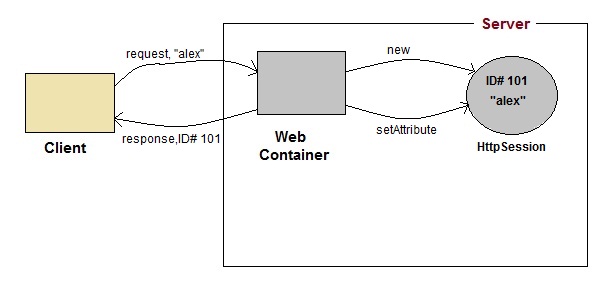
**What is HttpSession?**

**HttpSession** object is used to store entire session with a specific client. We can store, retrieve and remove attribute from **HttpSession** object. Any servlet can have access to **HttpSession** object throughout the getSession() method of the **HttpServletRequest** object.

**How HttpSession works**



1. On client's first request, the **Web Container** generates a unique session ID and gives it back to the client with response. This is a temporary session created by web container.
2. The client sends back the session ID with each request. Making it easier for the web container to identify where the request is coming from.
3. The **Web Container** uses this ID, finds the matching session with the ID and associates the session with the request.

**HttpSession Interface**



**Some Important Methods of HttpSession**

|  |  |
| --- | --- |
| **Methods** | **Description** |
| long getCreationTime() | returns the time when the session was created, measured in milliseconds since midnight January 1, 1970 GMT. |
| String getId() | returns a string containing the unique identifier assigned to the session. |
| long getLastAccessedTime() | returns the last time the client sent a request associated with the session |
| intgetMaxInactiveInterval() | returns the maximum time interval, in seconds. |
| void invalidate() | destroy the session |
| booleanisNew() | returns true if the session is new else false |
| void setMaxInactiveInterval(int interval) | Specifies the time, in seconds,after servlet container will invalidate the session. |

**Complete Example demonstrating usage of HttpSession**

All the files mentioned below are required for the example.

**index.html**

<form method="post" action="**Validate**">

User: <input type="text" name="user" /><br/>

Password: <input type="text" name="pass" ><br/>

<input type="submit" value="submit">

</form>

**web.xml**

<web-app..>

<servlet>

<servlet-name>**Validate**</servlet-name>

<servlet-class>*Validate*</servlet-class>

</servlet>

<servlet>

<servlet-name>**Welcome**</servlet-name>

<servlet-class>*Welcome*</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>Validate</servlet-name>

<url-pattern>/Validate</url-pattern>

</servlet-mapping>

<servlet-mapping>

<servlet-name>Welcome</servlet-name>

<url-pattern>/Welcome</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>**index.html**</welcome-file>

</welcome-file-list>

</web-app>

**Validate.java**

import java.io.\*;

importjavax.servlet.\*;

importjavax.servlet.http.\*;

public class *Validate***extends**HttpServlet {

protected void **doPost**(HttpServletRequest request, HttpServletResponse response)

throwsServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

String **name** = request.getParameter("user");

String **pass** = request.getParameter("pass");

if(pass.*equals*("1234"))

{

//creating a session

HttpSession**session** = request.getSession();

session.**setAttribute**("*user*", *name*);

response.**sendRedirect**("Welcome");

}

}

}

**Welcome.java**

import java.io.\*;

importjavax.servlet.\*;

importjavax.servlet.http.\*;

public class *Welcome***extends**HttpServlet {

protected void **doGet**(HttpServletRequest request, HttpServletResponse response)

throwsServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

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

HttpSession**session** = request.getSession();

String **user** = (String)session.getAttribute("*user*");

out.println("Hello "+**user**);

}

}

**Using Cookies for Session Management**

**Cookies** are small pieces of information that are sent in response from the web server to the client. **Cookies** are the simplest technique used for storing client state.

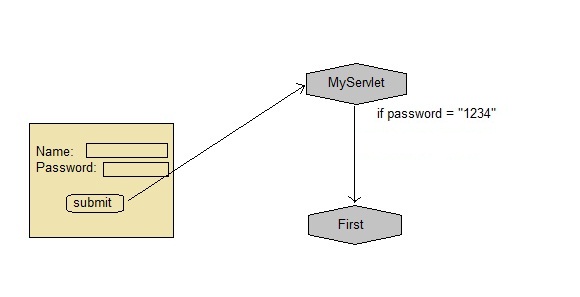
**Cookies** are stored on client's computer. They have a lifespan and are destroyed by the client browser at the end of that lifespan.

Using Cookies for storing client state has one shortcoming though, if the client has turned ofCOokie saving settings in his browser then, client state can never be saved because the browser will not allow the application to store cookies.

**Cookies API**

Cookies are created using **Cookie** class present in Servlet API. Cookies are added to **response** object using the addCookie() method. This method sends cookie information over the HTTP response stream. getCookies() method is used to access the cookies that are added to response object.

**Example demonstrating usage of Cookies**



Below mentioned files are required for the example:

**index.html**

<form method="post" action="**validate**">

Name:<input type="text" name="user" /><br/>

Password:<input type="text" name="pass" ><br/>

<input type="submit" value="submit">

</form>

**web.xml**

<web-app...>

<servlet>

<servlet-name>**validate**</servlet-name>

<servlet-class>**MyServlet**</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>**validate**</servlet-name>

<url-pattern>/validate</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>**First**</servlet-name>

<servlet-class>**First**</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>**First**</servlet-name>

<url-pattern>/First</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>**index.html**</welcome-file>

</welcome-file-list>

</web-app>

**MyServlet.java**

import java.io.\*;

importjavax.servlet.\*;

importjavax.servlet.http.\*;

public class *MyServlet***extends**HttpServlet {

protected void **doPost**(HttpServletRequest request, HttpServletResponse response)

throwsServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

String **name** = request.getParameter("*user*");

String **pass** = request.getParameter("*pass*");

if(pass.*equals*("1234"))

{

Cookie **ck** = new Cookie("*username*",*name*);

response.**addCookie**(*ck*);

response.sendRedirect("*First*");

}

}

}

**First.java**

import java.io.\*;

importjavax.servlet.\*;

importjavax.servlet.http.\*;

public class *First***extends**HttpServlet {

protected void **doGet**(HttpServletRequest request, HttpServletResponse response)

throwsServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

Cookie[] **cks** = request.getCookies();

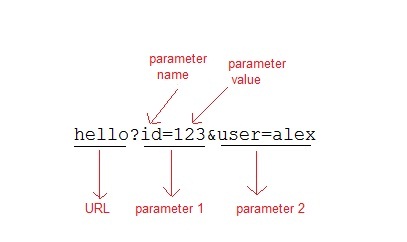
out.println("Welcome "+**cks[0].getValue()**);

}

}

In URL rewriting, a token(parameter) is added at the end of the URL. The token consist of name/value pair seperated by an equal(=) sign.

**For Example:**



When the User clicks on the URL having parameters, the request goes to the **Web Container** with extra bit of information at the end of URL. The **Web Container** will fetch the extra part of the requested URL and use it for session management.

The getParameter() method is used to get the parameter value at the server side.

**Example demonstrating usage of URL rewriting**

Below mentioned files are required for the example:

**index.html**

<form method="post" action="**validate**">

Name:<input type="text" name="user" /><br/>

Password:<input type="text" name="pass" ><br/>

<input type="submit" value="submit">

</form>

**web.xml**

<web-app...>

<servlet>

<servlet-name>**validate**</servlet-name>

<servlet-class>**MyServlet**</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>**validate**</servlet-name>

<url-pattern>/validate</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>**First**</servlet-name>

<servlet-class>**First**</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>**First**</servlet-name>

<url-pattern>/First</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>**index.html**</welcome-file>

</welcome-file-list>

</web-app>

**MyServlet.java**

import java.io.\*;

importjavax.servlet.\*;

importjavax.servlet.http.\*;

public class *MyServlet***extends**HttpServlet {

protected void **doPost**(HttpServletRequest request, HttpServletResponse response)

throwsServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

String **name** = request.getParameter("*user*");

String **pass** = request.getParameter("*pass*");

if(pass.*equals*("1234"))

{

response.sendRedirect(*"First?user\_name="+****name****+""*);

}

}

}

**First.java**

import java.io.\*;

importjavax.servlet.\*;

importjavax.servlet.http.\*;

public class *First***extends**HttpServlet {

protected void **doGet**(HttpServletRequest request, HttpServletResponse response)

throwsServletException, IOException {

response.setContentType("text/html;charset=UTF-8");

PrintWriter out = response.getWriter();

String **user** = request.getParameter("*user\_name*");

out.println("Welcome "+user);

}

}

# 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.\*;
5. **public** **class** FirstServlet **extends** HttpServlet {
6. **public** **void** doGet(HttpServletRequest request, HttpServletResponse response){
7. **try**{
9. response.setContentType("text/html");
10. PrintWriter out = response.getWriter();
12. String n=request.getParameter("userName");
13. out.print("Welcome "+n);
15. //creating form that have invisible textfield
16. out.print("<form action='servlet2'>");
17. out.print("<input type='hidden' name='uname' value='"+n+"'>");
18. out.print("<input type='submit' value='go'>");
19. out.print("</form>");
20. out.close();
22. }**catch**(Exception e){System.out.println(e);}
23. }
25. }

### SecondServlet.java

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

### web.xml

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