1 2 9 0
FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

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

2º Projeto / 2nd Assignment

Integração de Sistemas
Enterprise Application Integration

2021/22 – 1st Semester
MEI, MES

Deadline: 2021-11-12

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.

MUITO IMPORTANTE: o código entregue pelos alunos vai ser submetido a um sistema de deteção de fraudes.

VERY IMPORTANT: the code delivered by students will be submitted to a fraud detection system.

Three-tier Programming with Object-Relational Mapping

Objectives

- Gain familiarity with the development of three-tier enterprise applications using the **Java Enterprise Edition (Java/Jakarta 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.

Final Delivery

- If students are interested in using a technology that is not Jakarta EE, they should talk to the Professor to discuss their options. Be prepared to have less support in this case.
- The assignment should be made by groups of two students. Do not forget to associate the colleague during the submission process.

- This assignment contains two parts: one is for training only and does not count for evaluation. Students should only deliver the other part.
- Students must submit their project in a zip file using Inforestudante. The submission contents are:
    - Source code of the project ready to compile and execute.
    - A small report in pdf format (6 pages max) about the implementation of the project. See below for details

---

Software

Jakarta EE Platform

To deploy the implementation of this project, the professor strongly students to use the **WildFly Application Server.**

Software Configuration

Students have two options: either installing everything on their operating system or use Docker and Docker compose. The latter might be heavier for the computer but makes configuration easy – at least if everything goes according to plan. Additionally, students may gain some skills that are quite useful for the industry. The professor will make a version of Docker compose available that includes:

- Java 16
- WildFly 24
- PostgreSQL 13
- Maven 3

Integrated Development Environment

Eclipse IDE for Jakarta EE Developers and IntelliJ are the recommended IDEs. If students are using Eclipse, they might want to consider the installation of server adapters for WildFly[1], but this step is optional. Students should use **Maven** instead of Eclipse (or IntelliJ) managed projects.

If students feel in some venturous mood, they may want to give a try to **Visual Studio Code**, which integrates quite beautifully with the containerized environment, although this option is probably poorer in terms of Java support.

Data Persistence

Object/Relational Mapping (ORM) is a technique to convert data between object-oriented languages and relational databases. In the case of Java several ORM options exist. In this project, students will have to use Java Persistence API (**JPA**) as it is a Jakarta EE standard. The JPA engine to use should be **Hibernate**. The recommended

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

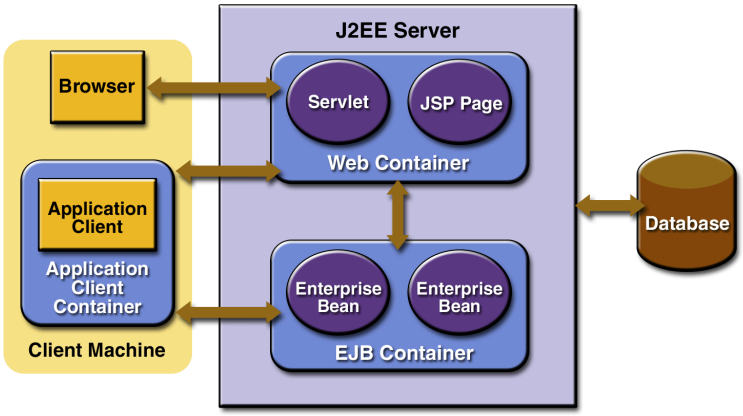
databases are **MySQL** and **PostgreSQL**, but students are free to choose another one (also, students should see the previous points regarding software configuration).

References

The professor will provide a support book that students should follow to create their first running examples of Jakarta EE. The title of the book is "Jakarta EE in Practice". Students are also invited to look for content on the Blog "Enterprise Application Integration" (https://eai-course.blogspot.com).

There are many online resources about Jakarta EE. One of the most important resources is the JavaTM Platform, Enterprise Edition (Java EE) Specification, v8, available at: https://javaee.github.io/javaee-spec/download/JavaEE8_Platform_Spec_FinalRelease.pdf. This document is huge and extremely detailed, but students should skim it, as it covers the whole Java EE 8 (the version that preceded Jakarta EE 8).

Introduction to Enterprise Applications



Jakarta EE (formerly known as Java EE) is a set of extensions to Java that aim at supporting the creation of distributed applications. Jakarta EE assumes that enterprise-level applications are structured in three layers: presentation, business logic and data, as shown in the image. While the presentation layer is increasingly developed with JavaScript frameworks, like Angular or React, in this course we are mostly concerned with the other two layers: business and data. These should run in a Jakarta EE application server, such as WildFly. 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 boilerplates enterprise-level functionalities. The container already provides these. A Jakarta 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. The advantages only become clear in large-scale contexts, where distributed transactions, transparent load balancing across several servers and millions of invocations must be made with guaranteed operation. ***In fact, for small-scale applications, Jakarta EE should not be the way to go.*** ☺

Training (not for evaluation)

JPA

1. Create an entity to store teams. Besides the team's name, teams can also have address, president's name, and other properties you find relevant. Teams will also have players (see next questions).

2. Create an Entity that stores information about football players. Each player has a name, a date of birth, and height. This is a one-to-many relation, as teams have many players.

3. Write a program to populate the database.

4. Write a program that lists all players taller than a given height.

5. Write a program that lists all the players from your favorite team in the database.

EJBs

1. Write a stateless EJB that computes the square root of a number. Students 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. Students should also write the client.

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

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

Project (for evaluation)

Overview

In this project, students will develop a web application to manage a bus company. The set of operations of this application is essentially limited to the purchase of tickets. Next, we go through the requirements of this application. Feel free to add some more functionality that you may need:

Appropriate web interface -> Apos estar tudo feito melhorar a interface

Requirements

1. As an unregistered user, I want to create an account, and insert my personal information, including email. Done
2. To create manager accounts the system should use a script, e.g., written in JPA.Fazer projeto a parte
3. As an unauthenticated user, I must only have access to the login/register screen. Done
4. As a user, I want to authenticate and start a session with my e-mail and password. Done
5. As a user, I want to logout from any location or screen. Done (kinda, falta dar opção em todos os ecras)
6. As a user, I want to edit my personal information. Done
7. As a user, I want to delete my account, thus erasing all traces of my existence from the system, including my available items. Done
8. As a user, I want to list the available trips, providing date intervals. Done
9. As a user, I want to charge my wallet to be able to purchase tickets. (You may ignore the step that involves the payment itself) Done
10. As a user, I want to purchase a ticket. I should be able to select the place. Done
11. As a user, I may be able to return a ticket for future trips and get a reimbursement in my wallet. Nope
12. As a user, I can list my trips.
13. As a company manager, I want to create future bus trips, including the departure date and hour, departure point, destination, capacity, and price. You may assume that departure and destination points are limited.
14. As a company manager, I want to delete future bus trips. The money of all tickets should be returned to the correct wallets, and the system should warn the affected users by email.
15. As a company manager I want to list the passengers that have made more trips (e.g., the top 5).
16. As a company manager I want to search for all bus trips sorted by date between two date limits.
17. As a company manager I want to search for all bus trips occurring on a given date.
18. As a company manager I want to list all passengers on a given trip listed during one of the previous searches.
19. The system sends a daily summary of the revenues of that day's trips to the managers.

Additional Remarks

This application should have three distinct layers:

- A data layer, working atop a database, to keep all the information. This layer should expose a CRUD (Create, Read, Update, Delete) functionality using EJBs. Internally, it should use Java Persistence API. **In this assignment, the result of the queries must be decided in this layer, i.e., the sorting order, what items to show etc. You should not resort to JavaScript for this purpose.**
- A business layer that interacts with the data layer, built with EJBs. Since the assignment is small, it is acceptable if the business layer uses entities and the separation of the two layers is not complete.
- A presentation layer developed in some Java-based technology, like JSFs or JSPs. Data between the presentation layer and the business layer **should be transferred using Data Transfer Objects or other techniques** (like sending a simple identifier). Students should not let the entities travel all the way to the presentation layer. For simplicity students might transfer entities between the data and the business layer.

Finally, the following points apply:

- **Passwords should not be stored in clear text**. Students should store hashed passwords in the database.
- **Students must use a logging tool**.
- Students should be very careful about choosing between stateless beans and stateful beans. Also, they should be very careful about not passing huge amounts of information between application layers (e.g., in queries), when they don't need to.
- Students should be careful to avoid violating layers, e.g., they must not perform business 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 after the implementation of the most important functionality. Please note that the weight of the web interface in the final grade is small.

Suggestion of Report

Students may divide the report according to the structure of the application, mentioning in each section the most important aspects, of which I suggest a few of them:

- Introduction
- Presentation Tier

    - Refer to the structure of the webpage, if possible, with a layout (namely templates, if students have used them).
    - Refer to the details that were most complex to implement in the interface.
    - Use a reduced number of images (possibly only one) to show the interface aspect.
    - Focus the aspect of password storage.
- Business Tier
    - Describe and, if necessary, justify the choices they made regarding the type of EJBs (stateless, stateful, interface types, transaction management, etc.).
    - Organize this section by EJBs and their services.
    - Detail the most important aspects. For example, what data is being transferred from the business layer to the presentation (and how).
    - What data do students collect from the data layer to the business layer (pay attention to quantity)?
- Data tier
    - Include ER.
    - Include a (partial) listing of entities.
- Project Management and Packaging
    - Include the description of the package of the program (pom.xml) and structure of the project(s).

Good Work!