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　二班

学　号　3018216084

姓　名　田家硕

一、实验要求

使用注解和反射来完成动态Sql编程。

实现对user表动态的增删改查的sql语句。

二、源代码

实现注解

```
import static java.lang.annotation.RetentionPolicy.RUNTIME;

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

```
import static java.lang.annotation.RetentionPolicy.RUNTIME;

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

注解：@Table，@Colume作用对象为类，且保留至运行时。

sql类继承了SqlUtil接口

```
package com;

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

public class Sql implements SqlUtil {
    @Override
```

```
    public String query(User user) throws ClassNotFoundException,
NoSuchMethodException, InvocationTargetException, IllegalAccessException {
        StringBuffer result = new StringBuffer();
        String CLASS_NAME = "com.User";
        Class<?> clazz = Class.forName(CLASS_NAME);
        Field[] fields = clazz.getDeclaredFields();

        boolean flag = clazz.isAnnotationPresent(Table.class);
        //获取表名
        if (flag){
            Table table = (Table)clazz.getAnnotation(Table.class);
            result.append("SELECT * FROM `" + table.tableName()+"`
WHERE");
        }
        //获取成员变量
        Set<Field> set = new HashSet<Field>();
        for (int i = 0; i < fields.length; i++) {
            boolean flag2 = fields[i].isAnnotationPresent(Column.class);
            if(flag2){
                set.add(fields[i]);
            }
        }
        //获取非空成员变量
        Map<String,String> selectColumn = new HashMap<>();
        for (Field m:set) {
            Column column = m.getAnnotation(Column.class);

 if(clazz.getMethod("get"+initCap(column.columnName())).invoke(user)!=null
){
                String content =
clazz.getMethod("get"+initCap(column.columnName())).invoke(user).toString(
);
                selectColumn.put(column.columnName(),content);
            }
        }
        for (Map.Entry<String,String> s:selectColumn.entrySet()){
            result.append("`"+s.getKey()+"` = "+"`"+s.getValue()+"`");
            result.append(" AND ");
        }
        String query = result.substring(0,result.length()-5);
```

```
        return query;
    }

    @Override
    public String insert(User user) throws ClassNotFoundException,
NoSuchMethodException, InvocationTargetException, IllegalAccessException {
        StringBuffer result = new StringBuffer();
        String CLASS_NAME = "com.User";
        Class<?> clazz = Class.forName(CLASS_NAME);
        Field[] fields = clazz.getDeclaredFields();

        boolean flag = clazz.isAnnotationPresent(Table.class);
        //获取表名
        if (flag){
            Table table = (Table)clazz.getAnnotation(Table.class);
            result.append("INSERT INTO `" + table.tableName() +"`");
        }
        //获取成员变量
        Set<Field> set = new HashSet<Field>();
        for (int i = 0; i < fields.length; i++) {
            boolean flag2 = fields[i].isAnnotationPresent(Column.class);
            if(flag2){
                set.add(fields[i]);
            }
        }
        //获取成员变量的值
        Map<String,String> selectColumn = new HashMap<>();
        for (Field m:set) {
            Column column = m.getAnnotation(Column.class);

 if(clazz.getMethod("get"+initCap(column.columnName())).invoke(user)!=null
){
                String content =
clazz.getMethod("get"+initCap(column.columnName())).invoke(user).toString(
);
                selectColumn.put(column.columnName(),content);
            }
        }

        StringBuffer keys = new StringBuffer();
```

```
        keys.append(" (");
        for (Map.Entry<String,String> s:selectColumn.entrySet()){
            keys.append("`"+s.getKey()+"`, ");
        }
        String string1 = keys.substring(0,keys.length()-2);
        result.append(string1+") VALUES");

        StringBuffer values = new StringBuffer();
        values.append(" (");
        for (Map.Entry<String,String> s:selectColumn.entrySet()){
            values.append("`"+s.getValue()+"`, ");
        }
        String string2 = values.substring(0,values.length()-2);
        result.append(string2+")");
        String insert = result.toString();
        return insert;
    }

    @Override
    public String insert(List<User> users) throws ClassNotFoundException,
NoSuchMethodException, InvocationTargetException, IllegalAccessException{
        StringBuffer result = new StringBuffer();
        String CLASS_NAME = "com.User";
        Class<?> clazz = Class.forName(CLASS_NAME);
        Field[] fields = clazz.getDeclaredFields();

        boolean flag = clazz.isAnnotationPresent(Table.class);
        //获取表名
        if (flag){
            Table table = (Table)clazz.getAnnotation(Table.class);
            result.append("INSERT INTO `" + table.tableName() +"`");
        }
        //获取成员变量
        Set<Field> set = new HashSet<Field>();
        for (int i = 0; i < fields.length; i++) {
            boolean flag2 = fields[i].isAnnotationPresent(Column.class);
            if(flag2){
                set.add(fields[i]);
            }
        }
```

```
        //获取成员变量的值
        int count = 0;
        for (int i = 0; i < users.size(); i++) {
            Map<String,String> map = new HashMap<>();
            for (Field m:set)
            {
                Column column = m.getAnnotation(Column.class);
                if (clazz.getMethod("get" +
initCap(column.columnName())).invoke(users.get(i)) != null) {
                    String content = clazz.getMethod("get" +
initCap(column.columnName())).invoke(users.get(i)).toString();
                    map.put(column.columnName(), content);
                }
            }
            if(count == 0){
                StringBuffer keys = new StringBuffer();
                keys.append(" (");
                for (Map.Entry<String,String> s:map.entrySet()){
                    keys.append("`"+s.getKey()+"`, ");
                }
                String string1 = keys.substring(0,keys.length()-2);
                result.append(string1+") VALUES");
            }

            StringBuffer values = new StringBuffer();
            values.append(" (");
            for (Map.Entry<String,String> s:map.entrySet()){
                values.append("`"+s.getValue()+"`, ");
            }
            String string2 = values.substring(0,values.length()-2);
            result.append(string2+") ,");
            count ++;
        }

        String insert = result.substring(0,result.length()-2);
        return insert;
    }

    @Override
```

```
    public String delete(User user) throws ClassNotFoundException,
NoSuchMethodException, InvocationTargetException, IllegalAccessException{
        StringBuffer result = new StringBuffer();
        String CLASS_NAME = "com.User";
        Class<?> clazz = Class.forName(CLASS_NAME);
        Field[] fields = clazz.getDeclaredFields();

        boolean flag = clazz.isAnnotationPresent(Table.class);
        if (flag){
            Table table = (Table)clazz.getAnnotation(Table.class);
            result.append("DELETE FROM `" + table.tableName()+"` WHERE");
        }

        Set<Field> set = new HashSet<Field>();
        for (int i = 0; i < fields.length; i++) {
            boolean flag2 = fields[i].isAnnotationPresent(Column.class);
            if(flag2){
                set.add(fields[i]);
            }
        }

        Map<String,String> map = new HashMap<>();
        for (Field m:set) {
            Column column = m.getAnnotation(Column.class);

 if(clazz.getMethod("get"+initCap(column.columnName())).invoke(user)!=null
){
                String content =
clazz.getMethod("get"+initCap(column.columnName())).invoke(user).toString(
);
                map.put(column.columnName(),content);
            }
        }
        for (Map.Entry<String,String> s:map.entrySet()){
            result.append(" `"+s.getKey()+"` = "+"`"+s.getValue()+"`");
        }
        String delete = result.toString();
        return delete;
    }
```

```
    @Override
    public String update(User user) throws  NoSuchMethodException,
InvocationTargetException, IllegalAccessException, ClassNotFoundException
{
        StringBuffer result = new StringBuffer();
        String CLASS_NAME = "com.User";
        Class<?> clazz = Class.forName(CLASS_NAME);
        Field[] fields = clazz.getDeclaredFields();

        boolean flag = clazz.isAnnotationPresent(Table.class);
        if (flag){
            Table table = (Table)clazz.getAnnotation(Table.class);
            result.append("UPDATE `" + table.tableName()+"` SET");
        }
        Set<Field> set = new HashSet<Field>();
        for (int i = 0; i < fields.length; i++) {
            boolean flag2 = fields[i].isAnnotationPresent(Column.class);
            if(flag2){
                set.add(fields[i]);
            }
        }
        Map<String,String> map = new HashMap<>();
        for (Field s:set) {
            Column column = s.getAnnotation(Column.class);

 if(clazz.getMethod("get"+initCap(column.columnName())).invoke(user)!=null
){
                String content =
clazz.getMethod("get"+initCap(column.columnName())).invoke(user).toString(
);
                map.put(column.columnName(),content);
            }
        }
        for (Map.Entry<String,String> m:map.entrySet()){
            if(!m.getKey().equals("id")){
                result.append(" `"+m.getKey()+"` =
"+"`"+m.getValue()+"`");
                result.append(",");
            }
        }
```

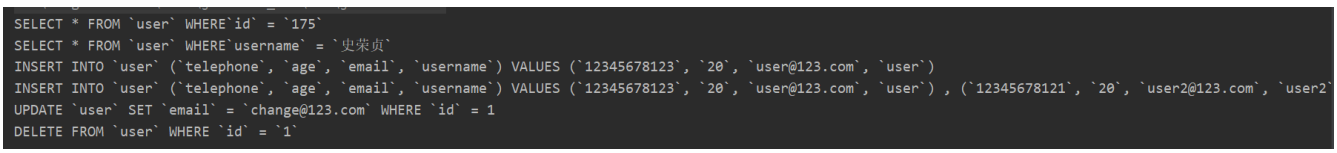
```
            int length = result.length();
            result.append(" WHERE `id` = "+map.get("id"));
            String update = result.delete(length-1,length).toString();
            return update;
        }

        public static String initCap(String str){
            //name->Name
            return str.substring(0,1).toUpperCase() + str.substring(1);
        }
    }
```

上述方法中利用反射和注解取得表名。（即User类中@Table的值）

通过Field类取得了该类的成员变量。后用每个成员变量@column的值取得变量对应的表中字段。

再通过invoke()方法，调用类中get方法，取得变量的值，识别出该值是否有效后，进行字符串的拼接和裁剪返回即可。

三、实验结果



SELECT * FROM `user` WHERE `id` = `175` SELECT * FROM `user` WHERE `username` = 史荣贞 INSERT INTO `user` (`telephone` , `age` , `email` , `username`) VALUES (`12345678123` , `20` , `user@123.com` , `user`) INSERT INTO `user` (`telephone` , `age` , `email` , `username`) VALUES (`12345678123` , `20` , `user@123.com` , `user`) , (`12345678121` , `20` , `user2@123.com` , `user2`) UPDATE `user` SET `email` = `change@123.com` WHERE `id` = 1 DELETE FROM `user` WHERE `id` = `1`