# JAVA编程进阶上机报告

****

**学 院 智能与计算学部**

**专 业 软件工程**

**班 级 五班**

**学 号 3018216242**

**姓 名 邢思洋**

1. **实验要求**

1. 提供用户表：user

表中包含字段：

id，用户名，性别，邮箱，电话等信息。

2. 要求通过注解和反射的方式封装一个小型的sql操作类，可以通过对应的方法生成增、删、改、查等操作的SQL语句。

3. 要求实现注解：

@Column：用来标注每个field对应的表中的字段是什么

@Table：用来标记表的名字

1. **源代码**

**Column.java:**

**package** xsy.lab3;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Target(ElementType.***FIELD***)

@Retention(RetentionPolicy.***RUNTIME***)

**public** **@interface** Column

{

String value();

}

**Table.java:**

**package** xsy.lab3;

**import** java.lang.annotation.ElementType;

**import** java.lang.annotation.Retention;

**import** java.lang.annotation.RetentionPolicy;

**import** java.lang.annotation.Target;

@Target(ElementType.***TYPE***)

@Retention(RetentionPolicy.***RUNTIME***)

**public** **@interface** Table

{

String value();

}

**User类无get方法，只能使用反射获取成员变量的值**

**User.java:**

**package** xsy.lab3;

@Table("user")

**public** **class** User

{

@Column("id")

**private** Integer id;

@Column("username")

**private** String username;

@Column("age")

**private** Integer age;

@Column("email")

**private** String email;

@Column("telephone")

**private** String telephone;

**public** **void** setId(Integer id)

{

**this**.id = id;

}

**public** **void** setUsername(String username)

{

**this**.username = username;

}

**public** **void** setAge(Integer age)

{

**this**.age = age;

}

**public** **void** setEmail(String email)

{

**this**.email = email;

}

**public** **void** setTelephone(String telephone)

{

**this**.telephone = telephone;

}

}

**SqlUtilInterface.java:**

**package** xsy.lab3;

**import** java.util.List;

**public** **interface** SqlUtilInterface

{

/\*\*

\* 根据传入的参数返回查询语句

\* **@param** user

\* **@return** 返回查询语句

\* **@throws** Exception

\*/

String query(User user) **throws** Exception;

/\*\*

\* 根据传入的参数返回插入语句

\* **@param** user

\* **@return** 返回插入语句

\* **@throws** Exception

\*/

String insert(User user) **throws** Exception;

/\*\*

\* 根据传入的参数返回插入语句

\* **@param** users

\* **@return** 返回插入语句

\* **@throws** Exception

\*/

String insert(List<User> users) **throws** Exception;

/\*\*

\* 根据传入的参数返回删除语句（删除id为user.id的记录）

\* **@param** user

\* **@return** 返回删除语句

\* **@throws** Exception

\*/

String delete(User user) **throws** Exception;

/\*\*

\* 根据传入的参数返回更新语句（将id为user.id的记录的其它字段更新成user中的对应值）

\* **@param** user

\* **@return** 返回更新语句

\* **@throws** Exception

\*/

String update(User user) **throws** Exception;

}

**SqlUtil.java**

**package** xsy.lab3;

**import** java.lang.reflect.Field;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** SqlUtil **implements** SqlUtilInterface

{

//获取注解Table的值，即获取表名

**public** String getTableName(Class clazz)

{

**boolean** flag = clazz.isAnnotationPresent(Table.**class**);

**if** (flag)

{

Table table = (Table) clazz.getAnnotation(Table.**class**);

**return** table.value();

}

**else**

{

**return** **null**;

}

}

//获取注解Column的值，即获取表中属性名

**public** String getColumnName(Field field)

{

**boolean** flag = field.isAnnotationPresent(Column.**class**);

**if** (flag)

{

Column column = (Column) field.getAnnotation(Column.**class**);

**return** column.value();

}

**else**

{

**return** **null**;

}

}

@Override

**public** String query(User user) **throws** Exception

{

Class clazz = user.getClass();

Field[] fields = clazz.getDeclaredFields();

List<Field> list = **new** ArrayList<Field>();

**for** (Field field : fields)

{

field.setAccessible(**true**);

**if** (field.get(user) != **null**)

{

list.add(field);

}

}

String str = "SELECT \* FROM " + **this**.getTableName(clazz) + " WHERE ";

**for** (**int** i = 0; i < list.size(); i++)

{

**if** (list.get(i).get(user) **instanceof** String)

{

str += **this**.getColumnName(list.get(i)) + " LIKE '" + list.get(i).get(user) + "' AND ";

}

**else**

{

str += **this**.getColumnName(list.get(i)) + " = " + list.get(i).get(user) + " AND ";

}

}

str = str.substring(0, str.length() - 5);

**return** str;

}

@Override

**public** String insert(User user) **throws** Exception

{

Class clazz = user.getClass();

Field[] fields = clazz.getDeclaredFields();

List<Field> list = **new** ArrayList<Field>();

**for** (Field field : fields)

{

field.setAccessible(**true**);

**if** (field.get(user) != **null**)

{

list.add(field);

}

}

String str = "INSERT INTO " + **this**.getTableName(clazz) + " (";

**for** (**int** i = 0; i < list.size(); i++)

{

str += **this**.getColumnName(list.get(i)) + ", ";

}

str = str.substring(0, str.length() - 2);

str += ") VALUES (";

**for** (**int** i = 0; i < list.size(); i++)

{

**if** (list.get(i).get(user) **instanceof** String)

{

str += "'" + list.get(i).get(user) + "', ";

}

**else**

{

str += list.get(i).get(user) + ", ";

}

}

str = str.substring(0, str.length() - 2);

str += ")";

**return** str;

}

@Override

**public** String insert(List<User> users) **throws** Exception

{

List<Field> list = **new** ArrayList<Field>();

**for** (**int** i = 0; i < users.size(); i++)

{

Class clazz = users.get(i).getClass();

Field[] fields = clazz.getDeclaredFields();

**for** (Field field : fields)

{

field.setAccessible(**true**);

**if** (field.get(users.get(i)) != **null** && !list.contains(field))

{

list.add(field);

}

}

}

String str = "INSERT INTO " + **this**.getTableName(users.get(0).getClass()) + " (";

**for** (**int** i = 0; i < list.size(); i++)

{

str += **this**.getColumnName(list.get(i)) + ", ";

}

str = str.substring(0, str.length() - 2);

str += ") VALUES (";

**for** (**int** i = 0; i < users.size(); i++)

{

**for** (**int** j = 0; j < list.size(); j++)

{

**if** (list.get(j).get(users.get(i)) **instanceof** String)

{

str += "'" + list.get(j).get(users.get(i)) + "', ";

}

**else**

{

str += list.get(j).get(users.get(i)) + ", ";

}

}

str = str.substring(0, str.length() - 2);

str += ") , (";

}

str = str.substring(0, str.length() - 4);

**return** str;

}

@Override

**public** String delete(User user) **throws** Exception

{

Class clazz = user.getClass();

Field field = clazz.getDeclaredField("id");

field.setAccessible(**true**);

String str = "DELETE FROM " + **this**.getTableName(clazz) + " WHERE " +

**this**.getColumnName(field) + " = " + field.get(user);

**return** str;

}

@Override

**public** String update(User user) **throws** Exception

{

Class clazz = user.getClass();

Field idField = clazz.getDeclaredField("id");

idField.setAccessible(**true**);

Field[] fields = clazz.getDeclaredFields();

List<Field> list = **new** ArrayList<Field>();

**for** (Field field : fields)

{

field.setAccessible(**true**);

**if** (field.get(user) != **null** && !field.getName().equals("id"))

{

list.add(field);

}

}

String str = "UPDATE " + **this**.getTableName(clazz) + " SET ";

**for** (**int** i = 0; i < list.size(); i++)

{

**if** (list.get(i).get(user) **instanceof** String)

{

str += **this**.getColumnName(list.get(i)) + " = '" + list.get(i).get(user) + "', ";

}

**else**

{

str += **this**.getColumnName(list.get(i)) + " = " + list.get(i).get(user) + ", ";

}

}

str = str.substring(0, str.length() - 2);

str += " WHERE " + **this**.getColumnName(idField) + " = " + idField.get(user);

**return** str;

}

}

**Test.java:**

**package** xsy.lab3;

**import** java.util.ArrayList;

**import** java.util.List;

**public** **class** Test

{

**public** **static** **void** main(String[] args) **throws** Exception

{

SqlUtilInterface util = **new** SqlUtil();

// test query1

User user = **new** User();

user.setId(175);

System.***out***.println(util.query(user));

// print: SELECT \* FROM user WHERE id = 175

// test query2

user = **new** User();

user.setUsername("史荣贞");

System.***out***.println(util.query(user));

// print: SELECT \* FROM `user` WHERE `username` LIKE '史荣贞';

// test insert

user = **new** User();

user.setUsername("user");

user.setTelephone("12345678123");

user.setEmail("user@123.com");

user.setAge(20);

System.***out***.println(util.insert(user));

// print: INSERT INTO `user` (`username`, `telephone`, `email`, `age`) VALUES ('user', '12345678123', 'user@123.com', 20)

// test insert list

User user2 = **new** User();

user2.setUsername("user2");

user2.setTelephone("12345678121");

user2.setEmail("user2@123.com");

user2.setAge(20);

List<User> list = **new** ArrayList<>();

list.add(user);

list.add(user2);

System.***out***.println(util.insert(list));

// print: INSERT INTO `user` (`username`, `telephone`, `email`, `age`) VALUES ('user', '12345678123', 'user@123.com', 20), ('user2', '12345678121', 'user2@123.com', 20)

// test delete

user = **new** User();

user.setId(1);

System.***out***.println(util.delete(user));

// print: DELETE FROM `user` WHERE `id` = 1;

// test update

user = **new** User();

user.setId(1);

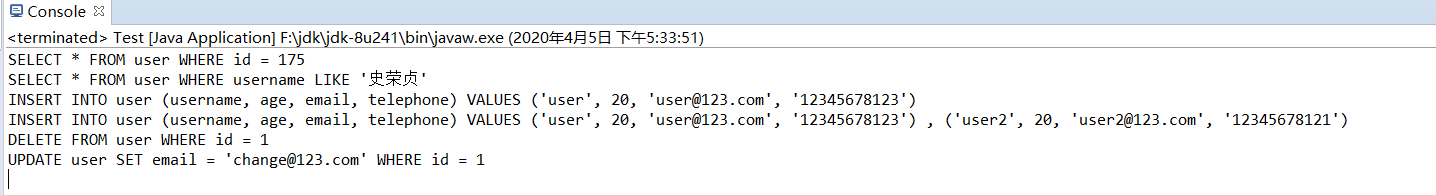
user.setEmail("change@123.com");

System.***out***.println(util.update(user));

// print: UPDATE `user` SET `email` = 'change@123.com' WHERE `id` = 1;

}

}

1. **实验结果**