one more line into the load event of the result.aspx page and will also retrieve data from query-string as given below :

```
protected void Page_Load(object sender, EventArgs e)
{

    string eid = Request.QueryString["id"].ToString();

    string passwd = Request.QueryString["password"].ToString();

    Response.Write("<h1>Welcome User , Your Registration is Done below are the details</h1>
<br/>");
    Response.Write(" <br/>");
    Response.Write(" <b> ID : " + eid + " </b> <br/>");
    Response.Write(" <b> PASSWORD : " + passwd + " </b> <br/>");
    Response.Write("It is <b>" + Request.HttpMethod + " </b> Request from user ");

}
```

Below is the output from user when user gives id and password in Login Page and submit the button:

Welcome User , Your Registration is Done below are the details

ID : abc@gmail.com
PASSWORD : p@ssw0rd1234

It is GET Request from user

D. Redirecting Pro-grammatically on the Server

You can also redirect pro-grammatically to a target page on the server by calling the Transfer method. In this scenario, the server simply transfers the current Web Forms source page context to the target page. The target page then renders in place of the source page. **The source and target pages must be in the same Web application. As with cross-page posting, the Transfer method has the advantage that it enables the target page to read control values and public property values from the source page.**

Because the transfer between source and target pages happens on the server, the **browser has no information about the changed page**, and it retains information about the original (source) URL.

For example, the Address box in Internet Explorer does not change after a transfer, and instead continues to show the URL of the page it most recently requested (which is usually the source page). The browser's history is not updated to reflect the transfer. This can result in unexpected behavior if the user refreshes the page in the browser or clicks the browser's back button. Therefore, calling the Transfer method is a strategy that is best used in applications where you are presenting pages to the

user with the URL hidden.

Let us see the simple code of Login.aspx page, which has simple HTML codes.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>

</head>
<body>
    <form id="form1" runat="server">
        <div id="div1" runat="server">
            First Name : <br />
            <asp:TextBox ID="TextBox1" runat="server" ></asp:TextBox>
            <br />
            <br />
            Last Name : <br />
            <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Button"
                onclick="Button1_Click" />

            <br />
            <br />
            <br />

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

OnClick event-handler do the below code :

```
protected void Button1_Click(object sender, EventArgs e)
{
    Server.Transfer("~/result.aspx");
}
```

Now see the very interesting thing about the submission of button and output ,

As you will submit the page, page will be taken to result.aspx from server without actual information to the browser. So browser will not be aware of server redirection due to which browser still shows the previous url or source url which can give unexpected error if in case of refreshing the page.

Note : In case of Server.Transfer() also source page is posted to target page as HTTP Post so that target page can also access the source page controls as in case of cross-page postback.

(I) Getting Public Property Values from the Source Page

Here we are going to retrieve the public properties defined in source page by target page .

See the details Page where user need to give first name and last name and these information will be

used by properties that we can access by target page .

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>

</head>
<body>
    <form id="form1" runat="server">
        <div id="div1" runat="server">
            First Name : <br />
            <asp:TextBox ID="TextBox1" runat="server" ></asp:TextBox>
            <br />
            <br />
            Last Name : <br />
            <asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
            <br />
            <br />
            <asp:Button ID="Button1" runat="server" Text="Button"
                onclick="Button1_Click" />

            <br />
            <br />
            <br />

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

As user will click on button , we will take user to result.aspx page where we will access the property defined in source page . **Let us see the details.aspx.cs page**

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }

    protected void Button1_Click(object sender, EventArgs e)
    {
        Server.Transfer("~/result.aspx");
    }

    public string FullName
    {
        get
        {
            return "Dear " + TextBox1.Text.ToUpper() + " " + TextBox2.Text.ToUpper();
        }
    }
}
```

We have defined a property name FullName which is returning the concat of textbox1.text and textbox2.text with “Dear” .

Now before we code in result.aspx page for accessing the Property FullName , we need to tell result.aspx page that it's previous page is Details.aspx

for that we need to add one declarative tag called `<%@ PreviousPageType %>` at page level in result.aspx page below page declaration :

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="result.aspx.cs" Inherits="result" %>
```

```
<%@ PreviousPageType VirtualPath="~/Default.aspx" %> // to be added
```

This tag holds the reference of previous page which allows us to access the public property FullName easily by using Previous Page Property of current page as in cross-page posting . Like the code below :

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("<h1>Welcome User , Your Full Name is Given below : </h1> <br/>");
    string name = PreviousPage.FullName;
    Response.Write(" <b> Full Name : " + name + " </b> <br/>");
}
```

(II)Getting Control Information from the Source Page in the Same Application

If the source page and target page are both ASP.NET Web Forms pages in the same Web application, and if you transfer execution from the source page to the target page on the server by using the Transfer method, you can read the values of controls on the source page while in the target page. You might use this strategy if the source page does not expose public properties containing the information you need.

To get the values of controls from the source page in the same application

On the target page, get a reference to the source page by using the target page's PreviousPage property, and then call the FindControl method to get a reference to the control you want.

The following code example gets the value of the source page's TextBox1 control and displays it in the control named Label1:

```
if (PreviousPage != null)
{
    TextBox SourceTextBox =
        (TextBox) PreviousPage.FindControl("TextBox1");
    if (SourceTextBox != null)
    {
        Label1.Text = SourceTextBox.Text;
    }
}
```

It is the same process as we have discussed in cross-page post back .

In this example we need to not add declarative tag as in accessing the public properties .

Do the single line code on button click event handler in details.aspx

```
protected void Button1_Click(object sender, EventArgs e)
{
    Server.Transfer("~/result.aspx");
}
```

We need to code also in result.aspx.cs page as given below :

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("<h1>Welcome User , Your Full Name is Given below : </h1> <br/>");

    TextBox t1 = (TextBox)PreviousPage.FindControl("TextBox1");
    TextBox t2 = (TextBox)PreviousPage.FindControl("TextBox2");

    Response.Write(" Full Name : <b> Mr. " + t1.Text.ToUpper() + " " + t2.Text.ToUpper()
+ " </b> <br/>");
}
```

It's nice working with Server.Transfer() along with its importance and usage. Let us see every option in depth by the below table so that you can choose as required:

Selecting a Redirect Option

Strategy	Characteristics	Usage
Hyperlinks	<ul style="list-style-type: none"> Performs new request on the target page. Does not pass current page information to the target page. Requires user initiation. Redirects to any page, not just pages in the same Web application. Enables you to share information between pages using query string or session state. (The HyperLink control enables you to create URL and query strings programmatically.) 	<ul style="list-style-type: none"> For navigation without any additional processing, as in menus or lists of links. When navigation should be under user control.
Cross-page posting	<ul style="list-style-type: none"> Posts current page information to the target page. Makes post information available in the target page. Requires user initiation. Redirects to any page, not just pages in the same Web application. Enables the target page to read public properties of the source page if the pages are both Web Forms pages and are in the same Web application. 	<ul style="list-style-type: none"> To pass current page information to the target page (as in multi-page forms). When navigation should be under user control.
Browser redirect	<ul style="list-style-type: none"> Performs a new HTTP GET request on the target page. Passes the query string (if any) to the target page. (The browser might limit the size of the query string.) Provides programmatic and dynamic control over the target URL and query string. Enables you to redirect to any page, not just pages in the same Web application. Enables you to share information between source and target pages using session state. 	<ul style="list-style-type: none"> For conditional navigation, when you want to control the target URL and control when navigation takes place. For example, use this option if the application must determine which page to navigate to based on data provided by the user.
Server transfer	<ul style="list-style-type: none"> Transfers control to a new page that renders in place of the source page. Redirects only to target pages that are in the same Web application as the source page. Enables you to read values and public properties from source page. Does not update browser information with information about the target page. Pressing the refresh or back buttons in the browser can result in unexpected behavior. 	<ul style="list-style-type: none"> For conditional navigation, when you want to control when navigation takes place and you want access to the context of the source page. Best used in situations where the URL is hidden from the user.

How to: Determine How ASP.NET Web Forms Were Invoked

There is a very good question that should come in your mind that there are many ways to redirect and submit details in ASP.NET but how will we check the how web form is invoked let us see the table below:

It is often useful to know how an ASP.NET Web page was invoked: whether by an original request (an HTTP GET), a post back (an HTTP POST), a cross-page post from another page (an HTTP POST), or a transfer from another page using the Transfer method or using a callback from the browser. The Page class exposes a set of properties that you can use to determine how a page was invoked.

To determine how an ASP.NET Web page was invoked

- Examine the values of the following Page class properties, and then refer the table to determine how the page was invoked:
 - [IsPostBack](#)
 - [PreviousPage](#)
 - [IsCrossPagePostBack](#)
 - [IsCallback](#)

The following table lists ways in which a page can be invoked and the corresponding Page property values.

Invocation method	Property values
Original request	<ul style="list-style-type: none">◊ IsPostBack is set to false.◊ PreviousPage is set to null (Nothing in Visual Basic).◊ IsCallback is set to false.
Postback	<ul style="list-style-type: none">◊ IsPostBack is set to true.◊ PreviousPage is set to null (Nothing in Visual Basic).◊ IsCallback is set to false.
Cross-page posting	<ul style="list-style-type: none">◊ IsPostBack is set to false.◊ PreviousPage references the source page.◊ IsCrossPagePostBack is set to true.◊ IsCallback is set to false.
Server transfer	<ul style="list-style-type: none">◊ IsPostBack is set to false.◊ PreviousPage references the source page.◊ IsCrossPagePostBack that is referenced in the PreviousPage is set to false.◊ IsCallback is set to false.
Callback	<ul style="list-style-type: none">◊ IsPostBack is set to false.◊ PreviousPage is set to null (Nothing in Visual Basic).◊ IsCallback is set to true.

Note : We have discussed about IsPostBack,Previous Page ,IsCrossPagePostBack but not about IsCallback . Callback() method is used in ASP.NET for requesting server for any details on the web page without submitting the page . ASP.NET AJAX works in the same methodology of asynchronous communication with server. You can very easily update your web page without any submission from client side by this method . For checking that page is requested or not by using callback() method , we use IsCallback property of page .