Step 20 – JDBC – BookRegister

This task is based on your previous BookRegister task in Session 17.

Source code can be found in below url:

https://github.com/kristiania-pgr112-bergen/2022/tree/main/code/Solutions/step17/src

We will create two tables booksDb.books, and booksDb.authors by elow commands

```
private void createTable() throws SQLException {
    try (Connection con = DriverManager

.getConnection("jdbc:mysql://localhost:3306/booksDb?allowPublicKeyRetrieval
" +
                    "=true&useSSL=false", "user1", "pass")) {
        Statement stmt = con.createStatement();
        String createTable = "CREATE TABLE IF NOT EXISTS books( "
                + "id INT NOT NULL AUTO_INCREMENT, "
                + "title VARCHAR(45), "
                + "authorId INT, "
                + "pages  INT(11), "
                + "genre  VARCHAR(45), "
                + "primary key (id), "
                + "foreign key (authorId) references authors(id)) ";
        boolean result = stmt.execute(createTable);
        createTable = "CREATE TABLE IF NOT EXISTS authors( "
                + "id INT NOT NULL AUTO_INCREMENT, "
                + "name VARCHAR(45), "
                + "nationality VARCHAR(45), "
                + "primary key (id)) ";
        result = stmt.execute(createTable);
    }catch (SQLException sqlException) {
        sqlException.printStackTrace();
    }

}
```

Note that books table has authorId, which is foreign key (authorId) referemces authors(id).

In order to get author information (author name and author nationality), we need to join authors and books tables. Below is the code example:

```
public ArrayList<Book> getBooks() {
    ArrayList<Book> books = new ArrayList<>();
    try (Connection con = DriverManager

.getConnection("jdbc:mysql://localhost:3306/booksDb?allowPublicKeyRetrieval
" +
                    "=true&useSSL=false", "user1", "pass")) {
        Statement stmt = con.createStatement();
        String joinSql = "SELECT book.title as title, book.pages as pages,
book.genre as genre, author.name as authorName, author.nationality as
authorNationality " +
                "FROM books book " +
                "INNER JOIN authors author " +
                "ON book.authorId = author.id";
        ResultSet rs = stmt.executeQuery(joinSql);
        while (rs.next()) {
            Book book = new Book();
            book.setTitle(rs.getString("title"));
            book.setNumberOfPages(rs.getInt("pages"));
            book.setGenre(Genre.valueOf(rs.getString("genre")));
            Author author = new Author();
            author.setName(rs.getString("authorName"));
            author.setNationality(rs.getString("authorNationality"));
            book.setAuthor(author);
            books.add(book);
        }
        return books;

    }catch (SQLException sqlException) {
        sqlException.printStackTrace();
    }
    return null;
}
```

When we design Book class, it contains below attributes.

```
private String title;
private Author author;
private int numberOfPages = 1;
private Genre genre;
```

OK, we can start!

The goals for this step:

- Protect your project against SQL injections, by using PreparedStatements
- Handle transaction, commitment and rollback

Task 1 – Design below functions using PreparedStatement:

1. registerBook(Book book) . When you register a new book, you only have author name and author nationality, while have no idea about authorId since authorId is author.id in authors table. So you only register books(title, pages, genre).

2. registerAuthor(Author author)

3. updateBook(Book book, Integer authorId) . This function is called after you find out authorId, you can update book record in bookd table.

4. ArrayList<Book> getBooks_byJoin(). Use inner join t o join authors and books. Populate the resultSet to ArrayList<Book>.

5. getIdbyAuthor(string authorName). This function is used to query authorId by author name in authors table.

Check if by using PreparedStatement we can efficiently prevent SQL injection

Task 2 - Handle transaction, commitment and rollback

1. addBook(Book book). First, you setAutoCommit(false). When you add a new book to books table, check if author exists in authors table. If it exists, then you can commit the transaction. Otherwise, output some error message and rollback.

   Note that you can use previous implemented functions such as getIdbyAuthor(String authorName) to get authorId.

2. addBookandAuthor(Book book). When you add a new book to book table, check if author exists in authors table. If it does not exist, you must register the author first, then get authorId from authors table, then update corresponding book record with new retrieved authorId in books table.

   Note that you can use previous implemented functions such as getIdbyAuthor(String authorName) to get authorId. registerAuthor(Author author) can be used to register a new author. And updateBook(Book book, String authorId) can be used to update book record.

   Have fun!