Cookies & Session Management

Agenda

Objectives

At the end of this module, you will be able to:

- Develop web applications that use Cookies
- Implement Session tracking in web applications

Cookies

Cookies

- A cookie is a small bit of textual information sent to the client by a web server and web server can later read it back from the browser

- The process of using cookies in servlets
  – Servlet sends a cookie with its response to the client
  – The client saves the cookie
  – The client returns a cookie back with subsequent requests

Cookies (Contd.).

- Uses of cookies
  - Identifying a user during an e-commerce session
  - Avoiding username and password
  - Customizing a website

- Limitations for using cookies for protection of the client
  - A Cookie size is limited to 4KB
  - Supports 20 cookies per website
  - Supports 300 cookies in total

Programming Cookies

- Creating cookies – Use a constructor
  - Cookie (String name, String value)

- Adding cookies to a response– Assume *response* is an HttpServletResponse
  - response.addCookie (cookie1) //cookie1 is a Cookie

- Retrieving cookies from a request – Assume *request* is an HttpServletRequest
  - request.getCookies() //returns array of Cookie or null

- **For example:**
  - Cookie[ ] cookies = request.getCookies();
    - String name = cookies[i].getName();
    - String value = cookies[i].getValue();

Demo for using cookies

- **A servlet that creates and adds cookies**

```
import java.io.*; import javax.servlet.*;   import javax.servlet.http.*;
public class CreateCookieDemo extends HttpServlet {
    public void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out;
        response.setContentType("text/html");
        out = response.getWriter();
    Cookie c1 = new Cookie("CName1", "Anny");
        Cookie c2 = new Cookie("CName2", "123");
        response.addCookie(c1);
        response.addCookie(c2);
        out.println("<HTML><HEAD><TITLE>");
        out.println("Output generated from a Servlet");
        out.println("</TITLE></HEAD><BODY>");
        out.println("2 Cookies are set");
        out.println("</BODY></HTML>");
        out.close();
    }
}
```

Create and add Cookies

1. Firstly, access the servlet CreateCookieDemo in the browser
2. Secondly, access the servlet GetCookiesDemo in the same browser window

Demo for using cookies in jsp

```
<!--  Start.jsp  -->
<body>
<form method="post" action="AddCookie.jsp">
<table>
    <tr>
        <td>Enter a value for MyCookie:</td>
        <td><input type = text name = "data" size="25"></td>
    </tr>
    <tr>
        <td colspan="2" align="center">
            <input type = submit>
        </td>
    </tr>
</table>
</form>
</body>
```

Demo for using cookies in jsp (Contd.).

```
<!--    AddCookie.jsp  -->
<body>
<%
     String data =
  request.getParameter("data");

     // Create  cookie
     Cookie ck = new Cookie("MyCookie",data);

     response.addCookie(ck);
     out.println("MyCookie has been set to :
  "+data);
 %>
</body>
```

What do these two arguments represent?

Demo for using cookies in jsp (Contd.).

```
<!--   GetCookie.jsp  -->
<%
   Cookie[] ck = request.getCookies();
   for(int i=0; i < ck.length; i++)  {
      String name   = ck[i].getName();
      String value  = ck[i].getValue();
      out.println("name : "+ name +"  Value:
  "+value);
   }
%>
```

Session Management

Need for Session Tracking

- Http is a stateless protocol
- Many applications require a series of requests from a same client to be associated with one another
- A mechanism is needed to maintain state across a series of requests from the same user (or coming from the same browser) over some period of time
  - Example: Online shopping cart
- Session tracking is keeping track of what has gone before in a particular conversation
  - Since HTTP is stateless, it does not do this for you
  - You have to do it yourself, in your servlets

Session Tracking Mechanisms

Three different session tracking mechanisms of passing “session id”

- Cookies – You can use a single cookie containing a session id
- URL rewriting - You can append a unique ID after the URL to identify user
- Hidden <form> fields – You can use these to store a unique ID

Cookies for Session Handling - Demo

- An example of using cookies to perform session handling.
  - Every time the servlet SessionCookieServlet services a request, it first checks for cookies in HttpServletRequest by calling request.getCookies()
  - If request contain cookies, the servlet will iterate over the list of cookies looking for a cookie with the name session_id
  - If request contain no cookies or list of cookies has no cookie named session_id, create one and add it to response
  - The code fragment for this is:

```
Cookie c = new Cookie("session_id", "abc123");
response.addCookie(c);
```

Cookies for Session Handling - Demo

- Test functionality of servlet by opening in browser to the SessionCookieServlet
- The first time it runs, you should get a response **“Welcome to simple.com website, A session is created for you."**
- Once you get this message, click Refresh button. You should see a new response **"Hello!!! Anny"**
- The servlet can now identify the user “Anny" by the session ID stored as a cookie

Session Tracking API

- Session tracking API is in javax.servlet.http.HttpSession
  - It is built on top of cookies and URL rewriting

- Servlet container creates HttpSession object
  - Attached to HttpServletRequest object in doXXXX methods

- Contains Methods to
  - View and manipulate information about a session, such as session identifier, creation time, and last accessed time
  - Bind objects to sessions, allowing user information to persist across multiple user connections

Using Session Tracking API

- **Create a session**
  - HttpSession session = request.getSession();
    - Returns session associated with this request
    - If there was no associated session, *new one is created*
    - getSession() method is overloaded
- We can also get an HttpSession object by using following method :
  - HttpSession session = request.getSession(true);
    - getSession(true) works exactly like getSession()
  - HttpSession session = request.getSession(false);
    - This method returns session associated with this request
    - If there was no associated session, *new one is **NOT** created*

HttpSession methods

- Store and retrieve user-defined data in the session

  To store values: session.setAttribute(name, value);

  To retrieve values: Object obj = session.getAttribute(name);

- **boolean session.isNew()**

  Determines if session is new to client (not page)

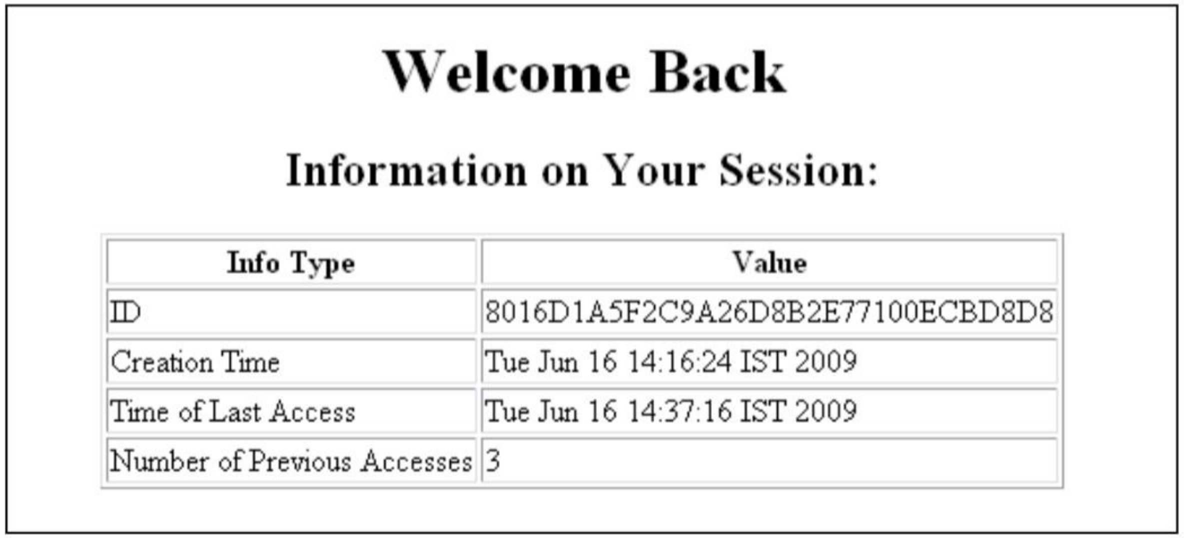
- **public void invalidate()**

  Expires the session and unbinds all objects with it

Demo for Session Information

- A simple servlet example of Session Tracking - It generates a Web page showing some information about the current session.

- Here is a screenshot shown after visiting the page several times without quitting the browser in between



Welcome Back

Information on Your Session:

Info Type	Value
ID	8016D1A5F2C9A26D8B2E77100ECBD8D8
Creation Time	Tue Jun 16 14:16:24 IST 2009
Time of Last Access	Tue Jun 16 14:37:16 IST 2009
Number of Previous Accesses	3

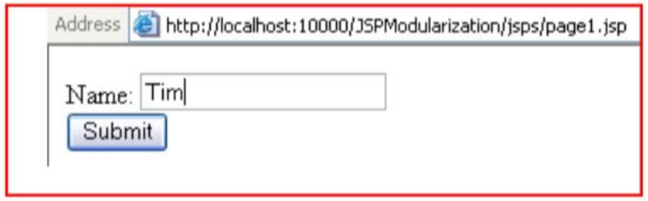
JSP Session Management

- JSP maintains session through the *HttpSession* object
- Session information is stored on the server, and a session ID number is stored in a cookie on the client machine
- Sessions are usually set by server default to expire after 30 minutes of user inactivity

Demo for using session in jsp

An example to track the session between different JSP pages:
The following code page1.jsp takes the input from user

```
<!--  page1.jsp  -->
<html>
    <head>
        <title>page1</title>
    </head>
    <body>
        <form method="post" action="savesession.jsp">
            Name: <b><input type="text" name="name" value=""></b><br>
            <input type="submit" value="Submit">
        </form>
    </body>
</html>
```


Address http://localhost:10000/JSPModularization/jsps/page1.jsp

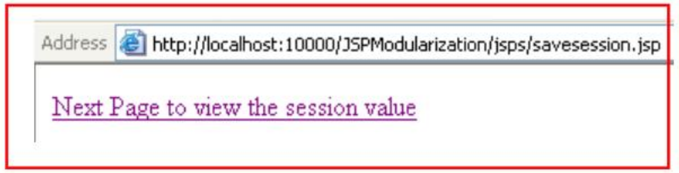
Name: Tim

Demo for using session in jsp (Contd.).

- The savesession.jsp saves the name into session

```
<!--  savesession.jsp  -->
<%
        String name = request.getParameter("name");
        session.setAttribute(“username", name);
%>
<html>
    <head>
        <title>Name Saved</title>
    </head>
    <body>
        <p><a href="showsession.jsp">Next Page to view the session
  value</a><p>
    </body>
</html>
```

Saves user’s name in the session


Address http://localhost:10000/JSPModularization/jsps/savesession.jsp

Next Page to view the session value

Quiz

- Pick up the valid methods from the following list, using which we can get an HttpSession object.
  1. request.getSession() ←
  2. request.getSession("true")
  3. request.getSession("false")
  4. request.getSession(true) ←
  5. request.getSession(false) ←
  6. request.getSession(1);

The servlet sends cookies to the browser by using the method

HttpServletResponse.addCookie(*Object of type Cookie*)

The servlet retrieves cookies by using the method

HttpServletRequest..getCookies()

Summary

In this module, you were able to:

- Use ServletConfig and ServletContext object in web applications
- Create web applications that implement Servlet Chaining
- Develop web applications that use Cookies
- Implement Session tracking in web applications

References

1. Oracle (2006).Understanding and using Servlet Sessions. Retrieved April 26, 2012, from, http://docs.oracle.com/cd/B31017_01/web.1013/b28959/sessions.htm

2. Tutorial Point (2012). JSP - Cookies Handling. Retrieved April 26, 2012, from, http://www.tutorialspoint.com/jsp/jsp_cookies_handling.htm

3. JavaTPoint (2012). ServletConfig Interface. Retrieved April 26, 2012, from, http://www.javatpoint.com/sonoojaiswal/servletconfig

Thank You