Branch: master maktaba / README.md Find file Copy path

Abrahalhabachi Update README.md df1c9bf now

1 contributor

122 lines (97 sloc) 4.44 KB Raw Blame History

OpenLibrary

FHE Java 2 Project

OpenLibrary is a basic library management software, that allows users to search the catalogue, and the staff to add books to the library, and do borrowing/returning operations for customers. OpenLibrary is written in java, and requires the following softwares/plug-ins to function optimally:

- Apache Tomcat Server v9.0
- Java jre v1.8.0_181
- MariaDB v10.1.31
- Mysql-connector v8.0.15
- Developed on Eclipse IDE v2018-12 (4.10.0)

In the Code as well as in the Documentation, there is a distinction between Book and BookItem, with the first being an abstract kind, and the latter being the physical copy of the book. Please note that the word abstract here has nothing to do with an abstract class.

Basic use:

As a USER:

```
- Register
- Login
- Search the catalogue
- View borrowed items
```

As an ADMIN:

```
 - Login
 - Search the catalogue
 - Borrow a book (for a USER)
 - Return a book (for a USER)
 - Add new books and bookitems to the library
```

Structure:

Java Ressources:

- Package: account, Classes: User.
- Package: bookstore, Classes: Book, BookItem
- Package: helper, Classes: Database, BookDAO, UserDAO

Web Content:

- borrow.jsp
- index.jsp
- items.jsp
- login.jsp
- logout.jsp
- manage.jsp
- record.jsp
- return.jsp
- search.jsp
- signup.jsp
- api/search.jsp
- views/book_form.jsp
- views/borrow_form.jsp
- views/nav_bar.jsp
- views/return_form.jsp
- views/search_bar.jsp

Classes:

We wanted to keep things as simple as possible in this version 1.0, that's why we kept the classes and their characteristics to the minimum. For example, for a USER to register, they only need an Email address. Of course, a name, an address and maybe banking details are important in a real life scenario, but they can be easily added in a later version.

- account.User
- bookstore.Book
- bookstore.BookItem
- helper.Database
- helper.BookDAO
- helper.UserDAO

Informations to the attributes and methods of each class are in the source code.

Database:

The Database is also kept as simple as possible, the tables are as following:

- users: uid (int PK) | email (varchar255 U)| password (varchar255) | is_admin (bool)
- books: id (int PK) | title (varchar255) | title (varchar255) | ISBN (varchar20)
- bookitems: biid (int PK) | available (bool) | book_id (int FK)
- borrowings:brid (int PK) | biid (int FK) | uid (int FK) | brdate (timestamp) | archived (bool)

Although ISBN is a 13 digits number, we thought a string could be more suitable, because one could want to store the dashes (e.g: 971-23...)

A bookitem is a physical copy of a book, it needs a book id (book_id) as foreign key, it can also be available or not.

A borrowing needs a user and a bookitem, so it has 2 foreign keys. Archived is set to 1 if the user returns the bookitem.

Inserting bookitems:

Because usually there are more than 1 bookitem inserted at a time, we thought a stored procedure would be better than multiple queries, for that we use the following procedure:

```
PROCEDURE `insertbooks`(
        IN `vtitle` VARCHAR(255),
        IN `vauthor` VARCHAR(255),
        IN `visbn` VARCHAR(20),
        IN `vcopies` INT)
BEGIN
DECLARE i int DEFAULT 0;
INSERT INTO books (`id`, `author`, `ISBN`, `title`) VALUES (NULL,
vauthor, visbn, vtitle)
ON DUPLICATE KEY UPDATE id = id;
WHILE i < vcopies DO
INSERT INTO bookitems (`biid`, `available`, `book_id`)
VALUES(NULL, true, (SELECT `id` FROM books WHERE ISBN = visbn));
SET i = i + 1;
END WHILE;
END
```

Book availability:

Because ORDER BY followed by GROUP BY doesn't work in SQL, we had to find a way to get the availability of a book (meaning, at least one physical copy is available). To do so we used the MAX command, which gets the value 1 if at least one bookitem is available, and then we set the availability depending on the new MAX(bookitems.available) column.

```
SELECT * FROM `books` JOIN ( SELECT bookitems.*,
MAX(bookitems.available)
FROM bookitems GROUP BY bookitems.book_id )
as t ON `books`.id = t.book_id where books.title LIKE 'cat'
```

JSON

We wanted to make the catalogue available for web and mobile apps, so we have an interface to get Json formatted responses. This takes place for GET requests to /api/search.jsp with the parameters keyword and search_by

```
/api/search.jsp?keyword=dog&search_by=title
```