POLITÉCNICO DE LEIRIA | ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

DEI Departamento Engenharia Informática

Desenvolvimento de Aplicações Empresariais – 2019-20-1S

Engenharia Informática – 3.º ano – Ramo SI

Worksheet 1

Topics: Java EE first Cup ☺ - Creating a simple enterprise application with JPA Entities and EJBs.

In the laboratory classes of DAE we will develop an enterprise application for academic management. In this first worksheet, please execute the following steps:

1. Download and install IntelliJ IDEA Ultimate edition;
2. Install the Wildfly 17.0.1 Application Server (download and unzip it to a directory where you want to put the server, preferably to a short path in your system and without any special characters);
3. Define Wildfly as an Application Server in IntelliJ IDEA:
    - Open the Settings/Preferences dialog;
    - In the left-hand pane, in the Build, Execution, Deployment category, select Application Servers;
    - On the Application Servers page that opens in the right-hand part of the dialog, click +;
    - Select the server you are going to use (JBoss Server, *Wildfly* for friends);
    - In the dialog that opens, specify the server home, that is, the server installation directory.
4. Create a new project in Intellij IDEA:
    - Click “Create New Project”;
    - Choose Java Enterprise;
    - In the dialog on the right, select “Web Application” and click “Next”;
    - Project Name: AcademicManagement;
    - Finish.
5. Add Frameworks support to your project:
    - Once your new project opens, right-click the “AcademicManagement” project folder and click “Add Framework support…”;
    - Add the following frameworks:
        - EJB Enterprise Java Beans;
        - JavaEE Application;
        - Java EE Persistence; In the dialog below choose EclipseLink as the provider; leave “Download” selected.
    - Click “OK”.

6. Now, before continuing with the project, you need to configure the EclipseLink module on WildFly:

- Download the eclipselink.jar library file from moodle and copy it to the following folder:

  `../wildfly17.0.1.Final/modules/system/layers/base/org/eclipse/persistence/main/`
- Open the `module.xml` file in that folder and add the following element to the `<resources>` element:

```
<resource-root path="eclipselink.jar">
   <filter>
       <exclude path="javax/**" />
   </filter>
</resource-root>
```

  (right below the `<resource-root path="jipijapa-eclipselink-10.0.0.Final.jar"/>` element);
- Save and close the file.

7. Return to your project in IntelliJ;

8. Create a singleton EJB named `ConfigBean` in the `ejbs` package, on the source (src) directory of your project. This EJB should be instantiated by the server when the application starts up, so use `@Startup` JavaEE annotation right above the class declaration of the EJB;

9. Create a method `populateDB()`, which should be called by the server right after it instantiates the `ConfigBean` EJB (annotate the method with the `@PostConstruct` annotation). For now, this method should just print some welcome message in the console. We will be back here later.

10. Run the application.

11. Create the `Student` entity in the `entities` package (use the `@Entity` annotation above the class declaration). It should have the following attributes: `username` (which is the entity's ID – use the `@Id` annotation above this attribute), `password`, `name` and `email`. The entity must have at least a default empty constructor and a getter and setter method for each attribute. Also write a constructor that receives and sets all attributes' values;

12. Create the stateless Enterprise Java Bean (EJB) `StudentBean` in the `ejbs` package and write the method `create(…)` with all student attributes as parameters. This method should create and persist a student in the database. Do not forget to declare and use an `EntityManager` in the EJB (use the `@PersistenceContext` annotation right above the entity manager declaration).

13. Modify the populated method of the `ConfigBean` EJB so that it creates students and persists them in the database. You need to call the `create(…)` method of the `StudentBean` EJB. To do so, you need to declare a `StudentBean` variable in the `ConfigBean` EJB and annotate it with the `@EJB` annotation. Remember that the `populatedDB()` method is called by the server right after it instantiates the `ConfigBean` EJB (due to the `@PostConstruct` annotation).

14. You must use a database in order to persist all data from your enterprise application. We will use the database engine shipped with Wildfly (H2), and create our "dae" database:

- Go to `../wildfly17.0.1.Final/modules/system/layers/base/com/h2database/h2/main/` and run the `h2-1.4.193.jar` Java Application. You'll be redirected to a web console for the H2 database engine;
- To create a `dae` database, simply write:
  - JDBC URL: `jdbc:h2:tcp://localhost/~/dae`;
  - Username: `sa`
  - Password: `sa`
  - Optionally, you can save these settings as `dae`;
- Click "Test Connection", and verify that the "Test successful" message appears below.

15. You now need to create a datasource in the Wildfly application server, to connect to this database. Do the following steps:

- Open a command line console;
- Go to directory `../wildfly-17.0.1.Final/bin/`
- Run command `add-user`:
  - Choose option `(a) (Management User)`;
  - Username: `admin`;
  - Choose option `(a) (Update the existing user password and roles)`;
  - Enter and reenter password (e.g.: `DAEdae2020_`);
  - Answer "no";
- Open the Server Administration Console (make sure that the Wildfly server is still running, and point your browser to `http://localhost:9990`);
- On the Configuration section, click "Start";
- Choose Subsystems -> Datasources & Drivers -> Datasources -> click on '+' / add Datasource:
  - ChooseTemplate step: Choose H2, Next
  - Attributes step: Name: "dae"; JNDI: "java:jboss/datasources/dae"; Next;
  - JDBCDriver step: just click Next;
  - Connection step: Connection url: "jdbc:h2:tcp://localhost/~/dae"; User Name: "sa"; Password: "sa"; Security Domain: Leave empty Next;
  - Test connection; Finish;

16. Go back to your IntelliJ project, and in the `persistence.xml` file, replace the `<persistence-unit>` element for the following one:

```
<persistence-unit name="NewPersistenceUnit">
    <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <jta-data-source>java:jboss/datasources/dae</jta-data-source>
    <properties>
        <property name="javax.persistence.schema-generation.database.action" value="drop-
and-create"/>
    </properties>
</persistence-unit>
```

17. Run the application.

18. In order to see the generated data table, do the following:

- Go back to the web console for the H2 database engine, select your database, fill in the username and password, and click "Connect";
- Verify that there is a STUDENT table, click on it and run the "SELECT * FROM STUDENT" query;

19. Notice that you can also see the database contents by configuring your datasource in IntelliJ:

- On the right side of the main editor, click on the "Database" tab to expand it;
- Click on + -> Data Source -> H2;
- On the dialog box that opens:
  - Name: `dae`;
  - User: `sa`;
  - Password:`sa`;
  - URL: `jdbc:h2:tcp://localhost/~/dae`;
  - Leave the other fields as they are;
  - Click "Test Connection";
  - Click "Ok".
- Expand the `dae` datasource until you see the STUDENT table icon (if you cannot see it, click on the refresh button in the window toolbar).
  Double click on the STUDENT table to show its contents.