**EXPERIMENT 11**

# ASP.net Passing values across webpages and SQL Injection

**Objective**

## Familiarize you with how values can be passed across webpages

## Introduce you to SQL injection.

## Ways of passing values between ASP.net pages

There are number of ways to transfer values between asp pages

1. Query String
2. Session Variable
3. Application Variable
4. Cookies  
   and number of others

Below are references which will explain how to transfer values with examples:  
  
<https://learn.microsoft.com/en-us/previous-versions/aspnet/6c3yckfw(v=vs.100)>

[http://www.codeproject.com/Articles/8055/Transferring-page-values-to-another-page](http://www.google.com/url?q=http%3A%2F%2Fwww.codeproject.com%2FArticles%2F8055%2FTransferring-page-values-to-another-page&sa=D&sntz=1&usg=AFQjCNEhDxb6GR-wB1OHUgZMM1qlwV1lkA)  
[http://www.codeproject.com/Articles/8350/Passing-information-between-pages-The-NET-way](http://www.google.com/url?q=http%3A%2F%2Fwww.codeproject.com%2FArticles%2F8350%2FPassing-information-between-pages-The-NET-way&sa=D&sntz=1&usg=AFQjCNGxaRxhTSVNv7pIpZlpZjt0DoJUQA)  
[http://www.codeproject.com/Articles/5876/Passing-variables-between-pages-using-QueryString](http://www.google.com/url?q=http%3A%2F%2Fwww.codeproject.com%2FArticles%2F5876%2FPassing-variables-between-pages-using-QueryString&sa=D&sntz=1&usg=AFQjCNHwfhqjms81ynH4DY0dlhyGIm12ag)

**Transfer values across ASP.net pages using query string**

In your asp.net project (you can use the one, created in Lab10). Add 2 new webforms named Login.aspx and Profile.aspx

In Login.aspx, Add the following code within asp:content tag with id=content2 as shown below.

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

User:

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

<br/>

Password:

<asp:TextBox ID="txtpassword" runat="server" TextMode="Password"></asp:TextBox>

<br/>

<asp:Button ID="submit" runat="server" Text="submit" onclick="submit\_Click"/>

</asp:Content>

In Login.aspx.cs file Add the following code in the submit\_Click method

as shown below.

Protected void submit\_Click(object sender, EventArgs e)

{

Response.Redirect("Profile.aspx?Name="+txtusrname.Text+"&pwd="+txtpassword.Text);

}

In Profile.aspx page add a label within the appropriate asp:content tag, as shown

<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

<asp:Label ID="lblUserName" runat="server" Text="Label"></asp:Label>

</asp:Content>

In Profile.aspx.cs file add the following code as shown

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

{

lblUserName.Text=Request.QueryString["Name"];

}

Build and run your Login.aspx page and see what happens on clicking the submit button. Paste a screen shot of both pages in your lab report.

**Grid Transfer value**

Add the following templatefield before ItemNo template field in Home.aspx page made in last lab as shown in figure 1.

<asp:TemplateField>

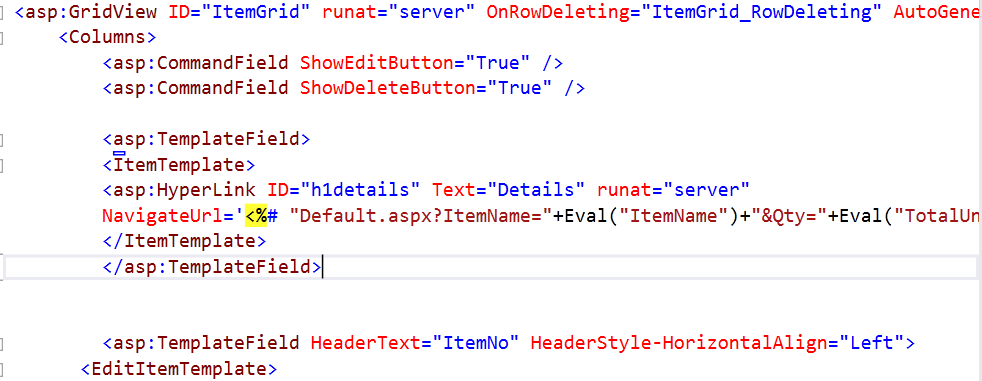
<ItemTemplate>

<asp:HyperLink ID="h1details" Text="Details" runat="server"

NavigateUrl='<%# "Default.aspx?ItemName="+Eval("ItemName")+"&Qty="+Eval("TotalUnits")%>'/>

</ItemTemplate>

</asp:TemplateField>



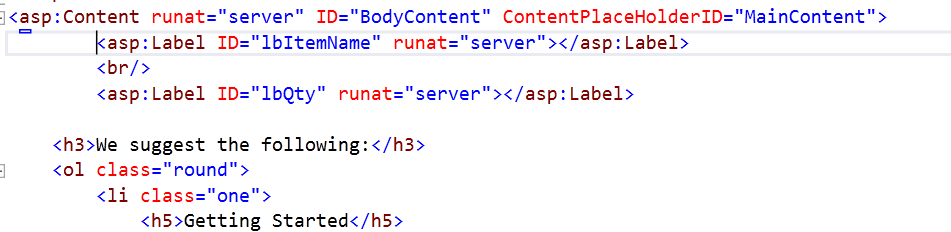
**Figure 1**

In Default.aspx Add the following two labels as shown in figure 2.

<asp:Label ID="lbItemName" runat="server"></asp:Label>

<br/>

<asp:Label ID="lbQty" runat="server"></asp:Label>



**Figure 2**

In page load method of **Default.aspx.cs** add the following code

lbItemName.Text = Request.QueryString["ItemName"];

lbQty.Text = Request.QueryString["Qty"];

Buid and run your Home.aspx and see what happens on clicking the Details link present in the grid view

## SQL Injection

A SQL Injection attack is a form of attack that comes from user input that has not been checked to see that it is valid. The objective is to fool the database system into running malicious code that will reveal sensitive information or otherwise compromise the server.

There are two main types of attacks. First-order attacks are when the attacker receives the desired result immediately, either by direct response from the application they are interacting with or some other response mechanism, such as email. Second-order attacks are when the attacker injects some data that will reside in the database, but the payload will not be immediately activated. We will discuss each in more detail later in this article.

**An example of what an attacker might do**

In the following example, assume that a web site is being used to mount an attack on the database. If you think about a typical SQL statement, you might think of something like:

SELECT ProductName, QuantityPerUnit, UnitPrice

FROM Products

WHERE ProductName LIKE 'G%'

The objective of the attacker is to inject their own SQL into the statement that the application will use to query the database. If, for instance, the above query was generated from a search feature on a web site, then they user may have inserted the "G" as their query. If the server side code then inserts the user input directly into the SQL statement, it might look like this:

String sql = "SELECT ProductName, QuantityPerUnit, UnitPrice "+

"FROM Products " +

"WHERE ProductName LIKE '"+this.search.Text+"%';

SqlDataAdapter da = new SqlDataAdapter(sql, DbCommand);

da.Fill(productDataSet);  
  
This is all fine if the data is valid, but what if the user types something unexpected? What happens if the user types:

' UNION SELECT name, type, id FROM sysobjects;--

Note the initial apostrophe; it closes the opening quote in the original SQL statement. Also, note the two dashes at the end; that starts a comment, which means that anything left in the original SQL statement is ignored.

Now, when the attacker views the page that was meant to list the products the user has searched for, they get a list of all the names of all the objects in the database and the type of object that they are. From this list, the attacker can see that there is a table called *Users*. If they take note of the id for the *Users* table, they could then inject the following:

' UNION SELECT name, '', length FROM syscolumns WHERE id = 1845581613;--

This would give them a list of the column names in the *Users* table. Now they have enough information to get access to a list of users, passwords, and if they have admin privileges on the web site.

' UNION SELECT UserName, Password, IsAdmin FROM Users;--

Assume that there is a table called *Users* which has columns called UserName and Password, it is possible to union that with the original query and the results will be interpreted as if the UserName was the name of the product and the Password was the quantity per unit. Finally, because the attacker discovered that there is aIsAdmin column, they are likely to retrieve the information in that too.

**Solution of SQL Injection Attack .net?**

From server side you can use this method

string surname = this.surnameTb.Text.Replace("'", "''");

string sql = "Update Users SET Surname='"+surname+"' "+

"WHERE id="+userID;  
  
from client side you can use this method by using JavaScript validation   
include JavaScript file in solution or embed same JavaScript code in your aspx page

Example of sql injection protection is shared with you in SQL-INJECTION SAMPLE-Login.aspx  
  
  
<asp:Button ID="BtnLogin" runat="server" Text="Login"   
 onclick="BtnLogin\_Click" OnClientClick= "javascript: return validation(); " />

<script type="text/javascript">  
 function validation() {  
 var username = document.getElementById('<%=TextBoxUserName.ClientID %>').value;  
 var password = document.getElementById('<%=TextBoxPassword.ClientID %>').value;

if (username.search("'") >= 0 || username.search("--") >= 0) // sql injection characters  
 {  
 alert('Please Enter Username');  
 return false;

}  
 else   
 {  
 if (password.search("'") >= 0 || password.search("--") >= 0)   
 {  
 alert('Please Enter Password');

return false;  
 }  
 return true;  
 }  
  
 }  
</script>

Use this link as ref:  
  
[http://msdn.microsoft.com/en-us/library/ms161953%28SQL.105%29.aspx](http://www.google.com/url?q=http%3A%2F%2Fmsdn.microsoft.com%2Fen-us%2Flibrary%2Fms161953%2528SQL.105%2529.aspx&sa=D&sntz=1&usg=AFQjCNHa-fKRaRsuah53aa6IRpgMzudUfA)  
[http://msdn.microsoft.com/en-us/library/ff648339.aspx](http://www.google.com/url?q=http%3A%2F%2Fmsdn.microsoft.com%2Fen-us%2Flibrary%2Fff648339.aspx&sa=D&sntz=1&usg=AFQjCNFKc-4PcJ6UKgIGzJ_5NNiux_U6yg)