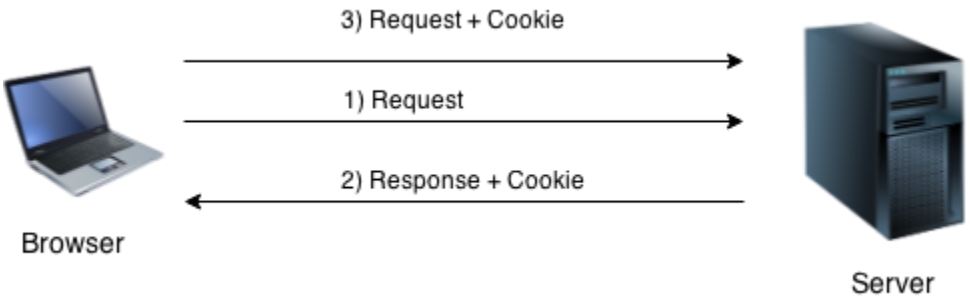
Cookies in Servlet

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 session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

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.

Note: Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.

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
```

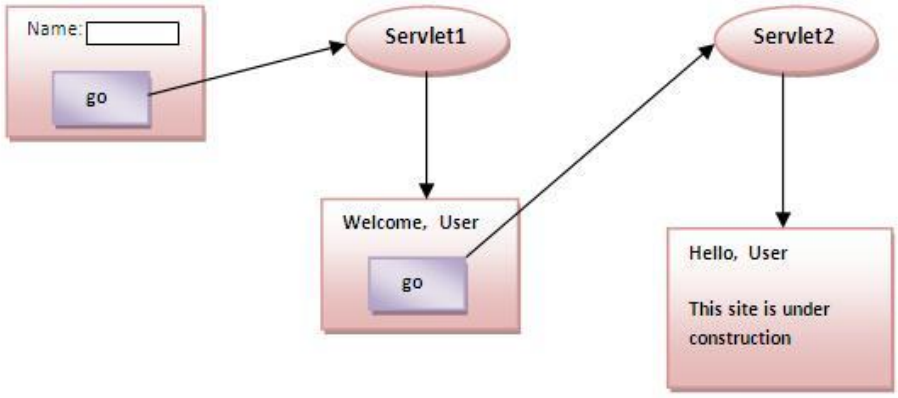
How to get Cookies?

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 coo
   kie
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.
5.
6. public class FirstServlet extends HttpServlet {
7.
8.   public void doPost(HttpServletRequest request, HttpServletResponse response){
9.     try{
10.
11.    response.setContentType("text/html");
12.    PrintWriter out = response.getWriter();
13.
14.    String n=request.getParameter("userName");
15.    out.print("Welcome "+n);
```

```
16.
17.    Cookie ck=new Cookie("uname",n);//creating cookie object
18.    response.addCookie(ck);//adding cookie in the response
19.
20.    //creating submit button
21.    out.print("<form action='servlet2'>");
22.    out.print("<input type='submit' value='go'>");
23.    out.print("</form>");
24.
25.    out.close();
26.
27.        }catch(Exception e){System.out.println(e);}
28.   }
29. }
```

SecondServlet.java

```
1.  import java.io.*;
2.  import javax.servlet.*;
3.  import javax.servlet.http.*;
4.
5.  public class SecondServlet extends HttpServlet {
6.
7.  public void doPost(HttpServletRequest request, HttpServletResponse response){
8.     try{
9.
10.    response.setContentType("text/html");
11.    PrintWriter out = response.getWriter();
12.
13.    Cookie ck[]=request.getCookies();
14.    out.print("Hello "+ck[0].getValue());
15.
16.    out.close();
17.
18.        }catch(Exception e){System.out.println(e);}
19.    }
20.
21.
22. }
```

web.xml

```
1.  <web-app>
2.
3.  <servlet>
4.  <servlet-name>s1</servlet-name>
```

```
<servlet-class>FirstServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>s1</servlet-name>
<url-pattern>/servlet1</url-pattern>
</servlet-mapping>

<servlet>
<servlet-name>s2</servlet-name>
<servlet-class>SecondServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>s2</servlet-name>
<url-pattern>/servlet2</url-pattern>
</servlet-mapping>

</web-app>
```

EXAMPLE:

servlet login and logout example using cookies (Real Login App)

Servlet Login and Logout Example using Cookies

A **cookie** is a kind of information that is stored at client side.

In the previous page, we learned a lot about cookie e.g. how to create cookie, how to delete cookie, how to get cookie etc.

Here, we are going to create a login and logout example using servlet cookies.

In this example, we are creating 3 links: login, logout and profile. User can't go to profile page until he/she is logged in. If user is logged out, he need to login again to visit profile.

In this application, we have created following files.

1. index.html

2. link.html
3. login.html
4. LoginServlet.java
5. LogoutServlet.java
6. ProfileServlet.java
7. web.xml

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Servlet Login Example</title>
</head>
<body>

<h1>Welcome to Login App by Cookie</h1>
<a href="login.html">Login</a>|
<a href="LogoutServlet">Logout</a>|
<a href="ProfileServlet">Profile</a>

</body>
</html>
```

File: link.html

```
<a href="login.html">Login</a> |
<a href="LogoutServlet">Logout</a> |
<a href="ProfileServlet">Profile</a>
<hr>
```

File: login.html

```
<form action="LoginServlet" method="post">
Name:<input type="text" name="name"><br>
Password:<input type="password" name="password"><br>
<input type="submit" value="login">
</form>
```

File: LoginServlet.java

```
package com.javatpoint;

import java.io.IOException;
import java.io.PrintWriter;
```

```
5. import javax.servlet.ServletException;
6. import javax.servlet.http.Cookie;
7. import javax.servlet.http.HttpServlet;
8. import javax.servlet.http.HttpServletRequest;
9. import javax.servlet.http.HttpServletResponse;
10. public class LoginServlet extends HttpServlet {
11.     protected void doPost(HttpServletRequest request, HttpServletResponse response)
12.                         throws ServletException, IOException {
13.         response.setContentType("text/html");
14.         PrintWriter out=response.getWriter();
15.
16.         request.getRequestDispatcher("link.html").include(request, response);
17.
18.         String name=request.getParameter("name");
19.         String password=request.getParameter("password");
20.
21.         if(password.equals("admin123")){
22.             out.print("You are successfully logged in!");
23.             out.print("<br>Welcome, "+name);
24.
25.             Cookie ck=new Cookie("name",name);
26.             response.addCookie(ck);
27.         }else{
28.             out.print("sorry, username or password error!");
29.             request.getRequestDispatcher("login.html").include(request, response);
30.         }
31.
32.         out.close();
33.     }
34.
35. }
```

File: LogoutServlet.java

```
1. package com.javatpoint;
2.
3. import java.io.IOException;
4. import java.io.PrintWriter;
5. import javax.servlet.ServletException;
6. import javax.servlet.http.Cookie;
7. import javax.servlet.http.HttpServlet;
8. import javax.servlet.http.HttpServletRequest;
9. import javax.servlet.http.HttpServletResponse;
10. public class LogoutServlet extends HttpServlet {
11.     protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
                throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();


        request.getRequestDispatcher("link.html").include(request, response);

        Cookie ck=new Cookie("name","");
        ck.setMaxAge(0);
        response.addCookie(ck);

        out.print("you are successfully logged out!");
    }
}
```

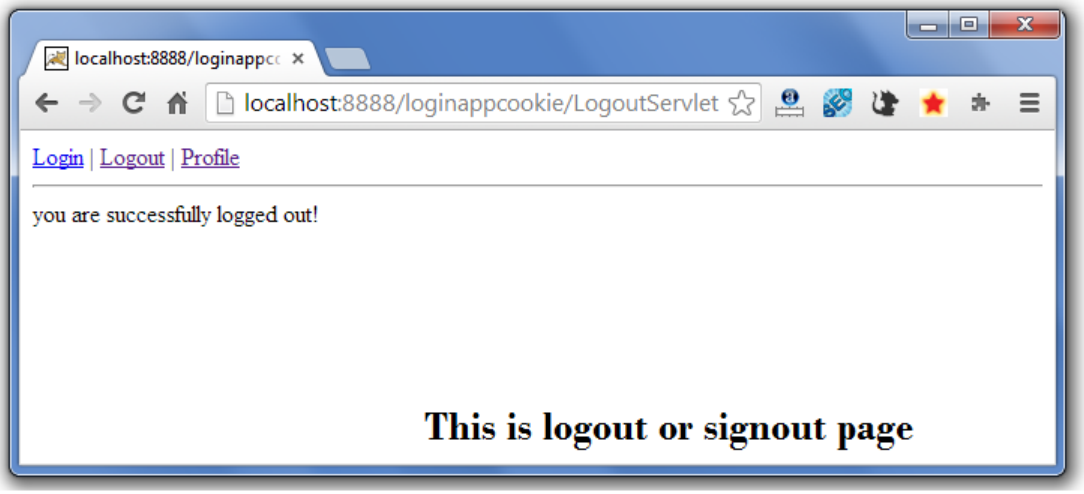
File: ProfileServlet.java

```
package com.javatpoint;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class ProfileServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                    throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();

        request.getRequestDispatcher("link.html").include(request, response);

        Cookie ck[]=request.getCookies();
        if(ck!=null){
         String name=ck[0].getValue();
        if(!name.equals("")||name!=null){
            out.print("<b>Welcome to Profile</b>");
            out.print("<br>Welcome, "+name);
        }
        }else{
            out.print("Please login first");
            request.getRequestDispatcher("login.html").include(request, response);
        }
```

```
        out.close();
    }
}
```

File: web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">

  <servlet>
    <description></description>
    <display-name>LoginServlet</display-name>
    <servlet-name>LoginServlet</servlet-name>
    <servlet-class>com.javatpoint.LoginServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LoginServlet</servlet-name>
    <url-pattern>/LoginServlet</url-pattern>
  </servlet-mapping>
  <servlet>
    <description></description>
    <display-name>ProfileServlet</display-name>
    <servlet-name>ProfileServlet</servlet-name>
    <servlet-class>com.javatpoint.ProfileServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>ProfileServlet</servlet-name>
    <url-pattern>/ProfileServlet</url-pattern>
  </servlet-mapping>
  <servlet>
    <description></description>
    <display-name>LogoutServlet</display-name>
    <servlet-name>LogoutServlet</servlet-name>
    <servlet-class>com.javatpoint.LogoutServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LogoutServlet</servlet-name>
    <url-pattern>/LogoutServlet</url-pattern>
  </servlet-mapping>
</web-app>
```



2) Hidden Form Field

1. Hidden Form Field
2. Example of Hidden Form Field

In case of Hidden Form Field **a hidden (invisible) textfield** is used for maintaining the state of an user.

In such case, we store the information in the hidden field and get it from another servlet. This approach is better if we have to submit form in all the pages and we don't want to depend on the browser.

Let's see the code to store value in hidden field.

```
1. <input type="hidden" name="uname" value="Vimal Jaiswal">
```

Here, uname is the hidden field name and Vimal Jaiswal is the hidden field value.

Real application of hidden form field

It is widely used in comment form of a website. In such case, we store page id or page name in the hidden field so that each page can be uniquely identified.

Advantage of Hidden Form Field

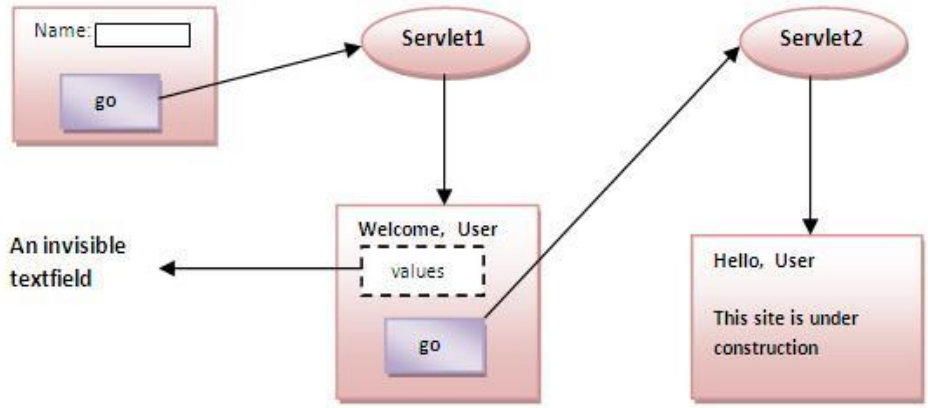
1. It will always work whether cookie is disabled or not.

Disadvantage of Hidden Form Field:

1. It is maintained at server side.
2. Extra form submission is required on each pages.
3. Only textual information can be used.

Example of using Hidden Form Field

In this example, we are storing the name of the user in a hidden textfield and getting that value from another servlet.



index.html

```
1. <form action="servlet1">
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.
5. public class FirstServlet extends HttpServlet {
6. public void doGet(HttpServletRequest request, HttpServletResponse response){
7.         try{
8.
9.         response.setContentType("text/html");
```

```
        PrintWriter out = response.getWriter();

        String n=request.getParameter("userName");
        out.print("Welcome "+n);

        //creating form that have invisible textfield
        out.print("<form action='servlet2'>");
        out.print("<input type='hidden' name='uname' value='"+n+"'>");
        out.print("<input type='submit' value='go'>");
        out.print("</form>");
        out.close();

                }catch(Exception e){System.out.println(e);}
    }

}
```

SecondServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class SecondServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response)
        try{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        //Getting the value from the hidden field
        String n=request.getParameter("uname");
        out.print("Hello "+n);

        out.close();
                }catch(Exception e){System.out.println(e);}
    }
}
```

web.xml

```
<web-app>

<servlet>
<servlet-name>s1</servlet-name>
<servlet-class>FirstServlet</servlet-class>
</servlet>

```

```
<servlet-mapping>
<servlet-name>s1</servlet-name>
<url-pattern>/servlet1</url-pattern>
</servlet-mapping>

<servlet>
<servlet-name>s2</servlet-name>
<servlet-class>SecondServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>s2</servlet-name>
<url-pattern>/servlet2</url-pattern>
</servlet-mapping>

</web-app>
```

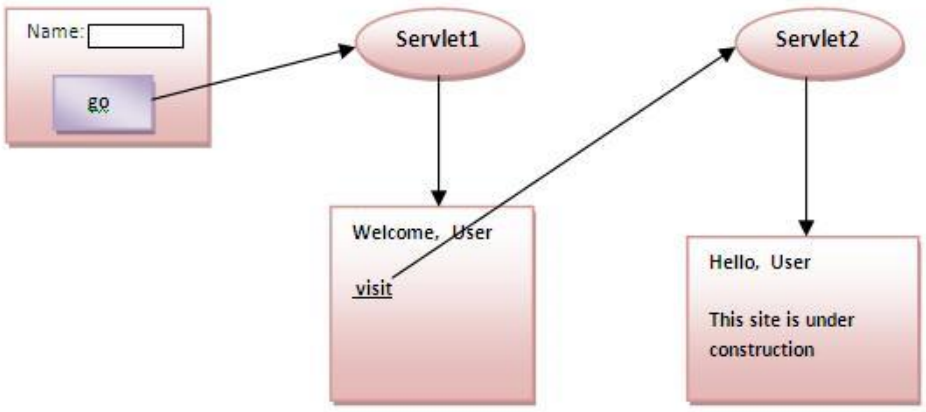
---

3)URL Rewriting

In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/value pairs using the following format:

url?name1=value1&name2=value2&??

A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&). When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter() method to obtain a parameter value.



Advantage of URL Rewriting

1. It will always work whether cookie is disabled or not (browser independent).
2. Extra form submission is not required on each pages.

Disadvantage of URL Rewriting

1. It will work only with links.
2. It can send Only textual information.

Example of using URL Rewriting

In this example, we are maintaning the state of the user using link. For this purpose, we are appending the name of the user in the query string and getting the value from the query string in another page.

index.html

```
<form action="servlet1">
Name:<input type="text" name="userName"/><br/>
<input type="submit" value="go"/>
</form>
```

FirstServlet.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;


public class FirstServlet extends HttpServlet {
```

```
7.
8.  public void doGet(HttpServletRequest request, HttpServletResponse response){
9.          try{
10.
11.         response.setContentType("text/html");
12.         PrintWriter out = response.getWriter();
13.
14.         String n=request.getParameter("userName");
15.         out.print("Welcome "+n);
16.
17.         //appending the username in the query string
18.         out.print("<a href='servlet2?uname="+n+"'>visit</a>");
19.
20.         out.close();
21.
22.                 }catch(Exception e){System.out.println(e);}
23.     }
24.
25. }
```

SecondServlet.java

```
1.  import java.io.*;
2.  import javax.servlet.*;
3.  import javax.servlet.http.*;
4.
5.  public class SecondServlet extends HttpServlet {
6.
7.  public void doGet(HttpServletRequest request, HttpServletResponse response)
8.          try{
9.
10.         response.setContentType("text/html");
11.         PrintWriter out = response.getWriter();
12.
13.         //getting value from the query string
14.         String n=request.getParameter("uname");
15.         out.print("Hello "+n);
16.
17.         out.close();
18.
19.                 }catch(Exception e){System.out.println(e);}
20.     }
21.
22.
23. }
```

web.xml

```
<web-app>

<servlet>
<servlet-name>s1</servlet-name>
<servlet-class>FirstServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>s1</servlet-name>
<url-pattern>/servlet1</url-pattern>
</servlet-mapping>

<servlet>
<servlet-name>s2</servlet-name>
<servlet-class>SecondServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>s2</servlet-name>
<url-pattern>/servlet2</url-pattern>
</servlet-mapping>

</web-app>
```