Software architecture
Assignment 2

Dehouck Samuel, Delhaye Quentin

March 20, 2014

Chapter 1

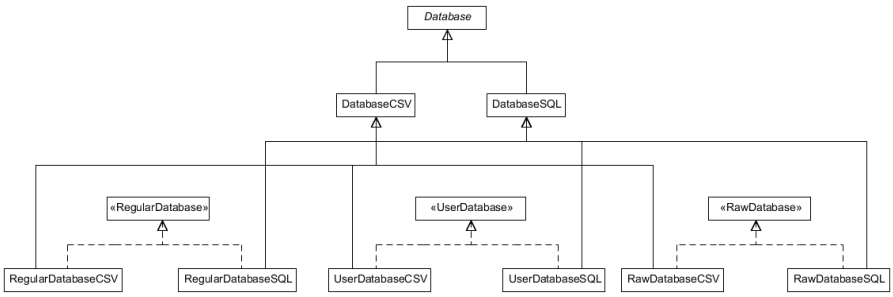
Introduction

In this project, we were asked to refactor (flawed) three-tier architecture that implements a web portal applicaton which allows to store and retreive informations about books, articles, etc... Moreover, we needed to identify the different flaws of this architecture.
More precisely, we had to refactor the database layer in order to be able to easily insert a new format of database. The details of this new implementation will be detailed in the first section. Next, we will give the flaws that we found in the architecture.

Chapter 2

Refactoring

For this project, we had to refactor the database layer in order to add a new database based on *CSV* files. The original implementation didn't allow us to easily add this new layer and that is where the refactoring is done.
First, we needed to change some names to make them more precise: *RawDatabase* became *RawDataseSQL*, *UserDatabase* *UserDatabaseSQL*, *RegularDatabase* *RegularDatabaseSQL* and finally *Database* *DatabaseSQL*.
In a second time, we added a new level of abstraction with some interfaces that are implemented by the SQL components inherit: *RawDatabase*, *UserDatabase*, *RegularDatabase*. We also created a new abstract class Database from which DatabaseSQL inherits. With this generic interfaces and the abstract class, it became much easier to add a new kind of database.
Finally, we created the *CSV* database with some new classes *RawDatabaseCSV*, *UserDatabaseCSV* and *RegularDatabase* that implement the corresponding interfaces and *DatabaseCSV* inheriting from *Database*.
All these classes have been reorganized in the new packages *db.flatfile* and *db.sql*.



As we can see with this new architecture, it is much easier to add a new kind of database.

Some others changes needed to be done in order to have a working implementation:

- We added a new line in the file web_portal.cfg to specify the format of the database.
- The constructor of the class *ApplicationFacade* has been modified to take the format into account.
- The constructor of the class *DatabaseFacade* has been modified in the same way and now build the database accordingly.
- In order to store the user profiles into the database, a new method *asCSV* has been added in the classes *UserProfile* and children.

Chapter 3

Design flaws

When we refactored the database layer, we found that the database needed to ask the *UserProfile* to give its information in a format that it was possible to store (*asSQL* and *asCSV* methods). It clearly introduce some coupling that could be avoided if the database could ask the *UserProfile* to give a generic format of itself. Then, it would be the job of the database to convert it into *SQL* or *CSV* in order to store it.

Chapter 4

Conclusion

Contents