Fakulti Sains Komputer dan Teknologi Maklumat
Universiti Tun Hussein Onn Malaysia

LABORATORY 6

This laboratory exercise is about ASP.NET threads and using session.

A. Threads

A thread is defined as the execution path of a program. Each thread defines a unique flow of control. So far we compiled programs where a single thread runs as a single process which is the running instance of the application. However, this way the application can perform one job at a time. To make it execute multiple tasks at a time, it could be divided into smaller threads.

Sample 1:

Markup code

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

<!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>
            <h3>Thread Example</h3>
         </div>

         <!-- <asp:Label ID="lblmessage" runat="server" Text="Label">
         </asp:Label>-->
      </form>
   </body>

</html>
```

Code behind

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;

using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;
using System.Threading;

namespace threaddemo
{

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

            ThreadStart childthreat = new ThreadStart(childthreadcall);
            Response.Write("Child Thread Started <br/>");
            Thread child = new Thread(childthreat);

            child.Start();

            Response.Write("Main sleeping  for 2 seconds.......<br/>");
            Thread.Sleep(2000);
            Response.Write("<br/>Main aborting child thread<br/>");

            child.Abort();
        }

        public void childthreadcall()
        {
            Response.Write("We are in");
            try
            {
                Response.Write("<br />Child thread started <br/>");
                Response.Write("Child Thread: Counting to 10");

                for (int i = 0; i < 10; i++)
                {
                    Thread.Sleep(500);
                    Response.Write("<br/> in Child thread </br>");
                }

                Response.Write("<br/> child thread finished");

            }
            catch (ThreadAbortException e)
            {

                Response.Write("<br /> child thread - exception");

            }
            finally
            {
                Response.Write("<br /> child thread - unable to catch the
exception");
            }
        }
    }
}
```

Exercise 1: Observe the codes above and write down the output. Explain the *threads*.

B. Sessions:

Normally, whenever server provides a response, afterwards the instance of the page and the value of the control are destroyed. What if we have a requirement to store the values of the controls and pass them into another web form then a State Management Technique is used? We need to use session.

Session is a State Management Technique. A Session can store the value on the Server. It can support any type of object to be stored along with our own custom objects.

Sample 2:

Markup code

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs"
Inherits="_Default" %>
<!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>Sessions</title>
</head>
<body runat="server" id="BodyTag">
    <form id="form1" runat="server">
    <asp:DropDownList runat="server" id="ColorSelector" autopostback="true"
onselectedindexchanged="ColorSelector_IndexChanged">
        <asp:ListItem value="White" selected="True">Select
color...</asp:ListItem>
        <asp:ListItem value="Red">Red</asp:ListItem>
        <asp:ListItem value="Green">Green</asp:ListItem>
        <asp:ListItem value="Blue">Blue</asp:ListItem>
    </asp:DropDownList>
    </form>
</body>
</html>
```

Code behind

```
using System;
using System.Data;
using System.Web;

public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if(Session["BackgroundColor"] != null)
        {
            ColorSelector.SelectedValue =
Session["BackgroundColor"].ToString();
            BodyTag.Style["background-color"] = ColorSelector.SelectedValue;
        }
    }

    protected void ColorSelector_IndexChanged(object sender, EventArgs e)
    {
        BodyTag.Style["background-color"] = ColorSelector.SelectedValue;
        Session["BackgroundColor"] = ColorSelector.SelectedValue;
    }
}
```

Exercise 1: Observe the codes above and write down the output. Explain the *session*.

<u>Instruction for submission:</u>

- Your lab report must be in pdf.
- Copy your code program in asp.net, C# code in codebehind and screenshot the output displayed in the browser.
- Submission at **AUTHOR** (Tab Individual Activities).
- All work is to be done on an individual basis.
- Duration: 1 week only.