Advanced Internet Technology Lab

Lab # 7

Handling Cookies

Eng. Haneen El-Masry

April, 2014
**Objectives**

- Understanding what are Cookies.
- Understanding the benefits of cookies.
- Sending outgoing cookies.
- Receiving incoming cookies.
- Specifying cookie attributes.

**What are Cookies?**

Cookies are small bits of textual information that a Web server sends to a browser and that the browser later returns unchanged when visiting the same Web site or domain.

**Typical Uses of Cookies**

- Identifying a user during an e-commerce session.
- Avoiding username and password.
- Customizing a site.
- Focusing advertising.

**Stages to using a cookie**

There are generally two stages to using a cookie:

1. Sending Cookies to the Client.
2. Reading Cookies from the Client.

**Sending Cookies to the Client**

1. **Create a Cookie object.**

Call the Cookie constructor with a cookie name and a cookie value, both of which are strings.

```java
Cookie c = new Cookie("Cname", "Cvalue");
```
2. Set the maximum age.

To tell the browser to store the cookie on disk instead of just in memory, use setMaxAge method that its argument is in seconds.

\[
c.setMaxAge(3600);
\]

3. Place the Cookie into the HTTP response.

Use addCookie method that is in HttpServletResponse.

\[
response.addCookie(c);
\]

Notes:

- addCookie method add the cookie to “set-cookie” header in the HTTP response.
- If you forget this step, no cookie is sent to the browser.

Reading Cookies from the Client

1. Call getCookies method.

getCookies() method is in HttpServletRequest class. It returns an array of Cookie objects that exist in “Cookies” HTTP request header.

\[
Cookie [] cookies = request.getCookies();
\]

2. Find the value of the cookie of interest.

Loop down the array, calling getName() on each entry until you find the cookie of interest. Then use the value of the cookie that is get from getValue() method.
Using Cookie Attributes

Before adding the cookie to the response header, you can set various characteristics of the cookie. Also, you can retrieve the attribute value.

**Note:**

The attributes are part of the header sent from the server to the browser; they are not part of the header returned by the browser to the server.

**Methods that set and get the cookie attributes**

- **getName()**
  
  The getName method retrieves the name of the cookie. There is no setName method, you cannot change the name once the cookie is created.

- **setValue(String cookieValue) / getValue()**
  
  The setValue method specifies the value associated with the cookie; getValue looks it up.

- **setMaxAge(int lifetime) / getMaxAge()**
  
  These methods tell how much time (in seconds) should elapse before the cookie expires. A negative value, which is the default, indicates that the cookie will last only for the current browsing session. Specifying a value of 0 instructs the browser to delete the cookie.

- **setDomain(String domainPattern) / getDomain()**
  
  These methods set or retrieve the domain to which the cookie applies. Normally, the browser returns cookies only to the exact same hostname that sent the cookies.

- **setPath(String path) / getPath()**
  
  These methods set or get the path to which the cookie applies. If you don’t specify a path, the browser returns the cookie only to URLs in or below the directory containing the page that sent the cookie.

To specify that a cookie apply to all URLs on your site, use cookie.setPath("/").
 **setComment (String comment) / getComment()**

These methods specify or look up a comment associated with the cookie.

 **setSecure (boolean secureFlag) / getSecure()**

This pair of methods sets or gets the boolean value indicating whether the cookie should only be sent over encrypted (i.e., SSL) connections. The default is false; the cookie should apply to all connections.

 **setVersion (int version) / getVersion()**

These methods set and get the cookie protocol version with which the cookie complies.
Example 1: Using Cookies to Remember User Preferences

FormServlet

- Uses Empty values if no cookies are found.
- Uses cookie values to prepopulate form field values.
- Notify the user to the missing form values.

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    String FName="";
    String LName="";
    String Email="";
    String Missing="";
    Cookie []cookies=request.getCookies();
    if(cookies!=null){
        for(int i=0;i<cookies.length;i++){
            if(cookies[i].getName().equals("FName"))
                FName=cookies[i].getValue();
            else if(cookies[i].getName().equals("LName"))
                LName=cookies[i].getValue();
            else if(cookies[i].getName().equals("Email"))
                Email=cookies[i].getValue();
        }
        if(FName==""||LName==""||Email==""){
            Missing="Missing: ";
            if(FName=="")
                Missing+=" FName ";
            if(LName=="")
                Missing+=" LName ";
            if(Email=="")
                Missing+=" Email ";
        }
    }
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    out.print("<html><body><h1> Please Register </h1>"+
        "<form action="/Cookies/Register"">+
        "<h5 style="color:red">"+Missing+"</h5>"+
        "FName: <input type="text" name="FName" value=""+FName+""/>"+"</br>" +
        "LName: <input type="text" name="LName" value=""+LName+""/>"+"</br>" +
        "E-mail : <input type="text" name="Email" value=""+Email+""/>"+"</br>"+"<input type="submit" value="OK"/>" +
        "</form></body></html>");
```
Register Servlet

- Creates cookies based on request parameters received.
- Displays values if all parameters are present.
- Redirects to form if any parameter is missing.

```java
protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out=response.getWriter();
    String FName=request.getParameter("FName");
    String LName=request.getParameter("LName");
    String Email=request.getParameter("Email");
    Cookie c1=new Cookie("FName", FName);
    c1.setMaxAge(24*60*60);
    response.addCookie(c1);
    Cookie c2=new Cookie("LName", LName);
    c2.setMaxAge(24*60*60);
    response.addCookie(c2);
    Cookie c3=new Cookie("Email", Email);
    c3.setMaxAge(24*60*60);
    response.addCookie(c3);
    if(FName==""||LName==""||Email==""){
        response.sendRedirect("/Cookies/FormServlet");
    }
    else{
        out.print("<html><body><h1>Thank You For Registration</h1><br><br>
        "<b>F-Name: </b>"+FName+"</br>"
        "<b>L-Name: </b>"+LName+"</br>"
        "<b>Email: </b>"+Email+"</body></html>");
    }
}
```

Running

![Registration Form Image]
Please Register

FName: Haneen
LName: Ibrahim
E-mail: 
OK

Please Register

Missing: Email

FName: Haneen
LName: Ibrahim
E-mail: 
OK

Please Register

Missing: Email

FName: Hanoon
LName: Ibrahim
E-mail: hmasry@jugaza.edu.ps
OK

Thank You For Registration

F-Name: Haneen
L-Name: Ibrahim
Email: hmasry@jugaza.edu.ps
Exercise:

Use the cookies to differentiate first-time visitor from repeat visitors, such that you will display the following page to the first time user:

Welcome First Time Visitor

username: Haneen
password: ...
OK

But, if the user re-visit the page after he entered the username and the password, you asked to display the following page:

Welcome back: Haneen