МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

НГТУ

Лабораторная работа №7

по предмету «Технология программирования»

на тему «Организация в Java хранения данных в базах данных»

Факультет: АВТ Преподаватель:

Группа: АВТ-809 Михайленко Д.А.

Студент: Швыдко Е.А.

Новосибирск 2020

Цели работы:

1. Изучить особенности реализации хранения данных в базе данных в приложениях Java.
2. Доработать программу, созданную в лабораторной работе № 5:

* Добавить в меню пункты сохранение и загрузку объектов в базу данных. Можно использовать любую СУБД.
* Загрузку и сохранение объектов только заданного типа.

Ход разработки:

* В меню были добавлены пункты работы с базой данных.

JMenuItem menuShowTime = new JMenuItem("Показать время");

JMenuItem showConsole = new JMenuItem("Показать консоль");

JMenuItem saveItem = new JMenuItem("Сохранить объекты");

JMenuItem loadItem = new JMenuItem("�-агрузить объекты");

JMenuItem inputDB = new JMenuItem("�'ыгрузить в базу данных все объекты");

JMenuItem inputCatDB = new JMenuItem("�'ыгрузить в базу данных кошек");

JMenuItem inputDogDB = new JMenuItem("�'ыгрузить в базу данных собак");

JMenuItem outputDB = new JMenuItem("�-агрузить из базы данных все объекты");

JMenuItem outputCatDB = new JMenuItem("�-агрузить из базы данных кошек");

JMenuItem outputDogDB = new JMenuItem("�-агрузить из базы данных собак");

mainMenu.add(menuStart);

mainMenu.add(menuStop);

mainMenu.add(menuShowTime);

mainMenu.add(showConsole);

mainMenu.add(saveItem);

mainMenu.add(loadItem);

mainMenu.add(inputDB);

mainMenu.add(inputCatDB);

mainMenu.add(inputDogDB);

mainMenu.add(outputDB);

mainMenu.add(outputCatDB);

mainMenu.add(outputDogDB);

Bar.add(mainMenu);

setJMenuBar(Bar);

inputDB.addActionListener(new ActionListener() {

**@Override**

**public** void actionPerformed(ActionEvent e) {

InputDB();

}

});

inputCatDB.addActionListener(new ActionListener() {

**@Override**

**public** void actionPerformed(ActionEvent e) {

InputDBCats();

}

});

inputDogDB.addActionListener(new ActionListener() {

**@Override**

**public** void actionPerformed(ActionEvent e) {

InputDBDogs();

}

});

outputDB.addActionListener(new ActionListener() {

**@Override**

**public** void actionPerformed(ActionEvent e) {

**try** {

OutputFullDB();

} **catch** (**SQLException** ex) {

ex.printStackTrace();

}

}

});

outputCatDB.addActionListener(new ActionListener() {

**@Override**

**public** void actionPerformed(ActionEvent e) {

**try** {

OutputDB(**true**);

} **catch** (**SQLException** ex) {

ex.printStackTrace();

}

}

});

outputDogDB.addActionListener(new ActionListener() {

**@Override**

**public** void actionPerformed(ActionEvent e) {

**try** {

OutputDB(**false**);

} **catch** (**SQLException** ex) {

ex.printStackTrace();

}

}

});

* Были реализованы методы для загрузки выгрузки объектов в коллекции:

**public** void InputDB() {

DB dbHandler = **new** DB();

**for**(int i = 0; i < Singleton.getVector().getArray().size(); i++) {

dbHandler.addObj(Singleton.getVector().getArray().get(i));

}

}

**public** void InputDBCats() {

DB dbHandler = **new** DB();

**for**(int i = 0; i < Singleton.getVector().getArray().size(); i++) {

**if**(Singleton.getVector().getArray().get(i) **instanceof** Cat)

dbHandler.addCat(Singleton.getVector().getArray().get(i));

}

}

**public** void InputDBDogs() {

DB dbHandler = **new** DB();

**for**(int i = 0; i < Singleton.getVector().getArray().size(); i++) {

**if**(Singleton.getVector().getArray().get(i) **instanceof** Dog)

dbHandler.addDog(Singleton.getVector().getArray().get(i));

}

}

**public** void OutputDB(boolean isCat) **throws** SQLException {

DB dbHandler = **new** DB();

**ResultSet** result = dbHandler.getObj(isCat);

//int columns = result.getMetaData().getColumnCount();

**while**(result.next()) {

**if**(result.getString(4).equals("cat"))

Singleton.getVector().getArray().add(**new** Cat(**Integer**.parseInt(result.getString(1)), **Integer**.parseInt(result.getString(2)),

**Integer**.parseInt(result.getString(3))));

**else** **if**(result.getString(4).equals("dog"))

Singleton.getVector().getArray().add(**new** Dog(**Integer**.parseInt(result.getString(1)), **Integer**.parseInt(result.getString(2)),

**Integer**.parseInt(result.getString(3))));

}

habitat.repaint();

}

**public** void OutputFullDB() **throws** SQLException {

DB dbHandler = **new** DB();

**ResultSet** result = dbHandler.getObjs();

//int columns = result.getMetaData().getColumnCount();

**while**(result.next()) {

**if**(result.getString(4).equals("cat"))

Singleton.getVector().getArray().add(**new** Cat(**Integer**.parseInt(result.getString(1)), **Integer**.parseInt(result.getString(2)),

**Integer**.parseInt(result.getString(3))));

**else** **if**(result.getString(4).equals("dog"))

Singleton.getVector().getArray().add(**new** Dog(**Integer**.parseInt(result.getString(1)), **Integer**.parseInt(result.getString(2)),

**Integer**.parseInt(result.getString(3))));

}

habitat.repaint();

}

* Был реализован класс DB, реализующий функции обработчика базы данных, в котором были разработаны методы для обращения к базе данных и для сохранения и загрузки объектов заданного типа или всех объектов из базы данных.

**public** **class** DB{

**Connection** dbConnection;

**public** **Connection** getDbConnection() **throws** **ClassNotFoundException**, **SQLException**{

**String** connectionString = "jdbc:mysql://localhost/world?useUnicode=true&serverTimezone=UTC&useSSL=true&verifyServerCertificate=false";

**Class**.forName("com.mysql.cj.jdbc.Driver");

dbConnection = **DriverManager**.getConnection(connectionString, "root", "123456789");

**return** dbConnection;

}

**public** void addObj(Animal animal) {

**String** insert = "INSERT INTO objs (X,Y,ID,type) VALUES (?,?,?,?)";

**String** type = **null**;

**if**(animal **instanceof** Cat) type = "cat";

**else** **if**(animal **instanceof** Dog) type = "dog";

**PreparedStatement** prST = **null**;

**try** {

prST = getDbConnection().prepareStatement(insert);

prST.setString(1, **String**.valueOf(animal.getX()));

prST.setString(2, **String**.valueOf(animal.getY()));

prST.setString(3, **String**.valueOf(animal.getID()));

prST.setString(4, type);

prST.executeUpdate();

} **catch** (**SQLException** e) {

e.printStackTrace();

} **catch** (**ClassNotFoundException** e) {

e.printStackTrace();

}

}

**public** void addCat(Animal cat) {

**String** insert = "INSERT INTO objs (X,Y,ID,type) VALUES (?,?,?,?)";

**PreparedStatement** prST = **null**;

**try** {

prST = getDbConnection().prepareStatement(insert);

prST.setString(1, **String**.valueOf(cat.getX()));

prST.setString(2, **String**.valueOf(cat.getY()));

prST.setString(3, **String**.valueOf(cat.getID()));

prST.setString(4, "cat");

prST.executeUpdate();

} **catch** (**SQLException** e) {

e.printStackTrace();

} **catch** (**ClassNotFoundException** e) {

e.printStackTrace();

}

}

**public** void addDog(Animal dog) {

**String** insert = "INSERT INTO objs (X,Y,ID,type) VALUES (?,?,?,?)";

**PreparedStatement** prST = **null**;

**try** {

prST = getDbConnection().prepareStatement(insert);

prST.setString(1, **String**.valueOf(dog.getX()));

prST.setString(2, **String**.valueOf(dog.getY()));

prST.setString(3, **String**.valueOf(dog.getID()));

prST.setString(4, "dog");

prST.executeUpdate();

} **catch** (**SQLException** e) {

e.printStackTrace();

} **catch** (**ClassNotFoundException** e) {

e.printStackTrace();

}

}

**public** **ResultSet** getObj(boolean isCat) {

**ResultSet** resSet = **null**;

**String** select = "SELECT \* FROM objs WHERE type=?";

**PreparedStatement** prST = **null**;

**try** {

prST = getDbConnection().prepareStatement(select);

prST.setString(1, isCat ? "cat" : "dog");

resSet = prST.executeQuery();

} **catch** (**SQLException** e) {

e.printStackTrace();

} **catch** (**ClassNotFoundException** e) {

e.printStackTrace();

}

**return** resSet;

}

**public** **ResultSet** getObjs() {

**ResultSet** resSet = **null**;

**String** select = "SELECT \* FROM objs";

**PreparedStatement** prST = **null**;

**try** {

prST = getDbConnection().prepareStatement(select);

resSet = prST.executeQuery();

} **catch** (**SQLException** e) {

e.printStackTrace();

} **catch** (**ClassNotFoundException** e) {

e.printStackTrace();

}

**return** resSet;

}

}

Выводы:

В результате выполнения работы были изучены особенности реализации хранения данных в базе данных в приложениях, написанных на Java.