Lecture 4 – Session Management

Using the HTTP protocol, we process a lot more requests where each request is coupled with an appropriate response. This is the stateless implementation where each request - response pair is independent. However, this system doesn’t track the users creating the requests. Here we use session management.

Assume a login servlet which would authenticate a user for sending an email. To implement this, we would send a token (session) along with the request & response – request & response are destroyed but not the token. This token would be sent back-and-forth and hold all the necessary information required to identify the user in the future.

There are 4 ways of implementing session management,

1. URL rewriting

We concatenate required information as a query parameter to hold information that we desire for future requests.

<http://host:port/app/url?name=val>

1. Hidden fields

We create an hidden field (type attribute) whose value is used hold the information in the value tag, we is used in future requests. Only limitation of this method, a form must be submitted initially for the information to be marked as hidden.

<input type=’hidden’ name=’token’ value=’token’>

1. Cookie

As part of the javax.servlet.http package, we create a Cookie new Cookie(“value”); which is coupled with the response using response.addCookie() method. Later we can retrieve the cookies from the request using the request.getCookies(). Apart from that we also can set the age of cookie.

1. Session objects

Session objects are analogous to college lockers. Tomcat stores & associates a unique session-id to each user. Later this session-id would be passed back-and-forth to locate the user and retrieve all stored information. We could extract the session-id from request using getSession().

About DTD,

A web.xml should have DTD (document type definition) associated with it, which is later used by an IDE for validation. The DTD holds the actual validation rules enforced on each elements found int web.xml, for e.g. servlet must be child web-app, etc.

About HTTP codes,

|  |  |
| --- | --- |
| Code | Description |
| 403 | Unauthorized access |
| 404 | Resource not found |
| 405 | Method not supported |
| 500 | Internal server error |

Most commonly we stumble upon HTTP 404 errors caused by either incorrect directory structure or web.xml formatting issues (as shown in DTD section), missing index file, incorrect class name, incorrect java used for servlet compilation or incorrect URL (http://host:port/app).