University of Central Lancashire
UCLan

CO3409 Distributed Enterprise Systems

Lab : Java Enterprise Systems

Summary

Follow a NetBeans tutorial to create a JEE Application that provide a simple news service that uses a Message Bean, a Session Bean and an Entity Class.

Purpose

On completion of this weeks activity you should be able to:

- Create JEE applications using a variety of Enterprise Beans & classes.
- Explain the use of annotations.

Things to Remember

Some things to remember...

- Run Netbeans as "Administrator".
- When running your application, if you are using a database please ensure your database is running first (see previous lab worksheets).

Activities

The product of this weeks series of activities you will end up with 3 projects. The main one is **NewsApp**. The subprojects are an Enterprise module `ejb` and a Web module (*war*). If you close the projects, after opening the main project you can open the others by right-clicking on the *war* or *ejb* file in the Java EE Modules folder.

Creating a simple news application

Read the notes below. Carry out the NetBeans tutorial, Creating an Enterprise Application with EJB 3.1

- http://netbeans.org/kb/docs/javaee/javaee-entapp-ejb.html
- https://netbeans.apache.org/kb/docs/javaee/javaee-entapp-ejb.html

Notes

- When fixing imports, make sure you use the right package. For example, there are several Connection classes, but you aren't creating an SQL application here.
- Make sure it's the main application that you start. If it fails to run because it's not able to find the queue, see the trouble-shooting section.
- Use Payara rather than Glassfish.
- Use EclipseLink (JPA 2.1) rather than 2.0
- The message bean now uses annotations rather than an XML deployment descriptor to tell the server to create the queue so instead of the code in the tutorial, you will see the following annotations on the message bean:

```
@JMSDestinationDefinition(name = "java:app/jms/NewMessage",
interfaceName = "javax.jms.Queue", resourceAdapter = "jmsra",
destinationName = "NewMessage")
```

- To implement the `HttpSessionListener` methods, you must right-click in the class and choose **Insert Code** and **Implement Method**.
- If you need to create the `jms` resources, expand JMS resources in common tasks. Click on Connection Factories to add a new connection factory and on Destination Resources to add a new queue. Use NewMessage as the physical name of the queue. I found I needed to close the admin console and reopen it to see the newly created queue.
- If there are problems connecting to the sample database displayed in the Payara Server Log, e.g something like:

```
SEVERE:  Exception while invoking class
org.glassfish.persistence.jpa.JPADeployer prepare method …

org.glassfish.deployment.common.DeploymentException: Invalid resource :
java:module/jdbc/sample__pm),
```

create a new database, and add a jdbc resource and a jdbc connection pool to Payara.

1. Right-click on **Java DB**, **Create Database**, enter the name: `newsapp`, User Name: `APP`, Password: `APP`. Then click OK.

2. Open the Payara Admin Console (Services Tab, Servers, Right-click on Payara server). Click on **Common Tasks**, **Resources**, **JDBC**, **JDBC Connection Pools**. Click on **New**.
    1. In the General tab, enter
        - Pool Name: `newsapp`,
        - Resource Type: `javax.sql.DataSource`,
        - Database Driver Vendor: `Derby`,
        - Datasource Classname: `org.apache.derby.jdbc.ClientDataSource`
    2. In the Additional Properties tab, enter
        - Password: `APP`
        - User: `APP`
        - serverName: `localhost`
        - DatabaseName: `newsapp`
        - connectionAttributes: `create=true`

3. Open the Payara Admin Console (Services Tab, Servers, Right-click on Payara server). Click on **Common Tasks**, **Resources**, **JDBC**, **JDBC Resources**. Click on **New**.
    - JNDI Name: `jdbc/newsapp`
    - Pool Name: `newsapp`

> **Note**
>
> If your database isn't being created, open `persistence.xml` (in the configurations folder of the EJB). Switch to source mode and add the following property to `persistence.xml`
>
> ```
> <property name="eclipselink.ddl-generation" value="drop-and-create-tables"/>
> ```
>
> To change the value, erase all the string for the value, including the quotes. Type a `" (value=" )` and wait for the options to download.

Questions

Answer the following questions.

1. What is the purpose of the Persistence unit?

2. What language is the Persistence unit written in?

3. Which of the classes is an entity class? What does this mean?

4. Which import is necessary to use the `@Entity` annotation?

5. What is the meaning of `implements Serializable` in the class `NewsEntity`?

6. What is the meaning of the annotations in the declaration of id in `NewsEntity`?

```
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Long id;
```

7. In `NewMessage` Bean, dependency injection is used, e.g. What does dependency injection mean?

```
@Resource
private MessageDrivenContext mdc;
```

8. What implies that `NewMessage` Bean must have an `onMessage` method?

9. What messaging model is used: point-to-point (queuing) or publish & subscribe (topic)? What's the difference?

10. Explain the method `List<T> findAll()` in `AbstractFacade`

11. Explain public class `NewsEntityFacade extends AbstractFacade<NewsEntity>`

12. Where is the data stored and when did you create that table?

13. Identify the servlets in the application. How are they mapped to URLs? What is the alternative way of mapping servlets to URLs?