ASSIGNMENT 5

write a simple app to manage university students

Date : 11.10.2017
Enrico Magnago 186957

Introduction:
This project is an implementation of a server using wildfly connected to a MySQL database through Hibernate. There are 3 different client that use 3 different technologies to perform the same actions.

Implementation:
The project is composed of 5 maven submodules: *Client*, *Common*, *Ear*, *Ejbs* and *Storage*.
The dependencies between the modules are handled using the *pom.xml* of the Maven technology.

The *Common* module contains the definition of the shared interfaces between clients and the EJB wildfly *Server*. This interfaces are partitioned into 4 groups: local, business and rmi interfaces; additionally each pair of corresponding business and local interfaces extend another common interface. All interfaces provide the same methods with the same functionality, it has been necessary to split the interfaces for the RMI object because they are requirer to throw *RemoteException* while the EJB can not. RMI interfaces are places under the business one in a separate package.

The *Storage* module contains the annotated classes that are used by Hibernate to create the database relations: Course and Student with their many to many relationship mapped into an additional relation. The configuration files for Hibernate however are places in the EJB module because in this way when the ear archive is deployed the software is able to find them at the root of the project. Both entity have an autogenerated id, a name and a *List* of the other entity type. The annotations that specify how the many to many is mapped are in the *Course* class.

The *Ejbs* contains the implementation of the interfaces in *Common*: the stateless EJBs are *CourseBean*, *StudentBean*, *RmiBean* and *CurrentTimeBean* (this implements the functionality of the 4th assignment).
The methods of the *StudentBean* and *CourseBean*, get a Hibernate session in order to perform queries and retrieve the required information which is then sent back to the client.
The *RmiBean* is annotated in such a way that only 1 instance is create right after the startup of wildfly, the function executed after his instantiation creates a Rmi registry on the default port and binds on 2 different names 2 instances of the classes implementing the Rmi interfaces: *RmiCourse* and *RmiStudent*.
The implementation of the Rmi interfaces is placed in the *rmi* subpackage, this classes are not EJB and thus are not deployed by wildfly, their functions are implemented just like the functions of the *CourseBean* and *StudentBean*.

The *Client* module is composed of 3 submodules, one for each client: *EjbClient*, *RmiClient* and LocalClient.
The *EjbClient* gets an initialContext and uses it to get references to the ejb proxies.
The *RmiClient* performs directly the lookups to get references to the rmi proxies.
Once they have this classes that perform the exact same sequence of actions.

The *LocalClient* ignores the Ejb and Common modules and it relies only on the Storage module to access directly Hibernate, the *LocalClient* class implements some function that access directly a Hibernate session in order to perform queries on data. Once again I have implemented the same actions for the client.

I have chosen to make Hibernate discard and recreate the database every time, in a production scenario this behaviour has to be changed: it is sufficient to edit the value of the *hibernate.hbm2ddl.auto* to *validate (*for example) in the configuration file placed in the resources of the *Ejbs* module.

The *Ear* implements no functionality, it is a Maven module I use in order to create the ear archive that is then deployed by wildfly.

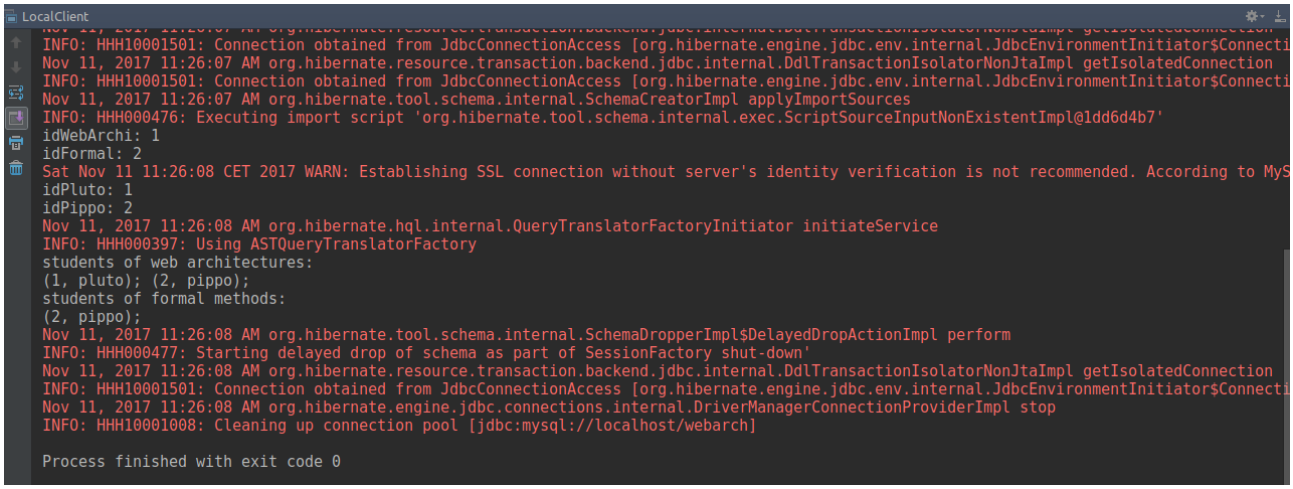
Deployment:
I have used a *MySQL* server and linked *Hibernate* to it using jdbc with the appropriate drivers and SQL dialect.
In order to deploy the application is sufficient to perform the Maven package so that in *ear/target* the *Assignment5.ear* archive is created, this archive has to be placed under the deployment directory of wildfly.
The Clients can be run as simple Java applications.

Problems encountered:
For me it has been hard to get all the configuration files right and in the correct places so that wildfly was able to find them, the errors that wildfly logs in these cases are often not much informative.
In this assignment I tried to structure the code in a general way, with the idea to reuse the same directory and module structure for the future works. However as a first attempt it has been successful in obtaining some higher level of separation between the different components and a less “black box” deployment of the wildfly application. On the other side it ended up being a little bit messy in where the things are placed, so it needs a little bit of rethinking in order to achieve a more logical subdivision of the classes in the different modules that will allow a lower degree of duplicated code (the code that performs the query to hibernate).

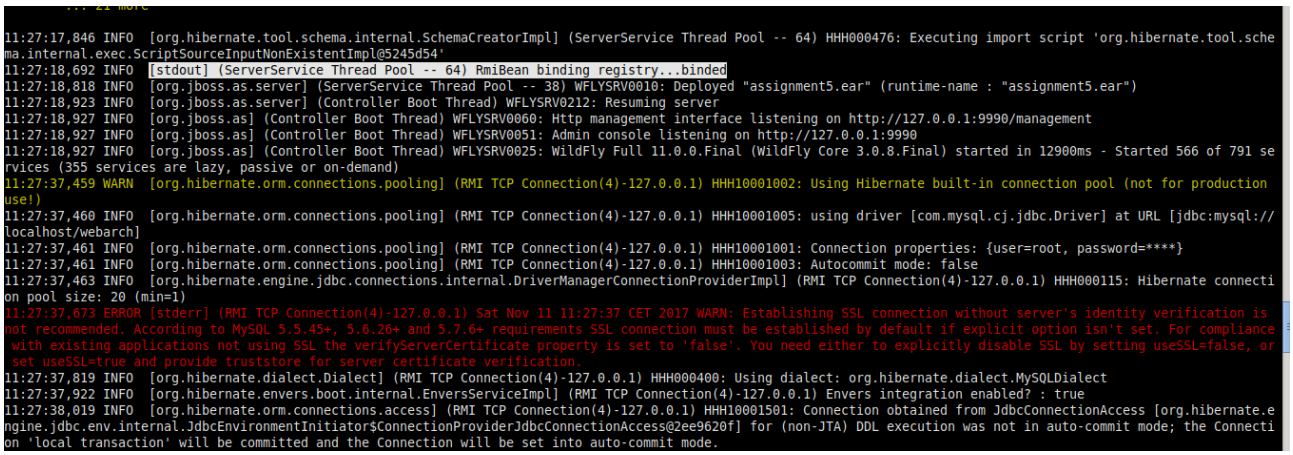
Note:

The following images show, in order, the log of the LocalClient, Server, RmiClient and EjbClient.
In the log of the Server I have highlighted the string written on stdout by the *RmiBean,* in my opinion is relevant to see that it is done only once at the beginning.
All clients create 2 students and 2 courses, both the student are enrolled in the Web Architecture course and only Pippo is attending the Formal Methods course.
The log of the client shows first the ids assigned to the courses, then the ids of the students and in conclusion for each course the list of (courseId, studentName) is printed.

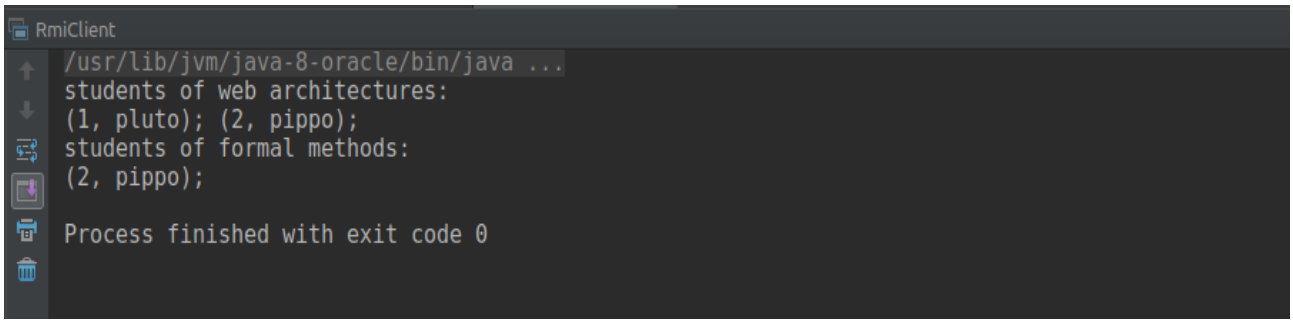
LocalClient:



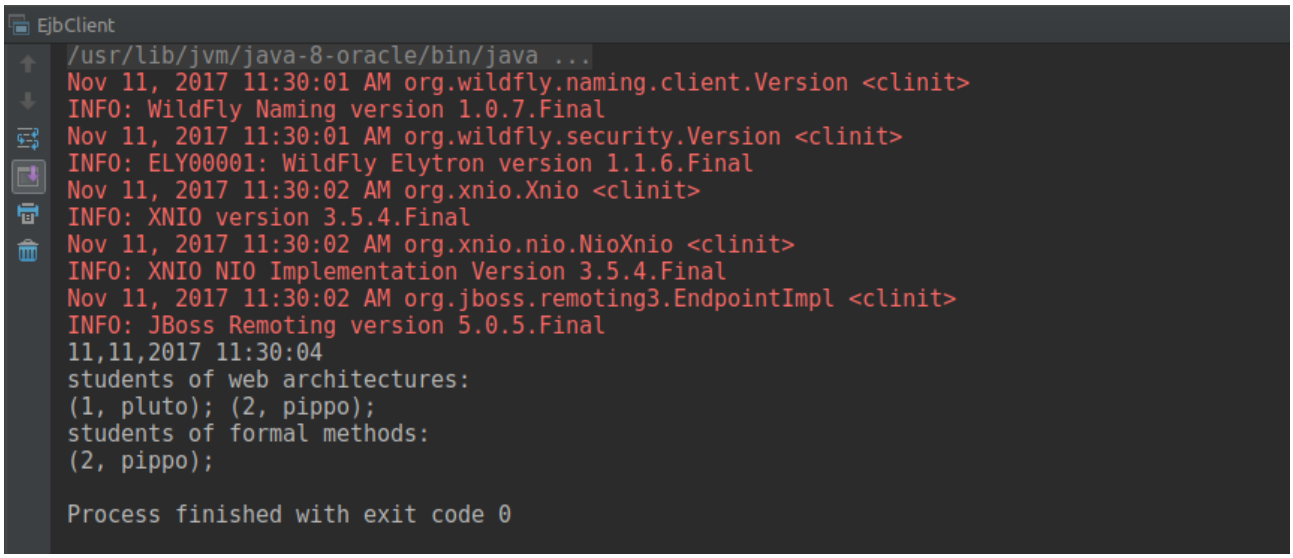
Server:



RmiClient:



EjbClient:


EjbClient
/usr/lib/jvm/java-8-oracle/bin/java ...
Nov 11, 2017 11:30:01 AM org.wildfly.naming.client.Version <clinit>
INFO: WildFly Naming version 1.0.7.Final
Nov 11, 2017 11:30:01 AM org.wildfly.security.Version <clinit>
INFO: ELY00001: WildFly Elytron version 1.1.6.Final
Nov 11, 2017 11:30:02 AM org.xnio.Xnio <clinit>
INFO: XNIO version 3.5.4.Final
Nov 11, 2017 11:30:02 AM org.xnio.nio.NioXnio <clinit>
INFO: XNIO NIO Implementation Version 3.5.4.Final
Nov 11, 2017 11:30:02 AM org.jboss.remoting3.EndpointImpl <clinit>
INFO: JBoss Remoting version 5.0.5.Final
11,11,2017 11:30:04
students of web architectures:
(1, pluto); (2, pippo);
students of formal methods:
(2, pippo);
Process finished with exit code 0