Java client for comic books

Students

BARBEROT Mathieu - RACENET Joan

Tutors

BOUQUET Fabrice - COQBLIN Matthias

May 9, 2012
UFR ST - Besançon

Table des matières

Introduction

Managing a library can be hard, especially for collectors, who could possess hundreds of books. That is why websites like BDovore [1] have been created. BDovore is specialized in comic books and allows its users to manage their library online. But because we are not always connected to the Internet, UFR ST and BDovore started a project : create a software, based on the Java language, that you could use offline on your computer and that would providing the same functionalities as the website.

In this report, we will talk about our contributions in this project. In a first part, we will explore the software with the user's point of view and see what are the actual implemented features. Then, we will present our work on these features, to reinforce them or to make them better. Finally, we will see the totally new features that we have added, before concluding.

1. www.bdovore.com

Chapitre 1

The Software

In this chapter, we will present the software with the user's point of view. We will, in one part, list the achieved functionalities which are yet implemented. Next, we will talk about the unachieved one and so what will be our future contributions.

1.1 The features

1.1.1 Searching an album

As the user will need to find the albums he owns in the software, a research function was necessary. Really basic, it was only using a single keyword and the user had to take care of capital letters and other special characters, as accents. By example, the research "asterix" did not give any result, in contrary to "Astérix".
The results are displayed in a list with the title, the tome number, the series, the genre and the ISBN (a unique numeric book identifier). A double click shows a window with details about the album.
If your research has a lot of result, four buttons are available in the bottom of the list to go to the first, previous, next or last page.

1.1.2 Add or remove an album oh the user library

Once the window has appear after a double click, some check-boxes are displayed. To add an album, the user has to check the box named "Owned". Then three check-boxes become available for details, which enable the user to notice if the album is dedicated, lent or if he predicts to buy it later.

1.1.3 See statistics

Once the user library is filled up, he can see some statistics on that library. There are two types of statistics : the first type, on the top, is displaying amounts of album of each type (albums in the library, owned and to buy). The second one shows percentages of genres, editors, scriptwriters or cartoonists.

1.1.4 Configure the software

Finally, a configuration tool is provided for setting a proxy server or to connect the user on its website's account (which allow him to synchronize his library with the online one).

1.2 Unachieved functionalities

1.2.1 Update the research base

The research feature is working with a database of all the albums which are listed on the website. To keep it to date, a feature that download all the new albums on the website was planned but previous groups did not have the time to finish it.

1.2.2 Synchronizing user account

As user can suggest albums on the website or changes details of the albums on the software, the synchronizing feature had to get the two libraries, search the differences between them and update them. For ambiguous cases, a window shows the differences and asks user to decide whether the online one or the offline one is the more up to date. All the reflections on the feature were done but were not integrated in the software.

1.3 Our contributions

For the time we had to work on this project, our main goal was to finish to put in place some unachieved functionalities. In this purpose, we had to :

– Change the **research tool**, to make it more efficient and intuitive, so the user could write its requests like in any search engine : the user would be able to make a request with several words in any orders and without caring about accents or syntax. He could also use some special characters and keywords to specify its research (like the "*" joker or the double quotes).
– Write the **web service**'s functions. It was a necessary step to do before taking care of the synchronization part. Indeed, the web service is the key between the both databases and it will permit to get the informations in them, like the comic books that the user owns or the details about a volume.
– As we see, a certain reflection was done about the **synchronization** by the previous students groups who worked on this project. Our role in this part was to make the reflection become real and so finishing a big feature of the software. So we had to make the mechanisms which would resolve the eventual conflicts between the online and offline libraries of the user and make them the same.
– Finally, all along the time we worked on the software, we have done some **minor adjustments** (for example for the ergonomics of the Graphic User Interface).

Chapitre 2

Reinforcing the software

Before any modifications on the software, we had to discover how this one was built. It was necessary to see where we would add our functionalities and, eventually, where we would have to make modifications before any additions.

2.1 Understanding previous students work

Working on a software with a substantial base coded by other people is a pretty difficult exercise. That's why before any reflections, we had to appropriate the architecture and the code. Even if it could take a lot of time, this is not a waste : it's important to see where exactly we would add our modifications and if these modifications would be immediately possible.
This preliminary step was particularly difficult on this project. Indeed, two groups of students worked on it at the same time the last year, so we had to discover the work of both. Furthermore, it was the first time for us that we had to work on a software with such an architecture : embedded database, web service (coded with the web-oriented language PHP) and distant database (the one of the website).
After watching the source code of the two actual programs and read the documentations left by the students, we quickly understand that :

- One of the group had totally remake the organisation of the local database (cf annexes for schemas of the previous and the actual databases).
- The other group works especially on the synchronization part. They reflected on the conflicts that the user could have if certain informations are different between the website and the software (example : a volume is in the user's library on the website, but not in the program).
- They thought of the function that the web service must have ... but don't write them !

That's why we saw that a complete review of the software would be necessary before anything else.

2.2 Laying the foundation

First of all, we decided to unify the two different projects in an only one. To work on this, we had to install our work environment :

- A collaboration system (Subversion). Not necessary, but very useful. It permits to store the projects online (and work on them with any computer) and handle the different versions of the project (if the last version didn't work -for example - we still could come back to a previous one).
- A web server on the local computer, to "simulate" the server of the website. We stored on it a copy of the database of BDOvore in order to test our future web service.
- The embedded database that we could generate with some commands.

So we could finally work on the project. To unify the two versions, the main work was to adapt the functions which use the embedded base (with the SQL language) to the new model of this database and check every of them, so the software could -simply- work. To be sure of our work, we tested all the new requests to the database out of the software, directly on it (to be sure, in case of an error, that the mistake is really in the request and not in another place of the program).

The software with the contributions of the two previous groups was finally unified. We could now testing the actual features and their efficiency.

2.3 Test and updates

Our first observation was that the search engine (the tool who allows to find a specific volume in the user's library or in the volumes referenced in the website) was functional but very limited. In fact, the research gave you results which fit perfectly with the syntax of what you search. As we saw in the introduction, typing a research with the keyword "asterix" won't give you any results with the word "Astérix" because of the capital letter and the accent that the search engine didn't know how to ignore.
To fix this problem, we decided to update the search engine (Lucene, an external search engine that you can use with many programming language) to its new version. This one was improved and allows many new possibilities, like the use of a joker character ("*") or the use of some keywords like AND, OR or NOT to precise our research. It allows too to fix the problems of accents or capital letters.
Furthermore, we noticed some things we already knew : the synchronization tool doesn't work, so the base of referenced books was nearly empty (except for some entries that was inserted manually to test the search engine) as the library of the user.
Finally, while we was at it, we updated the database engine ("H2" the one which is used to the embedded base) in the purpose to increase the performance and the rapidity of the recovery of the data. Additionally, we brought some minors modifications on the GUI (Graphic User Interface, what the user use to communicate with the program, like the buttons or the text fields) and in the code.
As we see, we did an important work to improve the functionalities which was still

implemented in the program, in the purpose to make it more user-friendly, efficient and stable.

Chapitre 3

Web service and Synchronization

In this part, we will focus on our biggest addition on the software : the mechanisms for the synchronization of the library of the user, both in the software and on the website, and, for that, focus on the creation of the web service (and, before, what exactly a web service is), then, on the synchronization of the local database.

3.1 What is a web service ?

The first question we had to give an answer was : what exactly a web service is ? In fact, we had never approach such a concept before this project. After researches, we learned that a web service is an application, running on a server, which allows communication between other programs, without caring about the languages or the technology they are made with. It makes public a list of functions that these programs can use : it's a kind of layer between the client (the application which will use the web service) and the server.
(For example, Google provides a web service that anybody can call in its own application. With it, you can for example incorporate the search engine or use its speller).

3.2 Why do we need it ?

In our case, we had to write a web service to enable interactions between the website and the software. Theses interactions consist of getting data on BDOvore : the list of the referenced albums on the website, their details (authors, genres, ...) or the current user's library with all its informations (if some albums are lent or dedicated, ...). The web service had to be able to do modifications on the web service (for example, if the user add an album in the software, it must be reported on his online account).
A list of functions was still written in the WSDL file (a file who listed all the functions of the web service that the clients can use) by the previous groups, but weren't particularly useful in our point of view.
Our work was finally to completely rewrite the functions of the web service. We wrote

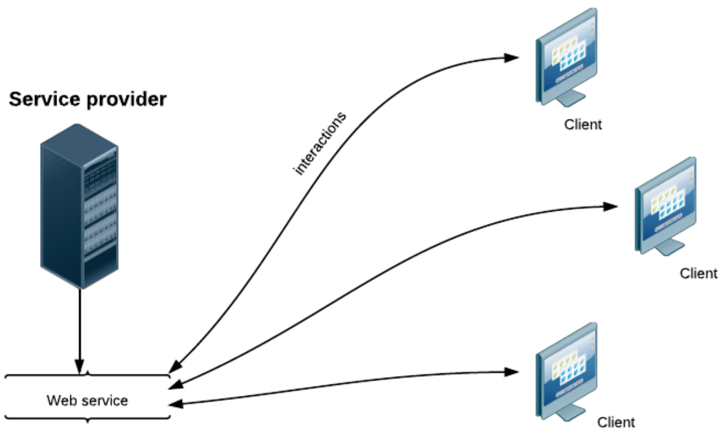


FIGURE 3.1 – How a webservice work - very simplified !

it in the PHP language, because of its facility to work with a database and because it's compatible with the most of web servers.
For such a small development, we could think that a web service is not necessary. In fact, it was "just" advisable for some reasons :

– Simplifying the development, with the PHP language
– With it, if the server change (for example, modifications in the structure of the web site's database), we won't have to modify the client (the software), but just the code of the web service. It's more handy for the future users (who won't have to update their software) and the developers.

With finishing the web service, we were now able to get all the data we need from the website to use them in the software, via the synchronization tool.

3.3 The synchronization

As we told before, the synchronization is the mechanism which will certify the consistency of the data in the software and in the website. It is divided in two parts : the synchronization of the whole list of albums referenced and the one of the user library.
For the first one, the problem was to avoid to download at every start of the program the more than 120.000 volumes (and their informations) that the website counts, because of the time it takes to do it. We solved this problem by only downloading the new volumes that was added on the website since the last time we launched the program (thanks to a comparison between the identifiers - a number which guarantee the uniqueness of a volume in a base - in the two bases).
(For example : if the biggest identifier for a volume in the embedded database is "120",

we will download every volumes in the website's database with an identifier superior to 120).
The second one was harder to handle, but one of the previous groups greatly simplified the task. The problem was to know what to do if there are differences between the libraries of the user in the program and in the website. These students prepared a table with every cases which could arrive and we used it to write the functions for the synchronization. We just added some cases they didn't think about (for the table, cf annexes).
After that work, it's now possible :

- To download the list of the new volumes available on the website and visualizing the progression of the update.
- To show the conflicts existing in the two libraries and asking to the user to solve them : he can now decide which informations he want to keep.

Conclusion

As we see, working on a project which was began and continued by a lot of students is not a simple thing. That fact justified the time we spent to analyse and understand the architecture and programming of this software. Thanks to this analysis, we have been able to improved some features (like the search engine) and added new one (communication with the website).
Unfortunately, we hadn't got enough time to put finishing touches on our work. For example, we wanted to make more tests on our new features, in order to optimize them or to make them more stable.
Furthermore, we can see that some features are still not implemented : for example the user can't manually add a comic book which is not referenced in the website and the graphic interface could be more user-friendly.

Appendices

Annexe A

Synchronisation

Le tableau ci-dessous indique les opérations à faire lors de conflits entre la base de données locales (celle du logiciel) et la base de données distantes (celle du site)

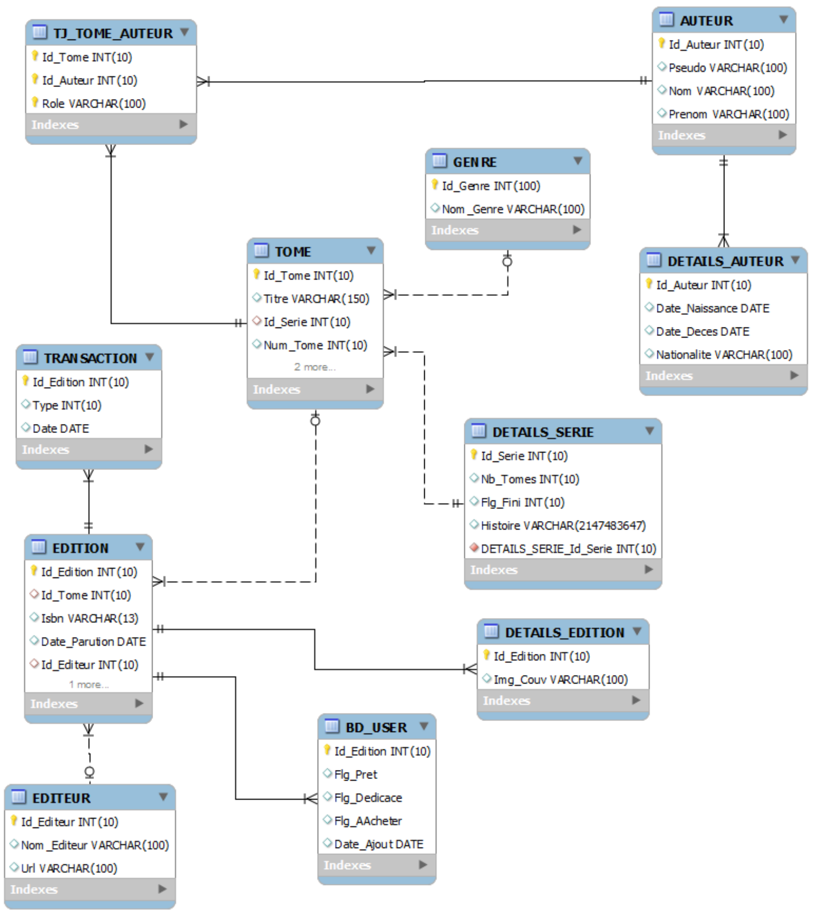
Base Locale	Base Distante	Flags	Table transaction	Décision
Présent	Présent	Identiques	Ajout	Ne rien faire
			Ajout	
			Modification	
			Suppression	Cas impossible
		Différents	Rien	Demander à l'utilisateur
			Ajout	
			Modification	
			Suppression	Cas impossible
	Absent	X	Rien	Supprimer local
			Ajout	Ajouter site
			Modification	Demander à l'utilisateur
			Suppression	Cas impossible
Absent	Présent	X	Rien	Ajouter local
			Ajout	Cas impossible
			Modification	
			Suppression	Supprimer site
	Absent	X	Rien	Rien à faire
			Ajout	Cas impossible
			Modification	
			Suppression	Rien à faire

Annexe B

Base de données

Un script SQL permettant de générer la base de données et ses indexes pour les recherches via Lucene est disponible dans le répertoire db/scripts. Lancer le h2.jar en tant qu'exécutable permet d'exécuter ce script et de créer une base exploitable par le programme.

Schéma relationnel de la base de données embarquée :



Annexe C

Le service web

Le service web est le programme permettant de récupérer les données de la base de données du site grâce aux fonctions décrites dans son fichier server.wsdl.
Les fonctions sont implémentées et commentées dans le fichier BDOvore.class.php. Le dossier "Data" contient les classes PHP correspondant aux types complexes définis dans le WSDL. L'ensemble des fichiers du webservice doit être placé dans un dossier du serveur web contenant la base de données du site (l'adresse actuellement définie pour le fichier server.php est la suivante : http ://localhost/bdovore/webservice/server.php).
Au niveau du client, les classes pour se connecter et utiliser le webservice se trouvent dans le package wsdl.server. En cas de mise à jour du wsdl, il est possible de regénérer ces classes automatiquement (la version J2EE d'Eclipse contient de base toutes les fonctionnalités pour le faire).

Annexe D

Environnement de travail

Afin de faciliter la mise en place d'un environnement de travail pour les futurs développeurs, voici un petit récapitulatif des différents composants à installer : :

- Un serveur web Apache supportant le PHP et une base de données MySQL (en d'autres termes : WAMP/LAMPP/MAMP). Au niveau de la configuration, ne pas oublier d'ajouter l'extension PHP "soap" (SOAP étant le protocole utilisé par le webservice pour effectuer ses requêtes).
- Effectuer un dump de la copie de la base de données du site. Effectuer les éventuels changements des paramètres de connexion à la base dans le webservice (server.php -> fonction getMySQLConnection()).
- Copier les fichiers du webservice sur le serveur local (à l'adresse vue plus haut). En cas de changement d'adresse, modifier le fichier WSDL (au niveau de la balise <soap :address>) et regénérer les classes Java permettant la communication avec le webservice.
- Générer la base de données embarquée avec H2 (cf Annexe B).

Abstract

BDovore is a website focused on comic books that allows collectors to manage their library. For several years, students in computer science have been developing a software in cooperation with that web site. The principal purpose of this software is to give the same features that the website, but without the obligation of being connected to the Internet. By this fact, the user will be able to handle his library both with the website and the program.
This report is about the work done by students on this software, in 2012. After a review of the state of the software before their contributions, the present report will focus on these one : first, the reinforcing of some functionalities of the program and then, the synchronization feature (aka the mechanisms to unified the library of the user on the website and in the software).

Thanks

For their help, their advises and for having provided this subject, we would like to thank our tutors : Mr BOUQUET et Mr COQBLIN.