Server-side Web Programming

Lecture 8:
Introduction to Sessions

Sessions

• Session:
  Set of pages submitted by user to accomplish goal
  – Example: Most on-line shopping
Need for Session Handling

- Problem:
  No easy way to associate steps if multiple clients
  - Nothing built into the web allows server to know where a request is coming from.
  - Server may have to simultaneously manage thousands of sessions.

Session Handling

- Basic steps:
  - Assign each new client unique ID at start of session.
  - Pass ID to client as part of each response
    - Now client knows it as well
    - Stored as cookie by default
  - Client passes ID back to server with subsequent requests
    - Server can associate this request can be associated with initial request.
  - Server stores client data in table indexed by session ID
Session Handling

- Sessions can be accessed from both servlet and JSP
  - Servlet: Construct a new session object from the request
    ```java
    HttpSession session = request.getSession();
    ```
  - JSP: Just use built-in `session` object which Tomcat creates from request (like `request` object)

Creating a New Session

- Done automatically first time session requested by servlet or JSP
  - `HttpSession session = request.getSession();`
    in servlet
  - Use of `session` object in JSP
- Tomcat:
  - Knows this because no session ID included in request
  - Generates new ID not used for current session (or recent past session)
  - Creates new session table entry for that ID
Passing Session IDs

- Automatically included in **response** sent back to client
- Stored in **cookie** on client machine
  - Cookies only data that **persist** between pages in browser
  - Associated with server domain name, directory, etc.

---

Client computer

- **Browser**
  - Store session ID
- **Cookies**
  - session ID + server name

Server

- **Servlet** or **JSP**
- Create response

Response = web page + session ID

---

Passing Session IDs

- Automatically included in **request** sent in future to **same server**
  - All cookie values associated with server sent with request
  - Server now knows who client is!

Client computer

- **Browser**
  - Retrieve session ID
- **Cookies**
  - session ID + server name

Server

- **Servlet** or **JSP**
- Handle request

Request = parameters + session ID
Associating Session Data

- Servlets/JSPs can **store** data associated with session ID
- Servlets/JSPs can **look up** that data in future when passed the session ID in request

![Diagram of session data association]

Storing Session Data

- **Syntax:**
  ```java
  session.setAttribute("name", object);
  ```
  - Like parameters, session data stored as name/value pairs
  - Like attributes, can store **any Java object**
    - Often a “shopping cart” object

```
<table>
<thead>
<tr>
<th>All session data</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
</tr>
<tr>
<td>...</td>
</tr>
<tr>
<td>Session ID = fieh4K39Rdk</td>
</tr>
<tr>
<td>Session data</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>email</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
```

Storing Session Data

```java
protected void processRequest(HttpServletRequest request, )
    throws ServletException, IOException {
    String name = request.getParameter("customerName");
    String email = request.getParameter("customerEmail");
    HttpSession session = request.getSession();
    session.setAttribute("name", name);
    session.setAttribute("email", email);
    RequestDispatcher dispatcher = getServletContext().get
    dispatcher.forward(request, response);
}
```

Retrieving Session Data

- Syntax:
  ```java
type variable = (type)session.getAttribute("name");
```
  
  - Same syntax as retrieving attribute added to request
  - Since value could be any object, must cast back to original type

- Will be null if
  - No session created for this client
  - That value not stored for this client
Retrieving Session Data

Session Example

```
<title>Order Confirmation</title>
</head>
<body>

<% int quantity = Integer.parseInt(request.getParameter("quantity"));
    double totalCost = quantity * 9.95;
    String name = (String)session.getAttribute("name");
    String email = (String)session.getAttribute("email");%>

<h2>Order Confirmation</h2>

<p>Thank you for your order of <%= quantity %> widgets, <%= name %>.<br>
Your bill will be $<%= totalCost %>.<br>You will shortly receive an email confirmation at <%= email %>.
</p>
```

StoreInfo servlet creates session and stores the information in new session

 Sessions ID = fieh4K39Rdk

<table>
<thead>
<tr>
<th>Session data</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>&quot;Mai Anh Tho&quot;</td>
</tr>
<tr>
<td>email</td>
<td>&quot;<a href="mailto:tho@hcmuaf.edu.vn">tho@hcmuaf.edu.vn</a>&quot;</td>
</tr>
</tbody>
</table>

"Mai Anh Tho", tho@hcmuaf.edu.vn passed to server
Session Example

Cookies

ID = fieh4K39Rdk
server = www.widgets.com

StoreInfo servlet adds session ID to response

getQuantity JSP sends session ID to client as part of page

Response = page + Session ID

Sending request to www.widgets.com, so retrieve its cookies

Server at www.widgets.com

quantity = 27 & ID = fieh4K39Rdk submitted in request
Session Example

Receipt JSP retrieves information associated with the session ID and inserts into the response page.

Session ID = fieh4K39Rdk

<table>
<thead>
<tr>
<th>Session data</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
</tr>
<tr>
<td>email</td>
</tr>
</tbody>
</table>

URL Encoding

- Many users disable cookies!
  - Often default in some browsers
  - Need alternative way of storing session information on server

Solution:

- Pass session ID to the client as part of every response
- Insure that client sends that session ID back to the server as part of every request
- Since you have no way of knowing whether user has cookies, you must do this!
URL Encoding

- **Syntax:**
  ```html
  <form action="<%= response.encodeURL("url") %>"
  method=...>
  </form>
  ```

- If browser detects cookies **not enabled**, it **appends** the session ID to the request
  - Like other form data

```
16  <body>
17  <form action="<%= response.encodeURL("Receipt.jsp") %>" method="get">
18  <table cellspacing="5">
19  <tr>
```

Session Expiration

- Can set time until session expiration
  - Property of `web.xml` file

```
General
Description:
  Session Timeout: [24:00] min.
```

- Session expires if **no request** within time limit
  - Session inactive
  - Session id and all attributes destroyed
  - Request for session attributes returns **null**
Sessions for Access Control

- Users can skip pages in a sequence
  - Bookmarked page in middle

Goal:
Prevent users from directly going to other pages without first going to initial page

Solution:
- Set session attribute at servlet called from first page
  - Use this in other pages to determine whether initial page requested in this session

```java
protected void processRequest(HttpServletRequest request
  throws ServletException, IOException {
    HttpSession session = request.getSession();
    // This attribute used for access control
    session.setAttribute("sessionOK", "yes");
```
Sessions for Access Control

• All other JSPs test whether attribute is \texttt{null}
• If so, \texttt{redirect} to another page
  – Initial page in sequence
  – Error page telling session has expired
• Syntax for redirection from JSP:
  \texttt{<jsp:forward page=\textit{url to forward to}/>}

\begin{verbatim}
<% if (session.getAttribute("sessionOK") == null) { %>
  <jsp:forward page="SessionExpired.jsp"/>
<% } %>
<%
\end{verbatim}