# ORM-开发文档v1.0

|  |  |  |
| --- | --- | --- |
| 序号 | 更新内容 | 更新时间 |
| 1 | 增加了Columns类，IORMListener | 2017-11-03 |
| 2 | IORMListener 监听 insert/update/delete事件 | 2017-11-04 |
| 3 | IDefaultListener 绑定默认参数 | 2017-11-10 |
| 4 | IChangeListener 对象变更（insert/update/delete）  ORMUtils工具类 | 2017-11-11 |
| 5 | 数据库支持H2,MySQL,SQLServer, Oracle | 2018-07-06 |

1. 目标
   1. 高效开发
   2. 易于维护
   3. 规范化
   4. 学习成本低
2. 设计
   1. 使用标准的SQL语句 (http://www.w3school.com.cn/sql/index.asp)
   2. 查询多种实现，简单查询以及复杂查询（保持数据安全性，删除只提供主键删除）
   3. 常用方法封装
   4. 事务由spring-orm管理
   5. v1.1 支持H2,MySQL、Oracle、SQLServer
   6. 定义：
      1. 条件(Condition/Terms) 是表达式(Expression/Express)的集合；
      2. 表达式由列(Column) ，值Object（Value）符号 ExpressionType组成。
      3. 列（Column）由函数（function）表别名（alias）名称（name）别名（asName）组成。
3. 数据库命名规范
   1. **统一使用大写**：表、字段、索引等。
      1. 系统表 SYS\_TABLE
      2. 其他业务类型表 XX\_TABLE
   2. SYS\_USER表，SD\_CASE等，需备注 COMMENT

|  |
| --- |
| CREATE TABLE `T\_TEST` (  `ID` INT(11) NOT NULL AUTO\_INCREMENT COMMENT '编号;自增数字',  `BALANCE` DECIMAL(10,2) NULL DEFAULT '0.00' COMMENT '金额;金额',  `NAME` VARCHAR(100) NULL DEFAULT NULL COMMENT '名称;姓名',  `URL` VARCHAR(200) NULL DEFAULT NULL COMMENT '图片;图片路径',  `CONTENT` TEXT NULL COMMENT '备注;备注',  `CREATOR` VARCHAR(100) NULL DEFAULT NULL COMMENT '创建者;创建者',  `CREATE\_DATE` TIMESTAMP NULL DEFAULT NULL COMMENT '创建日期;创建时间',  `UPDATOR` VARCHAR(100) NULL DEFAULT NULL COMMENT '修改者;修改者',  `UPDATE\_DATE` TIMESTAMP NULL DEFAULT NULL COMMENT '修改日期;修改时间',  PRIMARY KEY (`ID`)  )COLLATE='**utf8mb4\_bin**' ENGINE=InnoDB; |

* 1. MySQL/SQLServer/Oracle/H2

|  |  |  |
| --- | --- | --- |
| **Java对象** | **数据库列类型** | **说明** |
| String | Varchar/varchar2/char | 需指定长度，定义了长度，输入框需做限制，如UUID 36位 |
| String | Text | 较长文本使用 Oracle使用clob、SQL Server 使用NText |
| Integer | Int(8)/Integer/Number |  |
| Long | bigInt(15)/Number | 精度15将自动转化java.util.Date类型 |
| Double | Decimal/Numberic | 精度15将自动转化java.util.Date类型 |
| Date | Timestamp/Datetime | 不允许使用currentTimestamp（默认为null） |
| Clob | Clob | Strin/Clob自动转相互转换，编码统一使用UTF-8 |
| Blob | Blob | Strin/Blob自动转相互转换，编码统一使用UTF-8 |
| byte[] | Binary | byte[]/Binary自动转相互转换 |
| Decimal | Decimal |  |
| Float | Float/Decimal/  Numberic |  |

* 1. 外键命名规范

正常命名：

SYS\_USER(ID, NAME);

不能这样命名：

SYS\_USER(SYS\_USER\_ID, NAME);

则关联的表命名：

SYS\_USER\_CONFIG(ID, SYS\_USER\_ID, NAME);

* 1. 长度规范

主键建议使用

ORACLE varchar2 最大长度4000（通常一个中文占3字节）

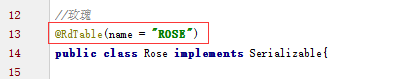
主键（组合主键）长度不超过192(mysql5.6.0一中文占用4个字节，其它版本3个字节）

姓名、电话等长度通常可以定义：64

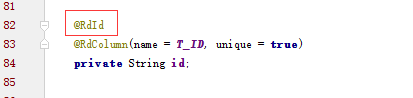
URL、地址 256

普通描述：1024中文

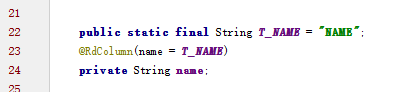
1. 标记说明
   1. RdTable ： 表名



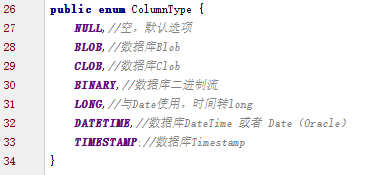
* 1. RdId ： 主键



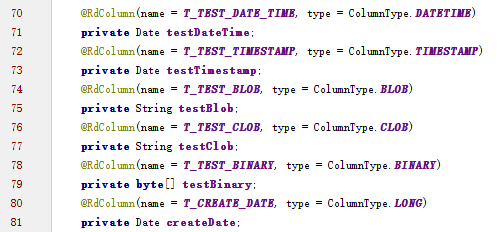
* 1. RdColumn ： 列, T\_NAME 对应数据库列的名称



ColumnType 说明：



如：



注：Date 为 java.util.Date

1. 基本查询
   1. get 获取对象；

允许使用对象（对象优先使用id，之后使用其他条件）；主键ID；IQuery或者IMultiQuery；

* 1. update 更新；

若要更新null, update(t, true)；

* 1. delete 删除

根据主键删除；不提供按条件删除，防止误删

1. 查询
   1. list() 全表查询
   2. list(start, size) 范围查询
   3. list(Express … expresses) 简单表达式查询
   4. list(Terms terms) 简单条件查询
   5. list(Terms terms, MultiOrder multiOrder) 简单条件+排序查询

1. 类说明
   1. **Column**

|  |  |
| --- | --- |
| **对象** | **描述** |
| new Column(“name”) | name |
| new Column(“a”, “name”) | a.name |
| new Column(“a”, “name”, “username”) | a.name AS username |
| new Column(Expression.SUM,  “a”, “name”, “username”) | sum(a.name) AS username |
| new Column().function(Expression.SUM)  .alias("a")  .name("name")  .as("username"); | sum(a.name) AS username |

* 1. **Columns**

|  |  |
| --- | --- |
| **对象** | **描述** |
| new Column(“a”, “name”，“value”) | 1. name, a.value   通常用于查询，如  query.createQuery(Class,  new Column(“a”,”name”,”value”) |

* 1. **Expression(Express) ExpressionType**

注：若无跨数据库要求，日期可以是：Timestamp/DateTime(SQLServer)/Date(Oracle)等，但是时间的比较需要相关函数。所以不建议，建议直接定义为long类型，方便比较。

|  |  |
| --- | --- |
| **类型** | **描述** |
| EXPRESSION\_ALL | \* |
| EXPRESSION\_SUM | SUM |
| EXPRESSION\_MAX | MAX |
| EXPRESSION\_MIN | MIN |
| EXPRESSION\_COUNT | COUNT |
| EXPRESSION\_AVG | AVG |

|  |  |
| --- | --- |
| **对象** | **描述** |
| new Express(“age”, 1,  ExpressionType.CDT\_More) | age > 1 |
| new Express(Expression.SUM ,“age”,1,  ExpressionType.CDT\_More) | SUM(age) > 1 |
| new Expression(column1, column2)  column1 = new Column("a","type");  column2 = new Column("b","id"); | a.type = b.id |
| new Expression(column1, 1,  ExpressionType.CDT\_More)  column1 = new Column("a","type"); | a.type > 1 |

* 1. **Condition(Terms)**

|  |  |
| --- | --- |
| **对象** | **描述** |
| new Condition()  .or(expression1).or(expression2)  .and(Expression3)  .and(Expression4) | (expression1 or expression2)  AND expression3 AND expression4 |
| new Terms().or(express1).  and(express2).and(express3) | express1 AND express2 AND express3 |

* 1. **Join(on) JoinType**

|  |  |
| --- | --- |
| **类型** | **描述** |
| FULL\_JOIN | FULL JOIN |
| INNER\_JOIN | INNER JOIN |
| LEFT\_JOIN | LEFT JOIN |
| RIGHT\_JOIN | RIGHT JOIN |

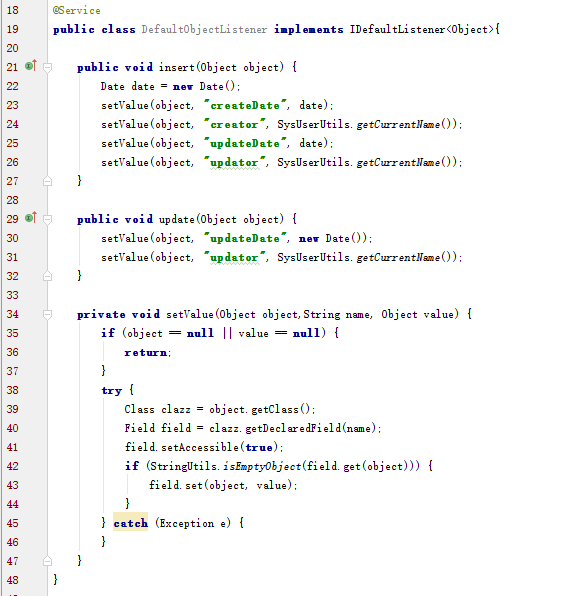
|  |  |
| --- | --- |
| **对象** | **描述** |
| new Join("a", Test.class).on(column1, column2)  column1 = new Column("a","type");  column2 = new Column("b","id"); | 默认Left Join  LEFT JOIN Test a ON  a.type = b.id |
| new Join("a", Test.class).  on(column1, column2)  on("type", 1, ExpressionType.CDT\_MORE) | LEFT JOIN Test a  ON  a.type = b.id AND a.type > 1 |
| new Join("a", Test.class, JoinType.RIGHT\_JOIN).  on(column1, column2) | RIGHT JOIN Test a ON  a.type = b.id |

* 1. **Order/MultiOrder**

|  |  |
| --- | --- |
| **对象** | **描述** |
| new Order(column1, Order.ASC)  column1 = new Column("a","type"); | order by a.type ASC |
| new MultiOrder().asc("type").desc("id"); | order by type asc,  id desc |
| new MultiOrder().asc(column1).desc(column2);  column1 = new Column("a","type");  column2 = new Column("b","id"); | order by a.type ASC,  b.id DESC |

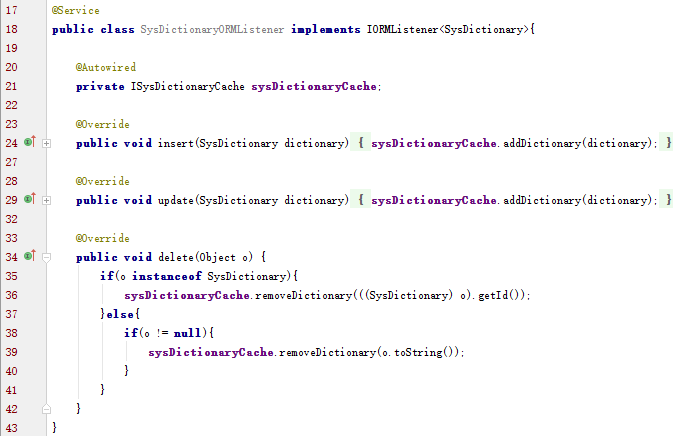
* 1. **Page**
     1. size 每页数量
     2. page 当前页
     3. rows 数据List
     4. total 总数
  2. **Listener**
  3. **IDefaultListener**

**实现该类，默认触发该监听器。**



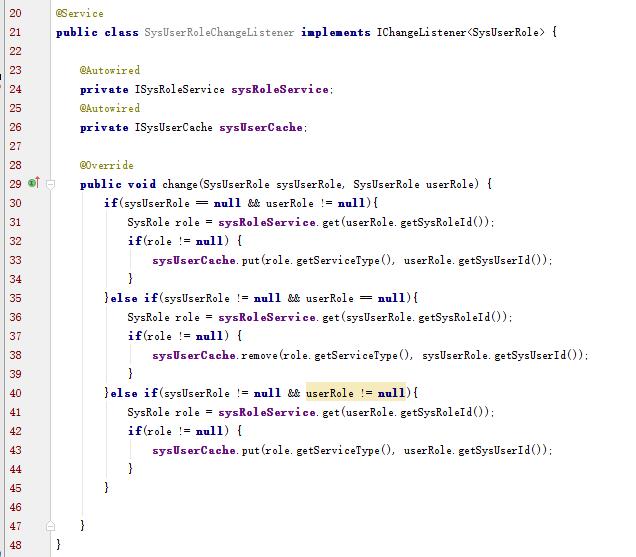
* 1. **IORMListener**

**实现该类，可以对表的插入、更新、删除进行监听。**

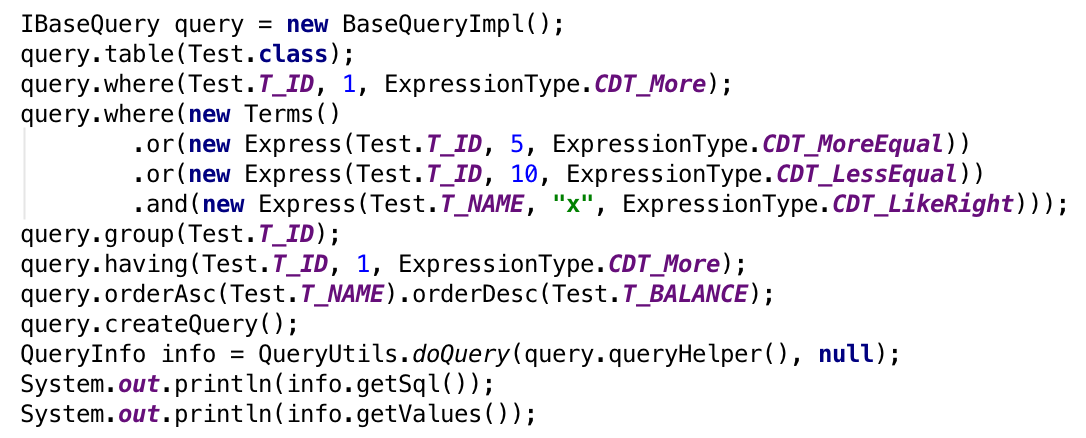


* 1. **IChangeListener**

**实现该类，可以对表的对象发生变化时，出发该动作。**



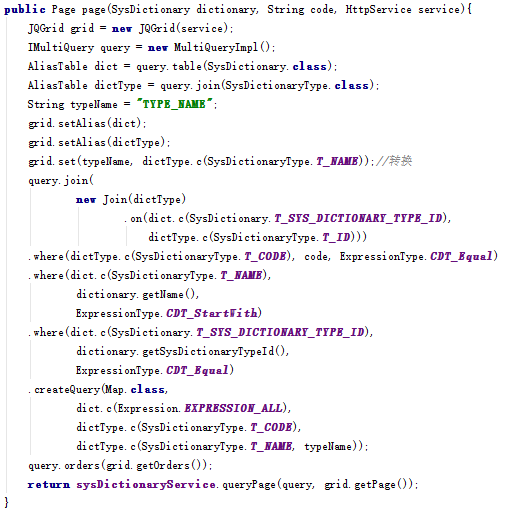
1. 简单查询 IBaseQuery
   1. **示例**



|  |
| --- |
| IBaseQuery query = **new** BaseQueryImpl(); query.table(Test.**class**); query.where(Test.***T\_ID***, 1, ExpressionType.***CDT\_More***); query.where(**new** Terms()  .or(**new** Express(Test.***T\_ID***, 5, ExpressionType.***CDT\_MoreEqual***))  .or(**new** Express(Test.***T\_ID***, 10, ExpressionType.***CDT\_LessEqual***))  .and(**new** Express(Test.***T\_NAME***, **"x"**, ExpressionType.***CDT\_LikeRight***))); query.group(Test.***T\_ID***); query.having(Test.***T\_ID***, 1, ExpressionType.***CDT\_More***); query.orderAsc(Test.***T\_NAME***).orderDesc(Test.***T\_BALANCE***); query.createQuery(); QueryInfo info = QueryUtils.*doQuery*(query.queryHelper(), **null**); System.***out***.println(info.getSql()); System.***out***.println(info.getValues()); |

|  |
| --- |
| SELECT \* FROM T\_TEST WHERE ID > ? AND ( ID >= ? OR ID <= ?) AND NAME like ? GROUP BY ID HAVING ( ID > ?) ORDER BY NAME ASC,BALANCE DESC  [1, 5, 10, x%, 1] |

1. 复杂查询IMultiQuery
   1. **示例一**

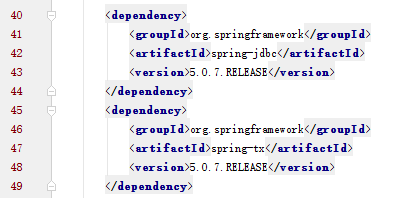


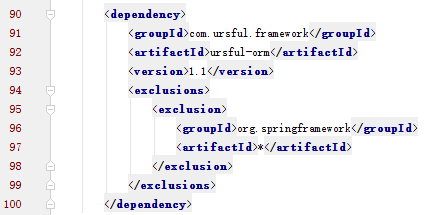
**代码：**

|  |
| --- |
| JQGrid grid = **new** JQGrid(service); IMultiQuery query = **new** MultiQueryImpl(); AliasTable dict = query.table(SysDictionary.**class**); AliasTable dictType = query.join(SysDictionaryType.**class**); String typeName = **"TYPE\_NAME"**; grid.setAlias(dict); grid.setAlias(dictType); grid.set(typeName, dictType.c(SysDictionaryType.***T\_NAME***));*//转换* query.join(  **new** Join(dictType)  .on(dict.c(SysDictionary.***T\_SYS\_DICTIONARY\_TYPE\_ID***),  dictType.c(SysDictionaryType.***T\_ID***))) .where(dictType.c(SysDictionaryType.***T\_CODE***), code, ExpressionType.***CDT\_Equal***) .where(dict.c(SysDictionaryType.***T\_NAME***),  dictionary.getName(),  ExpressionType.***CDT\_StartWith***) .where(dict.c(SysDictionary.***T\_SYS\_DICTIONARY\_TYPE\_ID***),  dictionary.getSysDictionaryTypeId(),  ExpressionType.***CDT\_Equal***) .createQuery(Map.**class**,  dict.c(Expression.***EXPRESSION\_ALL***),  dictType.c(SysDictionaryType.***T\_CODE***),  dictType.c(SysDictionaryType.***T\_NAME***, typeName)); query.orders(grid.getOrders()); **return sysDictionaryService**.queryPage(query, grid.getPage()); |

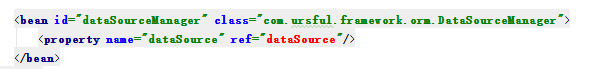
1. 用法

依赖Spring-jdbc,Spring-tx

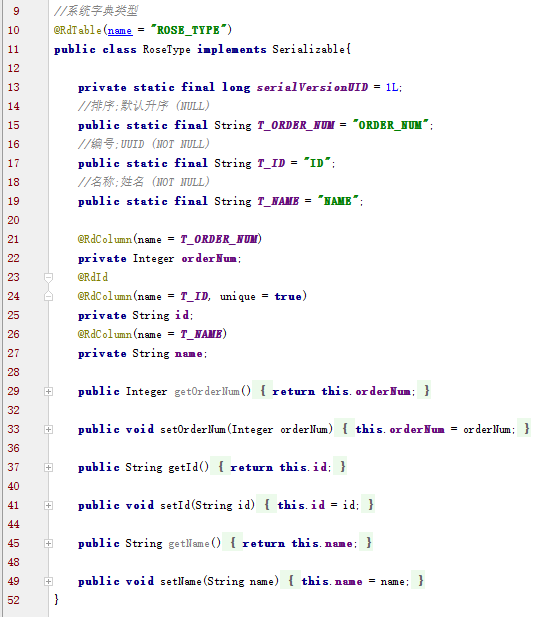




A.Bean：



B. 模型



C.接口：



D.实现类



E.用法

