Session Management in Java - HttpServlet, Cookies, URL Rewriting

Session Management in Java

This article is aimed to explain about session management in servlets using different techniques and with example programs.

5. What is a Session?

HTTP protocol and Web Servers are stateless, what it means is that for web server every request is a new request to process and they can't identify if it's coming from client that has been sending request previously. But sometimes in web applications, we should know who the client is and process the request accordingly. For example, a shopping cart application should know who is sending the request to add an item and in which cart the item has to be added or who is sending checkout request so that it can charge the amount to correct client. **Session** is a conversional state between client and server and it can consists of multiple request and response between client and server. Since HTTP and Web Server both are stateless, the only way to maintain a session is when some unique information about the session (session id) is passed between server and client in every request and response. There are several ways through which we can provide unique identifier in request and response.

1. **User Authentication** - This is the very common way where we user can provide authentication credentials from the login page and then we can pass the authentication information between server and client to maintain the session. This is not very effective method because it wont work if the same user is logged in from different browsers.
2. **HTML Hidden Field** - We can create a unique hidden field in the HTML and when user starts navigating, we can set its value unique to the user

and keep track of the session. This method can’t be used with links because it needs the form to be submitted every time request is made from client to server with the hidden field. Also it’s not secure because we can get the hidden field value from the HTML source and use it to hack the session.

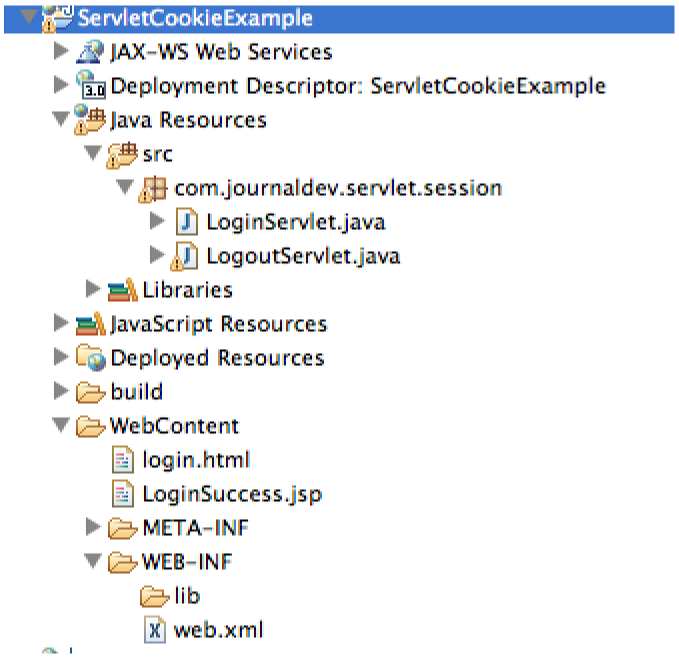
3. **URL Rewriting** - We can append a session identifier parameter with every request and response to keep track of the session. This is very tedious because we need to keep track of this parameter in every response and make sure it’s not clashing with other parameters.
4. **Cookies** - Cookies are small piece of information that is sent by web server in response header and gets stored in the browser cookies. When client make further request, it adds the cookie to the request header and we can utilize it to keep track of the session. We can maintain a session with cookies but if the client disables the cookies, then it won’t work.
5. **Session Management API** - Session Management API is built on top of above methods for session tracking. Some of the major disadvantages of all the above methods are:

    - Most of the time we don’t want to only track the session, we have to store some data into the session that we can use in future requests. This will require a lot of effort if we try to implement this.
    - All the above methods are not complete in themselves, all of them won’t work in a particular scenario. So we need a solution that can utilize these methods of session tracking to provide session management in all cases.

    That’s why we need **Session Management API** and J2EE Servlet technology comes with session management API that we can use.

6. Session Management in Java - Cookies

Cookies are used a lot in web applications to personalize response based on your choice or to keep track of session. Before moving forward to the Servlet Session Management API, I would like to show how can we keep track of session with cookies through a small web application. We will create a dynamic web application **ServletCookieExample** with project structure like below image.


```
ServletCookieExample
  JAX-WS Web Services
  Deployment Descriptor: ServletCookieExample
  Java Resources
    src
      com.journaldev.servlet.session
        LoginServlet.java
        LogoutServlet.java
    Libraries
  JavaScript Resources
  Deployed Resources
  build
  WebContent
    login.html
    LoginSuccess.jsp
    META-INF
    WEB-INF
      lib
      web.xml
```

Deployment descriptor web.xml of the web application is:

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="https://www.w3.org/2001/XMLSchema-instance"
xmlns="https://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="https://java.sun.com/xml/ns/javaee
https://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID"
version="3.0">

  <display-name>ServletCookieExample</display-name>

  <welcome-file-list>

    <welcome-file>login.html</welcome-file>

  </welcome-file-list>

</web-app>
```

Welcome page of our application is login.html where we will get authentication details from user.

```
<!DOCTYPE html>

<html>
```

```
<head>

<meta charset="US-ASCII">

<title>Login Page</title>

</head>

<body>



<form action="LoginServlet" method="post">



Username: <input type="text" name="user">

<br>

Password: <input type="password" name="pwd">

<br>

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

</form>

</body>
```

Session in Java Servlet - HttpSession

Servlet API provides Session management through `HttpSession` interface. We can get session from HttpServletRequest object using following methods. HttpSession allows us to set objects as attributes that can be retrieved in future requests.

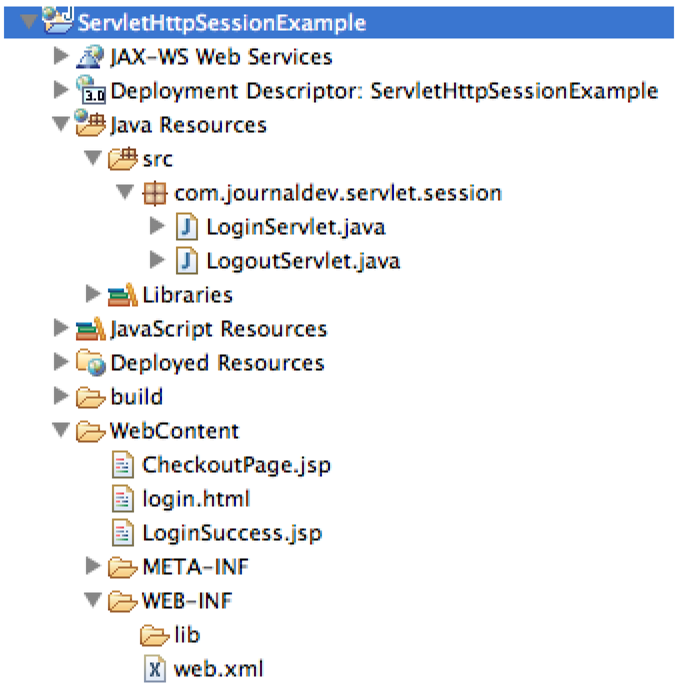
1. **HttpSession getSession()** - This method always returns a HttpSession object. It returns the session object attached with the request, if the request has no session attached, then it creates a new session and return it.
2. **HttpSession getSession(boolean flag)** - This method returns HttpSession object if request has session else it returns null.

Some of the important methods of HttpSession are:

1. **String getId()** - Returns a string containing the unique identifier assigned to this session.
2. **Object getAttribute(String name)** - Returns the object bound with the specified name in this session, or null if no object is bound under the name. Some other methods to work with Session attributes are `getAttributeNames()`, `removeAttribute(String name)` and `setAttribute(String name, Object value)`.
3. **long getCreationTime()** - Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT. We can get last accessed time with `getLastAccessedTime()` method.
4. setMaxInactiveInterval(int interval) - Specifies the time, in seconds, between client requests before the servlet container will invalidate this session. We can get session timeout value from `getMaxInactiveInterval()` method.
5. **ServletContext getServletContext()** - Returns ServletContext object for the application.
6. **boolean isNew()** - Returns true if the client does not yet know about the session or if the client chooses not to join the session.
7. **void invalidate()** - Invalidates this session then unbinds any objects bound to it.

Understanding JSESSIONID Cookie

When we use HttpServletRequest getSession() method and it creates a new request, it creates the new HttpSession object and also add a Cookie to the response object with name JSESSIONID and value as session id. This cookie is used to identify the HttpSession object in further requests from client. If the cookies are disabled at client side and we are using URL rewriting then this method uses the jsessionid value from the request URL to find the corresponding session. JSESSIONID cookie is used for session tracking, so we should not use it for our application purposes to avoid any session related issues. Let’s see example of session management using HttpSession object. We will create a dynamic web project in Eclipse with servlet context as ServletHttpSessionExample. The project structure will look like below image.

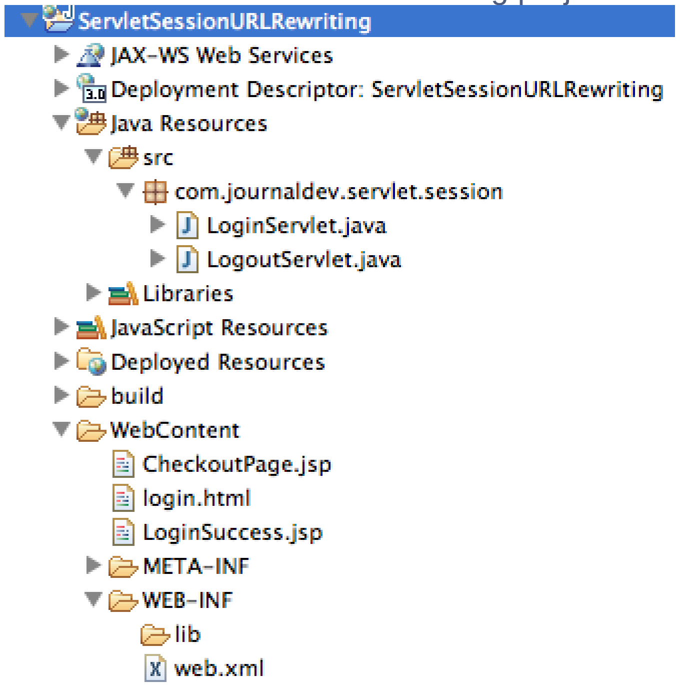


```
ServletHttpSessionExample
    JAX-WS Web Services
    Deployment Descriptor: ServletHttpSessionExample
    Java Resources
        src
            com.journaldev.servlet.session
                LoginServlet.java
                LogoutServlet.java
        Libraries
    JavaScript Resources
    Deployed Resources
    build
    WebContent
        CheckoutPage.jsp
        login.html
        LoginSuccess.jsp
        META-INF
        WEB-INF
            lib
            web.xml
```

login.html is same like earlier example and defined as welcome page for the application in web.xml LoginServlet servlet will create the session and set attributes that we can use in other resources or in future requests.

Session Management in Java Servlet - URL Rewriting

As we saw in last section that we can manage a session with HttpSession but if we disable the cookies in browser, it won’t work because server will not receive the JSESSIONID cookie from client. Servlet API provides support for URL rewriting that we can use to manage session in this case. The best part is that from coding point of view, it’s very easy to use and involves one step - encoding the URL. Another good thing with Servlet URL Encoding is that it’s a fallback approach and it kicks in only if browser cookies are disabled. We can encode URL with HttpServletResponse `encodeURL()` method and if we have to redirect the request to another resource and we want to provide session information, we can use `encodeRedirectURL()` method. We will create a similar project like above except that we will use URL rewriting methods to make sure session management works fine even if cookies are disabled in browser. ServletSessionURLRewriting project structure



```
ServletSessionURLRewriting
  JAX-WS Web Services
  Deployment Descriptor: ServletSessionURLRewriting
  Java Resources
    src
      com.journaldev.servlet.session
        LoginServlet.java
        LogoutServlet.java
    Libraries
  JavaScript Resources
  Deployed Resources
  build
  WebContent
    CheckoutPage.jsp
    login.html
    LoginSuccess.jsp
    META-INF
    WEB-INF
      lib
      web.xml
```

in eclipse looks like below image.