Project 3

Enterprise Application Integration (MIEBIOM)
Department of Informatics Engineering
Delivery date: 12 December 2022, 13h00

Objectives

- Gain familiarity with message-based communication.
- Learn how to use the Java Message Service (JMS) 2.0 API.
- Create asynchronous loosely-coupled applications.

Final Delivery

- 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 requested applications ready to compile and execute.
- 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=dHFha3NuWE1pYWJDbTJMTWcxTWRMS1E6MA

Training

Go through Section 7.1 of the book Jakarta EE in Practice. In the end, you should be able to run message producers and consumers using queues, create publishers and subscribers (including durable subscribers), and reply to temporary queues.

Project

Description

In this project, we will build a scenario that makes use of messaging to support interaction between one series director and a group of actors. In this scenario, there are three nodes (director, system, and actor), which make use of the series database from the previous assignment. The director and actor are standalone applications, whereas the system node runs within the application server. Since one of the main goals is to learn the JMS API, all communication between nodes will be supported by JMS, including the synchronous ones (except if otherwise noted).

System node

This is the only node that has direct access to the database and is responsible for carrying out any operations that require database access. As there is no user interface in this node, you can deploy a message driven bean to take care of the necessary functionality.

Director node

This node refers to a single application instance that would be used by a director. To simplify the assignment there are no authentication procedures or multiple users using this application. It allows executing the following functions:

- Add a new series. This results in all actors receiving a notification with information about the series.
- List the Director's series.
- Invite all actors to fill in a cast position in a given series.
- Select one actor among those that accepted an advertised position and add it to the respective series.

Actor node

Each actor, besides complying with the above functionality, can perform the following operations:

- Register using a name and a password.
- Login in the application, using the name and password. In case of success, an access token is generated and delivered to the user, which should then use it in all subsequent operations.
- List all series in the system.
- Show pending invitations. The pending invitations should be available even if the user restarts the application.
- Accept or reject an invitation.
- Logout.

Final remarks

- All incoming messages must have an immediate on-screen print.
- You must use JPA if database access is required.
- Depending on your architectural choices, you might want to run code in an additional Thread. You can see how to use one at:
  http://docs.oracle.com/javase/tutorial/essential/concurrency/runthread.html
- Some of the functionality has been simplified to shorten the assignment. Make sure you identify which points should be built differently in a real system.
- As you may notice, the description of the assignment is intentionally left incomplete. You are expected to be able to make sensible implementation choices at specific points. Refer to your Professor for any architecture/functionality doubts.

Good Work!