# Lucky\_2.1.0的使用文档

## 简介

Lucky是一款基于java的数据持久层框架，是对JDBC高度封装后的产物，主要作用是帮助开发人员简单且高效的开发数据库。

原生的JDBC操作相对繁琐，主要的原因如下：

1. **JDBC操作数据库需要用到主要三个类：获取链接的类（Connection），预处理类（PreparedStatement）和结果集类（ResultSet），涉及到的类太多，难以记忆。**
2. **JDBC对非查询操作的处理算是比较完美的，最令人抓狂的是对查询操作的处理，当我们执行完查询操作后JDBC只返回给我们一个结果集对象（ResultSet），想要取出详细的结果我们还需要进行繁琐的遍历操作。**
3. **JDBC提供的方法不具备扩展性，就拿查询操作来举例：一般情况下，我们数据库中每多一张表时我们都会为其创建对应的实体类。那么就产生了一个很令人头疼的问题：如果想得到实体类形式的查询结果，我们就不得不一遍又一遍的去写“先遍历结果集，后封装数据”的代码**

初次使用JBDC的同学会有更深刻的体会吧！在开发之初使用JDBC也是令我非常头痛的，所以一直都在思考有没有什么好的解决方案,或者写出一个通用的一劳永逸的方法？答案是有的，因为解决问题的方法永远都比问题本身要多！所以Lucky诞生了！

Lucky的诞生有效的解决了以上的三大问题：

1. **Lucky操作数据库只需要一个类：数据控制类(SqlControl),只要获得了这个类的对象便可以轻松的操作数据库**
2. **Lucky执行非查询操作后返回的是一个布尔类型的结果，执行查询操作后返回的是Object对象或者List<?>集合，我们只需要做一次强制类型转换就能拿到实体类形式的查询结果Lucky中提供的方法几乎就是通用的，想要封装不同结果时只需要传入实体类对象对应的Class对象就好了，Lucky就会根据配置进行自动封装。**

开发Lucky主要使用到的技术有：java的反射机制，泛型，注解，配置文件，JDBC，字符串处理，基本的数据结构。

## 配置问题

俗话说的好：想要马儿跑就得让马儿吃草，所以想要Lucky稳定的工作就必须为她提供可靠的信息，具体信息如下

* **得告诉Lucky你的数据库信息(驱动包，数据库位置，用户名，密码)。**
* **构建表对应的实体类，而且实体类的书写必须符合一定的规范。**
* **建立实体类与表的映射关系（配置方式和注解方式）**

## 使用流程

1. **如何告诉Lucky我的数据库信息？**

这个部分基本是固定格式，在src下创建一个名为lucky.pqoperties的配置文件，properties文件中的内容都是以key-value的形式体现的，所以在配置文件中key的部分大体上都是固定的（左边），有用的信息为value（右边）,这部分的类容是需要我们因人而异的进行配置的。

配置数据库信息：

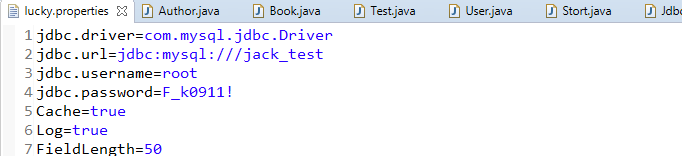
jdbc.driver=com.mysql.jdbc.Driver

jdbc.url=jdbc:mysql:///databasename

jdbc.username=root

jdbc.password=passeord

**具体场景：**



1. **构建实体类的规范有哪些？**

**注：想要Lucky正常工作实体类的编写一定要符合如下规范：**

**1. 实体类的属性名必须与表的字段名一致**

**2. 实体类的属性的类型必须与表字段的类型一致**

**3. 表的主键必须为int类型**

**4. 实体类的属性类型必须为java基本类型的包装类型（int-Integer double-Double）**

**5. 每个实体类都必须由系统生成toString()方法**

1. **如何建立实体类与表的映射关系？**

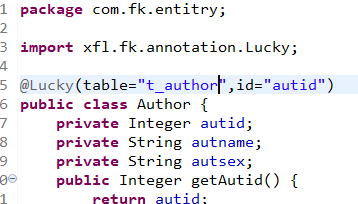
实体类创建完成后必须与对应的表建立映射关系后Lucky才能通过操作实体类来间接的操作表。

建立映射关系的实质就是通过一些方法告诉实体类表的关键信息，如：这个类对应数据库中那张表？这张表的主键是哪一个字段？

通过注解建立映射关系：

**@Lucky(table=”数据库表名”,id=”表的主键”)**

**注：@Lucky注解的必须写在在类的声明前（如图）**

****

1. **如何获得操作数据库的数据控制对象（SqlControl）**

做好之前三步骤之后我们完成了所有的基本操作，接下来就可以创建对象来操作数据库了，数据控制对象的创建和普通类对象的创建没有区别：

**SqlControl sql=new SqlControl()**;

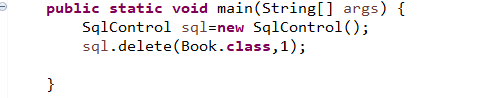
1. **操作数据库**

* Id操作

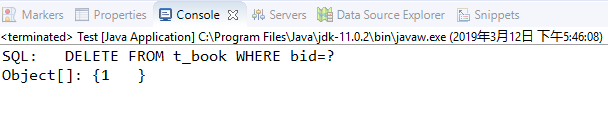
**D查询**

**sql.delete(实体类的class,ID)**

**具体场景：**

****

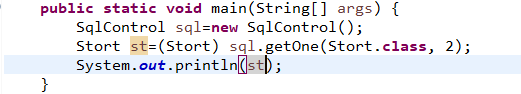
**执行效果：**

****

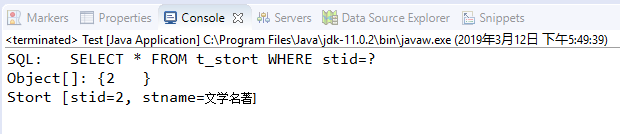
**ID查询**

**Object obj=sql.getOne(实体类的class,ID)**

**具体场景：**

****

**执行效果：**

****

* 对象操作
* 预编译sql操作

## 特色功能