Networks and Database Systems

Professor Tom Ellman
Lecture 12
HTTP is “Stateless”

- The HTTP protocol has no built-in notion of multiple request/receive transactions between client and server.
- Applications can build their own systems for tracking multiple transactions.
- Cookies provide this capability.
- Allows web sites to engage in mischief!
Cookies

- A cookie encodes a pair of strings: (name,value).
- Sent from web server to web browser.
- Stored on client’s computer.
- Returned to web server on subsequent queries to web server.
- Allows web sites to keep track of client information over extended periods of time.
Cookies Passed between Client and Server

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request (Cookies = {})</td>
<td>Response (Cookies={name1=v1})</td>
</tr>
<tr>
<td>Request (Cookies={name1=v1})</td>
<td>Response (Cookies={name2=v2})</td>
</tr>
<tr>
<td>Request (Cookies={name1=v1;name2=v2})</td>
<td>… Cookie “name1=v1” expires …</td>
</tr>
<tr>
<td>Request (Cookies={name2=v2})</td>
<td></td>
</tr>
</tbody>
</table>
Browser to Server

GET /index.html HTTP/1.1
Host: www.example.org

… Remainder of Request …
HTTP/1.1 200 OK
Content-type: text/html
Set-Cookie: name1=value1; Expires=Wed, 09 Jun 2021 10:18:14 GMT

… Content of Page …
Browser to Server

GET /spec.html HTTP/1.1
Host: www.example.org
Cookie: name1=value1

… Remainder of Request …
Java HttpCookie Class

• Constructor:
  ```java
  HttpCookie(String name, String value).
  ```

• Attributes:
  – Time created.
  – Maximum age, measured in seconds.
  – Domain on which cookie was constructed.
  – Path on server to which browser returns cookie.
  – …
Expiration of Cookies

- Cookies expire when time since creation exceeds maximum age.

- Maximum age of -1 indicated cookie expires when browser closes.
Browsers and Cookie Transmisssion

- Browser sends cookies only to:
  - Servers on the domain stored in the cookie.
  - In a directory descended from the path stored in the cookie.
- A privacy / security feature: Keeps one server from unintentionally passing your data to another server.
- This “security feature” can very easily be circumvented.
Cookies returned with A are sent back to A, Y and R.
Cookies returned with Y are sent back to Y, but not to A and R.
Cookies returned with R are sent back to R, but not to A and Y.
Java and Cookie Transmission

• Suppose HTTP is invoked from a Java applet, e.g., using URLConnection?
• Does applet have access to the browser’s cookie jar? (Apparently not.)
• How can cookies be transmitted to the server via the URLConnection class?
• How can cookies be retrieved from the server via the URLConnection class?
Java Classes for Cookie Management

- HttpCookie
- CookieHandler
- CookieManager
- CookiePolicy
- CookieStore
public String getCookieFromURLConnection(URLConnection conn, String name) {
    try {
        Map<String, List<String>> headers = conn.getHeaderFields();
        // "Set-Cookie" may be capitalized differently some sites.
        List<String> values = headers.get("Set-Cookie");
        for (String cookie : values) {
            if (cookie.startsWith(name + "=")) {
                return cookie;
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
}
public void setCookieInURLConn(URLConnection conn,  
    String name, String value) {
    String cookie = name + "=" + value;
    try {
        conn.setRequestProperty("Cookie", cookie);
    } catch(Exception e) {
        System.out.println("Unable to set cookie in URL connection");
        e.printStackTrace();
    } 
}
Session

- Stores data associated with one client across multiple HTTP queries.
- Sessions can be accessed by servlets and JSPs.
- Session can keep track of the state of the client/server interaction.
- Implemented using cookies to keep track of client identity.
Session Communication Diagram

Client

Request (Cookies = {}) → session = new HttpSession()
Response (Cookies={JSESSIONID="1234"})

Request(Cookies={JSESSIONID="1234"})

Server

Request(Cookies={JSESSIONID="1234"})

Request(Cookies={JSESSIONID="1234"})

Request(Cookies={JSESSIONID="1234"})
Session Expiration

• Session is terminated when client closes browser.

• Session is terminated when time limit (set in web.xml file) passes.
HttpSession Objects

- Available in servlet via method of HttpServletRequest objects.
- HttpSession class has methods for set, get and removal of attributes.
- Session attributes can be used to store client-specific data in a Java servlet.
- Session data is not persistent: It may not remain available if the servlet or server is restarted.
Storing User Data in HiddenFields v. Storing User Data in the Session

- **EditConfirmProcessServletController:**
  - Stores user’s data in hidden fields on client host.
  - Data does not expire over time.
  - Easier to code (if you know only HTML).

- **EditConfirmProcessServletControllerAlt:**
  - Stores user’s data in session on server host.
  - Avoids transmission of data back and forth.
  - Easier to code (if you know Java).
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
    <head>
        <meta http-equiv="content-type" content="text/html;charset=utf-8">
        <title>Edit Page</title>
    </head>
    <body>
        <p>This is a simple HTML page that has a form in it.</p>
        <form action="Controller">
            <p>If there is a value for the hobby in the query string, then it is used to initialize the hobby element.</p>
            <p>Hobby: <input type="text" name="hobby" value="${param.hobby}">
                <input type="submit" name="confirmButton" value="Confirm"></p>
        </form>
    </body>
</html>
public class Controller extends HttpServlet {

    @Override
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException {
        String address;
        if (request.getParameter("confirmButton") != null) {
            address = "Confirm.jsp";
        } else if (request.getParameter("processButton") != null) {
            address = "Process.jsp";
        } else {
            address = "Edit.jsp";
        }
        request.getRequestDispatcher(address).forward(request, response);
    }
}
EditConfirmProcessServletController
Confirm.jsp (Note use of hidden field.)

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
<meta http-equiv="content-type"
content="text/html; charset=utf-8">
<title>Confirmation Page</title>
</head>
<body>
<p>The value of the hobby that was sent to
this page is: <b>${param.hobby}</b>. <form action="Controller">
  
  If there is an error, then please click the edit button to
  return to the previous page.
  
  <input type="hidden" name="hobby" value="${param.hobby}"/>
  <input type="submit" name="editButton" value="Edit">
  <input type="submit" name="processButton" value="Process">

</form>
</body>
</html>
EditConfirmProcessServletController
Process.jsp

<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
 <head>
  <meta http-equiv="content-type"
       content="text/html;charset=utf-8">
  <title>Process Page</title>
 </head>
 <body>
  <p>The value of the hobby that was sent to this page is: <b>${param.hobby}</b>.</p>
 </body>
</html>
This is a simple HTML page that has a form in it.

Hobby initialized from session.

Hobby: <input type="text" name="hobby" value="${hobby}"
    <input type="submit" name="confirmButton" value="Confirm">
</form>
public class Controller extends HttpServlet {
    @Override
    public void doGet(HttpServletRequest request,
        HttpServletResponse response)
        throws IOException, ServletException {
        String address;
        if (request.getParameter("confirmButton") != null) {
            String hobby = request.getParameter("hobby");
            HttpSession session = request.getSession();
            session.setAttribute("hobby", hobby);
            address = "Confirm.jsp";
        } else if (request.getParameter("processButton") != null) {
            address = "Process.jsp";
        } else {
            address = "Edit.jsp";
        }
        request.getRequestDispatcher(address).forward(request, response);
    }
}
Hobby initialized from parameters: <b>${param.hobby}</b>.  
Hobby initialized from session: <b>${hobby}</b>. 

If there is an error, then please click the edit button to
return to the previous page.

<input type="submit" name="editButton" value="Edit">
<input type="submit" name="processButton" value="Process">
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
<head>
  <meta http-equiv="content-type" content="text/html;charset=utf-8">
  <title>Process Page</title>
</head>
<body>
  <p>Hobby initialized from session: <b>${hobby}</b>.</p>
</body>
</html>
Cookie Experiments

- Start Firefox.
- Invoke Tools-Options-Privacy-ShowCookies.
- Delete all your cookies.
- Visit a large web site: (NY Times, Amazon).
- Invoke Tools-Options-Privacy-ShowCookies.
- Notice what cookies you have received.
- Many parties are involved in your transactions!
Making Mischief with Cookies

• The cookies stored on your computer form a record of your web activity.

• They may record the sites you’ve visited, the pages you’ve viewed, the links you’ve clicked and more.

• Cookies provide a mechanism by which web sites can lead you to reveal information about yourself to third parties.
Scenario

- Tom shops for a TV on several sites, including Amazon, Best Buy, etc.
- He likes what he sees about Samsung model X, but decides to wait for a lower price.
- The next week, Tom visits a Yahoo news site, which he has never visited before.
- Yahoo is advertising that very same Samsung TV model X on its front news page!
- Coincidence?
Scenario, Cont’d

• Tom is suspicious to the point of paranoia.
• He deletes his cookies and closes his browser.
• Then he starts up his browser and opens the front news page at Yahoo.
• Lo and behold:
  – No Samsung TV model X ad.
  – No TV ads at all.
• What is going on?
AmaBuy enables Agent to place cookie recording Tom’s TV interest on Tom’s machine:

AmaBuy

Tom

Agent

view=SamsungTVX

Embed[Agent.com?like=SamsungTVX]

like=SamsungTVX

Cookies={like=SamsungTVX}
Yahoo sends Tom page with embedded ad based on his TV interest via Agent:

Yahoo

Tom

Agent

WorldNews

Embed[Agent.com?action=GetAd]

action=GetAd
[Cookies={like=SamsungTVX}]

Samsung TV X Ad
Comments

• No direct communication is necessary between AmaBuy and Yahoo.
• Neither AmaBuy, Yahoo nor Agent need ever know Tom’s true identity.
• AmaBuy and Agent know Tom likes Samsung TV X, but Yahoo does not necessarily know.
• If Agent does this work for lots of vendors, it can build a detailed profile of the person we know as “Tom”.
  – Goes to movies in Poughkeepsie.
  – Reads books on IT and liberal arts education.
  – Has children and coaches soccer.
  – Interested in politics and probably vegetarian.
  – …?… Name, Address, Phone, SS# … ? …
Monkey Business

- Spyware looking in the browser’s cookie jar.
- Hackers stealing cookies in transit.
- Zombie cookies that come back from death.
Class Presentations

• Student proposes topic by professor by sending a 100 word description by email.
• Presentation dates are indicated in the schedule on the class wiki.
• Each individual presentation is 20 minutes long, plus 5 minutes for questions.
• Group presentations are appropriately longer.
Previous Topics

• Bit Torrent
• Implementing MMORPGS
• Implementing Multi-Player Shooters.
• Bot Nets
• Hacking versus Security.
• Grid Computing.
• Networked Social Activism
• TOR Onion Routing
• NSA penetrating everything