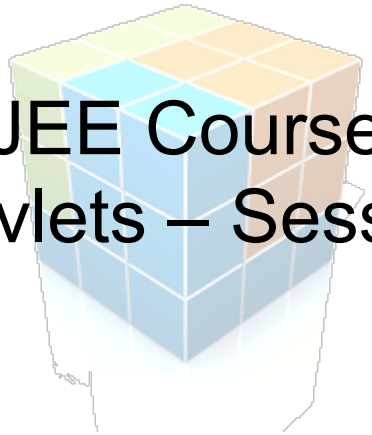


JEE Course
Servlets – Session

Márcio Fuckner

PUCPR - Pontifical Catholic University of Parana

ESIGELEC – Graduate School of Engineering

Content

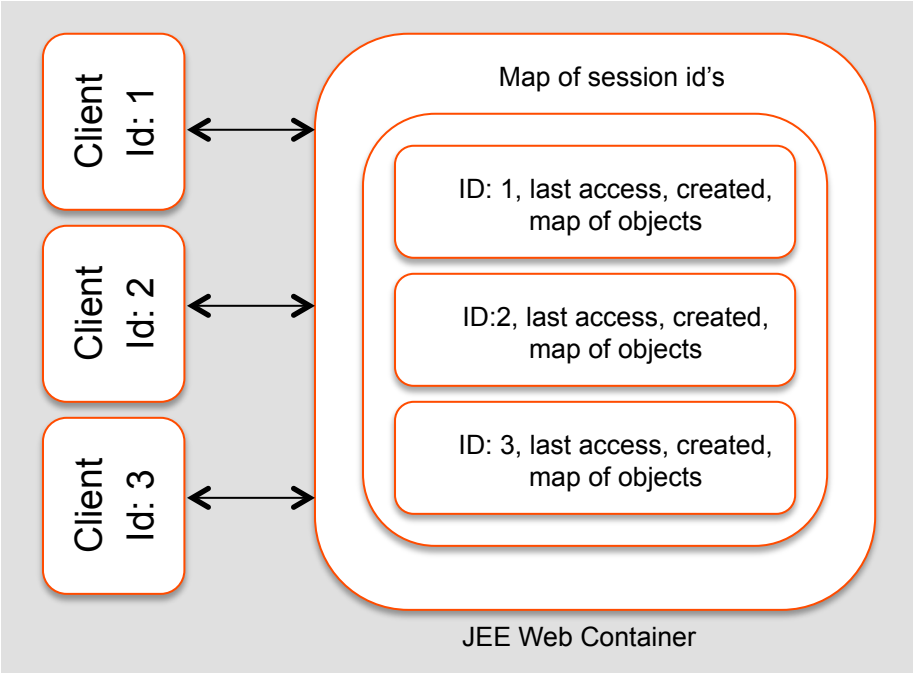
- Session definition
- Session tracking techniques
- HttpSession API
- Creating and retrieving a session
- Managing session attributes

What is a session

- The HTTP protocol is stateless: it does not store information about connections.
- However, session tracking is a « must » feature in web-based applications
- Languages and web development frameworks have different methods for session tracking

Session tracking (1 of 2)

- Session tracking is a set of techniques to manage client data during the usage.
- The most important activities of session tracking are:
  - Associate each request with a particular session
  - Manage unique identifiers for each user
  - Provide functions to manage the session data and its lifecycle



Session tracking (2 of 2)

- Three most common mechanisms for session tracking
  - Cookies

```
Set-Cookie: sessionId=324984; …
```

  - Hidden fields in HTML forms

```
<form …>
  <input type= "hidden"  name="sessionId" value="324984">
</form>
```

  - URL rewriting

```
http://www.acme.com;sessionId=324984
```

- JEE Servlet API provide a class with a set of features for session tracking called *HttpSession*

HttpSession API

- *HttpSession* has the following features:
  - Generates unique ids for each client.
  - Keeps track of all current sessions (storage is implementation dependent).
  - Associates client requests with their respective session.
  - Uses two methods of session tracking: When cookies are not available, it uses URL rewriting
  - Allows users to store objects on the server side
  - Manages the session lifecycle (explicitly opening and closing sessions or closing for inactivity)

Creating and getting the session object

- Use the *HttpServetRequest.getSession* method.
- If there is no session, the method will create and return a new one.

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

- An alternative method receives a boolean value, indicating whether the session must be created or not.

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

- In the example above, the method will return null if a session was not found
- The method *HttpSession.isNew* is used to check if a new session has been created during the call.

Example: module05.example01

Adding attributes

- HttpSession allows storing objects as attributes
- Syntax:

```
void setAttribute (String key, Object value);
```

- Objects are available until the session ends (the Garbage collector won't kill them).
- Data is stored on the server side and is solution-dependent
- It is strongly recommended to store objects that implement the *Serializable* interface.
  - Some servers implements load balancing capabilities, and they share session information, sending objects through the network

Example: module05.example02

Retrieving attributes

- *HttpSession.getAttribute* is used to retrieve session objects
- Syntax

```
Object getAttribute (String name);
```

- The method returns null If the attribute does not exist

- Use *HttpSession.getAttributeNames* to retrieve all the attributes of a session
- Syntax :

```
Enumeration<String> getAttributeNames ();
```


Example: module05.example03

Other useful methods

- *removeAtribute*: remove the specified attribute from the session
- *getId*: retrieves the unique Id generated.
- *getCreationTime*: retrieves the creation time in milliseconds.
- *getLastAccessedTime* - retrieves the last access time in milliseconds.
- *setMaxInactiveInterval* - change the inactive interval (seconds)
- *getMaxInactiveInterval* () - retrieves the inactive interval in seconds.

Example: module05.example04

Discarding a session

- Use the method *invalidate* to destroy a session
- This method ends the session and removes the link with all associated objects
- Note that it does not release the associated objects, but its references. If these are the only references to stored objects, then the GC will remove those objects

Example: module05.example05

Encoding URLs

- If your application uses session objects, you must ensure that session tracking is enabled by having the application rewrite URLs whenever the client turns off cookies
- You do this by calling the response's *encodeURL(URL)* method on all URLs returned by a servlet
- This method includes the session ID in the URL

Example: module05.example03

Further reading

HttpSession interface

http://tomcat.apache.org/tomcat-8.0-doc/servletapi/javax/servlet/http/HttpSession.html