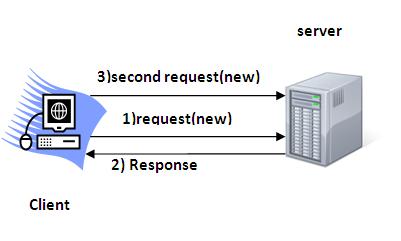
Session Tracking in Servlets

**Session** simply means a particular interval of time.

**Session Tracking** is a way to maintain state (data) of a user. It is also known as **session management** in servlet.

Http protocol is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.

HTTP is stateless that means each request is considered as the new request. It is shown in the figure given below:



**Why use Session Tracking?**

**To recognize the user** It is used to recognize the particular user.

Session Tracking Techniques

There are four techniques used in Session tracking:

1. **Cookies**
2. **Hidden Form Field**
3. **URL Rewriting**
4. **HttpSession**

# **Cookies in Servlet**

A **cookie** is a small piece of information that is persisted between the multiple client requests.

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.

### How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



### Types of Cookie

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

### Non-persistent cookie

It is **valid for single session** only. It is removed each time when user closes the browser.

### Persistent cookie

It is **valid for multiple sessions**. It is not removed each time when user closes the browser. It is removed only if user logout or sign-out.

### Advantage of Cookies

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

### Disadvantage of Cookies

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

Cookie class

**javax.servlet.http.Cookie** class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

**Constructor of Cookie class**

|  |  |
| --- | --- |
| **Constructor** | **Description** |
| Cookie() | constructs a cookie. |
| Cookie(String name, String value) | constructs a cookie with a specified name and value. |

**Useful Methods of Cookie class**

There are given some commonly used methods of the Cookie class.

|  |  |
| --- | --- |
| **Method** | **Description** |
| public void setMaxAge(int expiry) | Sets the maximum age of the cookie in seconds. |
| public String getName() | Returns the name of the cookie. The name cannot be changed after creation. |
| public String getValue() | Returns the value of the cookie. |
| public void setName(String name) | changes the name of the cookie. |
| public void setValue(String value) | changes the value of the cookie. |

Other methods required for using Cookies

|  |
| --- |
| For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:   1. **public void addCookie(Cookie ck):**method of HttpServletResponse interface is used to add cookie in response object. 2. **public Cookie[] getCookies():**method of HttpServletRequest interface is used to return all the cookies from the browser. |

How to create Cookie?

Let's see the simple code to create cookie.

1. Cookie ck=**new** Cookie("user","sonoo jaiswal");//creating cookie object
2. response.addCookie(ck);//adding cookie in the response

How to delete Cookie?

Let's see the simple code to delete cookie. It is mainly used to logout or signout the user.

1. Cookie ck=**new** Cookie("user","");//deleting value of cookie
2. ck.setMaxAge(0);//changing the maximum age to 0 seconds
3. response.addCookie(ck);//adding cookie in the response

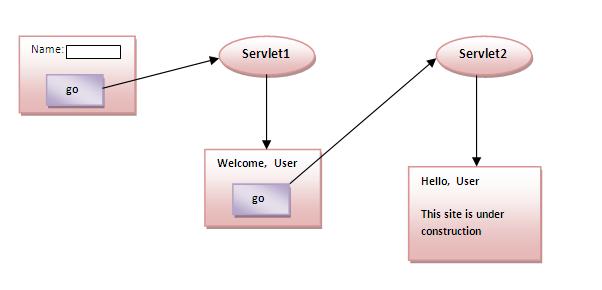
How to get Cookies?

Let's see the simple code to get all the cookies.

1. Cookie ck[]=request.getCookies();
2. **for**(**int** i=0;i<ck.length;i++){
3. out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie
4. }

Simple example of Servlet Cookies

In this example, we are storing the name of the user in the cookie object and accessing it in another servlet. As we know well that session corresponds to the particular user. So if you access it from too many browsers with different values, you will get the different value.



**index.html**

1. <form action="servlet1" method="post">
2. Name:<input type="text" name="userName"/><br/>
3. <input type="submit" value="go"/>
4. </form>

**FirstServlet.java**

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
3. **import** javax.servlet.http.\*;
4. **public** **class** FirstServlet **extends** HttpServlet {
5. **public** **void** doPost(HttpServletRequest request, HttpServletResponse response){
6. **try**{
7. response.setContentType("text/html");
8. PrintWriter out = response.getWriter();
9. String n=request.getParameter("userName");
10. out.print("Welcome "+n);
11. Cookie ck=**new** Cookie("uname",n);//creating cookie object
12. response.addCookie(ck);//adding cookie in the response
13. //creating submit button
14. out.print("<form action='servlet2'>");
15. out.print("<input type='submit' value='go'>");
16. out.print("</form>");
17. out.close();
18. }**catch**(Exception e){System.out.println(e);} } }

**SecondServlet.java**

1. **import** java.io.\*;
2. **import** javax.servlet.\*;
3. **import** javax.servlet.http.\*;
5. **public** **class** SecondServlet **extends** HttpServlet {
7. **public** **void** doPost(HttpServletRequest request, HttpServletResponse response){
8. **try**{
9. response.setContentType("text/html");
10. PrintWriter out = response.getWriter();
11. Cookie ck[]=request.getCookies();
12. out.print("Hello "+ck[0].getValue());
13. out.close();
14. }**catch**(Exception e){System.out.println(e);}
15. }
16. }

**web.xml**

1. <web-app>
2. <servlet>
3. <servlet-name>s1</servlet-name>
4. <servlet-**class**>FirstServlet</servlet-**class**>
5. </servlet>
6. <servlet-mapping>
7. <servlet-name>s1</servlet-name>
8. <url-pattern>/servlet1</url-pattern>
9. </servlet-mapping>
10. <servlet>
11. <servlet-name>s2</servlet-name>
12. <servlet-**class**>SecondServlet</servlet-**class**>
13. </servlet>
14. <servlet-mapping>
15. <servlet-name>s2</servlet-name>
16. <url-pattern>/servlet2</url-pattern>
17. </servlet-mapping>
18. </web-app>