**Lab 5 – CS 3340**

To make this document easier to read, it is recommend that you turn off spell checking and grammar checking in Word:

1. Choose: File, Option, Proofing
2. At the very bottom, check: “Hide spelling errors…” and “Hide grammar errors…”

This lab will take 1-1.5 hours if you do it properly.

**Lab Objectives**

1. Understand how postback works with a single page
2. Understand how session memory works with a single page
3. Understand how postback works when you leave a page and return
4. Understand how to use session memory across multiple pages.
5. Use a query string to pass information between pages

**Lab Organization**

There are 6 stages to complete Lab 5.

|  |  |
| --- | --- |
| Stage | Description |
| 1 | Understanding Postback |
| 2 | Understanding Session Memory |
| 3 | Understanding Postback, Multiple Pages |
| 4 | Using Session Memory across Multiple Pages |
| 5 | Using a QueryString to Pass Information Between Pages |
| 6 | Package Assignment for Submission |

Follow the directions below to complete the lab.

**Stage 0 – Understanding Postback**

(**Optional**) Watch the 19 minute video, [Submit & Postback](https://learn.microsoft.com/en-us/shows/asp-net-site-videos/submit-postback):

This video will introduce you the what goes on "under the covers" when your ASP.NET application’s browser and server interact.

This is a very well-done video that touches on information from the HTML Forms tutorial you went through and some of the things I’ve briefly mentioned in class. I encourage you to watch this video, but it is not required.

**Stage 1 – Understanding Postback**

(Read, no action required) There is a difference between the “first” time a page is requested and a *postback*.

1. First time – The user types in a URL, request is sent to the server, the page is located, run, and the resulting HTML is sent to the browser. The first time a page is accessed, a *Session* object is created on the server which represents a particular computer (IP address) using the application (might be one page or many in an application). This object is alive until the user stops interacting with the page (*i.e.* a timeout), or, the session object can be terminated programmatically (*e.g.* when a user logs out of a page, the code typically terminates the session object).
2. Postback – Next, the user interacts with the page and eventually causes the page to be posted-back to the server (*e.g.* user presses a submit button). The session object knows that a *postback* has occurred (as opposed to the first time on the page). This means that all the form data (*e.g.* value typed in a text box, item selected in a dropdown, *etc*) is sent back to the server to the same page on the server. In .NET, you always postback to the same page you are viewing.
3. Postback – Postbacks continue until (a) server-code transfers control to another page, (b) the user navigates to another page (or logs out), (c) or the session times-out due to inactivity for a certain amount of time.

Here, we will explore the difference between the “first time” on a page and a “postback”. We will also observe that the contents of controls on a web page are persisted between postbacks.

1. Create your *lab05* project.



1. Add a web form named, *postback.*
2. Add a *ListBox* and set the *ID* to *lbxNames.* Stretch it extra tall.
3. Add a *Button* beside it and set the *ID* to *btnDisplay*
4. Add a *TextBox* below and set: (a) *ID* to *txtMessage*, (b) *TextMode* to *MultiLine* and then stretch it larger.
5. Add this code to *Page\_Load*

lbxNames.Items.Add(new ListItem("Dave", "11"));

lbxNames.Items.Add(new ListItem("Paul", "44"));

lbxNames.Items.Add(new ListItem("Anna", "3"));

1. Run (the button is not active yet) and observe that the listbox displays the names.



1. Create an event handler for the button’s click event which displays the text and value of the selected item from the listbox in the multiline text box.
2. Do the following:
3. Run the page
4. Select a name and press the button.
5. Again, select a name and press the button. Note that every time you press the button, the three names are appended onto the existing list in the listbox.
6. (Read, no action required) Why are items added to the listbox every time you press the button?
7. **Important Point**: **The values in a control (*e.g.* the Text and Values in the listbox) persist between postbacks.**
8. Remember that every time the button is pressed, the page is posted-back to the server where two things occur:
* Page\_Load is executed, which adds the three names to the listbox (on top of what is already there)
* btnDisplay\_Click is executed

Thus, each time *Page\_Load* executes it adds to the listbox.

1. In web programming, a *session* begins when a user requests a page, the page is delivered, the user interacts with the page, posting back to the server over and over until the user has completed the task they set out to do. At this point the session is considered over (either explicitly by the user logging out, or the user navigating to another site, or leaving the computer altogether, in which case, the session times out.
2. It is frequently useful for the programmer to be able to detect whether it is the first request for a page (*i.e.* the session is beginning), or the request for a page is a *postback* (*i.e.* the session is onoing).
3. Every time a page is requested (either the first time, or a post back), an instance of the page is created in memory on the server. In other words, an instance of the code-behind file is created. The instance of the page itself can be referenced by *this* (the same as Java), or with the more common alias, *Page*. The page class has a property, *IsPostBack* that returns *true* if a post-back occurs or *false* if it is the first time on the page. To detect whether it is the first time on the page or a postback we can use code like this:

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

 if (!Page.IsPostBack) {

 // First time on page

 }

 else {

 // Postback

 }

}

Thus, to correct this problem, we only want to build the listbox the first time we visit the page, not on postbacks.

1. Fix the problem above so that the page works correctly.

**Stage 2 – Understanding Session Memory**

You are going to write a page that allows the user to enter a number and press an “Add” button, repeatedly. Each time the button is pressed, the current number is added to the sum of the previous numbers and then the new sum is displayed. To accomplish this, we need to “remember” the previous sum between postbacks. The solution we illustrate is to use *session memory*.



1. Add a web form named, *session1.*
2. Add a *TextBox* and set the *ID* to *txtNum.*
3. Add a *Button* beside it and set the (a) *ID* to *btnAdd,* (b) *Text* to “Add”
4. Add a *TextBox* below and set: (a) *ID* to *txtMessage*, (b) *TextMode* to *MultiLine* and stretch it larger.
5. Do the following:
6. Add a click event-handler for the button (empty, we’ll add the code shortly).
7. Add an instance variable to the page, *sum* which will hold the sum of the numbers entered.
8. Add code to the click event to add the current number to the sum and display the new sum.

public partial class session1 : System.Web.UI.Page {

 int sum;

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

 }

 protected void bthAdd\_Click(object sender, EventArgs e) {

 int curNum = Convert.ToInt32(txtNum.Text);

 sum += curNum;

 txtMessage.Text += "Current Num=" + curNum + ", Sum=" + sum + "\n";

 txtNum.Focus();

 txtNum.Attributes["onfocus"] = "this.select();";

 }

}

1. Run, type a number in, Enter, repeat several times. Notice that it does not work correctly.
2. (Read, no action required) **Important Points:**
3. When a request for a page is made (either the first time or a post back), as stated earlier, an instance of the page is created. After the code is executed, and the output is converted to HTML and sent back to the browser, the page object is destroyed. Thus, all instance variables defined in the page are destroyed when the page is sent back to the browser. Thus, the *sum* is not remembered across postbacks.
4. The internet programming paradigm is *stateless*, *e.g.* all variable values are destroyed when the page is sent back to the browser. However, we can explicitly tell the page to store any variables we want in *Session* memory. *Session* is memory that is allocated on the Server which can be used to store information between postbacks. Thus, when the page posts back to the server we can program the page to retrieve any variables we have stored in Session.
5. To put a *value* in Session, we associate it with a *key* which we use to retrieve it later. For example, to put a *value* in Session:

Session.Add("key", value);

And to retrieve it later:

*Type* value = (*Type*)Session["key"];

or

*Type* value = Convert.*ToType*(Session["key"]);

Note that values stored in Session are stored as Objects so we must cast (or convert) them when we retrieve them.

1. Next, we fix the problem above using session memory. Do the following:
2. Add this code to *Page\_Load*

if (!Page.IsPostBack) {

 Session.Add("Sum", 0);

}

else {

 sum = Convert.ToInt32(Session["Sum"]);

}

Note: the first time the page is accessed, we put “Sum” into session, initializing it to 0. On postbacks, we retrieve the value of “Sum” and store it in the instance variable, *sum*.

1. Add this line to the end of the click event:

Session.Add("Sum", sum);

**Important Point**: Anytime you change a variable that you want stored in Session you should immediately store it in Session. You will forget to do this and it will haunt you!

1. Run the page and verify that the cumulative sum of the numbers entered is correct.

**Stage 3 – Understanding Postback, Multiple Pages**

In this example we will see how “first time” (*e.g. !Page.IsPostBack*) changes when we navigate to a different page in our website, and then later return.

1. Do the following:

**NOTE: If something goes wrong with this step, close the page, delete it and start this step over. If you mess up the copy/paste, something will go wrong. Don’t panic, just do it over.**

1. Add a web form named, *pback1.*
2. Open the page in Source mode and replace everything except the page directive (the first line) with:

<!DOCTYPE html>

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

<head runat="server">

 <title></title>

</head>

<body>

 <form id="form1" runat="server">

 <div>

 <p><strong>Postback Multipage Example</strong></p>

 <p>This page illustrates when it is the first time on the page or postback and what happens when you navigate to a new page and then back. You should look carefully at the code for this page and for &quot;Page 2&quot; (pback2.aspx).</p>

 <p>

 <asp:Button ID="btnSubmit" runat="server" onclick="btnSubmit\_Click" Text="Submit/Postback" />&nbsp;

 <asp:Button ID="btnLoadPage" runat="server" onclick="btnLoadPage\_Click" Text="Reload" />&nbsp;

 <asp:Button ID="btnPage2" runat="server" onclick="btnPage2\_Click" Text="Page 2" />

 </p>

 <p>

 <asp:TextBox ID="txtMessage" runat="server" EnableViewState="False"

 Height="224px" TextMode="MultiLine" Width="384px"></asp:TextBox>

 </p>

 </div>

 </form>

</body>

</html>

1. Open the code behind file, *pback1.aspx.cs* and replace the class (everything between the *namespace* line and the last “}”) with the code below. Study the code above carefully

public partial class pback1 : System.Web.UI.Page {

 string newLine = Environment.NewLine;

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

 // First time on page

 if (!Page.IsPostBack) {

 txtMessage.Text += newLine + "\*\*\*First time on page" + newLine;

 }

 // Postback

 else {

 txtMessage.Text += newLine + "\*\*\*Post-back" + newLine;

 }

 }

 protected void btnSubmit\_Click(object sender, System.EventArgs e) {

 txtMessage.Text += "\*\*\*Submit callled" + newLine;

 }

 protected void btnLoadPage\_Click(object sender, EventArgs e) {

 // Reload the page.

 Response.Redirect(Request.RawUrl);

 }

 protected void btnPage2\_Click(object sender, EventArgs e) {

 // Go to page 2.

 Response.Redirect("pback2.aspx");

 }

}

1. Do the following:
2. Add a web form named, *pback2.*
3. Open the page in Source mode and replace everything except the page directive (the first line) with:

<!DOCTYPE html>

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

<head runat="server">

 <title></title>

</head>

<body>

 <form id="form1" runat="server">

 <div>

 <p><strong>Postback Multipage Example - Page 2</strong></p>

 <p>

 <asp:Button ID="btnSubmit" runat="server" onclick="btnSubmit\_Click" Text="Submit/Postback" />&nbsp;

 <asp:Button ID="btnPage1" runat="server" onclick="btnPage1\_Click" Text="Page 1" />

 &nbsp;</p>

 <p>

 <asp:TextBox ID="txtMessage" runat="server" Height="182px" TextMode="MultiLine"

 Width="279px"></asp:TextBox>

 </p>

 </div>

 </form>

</body>

</html>

1. Open the code behind file, *pback2.aspx.cs* and replace the class (everything between the *namespace* line and the last “}”) with the code below. Study the code above carefully

public partial class pback2 : System.Web.UI.Page {

 string newLine = System.Environment.NewLine;

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

 // First time on page

 if (!Page.IsPostBack) {

 txtMessage.Text += newLine + "\*\*\*First time on page" + newLine;

 }

 // Postback.

 else {

 txtMessage.Text += newLine + "\*\*\*Post-back" + newLine;

 }

 }

 protected void btnSubmit\_Click(object sender, EventArgs e) {

 txtMessage.Text += "\*\*\*Submit callled" + newLine;

 }

 protected void btnPage1\_Click(object sender, EventArgs e) {

 // Go to page 1

 Response.Redirect("pback1.aspx");

 }

}

1. Do the following:
2. Run *pback1.aspx*
3. Note the message in the text area (“First time on page”)
4. Press the “Submit/Postback” button
5. Note:
6. *Page\_Load* is called and a postback is detected and a message is displayed, “Post-back”
7. Then, the *btnSubmit\_Click* event is executed which displays the message, “Submit called”
8. Press the “Submit/Postback” button several more times and observe the output.
9. Go back to VS and open the code behind file, *pback1.aspx.cs* and find the *btnPage2\_Click* event code:

protected void btnPage2\_Click(object sender, EventArgs e) {

 // Go to page 2.

 Response.Redirect("pback2.aspx");

}

Notice that we use the ASP.NET *Response* object to load a page:

 Response.Redirect("pback2.aspx");

1. Return to the browser where *pback1.aspx* is running. Press the “Page 2” button and control is transferred to *pback2.aspx*. Note the message in the text area (“First time on page”)
2. Press the “Submit/Postback” button several times and observe the output.
3. Press the “Page 1” button and control is transferred to *pback1.aspx*. Note the message in the text area (“First time on page”).

**Thus, when you leave a page and return, it is the first time on the page, *i.e.* it is not a postback.**

1. Press the “Submit/Postback” button several times and observe the output.
2. Go back to VS and open the code behind file, *pback1.aspx.cs* and find the *btnLoadPage\_Click* event code:

protected void btnLoadPage\_Click(object sender, EventArgs e) {

 // Reload the page.

 Response.Redirect(Request.RawUrl);

}

Notice that we redirect to: *Request.RawUrl* which is simply the page we are on. We are asking the server to reload the page which will, in effect, be the first time on the page.

1. Return to the browser where *pback1.aspx* is running and press the “Reload” button and observe that it is now the “First time on the page”.
2. Repeat the steps above, or simply experiment making sure you understand how postback works with multiple pages and what it means to reload a page.

**Stage 4 – Using Session Memory across Multiple Pages**

1. Do the following:
2. Add a web form named, *session2a.*
3. Open the page in Source mode and replace everything except the page directive (the first line) with:

<!DOCTYPE html>

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

<head runat="server">

 <title></title>

</head>

<body>

 <form id="form1" runat="server">

 <div>

 <p>

 Name

 <asp:TextBox ID="txtName" runat="server"></asp:TextBox>

 </p>

 <p>

 Favorite Integer

 <asp:TextBox ID="txtFavInt" runat="server"></asp:TextBox>

 </p>

 <p>

&nbsp;

 <asp:Button ID="btnSave" runat="server" OnClick="btnSave\_Click" Text="Save" />

&nbsp;<asp:Button ID="btnAdd1" runat="server" OnClick="btnAdd1\_Click" Text="Add 1" />

&nbsp;

 <asp:Button ID="btnPage2" runat="server" OnClick="btnPage2\_Click" Text="Page 2" />

 &nbsp;

 </p>

 <p>

 Your current sum is: <asp:Label ID="lblSum" runat="server" ForeColor="Red" Text="n/a"></asp:Label>

 <br />

 </p>

 </div>

 </form>

</body>

</html>

1. Open the code behind file, *session2a.aspx.cs* and replace the class (everything between the *namespace* line and the last “}”) with the code below. Study the code above carefully.

public partial class session2a : System.Web.UI.Page {

 private int sum;

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

 if (!Page.IsPostBack) {

 // If first time on page, initialize sum to 0 and put in Session.

 sum = 0;

 Session.Add("Sum", sum);

 }

 else {

 // If postback, pull sum from Session

 sum = Convert.ToInt32(Session["Sum"]);

 }

 }

 protected void btnSave\_Click(object sender, EventArgs e) {

 // Get the value entered ...

 int num = Convert.ToInt32(txtFavInt.Text);

 // ...which serves as the initial sum

 sum = num;

 // Add the sum and name to Session.

 Session.Add("Sum", sum);

 Session.Add("Name", txtName.Text);

 // Display the sum

 lblSum.Text = num.ToString();

 }

 protected void btnAdd1\_Click(object sender, EventArgs e) {

 // Add 1 to the current sum and then store in Session.

 sum++;

 Session.Add("Sum", sum);

 lblSum.Text = sum.ToString();

 }

 protected void btnPage2\_Click(object sender, EventArgs e) {

 // Go to new page.

 Response.Redirect("session2b.aspx");

 }

}

1. Do the following:
2. Add a web form named, *session2b.*
3. Open the page in Source mode and replace everything except the page directive (the first line) with:

<!DOCTYPE html>

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

<head runat="server">

 <title></title>

</head>

<body>

 <form id="form1" runat="server">

 <div>

 <p>

 Welcome,

 <asp:Label ID="lblName" runat="server" ForeColor="Red" Text="Label"></asp:Label>

 </p>

 <p>

 Your current sum is:

 <asp:Label ID="lblSum" runat="server" ForeColor="Red" Text="Label"></asp:Label>

 </p>

 <p>

 <asp:Button ID="btnAdd1" runat="server" OnClick="btnAdd1\_Click" Text="Add 1" />

&nbsp;

 <asp:Button ID="btnPage1" runat="server" OnClick="btnPage1\_Click" Text="Page 1" />

 <br />

 </p>

 </div>

 </form>

</body>

</html>

1. Open the code behind file, *session2b.aspx.cs* and replace the class (everything between the *namespace* line and the last “}”) with the code below. Study the code above carefully.

public partial class session2b : System.Web.UI.Page {

 private int sum;

 private string name;

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

 // Always get the sum from Session. However, note that if the

 // user starts on this page (e.g. has it bookmarked), the sum

 // may not be in session, which could cause a problem.

 sum = Convert.ToInt32(Session["Sum"]);

 lblSum.Text = sum.ToString();

 if (!Page.IsPostBack) {

 // Get the name out of Session and put in label

 name = Convert.ToString(Session["Name"]);

 lblName.Text = name;

 }

 }

 protected void btnAdd1\_Click(object sender, EventArgs e) {

 // Increase the sum by 1, put in Session, and display.

 sum++;

 Session.Add("Sum", sum);

 lblSum.Text = sum.ToString();

 }

 protected void btnPage1\_Click(object sender, EventArgs e) {

 // Go back to first page.

 Response.Redirect("session2a.aspx");

 }

}

1. Do the following:
2. Run *session2a.aspx,* type in a name and an integer, and then press, “Save”.
3. Press, “Add 1” a handful of time and observe the sum increase.
4. Press, “Page 2”, and *session2b.aspx* will be displayed. Note that the name and sum are displayed.
5. Press, “Add 1” several times and observe the sum being updated.
6. Press, “Page 1” and *session2a.aspx* will be displayed. Note that the sum is no longer correct.
7. (Read, no action required) Look carefully at *Page\_Load* in *session2a.apx.*

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

 if (!Page.IsPostBack) {

 // If first time on page, initialize sum to 0 and put in Session.

 sum = 0;

 Session.Add("Sum", sum);

 }

 else {

 // If postback, pull sum from Session

 sum = Convert.ToInt32(Session["Sum"]);

 }

}

Remember that when we leave a page and then return later, it is the “first time” on the page, *i.e.* it is not a postback. Thus, when we return to this page we are setting sum to 0.

1. To fix this, we need to detect not only if it is the first time on the page, but also if “Sum” exists in Session. Replace the code in *Page\_Load* with:

if (!Page.IsPostBack) {

 // If first time on page, and sum is not in Session,

 // initialize sum to 0 and put in Session.

 if (Session["Sum"] == null) {

 sum = 0;

 Session.Add("Sum", sum);

 }

 // If first time on page and sum is in Session, get it

 // and display it.

 else {

 int sum = Convert.ToInt32(Session["Sum"]);

 lblSum.Text = sum.ToString();

 }

}

else {

 // If postback, pull sum from Session

 sum = Convert.ToInt32(Session["Sum"]);

}

1. Run the page and verify that it is working correctly now.

**Stage 5 – Using a QueryString to Pass Information Between Pages**

Next, we are going to use a *QueryString* to pass information between pages.

1. (Read, no action required) Using a *QueryString* is a technique where you put the information you want to pass to another page in the URL in key/value pairs. For example, to send 2 key/value pairs to the page, *newPage.aspx*, we would write code like this:

Response.Redirect( “newPage.aspx?key1=value1&key2=value2” );

Note that the query string:

* Begins with a “?”
* Has any number of, “key=value” separated by “&”

The receiving page writes code like this to retrieve the values:

 value1 = Request.QueryString[“key1”];

 value2 = Request.QueryString[“key2”];

Note that the value is a string, so it will need to be converted if a number is sent.

1. Do the following:
2. Add a web form named, *qString1a.*
3. Open the page in Source mode and replace everything except the page directive (the first line) with:

<!DOCTYPE html>

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

<head runat="server">

 <title></title>

</head>

<body>

 <form id="form1" runat="server">

 <div>

 <table>

 <tr>

 <td>

 <asp:Label ID="Label1" runat="server" Text="First Name"></asp:Label>

 </td>

 <td>

 <asp:TextBox ID="txtFName" runat="server"></asp:TextBox>

 </td>

 </tr>

 <tr>

 <td>

 <asp:Label ID="Label2" runat="server" Text="Last Name"></asp:Label>

 </td>

 <td>

 <asp:TextBox ID="txtLName" runat="server"></asp:TextBox>

 </td>

 </tr>

 </table>

 <asp:Button ID="btnLogin" runat="server" Text="Login" OnClick="btnLogin\_Click" />

 <br />

 <asp:TextBox ID="txtMessage" runat="server" Height="189px" TextMode="MultiLine" Width="304px"></asp:TextBox>

 </div>

 </form>

</body>

</html>

1. Open the code behind file, *qString1a.aspx.cs* and replace the class (everything between the *namespace* line and the last “}”) with:

public partial class qString1a : System.Web.UI.Page {

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

 }

 protected void btnLogin\_Click(object sender, EventArgs e) {

 string fName = txtFName.Text;

 string lName = txtLName.Text;

 string url = "qString1b.aspx?fName=" + fName + "&lName=" + lName;

 Response.Redirect(url);

 }

}

1. Study the code above carefully.
2. Do the following:
3. Add a web form named, *qString1b.*
4. Open the page in Source mode and replace everything except the page directive (the first line) with:

<!DOCTYPE html>

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

<head runat="server">

 <title></title>

</head>

<body>

 <form id="form1" runat="server">

 <div>

 <asp:Button ID="btnLogOut" runat="server" OnClick="btnLogOut\_Click" Text="Log Out" />

 <br />

 <asp:TextBox ID="txtMessage" runat="server" Height="172px" TextMode="MultiLine" Width="281px"></asp:TextBox>

 </div>

 </form>

</body>

</html>

1. Open the code behind file, *qString1b.aspx.cs* and replace the class (everything between the *namespace* line and the last “}”) with:

public partial class qString1b : System.Web.UI.Page {

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

 if (Request.QueryString.AllKeys.Contains("fName")) {

 string fName = Request.QueryString["fName"];

 string lName = Request.QueryString["lName"];

 string msg = "Successful Login\n";

 msg += "First Name=" + fName + ", Last Name:" + lName;

 txtMessage.Text = msg;

 }

 }

 protected void btnLogOut\_Click(object sender, EventArgs e) {

 }

}

1. Study the code above carefully. We will program *btnLogOut\_Click* shortly.
2. Run the *qString1a.aspx page*. Type in names and press Login:



Then, *qString1b.aspx* is displayed. Note the names contained in the query string for the URL:



1. Next, we program the “Log Out” event handler. When the Log Out button is pressed on the *qString1b.aspx* page we will redirect to the login page (*qString1a.aspx*) passing a “status” in the query string indicating that the user has logged out. Then, in *Page\_Load* for *qString1a.aspx*, we will detect this status and display a message that the user has logged out.Do the following:
2. Open *qString1b.aspx.cs* and add this code to *btnLogOut\_Click:*

string url = "qString1a.aspx?status=logout";

Response.Redirect(url);

1. Open *qString1a.aspx.cs* and add this code in *Page\_Load*:

if (Request.QueryString.AllKeys.Contains("status")) {

 string status = Request.QueryString["status"];

 if (status.Equals("logout")) {

 txtMessage.Text = "You are logged out";

 }

}

1. Do the following:
2. Run the *qString1a.aspx* page.
3. Type in names and press Login.
4. Once *qString1b.aspx* is displayed, press Log Out. You will be returned to *qString1a.aspx*. Note the “status” in the query string and the message that is displayed
5. (Read, no action required) Suppose a user tries to load *qString1b.aspx* without logging in. In this case we want to direct them back to the login page (*qString1a.aspx*). Note the first line of *Page\_Load* in *qString1b.aspx*:

if (Request.QueryString.AllKeys.Contains("fName")) {

If the URL does not contain a “fName” key, then we will view this as an illegal attempt. Thus, we will add an *else* block to this code to build a URL with a query string, “status=illegalAccess” and redirect back to the login page. Finally, we will detect this in the login page and display a message.

1. Do the following:
	1. Paste this code immediately after the *if* block in *Page\_Load* for *qString1b.aspx*:

else {

 string url = "qString1a.aspx?status=illegalAccess";

 Response.Redirect(url);

}

* 1. Paste this code immediately after the inner *if* block in *Page\_Load* for *qString1a.aspx:*

else if (status.Equals("illegalAccess")) {

 txtMessage.Text = "You must login";

}

 When done, *Page\_Load* in *qString1a.aspx.cs* is:

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

 if (Request.QueryString.AllKeys.Contains("status")) {

 string status = Request.QueryString["status"];

 if (status.Equals("logout")) {

 txtMessage.Text = "You are logged out";

 }

 else if (status.Equals("illegalAccess")) {

 txtMessage.Text = "You must login";

 }

 }

}

1. Do the following:
2. Run the *qString1b.aspx* page.
3. This page will not be displayed, you will be redirected to *qString1a.aspx*. Note the message and the query string in the URL.

**Stage 6 – Package Assignment for Submission**

1. Close VS and zip your *lab05\_lastName* project folder and submit on Blazeview in the *Lab 05* dropbox.