SESSION-14

JSP-SESSION TRACKING

Session Tracking Mechanisms

JSP Session Tracking – If a web application is capable of remembering a client data during a session across the multiple requests then that web application is called as a Stateful Web application. Even though HTTP is a stateless protocol, web applications should be made as Statefu**l** web applications.

There are four techniques which can be used to identify a user session.

a) Cookies

b) Hidden Fields

c) URL Rewriting

d) Session Object

With Cookies , Hidden Fields and URL rewriting approaches, client always sends a unique identifier with each request and server determines the user session based on that unique identifier where as session tracking approach using Session Object uses the other three techniques internally.

1.Cookie

Cookie is a key value pair of information, sent by the server to the browser and then browser sends back this identifier to the server with every request there on.

There are two types of cookies:

- **Session cookies** - are temporary cookies and are deleted as soon as user closes the browser. The next time user visits the same website, server will treat it as a new client as cookies are already deleted.
- **Persistent cookies** - remains on hard drive until we delete them or they expire.

If cookie is associated with the client request, server will associate it with corresponding user session otherwise will create a new unique cookie and send back with response.

We will discuss Cookie in detail in one of the upcoming chapters .

Cookie object can be created using a name value pair. For example we can create a cookie with name sessionId with a unique value for each client and then can add it in a resposne so that it will be sent to client:

```
1    Cookie cookie = new Cookie(“sessionID”, “some unique value”);

2    response.addCookie(cookie);
```

The major disadvantage of cookies is browser provides a way to disable the cookies and in that case server will not be able to identify the user.If our application uses cookies for session management and our users disable the cookie , then we will be in a big trouble.

2.Hidden Field

Hidden fields are similar to other input fields with the only difference is that these fields are not displayed on the page but its value is sent as other input fields. For example

<input type=”hidden” name=”sessionId” value=”unique value”/>

is a hidden form field which will not displayed to the user but its value will be send to the server and can be retrieved using request.getParameter(“sessionId”) .

With this approach ,we have to have a logic to generate unique value and HTML does not allow us to pass a dynamic value which means we cannot use this approach for static pages. In short with this approach, HTML pages cannot participate in session tracking.

3.URL Rewriting

URL Rewriting is the approach in which a session (unique) identifier gets appended with each request URL so server can identify the user session. For example if we apply URL rewriting on **http://localhost:8080/jsp-tutorial/home.jsp** , it will become something like

SessionId=XYZ where SessionId=XYZ is the attached session identifier and value XYZ will be used by server to identify the user session.

There are several advantages of URL rewriting over above discussed approaches like it is browser independent and even if user’s browser does not support cookie or in case user has disabled cookies, this approach will work.

Another advantage is , we need not to submit extra hidden parameter.

As other approaches, this approach also has some disadvantages like we need to regenerate every url to append session identifier and this need to keep track of this identifier until the conversation completes.

4.Session Object

Session object is representation of a user session. User Session starts when a user opens a browser and sends the first request to server. Session object is available in all the request (in entire user session) so attributes stored in Http session in will be available in any servlet or in a jsp.

When session is created, server generates a unique ID and attach that ID with the session. Server sends back this Id to the client and there on , browser sends back this ID with every request of that user to server with which server identifies the user

How to get a Session Object – By calling getSession() API on HttpServletRequest object (remember this is an implicit object)

a) HttpSession session = request.getSession()

b) HttpSession session = request.getSession(Boolean)

Destroy or Invalidate Session –

This is used to kill user session and specifically used when user logs off. To invalidate the session use -

session.invalidate();

Other important methods

· **Object getAttribute(String attributeName)** – this method is used to get the attribute stored in a session scope by attribute name. Remember the return type is Object.

· **void setAttribute(String attributeName, Object value)**- this method is used to store an attribute in session scope. This method takes two arguments- one is attribute name and another is value.

· **void removeAttribute(String attributeName)**- this method is used to remove the attribute from session.

· **public boolean isNew()**- This method returns true if server does not found any state of the client.

Browser session and server sessions are different. Browser session is client session which starts when you opens browser and destroy on closing of browser where as server session are maintained at server end.

Let's discuss Session Tracking using Session Object with the help of examples.

Session Using URLRewriting

URLRewriting can be used in place where we don't want to use cookies. It is used to maintain the session. Whenever the browser sends a request then it is always interpreted as a new request because http protocol is a stateless protocol as it is not persistent. Whenever we want that out request object to stay alive till we decide to end the request object then, there we use the concept of session tracking. In session tracking firstly a session object is created when the first request goes to the server. Then server creates a token which will be used to maintain the session. The token is transmitted to the client by the response object and gets stored on the client machine. By default the server creates a cookie and the cookie get stored on the client machine.

URLRewriting can be used where we the cookies are disabled. Its a good practice to use URL Rewriting. In **URLRewriting** a string is appended

Code of the program is given below:

SessionURL.jsp

```
<html>
        <head>
                <title>How to use Session-URL in jsp</title>
        </head>
        <body>
                <form method = "post" action = "SessionURLProgram.jsp">
                <font size = 6>Enter your name  :
<input type = "text" name = "name"></font><br><br>
                <font size = 6>Enter your password  :
<input type="password" name = "pwd" ></font><br><br>
                <input type = "submit" name = "submit" value = "submit" >
                </form>
        </body>
</html>
```

SessionURLProgram.jsp

```
<%@ page session ="true" %>
<%
    String name = request.getParameter("name");
        String password = request.getParameter("pwd");
        if(name.equals("Williams") && password.equals("abcde"))
        {
                session.setAttribute("username",name);
                String string = response.encode
URL("NextPageAfterFirst.jsp?name=+name+&password=+password");
                %>
                <font size = 6><p>Please click here to go forward : </p></font>
                <font size = 8><a href ='<%= string %>'>WelcomeEncodeURL.jsp</a><
/font>
        <%}
        else
        {
                String string = response.encode
```

```
URL("encodeURL.jsp?name=+name+&password=+password");%>
                <font size = 6><p>You have entered a wrong
value  : Click here to go back : </p></font>
                <font size = 6><a href ='<%= string %>'>encodeURL.jsp</a></font>
        <% }
        %>
```

Success.jsp

```
<html>
        <head>
                <title>Welcome in In the program of URL rewriting</title>
        </head>
        <body>
<font size = 6>Hello</font> <%= session.getAttribute("username") %>
        </body>
</html>
```