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.

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 JavaScipt function to the rendered HTML Page. This function is named \_doPostBack() . When Called, it triggers a PostBack, sending data back to the web Server.

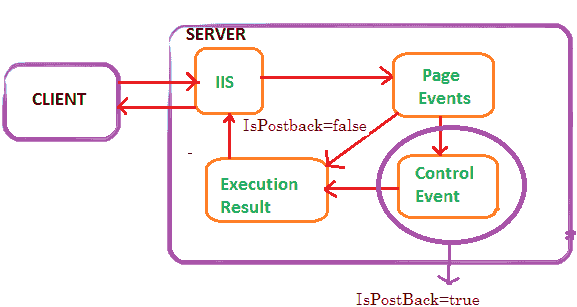
Life Cycle of a Web Page:

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.

**IsPostBack**

**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.



**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.

C# Source Code

protected void Page\_Load(object sender, EventArgs e)

{

if (!IsPostBack)

{

// generate form;

}

else

{

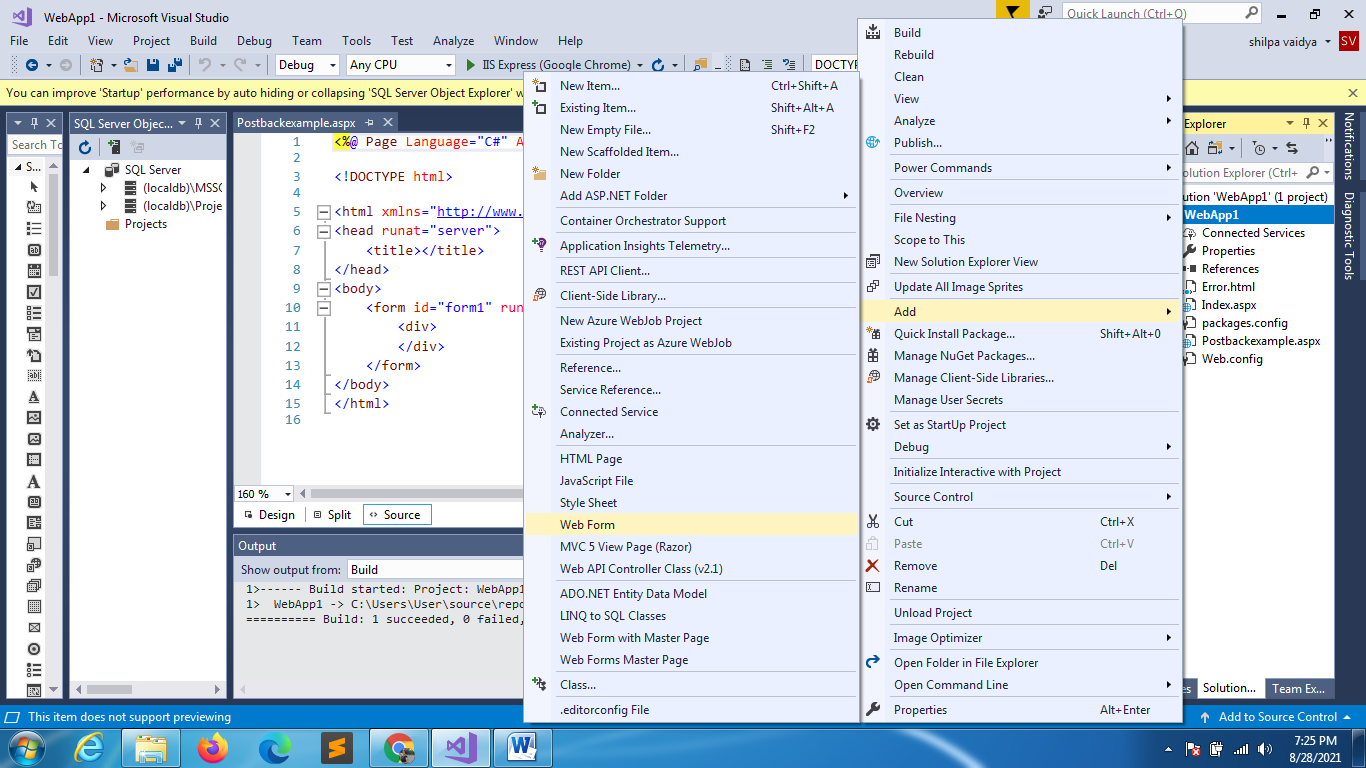
//process submitted data;

}

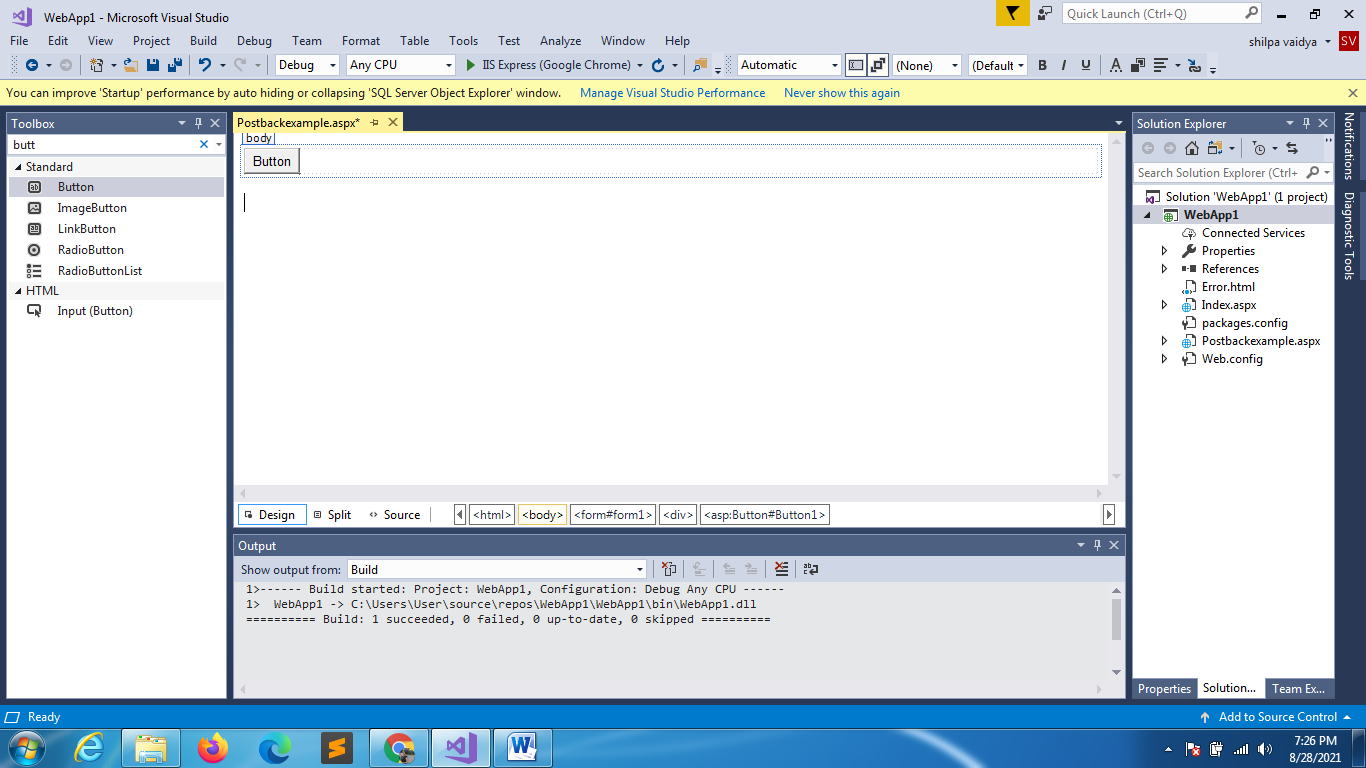
}

Example:

Create a webform ->



DRAG AND DROP BUTTON



using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WebApp1

{

    public partial class Postbackexample : System.Web.UI.Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        {

        }

        protected void Button1\_Click(object sender, EventArgs e)

        {

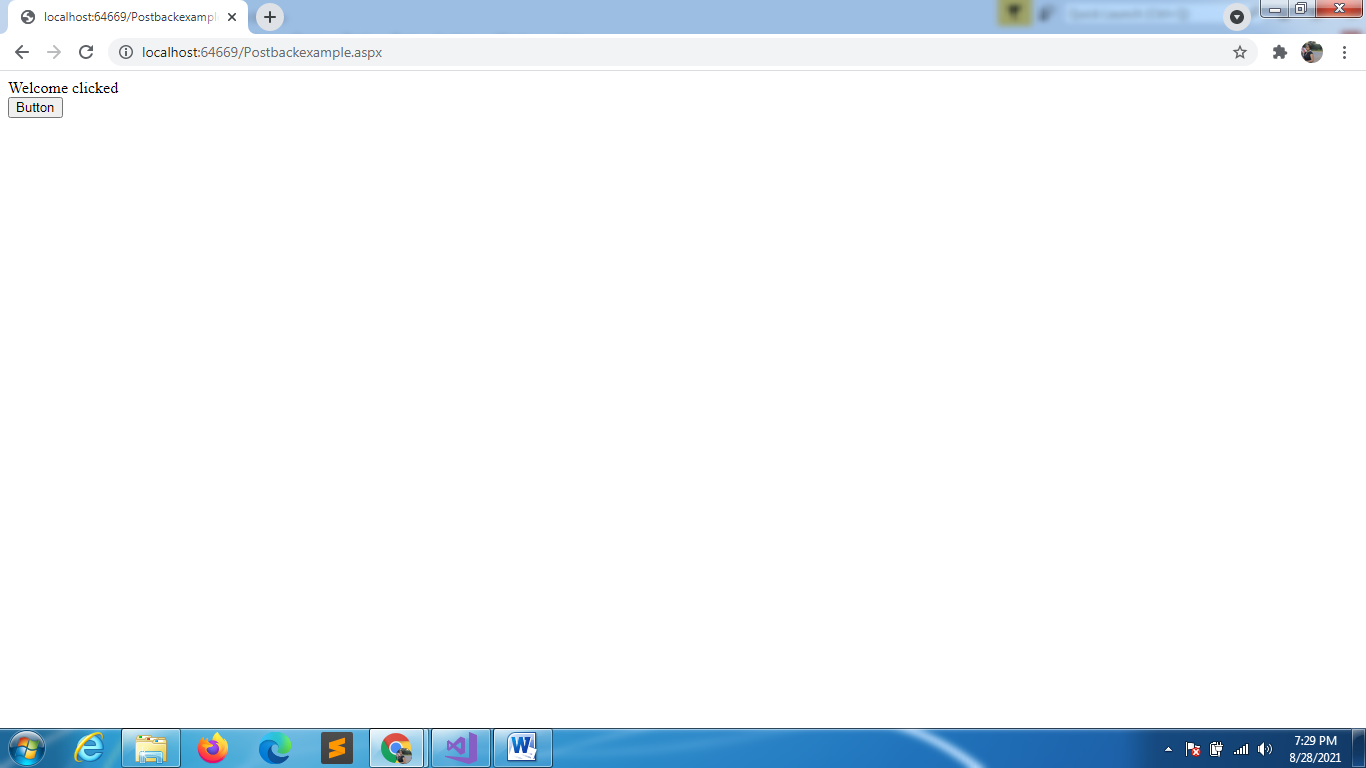
            Response.Write("Welcome clicked");

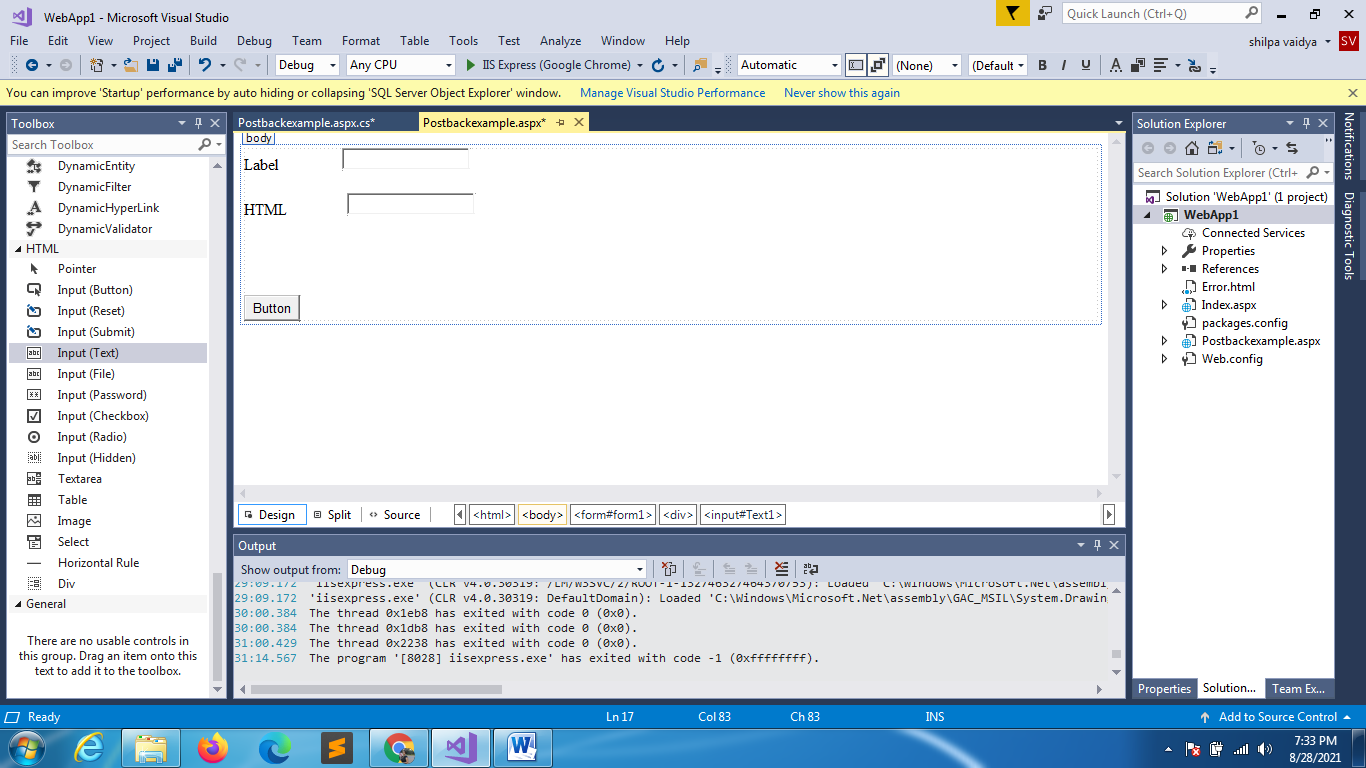
        }

    }

}

When you run



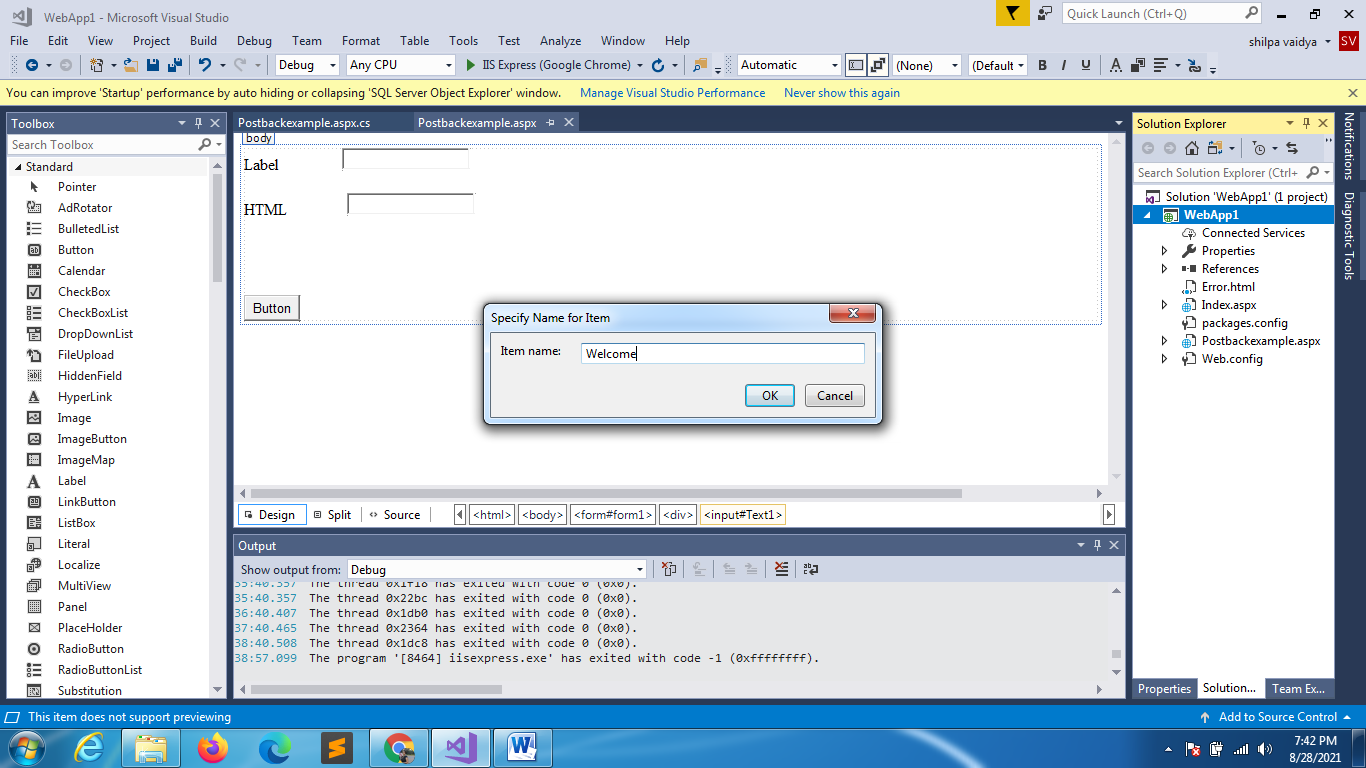


WHEN YOU RUN DUE TO INTERNAL STATE MANAGEMENT YOU WILL NOT SEE TEXT GETTING CLEARED IN ASP.NET CONTROLS BUT YOU WILL SEE PLAIN HML INPUT BOX CLEARS AFTER BUTTON CLICKED

HTML TEXTBOX

STANDARD TEXTBOX

CREATE ANOTHER WEBFORM



<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Welcome.aspx.cs" Inherits="WebApp1.Welcome" %>

<!DOCTYPE html>

<html xmlns="http://www.w3.org/1999/xhtml">

<head runat="server">

    <title></title>

</head>

<body>

    <form id="form1" runat="server">

        <div>

            <h1>Hi how are you!!! this is WELCOME page</h1>

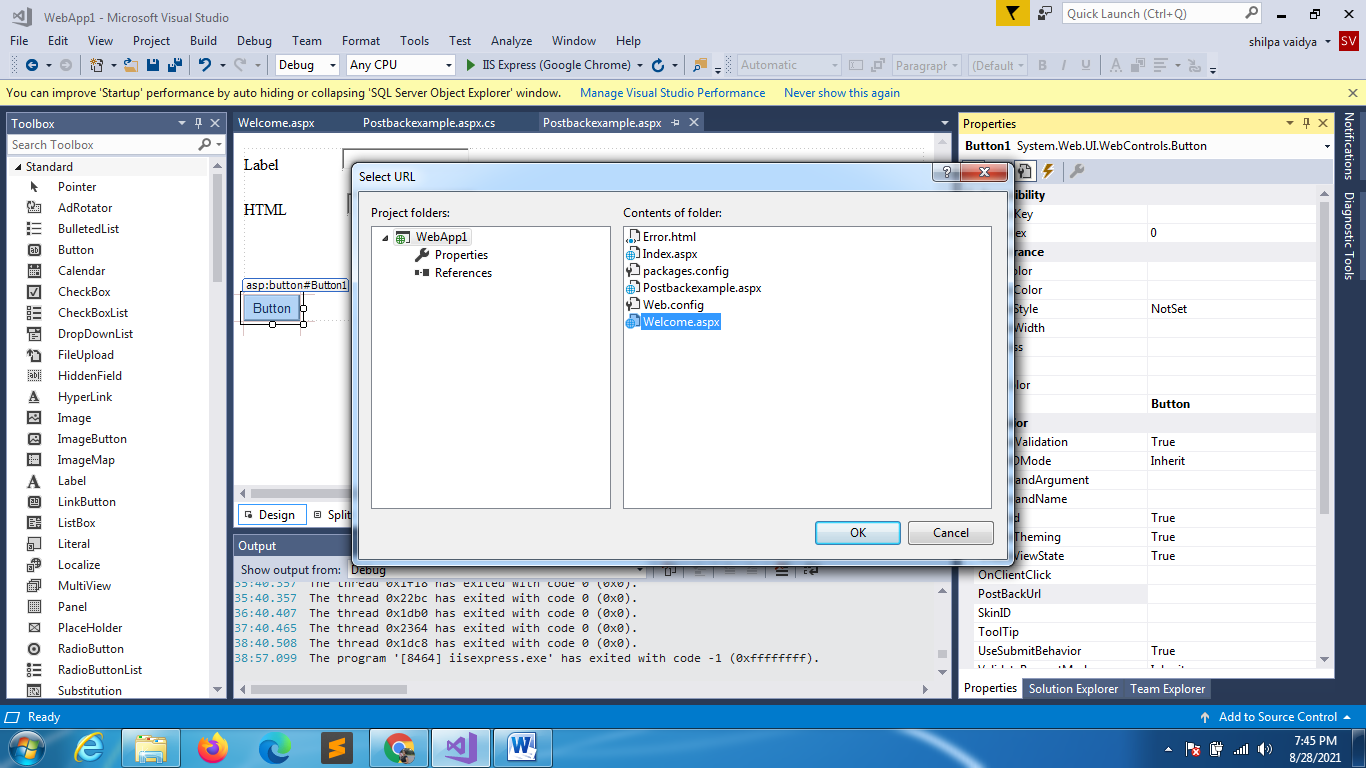
        </div>

    </form>

</body>

</html>

In the previous webpage (index.aspx)



Run application

# HTTP Methods GET vs POST

HTTP defines a set of **request methods** to indicate the desired action to be performed for a given resource. GET and POST are two different types of **HTTP requests** . If the method is not specified in the html form GET will be used by default.

## HTTP GET

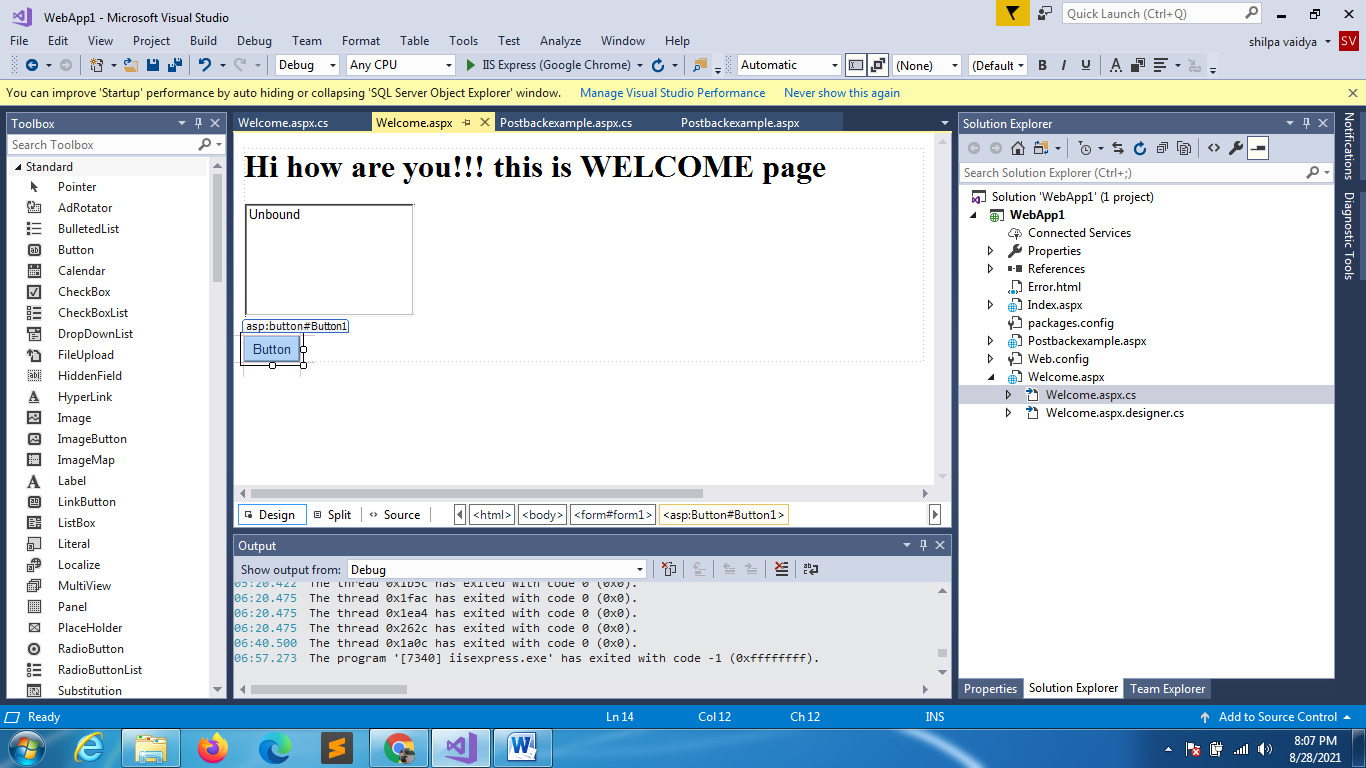
GET method places name value pairs as plain text in URL. **GET requests** are often cached by default by some browsers if you are not careful. It can send but the parameter data is limited to what we can stuff into the request line (URL). Safest to use less than 2K of parameters, some servers handle up to 64K. Also, **GET** is less secure compared to POST because data sent is part of the URL. So it's saved in browser history and server logs in **plaintext** . Finally, GET method is visible to everyone (it will be displayed in the browser's address bar) and has limits on the amount of information to send.

## HTTP POST

POST requests supply additional data from the client (browser) to the server in the message body. It **submits data** to be processed (e.g., from an HTML form) to the identified resource. The data is included in the body of the request. This may result in the creation of a **new resource** or the updates of existing resources or both. In the case of security, **POST** is a little safer than GET because the parameters are not stored in browser history or in web server logs. **POST method** variables are not displayed in the URL and cannot be cached. With POST you can also do multipart **mime encoding** which means you can attach files as well. Also if you are using post variables across navigation of pages, the user will get a warning asking if they want to resubmit the post parameter.

Ispostback Example

Add listbox



using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WebApp1

{

With postback property it adds same elements multiple times

    public partial class Welcome : System.Web.UI.Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        {

            ListBox1.Items.Add("Apple");

            ListBox1.Items.Add("Mango");

            ListBox1.Items.Add("Grapes");

        }

    }

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

namespace WebApp1

{

    public partial class Welcome : System.Web.UI.Page

    {

        protected void Page\_Load(object sender, EventArgs e)

        {

            if(!Page.IsPostBack)

            {

                ListBox1.Items.Add("Apple");

                ListBox1.Items.Add("Mango");

                ListBox1.Items.Add("Grapes");

            }

        }

    }

}