Departamento de Engenharia Informática
Faculdade de Ciência e Tecnologia
Universidade de Coimbra

Java Enterprise Edition

Enterprise Application Integration

Coimbra, 13th of November 2015

Flávio J. Saraiva, nº 2006128475, flavioj@student.dei.uc.pt
Mário A. Pereira, nº 1998018322, mgreis@student.dei.uc.pt

Index:

Objectives

The main goal of this project was to build a three-tier enterprise application using the Java Enterprise Edition (JEE) model that could be used as an Enterprise archive and deployed on a Server.

In order to achieve we had to develop an application based on Enterprise JavaBeans (EJB) running on a Wildfly 9.0.1. server.

To deal with the Data tier of the proposed application the use of the Java Persistence API (JPA) and Java Persistence Query Language (JPQL) was also necessary.

Finally the use of logging was also asked.

Project Overview

Based on the proposed objectives we were asked to develop an application that managed music playlists. This application should be able to create, update and delete such playlists. First of all it would be necessary for the user to be registered and signed in to the application. In order to achieve such goal an authentication and session mechanism had to be implemented.
To produce such playlists authenticated users should be able to upload musics into the system. Beside the actual music file, information about each song would have to be permanently stored in the system. In order to consult this information and enable the user to build its playlist, all the business logic that allowed the authenticated user to query such information and select the desired musics to his playlist had to be built.
All the information regarding the users, musics and playlists had to be permanently stored in the system to allow for latter use.
Finally a user interface that would allow users to interact with the system had to be developed.
System requirements can be consulted in the project overview document presented by the teacher.

Implementation

To produce such a system, we divided our application into three components, each one taking care of a layer of the 3-tier application model:

- The presentation layer was implemented using a command line terminal and via dynamic web pages constructed using JSP technology that can be presented to the user in any common web browser;
- The business logic layer was taken care of by the use of Enterprise JavaBeans running in a Wildfly 9.0.1. server. This tier was responsible for receiving and processing the client's requests and querying the data layer for the appropiate information which could be used by the presentation layer to generate dynamic web

pages or simple strings containing the requested information that could be presented to the user via console;

- The data layer, that was responsible for storing the information was constituted by a MySQL database and the Java Persistence API which allowed all the information to be queried via Java Persistence Query Language and the results to be aggregated in Java obje

Each of these tiers will be described in detail.

Presentation Tier

Business Logic Tier

To implement this tier we used the Enterprise JavaBeans Architecture. This architecture shares many similarities with the classic Remote Method Invocation model in which stubs of the object interface are made available to the remote client. This client invokes the remote methods of the interface as if they were from locally available objects.
In order for such system to be implemented these Enterprise beans must be run in a container which is part of the server and provides additional services to the bean which include, transactions, synchronization and security.
In order to provide such services the bean container must be able to intercept remote calls from the client to bean methods. The container is able to interpose code to provide these services before the method call, and when the method completes before any values are returned to the client.

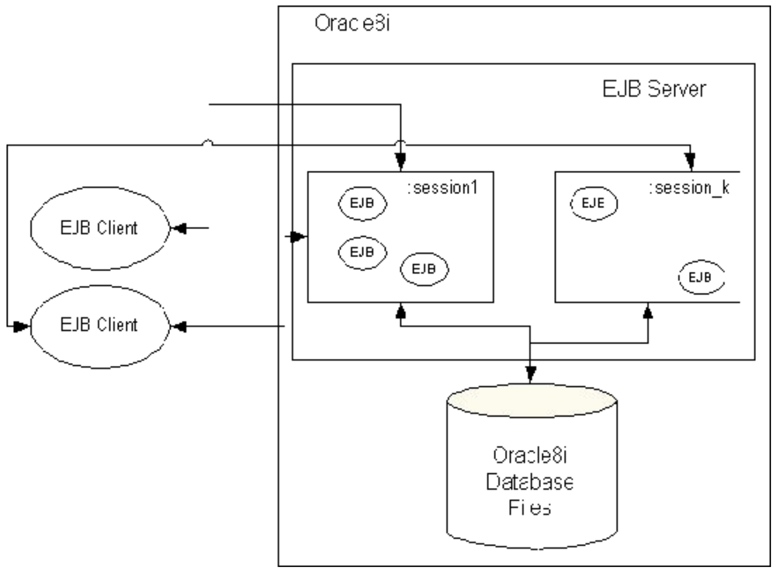

Fig. 1 - EJB Architecture

By allowing the client to interact with a remote interface with simple method invocation of a local stub, a high level of abstraction is achieved and all the distributed system complexity regarding transactions, synchronization and security are hidden from the client, allowing faster development of complex enterprise systems. Developers only have to construct EJB objects and interfaces and define their methods and behavior with the use of notations.

In our application two types of EJB were used:

- Stateless EJB: this type of enterprise bean is normally used for independent operations. As the name indicates it does not have any associated client. The EJB container normally creates a stateless bean pool and assigns them freely to process the clients requests. Our system has three different stateless EJB one to support accounts, one to support musicfiles and other tyo support playlist operations.
- Singleton EJB: when using this type of EJB only one instance of it is ever created inside the container. This bean is initialized when the system is started. this type of bean supported our database.

In order for the EJB to share each others methods. it is possible to use EJB injection via the @EJB annnotation. This allows for dependency creation between instances of each enterprise bean.

Finally it is possible to make the EJB available to local and remote users by using annotation. This allows for the an interface with the EJB public methods to be made available in a process very similar to remote method invocation.

Data Tier

In order to make our data persistence our application uses a mysql database. To interact with it we used Java Persistence API. This API allowed us the creation of entity classes that mimetized rows in a relational database and allowed for relations to be established between entities. By using a persistence context and configuration defined by the persistence XML file, we were able to make this objects persist in the database, and also query the database utilizing Java persistence Query Language.
This allowed for an abstraction of the database rows to be created and manipulated as if they were Java objects. And also allowed this objects to be made persistent without the need of disk files.

Discussion of Results

While implementing this project there were certain difficulties regarding system configurations. As such the Presentation layer was not complete.
Nevertheless it is possible to control the application via console.
The logger was also not implemented

Conclusions

Although the system is not completed, the data and business logic, containing EJB and JPA technology were completed and operational.
Most of the proposed objectives were attained.

Annexes