About setting the retrieval table ( use the login id to call the borrowed books under the id )

April 10, 2023

8:42 AM

* 1. First, we need to create a new method listBook in the UserDAO class , which receives a parameter userId and returns a list containing the books borrowed by a specific user. To do this, we need to execute a SQL query to filter out the books associated with a user ID . Here is an example implementation of the listBook method in the UserDAO class:

From < [*https://chat.openai.com/chat?model=gpt-4*](https://chat.openai.com/chat?model=gpt-4) >

public static List<BookAndBorrow> listBook(String userId) {

// Instantiate an empty BookAndBorrow list

List<BookAndBorrow> bookAndBorrows = new ArrayList<>();

// Execute a SQL query to get the books associated with a specific user

// Here, we use the JOIN operation to associate the books table with the borrows table

String sql = "SELECT b.isn, b.title, b.author, bo.borrow\_date FROM books b JOIN borrows bo ON b.isn = bo.isn WHERE bo.user\_id = ?";

try (Connection connection = DbUtil. getCon();

PreparedStatement preparedStatement = connection. prepareStatement(sql)) {

/ / Set the parameter value in the prepared statement

preparedStatement.setString(1, userId);

// Execute the query and process the result set

ResultSet resultSet = preparedStatement. executeQuery();

while (resultSet. next()) {

BookAndBorrow bookAndBorrow = new BookAndBorrow();

bookAndBorrow.setIsn(resultSet.getString("isn"));

bookAndBorrow.setTitle(resultSet.getString("title"));

bookAndBorrow.setAuthor(resultSet.getString("author"));

bookAndBorrow.setBorrow\_date(resultSet.getDate("borrow\_date"));

bookAndBorrows. add(bookAndBorrow);

}

} catch (Exception e) {

e.printStackTrace();

}

return bookAndBorrows;

}

* 1. Next, we need to add a userId parameter to the fillTable method in the MyBooks class so that this parameter can be passed when calling the UserDAO.listBook method. Here is a modified implementation of the fillTable method:

public void fillTable( String userId ) {

DefaultTableModel dtm = (DefaultTableModel) table. getModel();

dtm.setRowCount(0);

List<BookAndBorrow> bookAndBorrows = UserDAO.listBook(userId); // Input userId parameter

for (BookAndBorrow bookAndBorrow : bookAndBorrows) {

Vector<Object> vector = new Vector<>();

vector.add(bookAndBorrow.getIsn());

vector.add(bookAndBorrow.getTitle());

vector.add(bookAndBorrow.getAuthor());

vector.add(bookAndBorrow.getBorrow\_date());

dtm.addRow(vector);

}

}

* 1. In order to pass the userId from the Login class to the MainFrm and MyBooks classes, we need to make some modifications to these classes. First, in the Login class, we need to save the userId into the LoginConfig class so that it can be accessed in other classes:

// Add the following code in the LoginActionPerformed method

LoginConfig.save(new User(userId, pwd, user\_type));

Next, in the MainFrm class, we need to modify the actionPerformed method of the "My books" button to pass the userId when creating the MyBooks instance :

mntmNewMenuItem. addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String userId = LoginConfig.getCurrentUser().getUserId();

MyBooks userBooks = new MyBooks(userId); // Input userId parameter

userBooks. setVisible(true);

table. add(userBooks);

}

});

* 1. Finally, we need to receive the userId parameter in the MyBooks class and pass it to the fillTable method. This requires modifying the constructor of the MyBooks class to accept a userId parameter, and save it as an instance variable:

public class MyBooks extends JInternalFrame {

// ... other code ...

private String userId; // add userId instance variable

public MyBooks(String userId) {

// ... other code ...

this.userId = userId; // save userId to instance variable

fillTable(userId); // Call the fillTable method to fill the table

}

// ... other code ...

}