Cookies

Cookies: what are they

A Cookie is a small amount of information sent by a servlet to a Web browser, saved by the browser, and later sent back to the server.

A cookie's value can uniquely identify a client, so cookies are commonly used for session management.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

Some Web browsers have bugs in how they handle the optional attributes, so use them sparingly to improve the interoperability of your servlets.

Cookies

Cookies affect the caching of the Web pages that use them. HTTP 1.0 does not cache pages that use cookies created with this class.

The Java class "Cookie" does not support the cache control defined with HTTP 1.1. This class supports both the Version 0 (by Netscape) and Version 1 (by RFC 2109) cookie specifications. By default, cookies are created using Version 0 to ensure the best interoperability

Cookies: why?

To maintain status across a "user session"

To maintan infos across sessions

- Customer identification
- Targeted advertisement
- Elimination of username e password

Attribute summary

String getComment() / void setComment(String s)

Gets/sets a comment associated with this cookie.

String getDomain() / setDomain(String s)

Gets/sets the domain to which cookie applies. Normally, cookies are returned only to the exact hostname that sent them. You can use this method to instruct the browser to return them to other hosts within the same domain. Note that the domain should start with a dot (e.g. .prenhall.com), and must contain two dots for non-country domains like .com, .edu, and .gov, and three dots for country domains like .co.uk and .edu.es.

Attribute summary

int getMaxAge() / void setMaxAge(int i)

Gets/sets how much time (in seconds) should elapse before the cookie expires. If you don't set this, the cookie will last only for the current session (i.e. until the user quits the browser), and will not be stored on disk. See the LongLivedCookie class below, which defines a subclass of Cookie with a maximum age automatically set one year in the future.

String getName() / void setName(String s)

Gets/sets the name of the cookie. The name and the value are the two pieces you virtually always care about. Since the getCookies method of HttpServletRequest returns an array of Cookie objects, it is common to loop down this array until you have a particular name, then check the value with getValue. See the getCookieValue method shown below.

Attribute summary

String getPath() / void setPath(String s)

Gets/sets the path to which this cookie applies. If you don't specify a path, the cookie is returned for all URLs in the same directory as the current page as well as all subdirectories. This method can be used to specify something more general. For example, someCookie.setPath("/") specifies that all pages on the server should receive the cookie. Note that the path specified must include the current directory.

boolean getSecure / setSecure(boolean b)

Gets/sets the boolean value indicating whether the cookie should only be sent over encrypted (i.e. SSL) connections.

Attribute summary

String getValue() / void setValue(String s)

Gets/sets the value associated with the cookie. Again, the name and the value are the two parts of a cookie that you almost always care about, although in a few cases a name is used as a boolean flag, and its value is ignored (i.e the existence of the name means true).

int getVersion() / void setVersion(int i)

Gets/sets the cookie protocol version this cookie complies with. Version 0, the default, adheres to the original Netscape specification. Version 1, not yet widely supported, adheres to RFC 2109.

Placing Cookies in the Response Headers

The cookie is added to the Set-Cookie response header by means of the addCookie method of HttpServletResponse. Here's an example:

```
Cookie userCookie = new Cookie("user", "uid1234");
response.addCookie(userCookie);
```

Reading Cookies from the Client

To read the cookies that come back from the client, you call getCookies on the HttpServletRequest. This returns an array of Cookie objects corresponding to the values that came in on the Cookie HTTP request header.

Once you have this array, you typically loop down it, calling getName on each Cookie until you find one matching the name you have in mind. You then call getValue on the matching Cookie, doing some processing specific to the resultant value. This is such a common process that the following section presents a simple getCookieValue method that, given the array of cookies, a name, and a default value, returns the value of the cookie matching the name, or, if there is no such cookie, the designated default value.

Cookies: examples

```
Cookie userCookie = new Cookie("user","uid1234");
userCookie.setMaxAge(60*60*24*365);
response.addCookie(userCookie);
```

Code to check if the client accepts cookies:

See http://www.purpletech.com/code/src/com/purpletech/servlets/CookieDetector.java

SetCookies

```
import java.io.*;   import javax.servlet.*; import javax.servlet.http.*;
/** Sets six cookies: three that apply only to the current session
 * (regardless of how long that session lasts) and three that persist for an hour
 * (regardless of whether the browser is restarted).
*/
public class SetCookies extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
    response)
                        throws ServletException, IOException {
   for(int i=0; i<3; i++) {
    // Default maxAge is -1, indicating cookie
    // applies only to current browsing session.
    Cookie cookie = new Cookie("Session-Cookie-" + i,
                               "Cookie-Value-S" + i);
    response.addCookie(cookie);
```

SetCookies

```
cookie = new Cookie("Persistent-Cookie-" + i,"Cookie-Value-P" + i);
      // Cookie is valid for an hour, regardless of whether
      // user quits browser, reboots computer, or whatever.
      cookie.setMaxAge(3600);
      response.addCookie(cookie);
    }
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Setting Cookies";
    out.println (("<HTML><HEAD><TITLE>" +title+ "</TITLE></HEAD>" +
       "<BODY BGCOLOR=\"#FDF5E6\">\n" +"<H1 ALIGN=\"CENTER\">"
     + title + "</H1>\n" +"There are six cookies associated with this page.\n" +
"</BODY></HTML>");
  }
}
```

ShowCookies

```
import java.io.*;  import javax.servlet.*;   import javax.servlet.http.*;
/** Creates a table of the cookies associated with the current page. */
public class ShowCookies extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
     response)
                      throws ServletException, IOException {
    response.setContentType("text/html");
    PrintWriter out = response.getWriter();
    String title = "Active Cookies";
    out.println(("<HTML><HEAD><TITLE>" +title+ "</TITLE></HEAD>" +
            "<BODY BGCOLOR=\"#FDF5E6\">\n" +
            "<H1 ALIGN=\"CENTER\">" + title + "</H1>\n" +
            "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
            "<TR BGCOLOR=\"#FFAD00\">\n" +
            "  <TH>Cookie Name\n" + "  <TH>Cookie Value");
```

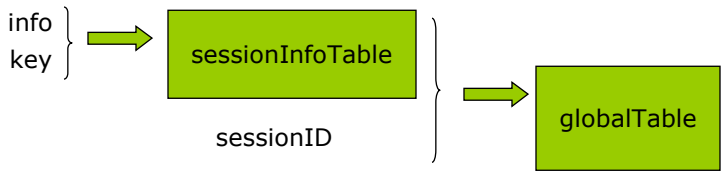
ShowCookies

```
Cookie[] cookies = request.getCookies();
   Cookie cookie;
   for(int i=0; i<cookies.length; i++) {
     cookie = cookies[i];
     out.println("<TR>\n" +
                 "  <TD>" + cookie.getName() + "\n" +
                 "  <TD>" + cookie.getValue());
   }
   out.println("</TABLE></BODY></HTML>");
  }
}
```

Sessions

Session tracking using cookies

```
String sessionID = makeUniqueString();
Hashtable sessionInfoTable = new Hashtable();
Hashtable globalTable = getTableStoringSession();
globalTable.put(sessionID, sessionInfoTable );
Cookie sessionCookie=new Cookie("SessionID",sessionID);
sessionCookie.setPath("/");
response.addCookie(sessionCookie);
```



HttpSession Class

Provides a way to identify a user across more than one page request or visit to a Web site and to store information about that user.

The servlet container uses this interface to create a session between an HTTP client and an HTTP server. The session persists for a specified time period, across more than one connection or page request from the user.

A session usually corresponds to one user, who may visit a site many times. The server can maintain a session in many ways such as using cookies or rewriting URLs.

HttpSession Class

This interface allows servlets to View and manipulate information about a session, such as the session identifier, creation time, and last accessed time Bind objects to sessions, allowing user information to persist across multiple user connections.

When an application stores an object in or removes an object from a session, the session checks whether the object implements HttpSessionBindingListener. If it does, the servlet notifies the object that it has been bound to or unbound from the session.

Session tracking API

```
HttpSession session = request.getSession(true);
ShoppingCart cart = (ShoppingCart)session.getValue("carrello");
   // 2.1
// 2.2   (ShoppingCart)session.getAttribute("carrello");
if (cart==null) {
   cart=new ShoppingCart();
   session.putValue("carrello",cart); //2.1
//2.2 session.putValue("carrello",cart);
}
doSomeThingWith(cart);
```

Session tracking API

```
public void putValue(String name, Object value);        //2.1
public void setAttribute(String name, Object value);    //2.2

public void removeValue(String name);  //2.1
public void removeAttribute(String name);        //2.2

public String[] getValueNames()            //2.1
public Enumeration getAttributeNames() //2.2
```

Session tracking API

```
public long getCreationTime();
public long getLastAccessdTime();
   milliseconds since midnight,  1.1.1970

public int getMaxInactiveInterval();
public void setMaxInactiveInterval(int sec);

public void invalidate();
```

ShowSession

```
import java.io.*; import javax.servlet.*; import javax.servlet.http.*;
import java.net.*; import java.util.*;
/** Simple example of session tracking. */
public class ShowSession extends HttpServlet {
  public void doGet(HttpServletRequest request, HttpServletResponse
    response)
                    throws ServletException, IOException {
   response.setContentType("text/html");
   PrintWriter out = response.getWriter();
   String title = "Session Tracking Example";
   HttpSession session = request.getSession(true);
   String heading;
   // Use getAttribute instead of getValue in version 2.2.
   Integer accessCount = (Integer)session.getValue("accessCount");
```

ShowSession

```
if (accessCount == null) {
     accessCount = new Integer(0);
     heading = "Welcome Newcomer";
   } else {
     heading = "Welcome Back";
     accessCount = new Integer(accessCount.intValue() + 1);
   }
   // Use setAttribute instead of putValue in version 2.2.
   session.putValue("accessCount", accessCount);
```

ShowSession

```
out.println(("<HTML><HEAD><TITLE>" +title+ "</TITLE></HEAD>" +
            "<BODY BGCOLOR=\"#FDF5E6\">\n" +
            "<H1 ALIGN=\"CENTER\">" + heading + "</H1>\n" +
            "<H2>Information on Your Session:</H2>\n" +
            "<TABLE BORDER=1 ALIGN=\"CENTER\">\n" +
            "<TR BGCOLOR=\"#FFAD00\">\n" +
            "  <TH>Info Type<TH>Value\n" +
            "<TR>\n" +"  <TD>ID\n" +"  <TD>" + session.getId() + "\n" +
            "<TR>\n" +"  <TD>Creation Time\n" +
            "  <TD>" + new Date(session.getCreationTime()) + "\n" +
            "<TR>\n" +"  <TD>Time of Last Access\n" +
            "  <TD>" +new Date(session.getLastAccessedTime()) + "\n" +
            "<TR>\n" +"  <TD>Number of Previous Accesses\n" +"  <TD>" +
            accessCount + "\n" + "</TABLE>\n" +"</BODY></HTML>");
}
```

ShowSession

```
/** Handle GET and POST requests identically. */

public void doPost(HttpServletRequest request,
                   HttpServletResponse response)
  throws ServletException, IOException {
  doGet(request, response);
 }
}
```