C# & .NET Framework

Chapter 24:
ASP.NET 2.0 Web Applications

Review

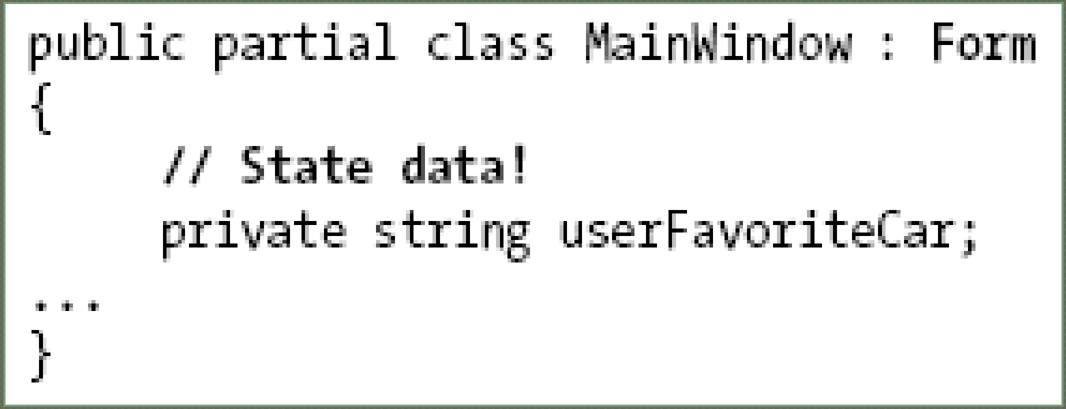
- HTTP: a protocol for transferring HTML document
- HTML: a markup language
- Client-Side Scripting: JavaScript, VBScript
- GET and POST method
- ASP.NET Web Page Code Model: Single and Code-Behind
- HTTP Request + HTTP Response
- Building a Simple ASP.NET 2.0 Website: Master page, Menu, GridView, Wizard
- Validation Controls: RequiredFieldValidator, RegularExpressionValidator, RangeValidator, CompareValidator

Chapter 24: Objectives

- The Issue of State
- ASP.NET State Management Techniques
- Understanding the Role of ASP.NET View State
- The Role of the Global.asax File
- Understanding the Application/Session Distinction
- Working with the Application Cache
- Maintaining Session Data
- Understanding Cookies
- Configuring Your ASP.NET Web Application Using Web.config
- Configuration Inheritance

The Issue of State

- When building a Windows Forms application, any member variables defined in the Form-derived class will typically exist in memory until the user explicitly shuts down the executable:

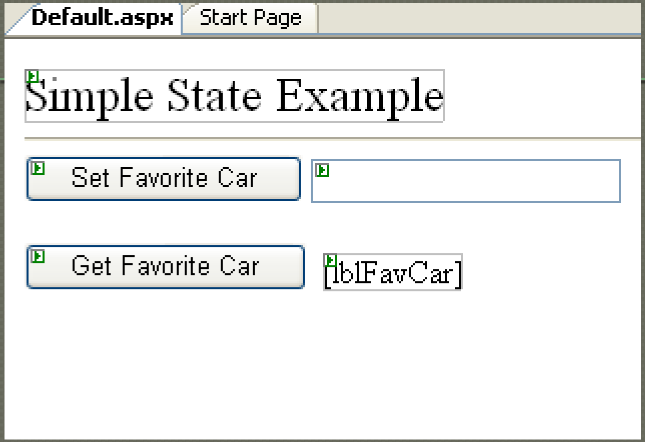


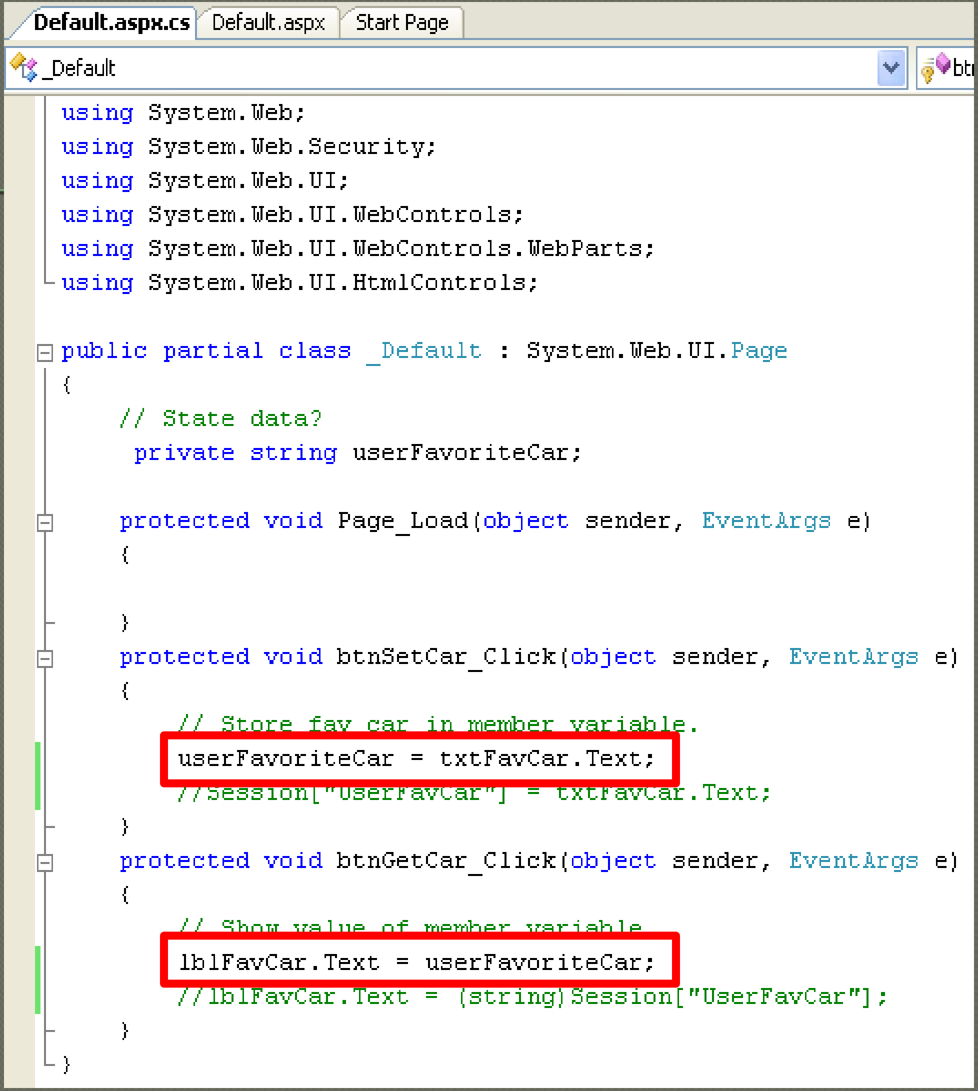
```
public partial class MainWindow : Form
{
      // State data!
      private string userFavoriteCar;
...
}
```

The Issue of State

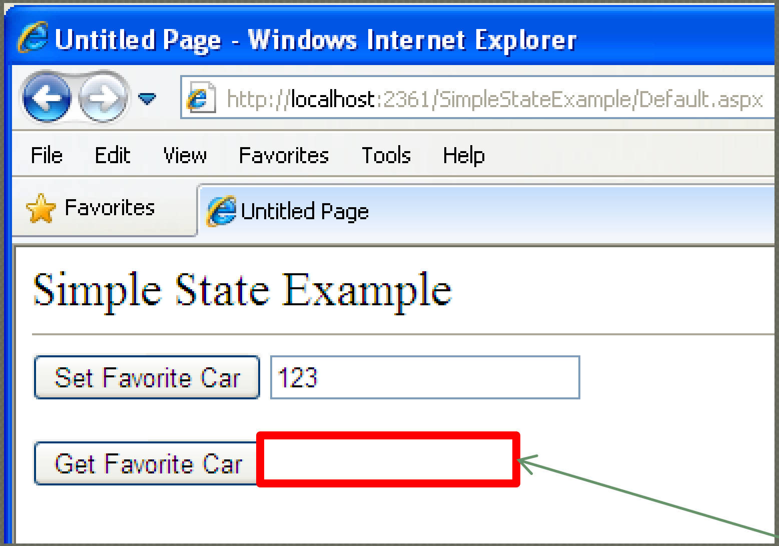
- HTTP has no clue how to automatically remember data once the HTTP response has been sent, it stands to reason that the Page object is destroyed instantly. Therefore, when the client posts back to the *.aspx file, a *new* Page object is constructed that will reset any page-level member variables.
- It is the same to Java servlets, CGI applications, classic ASP, and PHP.

Example: SimpleStateExample





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

public partial class _Default : System.Web.UI.Page
{
    // State data?
     private string userFavoriteCar;

    protected void Page_Load(object sender, EventArgs e)
    {

    }
    protected void btnSetCar_Click(object sender, EventArgs e)
    {
        // Store fav car in member variable.
        userFavoriteCar = txtFavCar.Text;
        //Session["UserFavCar"] = txtFavCar.Text;
    }
    protected void btnGetCar_Click(object sender, EventArgs e)
    {
        // Show value of member variable
        lblFavCar.Text = userFavoriteCar;
        //lblFavCar.Text = (string)Session["UserFavCar"];
    }
}
```


Untitled Page - Windows Internet Explorer
http://localhost:2361/SimpleStateExample/Default.aspx
File Edit View Favorites Tools Help
Favorites
Untitled Page
Simple State Example
Set Favorite Car
123
Get Favorite Car

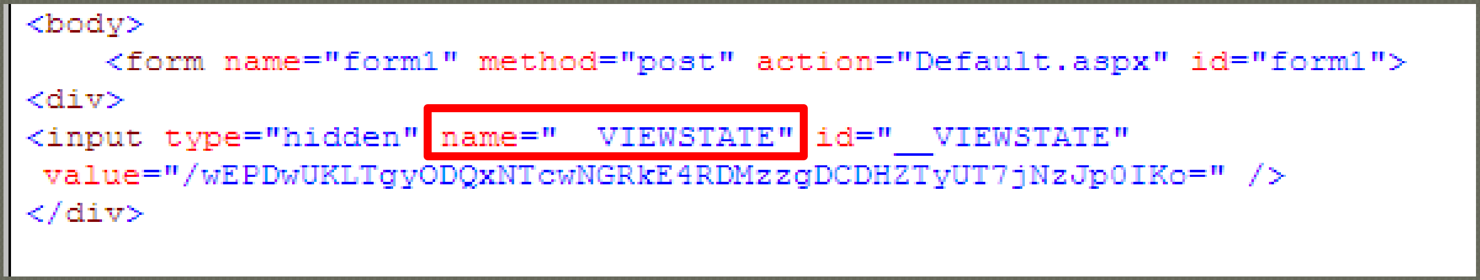
No value displayed here

ASP.NET State Management Techniques

- ASP.NET provides several mechanisms to maintain stateful information in your web applications:
  - Make use of ASP.NET view state.
  - Make use of ASP.NET control state.
  - Define application-level variables.
  - Make use of the cache object.
  - Define session-level variables.
  - Interact with cookie data.

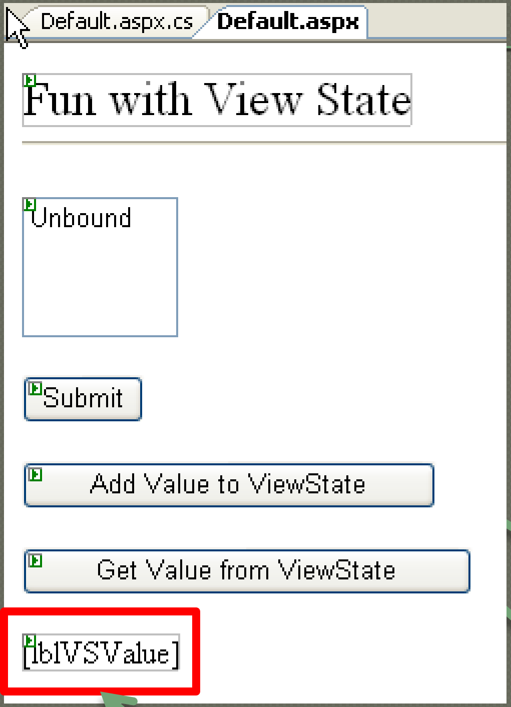
Understanding the Role of ASP.NET View State

- ASP.NET runtime will automatically embed a hidden form field (named __VIEWSTATE), which will flow between the browser and a specific page.
  - The data assigned to this field is a Base64-encoded string that contains a set of *name/value pairs that represent the values* of each GUI widget on the page at hand.
- Init event handler is the entity in charge of *reading the incoming values* found within the __VIEWSTATE field to populate the appropriate member variables in the derived class.

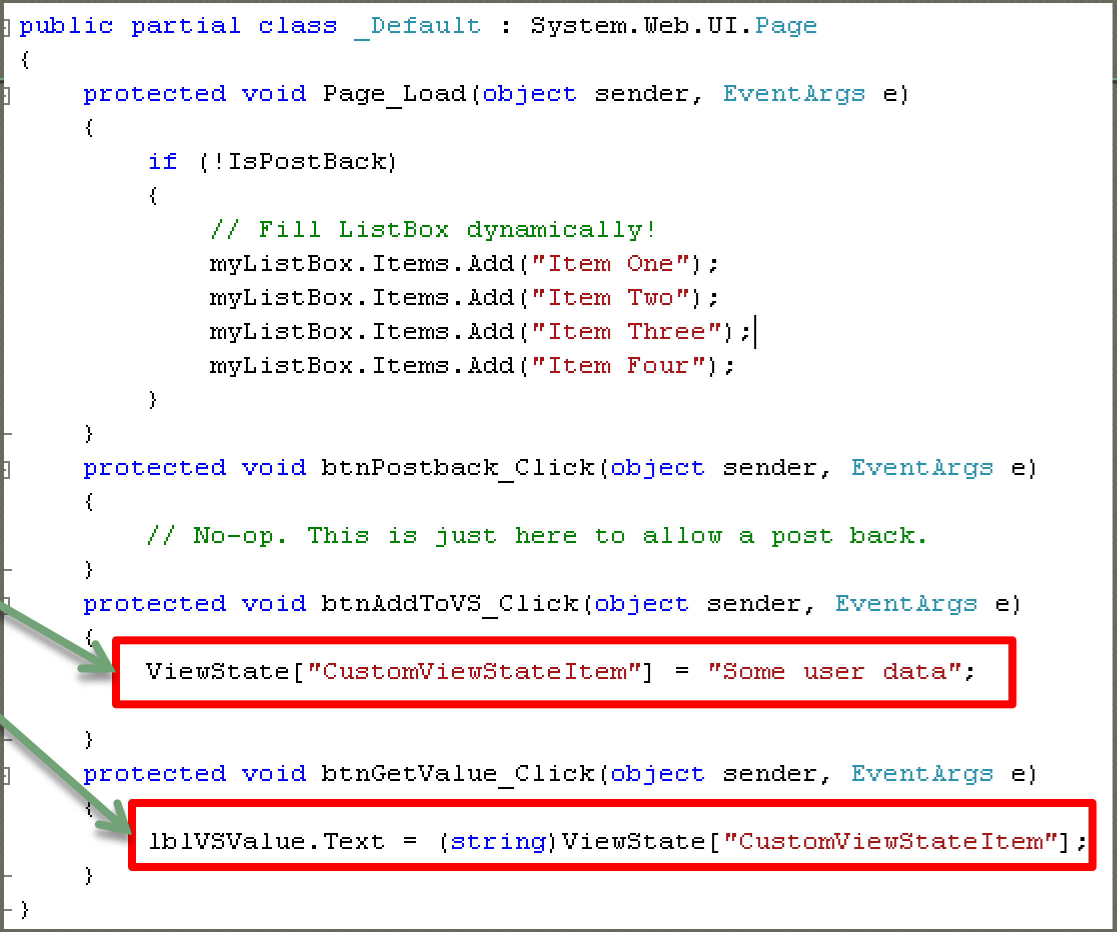


```
<body>
    <form name="form1" method="post" action="Default.aspx" id="form1">
<div>
<input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
 value="/wEPDwUKLTgyODQxNTcwNGRkE4RDMzzgDCDHZTyUT7jNzJp0IKo=" />
</div>
```

ViewState Example (Ch_24 code\ViewStateApp)



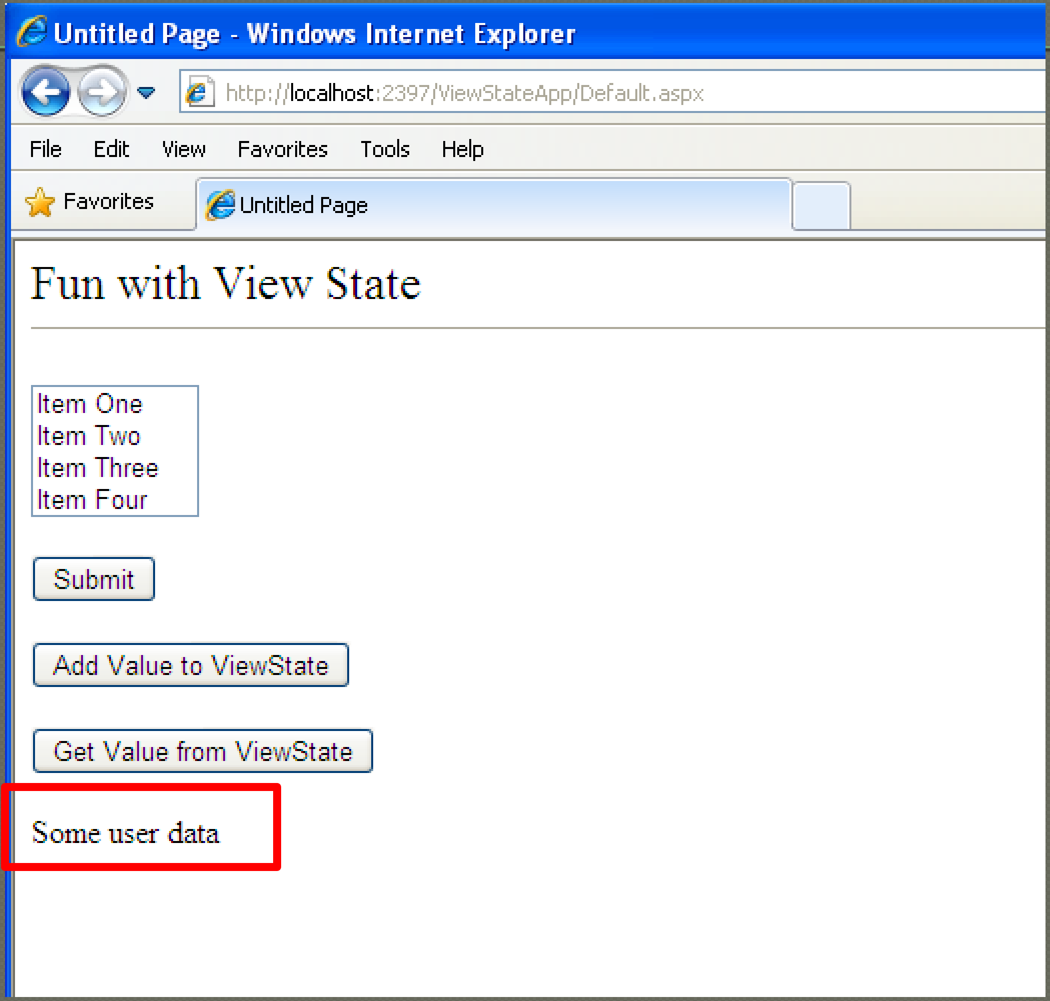
Fun with View State



```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            // Fill ListBox dynamically!
            myListBox.Items.Add("Item One");
            myListBox.Items.Add("Item Two");
            myListBox.Items.Add("Item Three");
            myListBox.Items.Add("Item Four");
        }
    }
    protected void btnPostback_Click(object sender, EventArgs e)
    {
        // No-op. This is just here to allow a post back.
    }
    protected void btnAddToVS_Click(object sender, EventArgs e)
    {
        ViewState["CustomViewStateItem"] = "Some user data";

    }
    protected void btnGetValue_Click(object sender, EventArgs e)
    {
        lblVSValue.Text = (string)ViewState["CustomViewStateItem"];
    }
}
```

Display value of ViewState

ViewState Example (Ch_24 code\ViewStateApp)


Untitled Page - Windows Internet Explorer
http://localhost:2397/ViewStateApp/Default.aspx
File Edit View Favorites Tools Help
Favorites
Untitled Page

Fun with View State

Item One
Item Two
Item Three
Item Four

Submit
Add Value to ViewState
Get Value from ViewState

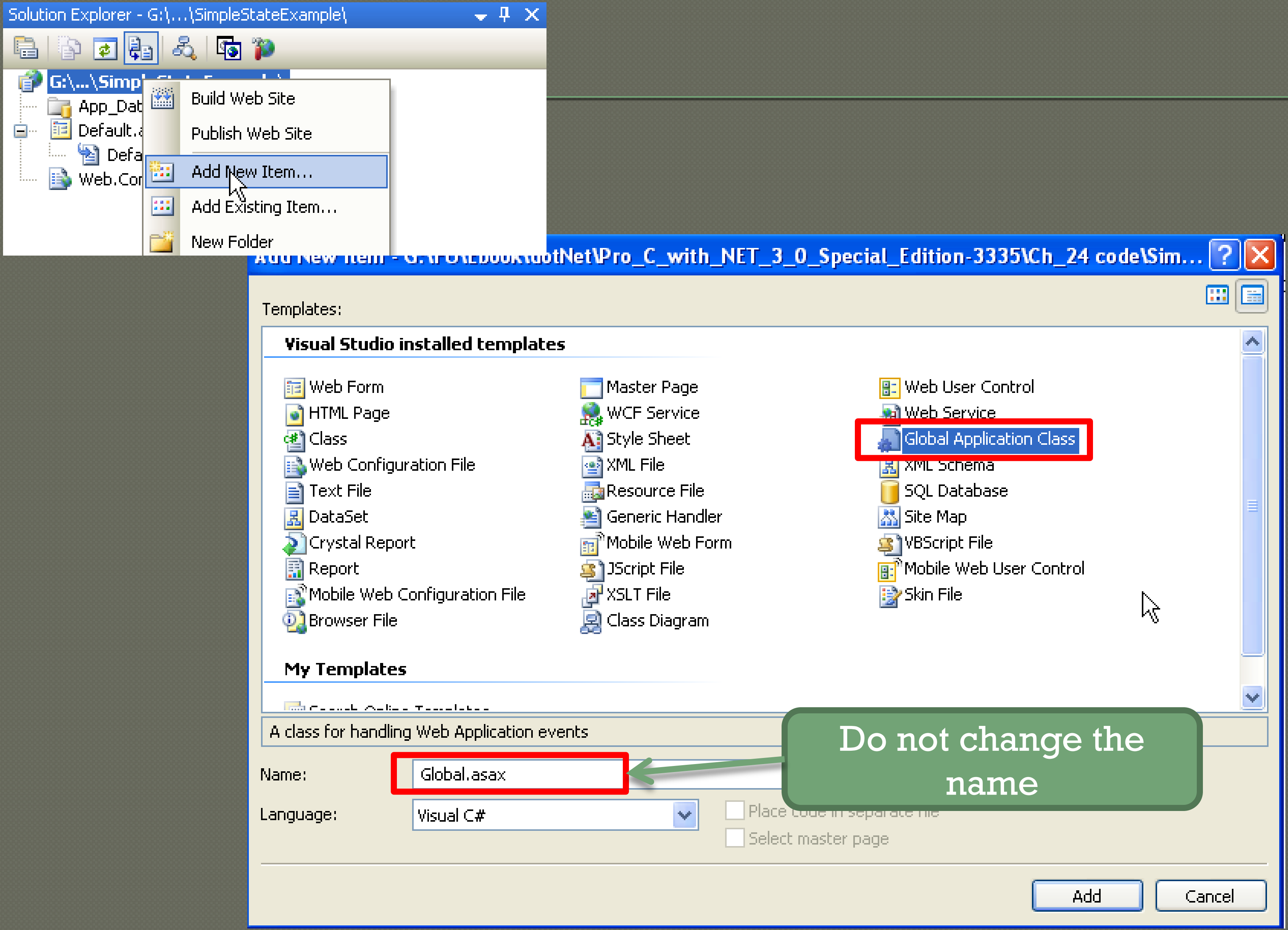
Some user data

The Role of the Global.asax File

- Each ASP.NET web application has one Global.asax file that contains some functions that can be invoke during that application life cycle

Event Handler	**Description**
Application_Start()	This event handler is called the very first time the web application is launched
Application_End()	This event handler is called when the application is shutting down
Session_Start()	This event handler is fired when a new user logs on to your application. Here you may establish any user-specific data points.
Session_End()	This event handler is fired when a user's session has terminated (typically through a predefined timeout).
Application_Error()	This is a global error handler that will be called when an unhandled exception is thrown by the web application.

Create Global.asax file



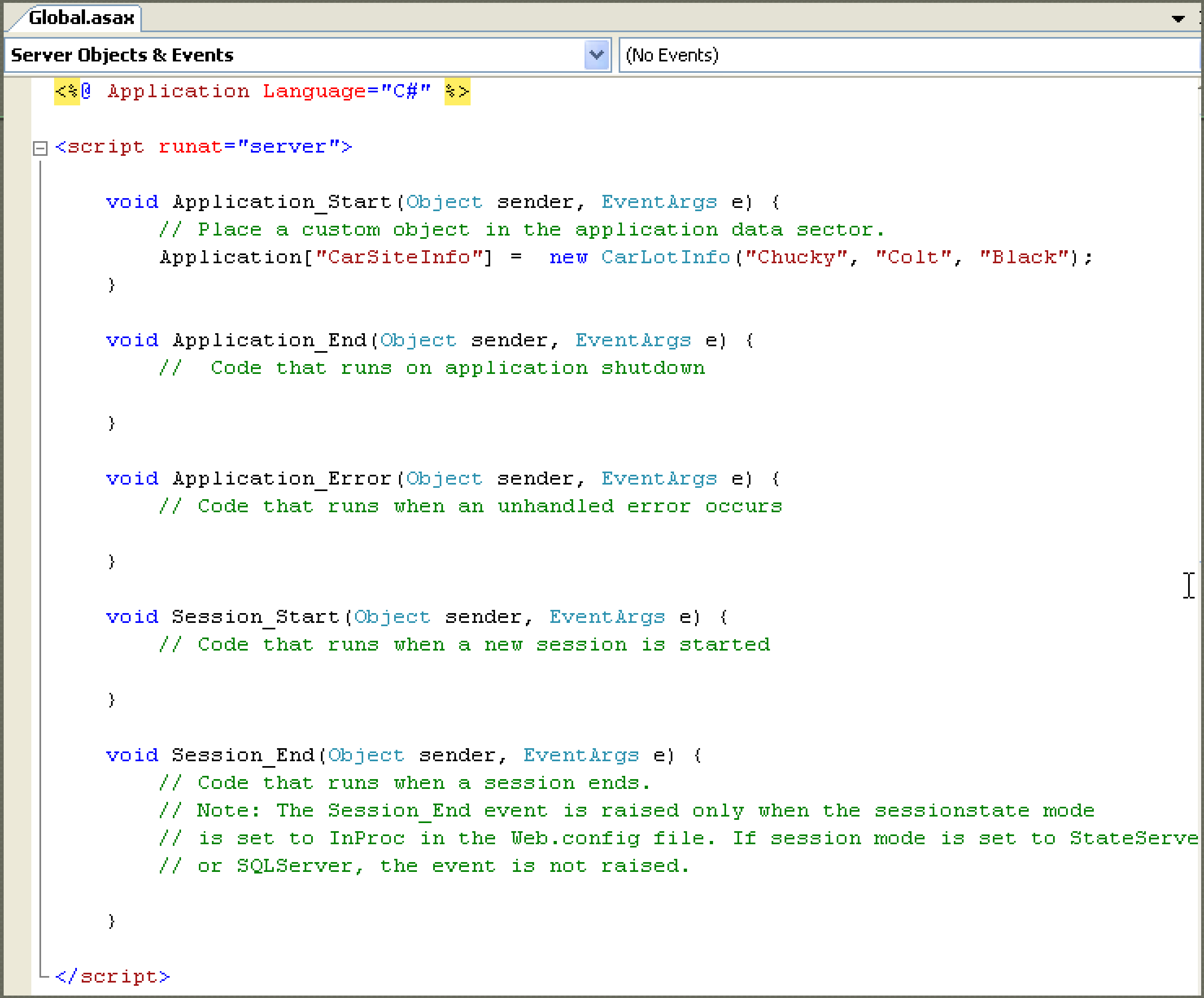
A class for handling Web Application events

Name: Global.asax

Language: Visual C#

Do not change the name

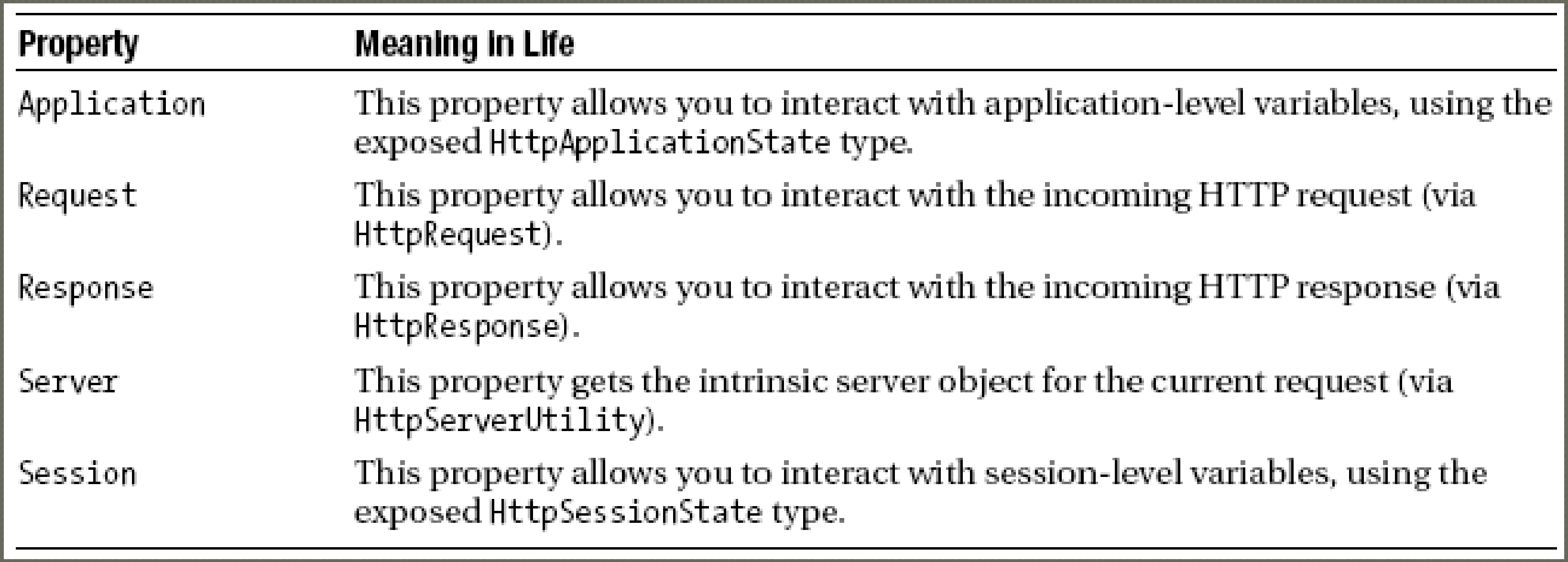
Create Global.asax file



```
<%@ Application Language="C#" %>

<script runat="server">

    void Application_Start(Object sender, EventArgs e) {
        // Place a custom object in the application data sector.
        Application["CarSiteInfo"] =  new CarLotInfo("Chucky", "Colt", "Black");
    }

    void Application_End(Object sender, EventArgs e) {
        //  Code that runs on application shutdown

    }

    void Application_Error(Object sender, EventArgs e) {
        // Code that runs when an unhandled error occurs

    }

    void Session_Start(Object sender, EventArgs e) {
        // Code that runs when a new session is started

    }

    void Session_End(Object sender, EventArgs e) {
        // Code that runs when a session ends.
        // Note: The Session_End event is raised only when the sessionstate mode
        // is set to InProc in the Web.config file. If session mode is set to StateServe
        // or SQLServer, the event is not raised.

    }

</script>
```

The HttpApplication Base Class

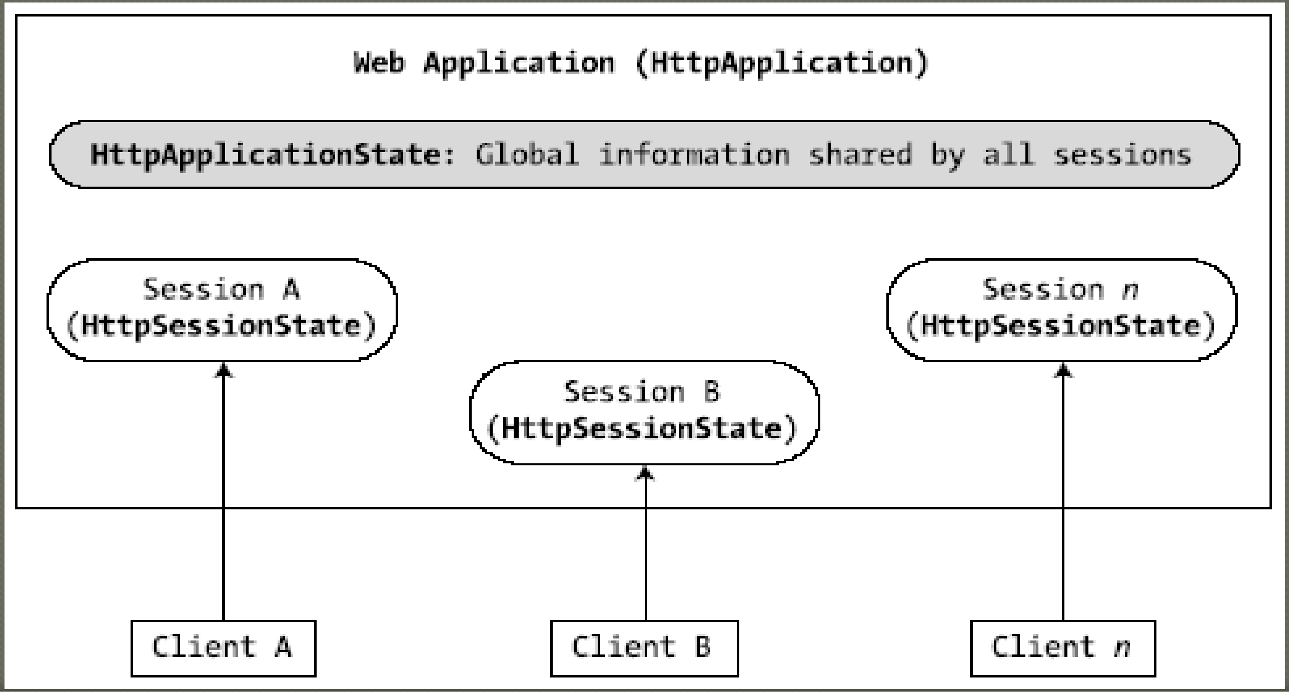
- *Key Members Defined by the* System.Web.HttpApplication *Type*



Property	Meaning in Life
`Application`	This property allows you to interact with application-level variables, using the exposed `HttpApplicationState` type.
`Request`	This property allows you to interact with the incoming HTTP request (via `HttpRequest`).
`Response`	This property allows you to interact with the incoming HTTP response (via `HttpResponse`).
`Server`	This property gets the intrinsic server object for the current request (via `HttpServerUtility`).
`Session`	This property allows you to interact with session-level variables, using the exposed `HttpSessionState` type.

Application/Session Distinction

- Application state is maintained by an instance of the HttpApplicationState type.
  - This class enables to share global information across all users (and all pages) who are logged on to your ASP.NET application.
- Session state is maintained by HttpSessionState class type
  - used to remember information for a specific user
  - When a new user logs on to an ASP.NET web application, the runtime will automatically assign that user a new session ID.

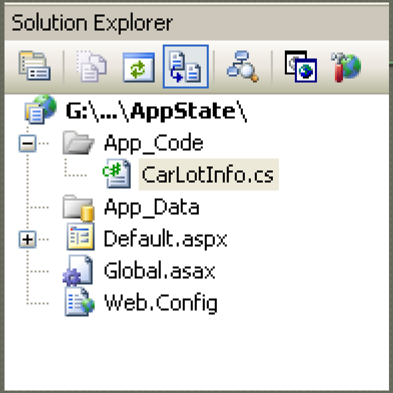

Web Application (HttpApplication)
HttpApplicationState: Global information shared by all sessions
Session A (HttpSessionState)
Session B (HttpSessionState)
Session n (HttpSessionState)
Client A
Client B
Client n

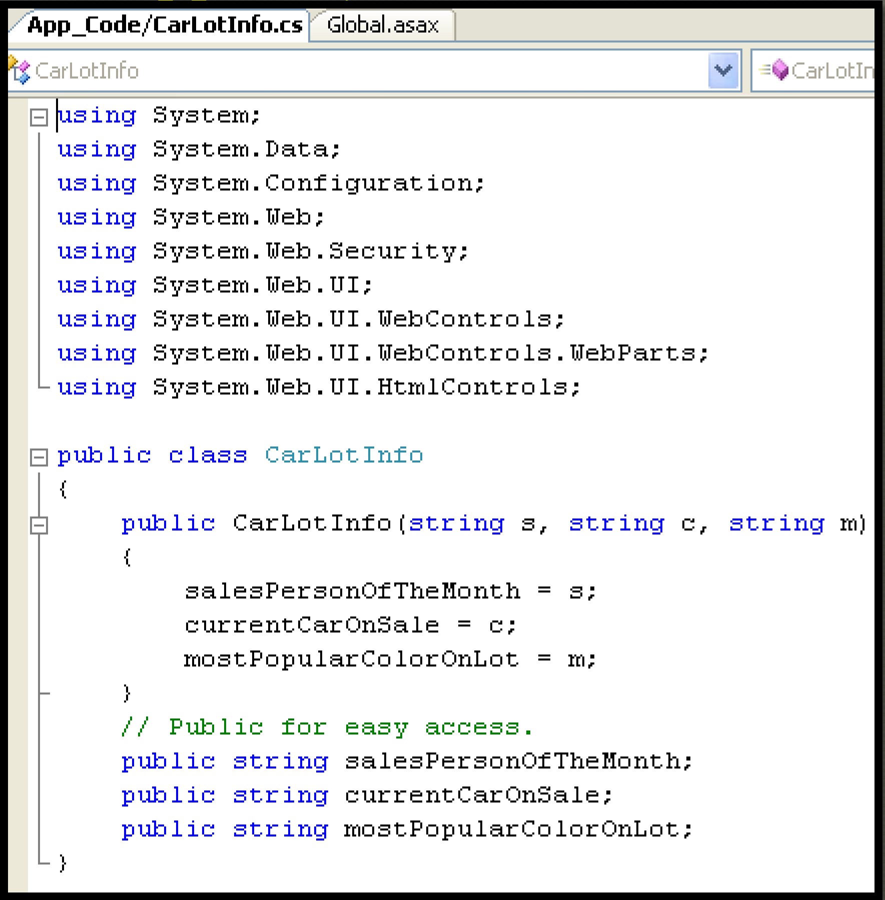
Maintaining Application-Level State Data

- The HttpApplicationState type enables developers to share global information across multiple sessions in an ASP.NET application.

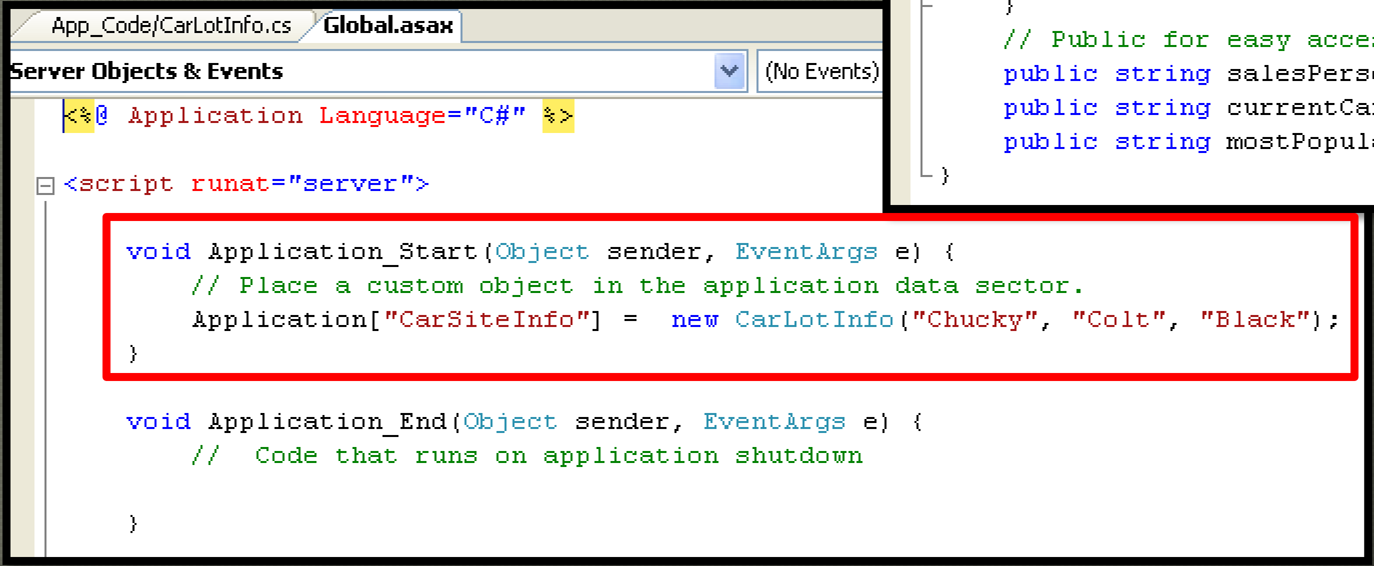
Members	**Description**
AllKeys	This property returns an array of System.String types that represent all the names in the HttpApplicationState type.
Count	gets the number of item objects in the HttpApplicationState type.
Add()	add a new name/value pair into the HttpApplicationState type.
Clear()	deletes all items in the HttpApplicationState type. This is functionally equivalent to the RemoveAll()method.
Lock(), Unlock()	alter a set of application variables in a thread-safe manner.
RemoveAll(), Remove(), RemoveAt()	remove a specific item (by string name) within the HttpApplicationState type. RemoveAt() removes the item via a numerical RemoveAt() indexer.

Application State Demo (Ch_24 code\AppState)



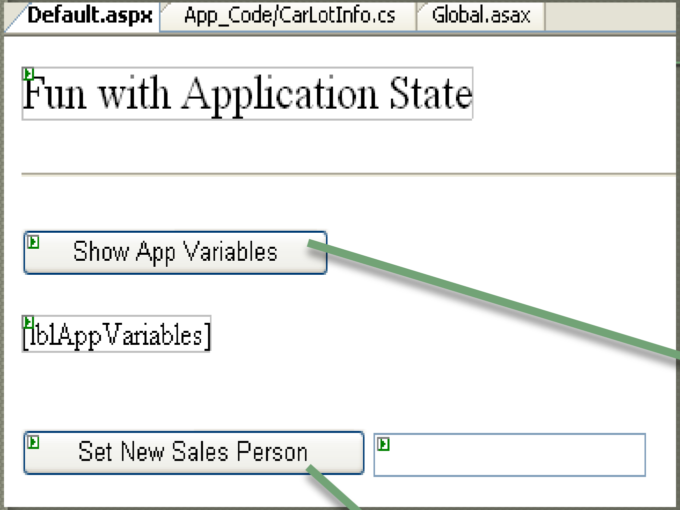


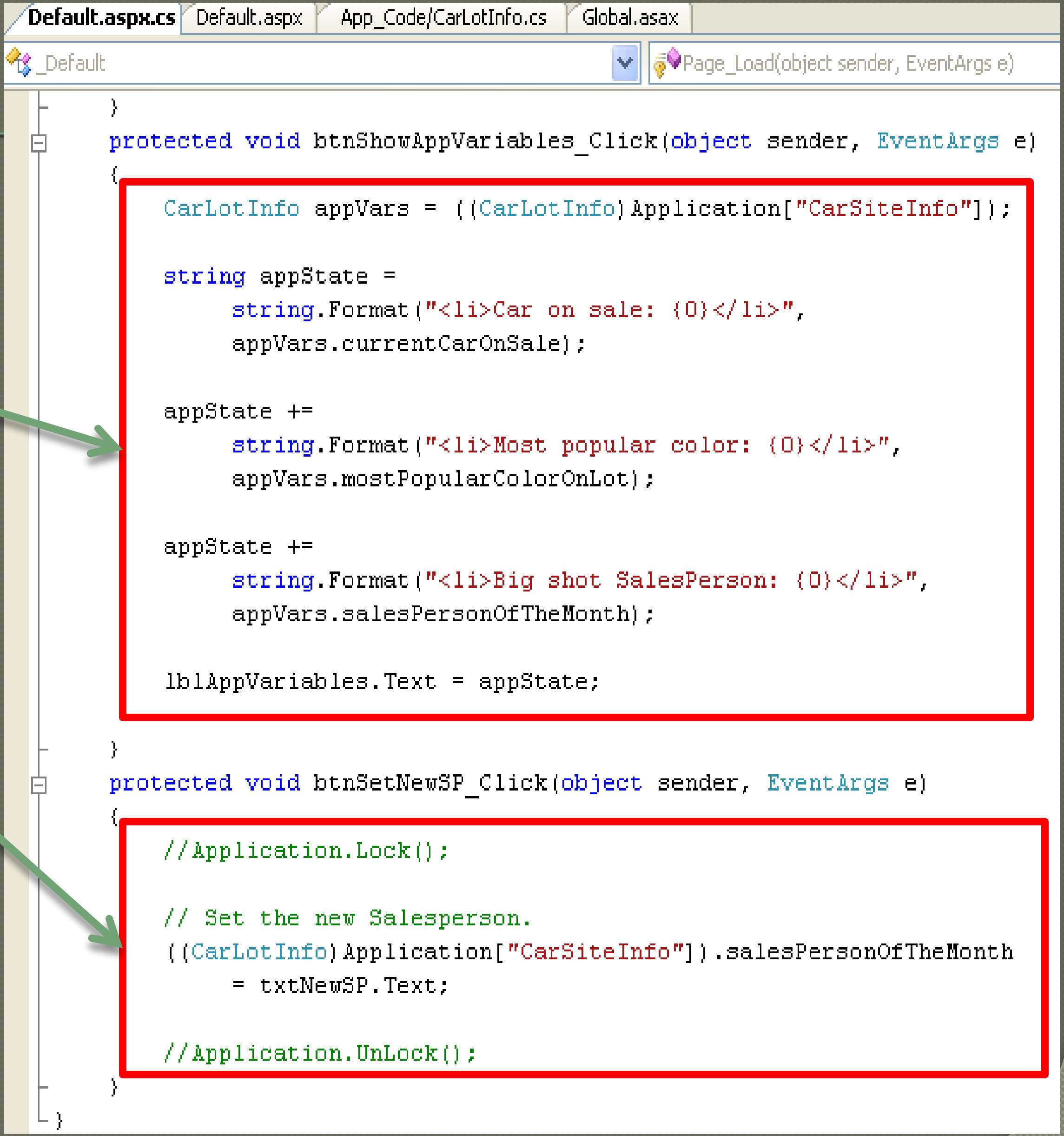
```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

public class CarLotInfo
{
    public CarLotInfo(string s, string c, string m)
    {
        salesPersonOfTheMonth = s;
        currentCarOnSale = c;
        mostPopularColorOnLot = m;
    }
    // Public for easy access.
    public string salesPersonOfTheMonth;
    public string currentCarOnSale;
    public string mostPopularColorOnLot;
}
```



```
<%@ Application Language="C#" %>

<script runat="server">

    void Application_Start(Object sender, EventArgs e) {
        // Place a custom object in the application data sector.
        Application["CarSiteInfo"] =  new CarLotInfo("Chucky", "Colt", "Black");
    }

    void Application_End(Object sender, EventArgs e) {
        //  Code that runs on application shutdown

    }
```

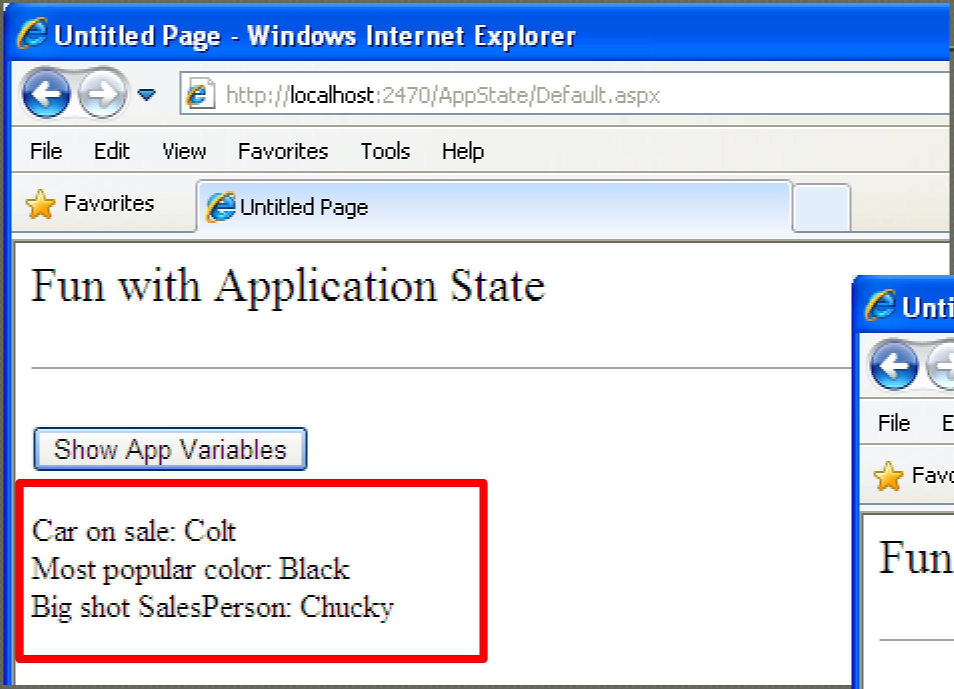
Application State Demo (Ch_24 code\AppState)





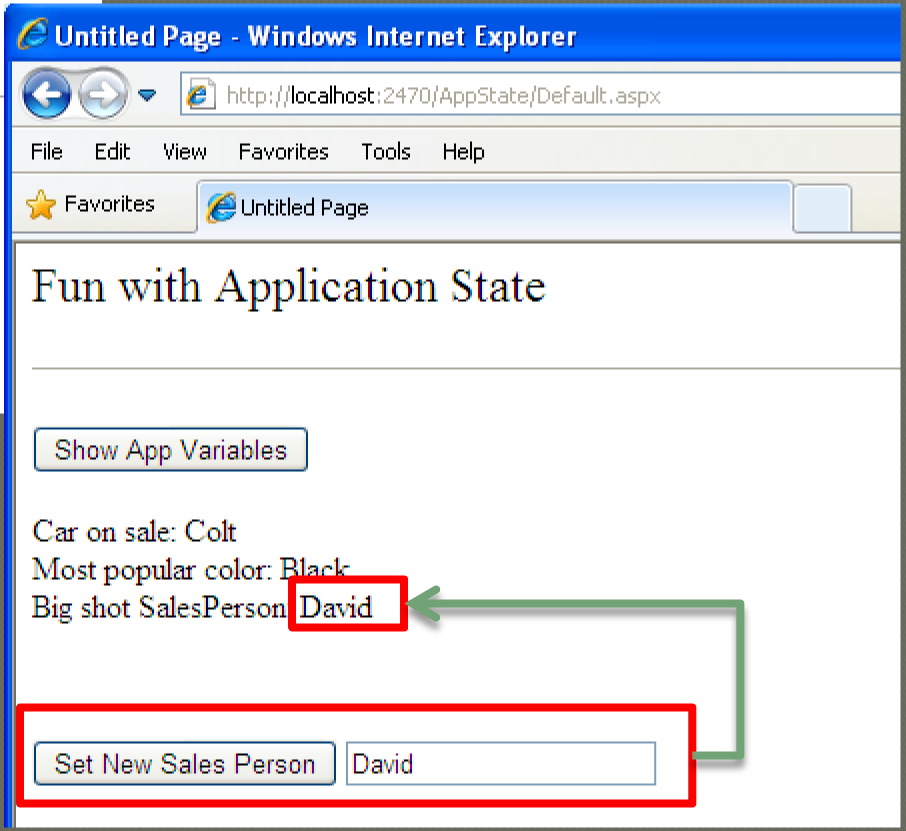
```
    }
    protected void btnShowAppVariables_Click(object sender, EventArgs e)
    {
        CarLotInfo appVars = ((CarLotInfo)Application["CarSiteInfo"]);

        string appState =
             string.Format("<li>Car on sale: {0}</li>",
             appVars.currentCarOnSale);

        appState +=
             string.Format("<li>Most popular color: {0}</li>",
             appVars.mostPopularColorOnLot);

        appState +=
             string.Format("<li>Big shot SalesPerson: {0}</li>",
             appVars.salesPersonOfTheMonth);

        lblAppVariables.Text = appState;

    }
    protected void btnSetNewSP_Click(object sender, EventArgs e)
    {
        //Application.Lock();

        // Set the new Salesperson.
        ((CarLotInfo)Application["CarSiteInfo"]).salesPersonOfTheMonth
             = txtNewSP.Text;

        //Application.UnLock();
    }
}
```

Application State Demo (Ch_24 code\AppState)


Untitled Page - Windows Internet Explorer
http://localhost:2470/AppState/Default.aspx
File Edit View Favorites Tools Help
Favorites
Untitled Page
Show App Variables

Fun with Application State

Car on sale: Colt
Most popular color: Black
Big shot SalesPerson: Chucky


Untitled Page - Windows Internet Explorer
http://localhost:2470/AppState/Default.aspx
File Edit View Favorites Tools Help
Favorites
Untitled Page
Show App Variables
Set New Sales Person
David

Fun with Application State

Car on sale: Colt
Most popular color: Black
Big shot SalesPerson: David

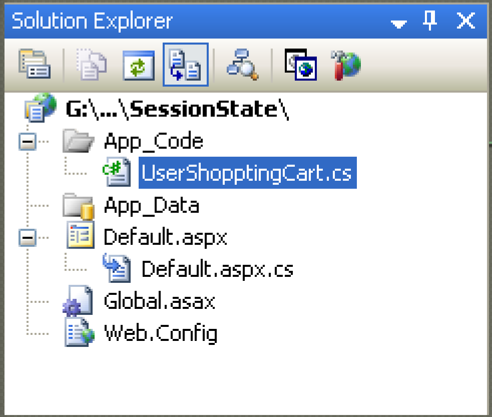
Working with the Application Cache

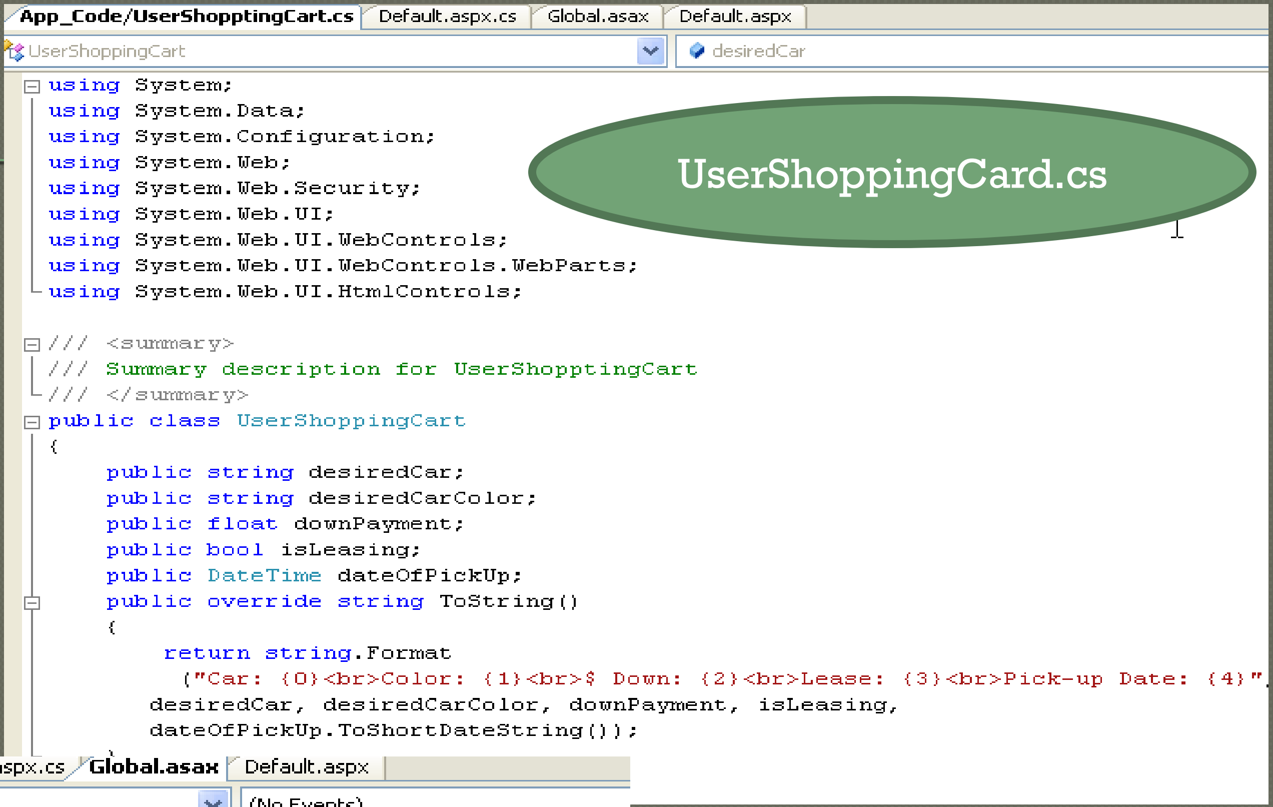
- The ASP.NET System.Web.Caching.Cache object (which is accessible via the Context.Cache property) allows you to define an object that is accessible by all users (from all pages) for a fixed amount of time.
- Fun with Data Caching
- Modifying the *.aspx File
- Example: CacheState

Maintaining Session Data

- When a new user logs on to your web application, the .NET runtime will automatically assign the user a unique session ID
- Each session ID is assigned a custom instance of the HttpSessionState type to hold on to user-specific data.

Session Data Example (Ch 24 code\SessionState)

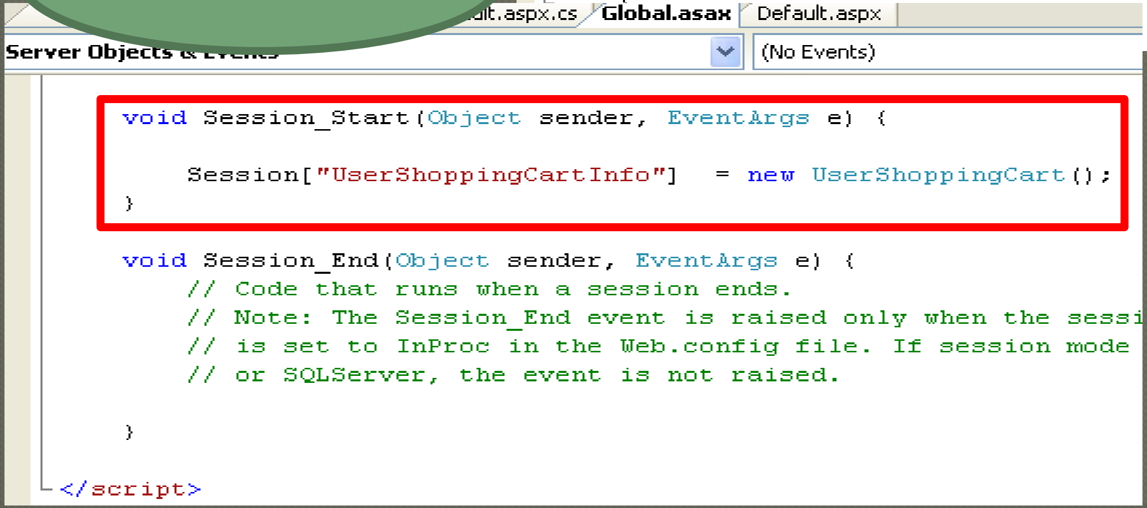




UserShoppingCard.cs

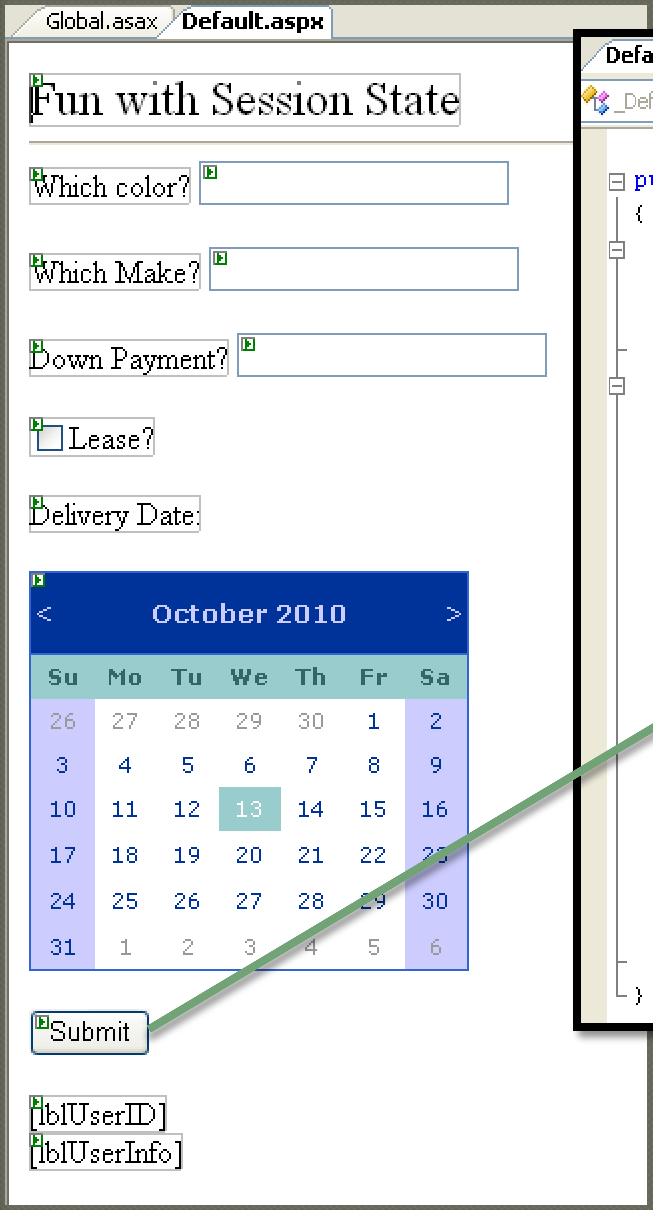
```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;

/// <summary>
/// Summary description for UserShopptingCart
/// </summary>
public class UserShoppingCart
{
    public string desiredCar;
    public string desiredCarColor;
    public float downPayment;
    public bool isLeasing;
    public DateTime dateOfPickUp;
    public override string ToString()
    {
        return string.Format
         ("Car: {0}<br>Color: {1}<br>$ Down: {2}<br>Lease: {3}<br>Pick-up Date: {4}",
        desiredCar, desiredCarColor, downPayment, isLeasing,
        dateOfPickUp.ToShortDateString());
```

Global.asax



```
    void Session_Start(Object sender, EventArgs e) {

        Session["UserShoppingCartInfo"]  = new UserShoppingCart();
    }

    void Session_End(Object sender, EventArgs e) {
        // Code that runs when a session ends.
        // Note: The Session_End event is raised only when the sessi
        // is set to InProc in the Web.config file. If session mode
        // or SQLServer, the event is not raised.

    }

</script>
```

Session Data Example (Ch 24 code\SessionState)



Fun with Session State

Which color?

Which Make?

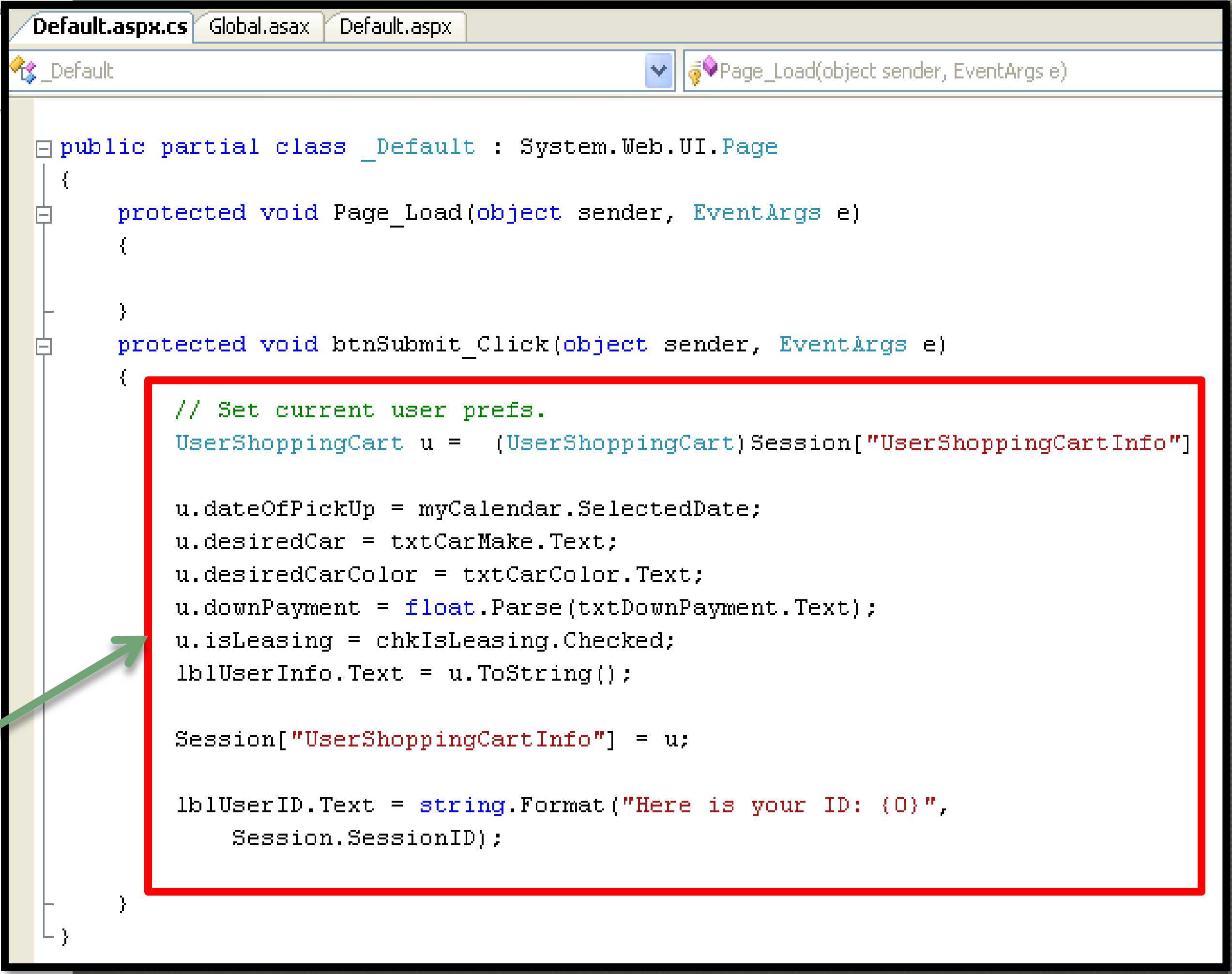
Down Payment?

Lease?

Delivery Date:

<			October 2010			>
Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

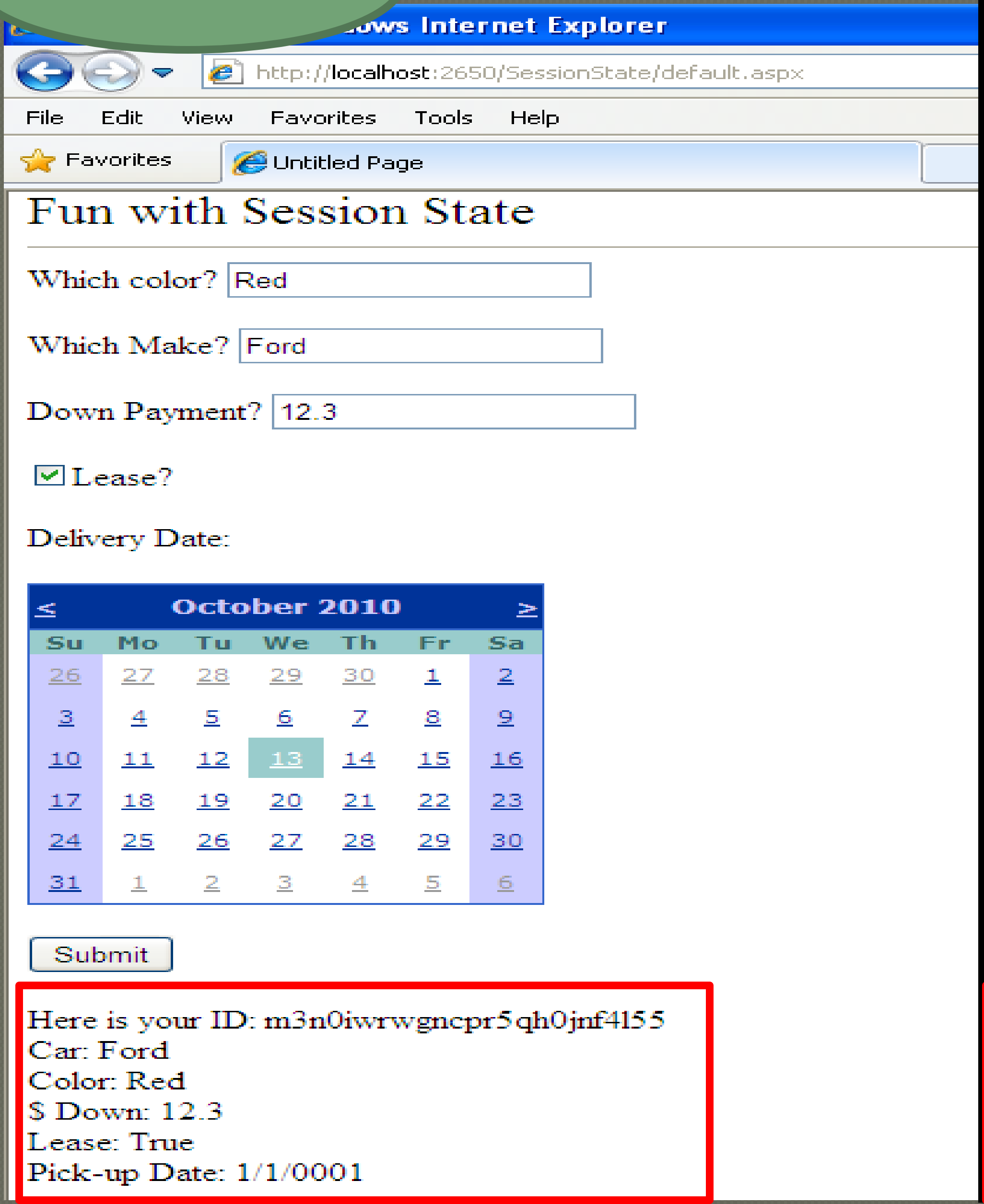
[lblUserID]
[lblUserInfo]



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

    }
    protected void btnSubmit_Click(object sender, EventArgs e)
    {
        // Set current user prefs.
        UserShoppingCart u =  (UserShoppingCart)Session["UserShoppingCartInfo"]

        u.dateOfPickUp = myCalendar.SelectedDate;
        u.desiredCar = txtCarMake.Text;
        u.desiredCarColor = txtCarColor.Text;
        u.downPayment = float.Parse(txtDownPayment.Text);
        u.isLeasing = chkIsLeasing.Checked;
        lblUserInfo.Text = u.ToString();

        Session["UserShoppingCartInfo"] = u;

        lblUserID.Text = string.Format("Here is your ID: {0}",
            Session.SessionID);

    }
}
```

IE browser

Chrome browser



...ows Internet Explorer
http://localhost:2650/SessionState/default.aspx
File Edit View Favorites Tools Help
Favorites
Untitled Page
October 2010
Submit

Fun with Session State

Which color? Red

Which Make? Ford

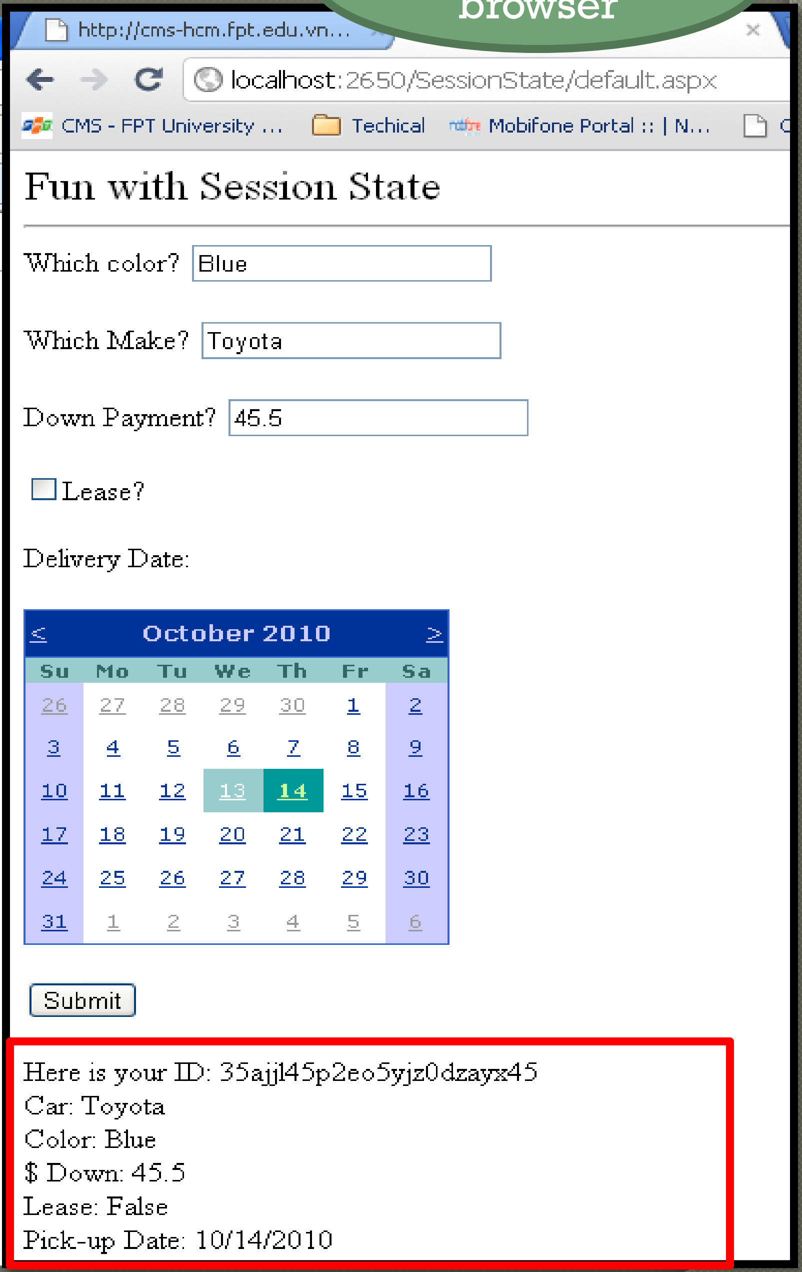
Down Payment? 12.3

✓ Lease?

Delivery Date:

<			October 2010			>
Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Here is your ID: m3n0iwrwgncpr5qh0jnf4155
Car: Ford
Color: Red
$ Down: 12.3
Lease: True
Pick-up Date: 1/1/0001



http://cms-hcm.fpt.edu.vn...
localhost:2650/SessionState/default.aspx
CMS - FPT University ...
Techical
Mobifone Portal :: | N...
Submit

Fun with Session State

Which color? Blue

Which Make? Toyota

Down Payment? 45.5

☐ Lease?

Delivery Date:

<			October 2010			>
Su	Mo	Tu	We	Th	Fr	Sa
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

Here is your ID: 35ajjl45p2eo5yjz0dzayx45
Car: Toyota
Color: Blue
$ Down: 45.5
Lease: False
Pick-up Date: 10/14/2010

Understanding Cookies

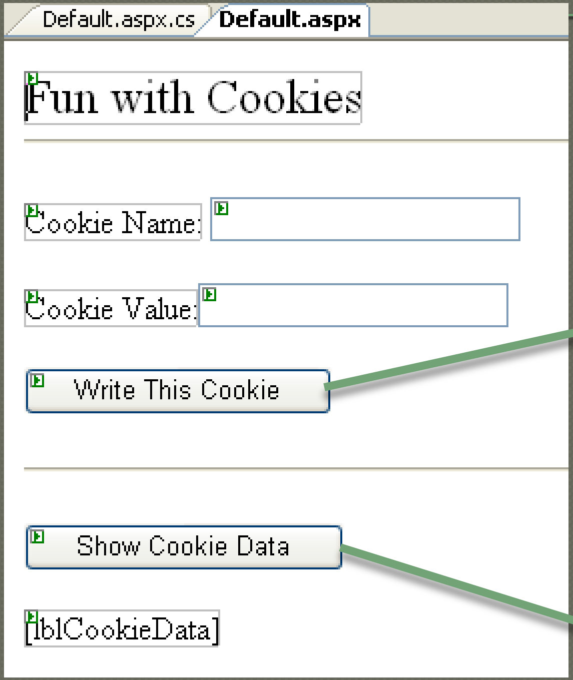
- Cookie is a text file (or set of files) on the user’s machine.
- When a user logs on to a given site, the browser checks to see if the user’s machine has a cookie file for the URL in question and, if so, appends this data to the HTTP request.
- The receiving server-side web page could then read the cookie data to create a GUI that may be based on the current user preferences.
- Cookies are stored by default under *C:\Documents and Settings\<loggedOnUser>\Cookies.*
- Cookies are a horrible choice when you wish to maintain sensitive information about the current user (such as a credit card number, password, or whatnot).

Understanding Cookies

- Creating Cookies
  - ASP.NET cookies can be configured to be either persistent or temporary:
    - A persistent cookie is typically regarded as the classic definition of cookie data, in that the set of name/value pairs is physically saved to the user's hard drive.
    - Temporary cookies (also termed *session cookies*) contain the same data as a persistent cookie, but the name/value pairs are never saved to the user's machine; rather, they exist *only* within the HTTP header. Once the user logs off your site, all data contained within the session cookie is destroyed.

Note: *Most browsers support cookies of up to 4,096 bytes => cookies are best used to store small amounts of data, such as a user ID that can be used to identify the user and pull details from a database.*

Cookie Demo (Ch_24 code\CookieStateApp)



Fun with Cookies

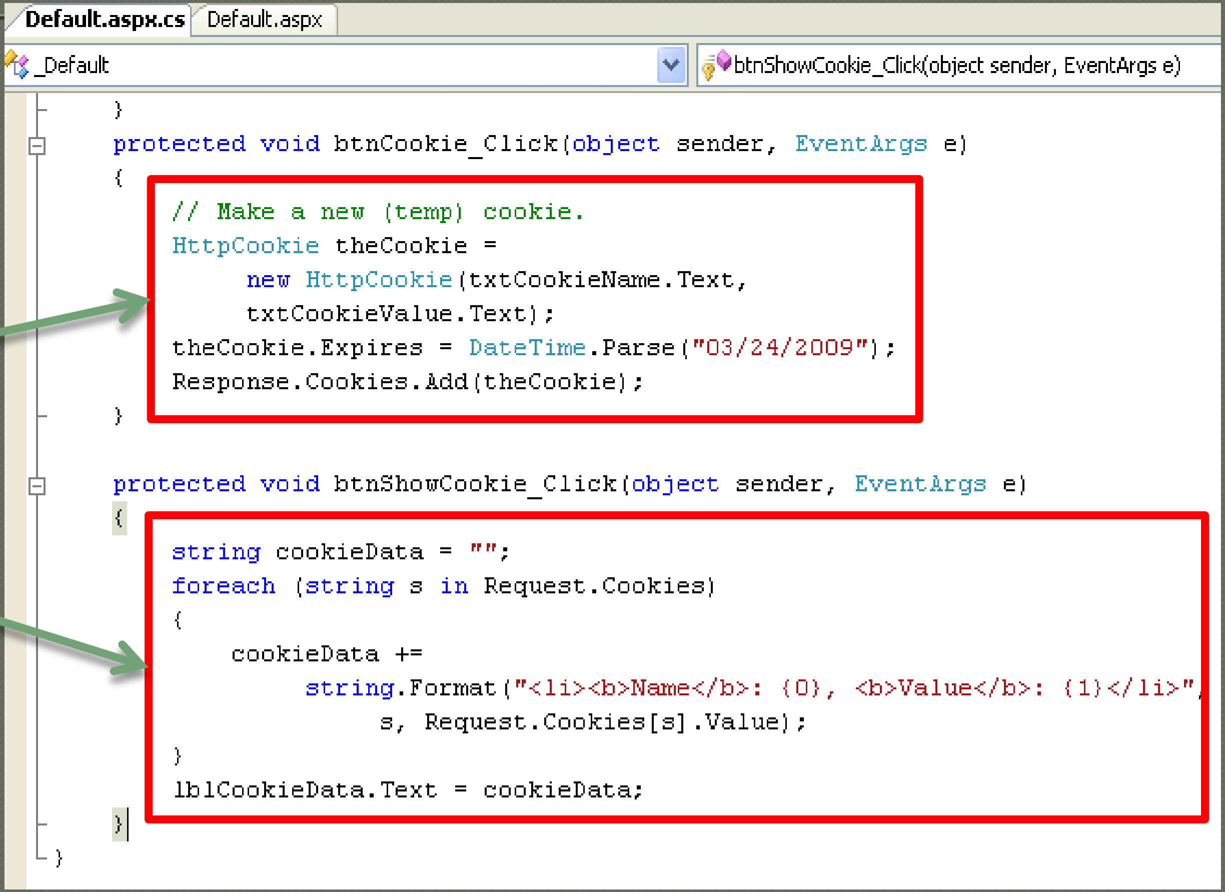
Cookie Name:

Cookie Value:

Write This Cookie

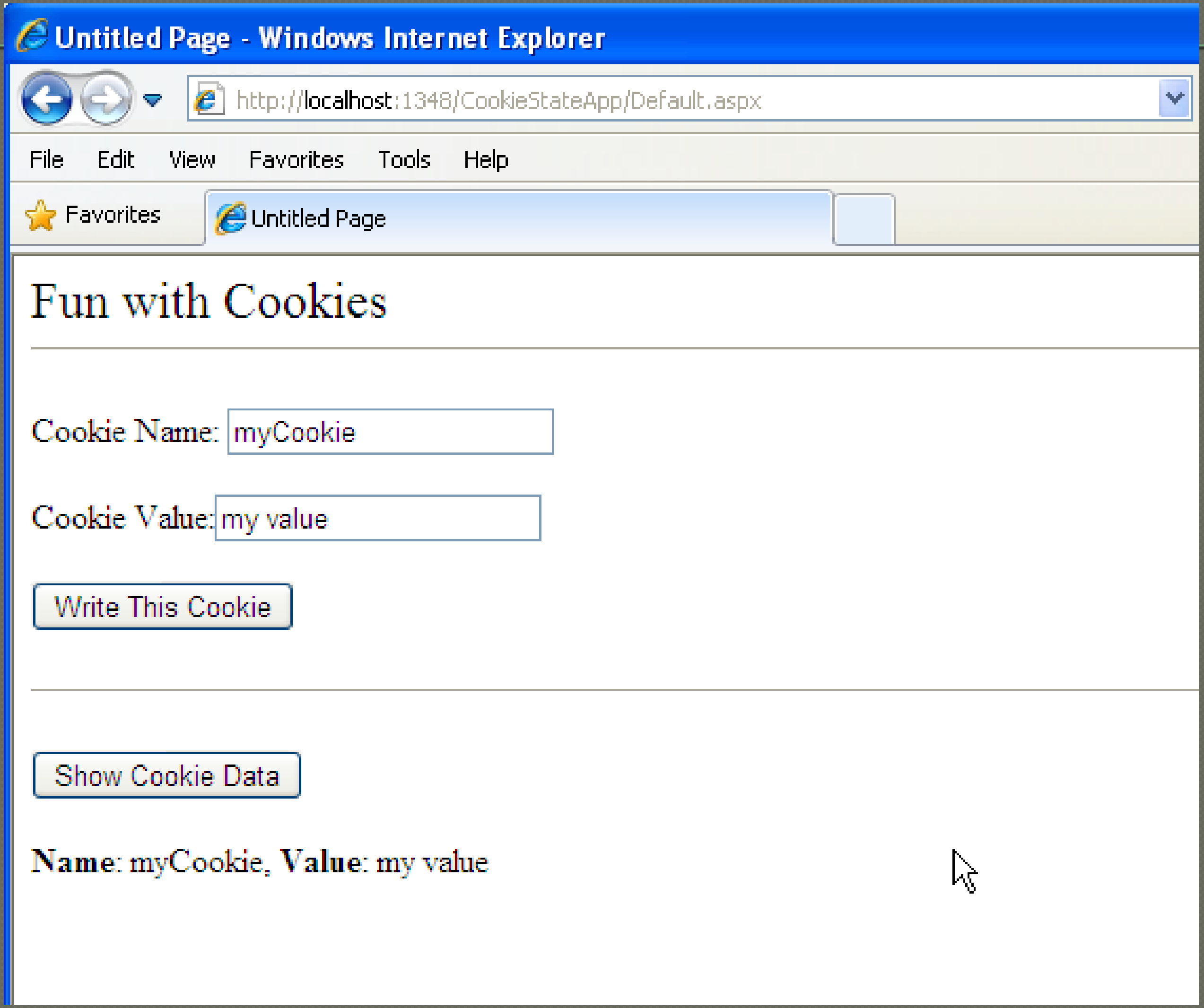
Show Cookie Data

[lblCookieData]



```
    }
    protected void btnCookie_Click(object sender, EventArgs e)
    {
        // Make a new (temp) cookie.
        HttpCookie theCookie =
             new HttpCookie(txtCookieName.Text,
             txtCookieValue.Text);
        theCookie.Expires = DateTime.Parse("03/24/2009");
        Response.Cookies.Add(theCookie);
    }

    protected void btnShowCookie_Click(object sender, EventArgs e)
    {
        string cookieData = "";
        foreach (string s in Request.Cookies)
        {
            cookieData +=
                 string.Format("<li><b>Name</b>: {0}, <b>Value</b>: {1}</li>",
                      s, Request.Cookies[s].Value);
        }
        lblCookieData.Text = cookieData;
    }
}
```

Cookie Demo (Ch_24 code\CookieStateApp)


Untitled Page - Windows Internet Explorer
http://localhost:1348/CookieStateApp/Default.aspx
File Edit View Favorites Tools Help
Favorites
Untitled Page

Fun with Cookies

Cookie Name: myCookie

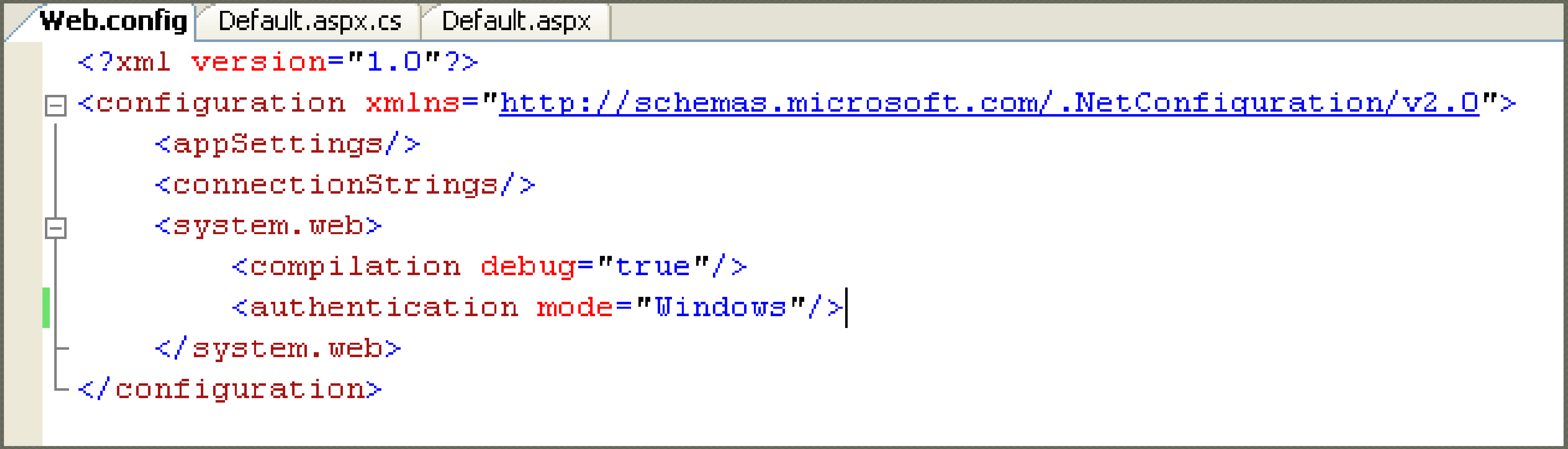
Cookie Value: my value

Write This Cookie
Show Cookie Data

Name: myCookie, **Value**: my value

Configuring ASP.NET Web Application Using Web.config

- In ASP .NET, the web-centric configuration files are always named Web.config, with default structure:



```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
    <appSettings/>
    <connectionStrings/>
    <system.web>
        <compilation debug="true"/>
        <authentication mode="Windows"/>
    </system.web>
</configuration>
```

Configuring ASP.NET Web Application

Element	Description
<appSettings>	establish custom name/value pairs that can be programmatically read in memory for use by your pages
<authentication>	used to define the authentication mode for this web application.
<authorization>	used to define which users can access which resources on the web server.
<compilation>	used to enable (or disable) debugging
<connectionStrings>	used to hold external connection strings
<customErrors>	used to tell the runtime exactly how to display errors
<globalization>	used to configure the globalization settings
<sessionState>	used to control how and where session state data will be stored by the .NET runtime.
<trace>	This element is used to enable (or disable) tracing support for this web application.

Enabling Tracing via <trace>

```
<trace    enabled = "true|false"
          localOnly = "true|false"
          pageOutput = "true|false"
          requestLimit = "integer"
          traceMode = "SortByTime|SortByCategory"/>
```

- localOnly: Indicates that the trace information is viewable only on the host web server and not by remote clients (the default is true).
- requestLimit: Specifies the number of trace requests to store on the server. The default is 10

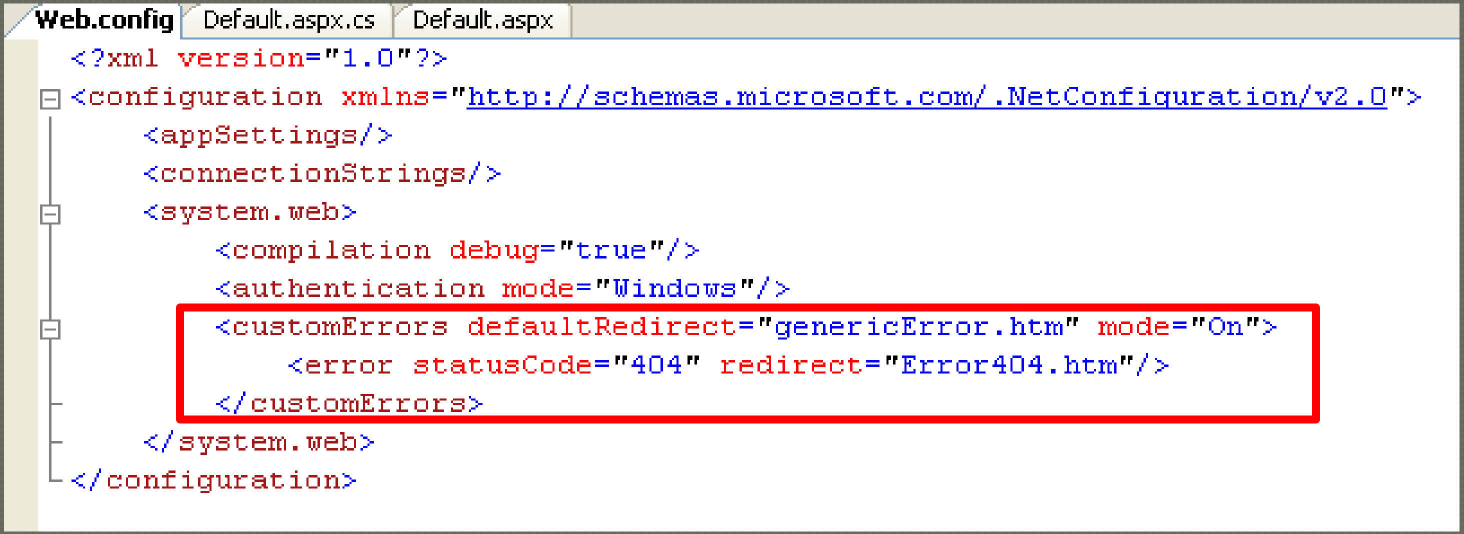
To view trace information, using:

```
http://localhost/MyWebApplication/trace.axd
```

Customizing Error Output via <customErrors>

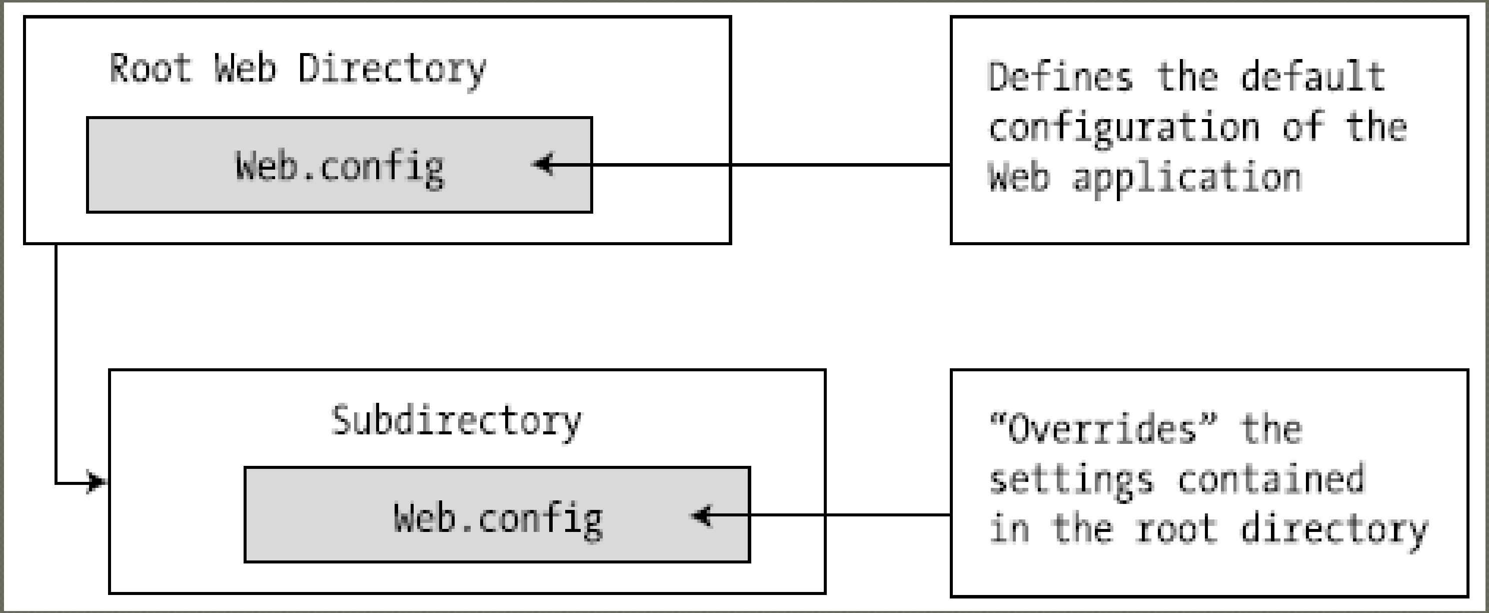
<customErrors defaultRedirect = "url" mode="On|Off|RemoteOnly">
    <error statusCode="statuscode" redirect="url"/>
</customErrors>

Example


Web.config | Default.aspx.cs | Default.aspx

```
<?xml version="1.0"?>
<configuration xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
    <appSettings/>
    <connectionStrings/>
    <system.web>
        <compilation debug="true"/>
        <authentication mode="Windows"/>
        <customErrors defaultRedirect="genericError.htm" mode="On">
            <error statusCode="404" redirect="Error404.htm"/>
        </customErrors>
    </system.web>
</configuration>
```

Configuration Inheritance


Root Web Directory
Web.config
Defines the default configuration of the Web application
Subdirectory
Web.config
"Overrides" the settings contained in the root directory

The machine.config file is the ultimate parent in the configuration inheritance hierarchy.

Summary

- HTTP issue: Stateless
- ASP.NET provides several mechanisms to maintain stateful information in your web applications:
  - Make use of ASP.NET view state.
  - Make use of ASP.NET control state.
  - Define application-level variables.
  - Make use of the cache object.
  - Define session-level variables.
  - Interact with cookie data.
- Configuring web site through web.config

Chapter 24: Q & A