FCTUC
UNIVERSIDADE DE COIMBRA
FACULDADE DE CIÊNCIAS E TECNOLOGIA
Departamento de Engenharia Informática

Project #2
Integração de Sistemas/
Enterprise Application Integration
2015/16 – 1st Semester
MEI

Deadline: 2015-11-13

Nota: A fraude denota uma grave falta de ética e constitui um comportamento não admissível num estudante do ensino superior e futuro profissional. Qualquer tentativa de fraude pode levar à reprovação na disciplina tanto do facilitador como do prevaricador.

Java Enterprise Edition

Objectives

- Gain familiarity with the development of three-tier enterprise applications using the **Java Enterprise Edition (Java EE)** model.
- This includes the development of applications based on **Enterprise JavaBeans (EJB)**,
- The **Java Persistence API (JPA)** and the **Java Persistence Query Language (JPQL)**.
- Learn to use logging.
- Overall, you should build an Enterprise Archive, which can be deployed on a Server.

Final Delivery

- This assignment contains two parts: one is for training only, and does not count for the evaluation. You should only deliver the other part.
- You must submit your project in a zip file using Inforestudante. Do not forget to associate your work colleague during the submission process.
- The submission contents are:
  - Source code of the project ready to compile and execute.
  - Project ready to deploy on the application server (an EAR file).
  - A small report in pdf format (5 pages max) about the implementation of the project.
- After submitting, you are required to register the (extra-class) effort spent solving the assignment. This step is mandatory. Please fill the effort form at: https://docs.google.com/forms/d/1cC5_TozubwUNGgoDOZihyF9zzNaZfvEqeWi9_oHvP70/viewform

Software

Java EE Platform

For this project you are required to use a Java EE platform. We will be using the **WildFly Application Server**. This platform is open-source and is available at: http://wildfly.org/downloads/ (you may use versions **8 or 9** for this assignment, take some care with version **10**, as it is not final yet). **Eclipse IDE for Java EE Developers** is the recommended IDE. (http://www.eclipse.org/). Eclipse will allow you to control the application server easily once you download the appropriate server adapter[1]. Nevertheless, other IDEs like NetBeans and IntelliJ can do the job perfectly well.

Data Persistence

When programming in Java EE, the use of a database or persistence engine to save data is quite common. In this project, students will have to use Java Persistence API (**JPA**) as it is a Java EE standard and provides Java developers with an object/relational mapping facility for managing relational data in Java applications. The JPA engine will be **Hibernate**. The recommended databases are **PostgreSQL** and **MySQL** but you are free to choose another one.

References

Enterprise Applications Tutorial

A warm-up tutorial on how to create Enterprise Applications is **available at this course's blog (http://eai-course.blogspot.pt).** The messages from October 2014 and October 2015 let you use JPA in a standalone environment and include notes for using JPA (and EJB) in a container-managed environment:

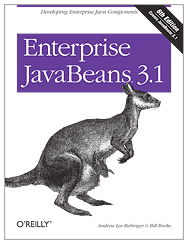
- http://eai-course.blogspot.pt/2014/10/an-enterprise-application-repository-to.html
- http://eai-course.blogspot.pt/2014/10/java-persistence-api-with-eclipselink.html
- http://eai-course.blogspot.pt/2012/10/a-simple-enterprise-javabeans-31.html

Books

There is extensible bibliography online about Java EE. Nevertheless, for this assignment, we strongly suggest that students start by reading a book that gives an overview about the basic concepts of Java EE. Contacting with a structured approach

[1] Check this site, for example: http://www.mastertheboss.com/jboss-server/wildfly-8/configuring-eclipse-to-use-wildfly-8.

to Java EE development, like what is traditionally presented in books, is important. Although students may feel overwhelmed by the size and complexity of these books, don't despair. For this assignment, you don't need to worry about security requirements. Thus, you can skip the corresponding chapters. **The following book is highly recommended**:



Enterprise JavaBeans 3.1 (6th Edition)
by Andrew Lee Rubinger and Bill Burke

O'Reilly Media
ISBN 0596158025
September 24, 2010

Online Resources

There are many online resources about Java EE and EBJs. One of the most important resources is the **Java EE Tutorial**, available at: http://docs.oracle.com/javaee/7/tutorial/doc/javaeetutorial7.pdf. This tutorial is huge and extremely detailed, but you should skim it, as it covers the whole Java EE.

Also, all documentation for WildFly is available at:
https://docs.jboss.org/author/display/WFLY9/Documentation

[Training] JPA

1. Create an Entity that stores information about football players. Each player has a name, a date of birth, height and position.

2. Add an option to the program to list all players on a given position.

3. Add an option to the program to list all players taller than a given value.

4. Create another entity to store Teams. Besides the team name, you can also have address, president's name, etc. Now, try to associate players with teams.

5. Finally, you should change your Entities and add another option to the program to allow a connection between players and their former or current team.

6. List all the players that played on a given team.

[Training] EJBs

1. Write a stateless EJB that computes the square root of a number. You should also write a client that uses this bean.

2. Write a stateful EJB that stores and retrieves a list of items given by the client. This EJB could be used to keep a shop basket. You should also write the client.

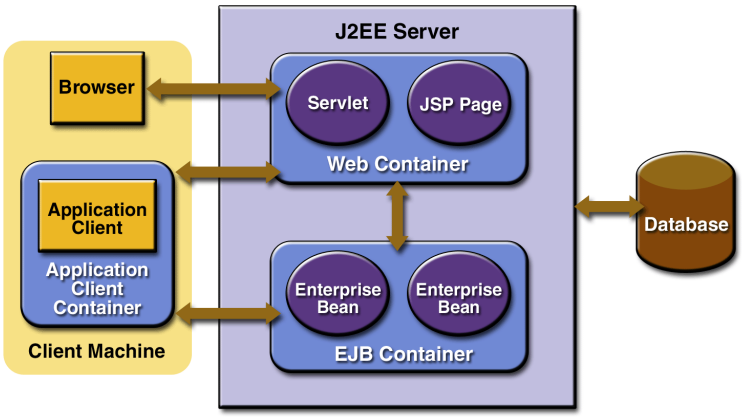
3. Write a singleton EJB to raise one alarm. The container should start the EJB automatically and should start by itself. Which solutions could you use to run multiple alarms?

4. Now, write an EJB that allows you to use all the functionalities that you developed in the JPA part.

5. Use a message bean to receive the XML data that you sent to a topic in Project 1.

Introduction to Enterprise Applications

Application Integration can be made across many types of different applications and systems. Nevertheless, nowadays, most large-scale Enterprise Systems are being developed either using the Java EE Enterprise JavaBeans model or by using .NET augmented with enterprise services (e.g. COM+ transactional and messaging capabilities). Also, most of the emphasis recently put on SOA (*Service Oriented Architecture*) and ESB (*Enterprise Service Bus*) rely on application servers and enterprise-grade systems. As a consequence, for the purpose of this course, it is important that you have a deeper contact with the reality of Application Servers before we can focus on SOA and ESB.



The idea of Java EE is that enterprise-level applications are structured in three layers: presentation, business logic and data. This is shown in the above image.
We are mostly concerned with the business code of the application. The business code runs inside a Java EE application server, which is fully responsible for managing it. The advantage of using an application server, when compared with developing a stand-alone application, is that the programmer does not have to implement many boilerplate enterprise-level functionalities. The container already provides these. In particular, a Java EE server provides for:

- Transactional contexts for guaranteeing data integrity and recovery in case of crashes;
- Connection polling for large number of invocations and optimized access;
- Integrated security management;
- In many cases, distributed computing, load balancing and clustering capabilities.

It should be noted that when one starts using an application server, it gets the feeling that it's slow, cumbersome, and that it has a difficult programming model. It is almost tempting to use a much simpler framework like JSP+Database or alike (Struts, Spring, etc.). The advantages only become clear in enterprise contexts where distributed transactions, transparent load balancing across several servers and millions of invocations have to be made with guaranteed operation. ***In fact, for small-scale applications, Java EE should not be the way to go.*** ☺

Overview

In this project, students will develop a web application to manage playlists of music. A playlist contains the identification of a set of music files to reproduce and the order in which such reproduction should occur. This application should resort to a database to keep all the information (except music files, which might be stored on the server file system).

Requirements

1. As a new user, I want to create an account, edit my personal information (at any moment) and delete my account (thus erasing all traces of my existence from the system).

2. As a user, I want to authenticate and start a session with my e-mail and password.

3. As a user, I want to be able to logout from any location or screen.

4. As an unauthenticated user, I must only have access to the login/register screen.

5. As a user, I want to create new playlists and assign them a name.

6. As a user, I want to edit the name of the playlists.

7. As a user, I want to list my playlists in ascending or descending order.

8. As a user, I want to list music files associated to each playlist. The user might have to select the playlist for that.

9. As a user, I want to be able to delete a playlist. Deleting a playlist should not delete the associated music.

10. As a user I want to add and delete music files from a playlist.

11. As a user, I want to add new music to the application, identifying the title, artist, album, year and path to the file to upload to the server.

12. As a user, I want to edit the data of music I added to the application.

13. As a user, I want to detach myself from music I uploaded. This should neither delete music from the server, nor from playlists.

14. As a user, I want to list all the music registered in the application by all other users.

15. As a user, I want to list all music registered in the application that satisfies some search criteria over the title and/or artist.

16. As a user, I want to add music to my playlists from the lists produced in (14) and (15).

Final Remarks

Finally, the following points apply:

- **Students should use a logging tool**.
- Students are free to use as many beans as necessary.
- Students should be very careful about choosing between stateless beans or stateful beans. Also, they should be very careful about not passing huge amounts of information between application layers, when they don't need to.
- Students should be careful to avoid violating layers, e.g., they must not perform logic operations in the presentation layer.
- Students need to be careful about authentication, when they access the EJBs. They must verify each access to protected resources, or otherwise anyone could do a lookup to invoke an EJB.
- Consider the possibility of developing an entire text-based interface, starting only the web tier once you have implemented all the more important functionality.

Good Work!