Advanced Internet Technology Lab

Lab # 8

Session Tracking

Eng. Haneen El-Masry

April, 2014
Objectives

- Understanding what is Session Tracking.
- Understanding the Session Tracking API.

The Need for Session Tracking

HTTP is a “stateless” protocol: each time a client retrieves a Web page, the client opens a separate connection to the Web server and the server does not automatically maintain contextual information about the client.

What is Session Tracking?

Session tracking enables you to track a user's progress over multiple servlets or HTML pages, which, by nature, are stateless.

A session is defined as a series of related browser requests that come from the same client during a certain time period.

Session tracking ties together a series of browser requests that may have some meaning as a whole, such as a shopping cart application.

Session Tracking

A session is associated with a particular client. When the session is created on the server, it is associated with a unique ID. The browser must provide this session ID with its request in order for the server to find the session data again. The server attempts to store this ID using:

1. Cookies

On the initial request a servlet could do something like the following:

```java
String sessionId = makeUniqueString();
HashMap sessionInfo = new HashMap();
HashMap globalTable = findTableStoringSessions();
globalTable.put(sessionId, sessionInfo);
Cookie sessionCookie = new Cookie("JSESSIONID", sessionId);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```
Once the cookie is set, each time the browser sends a request to the server it includes the cookie containing the ID. Then, the server could use the globalTable hash table to associate a session ID from the JSESSIONID cookie with the sessionInfo hash table of user-specific data.

2- URL Re-Writing

The server encodes the ID into the URL links in the pages sent back to the client. For this reason, you should always use the encodeURL() method when you include URLs in your servlet response. The client appends some extra data on the end of each URL. That data identifies the session, and the server associates that identifier with user-specific data it has stored. For example:

```
http://host/path/file.html;jsessionid=a1234
```

3- Hidden Form Field

HTML forms can have an entry that looks like the following:

```
<INPUT TYPE="HIDDEN" NAME="session" VALUE="a1234">
```

This entry means that, when the form is submitted, the specified name and value are automatically included in the GET or POST data.

Session Tracking Basics in Servlets

Servlets provide an outstanding session tracking solution: the HttpSession API. This high-level interface is built on top of cookies or URL rewriting.

Using sessions in servlets is straightforward and involves four basic steps:

1- Accessing the session object associated with the current request:

```
HttpSession session = request.getSession();
```

Session objects are of type HttpSession, but they are basically just hash tables that can store arbitrary user objects (each associated with a key).
Behind the scenes:

- The system extracts a user ID from a cookie or attached URL data, then uses that ID as a key into a table of previously created HttpSession objects.
- If no session ID is found in an incoming cookie or attached URL information, the system creates a new, empty session.
- And, if cookies are being used (the default situation), the system also creates an outgoing cookie named JSESSIONID with a unique value representing the session ID.

Note:

Call request.getSession() before you send any document content to the client.

2- Looking up information associated with a session:

```java
session.getAttribute("key");
```

- Call getAttribute on the HttpSession object.
- The return type is Object, so you must cast the return value to the appropriate type.
- Check whether the result is null.

3- Storing information in a session:

```java
session.setAttribute("key", value);
```

Value is an object.

4- Discarding session data:

Call removeAttribute to discard a specific value.

```java
session.removeAttribute("key");
```

Call invalidate to discard an entire session.

```java
session.invalidate();
```

Call logout to log the client out of the Web server and invalidate all sessions associated with that user.

```java
session.logout();
```
Example 1: A Servlet That Shows Per-Client Access Counts

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    HttpSession session=request.getSession();
    String header;
    Integer accessCount=(Integer)session.getAttribute("accessCount");
    if(accessCount==null){
        accessCount=0;
        session.setAttribute("accessCount",accessCount);
        header="Welcome, Newcomer";
    }
    else{
        header="Welcome Back";
        accessCount++;
        session.setAttribute("accessCount", accessCount);
    }
    out.print("<html><body><h1>"+header+"</h1>" +
    "<h2>Information on Your Session:</h2><br/>
    "<table border=1>
    "<tr bgcolor="#FFAD00">
    "<th>Info Type</th>
    "<th>Value</th>
    "</tr>
    "<tr>
    "<td>Id</td><td>" + session.getId()+"</td><td><tr>
    "<td>Creation Time</td><td>"+new Date(session.getCreationTime())+"</td><td><tr>
    "<td>Last Access</td><td>" + new Date(session.getLastAccessedTime())+"</td><td><tr>
    "<td>Number of Previous Accesses</td><td>" +
    "<td>" + accessCount + "</td><td><tr>
    "</table>" +"</body></html>";
}
```
Running

➤ 1st Request

Welcome, Newcomer

Information on Your Session:

<table>
<thead>
<tr>
<th>Info Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>D1F8044BA99E2C1F638FCC09723EA733</td>
</tr>
<tr>
<td>Creation Time</td>
<td>Sun Apr 20 20:03:34 GMT+02:00 2014&lt;</td>
</tr>
<tr>
<td>Time of Last Access</td>
<td>Sun Apr 20 20:03:34 GMT+02:00 2014</td>
</tr>
<tr>
<td>Number of Previous Accesses</td>
<td>0</td>
</tr>
</tbody>
</table>

➤ 11th Request

Welcome Back

Information on Your Session:

<table>
<thead>
<tr>
<th>Info Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>D1F8044BA99E2C1F638FCC09723EA733</td>
</tr>
<tr>
<td>Creation Time</td>
<td>Sun Apr 20 20:18:49 GMT+02:00 2014</td>
</tr>
<tr>
<td>Time of Last Access</td>
<td>Sun Apr 20 20:18:49 GMT+02:00 2014</td>
</tr>
<tr>
<td>Number of Previous Accesses</td>
<td>10</td>
</tr>
</tbody>
</table>
Example 2: Accumulating a List of User Data

➤ **OrderForm.html**

```html
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Order</title>
</head>
<body>
<h1>Order Form</h1>
<form action="/SessionTracking/ShowItems">
New Item to Order:
<input type="text" name="item"/>
<input type="submit" value="Order and Show"/>
</form>
</body>
</html>
```

➤ **ShowItems.java**

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    HttpSession session=request.getSession();
    ArrayList items=(ArrayList)session.getAttribute("items");
    if(items==null){
        items=new ArrayList();
        session.setAttribute("items",items);
    }
    String purchased=request.getParameter("item");
    if(purchased!=""){
        items.add(purchased);
        session.setAttribute("items",items);
    }
    out.print("<html><body><h1>Items Purchased</h1>");
    if(items.size()==0)
        out.print("No items");
    else{
        out.print("<ul>");
        for(int i=0;i<items.size();i++){
            out.print("<li>"+items.get(i)+"</li>");
        }
        out.print("</ul>");
    }
    out.print("</body></html>");
}
```
Running

Order Form

New Item to Order: | Order and Show

Items Purchased

No items

Order Form

New Item to Order: I1 | Order and Show

Items Purchased

- I1
Order Form

New Item to Order: 12
Order and Show

Items Purchased

- I1
- I2
Exercise

Using session tracking, program a web application runs as following:

Welcome To Books Shopping Site

Kids Books

Computer Books

If you click on Kids Books

All-Time Best Children's Fantasy Books

The Chronicles of Narina by C.D. Lewis ($20)

Add To shopping Cart

The Harry Potter Series by J.K. Rowling (60$)

Add To shopping Cart

Back To the Main Page

Back To the Main Page: returns you to the previous page.

When you choose a book to purchase
The Status of Your Order

<table>
<thead>
<tr>
<th>Book</th>
<th>Unit Cost</th>
<th>Count</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Chronicles of Narnia by C.D. Lewis</td>
<td>20</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>20</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Update Order

Checkout

**Update Order**: returns you to the main page.

If you returned to the main page and click on Computer Books

All-Time Best Computer Books

*Core Servlets and JavaServer Pages 2nd Edition (40$)*

Add To shopping Cart

*Core Web Programming, 2nd Edition (60$)*

Add To shopping Cart

Back To the Main Page

When you choose a book to purchase
### The Status of Your Order

<table>
<thead>
<tr>
<th>Book</th>
<th>Unit Cost</th>
<th>Count</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Servlets and JavaServer Pages 2nd Edition</td>
<td>40</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>The Chronicles of Narnia by C.D. Lewis</td>
<td>20</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>60</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Update Order**

**CheckOut**

If you return to the main page and order a book

### The Status of Your Order

<table>
<thead>
<tr>
<th>Book</th>
<th>Unit Cost</th>
<th>Count</th>
<th>Total Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Core Servlets and JavaServer Pages 2nd Edition</td>
<td>40</td>
<td>2</td>
<td>80</td>
</tr>
<tr>
<td>The Chronicles of Narnia by C.D. Lewis</td>
<td>20</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>100</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Update Order**

**CheckOut**

**CheckOut**: returns you to the final page.

**Thank You :)**