Cookies

Cookies are small bits of textual information that a Web server sends to a browser and that the browser returns unchanged when later visiting the same Web site or domain. By letting the server read information it sent the client previously, the site can provide visitors with a number of conveniences such as presenting the site the way the visitor previously customized it or letting identifiable visitors in without their having to enter a password. Most browsers avoid caching documents associated with cookies, so the site can return different content each time.

Benefits of Cookies

1) ***Identifying a User During an E-commerce Session***
2) ***Avoiding Username and Password***
3) ***Customizing a Site***
4) ***Focusing Advertising***

Some Problems with Cookies

1) Some people don't like the fact that search engines can remember that they're the user who usually does searches on certain topics.

2) Second privacy problem occurs when sites rely on cookies for overly sensitive data. For example, some of the big on-line bookstores use cookies to remember users and let you order without reentering much of your personal information.

Sending & Receiving Cookies

Creating Cookies:

You create a cookie by calling the Cookie constructor, which takes two strings: the cookie name and the cookie value.

Cookie c=new Cookie(“userID”,”a1234”);

Setting age:

```
c.setMaxage(60*60*24*7);     // one week
```

Place cookie in response header

```
response.addCookie(userID);
```

Sending & Receiving Cookies

Reading Cookies:

1) Call request.getCookies // this yields array of Cookie objects
2) Loop down the array calling getname.

```
String cookiename  = “userID”;
Cookie[] c1=request.getCookies();
If (c1 !=null) {
For (int i=0;i<c1.length;i++) {
Cookie c2= c1[i];
}
}
```

Sending & Receiving Cookies

Creating Cookies:

You create a cookie by calling the Cookie constructor, which takes two strings: the cookie name and the cookie value.

Cookie c=new Cookie(“userID”,”a1234”);

Setting age:

```
c.setMaxage(60*60*24*7);     // one week
```

Place cookie in response header

```
response.addCookie(userID);
```

Cookies to Detect First Time User

A Cookie is the perfect way to differentiate first visitors from repeat visitors.

```
Cookie[ ] c= request.getCookies();
if (c==null)
{
first time visitor;
}
 else
{
repeat visitor;
}
```

Using Cookies Attributes

Cookies attributes are the characteristics which may need to set to cookies.

1) **public void setComment (String comment)**
2) **Public String getComment()**
3) **Public void setDomain (String domainpattern)**
4) **Public String getDomain()**
5) **Public void setMaxAge(int lifetime)**
6) **Public int getMaxAge()**
7) **Public String getName()**
8) **Public void setPath(String path)**
9) **Public String getPath()**
10) **Public void setSecure(boolean flag)**
11) **Public boolean getSecure()**
12) **Public void setValue(String val)**
13) **Public String getValue()**

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.

There are three typical solutions to this problem

1) **Cookies**
2) **URL rewriting**
3) **Hidden form fields**

Session Tracking Basics

Using sessions in servlet is straightforward and involves four basic steps.

1) **Accessing session object associated with current request**
2) **Looking up information associated with a session**
3) **Storing information in a session**
4) **Discarding session Data**

The Session Tracking API

Public Object getAttribute(String name)

Public Enumeration getAttributeNames()

Public void setAttribute(String name, Object Value)

Public void removeAttribute(String name)

Public void invalidate()

Public void logout()

Public String getId()

Public boolean isNew()

Public long getCreationTime()

Public long getLastAccessedTime()

Public int getMaxInactiveInterval()

Public void setMaxInactiveInterval(int seconds)

Browser Session Vs Server session

Once the client started the session, the server regularly get the information from the client browser. If due to any conditions, the server does not get the Session attributes values from browser or if the browser quits the communication, the browser session level cookies to be lost, the session is effectively broken.

Even though one exception to the server waits until sessions time out. So the server can immediately remove all the values from the session and destroy the session object.

The Need for JSP

Servlets are not good at presentation, because these have following deficiencies .

1) It is hard to write and maintain HTML

2) You cannot use standard HTML tools

3) The HTML is inaccessible to non-Java developers.

Benefits of JSP

JSP pages are translated into servlets. So any JSP page can perform the task accomplished by servlet.

JSP provides following benefits.

1) It is easier to write and maintain HTML.

2) You can use standard web-site development tool.

3) You can divide up your development team.

Installation of JSP

JSP Directories for Tomcat

Main Location

Install_dir/webApps/Root

Corresponding URL

http://host/somefile.jsp

More Specific Location

install_dir/webapps/ROOT/somedirectory

Corresponding URL

http:// host/SomeDirectory/SomeFile.jsp

Basic Syntax

HTML text	**<H1> Hello </H1>**
HTML Comments	**<! - - some text - - >**
Template text	**Text sent unchanged to client, anything other than syntax.**
JSP Comment	**<%-- text -- %>**
JSP Expression	**<%= Java Value %>**
JSP Scriptlet	**<% Java Statement %>**
JSP Declaration	**<%! Defination %>**
JSP Directive	**<%@ directive att="val" %>**
JSP Action	**<jsp:include>…</jsp:inculde>**
JSP Expr. Lang Ele	**${EL Expression}**
Escaped text	**<\% . . . %\>**