НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Р.Е.АЛЕКСЕЕВА
Институт радиоэлектроники и информационных технологий
Кафедра “Прикладная математика”

Лабораторная работа №3 по курсу «Базы данных»
«Java ORM »

Выполнила:
Горенкова А. В.
Группа 16-ПМ
Проверил:
Моисеев А.Е.

НИЖНИЙ НОВГОРОД
2018 г.

Оглавление

Введение

ORM (Object-Relational Mapping, объектно-реляционное отображение) - технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, т.е. ORM — это прослойка между базой данных и кодом который пишет программист, которая позволяет взаимодействовать созданным в программе объектам с базой данных. При этом отпадает необходимость писать SQL-запросы самостоятельно.

Говоря конкретнее, использование ORM решает проблему так называемого «несоответствия», которая состоит в том, что объектные и реляционные модели не очень хорошо работают вместе. Реляционные базы представляют данные в табличном формате, в то время как объектно-ориентированные языки представляют их как связанный граф объектов. ORM как бы создает «виртуальную» схему базы данных в памяти и позволяет манипулировать данными уже на уровне объектов. Каждый объект и его свойства связанны с одной или несколькими таблицами и их полями в базе данных. ORM использует эту информацию для управления процессом преобразования данных между базой и формами объектов, а также для создания SQL-запросов для вставки, обновления и удаления данных.

Основным плюсом использования ORM в проекте является избавление разработчика от необходимости работы с SQL и написания большого количества кода. Весь генерируемый ORM код предположительно хорошо проверен и оптимизирован. Однако при этом появляется и минус — это потеря производительности. Это

происходит потому, что большинство ORM предназначены для обработки широкого спектра сценариев использования данных, гораздо большего, чем любое отдельное приложение когда-либо сможет использовать. Однако, нельзя забывать, что то, что с легкостью пишется с использованием ORM за неделю, можно реализовывать ни один месяц собственными усилиями. Кроме того, большинство современных ORM позволяют программисту при необходимости самому задавать код SQL-запросов. Таким образом, использование в проектах ORM может значительно облегчить жизнь разработчику.

Задание

- Перевести код из Л.Р №2 на код с использованием ORM и фреймворка ORMLite.
- Сделать несколько запросов к таблицам.

Решение

1. Создаём 2 класса: Furniture (мебель) и Shop (магазин)

В классе Furniture создаём переменную shop, через которую мы свяжем таблицы Furniture и Shop по принципу «многие к одному» (задаём в аннотации параметр foreign = **true**).

```
@DatabaseField(foreign = true, foreignAutoRefresh = true, columnName = "shopId")
private Shop shop;
```

2. Импортируем:

```
import com.j256.ormlite.dao.Dao;
import com.j256.ormlite.dao.DaoManager;
import com.j256.ormlite.jdbc.JdbcConnectionSource;
import com.j256.ormlite.stmt.QueryBuilder;
import java.util.List;
```

3. Подключаемся к базе данных и создаем 2 таблицы:

```
JdbcConnectionSource connectionSource = new
JdbcConnectionSource(DATABASE_URL);
Dao<Shop, Integer> shopDao = DaoManager.createDao(connectionSource,
Shop.class);
Dao<Furniture, Integer> furDao = DaoManager.createDao(connectionSource;
TableUtils.createTable(connectionSource, Shop.class);
TableUtils.createTable(connectionSource, Furniture.class);
```

4. Заполняем таблицы мебели (Furniture) и магазинов (Shop)

```
Shop shop1 = new Shop ("IKEA", "tc Mega");
shopDao.create(shop1);

...

Furniture mmbr = new Furniture (shop1, "Table", "black", 15000);
furDao.create(mmbr);
mmbr = new Furniture (shop2, "Shelf", "brown", 1000);
furDao.create(mmbr);

...
```

Результаты работы

1. Получаем из базы данных всю мебель в магазине «AKSON»:

```
List<Furniture> arr = getFurnitureByShop("AKSON", shopDao, furDao);
```

```
AKSON, Chair, red, 1500
AKSON, Table, red, 8000
AKSON, Chair, green, 1500
AKSON, Shelf, black, 1000
AKSON, Shelf, black, 1200
```

2. Получаем из базы данных всю мебель чёрного цвета:

```
arr = getFurnitureByColor("black", furDao);
```

```
IKEA, Table, black, 15000
OBI, Chair, black, 15000
AKSON, Shelf, black, 1000
AKSON, Shelf, black, 1200
```

Листинг

Main.java:

```
package lab3;
import java.util.List;

import com.j256.ormlite.dao.Dao;
import com.j256.ormlite.dao.DaoManager;
import com.j256.ormlite.jdbc.JdbcConnectionSource;
import com.j256.ormlite.stmt.QueryBuilder;
import com.j256.ormlite.table.TableUtils;

public class main {

	private final static String DATABASE_URL = "jdbc:h2:~/fDB";

	private static Dao<Shop, Integer> shopDao;
	private static Dao<Furniture, Integer> furDao;

	public static void main(String[] args) throws Exception {
		JdbcConnectionSource connectionSource = null;
		try {
			connectionSource = new JdbcConnectionSource(DATABASE_URL);
			shopDao = DaoManager.createDao(connectionSource, Shop.class);
			furDao = DaoManager.createDao(connectionSource,
Furniture.class);

			TableUtils.createTable(connectionSource, Shop.class);
			TableUtils.createTable(connectionSource, Furniture.class);

			Shop shop1 = new Shop ("IKEA", "tc Mega");
			shopDao.create(shop1);
			Shop shop2 = new Shop ("OBI", "tc Fantastic");
			shopDao.create(shop2);
			Shop shop3 = new Shop ("AKSON", "komsomolsk shosse");
			shopDao.create(shop3);

			Furniture mmbr = new Furniture (shop1, "Table", "black",
15000);
			furDao.create(mmbr);
			mmbr = new Furniture (shop1, "Shelf", "green", 13000);
			furDao.create(mmbr);
			mmbr = new Furniture (shop1, "Table", "red", 15000);
			furDao.create(mmbr);
			mmbr = new Furniture (shop2, "Table", "blue", 1500);
			furDao.create(mmbr);
			mmbr = new Furniture (shop2, "Chair", "black", 15000);
			furDao.create(mmbr);
			mmbr = new Furniture (shop2, "Shelf", "brown", 1000);
			furDao.create(mmbr);
			mmbr = new Furniture (shop2, "Table", "red", 12000);
			furDao.create(mmbr);
			mmbr = new Furniture (shop3, "Chair", "red", 1500);
			furDao.create(mmbr);
			mmbr = new Furniture (shop3, "Table", "red", 8000);
			furDao.create(mmbr);
			mmbr = new Furniture (shop3, "Chair", "green", 1500);
			furDao.create(mmbr);
```

```
                  mmbr = new Furniture (shop3, "Shelf", "black", 1000);
                  furDao.create(mmbr);
                  mmbr = new Furniture (shop3, "Shelf", "black", 1200);
                  furDao.create(mmbr);

                  List<Furniture> arr = getFurnitureByShop("AKSON", shopDao,
furDao);
                  for (Furniture tmp : arr) {
                        String color = tmp.getColor();
                        String shop = tmp.getShop().getName();
                        String name = tmp.getName();
                        int price = tmp.getPrice();
                        System.out.println(shop+", "+name+", "+color+",
"+price);
                  }
                  arr = getFurnitureByColor("black", furDao);
                  for (Furniture tmp : arr) {
                        String color = tmp.getColor();
                        String shop = tmp.getShop().getName();
                        String name = tmp.getName();
                        int price = tmp.getPrice();
                        System.out.println(shop+", "+name+", "+color+",
"+price);
                  }


            } finally {
                  if (connectionSource != null) {
                        connectionSource.close();
                  }
            }
      }

      public static List<Furniture> getFurnitureByShop(String shop, Dao<Shop,
Integer> shopDao, Dao<Furniture, Integer> furDao)
      {
            List<Furniture> temp = null;
            try {
                  QueryBuilder<Shop, Integer> shopBuilder =
shopDao.queryBuilder();
                  shopBuilder.where().like("name", shop);
                  QueryBuilder<Furniture, Integer> furBuilder =
furDao.queryBuilder();
                  furBuilder.join("shopId", "id", shopBuilder);
                  temp = furDao.query(furBuilder.prepare());
            }
            catch (Exception e) {
            }
            return temp;
      }

      public static List<Furniture> getFurnitureByColor(String color,
Dao<Furniture, Integer> furDao)
      {
            List<Furniture> temp = null;
            try {
                  QueryBuilder<Furniture, Integer> furBuilder =
furDao.queryBuilder();
                  furBuilder.where().like("color", color);
                  temp = furDao.query(furBuilder.prepare());
            }
            catch (Exception e) {}
```

```
            return temp;
      }

}
```

Furniture.java:

```
package lab3;

import com.j256.ormlite.field.DatabaseField;
import com.j256.ormlite.table.DatabaseTable;

@DatabaseTable(tableName = "furnitures")
public class Furniture {
      @DatabaseField(generatedId = true)
      private int id;

      @DatabaseField(foreign = true, foreignAutoRefresh = true, columnName =
"shopId")
      private Shop shop;

      @DatabaseField(canBeNull = false)
    private String name;

      @DatabaseField(canBeNull = false)
    private String color;

      @DatabaseField(canBeNull = false)
    private int price;

    public Furniture() {

    }
    public Furniture(Shop _shop, String _name, String _color, int _price) {
      shop = _shop;
      color = _color;
      name = _name;
      price = _price;
    }

    public int getId() {
      return id;
    }

    public Shop getShop() {
      return shop;
    }

    public void setShop(Shop _shop) {
      shop = _shop;
    }

    public String getName() {
      return name;
    }
    public void setName(String _name) {
      name = _name;
    }
    public String getColor() {
      return color;
    }
```

```
    public void setColor(String _color) {
      color = _color;
    }
    public int getPrice() {
      return price;
    }
    public void setPrice(int _price) {
      price = _price;
    }
}
```

Shop.java:

```
package lab3;

import com.j256.ormlite.dao.ForeignCollection;
import com.j256.ormlite.field.DatabaseField;
import com.j256.ormlite.field.ForeignCollectionField;
import com.j256.ormlite.table.DatabaseTable;

@DatabaseTable(tableName = "shops")
public class Shop {
      @DatabaseField(canBeNull = false)
      private String name;
      @DatabaseField(canBeNull = false)
      private String adress;

      @DatabaseField(generatedId = true)
      private int id;

      @ForeignCollectionField
      private ForeignCollection<Furniture> f;

      public Shop() {

      }

      public Shop(String _name, String _adress) {
            name = _name;
            adress = _adress;
      }

      public int getId() {
            return id;
      }

      public ForeignCollection<Furniture> getCollection() {
            return f;
      }

      public String getName() {
            return name;
      }

      public String getAdress() {
            return adress;
      }

      public void setName(String _name) {
      name = _name;
```

```
    }

      public void setAdress(String _adress) {
      adress = _adress;
    }
}
```