**注解反射生成SQL语句**

**1.任务说明**

        ①有一张用户表，字段包括：用户ID、用户名、昵称、年龄、性别、所在城市、邮箱、手机号；

        ②使用java注解来对用户表的每个字段或字段的组合条件进行动态生成SQL查询语句。

**2.代码实现**

**2.1 定义注解**

定义一个描述用户表的注解：

**[java]** [view plain](http://blog.csdn.net/zen99t/article/details/50351575) [copy](http://blog.csdn.net/zen99t/article/details/50351575)

[在CODE上查看代码片](https://code.csdn.net/snippets/1758767)

1. **package** dao;
2. **import** java.lang.annotation.ElementType;
3. **import** java.lang.annotation.Retention;
4. **import** java.lang.annotation.RetentionPolicy;
5. **import** java.lang.annotation.Target;
7. @Target({ElementType.TYPE})
8. @Retention(RetentionPolicy.RUNTIME)
9. **public** **@interface** Table {
11. String value();
12. }

定义一个描述用户表属性字段的注解：

**[java]** [view plain](http://blog.csdn.net/zen99t/article/details/50351575) [copy](http://blog.csdn.net/zen99t/article/details/50351575)

[在CODE上查看代码片](https://code.csdn.net/snippets/1758767)

1. **package** dao;
2. **import** java.lang.annotation.ElementType;
3. **import** java.lang.annotation.Retention;
4. **import** java.lang.annotation.RetentionPolicy;
5. **import** java.lang.annotation.Target;
7. @Target({ElementType.FIELD})
8. @Retention(RetentionPolicy.RUNTIME)
9. **public** **@interface** Column {
11. String value();
12. }

**2.2 定义映射到数据库的bean**

用户表，字段包括：用户ID、用户名、昵称、年龄、性别、所在城市、邮箱、手机号：

**[java]** [view plain](http://blog.csdn.net/zen99t/article/details/50351575) [copy](http://blog.csdn.net/zen99t/article/details/50351575)

[在CODE上查看代码片](https://code.csdn.net/snippets/1758767)

1. **package** dao;
3. @Table("user")
4. **public** **class** User {
6. @Column("id")
7. **private** **int** id;
9. @Column("user\_name")
10. **private** String userName;
12. @Column("nick\_name")
13. **private** String nickName;
15. @Column("age")
16. **private** **int** age;
18. @Column("city")
19. **private** String city;
21. @Column("email")
22. **private** String email;
24. @Column("mobile")
25. **private** String mobile;
27. **public** **int** getId() {
28. **return** id;
29. }
30. **public** **void** setId(**int** id) {
31. **this**.id = id;
32. }
33. **public** String getUserName() {
34. **return** userName;
35. }
36. **public** **void** setUserName(String userName) {
37. **this**.userName = userName;
38. }
39. **public** String getNickName() {
40. **return** nickName;
41. }
42. **public** **void** setNickName(String nickName) {
43. **this**.nickName = nickName;
44. }
45. **public** **int** getAge() {
46. **return** age;
47. }
48. **public** **void** setAge(**int** age) {
49. **this**.age = age;
50. }
51. **public** String getCity() {
52. **return** city;
53. }
54. **public** **void** setCity(String city) {
55. **this**.city = city;
56. }
57. **public** String getEmail() {
58. **return** email;
59. }
60. **public** **void** setEmail(String email) {
61. **this**.email = email;
62. }
63. **public** String getMobile() {
64. **return** mobile;
65. }
66. **public** **void** setMobile(String mobile) {
67. **this**.mobile = mobile;
68. }
69. }

**2.3 返回SQL查询语句的实现**

根据参数动态返回查询语句：

**[java]** [view plain](http://blog.csdn.net/zen99t/article/details/50351575) [copy](http://blog.csdn.net/zen99t/article/details/50351575)

[在CODE上查看代码片](https://code.csdn.net/snippets/1758767)

1. **package** dao;
2. **import** java.lang.reflect.Field;
3. **import** java.lang.reflect.Method;
5. **public** **class** ReturnQuery {
6. **public** **static** String query(Object u){
7. StringBuilder sqlStrBuilder = **new** StringBuilder();
8. //1.获取到Class
9. Class c = u.getClass();
10. //判断是否包含Table类型的注解
11. **if**(!c.isAnnotationPresent(Table.**class**)){
12. **return** **null**;
13. }
14. //2.获取到table的名字
15. Table t = (Table) c.getAnnotation(Table.**class**);
16. String tableName = t.value();
17. //加入 where 1=1 ，是为了防止未来如果没有条件的情况下也不会报错
18. sqlStrBuilder.append("select \* from ").append(tableName).append(" where 1=1");
20. Field[] fArray = c.getDeclaredFields();   //获取类属性的所有字段，
21. //3.遍历所有的字段
22. **for** (Field field : fArray) {
23. //4.处理每个字段对应的sql
24. //判断是否包含Column类型的注解
25. **if**(!field.isAnnotationPresent(Column.**class**)){
26. **continue**;
27. }
28. //4.1.拿到字段上面注解的值，即Column注解的值
29. Column column = field.getAnnotation(Column.**class**);
30. String columnName = column.value();
31. //4.2.拿到字段的名
32. String filedName = field.getName();
33. //获取相应字段的getXXX()方法
34. String getMethodName = "get" + filedName.substring(0, 1).toUpperCase()
35. + filedName.substring(1);
36. //字段的值
37. Object fieldValue = **null**;//属性有int、String等，所以定义为Object类
38. **try** {
39. Method getMethod = c.getMethod(getMethodName);
40. fieldValue = getMethod.invoke(u);
41. } **catch** (Exception e) {
42. e.printStackTrace();
43. }
45. //4.3.拼接sql
46. **if**(fieldValue==**null** || (fieldValue **instanceof** Integer && (Integer)fieldValue==0)){
47. **continue**;
48. }
49. sqlStrBuilder.append(" and ").append(columnName);
50. **if**(fieldValue **instanceof** String){
51. **if**(((String)fieldValue).contains(",")){
52. String[] values = ((String)fieldValue).split(",");
53. sqlStrBuilder.append(" in(");
54. **for** (String v : values) {
55. sqlStrBuilder.append("'").append(v).append("'").append(",");
56. }
57. sqlStrBuilder.deleteCharAt(sqlStrBuilder.length()-1);
58. sqlStrBuilder.append(")");
59. }
60. **else**{
61. sqlStrBuilder.append("=").append("'").append(fieldValue).append("'");
62. }
63. }
64. **else** **if**(fieldValue **instanceof** Integer){
65. sqlStrBuilder.append("=").append(fieldValue);
66. }
67. }
68. **return** sqlStrBuilder.toString();
70. }
71. }

**2.4 返回SQL语句的测试类**

**[java]** [view plain](http://blog.csdn.net/zen99t/article/details/50351575) [copy](http://blog.csdn.net/zen99t/article/details/50351575)

[在CODE上查看代码片](https://code.csdn.net/snippets/1758767)

1. **package** dao;
2. **public** **class** Test {
3. **public** **static** **void** main(String[] args) {
4. User u1 = **new** User();
5. u1.setId(9);  //查询id为9的用户
7. User u2 = **new** User();
8. u2.setUserName("JTZeng");   //模糊查询用户名为JTZeng的用户
9. u2.setAge(21);
11. User u3 = **new** User();
12. u3.setEmail("123@163.com,123@qq.com");  //查询邮箱有任意一个的用户
14. String sql1 = ReturnQuery.query(u1);    //查询id为9的用户
15. String sql2 = ReturnQuery.query(u2);    //查询用户名为JTZeng和年龄为21的用户
16. String sql3 = ReturnQuery.query(u3);    //查询邮箱中有任意一个的用户
18. System.out.println(sql1);
19. System.out.println(sql2);
20. System.out.println(sql3);
21. }
22. }

输出结果如下：

