Java Servlet Session Management Tutorial with Examples of Cookies, HttpSession and URL Rewriting

This is the third article in the series of Web Applications tutorial in Java, you might want to check out earlier two articles too.

1. Java Web Application Tutorial
2. Java Servlet Tutorial

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

1. What is a Session?
2. Session Management using Cookies
3. Session Management with HttpSession
4. URL Rewriting

1. 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.

A. **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.

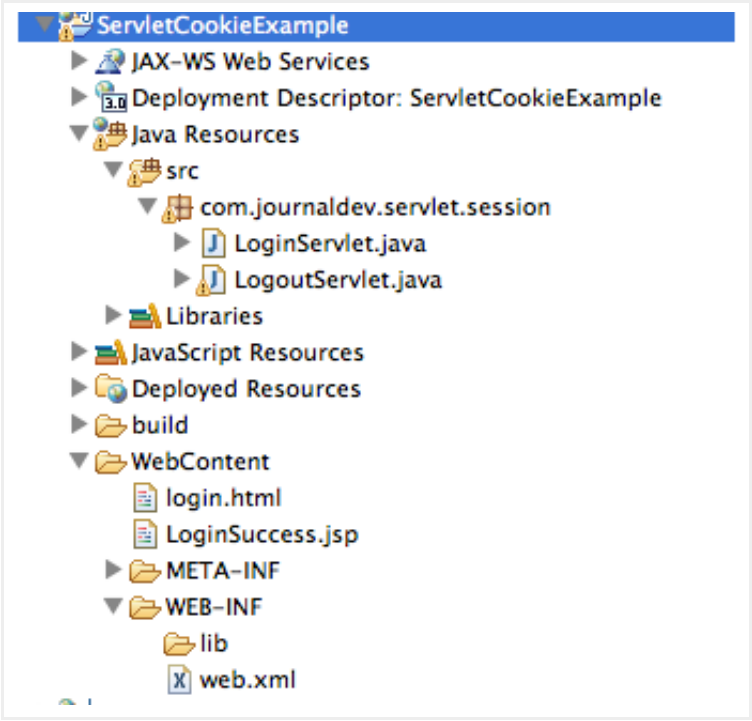
B. **HTML Hidden Field** – We can create a unique hidden field in the HTML and when user starts navigating, we can set its value unique to the user and keep track of the session. This method can't be used with links because it needs the form to be submitted every time request is made from client to server 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.
C. **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.
D. **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.
E. **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.

2. Session Management using 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.



Deployment descriptor of the web application is:

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http:
  <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.

login.html

```
<!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>
</html>
```

Here is the LoginServlet that takes care of the login request.

LoginServlet.java

```java
package com.journaldev.servlet.session;

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

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private final String userID = "Pankaj";
    private final String password = "journaldev";

    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOExcep

        // get request parameters for userID and password
        String user = request.getParameter("user");
        String pwd = request.getParameter("pwd");

        if(userID.equals(user) && password.equals(pwd)){
            Cookie loginCookie = new Cookie("user",user);
            //setting cookie to expiry in 30 mins
            loginCookie.setMaxAge(30*60);
            response.addCookie(loginCookie);
            response.sendRedirect("LoginSuccess.jsp");
        }else{
            RequestDispatcher rd = getServletContext().getRequestDispatche
            PrintWriter out= response.getWriter();
            out.println("<font color=red>Either user name or password is w
            rd.include(request, response);
        }

    }

}
```

Notice the cookie that we are setting to the response and then forwarding it to LoginSuccess.jsp, this cookie will be used there to track the session. Also notice that cookie timeout is set to 30 minutes. Ideally there should be a complex logic to set the cookie value for session tracking so that it won't collide with any other request.

LoginSuccess.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Login Success Page</title>
```

```
</head>
<body>
<%
String userName = null;
Cookie[] cookies = request.getCookies();
if(cookies !=null){
for(Cookie cookie : cookies){
    if(cookie.getName().equals("user")) userName = cookie.getValue();
}
}
if(userName == null) response.sendRedirect("login.html");
%>
<h3>Hi <%=userName %>, Login successful.</h3>
<br>
<form action="LogoutServlet" method="post">
<input type="submit" value="Logout" >
</form>
</body>
</html>
```

Notice that if we try to access the JSP directly, it will forward us to the login page. When we will click on Logout button, we should make sure that cookie is removed from client browser.

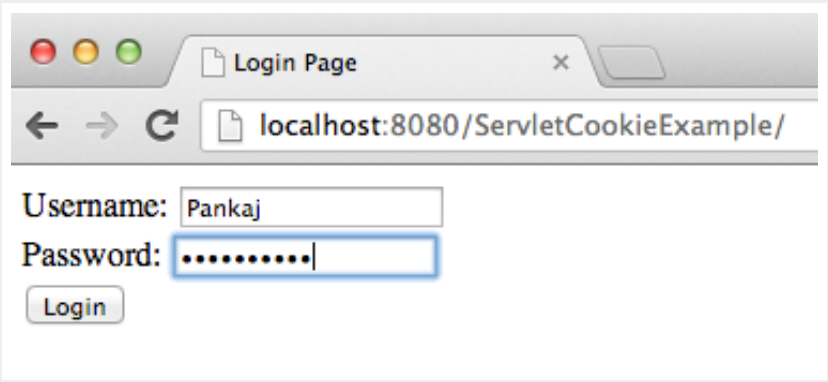
LogoutServlet.java

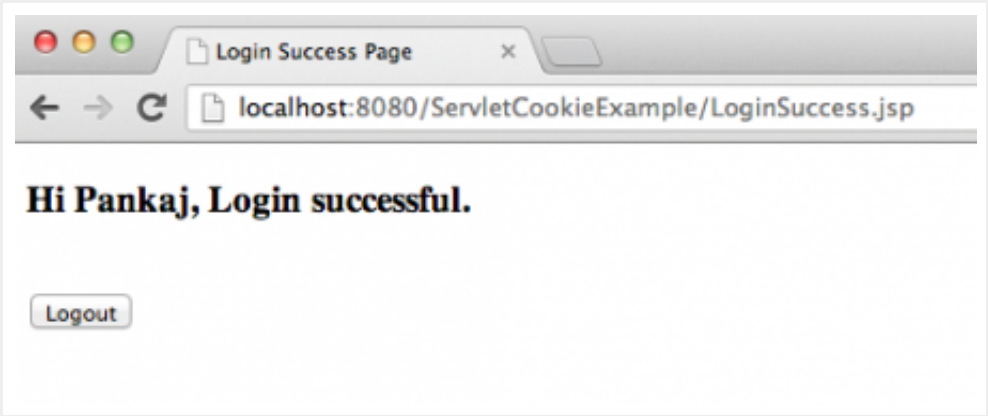
```
package com.journaldev.servlet.session;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class LogoutServlet
 */
@WebServlet("/LogoutServlet")
public class LogoutServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse
        response.setContentType("text/html");
        Cookie loginCookie = null;
        Cookie[] cookies = request.getCookies();
        if(cookies != null){
        for(Cookie cookie : cookies){
            if(cookie.getName().equals("user")){
                loginCookie = cookie;
                break;
            }
        }
        }
        if(loginCookie != null){
            loginCookie.setMaxAge(0);
            response.addCookie(loginCookie);
        }
        response.sendRedirect("login.html");
    }

}
```

```
39      }
```

There is no method to remove the cookie but we can set the maximum age to 0 so that it will be deleted from client browser immediately.

When we run above application, we get response like below images.





3. Session Management with 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.

A. **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.
B. **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:

A. **String getId()** – Returns a string containing the unique identifier assigned to this session.
B. **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

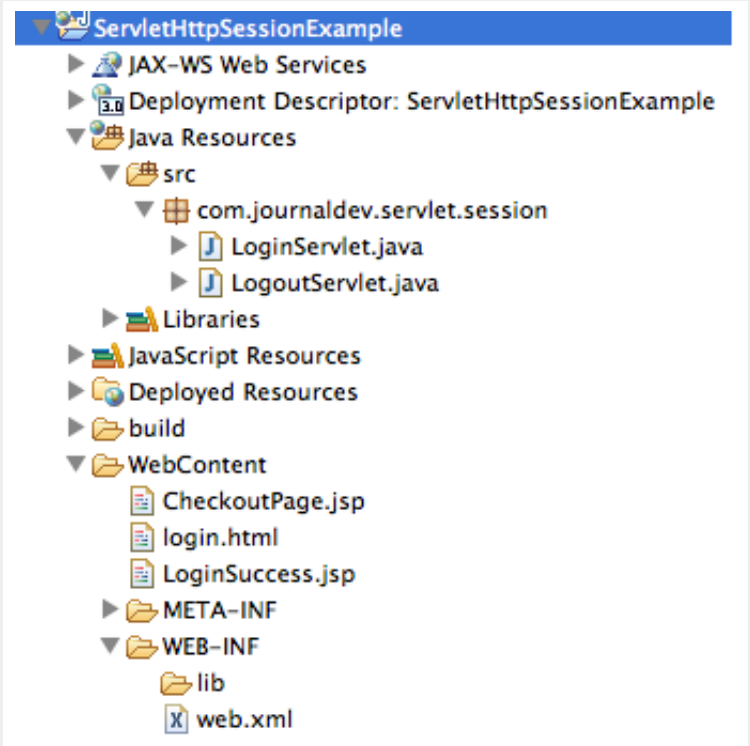
with Session attributes are `getAttributeNames()`, `removeAttribute(String name)` and `setAttribute(String name, Object value)`.

C. **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.
D. 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.
E. **ServletContext getServletContext()** – Returns ServletContext object for the application.
F. **boolean isNew()** – Returns true if the client does not yet know about the session or if the client chooses not to join the session.
G. **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.



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.

LoginServlet.java

```java
package com.journaldev.servlet.session;

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

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private final String userID = "admin";
    private final String password = "password";

    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOExcep

        // get request parameters for userID and password
        String user = request.getParameter("user");
        String pwd = request.getParameter("pwd");

        if(userID.equals(user) && password.equals(pwd)){
            HttpSession session = request.getSession();
            session.setAttribute("user", "Pankaj");
            //setting session to expiry in 30 mins
            session.setMaxInactiveInterval(30*60);
            Cookie userName = new Cookie("user", user);
            userName.setMaxAge(30*60);
            response.addCookie(userName);
            response.sendRedirect("LoginSuccess.jsp");
        }else{
            RequestDispatcher rd = getServletContext().getRequestDispatche
            PrintWriter out= response.getWriter();
            out.println("<font color=red>Either user name or password is w
            rd.include(request, response);
        }

    }

}
```

Our LoginSuccess.jsp code is given below.

LoginSuccess.jsp

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Login Success Page</title>
</head>
<body>
<%
//allow access only if session exists
String user = null;
if(session.getAttribute("user") == null){
    response.sendRedirect("login.html");
}else user = (String) session.getAttribute("user");
String userName = null;
String sessionID = null;
Cookie[] cookies = request.getCookies();
if(cookies !=null){
for(Cookie cookie : cookies){
    if(cookie.getName().equals("user")) userName = cookie.getValue();
    if(cookie.getName().equals("JSESSIONID")) sessionID = cookie.getValue(
}
}
%>
<h3>Hi <%=userName %>, Login successful. Your Session ID=<%=sessionID %></
<br>
User=<%=user %>
<br>
<a href="CheckoutPage.jsp">Checkout Page</a>
<form action="LogoutServlet" method="post">
<input type="submit" value="Logout" >
</form>
</body>
</html>
```

When a JSP resource is used, container automatically creates a session for it, so we can't check if session is null to make sure if user has come through login page, so we are using session attribute to validate request.

CheckoutPage.jsp is another page and it's code is given below.

CheckoutPage.jsp

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Login Success Page</title>
</head>
<body>
<%
//allow access only if session exists
if(session.getAttribute("user") == null){
    response.sendRedirect("login.html");
}
String userName = null;
String sessionID = null;
Cookie[] cookies = request.getCookies();
```

```
if(cookies !=null){
for(Cookie cookie : cookies){
    if(cookie.getName().equals("user")) userName = cookie.getValue();
}
}
%>
<h3>Hi <%=userName %>, do the checkout.</h3>
<br>
<form action="LogoutServlet" method="post">
<input type="submit" value="Logout" >
</form>
</body>
</html>
```

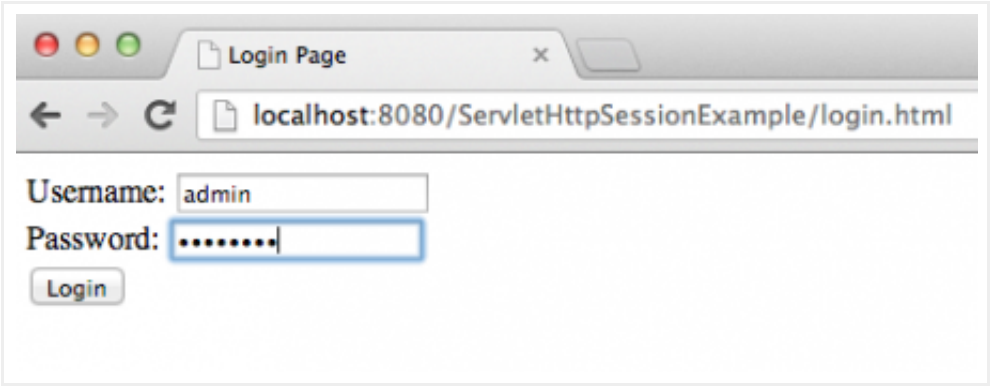
Our LogoutServlet code is given below.

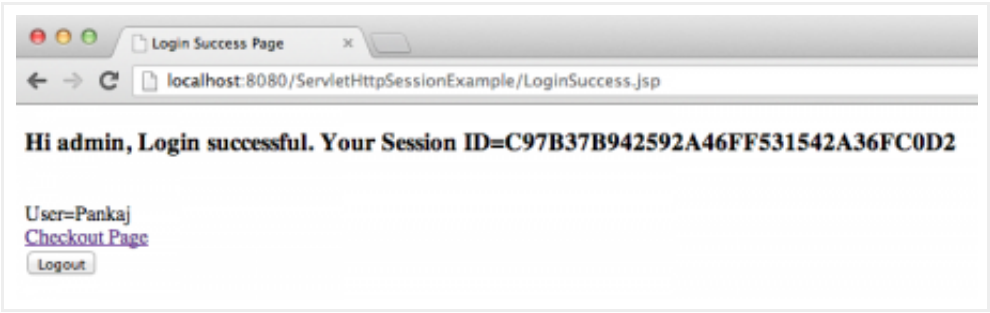
LogoutServlet.java

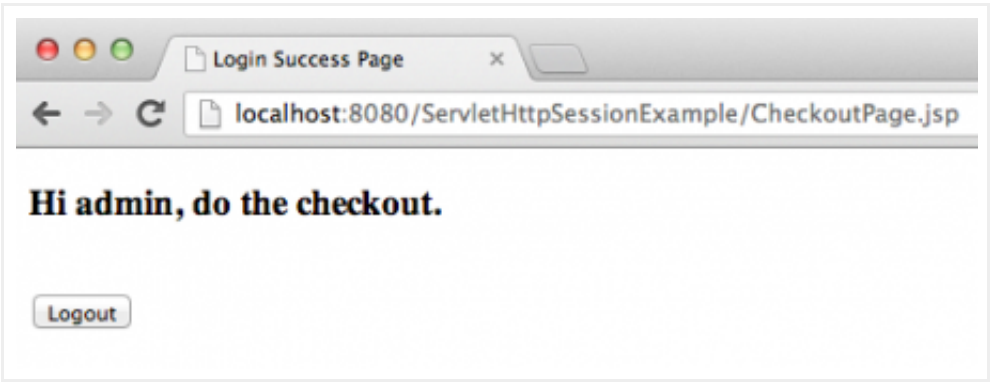
```
package com.journaldev.servlet.session;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class LogoutServlet
 */
@WebServlet("/LogoutServlet")
public class LogoutServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse
        response.setContentType("text/html");
        Cookie[] cookies = request.getCookies();
        if(cookies != null){
        for(Cookie cookie : cookies){
            if(cookie.getName().equals("JSESSIONID")){
                System.out.println("JSESSIONID="+cookie.getValue());
                break;
            }
        }
        }
        //invalidate the session if exists
        HttpSession session = request.getSession(false);
        System.out.println("User="+session.getAttribute("user"));
        if(session != null){
            session.invalidate();
        }
        response.sendRedirect("login.html");
    }

}
```

Notice that I am printing JSESSIONID cookie value in logs, you can check server log where it will be printing the same value as Session Id in LoginSuccess.jsp

Below images shows the execution of our web application.







4. URL Rewriting

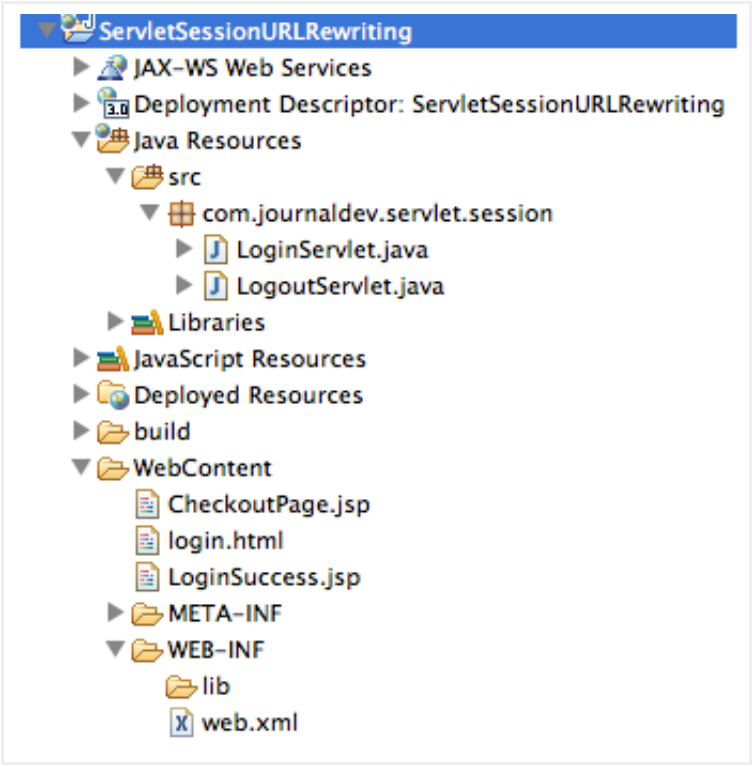
As we saw in last section that we can manage a session with HttpSession but if we disable the cookies in browser, it won't work because server will not receive the JSESSIONID cookie from client. Servlet API provides support for URL rewriting that we can use to manage session in this case.

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 in eclipse looks like below image.



LoginServlet.java

```java
package com.journaldev.servlet.session;

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

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class LoginServlet
 */
@WebServlet("/LoginServlet")
public class LoginServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private final String userID = "admin";
    private final String password = "password";

    protected void doPost(HttpServletRequest request,
            HttpServletResponse response) throws ServletException, IOExcep

        // get request parameters for userID and password
        String user = request.getParameter("user");
        String pwd = request.getParameter("pwd");

        if(userID.equals(user) && password.equals(pwd)){
            HttpSession session = request.getSession();
            session.setAttribute("user", "Pankaj");
            //setting session to expiry in 30 mins
            session.setMaxInactiveInterval(30*60);
            Cookie userName = new Cookie("user", user);
```

```
            response.addCookie(userName);
            //Get the encoded URL string
            String encodedURL = response.encodeRedirectURL("LoginSuccess.j
            response.sendRedirect(encodedURL);
        }else{
            RequestDispatcher rd = getServletContext().getRequestDispatche
            PrintWriter out= response.getWriter();
            out.println("<font color=red>Either user name or password is w
            rd.include(request, response);
        }

    }

}
```

LoginSuccess.jsp

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Login Success Page</title>
</head>
<body>
<%
//allow access only if session exists
String user = null;
if(session.getAttribute("user") == null){
    response.sendRedirect("login.html");
}else user = (String) session.getAttribute("user");
String userName = null;
String sessionID = null;
Cookie[] cookies = request.getCookies();
if(cookies !=null){
for(Cookie cookie : cookies){
    if(cookie.getName().equals("user")) userName = cookie.getValue();
    if(cookie.getName().equals("JSESSIONID")) sessionID = cookie.getValue(
}
}else{
    sessionID = session.getId();
}
%>
<h3>Hi <%=userName %>, Login successful. Your Session ID=<%=sessionID %></
<br>
User=<%=user %>
<br>
<!-- need to encode all the URLs where we want session information to be p
<a href="<%=response.encodeURL("CheckoutPage.jsp") %>">Checkout Page</a>
<form action="<%=response.encodeURL("LogoutServlet") %>" method="post">
<input type="submit" value="Logout" >
</form>
</body>
</html>
```

CheckoutPage.jsp

```
<%@ page language="java" contentType="text/html; charset=US-ASCII"
    pageEncoding="US-ASCII"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www
<html>
<head>
```

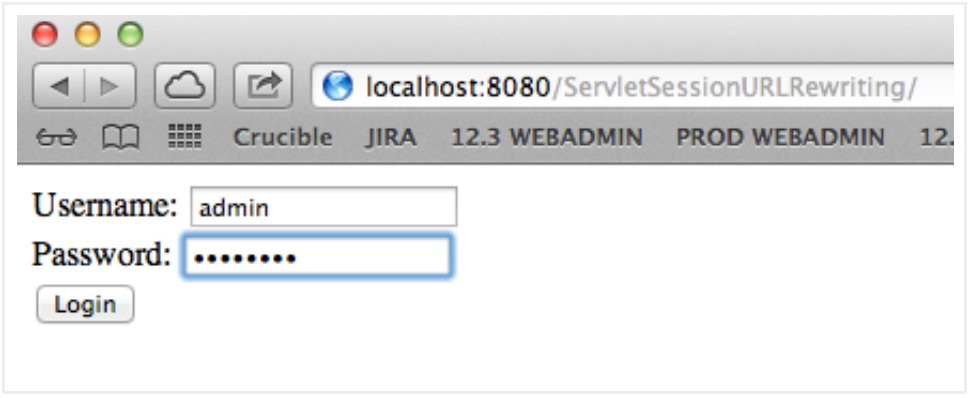
```
<meta http-equiv="Content-Type" content="text/html; charset=US-ASCII">
<title>Login Success Page</title>
</head>
<body>
<%
String userName = null;
//allow access only if session exists
if(session.getAttribute("user") == null){
    response.sendRedirect("login.html");
}else userName = (String) session.getAttribute("user");
String sessionID = null;
Cookie[] cookies = request.getCookies();
if(cookies !=null){
for(Cookie cookie : cookies){
    if(cookie.getName().equals("user")) userName = cookie.getValue();
}
}
%>
<h3>Hi <%=userName %>, do the checkout.</h3>
<br>
<form action="<%=response.encodeURL("LogoutServlet") %>" method="post">
<input type="submit" value="Logout" >
</form>
</body>
</html>
```

LogoutServlet.java

```
package com.journaldev.servlet.session;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

/**
 * Servlet implementation class LogoutServlet
 */
@WebServlet("/LogoutServlet")
public class LogoutServlet extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request, HttpServletResponse
        response.setContentType("text/html");
        Cookie[] cookies = request.getCookies();
        if(cookies != null){
        for(Cookie cookie : cookies){
            if(cookie.getName().equals("JSESSIONID")){
                System.out.println("JSESSIONID="+cookie.getValue());
            }
            cookie.setMaxAge(0);
            response.addCookie(cookie);
        }
        }
        //invalidate the session if exists
        HttpSession session = request.getSession(false);
        System.out.println("User="+session.getAttribute("user"));
        if(session != null){
            session.invalidate();
```

```
            }
            //no encoding because we have invalidated the session
            response.sendRedirect("login.html");
        }

    }
```

When we run this project keeping cookies disabled in the browser, below images shows the response pages, notice the jsessionid in URL of browser address bar. Also notice that on LoginSuccess page, user name is null because browser is not sending the cookie send in the last response.



If cookies are not disabled, you won't see jsessionid in the URL because Servlet Session API will use cookies in that case.

Thats all for session management in servlets, we will look into Servlet Filters and Listeners and Cookies in future articles.

Update: Check out next article in the series **Servlet Filter**.

Download Projects

Download Servlet Cookie Example Project
987 downloads

Download Servlet HttpSession Example Project
1118 downloads

Download Servlet Session URL Rewriting Example Project
931 downloads