JAVA 编程进阶上机报告

学　　院_智能与计算学部_____

专　　业_软件工程_____

姓　　名_王传安_____

学　　号_3018216301____

年　　级_2018 级_____

班　　级_软工六班_____

一、实验目的

使用注解和反射来完成动态 Sql 编程。

二、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

三、实验过程

1. 定义@Column 注解，用来标注每个 field 对应的表中的字段是什么：

```
@Target({ElementType.FIELD})
@Retention(RetentionPolicy.RUNTIME)
public @interface Column {
    String value();
}
```

@Target({ElementType.FIELD})表明作用范围为成员变量；

@Retention(RetentionPolicy.RUNTIME)表明注解保留时间为运行期保留，并且可以通过反射去获取注解信息。

2. 定义@Table注解，用来标记表的名字:

```
@Target({ElementType.TYPE})
@Retention(RetentionPolicy.RUNTIME)
public @interface Table {
    String value();
}
```

@Target({ElementType.TYPE})表明作用范围为类或接口;

@Retention(RetentionPolicy.RUNTIME)表明注解保留时间为运行期保留，并且可以通过反射去获取注解信息。

3. 实现接口关键部分

```
Class us = user.getClass();
    String tablename = null;
    if(us.isAnnotationPresent(Table.class)) {
        Table table = (Table) us.getAnnotation(Table.class);
        tablename = table.value();
    }
```

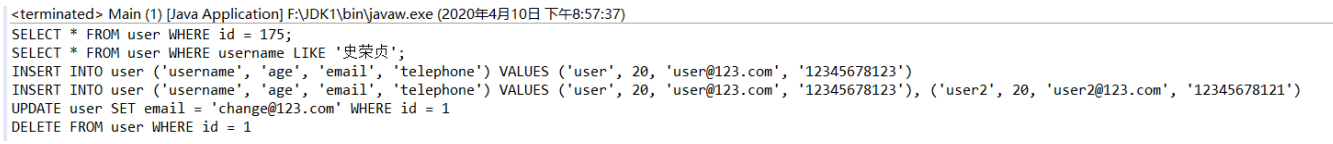
isAnnotationPresent 判断user是否含有自定义的@Table注解。

```
Field [] fields = us.getDeclaredFields();
    for(int i=0; i<fields.length; i++) {
        if(fields[i].isAnnotationPresent(Column.class)) {
            fields[i].setAccessible(true);
            try {
                    values += fields[i].get(user);
                } catch (IllegalArgumentException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IllegalAccessException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
        }
    }
```

首先判断field是否含有@Column注解，然后将其访问权限设置为true，然后通过

get(Object obj)函数得到field在obj的值

四、实验结果



五、实现接口的代码：

```
public class SqlUtil {

    /**
     * 根据传入的参数返回查询语句
     * @param user
     * @return 返回查询语句
     */
    String query(User user) {
    Class us = user.getClass();
    String tablename = null;
    int idvalue = 0;
    String namevalue = null;
    if(us.isAnnotationPresent(Table.class)) {
        Table table = (Table) us.getAnnotation(Table.class);
        tablename = table.value();
    }
    String result = "SELECT * FROM " + tablename;
    try {
            Field id = us.getDeclaredField("id");
            Field name = us.getDeclaredField("username");
            name.setAccessible(true);
            if(id.isAnnotationPresent(Column.class)) {
                id.setAccessible(true);
                idvalue = id.getInt(user);
            }
            if(name.isAnnotationPresent(Column.class)) {
```

```
                name.setAccessible(true);
                namevalue = (String) name.get(user);
            }
        } catch (NoSuchFieldException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SecurityException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalArgumentException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    if(idvalue!=0) {
        result += " WHERE id = " + String.valueOf(idvalue) + ";";
    }
    else {
        result += " WHERE username LIKE '" + namevalue + "';";
    }
        return result;
    }

    /**
     * 根据传入的参数返回插入语句
     * @param user
     * @return 返回插入语句
     */
    String insert(User user) {
    Class us = user.getClass();
    String tablename = null;
    if(us.isAnnotationPresent(Table.class)) {
        Table table = (Table) us.getAnnotation(Table.class);
        tablename = table.value();
    }
    String result = "INSERT INTO " + tablename + " ('username', 'age', 'email',
'telephone') VALUES (";
    Field [] fields = us.getDeclaredFields();
    for(int i=0; i<fields.length; i++) {
        if(fields[i].isAnnotationPresent(Column.class)) {
            fields[i].setAccessible(true);
            try {
```

```
                if(fields[i].getName().equals("username")) {
                    result += "'" + fields[i].get(user) + "'" + ", ";
                }
                if(fields[i].getName().equals("age")) {
                    result += String.valueOf(fields[i].getInt(user)) + ", ";
                }
                if(fields[i].getName().equals("email")) {
                    result += "'" + fields[i].get(user) + "'" + ", ";
                }
                if(fields[i].getName().equals("telephone")) {
                    result += "'" + fields[i].get(user) + "'" + ")";
                }
                } catch (IllegalArgumentException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IllegalAccessException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
            return result;
        }

        /**
         * 根据传入的参数返回插入语句
         * @param users
         * @return 返回插入语句
         */
        String insert(List<User> users) {
        Class temp = users.get(0).getClass();
        String tablename = null;
        if(temp.isAnnotationPresent(Table.class)) {
            Table table = (Table) temp.getAnnotation(Table.class);
            tablename = table.value();
        }
        String result = "INSERT INTO " + tablename + " ('username', 'age', 'email',
'telephone') VALUES ";
        for(User user : users) {
            Class us = user.getClass();
            Field [] fields = us.getDeclaredFields();
            for(int i=0; i<fields.length; i++) {
                if(fields[i].isAnnotationPresent(Column.class)) {
                    fields[i].setAccessible(true);
```

```
                try {
                    if(fields[i].getName().equals("username")) {
                        result += "('" + fields[i].get(user) + "'" + ", ";
                    }
                    if(fields[i].getName().equals("age")) {
                        result += String.valueOf(fields[i].getInt(user)) + ",
";
                    }
                    if(fields[i].getName().equals("email")) {
                        result += "'" + fields[i].get(user) + "'" + ", ";
                    }
                    if(fields[i].getName().equals("telephone")) {
                        result += "'" + fields[i].get(user) + "'" + ")";
                    }
                } catch (IllegalArgumentException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                } catch (IllegalAccessException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
            }
        }
        if(users.size() - 1 != users.indexOf(user)) {
            result += ", ";
        }
    }
        return result;
    }

    /**
     * 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
     */
    String delete(User user) {
    Class us = user.getClass();
    String tablename = null;
    int idvalue = 0;
    if(us.isAnnotationPresent(Table.class)) {
        Table table = (Table) us.getAnnotation(Table.class);
        tablename = table.value();
    }
    String result = "DELETE FROM " + tablename;
```

```
    try {
            Field id = us.getDeclaredField("id");
            if(id.isAnnotationPresent(Column.class)) {
                id.setAccessible(true);
                idvalue = id.getInt(user);
            }
        } catch (NoSuchFieldException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SecurityException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalArgumentException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    result += " WHERE id = " + String.valueOf(idvalue);
        return result;
    }
/**
     * 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
     */
    String update(User user) {
    Class us = user.getClass();
    String tablename = null;
    int idvalue = 0;
    String email = null;
    if(us.isAnnotationPresent(Table.class)) {
        Table table = (Table) us.getAnnotation(Table.class);
        tablename = table.value();
    }
    String result = "UPDATE " + tablename;
    try {
            Field id = us.getDeclaredField("id");
            Field name = us.getDeclaredField("email");
            name.setAccessible(true);
            if(id.isAnnotationPresent(Column.class)) {
                id.setAccessible(true);
```

```
                idvalue = id.getInt(user);
            }
            if(name.isAnnotationPresent(Column.class)) {
                name.setAccessible(true);
                email = (String) name.get(user);
            }
        } catch (NoSuchFieldException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (SecurityException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalArgumentException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    result += " SET email = '" + email + "'" + " WHERE id = " +
String.valueOf(idvalue);
        return result;
    }
}
```