对象化SQL开发指导手册

**2018年10月**

**文档变更记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **版本编号** | | **变更日期** | **操作人** | **说明** | **文件名** |
| V0.1 | 2018.10.25 | | 郑灶生 | 起草文档 | 《对象化SQL开发指导手册V0.1》 |

## 对象化SQL

### 配置方式

Maven引入

<**dependency**>  
 <**groupId**>com.egzosn.jdbc</**groupId**>  
 <**artifactId**>dao-params</**artifactId**>  
 <**version**>1.0.2</**version**>  
</**dep1endency**>

### 使用方式

创建Where对象，设置对应的值，where.builderParas().getSqlString()即可获取对应的where之后的sql，这里可包含 where， group、order

Params params = QueryParams.*WHERE*(**"name"**, **"张三"**, **"user"**).setAlias(**"user"**).and(**"age"**, 12).order(**"sex"**).group(**"name"**);  
System.***out***.println(params.builderParas().getSqlString());  
System.***out***.println(params.getParas());  
**输出结果：**

where user.name= ? and user.age= ? Group by user.name Order by user.sex DESC

[张三, 12]

params = **new** Where().setAlias(**"user"**).and(**"name"**, **"张三"**).and(**"age"**, **new** Integer[]{1, 12}, Restriction.***BW***).order(**"sex"**).group(**"name"**);  
System.***out***.println(params.builderParas().getSqlString());  
System.***out***.println(params.getParas());

**输出结果：**  
where user.name= ? and user.age between ? and ? Group by user.name Order by user.sex DESC

[张三, 1, 12]

System.***out***.println(params.builderAttrs().getSqlString());  
System.***out***.println(params.getAttrs());

**输出结果：**

where user.name= :name and user.age between :age1 and :age2 Group by user.name Order by user.sex DESC

{name=张三, age1=1, age2=12}

**通过代理对象OdOrderDaoParams形式(工具生成)**

DateFormat df = **new** SimpleDateFormat(**"yyyy-MM-dd"**);  
params = **new** OdOrderDaoParams()  
 .setName(**"张三"**, **false**)  
 .setOrderCreateTimeBetween(**new** Date[]{df.parse(**"2018-11-01"**), df.parse(**"2018-11-02"**)}, **false**)  
 .order(OdOrderDaoParams.Field.***shipType***.getColumn())  
 ;  
 System.***out***.println(params.builderParas().getSqlString());  
 System.***out***.println(params.getParas());

**输出结果：**

where o.name= ? and o.order\_create\_time between ? and ? Order by o.ship\_type DESC

[张三, Thu Nov 01 00:00:00 CST 2018, Fri Nov 02 00:00:00 CST 2018]

### 注意事项

本插件暂时无法支持相同字段名设置两次以上的条件，如果需要设置两次以上条件可用代替字符进行设置，得到sql后再进行替换，或者尝试用**between**

或者in等方式进行。这里看后期使用量是否进行优化填坑

**错误例子：**

params = **new** Where().setAlias(**"user"**).and(**"name"**, **"张三"**).and(**"age"**, 1, Restriction.***GT***).and(**"age"**, 12, Restriction.***LE***).order(**"sex"**).group(**"name"**);  
  
System.***out***.println(params.builderParas().getSqlString());  
System.***out***.println(params.getParas());

**输出结果：**

where user.name= ? and user.age<= ? Group by user.name Order by user.sex DESC

[张三, 12]

这里你会发现前者被后者给覆盖了。

**解决方式一：**

遇到这种范围的情况你可以用between进行，可以回看1.1.2使用方式，里面已有讲解

**解决方式二：**

给变量加前缀或者后缀等。最后sql进行字符替换

params = **new** Where().setAlias(**"user"**).and(**"name"**, **"张三"**).and(**"age"**, 1, Restriction.***GT***).and(**"$1$age"**, 12, Restriction.***LE***).order(**"sex"**).group(**"name"**);  
  
System.***out***.println(params.builderParas().getSqlString().replaceAll(**"\\.\\$[0-9]\\$"**, **"."**));  
System.***out***.println(params.getParas());

**输出结果：**

where user.name= ? and user.age> ? and user.age<= ? Group by user.name Order by user.sex DESC

[张三, 1, 12]

目前该工具只支持单项sql，不支持括号这种类型。如果需要编写这类sql，可以通过变量增加前缀后缀方式进行与上面类似，最后通过替换方式进行。这里看后期使用量是否进行优化填坑

## 规则配置式SQL

该组件是在《对象化SQL》基础上扩展的，通过字符串规则生成对应的sql，这里主要针对where部分，该组件的初衷是表单规则配置的，在请求中过滤出符合的规则，这里规则组成是由 表单域的 name+value组成的。

### 配置方式

Maven引入

<**dependency**>   
 <**groupId**>com.egzosn.jdbc</**groupId**>  
 <**artifactId**>sql-filter</**artifactId**>  
 <**version**>1.0.2</**version**>  
</**dep1endency**>

### 使用方式

表达式格式： **“Q^表别名#字段名^(|, !|)[^条件表达式^类型]”**

条件表达式：**com.egzosn.jdbc.params.enums.Restriction**

类型：**com. egzosn.jdbc.params.filter.Type**

**案例：**

Map<String, String[]> requests = **new** HashMap<String, String[]>();  
requests.put(**"Q^o#name^!|"**, **new** String[]{**"egan"**});  
requests.put(**"Q^o#follow\_people\_account^!|"**, **new** String[]{**"^NNUL=1"**});  
requests.put(**"Q^odOrderTmpSub#first\_warning"**, **new** String[]{**"!|^EQ^S=1"**});  
Params params = **new** SqlFilter(**new** SqlFilterRequest(requests)).setAlias(**"o"**).getQueryParams();  
System.***out***.println(params.builderParas().getSqlString());  
System.***out***.println(params.getParas());

**输出结果：**

where o.name= ? and o.follow\_people\_account is not null and odOrderTmpSub.first\_warning= ?

[egan, 1]

**注意事项**：这里看起来已经实现了，但是正常这个表达式是放至在前端表单域name上面的，那么如果被人加入了非法的注入语句怎么办，那不就都暴露出去了吗

**优化方案：**

这里需要用上 com.egzosn.jdbc.params.**Field4Column**，这只是一个接口定义，具体实现需要自行实现。

例子查阅 com.egzosn.jdbc.params.test.Field，这种类一般会用生成器进行生成

**public class** OdOrderDaoParams **extends** Where {  
 **public static final** String ***ALIAS*** = **"o"**;  
 **public static final** String ***TABLE*** = **"od\_order"**;  
 **public static final** Map<String, Field4Column> ***FIELD\_4\_COLUMNS*** = **new** HashMap<String, Field4Column>(Field.*values*().**length**);  
 **static** {  
 **for** (Field4Column field4Column : Field.*values*()){  
 ***FIELD\_4\_COLUMNS***.put(field4Column.getField(), field4Column);  
 }  
 }  
 **public enum** Field **implements** Field4Column {  
 ***pkId***(**"pk\_id"**),***name***(**"name"**),***orderCreateTime***(**"order\_create\_time"**),***shipType***(**"ship\_type"**);  
 **public** String **column**;  
 Field(String column) {  
 **this**.**column** = column;  
 }  
 @Override  
 **public** String getField(){  
 **return** name();  
 }  
 @Override  
 **public** String getColumn(){  
 **return column**;  
 }@Override  
 **public** String getSelect() {  
 **return** SQLTools.*getSelect*(***ALIAS***, **this**);  
 }@Override  
 **public** String getSelect(String prefix) {  
 **return** SQLTools.*getSelect*(prefix, **this**);  
 }  
 **public static** String getSelects(Field... ignoreFields) {  
 **return** *getSelects*(***ALIAS***, ignoreFields);  
 }  
 **public static** String getSelects(String prefix, Field... ignoreFields) {  
 **return** SQLTools.*getSelects*(prefix, Field.*values*(), ignoreFields);  
 }  
 **public static** String getSelect(Field... fields) {  
 **return** *getSelect*(***ALIAS***, fields);  
 }  
 **public static** String getSelect(String prefix, Field... fields) {  
 **return** SQLTools.*getSelects*(prefix, fields);  
 }  
 }

}

**使用案例**

requests.clear();  
requests.put(**"Q^o#login\_name^!|^LK"**, **new** String[]{**"zzs"**});  
requests.put(**"Q^o#shipType^!|^IN^i"**, **new** String[]{**"1"**,**"3"**});  
requests.put(**"Q^o#orderCreateTime^!|^BW^D"**, **new** String[]{**"2018-11-1,2018-11-2"**});  
params = **new** SqlFilter(**new** SqlFilterRequest(requests), OdOrderDaoParams.***FIELD\_4\_COLUMNS***).setAlias(**"o"**).getQueryParams();  
System.***out***.println(params.builderParas().getSqlString());  
System.***out***.println(params.getParas());

**输出结果：**

where o.order\_create\_time between ? and ? and o.ship\_type in (?,?) and o.login\_name like ?

[Thu Nov 01 00:00:00 CST 2018, Fri Nov 02 00:00:00 CST 2018, 1, 3, %zzs%]

**排序分页等**

requests.put(**"sort"**, **new** String[]{**"name"**});  
requests.put(**"order"**, **new** String[]{**"desc"**});  
requests.put(**"page"**, **new** String[]{**"1"**});  
requests.put(**"rows"**, **new** String[]{**"10"**});  
params = **new** SqlFilter(**new** SqlFilterRequest(requests), OdOrderDaoParams.***FIELD\_4\_COLUMNS***).setAlias(**"o"**).setOrder().setPageing().getQueryParams();  
System.***out***.println(params.builderParas().getSqlString());  
System.***out***.println(params.getParas());  
System.***out***.println(String.*format*(**"分页信息：开始页%s, 每页%s行"**, params.getPage().getPageIndex(), params.getPage().getPageSize()));

**输出结果：**

where o.order\_create\_time between ? and ? and o.ship\_type in (?,?) and o.login\_name like ? Order by o.name DESC

[Thu Nov 01 00:00:00 CST 2018, Fri Nov 02 00:00:00 CST 2018, 1, 3, %zzs%]

分页信息：开始页1, 每页10行

## 项目源码

https://github.com/egzosn/params-parent.git