

State Management

Prepared for Vth semester DDU-CE students
2022-23 WAD

Apurva A Mehta

*

Aishwarya Rai	P.V.Sindhu	Chitra Ramkrishna	Muthayya Vanitha and Ritu Karidhal
Actress	Badminton Player	Former CEO at NSE	Project Dir. - Chandrayaan- 2

Introduction

- Stateless web server
- Requirement to maintain data per user.
 - Login details
 - Browsing data
- Passing of data from one page to another page.

State Management Techniques

Client —————→ **Server**

I am Cody. I am shopping for books. I have two titles in my cart worth of 340 INR

I am session 856

Client —————→ **Server**

Session 856 is Cody. Cody is shopping for books. Cody has two titles in cart worth of 340 INR

Client Side

- Techniques
 - **View State**
 - Control State
 - Hidden Fields
 - **Cookies**
 - **Query Strings**
- Better Scalability
- Support for multiple web servers

View State

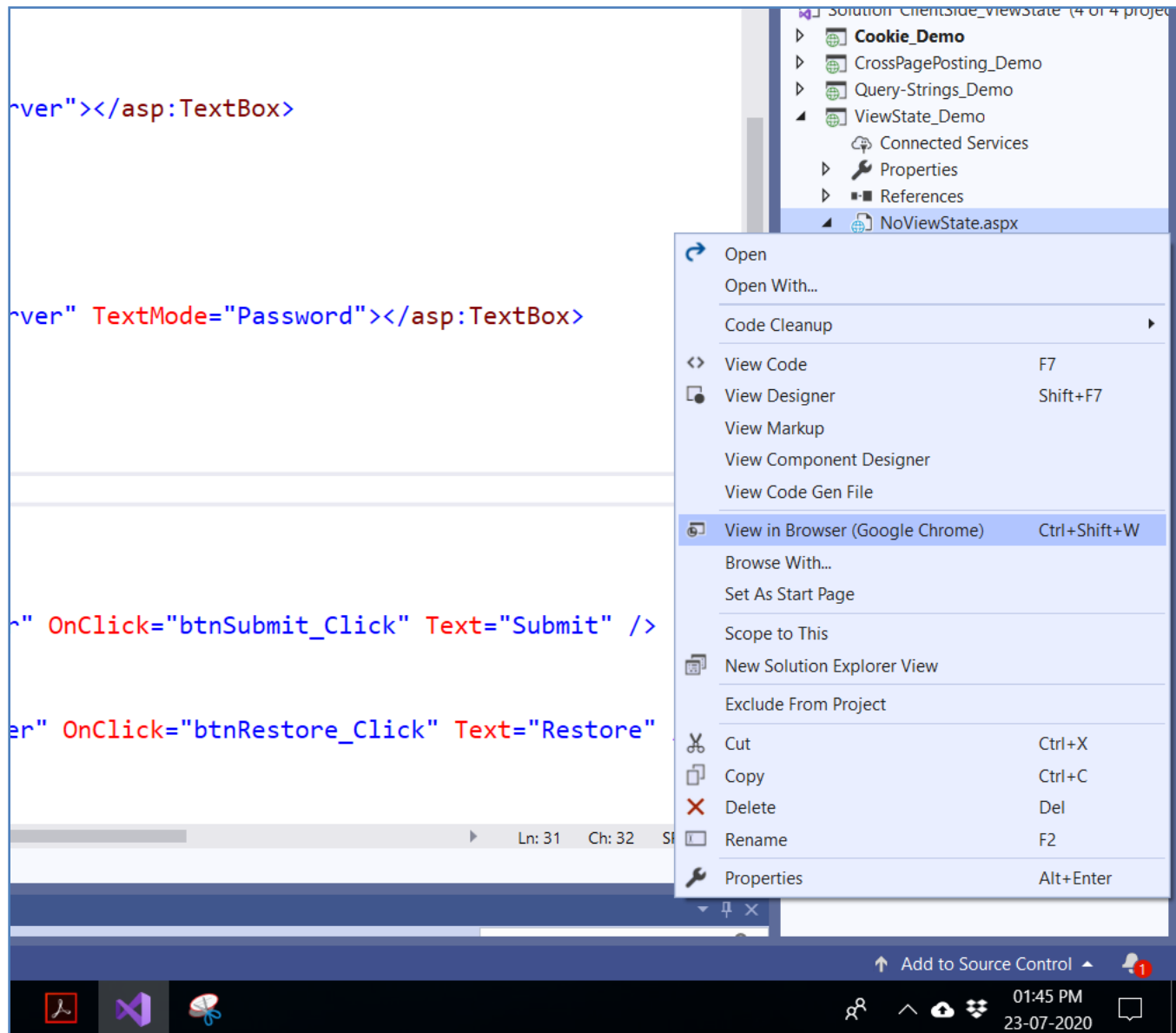
- Method to preserve the Value of the Page and Controls between round trips
- Data is not stored by server.
- Page level state management.
- Used to retrieve old values after page postback.
- Page.ViewState
 - __ViewState

Why ViewState is better than hidden fields in ASP.NET?

```
<tr>
    <td>Username</td>
    <td>
        <asp:TextBox ID="txtUsername" runat="server"></asp:TextBox>
    </td>
</tr>
<tr>
    <td>Password</td>
    <td>
        <asp:TextBox ID="txtPassword" runat="server" TextMode="Password"></asp:TextBox>
    </td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
</tr>
<tr>
    <td>
        <asp:Button ID="btnSubmit" runat="server" OnClick="btnSubmit_Click" Text="Submit" />
    </td>
    <td>
        <asp:Button ID="btnRestore" runat="server" OnClick="btnRestore_Click" Text="Restore" />
    </td>
</tr>
```



```
public partial class NoViewState : System.Web.UI.Page
{
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
    }
    string UName;
    string Password;
    0 references
    protected void btnSubmit_Click(object sender, EventArgs e)
    {
        UName = txtUsername.Text;
        Password = txtPassword.Text;
        txtPassword.Text = txtUsername.Text = string.Empty;
    }
    0 references
    protected void btnRestore_Click(object sender, EventArgs e)
    {
        txtUsername.Text = UName;
        txtPassword.Text = Password;
    }
}
```



← → ↻ ⓘ localhost:49659/NoViewState.aspx

📱 Apps 🌐 Laravel 📺 Laravel PHP Frame... 🌐 5 Fun Memory Gam... 🌐 H Fai Poon, Ph D |...

Username

Password

← → ↻ ⓘ localhost:49659/NoViewState.aspx

📱 Apps 🌐 Laravel 📺 Laravel PHP Frame... 🌐 5 Fun Memory Gam... 🌐 H Fai Poon, Ph D |...

Username

Password



localhost:49659/NoViewState.aspx

Apps Laravel Laravel PHP Frame... 5 Fun Memory Gam... H Fai Poon, Ph D |...

Username

Password

Submit

Restore

18 `protected void btnSubmit_Click(object sender, EventArgs e)`
19 `{`
20 `UName = txtUsername.Text;`
21 `Password = txtPassword.Text;` ≤ 2ms elapsed
22 `txtPassword.Text = txtUsername.Text = string.Empty;`
23 `}`

140 % No issues found

Autos

Search (Ctrl+E) Search Depth: 3

Name	Value	Type
System.Web.UI.WebControls.TextBox...	"aam"	string
Password	null	string
UName	"aam"	string
this	{ASP.noviewstate_aspx}	ViewState_Demo.N...
txtPassword	{System.Web.UI.WebControls.TextBox}	System.Web.UI.Web...
txtPassword.Text	"aam"	string
txtUsername	{System.Web.UI.WebControls.TextBox}	System.Web.UI.Web...
txtUsername.Text	"aam"	string

18 `protected void btnSubmit_Click(object sender, EventArgs e)`
19 `{`
20 `UName = txtUsername.Text;`
21 `Password = txtPassword.Text;`
22 `txtPassword.Text = txtUsername.Text = string.Empty;` ≤ 1ms
23 `}`
24

0 references

140 % No issues found

Autos

Search (Ctrl+E) Search Depth: 3

Name	Value	Type
Password	"aam"	string
this	{ASP.noviewstate_aspx}	ViewState_Demo.N...
txtPassword	{System.Web.UI.WebControls.TextBox}	System.Web.UI.Web...
txtPassword.Text	"aam"	string
txtUsername	{System.Web.UI.WebControls.TextBox}	System.Web.UI.Web...
txtUsername.Text	"aam"	string

← → ↻ ⓘ localhost:49659/NoViewState.aspx

Apps ↻ Laravel ▶ Laravel PHP Frame... ↻ 5 Fun Memory Gam... ↻ H Fai Poon, Ph D |...

Username

Password

0 references

```
25 protected void btnRestore_Click(object sender, EventArgs e)
26 {
27     txtUsername.Text = UName;
28     txtPassword.Text = Password; ≤ 1ms elapsed
29 }
30
```

140 % No issues found

Autos Search (Ctrl+E) Search Depth: 3

Name	Value	Type
Password	null	string
UName	null	string
this	{ASP.noviewstate.aspx}	ViewState_Demo.N...
txtPassword	{System.Web.UI.WebControls.TextBox}	System.Web.UI.Web...
txtPassword.Text	""	string
txtUsername	{System.Web.UI.WebControls.TextBox}	System.Web.UI.Web...
txtUsername.Text	""	string

Call Stack

Name
ViewStat [External]

```
<tr>
    <td>Username</td>
    <td><asp:TextBox ID="txtUsername" runat="server"></asp:TextBox></td>
</tr>
<tr>
    <td>Password</td>
    <td><asp:TextBox ID="txtPassword" runat="server" TextMode="Password"></asp:TextBox></td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
</tr>
<tr>
    <td><asp:Button ID="btnSubmit" runat="server" OnClick="btnSubmit_Click" Text="Submit" /></td>
    <td><asp:Button ID="btnRestore" runat="server" OnClick="btnRestore_Click" Text="Restore" /></td>
</tr>
<tr>
    <td>
        <asp:HyperLink ID="HyperLink1" runat="server" NavigateUrl="~/WebForm1.aspx">
            WebForm1
        </asp:HyperLink>
    </td>
    <td>&nbsp;</td>
</tr>
```

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    ViewState["Username"] = txtUsername.Text;
    ViewState["Password"] = txtPassword.Text;

    txtUsername.Text = txtPassword.Text = string.Empty;
}
```

0 references

```
protected void btnRestore_Click(object sender, EventArgs e)
{
    if (ViewState["Username"] != null)
    {
        txtUsername.Text = ViewState["Username"].ToString();
    }
    if (ViewState["Password"] != null)
    {
        txtPassword.Text = ViewState["Password"].ToString();
    }
}
```


localhost:49659/ViewState.aspx x +

localhost:49659/ViewState.aspx

Apps Laravel Laravel PHP Frame... 5 Fu

Username

Password

[WebForm1](#)

localhost:49659/ViewState.aspx x +

localhost:49659/ViewState.aspx

Apps Laravel Laravel PHP Frame...

Username

Password

[WebForm1](#)

localhost:49659/ViewState.aspx x +

localhost:49659/ViewState.aspx

Apps Laravel Laravel PHP Frame... 5

Username

Password

[WebForm1](#)

localhost:49659/ViewState.aspx x +

localhost:49659/ViewState.aspx

Apps Laravel Laravel PHP Frame... 5 Fun Memory Ga

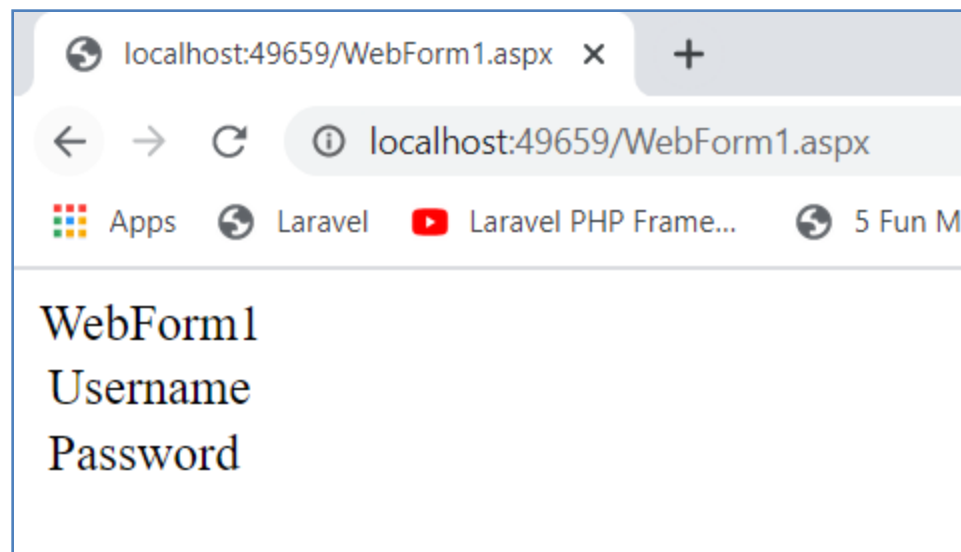
Username

Password

[WebForm1](#)

```
<div>
WebForm1
    <br />
    <table class="auto-style1">
        <tr>
            <td>Username</td>
            <td>
                <asp:Label ID="lblUsername" runat="server"></asp:Label>
            </td>
        </tr>
        <tr>
            <td>Password</td>
            <td>
                <asp:Label ID="lblPassword" runat="server"></asp:Label>
            </td>
        </tr>
    </table>
</div>
```

```
protected void Page_Load(object sender, EventArgs e)
{
    if (ViewState["Username"] != null)
    {
        lblUsername.Text = ViewState["Username"].ToString();
    }
    if (ViewState["Password"] != null)
    {
        lblPassword.Text = ViewState["Password"].ToString();
    }
}
```



Cont.

- No server resources are required
- Passed during every *postback* as hidden elements.
 - Add few Kbytes to the page
- Passed as plain text.
- Tightly bound to a single page.

How do ASP.NET web forms remember the settings for controls between user requests?

Is the view state lost if a user refreshes a webpage? What if the user uses email to send a URL to a friend?

Query Strings

- <https://www.google.com/search?q=tom+hanks+sully>
- Limited to simple Strings with URL legal characters
- Clearly visible to the user and to anyone else who cares to eavesdrop on the Internet
- Limit on size of Query Strings
- Submit page by HTTP GET command
- Query string data is included in bookmarks

Cont.

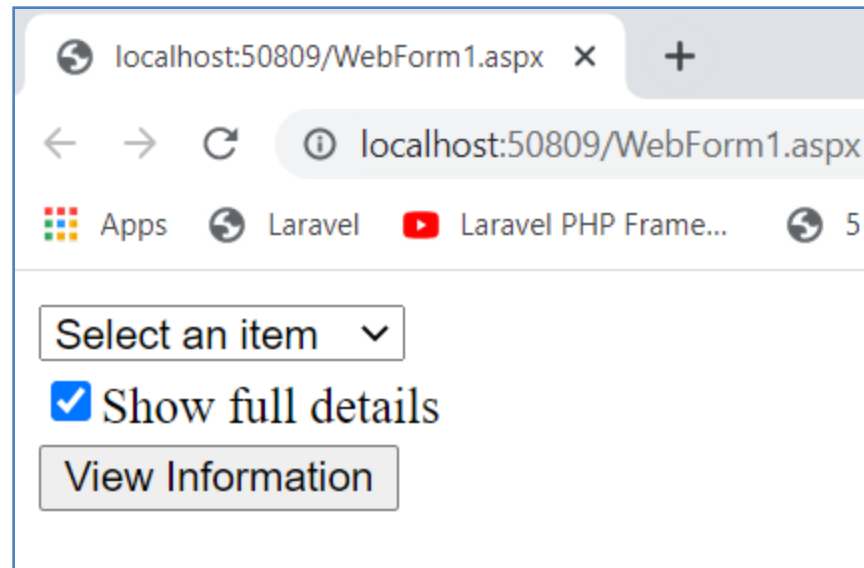
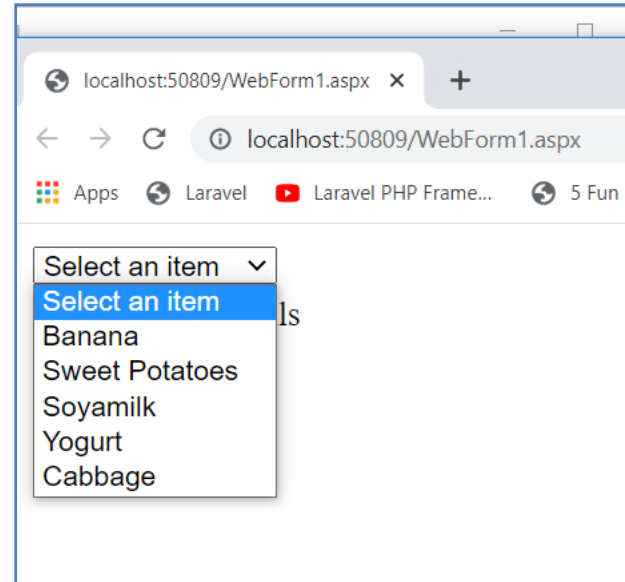
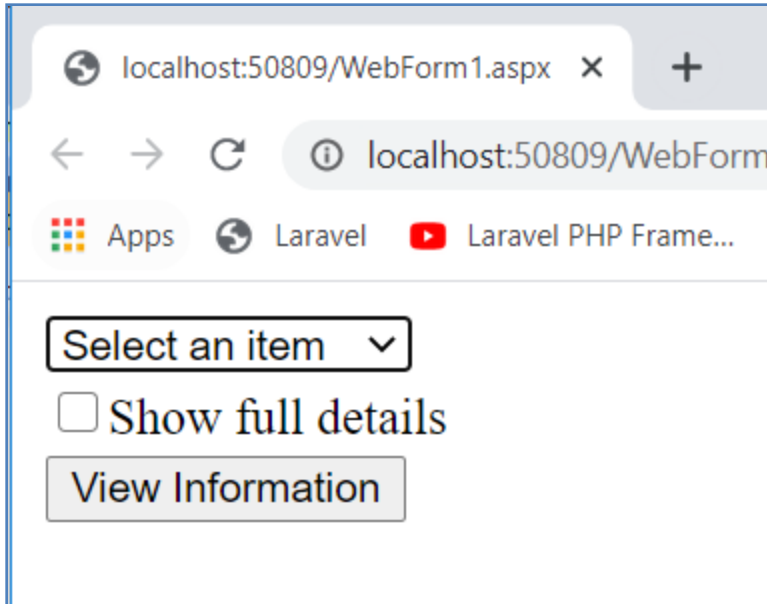
- Should always encode query string
 - `Server.UrlEncode()`, `Server.HtmlEncode()`
- Retrieve on server
 - `Page.Request.QueryString`
- Simple but limited way to maintain state information between pages

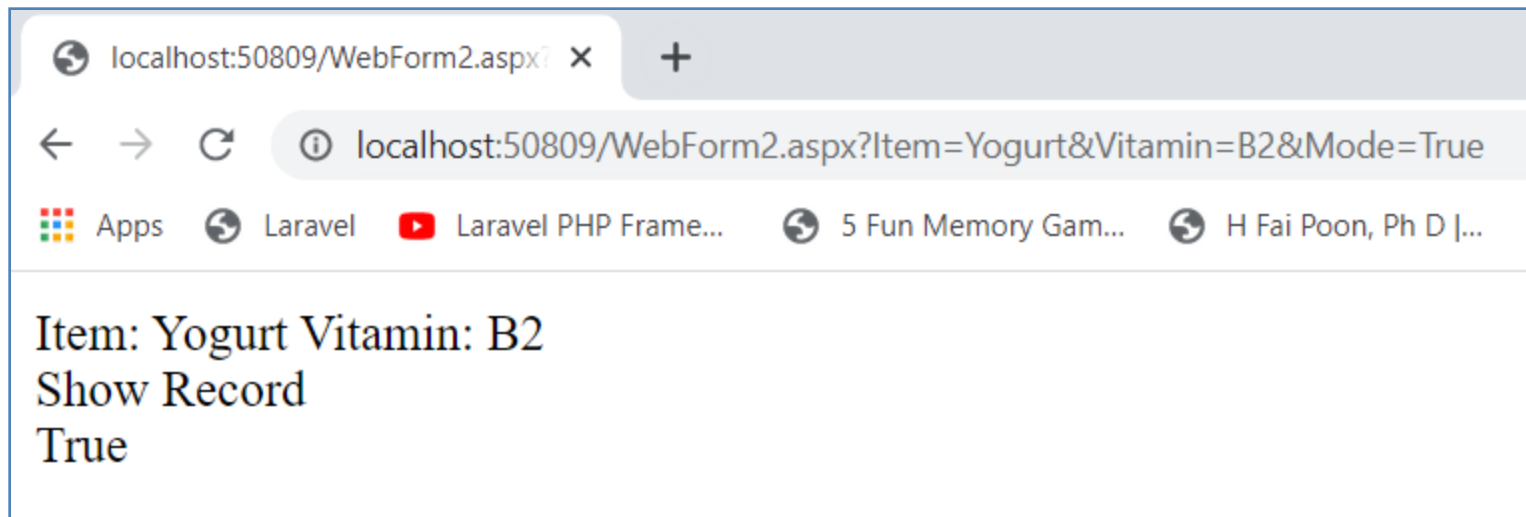
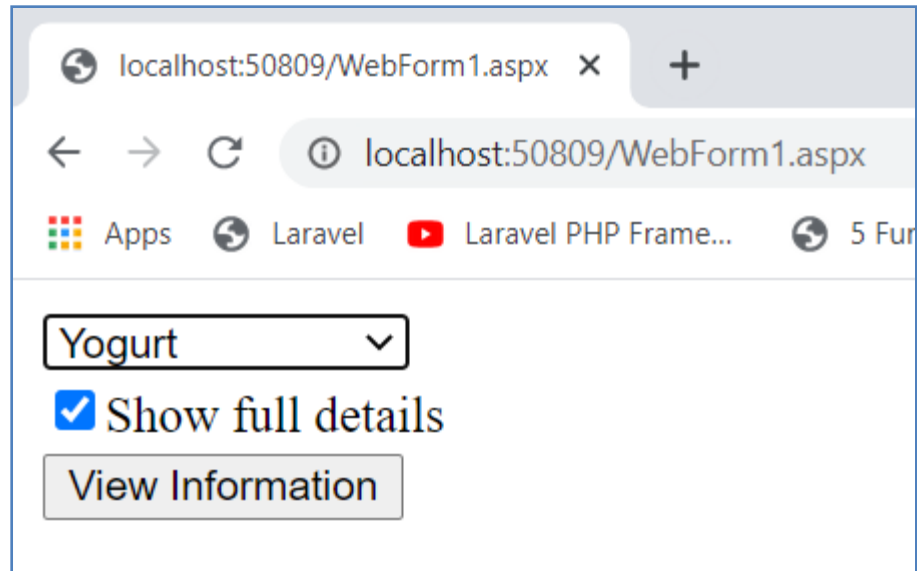
```
<tr><td>
    <asp:DropDownList ID="ddlFruits" runat="server">
        <asp:ListItem Value="-1">Select an item</asp:ListItem>
        <asp:ListItem Value="C">Banana</asp:ListItem>
        <asp:ListItem Value="A">Sweet Potatoes</asp:ListItem>
        <asp:ListItem Value="B1">Soyamilk</asp:ListItem>
        <asp:ListItem Value="B2">Yogurt</asp:ListItem>
        <asp:ListItem Value="K">Cabbage</asp:ListItem>
    </asp:DropDownList>
</td>
<td>&nbsp;</td>
</tr>
<tr><td>
    <asp:CheckBox ID="chkDetails" runat="server" Text="Show full details" />
</td>
<td>&nbsp;</td>
</tr>
<tr><td>
    <asp:Button ID="btnShow" runat="server" OnClick="btnShow_Click" Text="View Information" />
</td>
<td>
    <asp:Label ID="lblStatus" runat="server"></asp:Label>
</td>
</tr>
```

```
protected void btnShow_Click(object sender, EventArgs e)
{
    if(ddlFruits.SelectedItem.Value=="-1")
    {
        lblStatus.Text = "Please select an item";
    }
    else
    {
        string URL = "WebForm2.aspx?";
        URL += "Item=" + Server.UrlEncode(ddlFruits.SelectedItem.Text) + "&";
        URL += "Vitamin=" + Server.UrlEncode(ddlFruits.SelectedItem.Value) + "&";
        URL += "Mode=" + Server.HtmlEncode(chkDetails.Checked.ToString());
        Response.Redirect(URL);
    }
}
```

```
<tr>
    <td>
        <asp:Label ID="lblInfo" runat="server"></asp:Label>
    </td>
    <td>&nbsp;</td>
</tr>
```

```
public partial class WebForm2 : System.Web.UI.Page
{
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
        lblInfo.Text = "Item: " + Request.QueryString["Item"];
        lblInfo.Text += " Vitamin: " + Request.QueryString["Vitamin"];
        lblInfo.Text += "<br/> Show Record <br/>";
        lblInfo.Text += Request.QueryString["Mode"];
    }
}
```





Cookies

- Small files created in
 - Web browser's memory temporary
 - Client's hard drive permanent
- Reliable and Flexible
- Long term storage
 - Used by any page
 - Retained between visits
- Client can delete cookies
- Do not solve issue of user moving from one device to another.

Cont.

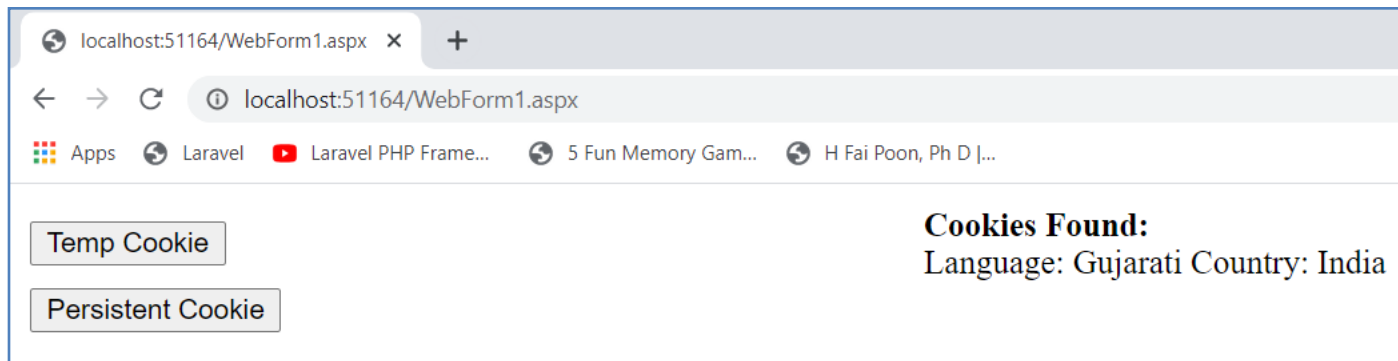
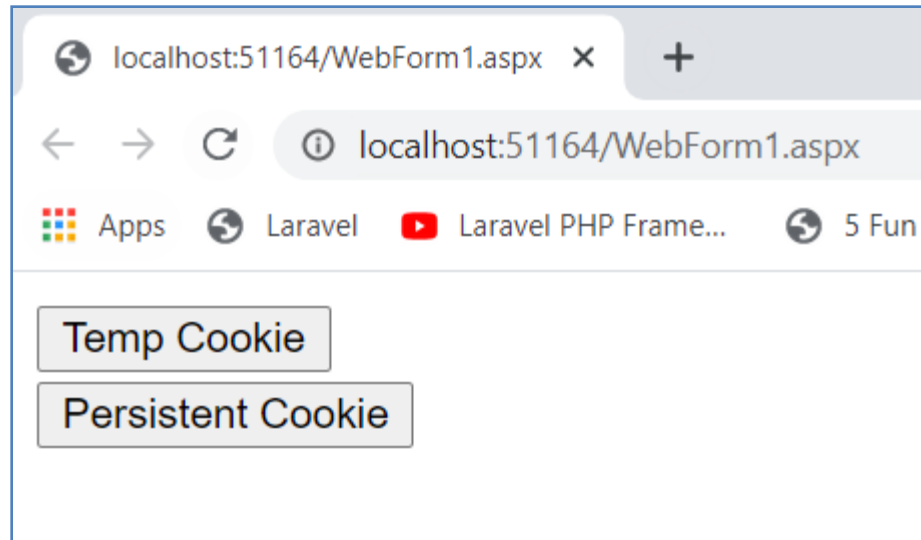
- using System.Net;
- Response.Cookies.Add()
- Page.Response
 - Cookies
 - HttpCookieCollection
 - HttpCookie
 - » Name, Value
- How to delete from server?


```
<tr>
  <td>
    <asp:Button ID="btnTemp" runat="server" OnClick="btnTemp_Click" Text="Temp Cookie" />
  </td>
  <td>
    <asp:Label ID="lblStatus" runat="server"></asp:Label>
  </td>
</tr>
<tr>
  <td>
    <asp:Button ID="btnPersistentCookie" runat="server" OnClick="btnPersistentCookie_Click"
      Text="Persistent Cookie" />
  </td>
  <td>&nbsp;</td>
</tr>
```

```
protected void btnTemp_Click(object sender, EventArgs e)
{
    HttpCookie cookie = new HttpCookie("Preferences");

    cookie["language"] = "Gujarati";
    cookie["country"] = "India";
    Response.Cookies.Add(cookie);
    //persist until the user closes the browser

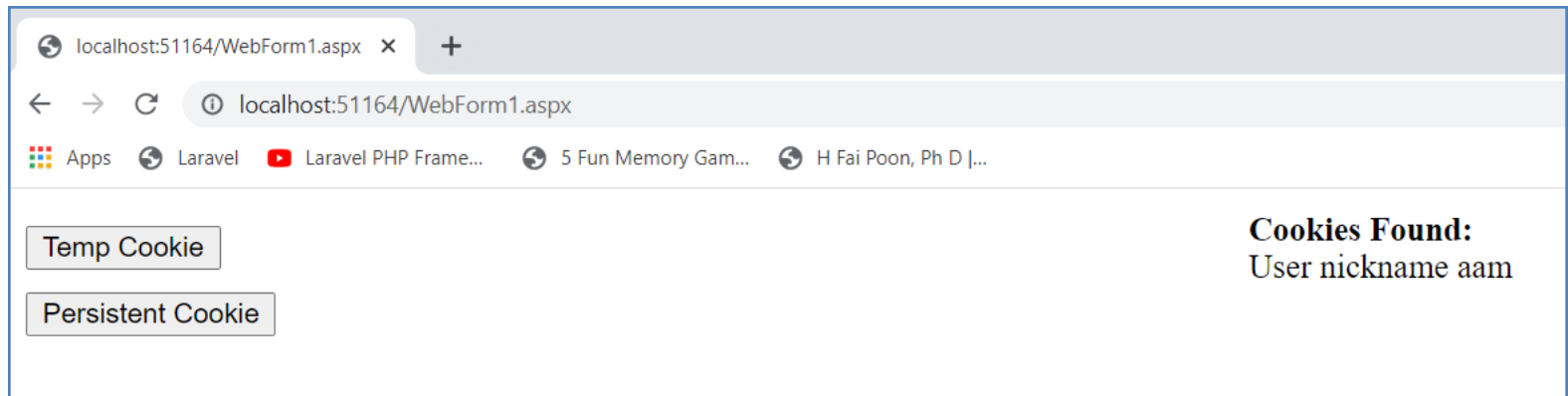
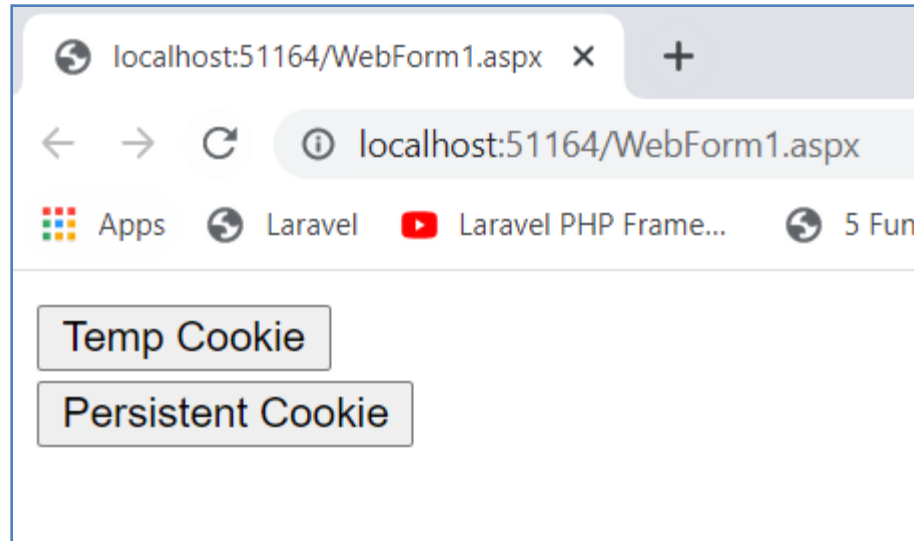
    //What happens if following code is written in Page Load event:::think
    HttpCookie cookies = Request.Cookies["Preferences"];
    if (cookies == null)
    {
        lblStatus.Text = "<b>No Preferences</b>";
    }
    else
    {
        lblStatus.Text = "<b>Cookies Found: </b><br/>";
        lblStatus.Text += "Language: " + cookies["language"] + " Country: " + cookies["country"];
    }
}
```



```
protected void btnPersistentCookie_Click(object sender, EventArgs e)
{
    //Check for cookie, create new only if does not exist

    HttpCookie cookie = Request.Cookies["UName"];
    if (cookie == null)
    {
        cookie = new HttpCookie("UName");
    }
    cookie["nickname"] = "aam";
    cookie.Expires = DateTime.Now.AddMinutes(1);
    Response.Cookies.Add(cookie);

    ////What happens if following code is written in Page Load event:::think
    HttpCookie cookies = Request.Cookies["UName"];
    if (cookies == null)
    {
        lblStatus.Text = "<b>Default user</b>";
    }
    else
    {
        lblStatus.Text = "<b>Cookies Found: </b><br/>";
        lblStatus.Text += "User nickname " + cookies["nickname"];
    }
}
```



Server Side

- Techniques
 - Application State
 - **Session State**
- Better security
- More involved and Large State
 - Reduced bandwidth
- Global State

Session State

- User specific state that is stored by server
 - Available only to pages accessed by a single user
- ASP.NET tracks each session by using a unique 120-bit identifier
- Client, SessionId, Server, Session
- SessionId
 - Cookies, URL
- `System.Web.SessionState.HttpSessionState`

Cont.

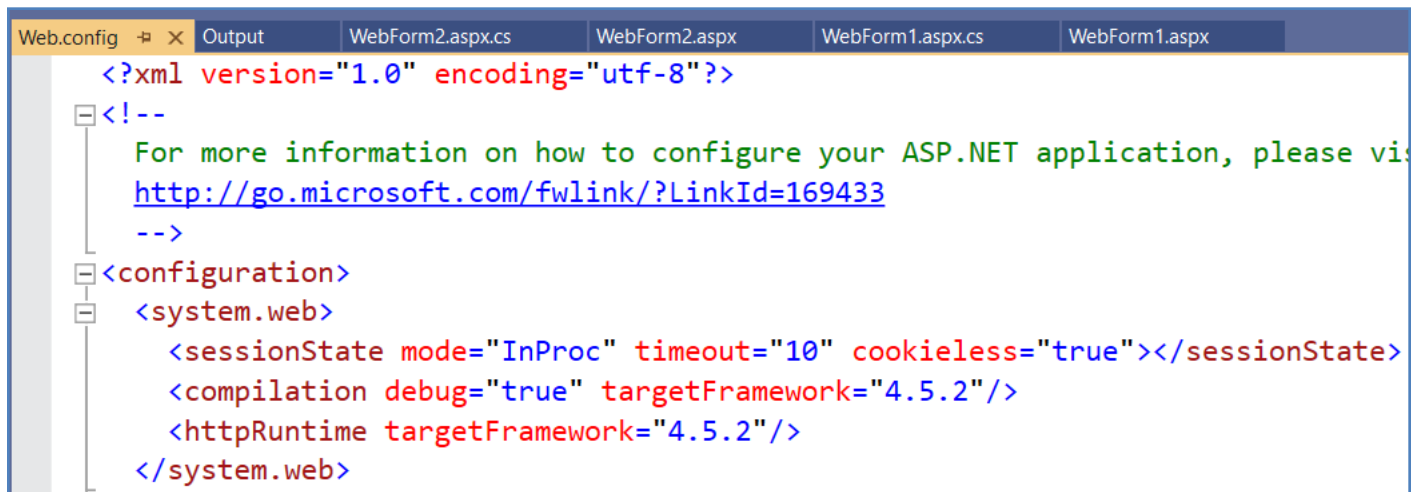
- Loss of session state
 - user closes and restarts the browser.
 - user accesses the same page through a different browser window
 - session times out due to inactivity.
 - `Session.Abandon()`

Cont.

- *HttpSessionState Members*
 - Count
 - IsCookieless
 - Keys
 - Mode
 - SessionID
 - Timeout
 - Abandon()
 - Clear()
- Web.config

Cookieless

- `cookieless="true"`
 - The session id is part of URL and is sent back and forth between the client and web server, with every request and response.
 - The web browser uses the session-id from the URL, to identify request has come from the same user or a different user.
- For cookieless session to work correctly relative URL must be used in the application when redirecting users to different web forms.



```
<?xml version="1.0" encoding="utf-8"?>
<!--
For more information on how to configure your ASP.NET application, please visit
http://go.microsoft.com/fwlink/?LinkId=169433
-->
<configuration>
  <system.web>
    <sessionState mode="InProc" timeout="10" cookieless="true"></sessionState>
    <compilation debug="true" targetFramework="4.5.2"/>
    <httpRuntime targetFramework="4.5.2"/>
  </system.web>
</configuration>
```

```

<tr>
    <td>Name:</td>
    <td>
        <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
    </td>
</tr>
<tr>
    <td>Email:</td>
    <td>
        <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
    </td>
</tr>
<tr>
    <td>&nbsp;</td>
    <td>&nbsp;</td>
</tr>
<tr>
    <td>
        <asp:Button ID="btnSubmit" runat="server" OnClick="btnSubmit_Click"
            Text="Submit" />
    </td>
    <td>&nbsp;</td>
</tr>

```

```

protected void btnSubmit_Click(object sender, EventArgs e)
{
    //Session is stored and accessed across web pages
    Session["Name"] = txtName.Text;
    Session["Email"] = txtEmail.Text;
    //Relative URL
    Response.Redirect("~/WebForm2.aspx");
}

```

```

<tr>
    <td>Name:</td>
    <td>
        <asp:Label ID="lblName" runat="server"></asp:Label>
    </td>
</tr>
<tr>
    <td>Email:</td>
    <td>
        <asp:Label ID="lblEmail" runat="server"></asp:Label>
    </td>
</tr>

```

```

protected void Page_Load(object sender, EventArgs e)
{
    if (Session["Name"] != null)
    {
        lblName.Text = Session["Name"].ToString();
    }
    if (Session["Email"] != null)
    {
        lblEmail.Text = Session["Email"].ToString();
    }
}

```

localhost:49352/(S(bzyy3lul3vtpi4 x +

localhost:49352/(S(bzyy3lul3vtpi4mcjl3vaynd))/WebForm1.aspx

Apps Laravel Laravel PHP Frame... 5 Fun Memory Gam... H Fai Poon, I

Name:

Email:

localhost:49352/(S(bzyy3lul3vtpi4 x +

localhost:49352/(S(bzyy3lul3vtpi4mcjl3vaynd))/WebForm1.aspx

Apps Laravel Laravel PHP Frame... 5 Fun Memory Gam... H Fai Poon

Name:

Email:



The screenshot shows a web browser window with a single tab. The address bar displays the URL `localhost:49352/(S(bzyy3lul3vtpi4mcjl3vaynd))/We...`. The browser's toolbar includes back, forward, and refresh buttons, along with search, star, and extension icons. Below the toolbar, a row of bookmarks is visible, including 'Apps', 'Laravel', 'Laravel PHP Frame...', '5 Fun Memory Gam...', and 'H Fai Poon, Ph D |...'. The main content area of the browser displays a paragraph of text explaining cookieless sessions, followed by a form with two fields: 'Name' and 'Email'.

localhost:49352/(S(bzyy3lul3vtpi4 x +

localhost:49352/(S(bzyy3lul3vtpi4mcjl3vaynd))/We... 🔍 ☆ 📄 ⚙️ 👤 ⋮

Apps 🔄 Laravel 📺 Laravel PHP Frame... 🔄 5 Fun Memory Gam... 🔄 H Fai Poon, Ph D |...

When cookieless sessions are enabled. The session id is part of URL and is sent back and forth between the client and web server, with every request and response. The web browser uses the session-id from the URL, to identify request has come from the same user or a different user. For cookieless session to work correctly relative URL must be used in the application when redirecting users to different web forms.

Name: test

Email: test@test.test

Session Mode

- InProc (inside process)
 - Session state variables are stored on web server memory inside asp.net worker process
 - Easy to implement; mode="InProc"
 - Perform best, as session state memory is kept on the web server
 - Suitable for web application hosted on a single web server
 - Objects can be added w/o serialization

Cont.

- InProc
 - Session state data is lost, when the worker process or application pool is recycled
 - Not suitable for web farms and web gardens
 - Scalability could be an issue

```
<configuration>  
  <system.web>  
    <sessionState mode="InProc" timeout="3"></sessionState>  
    <compilation debug="true" targetFramework="4.5.2"/>  
    <httpRuntime targetFramework="4.5.2"/>  
  </system.web>
```

```
<td colspan="2">
    <asp:Label ID="lblInfo" runat="server">Check Web.Config</asp:Label>
</td>
r>
>
<td>Name:</td>
<td>
    <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
</td>
r>
>
<td>Email:</td>
<td>
    <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
</td>
r>
>
<td>&nbsp;</td>
<td>&nbsp;</td>
r>
>
<td>
    <asp:Button ID="btnSubmit" runat="server" OnClick="btnSubmit_Click" Text="Submit" />
</td>
```

```
protected void Page_Load(object sender, EventArgs e)
{
    lblInfo.Text = "Sessions state variables are stored in web server by default. " +
        "Kept for the life time of a session. Default Session state mode is InProc. " +
        "Demonstrate: Cookies enabled true and false in browser.";
}
```

0 references

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    //Session is stored and accessed across web pages
    Session["Name"] = txtName.Text;
    Session["Email"] = txtEmail.Text;

    Response.Redirect("~/WebForm2.aspx");
}
```

```
<tr>
    <td colspan="2">
        <asp:Label ID="lblInfo" runat="server"></asp:Label>
    </td>
</tr>
<tr>
    <td>Name:</td>
    <td>
        <asp:Label ID="lblName" runat="server"></asp:Label>
    </td>
</tr>
<tr>
    <td>Email:</td>
    <td>
        <asp:Label ID="lblEmail" runat="server"></asp:Label>
    </td>
</tr>
```

```
public partial class WebForm2 : System.Web.UI.Page
{
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
        lblInfo.Text = "Check for Web.Config";
        if(Session["Name"]!=null)
        {
            lblName.Text = Session["Name"].ToString();
        }
        if (Session["Email"] != null)
        {
            lblEmail.Text = Session["Email"].ToString();
        }
    }
}
```

localhost:49443/WebForm1 x

Settings - Cookies and other x

+

—

□

×

←

→

↻

localhost:49443/WebForm1.aspx

🔍

☆

📄

⚙️

👤

⋮

🌐 Apps

🌐 Laravel

📺 Laravel PHP Frame...

🌐 5 Fun Memory Gam...

Sessions state variables are stored in web server by default. Kept for the life time of a session. Default Session state mode is InProc. Demonstrate: Cookies enabled true and false in browser.

Name:

Email:

localhost:49443/WebForm1 x

Settings - Cookies and other x

+

—

□

×

←

→

↻

localhost:49443/WebForm1.aspx

🔍

☆

📄

⚙️

👤

⋮

🌐 Apps

🌐 Laravel

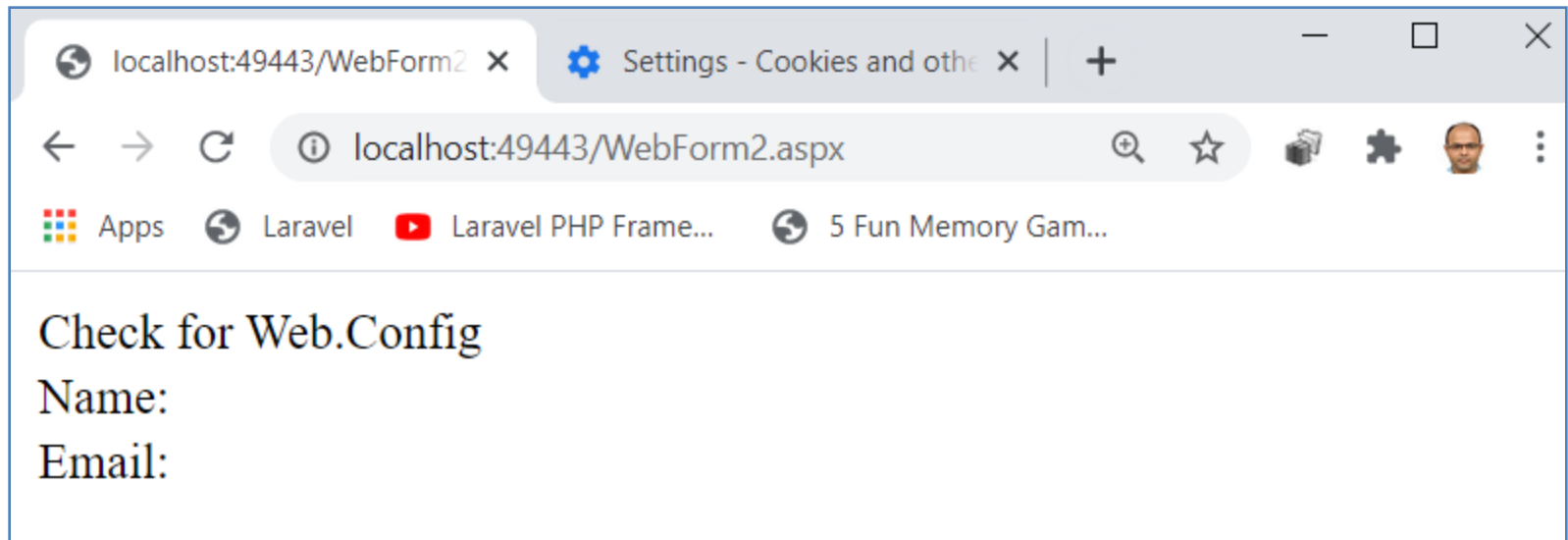
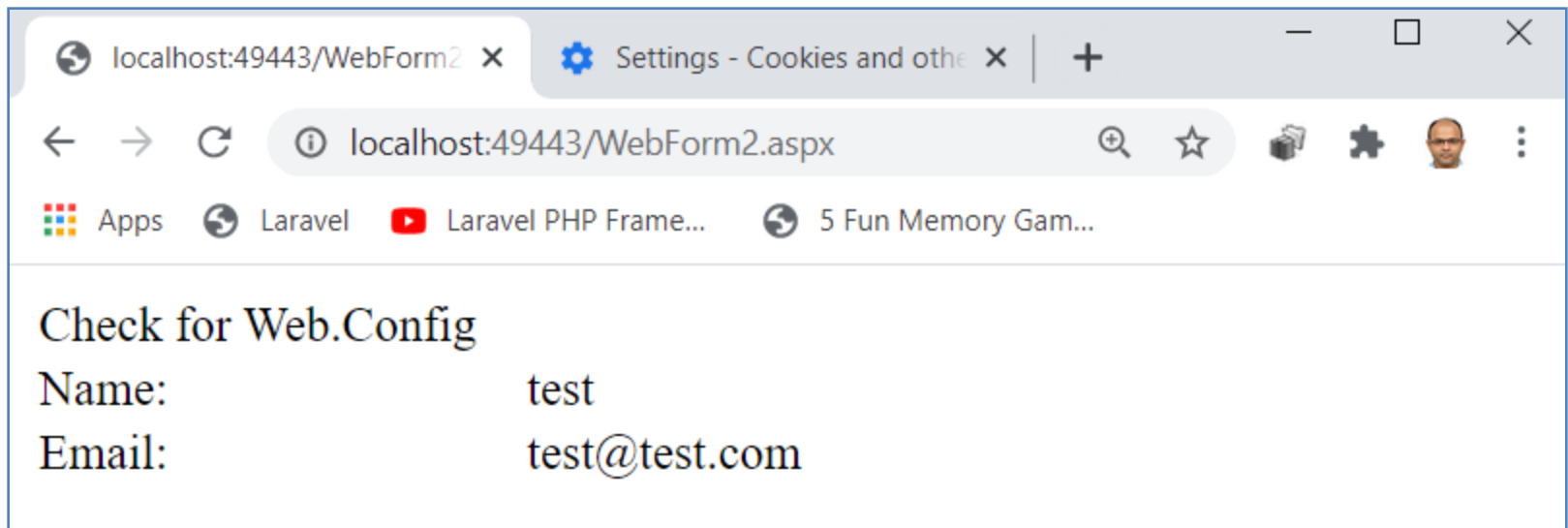
📺 Laravel PHP Frame...

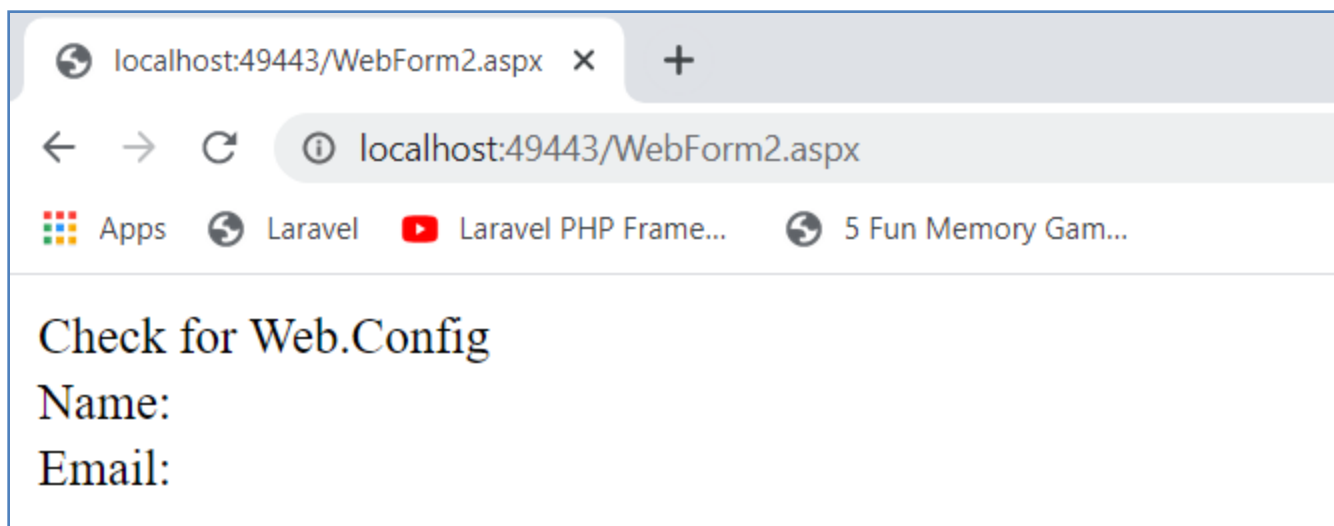
🌐 5 Fun Memory Gam...

Sessions state variables are stored in web server by default. Kept for the life time of a session. Default Session state mode is InProc. Demonstrate: Cookies enabled true and false in browser.

Name:

Email:





Session Mode

- StateServer
 - Session state variables are stored in a process, **asp.net state** service
 - **Start asp.net state service manually**
 - Can be present on a web server or a dedicated machine
 - ASP.NET worker process independent
 - Survives worker process restarts
 - Can be used with web farm and web garden
 - Offers more scalability than InProc

Cont.

- StateServer is slower than InProc
- Complex objects needed to be serialized and deserialized
- State server on dedicated machine, if it goes down/ restarts all session variables will be lost
 - Single point of failure

```
<configuration>
  <system.web>
    <sessionState mode="StateServer"
                  stateConnectionString="tcpip=localhost:42424"
                  timeout="20"></sessionState>
    <compilation debug="true" targetFramework="4.5.2"/>
    <httpRuntime targetFramework="4.5.2"/>
  </system.web>
```

```
<tr>
  <td colspan="2">
    <asp:Label ID="lblInfo" runat="server">Check Web.Config</asp:Label>
  </td>
</tr>
<tr>
  <td>Name:</td>
  <td>
    <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
  </td>
</tr>
<tr>
  <td>Email:</td>
  <td>
    <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
  </td>
</tr>
```

```
public partial class WebForm1 : System.Web.UI.Page
{
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
    }

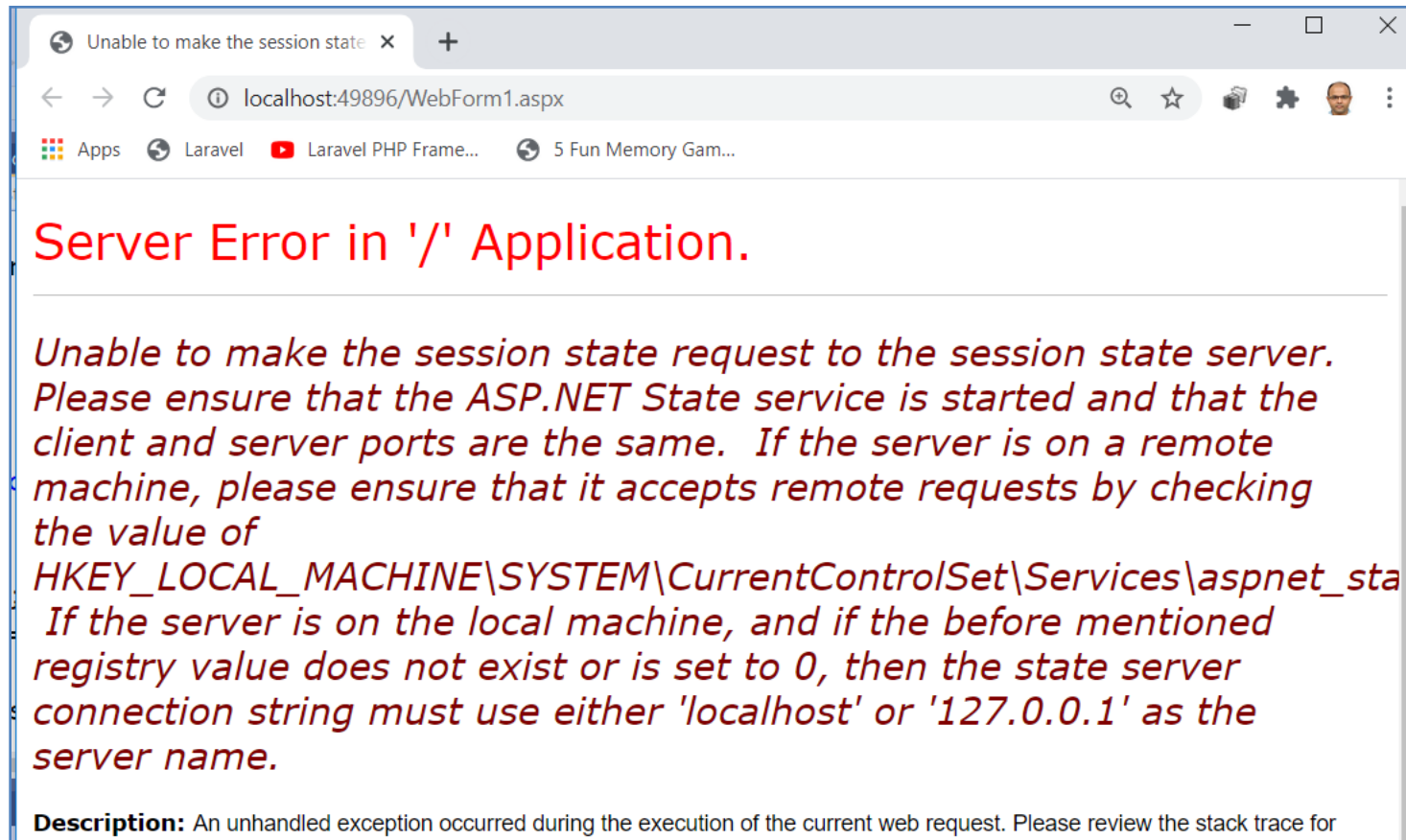
    0 references
    protected void btnSubmit_Click(object sender, EventArgs e)
    {
        //Session is stored and accessed across web pages
        Session["Name"] = txtName.Text;
        Session["Email"] = txtEmail.Text;

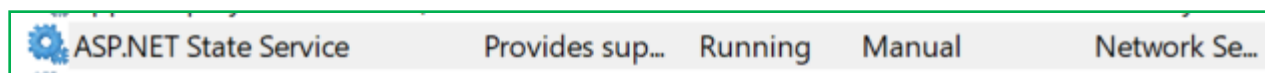
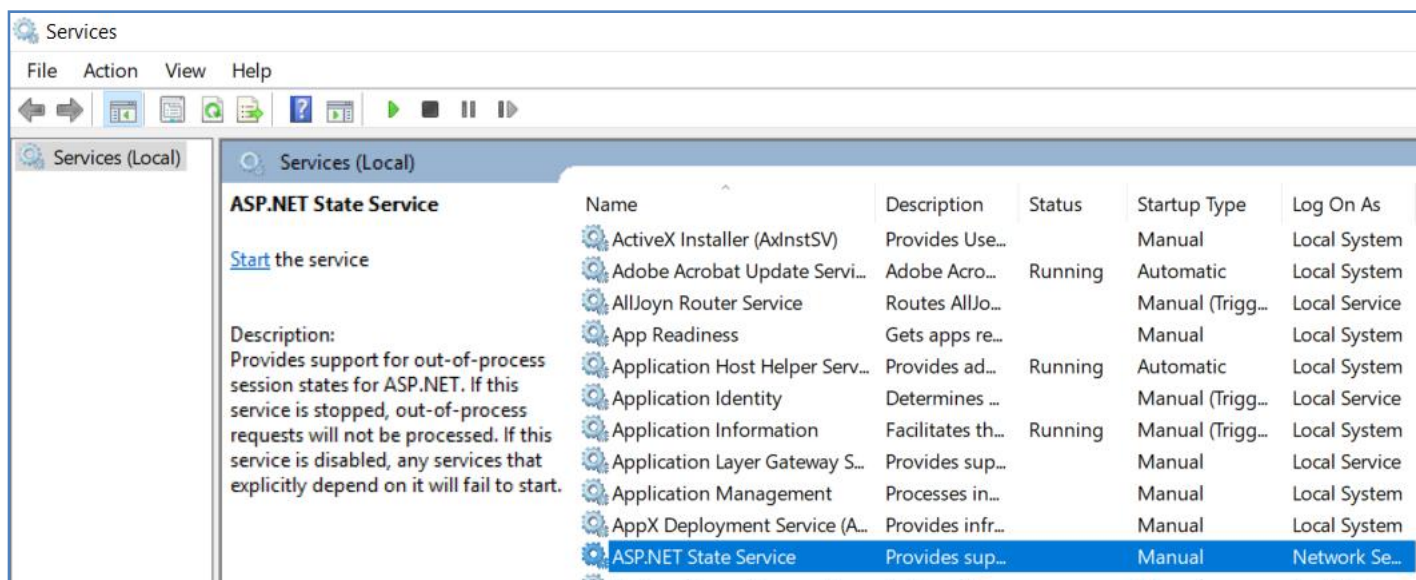
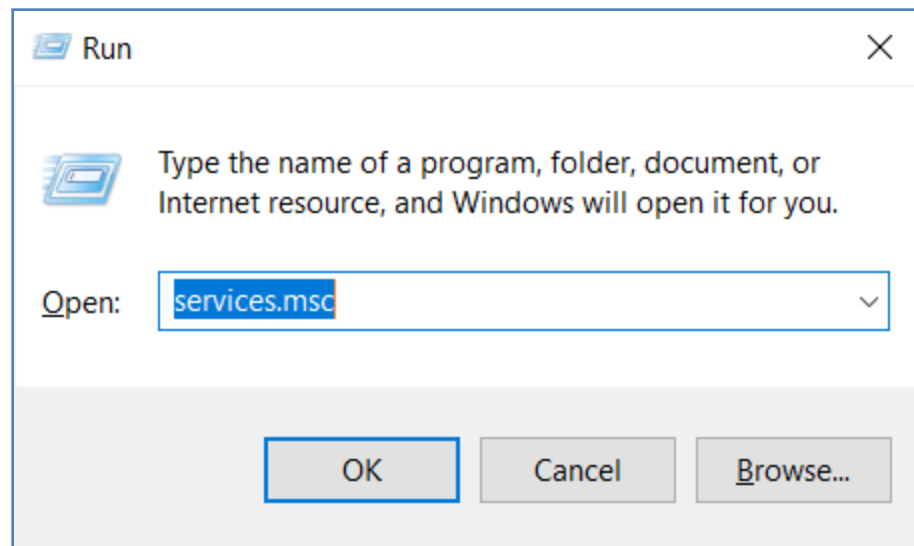
        Response.Redirect("~/WebForm2.aspx");
    }
}
```

```
<tr>
    <td colspan="2">
        <asp:Label ID="lblInfo" runat="server"></asp:Label>
    </td>
</tr>
<tr>
    <td>Name:</td>
    <td>
        <asp:Label ID="lblName" runat="server"></asp:Label>
    </td>
</tr>
<tr>
    <td>Email:</td>
    <td>
        <asp:Label ID="lblEmail" runat="server"></asp:Label>
    </td>
</tr>
```



```
public partial class WebForm2 : System.Web.UI.Page
{
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
        lblInfo.Text = "Check for Web.Config and delete woker process and run current page." +
            "Observe output.";
        if (Session["Name"] != null)
        {
            lblName.Text = Session["Name"].ToString();
        }
        if (Session["Email"] != null)
        {
            lblEmail.Text = Session["Email"].ToString();
        }
    }
}
```





localhost:49896/WebForm1.aspx x +

← → ↻ ⓘ localhost:49896/WebForm1.aspx

Apps Laravel Laravel PHP Frame... 5 Fun Memory Gam...

Check Web.Config

Name:

Email:

localhost:49896/WebForm2.aspx x +

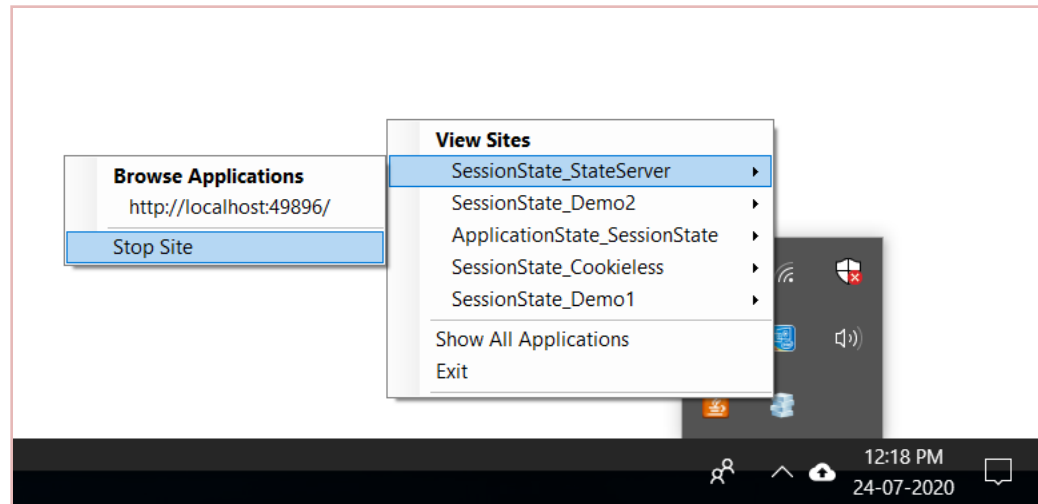
← → ↻ ⓘ localhost:49896/WebForm2.aspx 🔍 ☆

Apps Laravel Laravel PHP Frame... 5 Fun Memory Gam...

Check for Web.Config and delete woker process and run current page.Observe output.

Name: test

Email: test@test.test



localhost:49896/WebForm2.aspx x +

← → ↻ ⓘ localhost:49896/WebForm2.aspx 🔍 ☆

Apps ↻ Laravel 📺 Laravel PHP Frame... ↻ 5 Fun Memory Gam...

Check for Web.Config and delete woker process and run current page.Observe output.

Name: test

Email: test@test.test

Session Mode

- SQLServer
 - Session state variables are stored in SQLServer database
 - Most reliable
 - Can be used with web farms and web gardens
 - Most scalable
 - Slowest
 - Complex objects needs to be serialized and deserialized

SessionState: Mode

- Off-Disables session state for the entire application
- InProc
- StateServer
- SQLServer
- Custom-Build your own Session State provider

Cross Page Posting

- ViewState
 - Tightly bound to a single page
- Cross Page Posting
 - Extends the postback
 - User data from one page to another page
 - PostBackURL
- Page.PreviousPage
 - FindControl()
- Access to cross-page posted data through strongly typed properties


```
<tr>
  <td>Firstname</td>
  <td>
    <asp:TextBox ID="txtFirstname" runat="server"></asp:TextBox>
  </td>
</tr>
<tr>
  <td>Lastname</td>
  <td>
    <asp:TextBox ID="txtLastname" runat="server"></asp:TextBox>
  </td>
</tr>
<tr>
  <td>Age</td>
  <td>
    <asp:TextBox ID="txtAge" runat="server"></asp:TextBox>
  </td>
</tr>
<tr>
  <td>
    <asp:Button ID="txtSubmit" runat="server"PostBackUrl="~/WebForm2.aspx"
      Text="Cross Page Posting" />
  </td>
</tr>
```

```
public partial class WebForm1 : System.Web.UI.Page
{
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
    }

    1 reference
    public TextBox Firstname
    {
        get { return txtFirstname; }
    }

    1 reference
    public TextBox Lastname
    {
        get { return txtLastname; }
    }
}
```

```
<td>
    <asp:Label ID="lblStatus" runat="server"></asp:Label>
</td>
```

```
public partial class WebForm2 : System.Web.UI.Page
{
    0 references
    protected void Page_Load(object sender, EventArgs e)
    {
        if(PreviousPage!=null)
        {
            WebForm1 prev = PreviousPage as WebForm1;
            if(prev!=null)
            {
                lblStatus.Text = "You came from a page titled " + prev.Title +
                    " " + prev.Firstname.Text + " " + prev.Lastname.Text + " ";
                lblStatus.Text += Server.HtmlEncode(((TextBox)prev.FindControl("txtAge")).Text);
            }
        }
    }
}
```

WebForm1 x +

localhost:50487/WebForm1.aspx

Apps Laravel Laravel PHP Frame... 5 Fun Memory Gam... H Fai Poon, Ph D |...

Firstname

Lastname

Age

Cross Page Posting

WebForm1 x +

localhost:50487/WebForm1.aspx

Apps Laravel Laravel PHP Frame... 5 Fun Memory Gam... H Fai Poon, Ph D |...

Firstname

Lastname

Age

Cross Page Posting



	View State	Query String	Custom Cookies
Allowed Data Types	All serializable .NET data types.	A limited amount of string data.	String data.
Storage Location	A hidden field in the current web page.	The browser's URL string.	The client's computer (in memory or a small text file, depending on its lifetime settings).
Lifetime	Retained permanently for postbacks to a single page.	Lost when the user enters a new URL or closes the browser. However, this can be stored in a bookmark.	Set by the programmer. Can be used in multiple pages and can persist between visits.
Scope	Limited to the current page.	Limited to the target page.	The whole ASP.NET application.
Security	Tamperproof by default but easy to read. You can enforce encryption by using the ViewStateEncryptionMode property of the Page directive.	Clearly visible and easy for the user to modify.	Insecure, and can be modified by the user.
Performance Implications	Slow if a large amount of information is stored, but will not affect server performance.	None, because the amount of data is trivial.	None, because the amount of data is trivial.
Typical Use	Page-specific settings.	Sending a product ID from a catalog page to a details page.	Personalization preferences for a website.

	Session State	Application State
Allowed Data Types	All .NET data types for the default in-process storage mode. All serializable .NET data types if you use an out-of-process storage mode.	All .NET data types.
Storage Location	Server memory, state service, or SQL Server, depending on the mode you choose.	Server memory.
Lifetime	Times out after a predefined period (usually 20 minutes, but can be altered globally or programmatically).	The lifetime of the application (typically, until the server is rebooted).
Scope	The whole ASP.NET application.	The whole ASP.NET application. Unlike other methods, application data is global to all users.
Security	Very secure, because data is never transmitted to the client.	Very secure, because data is never transmitted to the client.
Performance Implications	Slow when storing a large amount of information, especially if there are many users at once, because each user will have their own copy of session data.	Slow when storing a large amount of information, because this data will never time out and be removed.
Typical Use	Storing items in a shopping basket.	Storing any type of global data.

Use client-side state management
when _____ is the top priority.
Use server-side state management
when _____ or when _____ is a
significant issue.

ASP.NET uses _____ by default to store information about controls in a web form. You can add custom values to _____ by accessing the _____ collection.

Use _____ to store data in forms when
view state is disabled. _____
values are available to users as
plaintext in the HTML.

On the client, _____ store data that the web browser submits with every webpage request. Use _____ to track users across multiple webpages.

_____ store small pieces of information in a hyperlink's URL. Use _____ when you want state management data to be bookmarked, such as when displaying multiple pages of search results.

You need to store a user's user name and password as he or she navigates to different pages on your site, so that you can pass those credentials to the server.
Which type of state management should you use?

- A. Client-side state management
- B. Server-side state management

You need to track non-confidential user preferences when a user visits your site, to minimize additional load on your servers. You distribute requests among multiple web servers, each running a copy of your application. Which type of state management should you use?

- A. Client-side state management
- B. Server-side state management

You are creating an ASP.NET webpage that allows a user to browse information in a database. While the user accesses the page, you need to track search and sorting values. You do not need to store the information between visits to the webpage. Which type of client-side state management would meet your requirements and be the simplest to implement?

- A. View state
- B. Control state
- C. Hidden fields
- D. Cookies
- E. Query strings

You are creating an ASP.NET website with dozens of pages. You want to allow the user to set user preferences and have each page process the preference information. You want the preferences to be remembered between visits, even if the user closes the browser. Which type of client-side state management meets your requirements and is the simplest to implement?

- A. View state
- B. Control state
- C. Hidden fields
- D. Cookies
- E. Query strings

You are creating an ASP.NET web form that searches product inventory and displays items that match the user's criteria. You want users to be able to bookmark or send search results in email. Which type of client-side state management meets your requirements and is the simplest to implement?

- A. View state
- B. Control state
- C. Hidden fields
- D. Cookies
- E. Query strings

Which typically consumes more server memory: application state or session state?

Which might not work if a user has disabled cookies in his or her web browser: application state or session state?

You need to store state data that is accessible to any user who connects to your web application and ensure that it stays in memory. Which collection object should you use?

- A. Session
- B. Application
- C. Cookies
- D. ViewState

You need to store a value indicating whether a user has been authenticated for your site. This value needs to be available and checked on every user request. Which object should you use?

- A. Session
- B. Application
- C. Cookies
- D. ViewState

Your application is being deployed in a load-balanced web farm. The load balancer is not set up for user server affinity. Rather, it routes requests to servers based on their load. Your application uses session state. How should you configure the SessionState mode attribute? (Choose all that apply.)

- A. StateServer
- B. InProc
- C. Off
- D. SqlServer