Лабораторная работа №6  
«Взаимодействие Java с базами данных»

*Лабораторная работа №5 выдаётся студентам в одном варианте. Срок выполнения – 2 недели. Форма представления отчёта – программа, написанная на языке Java (код и скрины примера работы программы).*

**Цель**: Получить навыки создания методов классов Java выполняющих функционал отправки запросов в базу данных для редактирования таблиц.

**Задание**: Реализовать для приложения, разработанного в рамках лабораторной работы №5 функционал для кнопок таким образом, чтобы процессы, исполняемые внутри приложения, применялись к подключенной к приложению базе данных.

В предыдущей лабораторной работа вами была написана программа, реализующая заполнение списка задач с помощью ввода в текстовом поле и подтверждения в виде addButton, отображения введенных задач в интерфейсе контейнера todos, а также, в рамках самостоятельного изучения, была реализована кнопка удаления removeButton, которая удаляет отображенную задачу из списка вместе с удалением соответствующего ему элемента интерфейса.

Однако, если мы хотим, чтобы приложение при закрытии запоминало наш созданный список задач, необходимо где-то хранить наши внесенные данные. Для решения этой задачи, в рамках лабораторной работы, мы научимся работать с содержимым базы данных.

Для работы с базой данных, в Java используется модуль java.sql . В проекте подключим его как java.sql.\*;

В первую очередь, создадим метод, который будет создавать в нашей базе данных таблицу, для хранения записанных пользователем задач. Для создания воспользуемся реализацией запроса SQL:

CREATE TABLE IF NOT EXISTS tasks id integer PRIMARY KEY, description text NOT NULL

Для того, чтобы передать этот запрос в базу данных, сначала зададим запрос в качестве строки, которую в последствии поместим в объект из java.sql типа Statement. Объект типа Statement используется для реализации записанных в них SQL запросов. Реализуем все это в методе createTable() с учетом возможных ошибок:

private static void createTable() {  
 String sql = "CREATE TABLE IF NOT EXISTS tasks (\n"  
 + " id integer PRIMARY KEY,\n"  
 + " description text NOT NULL\n"  
 + ");";  
  
 try (Statement stmt = *conn*.createStatement()) {  
 stmt.execute(sql);  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
}

Следующим этапом реализуем метод addTask, который будет передавать запрос в базу данных каждый раз, когда пользователь будет использовать addButton. В сам же метод, будем передавать значение description в качестве строки, которое будет формироваться из данных введенных в поле, для ввода текста новой задачи textField. В результате, с учетом возврата всех возможных ошибок, пойманных в try catch, получится следующий код:

private static void addTask(String description) {  
 String sql = "INSERT INTO tasks (description) VALUES (?)";  
  
 try (PreparedStatement stmt = *conn*.prepareStatement(sql, Statement.*RETURN\_GENERATED\_KEYS*)) {  
 stmt.setString(1, description);  
 int affectedRows = stmt.executeUpdate();  
  
 if (affectedRows == 0) {  
 throw new SQLException("Не удалось добавить задачу");  
 }  
  
 try (ResultSet generatedKeys = stmt.getGeneratedKeys()) {  
 if (generatedKeys.next()) {  
 *renderTask*(generatedKeys.getLong(1), description);  
 } else {  
 throw new SQLException("Не удалось получить id");  
 }  
 }  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 }  
}

Для отображения данных из базы необходимо задать запрос, который будет брать все эти данные и выводить их в интерфейсе приложения. Чтобы отобразить все занесенные строки с кнопками удаления, необходимо реализовать renderTask(), опираясь на механизм создания кнопки с подписью из предыдущей лабораторной работы.

Для этого реализуем метод loadTasks() с перебором по ключам из таблицы:

**private** **static** **void** loadTasks() {

String sql = "SELECT \* FROM tasks";

**try** (PreparedStatement stmt = conn.prepareStatement(sql)){

ResultSet rs = stmt.executeQuery();

**while** (rs.next()) {

renderTask(rs.getLong("id"), rs.getString("description"));

}

} **catch** (SQLException e) {

System.out.println(e.getMessage());

}

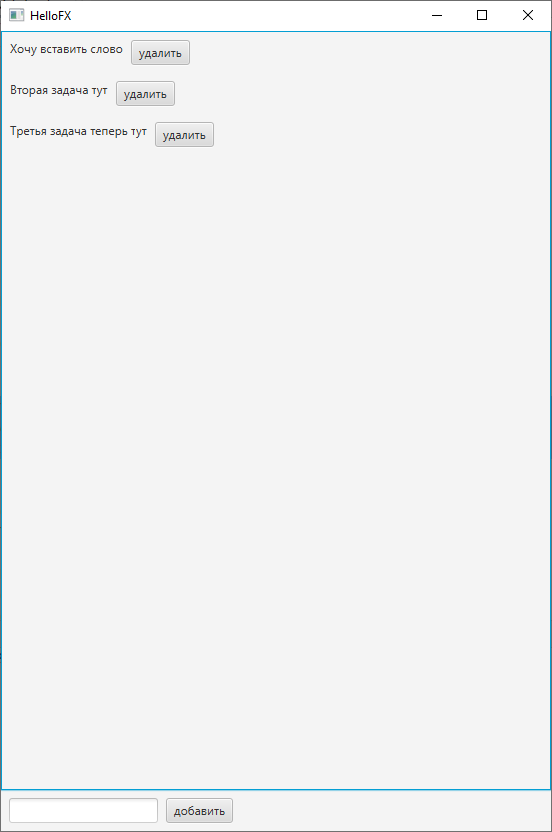
}

Для того чтобы мы смогли выбрать нужную нам базу данных для работы подключим еще java.nio.file.Path и java.nio.file.Paths. Настроим теперь чтобы в основной функции, при запуске программы, у нас подключалась (или если отсутствует, то создавалась) база данных и там была определена таблица с задачами внесенными через интерфейс приложения:

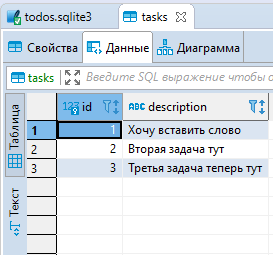
public static void main(String[] args) {  
 try {  
 Path currentPath = Paths.*get*(System.*getProperty*("user.dir"));  
 Path filePath = Paths.*get*(currentPath.toString(), "todos.sqlite3");  
  
 String url = "jdbc:sqlite:" + filePath;  
 *conn* = DriverManager.*getConnection*(url);  
  
 *createTable*();  
 *loadTasks*();  
  
 Application.*launch*(args);  
 } catch (SQLException e) {  
 System.*out*.println(e.getMessage());  
 } finally {  
 try {  
 if (*conn* != null) {  
 *conn*.close();  
 }  
 } catch (SQLException ex) {  
 System.*out*.println(ex.getMessage());  
 }  
 }  
}

А для кнопки addButton добавим событие при нажатии:  
addButton.setOnAction((event) -> {  
 *addTask*(textField.getText());  
 textField.setText("");  
});

Для того чтобы убедиться в том, что программа работает корректно, заполним через интерфейс несколько задач, которые в последствии будут добавлены в базу todos:



Для того, чтобы проверить содержимое базы данных, можно использовать любое из приложений, однако, в рамках примера, для просмотра данных будет представлен скриншот из программы для просмотра содержимого таблиц баз данных, приложение DBeaver (<https://dbeaver.io/>):



В рамках самостоятельной работы, реализуйте для приложения, сделанного в рамках самостоятельной работы лабораторной работы №5 событие при нажатии кнопки удаления задач таким образом, чтобы в базе данных также удалялась строка с содержимым этой задачи.