Telerik Academy

ASP.NET State Management

Session State, Application State, View State

ASP.NET Web Forms

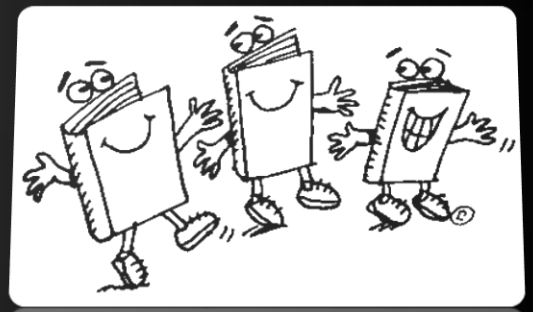
Telerik Software Academy

http://academy.telerik.com

Telerik Academy

Table of Contents

- ASP.NET Intrinsic Objects
- State Management in Web Applications
  - Cookies
  - Hidden Fields
  - Parameterized Addresses
- ASP.NET State Management
  - Client side – View State
  - Server side – Application State, Session State
- Manipulating the HTTP response headers



Telerik Academy

Intrinsic Objects in ASP.NET

Session, Application, Request, Response, …

Telerik Academy

Intrinsic Objects in ASP.NET

- Intrinsic objects in ASP.NET are available in the context of any `Page` or `Control`
  - `Application` (`HttpApplication` class)
  - `Session` (`HttpSession` class)
  - `Request` (`HttpRequest` class)
  - `Response` (`HttpResponse` class)
  - `Server` (`HttpServerUtility` class)
  - `Context` (`HttpContext` class)
  - `Cache` (`System.Web.Caching.Cache` class)

HttpApplication

- `HttpApplication` keeps the application state
- Provides access to other intrinsic objects
  - Properties `Application`, `Context`, `Request`, `Response`, `Server`, `Session` etc.
- Provide events for:
  - Start of a new request
  - Authentication
  - Authorization
  - Working with the cache
  - End of a request

HttpRequest

- `HttpRequest` contains information about the current HTTP request (`Request` object)
  - `ApplicationPath` – root path on the server
  - `Browser` – type, platform, capabilities, etc.
  - `Cookies` – get the cookies collection
  - `HttpMethod` – `GET` / `POST`
  - `QueryString` – e.g. `?id=7&lang=en`
  - `ServerVariables` – IIS server settings
  - `Url` – the requested URL

HttpResponse

- HttpResponse contains information about the HTTP response (Response object)
  - ContentType – MIME type (e.g. image/gif)
  - Charset – response encoding, e.g. UTF8
  - Cookies – sets cookies
  - Expires – sets browser's cache expiration
  - BufferOutput – buffer or not the response
  - ClearHeaders(…), AddHeader(…)
  - Write(…), BinaryWrite(…), WriteFile(…) – send text or binary data to the client

HttpServerUtility

- `HttpServerUtility` – helper methods for processing HTTP requests (`Server` object)
  - `HtmlEncode(…)` – escapes given HTML, e.g. `"<img>"` → `"<img>"`
  - `HtmlDecode(…)` – un-escapes escaped HTML
  - `UrlEncode(…)` – encode string for the browser URL, e.g. `"+.net 4"` → `"%2B.net+4"`
  - `UrlDecode(…)` – decode url-encoded string
  - `MapPath(…)` – returns the server-side path for given resource given as relative path

Telerik Academy

Intrinsic Objects – Examples

```
bool isSecureConnection =
  Request.IsSecureConnection;
```

```
Application.Add("key", "value");
```

```
LabelResult.Text =
  Server.UrlEncode("Did you try ASP.NET 4.0?");
```

```
Response.ContentType = "text/html";
Response.Charset = "UTF-8";
```

```
string imageFileName =
  Server.MapPath("img/logo.gif");
```

```
string url = Request.Url;
```

```
string browserType = Request.Browser.Type;
```

Telerik Academy

Intrinsic ASP.NET Objects

Live Demo

Telerik Academy

Redirecting to Another URL

- `Response.Redirect("Login.aspx")`
  - Client-side redirection (uses HTTP 302 Moved)
  - Asks the browser to request a new URL
  - Changes the URL address in the browser
- `Server.Transfer("WebTest.aspx")`
  - Server-side redirection
  - Keeps the URL in the browser
  - The browser does not even know about the redirection

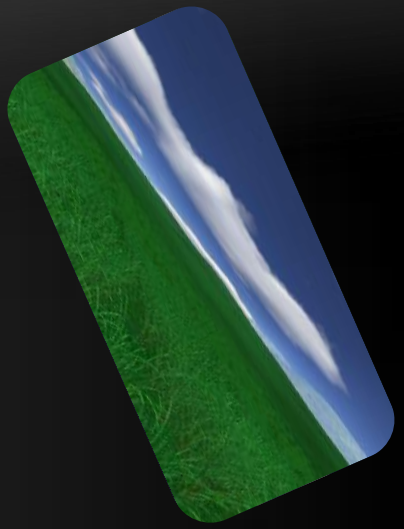
Telerik Academy

Microsoft ASP.net

Client and Server Redirection

Live Demo

Telerik Academy



State Management: Standard Mechanisms in Web Applications

Cookies, Hidden Fields, Parameterized Addresses

Telerik Academy

State Management in Web Applications

- The HTTP protocol is stateless
  - No built-in way to implement a stateful interaction (session / conversation)
- Preserving state between the HTTP requests:
  - Cookies (used by the ASP.NET session)
  - Hidden fields (used by the ASP.NET ViewState)
  - HTML5 local storage / session storage
  - Parameterized addresses (used to implement cookieless session in ASP.NET)

What is a Cookie?

- A small piece of information (up to 4KB)
  - Sent to a browser by the Web server
  - Saved locally at the client as a text file
  - Sent by the browser in all subsequent requests
- Sent as an HTTP header

HTTP Response:

```
Set-Cookie: UserID=baj.ivan; path=/; domain=academy.com;
Expires=Wed, 14 Jun 2015 10:18:14 GMT
```

HTTP Request:

```
Cookie: UserID: baj.ivan;
```

Cookie Properties

- Cookies is ASP.NET are represented by `HttpCookie` objects
  - `Expires`
    - Sets when the validity of the cookie expires
  - `Domain`
    - A domain to which the cookie belongs
  - `Path`
    - Sets the root directory of the cookie
  - `Secure` – transmit over encrypted channel
  - `HttpOnly` – no JavaScript access

Working with Cookies

- For Web applications
  - `System.Web.HttpCookie`
- For client applications
  - `System.Net.Cookie`
- `HttpRequest.Cookies` contains the cookies received by the server
- `HttpResponse.Cookies` contains the cookies sent to the client

Working with Cookies – Example

- Creating a cookie that will be sent to the client Web browser:

```
HttpCookie cookie =
  new HttpCookie("UserName", "baj.ivan");
Response.Cookies.Add(cookie);
```

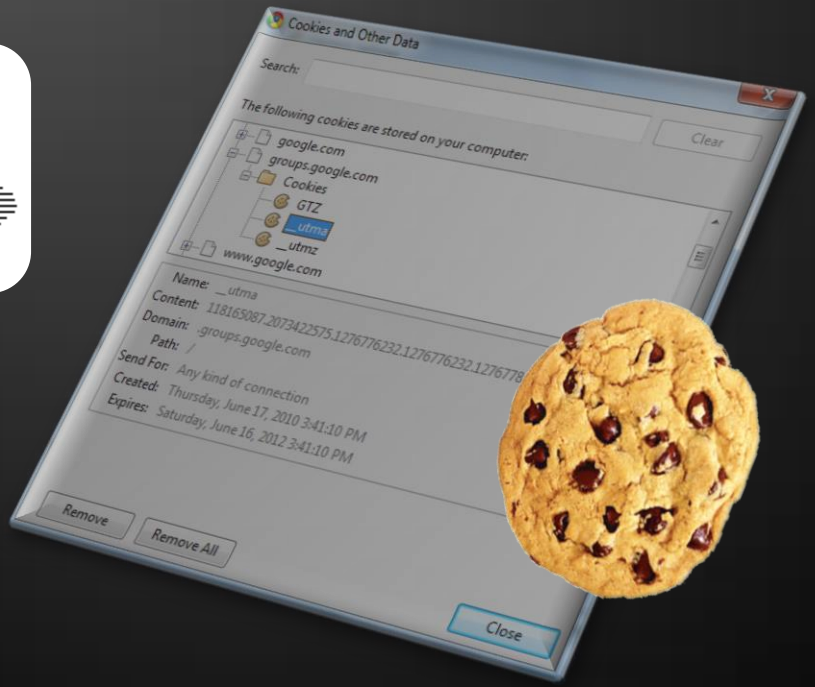
- Reading a cookie received at the server:

```
HttpCookie cookie = Request.Cookies["UserName"];
```

Telerik Academy

Cookies

Live Demo



What are Hidden Fields?

- Hidden form fields keep information, not visible in the Web page, sent on form submit
  - ASP.NET `HiddenField` is a control, which renders as a standard HTML hidden field
  - Not visible in the browser, but you can use it to store information directly in the page

```
<input type="hidden" name="Language" value="English">
```

- Insecure, because malicious user can easily access hidden fields and tamper it
  - ASP.NET ViewState is encrypted for security

Telerik Academy

Local Storage / Session Storage

- HTML5 provides client-side local and session storage (through JavaScript)
  - `localStorage` – stores data long term
  - `sessionStorage` – stores data for one session
- Local data is stored per domain
  - E.g. `academy.telerik.com` and `facebook.com` have different storages

```
localStorage.setItem('myColor', 'red');
var color = localStorage.getItem('myColor'); // -> 'red'
localStorage.removeItem('myColor');
```

Parameterized Addresses

- Also known as query strings
- Setting the parameters in the URL of a page after the "?" sign:

```
http://asp.net/getstarted/default.aspx?tabid=61
```

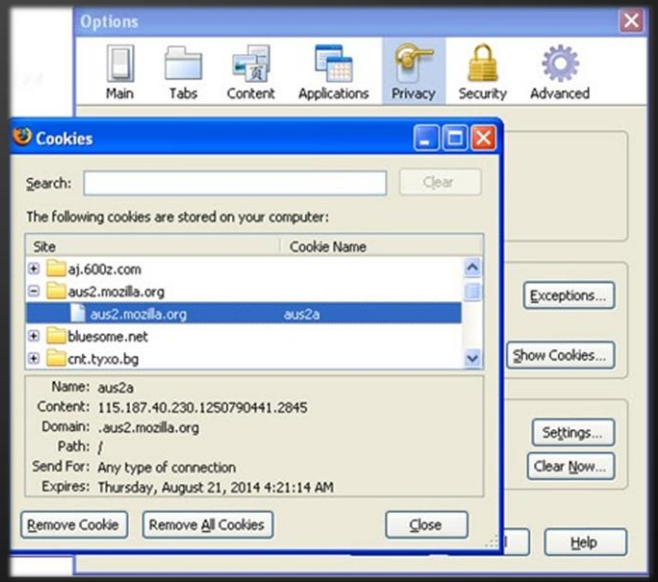
- Reading a query parameter:

```
string selectedTabID = Request.QueryString["tabid"];
```

- Used to pass data from one page to another
- Insecure, because malicious user can copy or change the address

Telerik Academy

ASP.NET
State Management



Telerik Academy

ASP.NET Based State Management

- Client side state
  - View state

- Server side state
  - Application state
  - Session state

Telerik Academy

ASP.NET Client Side State Management

ViewState

ViewState

- ViewState keeps the state of the controls over several consecutive requests to the same page (postbacks)
- Every change in the visualization of a control is saved in the ViewState
  - E.g. adding an element to a list control
- Can save custom data defined by developers

```
ViewState["Username"] = txtUsername.Text.Trim();
```

```
lblUsername.Text = (string) ViewState["Username"];
```

Telerik Academy

ViewState – Behind the Scene

- Data saved in the `ViewState` is:
  - Serialized, encrypted and is sent to the client in a hidden form field:

```
<input type="hidden" name="__VIEWSTATE“ id="__VIEWSTATE"
  value="/wEPDwUJODExMDE5NzY5D2QWAgIDD2QWAgIBDw8WA
  h4EVGV4dAUFS296bW9kZGR67yT0OasTSUMlwIXGj65FNx7ggA==" />
```

- At postback the `ViewState` is deserialized and the state of the controls is restored
  - To accomplish serialization the `ObjectStateFormatter` class is used
  - Encryption is based on machine key

Telerik Academy

ViewState Configuration

- To disable ViewState
  - At page level

```
<%@ Page EnableViewState="false" %>
```

  - At control level

```
<asp:Label ID="labelName" Runat="server"
 Text="Software Academy" EnableViewState="False" />
```

- ViewState support encryption:

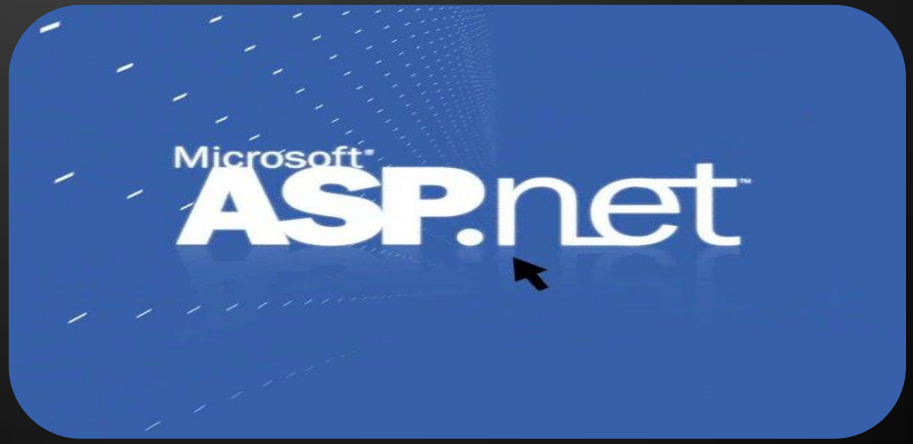
```
<%@ Page ViewStateEncryptionMode="Always" %>
```

- To save the ViewState at the server we can use SessionPageStatePersister

Telerik Academy

ASP.NET Server Side State Management

Application State and Session State



Application State

- The application state is shared storage of information at application level
  - Store information in the memory of the server
  - `Application` – a single object for all clients
- `HttpApplicationState`
  - A dictionary collection accessed through `HttpContext` or `Page`
- Available through all phases of the application lifecycle

Application State (2)

- In order to have synchronized access we use the `Lock()` and `Unlock()`methods

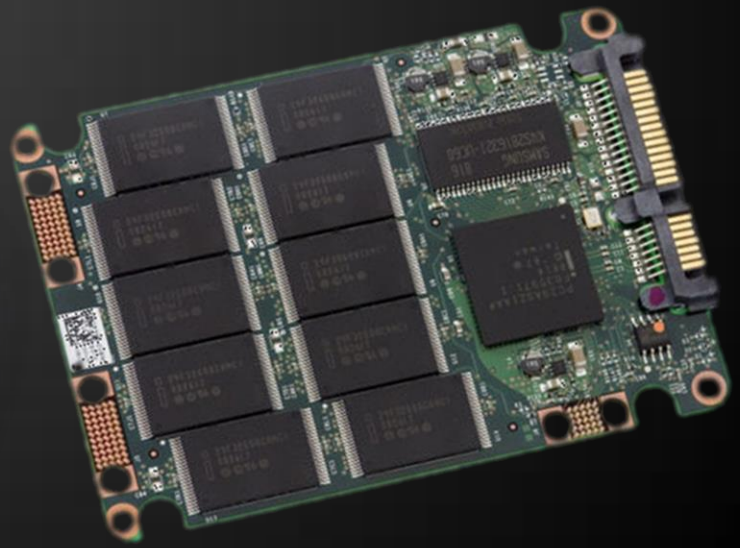
```
Application.Lock();
Application["Users"] = (int) Application["Users"] + 1;
Application.UnLock();
```

- Application State is rarely used in the real world (unlike the cache)
  - Using a database is a better choice
- Useful place to store small amounts of often-used data that is the shared for all users

Telerik Academy

ASP.NET Application State

Live Demo



Session State

- Storage of information at user level (different one for each user)
- The session is active:
  - Till the user closes the browser or
  - A certain period expires (20 minutes for example)
- Every session is identified by a unique `SessionID`
  - Created at first entry in the site
  - Transmitted in a `cookie` by default

Session State (2)

- The `HttpSessionState` dictionary collection is used through `HttpContext` or `Page`

```
Session["username"] = "pesho";
```

```
string = (string) Session["username"];
```

- To handle events fired when a session is started or ended we use `Session_OnStart` and `Session_OnEnd` in the `Global.asax` file
- To deny/restrict access to the session

```
<%@ Page EnableSessionState="False" %>
<%@ Page EnableSessionState="ReadOnly" %>
```

Session Configuration

- We can configure various aspects of the session mechanism
- Use the `sessionState` section in `Web.config`
- Example:

```
<system.web>
  <sessionState
    cookieless="true" mode="InProc"
    timeout="60" cookieName="MySite" />
</system.web>
```

Session Configuration (2)

- Important attributes
  - `Timeout`
    - A period for which the session is active
  - `Mode`
    - Where the session is stored – in the current process, SQL Server, external state server
  - `Cookieless`
    - A session that doesn't use cookies – the `SessionID` is sent as a parameter in the URL

Telerik Academy

ASP.NET Session State

Live Demo

State Management – Recommendations

- Use a wrapper class over the session
  - Put all session data in a property inside this wrapper class
  - All session data should be serializable
- Don't save too much information in the `Session`
- Don't save lots of information in the `ViewState`

Telerik Academy

Manipulating the HTTP Response Headers

The HTTP Response Headers

- HTTP headers are part of the server response
  - Allow the server to pass additional information about the response
    - Page content, MIME type, encoding, caching mode, cookies, HTTP response codes, etc.
  - Provide information about the server to the client Web browser
- Accessible from code behind through `Response.Headers` collection
  - If the body contents is started to render, headers cannot be modified

Manipulating the HTTP Response Headers

- Some widely-used response headers:
  - `HeaderEncoding` – sets header encoding
  - `Headers` – read only collection of headers
  - `ContentType` – MIME type of the output
  - `Expires` – numbers of minutes before page cached in browser expires
  - `StatusCode` – HTTP status code of the output
  - `AppendHeader()` – adds an HTTP header to the output stream

Manipulating the HTTP Response Headers – Example

- Downloading image file generated at the server by an ASP.NET page:

```
Response.Clear();
Bitmap generatedImage = new Bitmap(200, 200);
Graphics gr = Graphics.FromImage(generatedImage);
gr.FillRectangle(Brushes.MediumSeaGreen, 0, 0, 200, 200);
gr.FillPie(Brushes.Yellow, 25, 25,150, 150, 0, 45);
gr.FillPie(Brushes.Green, 25, 25, 150, 150, 45, 315);
Response.ContentType = "image/gif";
generatedImage.Save(
  Response.OutputStream,ImageFormat.Gif);
```

Telerik Academy

Dynamically Generate Image in ASP.NET Page

Live Demo

Telerik Academy

ASP.NET State Management

http://academy.telerik.com

Exercises

1. Create an ASP.NET Web Form, which prints the type of the browser and the client IP address requested `.aspx` page.
2. Create a ASP.NET Web Form which appends the input of a text field when a button is clicked in the `Session` object and then prints it in a `<asp:Label>` control. Use `List<string>` to keep all the text lines entered in the page during the session lifetime.
3. Create two pages that exchange user data with cookies. The first page is a login page. The second one redirects to the first one if the expected cookie is missing. The cookie must expire in 1 minute.

Exercises (2)

4. In ASPX page holding a `TextBox` run a JavaScript code that deletes the ViewState hidden field variable in ASPX page. What happens at postback?
5. Implement a graphical Web counter. It should display as JPEG image the total number of visitors of the requested `.aspx` page since the start of the Web application. Keep the number of visitors in the `Application` object. What happens when the Web server is stopped?
6. Re-implement the previous task to keep the total number of visitors in SQL Server database.

Exercises (3)

7. * Implement the Tic-Tac-Toe game which allows Internet users to play one against another. Multiple game sessions should be supported to run in parallel. The main page (`Default.aspx`) should list all games in the application (games now playing, finished games and games waiting for a second player). The user could start a new game or join existing game which has only one player or view who is the winner of any finished game. When starting new game or joining an existing game, the player should enter his or her name. Players who wait for an oponent to join to their game or to enter a valid move should check repeatedly at 1 second.

Free Trainings @ Telerik Academy

- "Web Design with HTML 5, CSS 3 and JavaScript" course @ Telerik Academy
  - html5course.telerik.com

- Telerik Software Academy
  - academy.telerik.com

- Telerik Academy @ Facebook
  - facebook.com/TelerikAcademy

- Telerik Software Academy Forums
  - forums.academy.telerik.com