| | |
|---|---|
| UNIVERSIDADE DE COIMBRA<br>FACULDADE DE CIÊNCIAS E TECNOLOGIA<br>*Departamento de Engenharia Informática* | **Project #2**<br>**Integração de Sistemas/**<br>**Enterprise Application Integration**<br>**2013/14 – 1st Semester**<br>MEI<br><br>**Deadline: 2013-11-14** |
| **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 Applications

## 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 development of a **Web Tier** and the use of a **Persistence Engine**. 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/spreadsheet/viewform?formkey=dG9KTWpla0dnRW1aQ1JNdzRVTUJJMFE6MA

## Software

### Java EE Platform

For this project you are required to use a Java EE platform. We will be using **JBoss Application Server**. This platform is open-source and is available at:

http://www.jboss.org/jbossas/downloads/[1] (version **7.1.1.Final** for this assignment). **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. The Eclipse configuration is quite easy, as shown in class, refer to your Professor if you need additional help.

### Data Persistence

When programming in Java EE it's quite common to use a database or persistence engine for saving the data. In this project students will have to use one persistence engine. We will be using the 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 database is **PostgreSQL**, but you are free to choose another one.

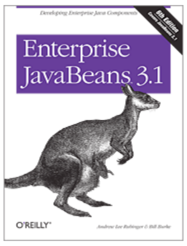
## References

### Enterprise Applications Tutorial

A warm-up tutorial on how to create Enterprise Applications is **available at Inforestudante.** It allows you to train JPA in a standalone environment and also includes notes for using JPA (and EJB) in a container-managed environment. Note that the tutorial is only a starting point and students will need to read the additional resources.

### 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 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 transactional and 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 EBJ 3.1. One of the most important resources is the **Java EE Tutorial**, available at:
http://docs.oracle.com/javaee/6/tutorial/doc/javaeetutorial6.pdf. Skim, at least: **Part One**, Chapter 1 – Overview; **Part Four** – Enterprise Beans; **Part Six** – Persistence.

[1] I have only successfully used JBoss up to versions 6.x.

You should also refer to the blog of this course. It may save you a lot of time:
Java Persistence API with MySQL, Eclipse and JBoss 7
A Simple Enterprise JavaBeans 3.1 Example with JBoss (and Eclipse)
Creating an Enterprise Application Repository in Eclipse

Also, all documentation for JBoss is available at:
https://docs.jboss.org/author/display/AS7/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?

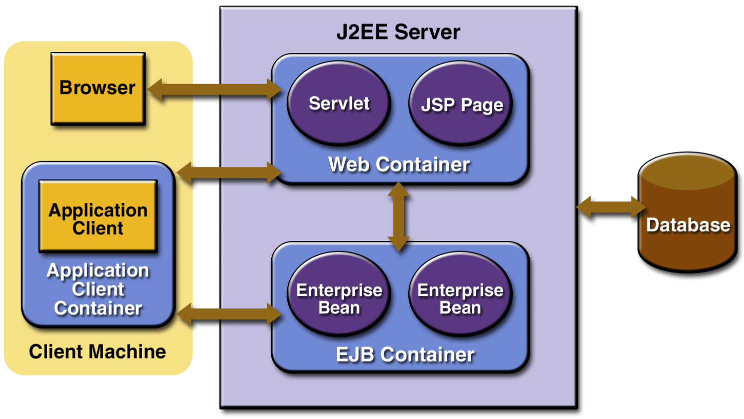
4. Now, write an EJB that allows you to use all the functionalities that you developed in the JPA part.

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.*** ☺

To conclude, note that there are two types of Enterprise Beans:

- Session Beans, which perform tasks for clients;
- Messaging Beans, which are activated when incoming messages arrive.

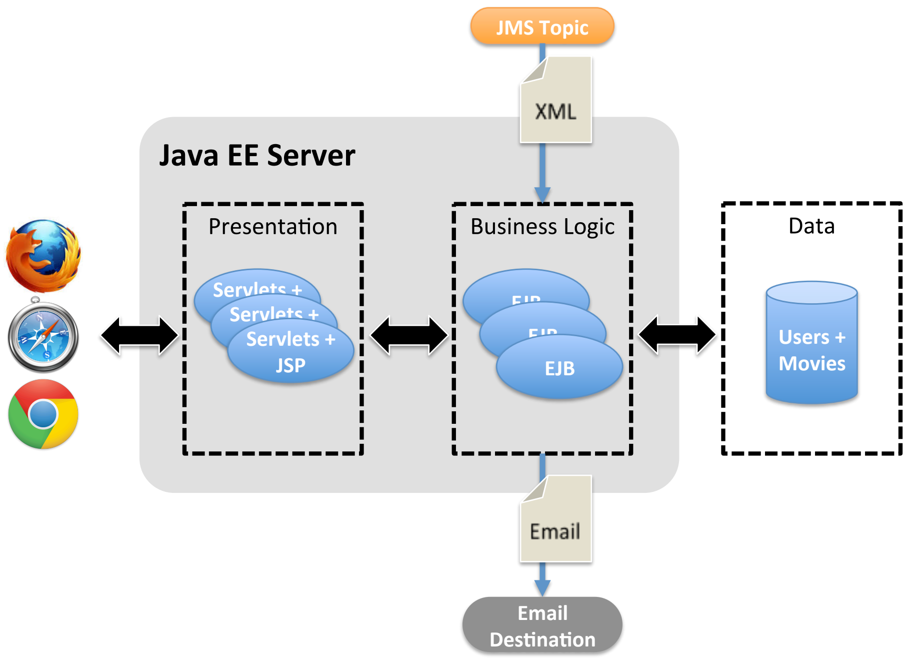
The notion of "Entity" will also play an important role in this project. An Entity is a lightweight persistence domain object, which typically represents a table in a database. Each Entity instance is a row in that table.

There has been a huge change and simplification from EJB 2.1 to EJB 3.0. In particular, (formerly) Entity Beans and "Container-Managed Persistence" (CMP) have been quite altered. In fact, in EJB 3.0 the emphasis is mostly on Session Beans and Messaging Beans. Communication with the database is either done directly with POJO and/or Java's Persistence API. Take this in consideration when reading online documentation about EJB: make a very clear view if you are reading about EJB 2.1 or EJB 3.0, because they are much different!

# Project 2 (for evaluation)

In this project, students will build a tiered application, using Java EE technologies. The JMS Topic from Project1 will feed the system with data containing upcoming movies.

In short, a user will use a browser to contact the system's front-end that will be using **Servlets/JSPs** and will communicate with **EJBs** to perform a few functions on the user's behalf (e.g., using **JPA** to read a list of movies from a database, to be displayed later at the browser). The following figure illustrates this scenario:



The movies information to be used in this project will be obtained via a message driven bean that reads movie data from the JMS Topic used in Project #1. Each reading from the Topic should result in the addition of the new movies to the database (to simplify, you can assume that each reading always corresponds to a set of new movies). The site should allow the following functionalities:

- Users should register in the site, providing a login, password and personal data, such as name and email.
- The users should be able to authenticate, using a login and a password. The site should always visually identify which user is logged on (e.g., displaying the user's login in each web page).
- Users may list all movies in the system and may also apply sorting or filtering:
  - Sorting alphabetically: Lists the movies sorted by name.
  - Filtering by Metascore: For example, show only movies with a Metascore higher than 60%, or lower than 30% or between 40% and 60%
  - Filtering by category: Example: show only movies that are in the *Animation* or *Comedy* categories.
- Users may choose (and update and delete) their favorite movie categories. Users will be informed by Email whenever a new movie that belongs to one of their favorite categories arrives at the system (via a reading from Project1 JMS Topic).

## Business logic

The business logic of the system will be implemented with mostly session beans managed by the Java EE server. Students are responsible to decide which session beans to use and should be prepared to justify their use. Also, they must carefully think if EJBs should be stateless or stateful. Be careful to define proper interfaces between the several tiers. As referred, a message driven bean should also be used to read movie data from the JMS topic used in Project #1.

## Data tier

The data tier needs, at least, data about: a) users and b) movies. Students can adopt any reasonable design for the classes, but they should use similar principles that apply to the design of tables. For example, they should avoid redundant data, unless this is necessary to favor performance.

## Final Remarks

Finally, the following points apply:

- Students are free to use as many beans as necessary.
- The web interface doesn't need to be sophisticated. The emphasis should be on the structuring of the business logic. In particular, the interfaces that are offered and published should be well thought.
- 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 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!**