OBJECT ORIENTED PROGRAMMING II (ADVANCED JAVA)

420-P33-SU

Sessions

Why use sessions?

- HTTP is **stateless**:
  - Each request is treated as if it came from a new client. The server “forgets” the client after each request is processed.
- To overcome the limitations statelessness creates the concept of sessions was created.
  - Sessions can store information about a client from request to request.

Session - features

- Session :
  - Memory is allocated for each user.
  - Allows information to be saved when a user visits.
- A session is stored until:
  - Closes their browser
  - Stays inactive for too long (defined in web.xml)
  - Disconnects from the site (manually destroys the session)

In Java we use the *HttpSession* object

Session - configuring duration

```
<?xml version= "1.0" encoding= "UTF-8"?> <web-app
xmlns:xsi = "http://w w w .w 3.org/2001/XM LSchemainstance"
xmlns = "http://xmlns.jcp.org/xml/ns/javaee"
xsi:schema Location = "http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/webapp_3_1.xsd" id=
"WebApp_ID " version= "3.1">
< display-name> MyServlets < /display-name>
< welcome-file-list>
    < welcome-file> index.html < /welcome-file>
< /welcome-file-list>
< session-config>
    < session-timeout> 30 < /session-timeout>
< /session-config>
< /web-app>
```

Session – Unique Client ID

- An ***HttpSession*** object **uniquely** dedicated to each client will be created or recovered when the client visits a page that executes a session call.
- When this happens, the server attributes a unique ID to the client.
- Then the server adds this to a cookie in the response under the key JSESSIONID.
- Each request processed by the server looks for a cookie with JSESSIONID, and if found is compared with the Ids that are stored in memory.
- A session cookie is created on the client's side once they receive the response, and it's lifespan is assigned. A session will expire if either the browser is closed or lifespan condition is met.

Use Cases

- Managing an online shopping cart and store the added products.
- Storing and managing information about users.

- Remarks :
  - Session objects are stored on the server
  - Be careful not to deplete too much disk space on the server!

Session Related Methods

- Enable session for a user

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

- Retrieve the Session ID

```
session.getId();
```

- Retrieve the time of creation

```
session.getCreationTime();
```

Methods : add / modify data

- Use the following method :

```
session.setAttribute(key,Value);
```

  - Key is of type String
  - Value is of type Object

- Any data type can be stored in a session.

- Modifying a value is done with the same setAttribute method.

Methods : retrieving a value

- Use the following method :
  `session.getAttribute(key);`

- Returns the value associated with key, with the predefined Object type.

- It is necessary to cast the returned object into the desired type.

- If the key has no assigned value, null with be returned.

Methods: retrieving a set of values

- Use the following method :

```
Enumeration<String> myKeys = session.getAttributeNames();
```

  - Retrieves all the keys contained in the session in an Enumeration of String.
  - To retrieve the values:
    Loop over the Enumeration calling getAttribute() on each key.

Methods : deleting data or a session

Use the method to remove data:

```
session.removeAttribute (key);
```

This deletes the value associated with the key.

Use the method to delete the whole session :

```
session.invalidate();
```

This deletes the entire session.