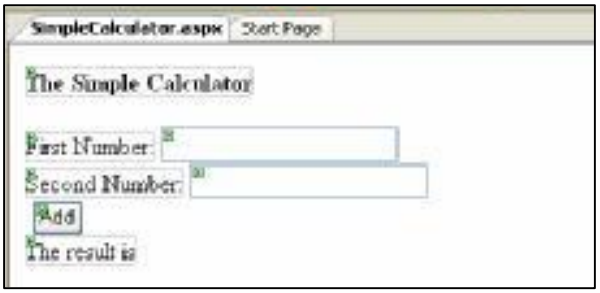
Lab 3- Introduction to ASP.NET

Question 1

A Simple Web Application

Create a web form as follow:

1. Drag and drop 4 labels, 2 textboxes and 1 button from the toolbox panel to the web form.
2. Change the text for the controls in the properties panel accordingly to produce a page that is same to the figure below.



3. Double click the Add button on the web form and type the following codes.

```
protected void Button1_Click(object sender, EventArgs e)
{
double result = 0.0;
result = Convert.ToDouble(TextBox1.Text)
+ Convert.ToDouble(this.TextBox2.Text);
Label4.Text = "The result is: " + result.ToString();
}
```

4. Click the Start Debugging icon or press F5 to debug the codes and preview the web form in browser.

Question 2

Parse HTTP Form Data with ASP.NET

When you use ASP.NET to process data that is entered on a Web page through a form, two different collections are used depending on which HTTP method is specified in the form declaration:

- GET uses the Request.QueryString collection.
- POST uses the Request.Form collection.

This article describes how to process form data regardless of which HTTP method is used on the form.

Use a Generic Collection to Parse HTTP Form Data

The following steps describe how to create a sample HTML page with two Web forms (one that uses the GET method and one that uses the POST method) and how to create a sample ASP.NET page that processes the form by using either method.

1. Save the following HTML code as **ParseData.html** in the folder of a Web site project.

```
<body>
<form action="showdata.aspx" method="GET">
GET
    <input type="text" name="txtData" />
    <input type="submit" />
</form>
<form action="showdata.aspx" method="POST">
POST
    <input type="text" name="txtData" />
    <input type="submit" />
</form>
</body>
```

2. Save the following ASP.NET code as showdata.aspx in the folder of a Web site Project.

```
<body>
<%
    Response.Write("Request Method = ");
    Response.Write(Request.ServerVariables["Request_Method"]);
    Response.Write("<br />");
    if (Request.ServerVariables["Request_Method"].Equals("GET"))
    {
        Response.Write("Content  = ");
        Response.Write(Request.QueryString["txtData"]);
    }
    if (Request.ServerVariables["Request_Method"].Equals("POST"))
    {
        Response.Write("Content = ");
        Response.Write(Request.Form["txtData"]);
    }
%>
</body>
```

3. When you open ParseData.htm by using HTTP, you may use either form to view the form results.

Question 3

Use Session Variables to Store and Retrieve HTTP Form Data

ASP.NET session variables enable you to store and retrieve values for a user as the user navigates the different ASP.NET pages that make up a Web application. Session variables are created by simply referring to the session variable by name.

1. Save the following code as NewParseData.aspx in the folder of a Web site project.

```
<body>
<form id="form1" runat="server">
<div>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
    <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="Submit Query" />
</div>
</form>
</body>
```

2. Double click the button and add the code below:

```
protected void Button1_Click(object sender, EventArgs e)
{
Session["txtData"] = this.TextBox1.Text;
Response.Redirect("NewShowData.aspx");
}
```

3. Save the following ASP.NET code as NewShowData.aspx in the folder of a Web site Project.

```
<body>
<%
    Response.Write("Session[\"txtData\"] = ");
    Response.Write(Session["txtData"]);
%>
</body>
```

4. Run and preview NewParseData.aspx.