State Management

Agenda

- **Types of state available in ASP.NET and where to use them**
  - Application state
  - Session state
  - Cookie state
  - Query strings
  - Items collection
  - View state
  - Profile
  - Cross-page posting
  - MultiView/View controls

Application State

- **Application state available globally in an application**
- **Application State accessed through HttpApplication object's Application property**
  - Application_Start event available for initialization
  - Must take care to call Application.Lock/Unlock when modifying shared state (try to avoid this)
  - Not shared across web-farms/gardens
  - Typically want to use data cache instead today

Application State Usage

```
protected void Application_Start(Object sender, EventArgs e)
{
  DataSet ds = new DataSet();
  // population of DataSet from ADO.NET query not shown

  // Cache DataSet reference
  Application["FooDataSet"] = ds;
}

private void Page_Load(object sender, System.EventArgs e)
{
  DataSet ds = (DataSet)Application["FooDataSet"];
  //...
  myDataGrid.DataSource = ds;
  //...
}
```

Session State

- **Stores state on behalf of individual clients**
  - Scoped by a single client session
  - Tagged with a unique (hard to guess) id
  - Session ID transmitted via cookie (by default)
  - Accessed through Session property of Page
  - Available through HttpContext.Session
  - Session_Start event available for initialization

Sample Use of Session State

```
protected void Session_Start(Object sender, EventArgs e)
{
  // Fires when the Session is started
  Session["Age"] = -1;
}

 // In some page of the application
private void enterButton_Click(object sender, EventArgs e)
{
  Session["Age"] = int.Parse(ageTextBox.Text);
}

// Inside another page of the application
// only let user vote if he/she is over 18
private void Page_Load(object sender, EventArgs e)
{
  voteButton.Enabled = ((int)Session["Age"]) > 18;
}
```

Session State in ASP.NET

- **ASP.NET brings several improvements to session state**
  - Can switch to 'cookieless' id management
  - Auto-detect cookieless mode (2.0)
  - May not serialize all requests from a given client
  - Can configure to survive process shut down
  - Can configure to work across machines in a Web farm
  - Pluggable implementation (2.0)

Configuring Session State

- **You configure session through web.config**
  - The <sessionState> element controls config
  - Features include cookieless ID management, session time out, storage location, ...

web.config

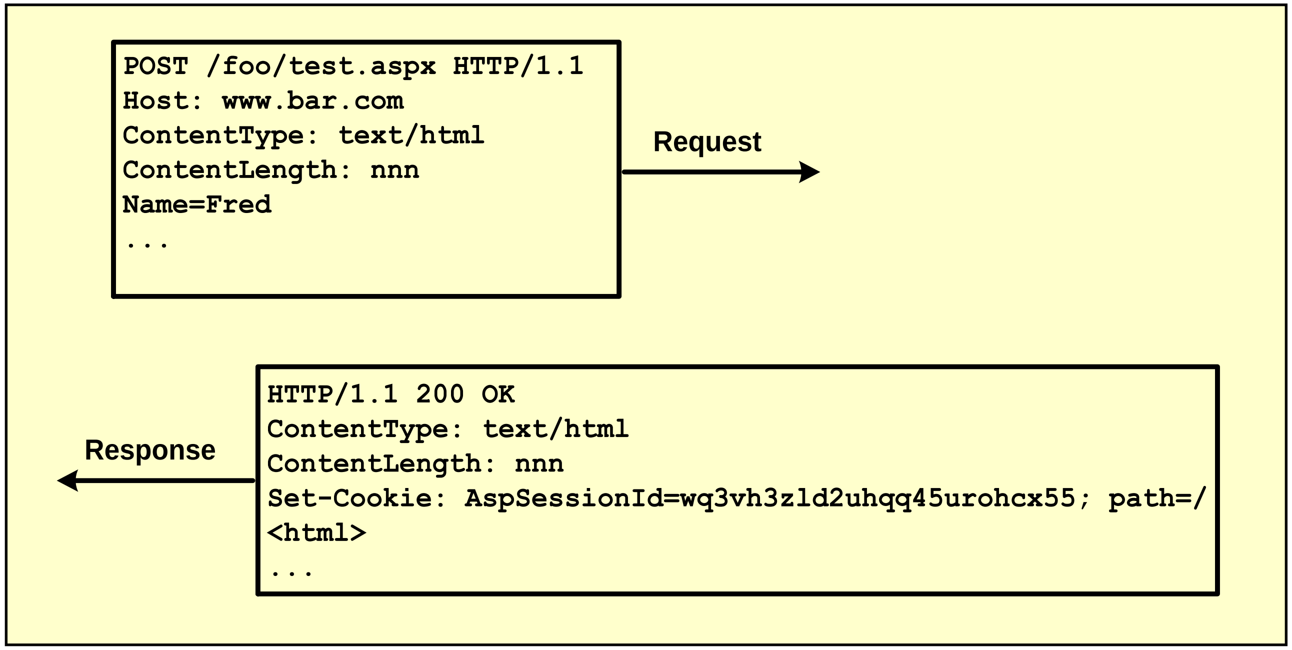
```
<configuration>
  <system.web>
    <sessionState cookieless="auto" />
  </system.web>
</configuration>
```

Session State Options

Attribute	Possible Values	Meaning
`cookieless`	`True, False, AutoDetect, UseDeviceProfile`	**Pass `SessionID` via cookies, URL mangling, or auto detect**
`mode`	`Off, InProc, SQLServer, StateServer`	**Where to store session state (or whether it is disabled)**
`stateConnectionString`	**Example:** `'192.168.1.100:42424'`	**Server name and port for** `StateServer`
`sqlConnectionString`	**Example:** `'server=192.168.1.100;uid=xx;pwd=yy'`	`SQLServer` **connection string excluding database (tempdb is implied)**
`timeout`	**Example:** `40`	**Session state timeout value (in minutes)**

Session Key Management

- By default, clients are tracked with a unique session key stored in a cookie



```
POST /foo/test.aspx HTTP/1.1
Host: www.bar.com
ContentType: text/html
ContentLength: nnn
Name=Fred
...
```

```
HTTP/1.1 200 OK
ContentType: text/html
ContentLength: nnn
Set-Cookie: AspSessionId=wq3vh3zld2uhqq45urohcx55; path=/
<html>
...
```

Auto-detect cookieless mode

web.config

```
<sessionState
cookieless="AutoDetect" />
```

initial request

```
GET /sessiontest/Default.aspx HTTP/1.1
```

```
HTTP/1.1 302 Found
Location: /sessiontest/Default.aspx?AspxAutoDetectCookieSupport=1
Set-Cookie: AspxAutoDetectCookieSupport=1; path=/ ...
```

client with cookies

```
GET /sessiontest/Default.aspx?AspxAutoDetectCookieSupport=1 HTTP/1.1
Cookie: AspxAutoDetectCookieSupport=1      ...
```

```
HTTP/1.1 200 OK
Set-Cookie: ASP.NET_SessionId=su4gsqiawe0el3zc3ktytc55;  ...
```

client without cookies

```
GET /sessiontest/Default.aspx?AspxAutoDetectCookieSupport=1 HTTP/1.1  ...
```

```
HTTP/1.1 302 Found
Location: /sessiontest/(X(1)S(3p2sjxvd05k5khn4lou24j45))/Default.aspx?AspxAutoDetectCookieSupport=1
```

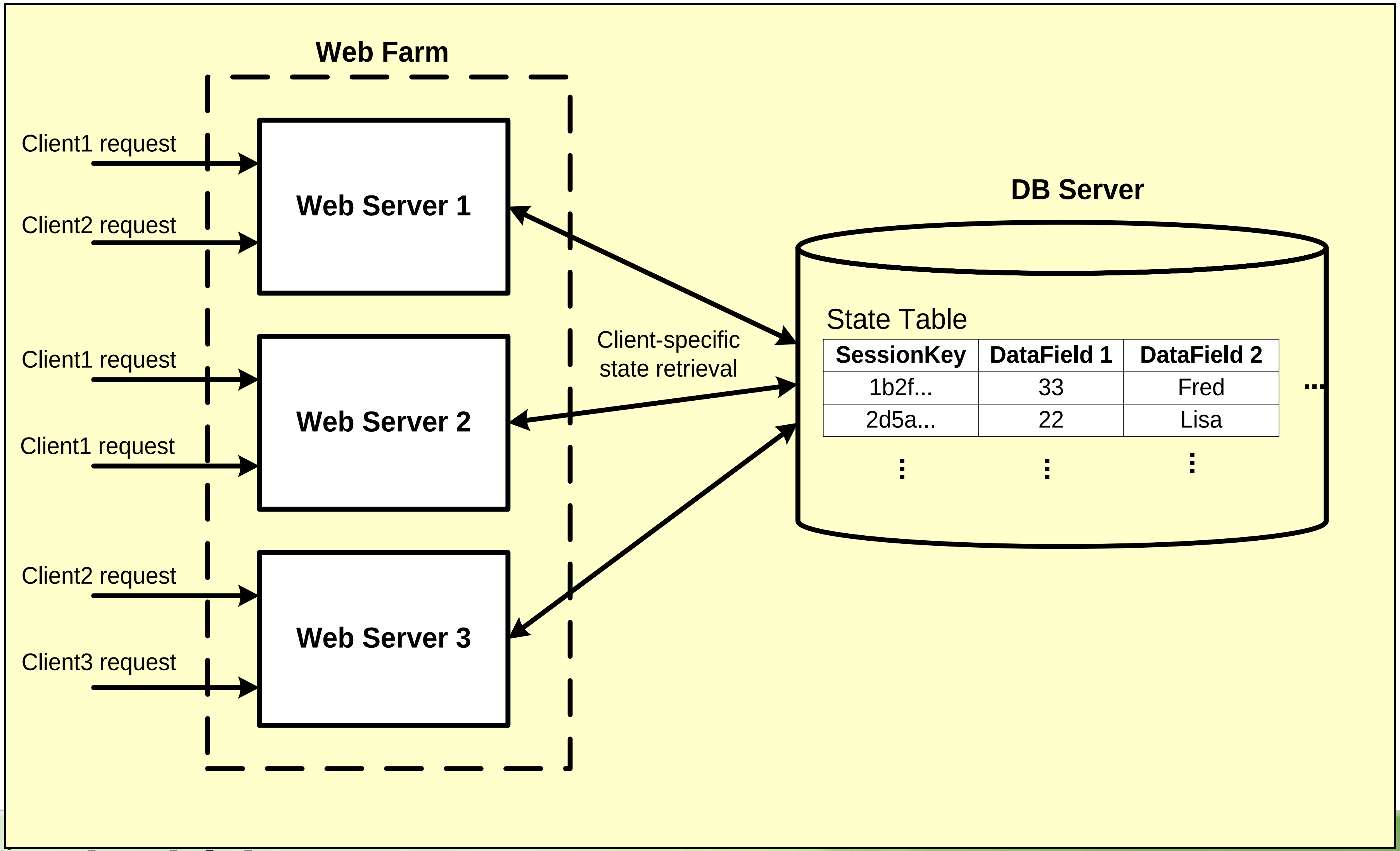
```
GET /sessiontest/(X(1)S(3p2sjxvd05k5khn4lou24j45))/Default.aspx?AspxAutoDetectCookieSupport=1 HTTP/1.1
```

```
HTTP/1.1 200 OK
```

Out of Process Session State

- **With ASP.NET it is possible to store session state out of process**
  - In local or remote NT service
  - In local or remote SQL server
  - Stored as opaque byte stream
  - Incurs round trips to retrieve / flush state

Out of Process Session



SessionKey	DataField 1	DataField 2
1b2f...	33	Fred
2d5a...	22	Lisa
⋮	⋮	⋮

Minimizing round trips

- **By default, when session is stored out of process 2-round trips per request incurred**
  - One to read, one to write
- **Can tell ASP.NET how many trips are necessary**

```
<%@ Page Language="C#" EnableSessionState="True" %>

<%@ Page Langauge="C#" EnableSessionState="False" %>

<%@ Page Language="C#" EnableSessionState="ReadOnly" %>
```

Cookies

- **Client-side cookies can be used to store user preferences / information**
  - Server requests client to set cookie in response
  - Client sends cookie values in subsequent requests
  - Cookies may be persisted if the Expires property is set
  - Browsers limit cookie data -- only 4096 bytes guaranteed
  - Clients may disable cookies

Using Cookies in ASP.NET

```
int age = 0;
if (Request.Cookies["Age"] == null)
{
  // "Age" Cookie not set, set with this response
  HttpCookie ac = new HttpCookie("Age");
  ac.Value = ageTextBox.Text;
  Response.Cookies.Add(ac);
  age = int.Parse(ageTextBox.Text);
}
else
{
  // use existing cookie value
  age = int.Parse(Request.Cookies["Age"].Value);
}
// use age value...
```

QueryString State

- **State can be passed between pages by appending a query string to the URL**
  - Must be passed as name/value pairs
  - Restricted to URL compatible strings
    - Must use % to represent restricted characters as encoded (including /.#?;:$,+@&={}|\^[]')
  - Indicate query string with '?' character
  - Delimit name/value pairs with '&' character
  - Access query string values through the indexer in HttpRequest

```
http://www.pluralsight.com/test.aspx?name=joe&age=21
```

QueryString Example

Signup.aspx.cs

```
private void _signupButton_Click(object sender, System.EventArgs e)
{
  StringBuilder url = new StringBuilder(); // prepare query string
  url.Append("ThankYou.aspx?firstname=");
  url.Append(firstnameTextBox.Text);
  url.Append("&lastname=");
  url.Append(lastnameTextBox.Text);
  url.Append("&zipcode=");
  url.Append(zipcodeTextBox.Text);
  Response.Redirect(url.ToString());
}
```

ThankYou.aspx.cs

```
private void Page_Load(object sender, System.EventArgs e)
{
  StringBuilder msg = new StringBuilder();
  msg.Append("<b>Registered user:</b> ");
  msg.Append(Request["firstname"]);
  msg.Append(" ");
  msg.Append(Request["lastname"]);
  msg.Append("<br/> <b>location=</b>");
  msg.Append(Request["zipcode"]);
  summaryParagraph.InnerHtml = msg.ToString();
}
```

Items State

- **The Items collection of HttpContext can be used to store per-request state**
  - Useful when passing data between elements in the pipeline (like modules)
  - Can be used to pass data between pages when using Server.Transfer

Items State Example

Signup.aspx.cs

```
private void signupButton_Click(object sender, System.EventArgs e)
{
   Context.Items["firstname"] = firstnameTextBox.Text;
   Context.Items["lastname"]  = lastnameTextBox.Text;
   Context.Items["zipcode"]   = zipcodeTextBox.Text;
   Server.Transfer("ThankYou.aspx");
}
```

ThankYou.aspx.cs

```
private void Page_Load(object sender, System.EventArgs e)
{
   StringBuilder msg = new StringBuilder();
   msg.Append("<b>Registered user:</b> ");
   msg.Append(Context.Items["firstname"]);
   msg.Append(" ");
   msg.Append(Context.Items["lastname"]);
   msg.Append("<br/> <b>location=</b>");
   msg.Append(Context.Items["zipcode"]);
   summaryParagraph.InnerHtml = msg.ToString();
}
```

ViewState

- **ViewState can be used as a means to store client-specific state**
  - Only retained across POST requests to the same page
  - Types must be serializable

```
private void Page_Load(object sender, EventArgs e)
{
  if (IsPostBack)
  {
    ArrayList cart = (ArrayList)ViewState["Cart"];
    if (cart == null)
      cart = new ArrayList();
    // use items stored in cart
  }
}
```

Cross-page posting support

- **ASP.NET 2.0 re-introduced support for cross page POSTing**

```
<%@ page language="C#" %>
<html>
<body>
    <form runat="server">
      Enter name: <asp:textbox id="_name" runat="server" />
    <asp:button runat="server" text="Page2"
                postbackurl="Page2.aspx" />
 </form>
</body>
</html>
```

Target page

- **Use PreviousPage property to retrieve page from which POST was made**

```
<!-- Page2.aspx -->
<%@ page language="C#" %>
<script runat="server">
  protected override void OnLoad(EventArgs e)
  {
    if (PreviousPage != null)
    {
      TextBox name = (TextBox)PreviousPage.FindControl("_name");
      Response.Write("Hi " + name.Text);
    }
    base.OnLoad(e);
  }
</script>
```

Strongly typed target page

```
<%@ Page Language="C#" %>
<%@ PreviousPageType VirtualPath="~/Page1.aspx" %>

<script runat="server">
  void Page_Load(object sender, EventArgs e) {
    if (PreviousPage != null)
      _nameLabel.Text = PreviousPage.Name;
  }
</script>

<html xmlns="http://www.w3.org/1999/xhtml" >
<body>
    <form id="form1" runat="server">
    <div>
      Thanks for providing us with your personal information!<br />
      <asp:Label ID="_nameLabel" Runat="server"></asp:Label>
    </div>
    </form>
</body>
</html>
```

Assumes Page1.aspx has a public property called Name that exposes the value of the TextBox

Notes on cross-page posting

- **Previous page is 'executed' as if it were posted to, up until LoadComplete**
  - Use `Page.IsCrossPagePostBack` to check whether this is a real post back or a cross page post back on initial page
  - Keep in mind that all page logic in previous page is executed
  - Server-side validation happens after post back on target page
- **Can be used as alternative to other cross-page state propagation techniques**
  - base64-encoded and less transparent than query string or plain POST data
  - ViewState state-bag can be a useful generic repository for cross-page state

Profile

- **Profile provides a persistent, per-client data store**
  - Backed by a provider that maps into a specific database (SqlServer provider available by default)
- **Anonymous information storage possible with migration to known client**
  - GUID used to identify unknown clients
  - Identified with cookie (.ASPXANONYMOUS)
  - Can set to be cookieless (url mangling) or auto (choose dynamically)
- **Strongly typed access to personalization information through configuration file initialization**
  - Usage is even easier than Session state

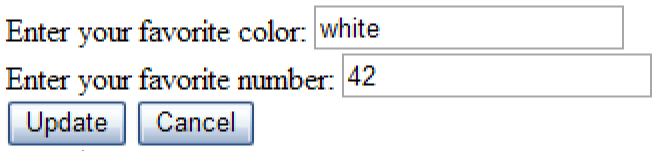
Profile example

- **Enabling profile**

```
<configuration>

  <system.web>
      <profile >
          <properties>
              <add name="FavoriteColor"
                   defaultValue="white" type="System.String"
                   allowAnonymous="true" />
             <add name="FavoriteNumber"
                  defaultValue="42" type="System.Int32"
                  allowAnonymous="true" />
          </properties>
       </profile>
     <anonymousIdentification enabled="true" />
  </system.web>
</configuration>
```

Accessing profile information


Enter your favorite color: white
Enter your favorite number: 42
Update
Cancel

```
void updateButton_Click(object sender, EventArgs e)
{
  Profile.FavoriteColor = favoriteColorTextBox.Text;
  Profile.FavoriteNumber = int.Parse(favoriteNumberTextBox.Text);
  Response.Redirect("~/default.aspx");
}
```

Strongly typed Profile object

```
void Page_Load(object sender, EventArgs e)
{
    theBody.Attributes["bgcolor"] = Profile.FavoriteColor;
    numLabel.Text = Profile.FavoriteNumber.ToString();
}
```

Where is it stored?

- **By default the SQL provider generates a local SQL Server Express database file (under App_Data) with ASP.NET tables to store profile and membership information**

<u>aspnet_Applications</u>

```
ApplicationName
ApplicationId
Description
```

<u>aspnet_Membership</u>

```
ApplicationId
UserId
email
Password
...
```

<u>aspnet_Users</u>

```
ApplicationId
UserName
UserId
IsAnonymous
```

<u>aspnet_Profile</u>

```
UserId
PropertyNames
PropertyValuesString
PropertyValuesBinary
LastUpdatedDate
```

How are clients identified?

- **HttpRequest class now provides an anonymous ID**
  - GUID generated uniquely for clients stored in .ASPXAUTH cookie
  - Must be enabled in web.config
  - Blank if authenticated

```
<configuration
  xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">

  <system.web>
    <anonymousIdentification enabled="true" />
  </system.web>
</configuration>
```

```
Request.AnonymousId
```

78b1d3f6-dfaf-4ffd-8466-f4ce632abced

Managing profile data

- **The static ProfileManager class can be used to manage profile**
  - Accessible outside of a Web application (service, console, ...)
  - Common use: clean up unused anonymous profiles

```
public static class ProfileManager
{
  public static int DeleteInactiveProfiles(
         ProfileAuthenticationOption authenticationOption,
         DateTime userInactiveSinceDate);
  public static bool DeleteProfile(string username);

  public static ProfileInfoCollection FindProfilesByUserName(...);
  public static ProfileInfoCollection GetAllProfiles(...);
  public static int GetNumberOfInactiveProfiles(...);
  public static int GetNumberOfProfiles(...);

    //...
}
```

Where else can profile data be stored?

- **You can change the data source for profile information by changing the associated provider**
  - AspNetSqlProvider available for SqlServer (default provider)
    - Defaults to file-based SQLExpress storage in App_Data
    - Can store in alternate database by changing the LocalSQLServer connection string and running aspnet_regsql.exe in the target database
  - Build your own provider to map onto whatever data store you like

```
<connectionStrings>
  <remove name="LocalSqlServer" />
  <add name="LocalSqlServer"
       connectionString="Server=.;Database=aspnetdb;trusted_connection=yes"/>
</connectionStrings>
```

Per-page state controls

- **MultiView / View**
  - Create multiple *views* and select the active view to display
  - Commonly done in 1.1 with Panel controls by hand
- **Wizard**
  - Similar to MultiView / View but with navigation
  - Next / Prev / Finish, Linked indices

MultiView / View Controls

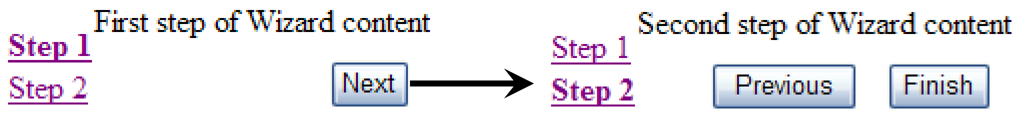
- **MultiView coordinates several child views**
  - One view displayed at a time based on ActiveIndex
  - Remaining views not sent to client
    - However their ViewState remains intact

```
<asp:DropDownList runat="server" ID="selectViewDropDownList"
                  OnSelectedIndexChanged="OnChangeView">
    <asp:ListItem>View 1</asp:ListItem>
    <asp:ListItem>View 2</asp:ListItem>
    <asp:ListItem>View 3</asp:ListItem>
</asp:DropDownList>
<asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="0">
    <asp:View ID="View1" runat="server">Content in view1</asp:View>
    <asp:View ID="View2" runat="server">Content in view2</asp:View>
    <asp:View ID="View3" runat="server">Content in view3</asp:View>
</asp:MultiView>
```

```
protected void OnChangeView(object sender, EventArgs e)
{
  MultiView1.ActiveViewIndex = selectViewDropDownList.SelectedIndex;
}
```

Wizard Control

- **Standard implementation of 'Wizard'**
  - Each 'step' displayed / hidden based on navigation



```
<asp:Wizard ID="Wizard1" runat="server" ActiveStepIndex="1">
   <WizardSteps>
      <asp:WizardStep ID="WizardStep1" runat="server" Title="Step 1">
         First step of Wizard content
      </asp:WizardStep>
      <asp:WizardStep ID="WizardStep2" runat="server" Title="Step 2">
         Second step of Wizard content
      </asp:WizardStep>
   </WizardSteps>
</asp:Wizard>
```

State Comparison

Type of State	Scope of State	Advantages	Disadvantages
Application	**Global to the application**	**•Shared across all clients within a single application**	**•Overuse limits scalability •Not shared across machines in a Web farm •Data cache usually better choice**
Session	**Per client**	**•Can be shared across machines in a Web farm**	**•Requires cookies or URL mangling to manage client association •Off-host storage can be inefficient**
Cookie	**Per client**	**•Works regardless of server configuration •State stored on client •State can live beyond current session**	**•Limited memory (~4KB) •Clients may not support cookies or may explicitly disable them •State is sent back and forth with each request**
Profile	**Per client**	**•Works in a Web farm •Persistent by default •Typesafe •Read on use / write on change only**	**•Less efficient than in-memory •Database schema is predefined**
Cross page POST	**Across `POST` request between two pages**	**•Works regardless of server configuration and between two pages**	**•Previous page is re-evaluated •Creates coupling between pages**
Items	**Within a single request**	**•Convenient mechanism for sharing data between elements of the pipeline**	**•Only spans a single request**

Summary

- Application state
- Session state
- Cookie state
- Query strings
- Items collection
- View state
- Profile
- Cross-page posting
- MultiView/View controls