# Lucky使用文档

## 环境配置

### 导入jar包

使用Lucky必须导入mysql的驱动包和Lucky的jar包



### 创建配置文件

配置文件为位置固定且名称固定的.properties文件

名称：lucky.properties

位置：scr文件夹下



## 如何操作数据库

### 实体类的编写规范（不使用建表机制）

1. 实体类的类名必须与表名一致

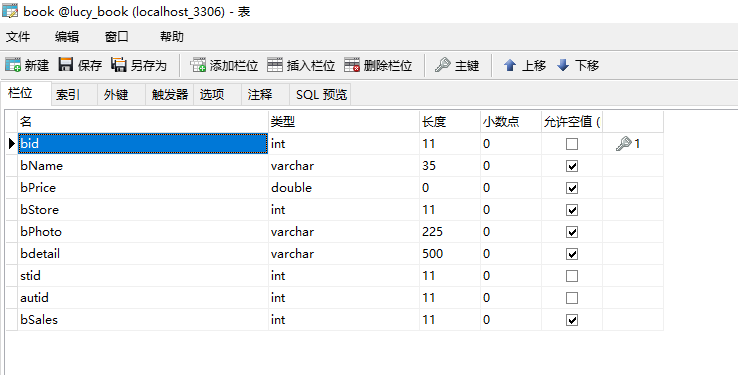
2. 实体类的属性名必须与表的字段名一致

3. 实体类的属性的类型必须与表字段的类型一致

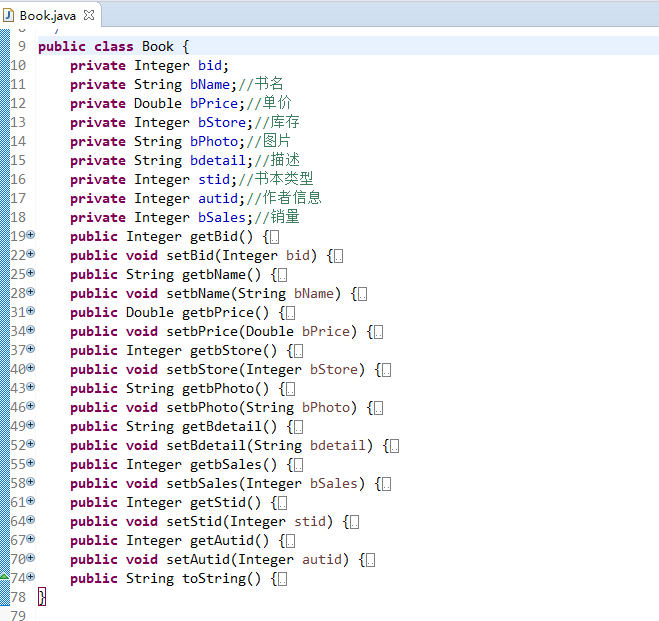
4. 表的主键必须为int类型

5. 实体类的属性类型必须为java基本类型的包装类型

6. 每个实体类都必须由系统生成toString()方法



Book表



Book类

### 使用SqlControl类操作数据库

#### 数据库配置

#MySql驱动配置

jdbc.driver=com.mysql.jdbc.Driver

#数据库路径配置

jdbc.url=jdbc:mysql:/ip:port//lucy\_book（数据库名）

#用于登录MySql服务取的用户名

jdbc.username=root

#密码

jdbc.password=1234

#### ID操作

1. **配置实现（在Lucky2.0.3中已被弃用）**

需要在lucky.properties文件中进行ID操作的相关配置

格式：实体类名.id=ID属性名

#id操作配置(Lucky执行ID操作时必要的配置，也可以使用注解的方式替代)

Admin.id=admid

Author.id=autid

Book.id=bid

Consigness.id=consid

Detail.id=detid

Notes.id=bufid

Orders.id=ordid

Shoppcart.id=shopid

Stort.id=stid

User.id=uid

**ID操作流程(删除,查询)：**



***SqlControl sql=new SqlControl();//使用无参构造获得对象***

***sql.~~delete~~(“实体类名”,ID) ≡ DELETE FROM 表名 WHERE 主键=ID;***

***sql.~~delete~~(“Book”,1)******≡ DELETE FROM Book WHERE bid=1;***



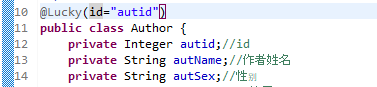
***sql.~~getOne~~(类名.Class,ID) ≡ SELECT \* FROM 表名 WHERE 主键=ID;***

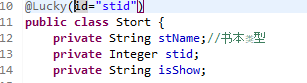
***sql.~~getOne~~(Book.class,3) ≡ SELECT \* FROM Book WHERE bid=3;***

1. **注解实现（推荐）**

需要在每个类声明之前加上@Lucky注解

@Lucky(id=”ID属性”)





**ID操作流程(删除,查询)：**



***SqlControl sql=new SqlControl();//使用无参构造获得对象***

***sql.delete(类名.class,ID) ≡ DELETE FROM 表名 WHERE 主键=ID;***

***sql.delete(Book.class,1)******≡ DELETE FROM Book WHERE bid=1;***



***SqlControl sql=new SqlControl();//使用无参构造获得对象***

***sql.getone(类名.class,ID) ≡ SELECT \* FROM 表名 WHERE 主键=ID;***

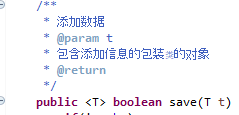
***sql. getone(Book.class,1)******≡ SELECT \* FROM Book WHERE bid=1;***

#### 对象方式

使用对象操作方式不需要其它的配置，我们只需要将操作信息封装到实体类中，然后使用Lucky中的方法就可以轻松的操作数据库。

注意：在生成预编译sql语句时，实体类中的信息是以 AND 连接的

**增加操作**



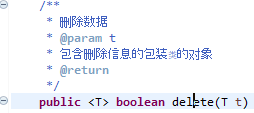
***Book book=new Book(); //构建实体类对象***

***book.setBname(“活着”);//书名***

***book,setPrice(“34.5”);//价格***

***sql.save(book) ≡ INSERT INTO Book表(bname,price) VALUES(“活着”，34.5)***

**删除操作**



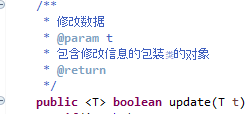
***Book book=new Book(); //构建实体类对象***

***book.setBname(“活着”);//书名***

***book,setPrice(“34.5”);//价格***

***sql.delete(book) ≡ DELETE FROM Book表 WHERE bname=”活着” AND price=34.5***

**修改操作**



注意：在使用对象方式更新时必须在对象中封装ID信息，否则会报如下错误！



***Book book=new Book(); //构建实体类对象***

***book,setBid(2);//ID信息***

***book.setBname(“活着”);//书名***

***book,setPrice(“34.5”);//价格***

***sql.delete(book) ≡ UPDATE Book表 SET bname=”活着” ,price=34.5 WHERE bid=2;***

**查询操作**

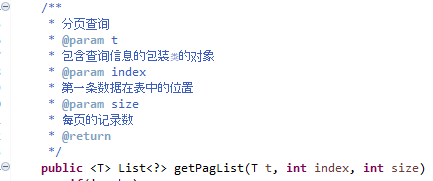
获得单一对象



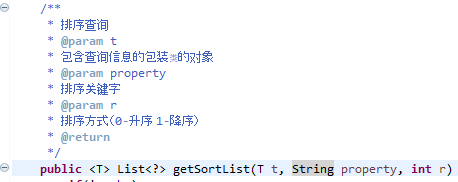
获得List集合



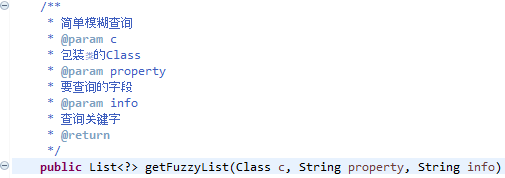
分页查询



排序查询



简单模糊查询



#### 预编译Sql方式

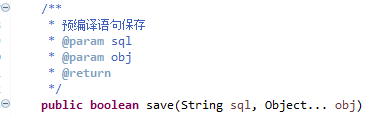
预编译 SQL：带占位（?）的SQL语句

例：SELECT \* FROM t\_book WHERE bid=? And price=?;

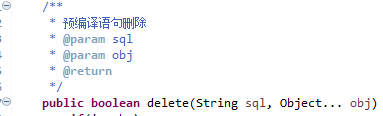
占位数组obj : 填充占位符的对象组成的数组

例：sql.save(“INSERT INTO t\_book(bname,price) VALUES(?,?)”,”活着”,34.5)

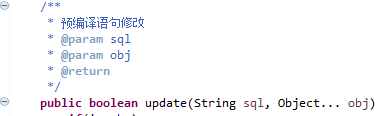
**增加操作**



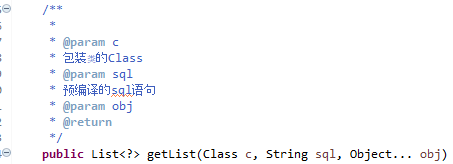
**删除操作**



**修改操作**



**查询操作**



## Lucky的事务处理机制

### 开启事务

### 提交事务

### 事务回滚

## Lucky的建表机制

### 配置文件

使用Lucky的建表机制前需要在配置文件中做必要的配置。

Class.Url.整数(n)=实体类所在的全路径

注意：在配置多个类时所有的整数(n)都必须要是连续的（请记住这里的整数(n)，它将在建表操作中起到十分重要的作用）



做好以上配置后Lucky就得到了建表的基本信息，但是你还需要告诉它更多的细节信息，比如：主键？外键？是否级联？这个时候Lucky的注解就派上用场了

### @Lucky注解

@Lucky注解的作用：

Lucky在拿到配置文件中的信息时建表信息的大体框架就已经明了了，但还缺少细节信息，@Lucky注解就是用来告诉Lucky细节的。

**表名**

**@Lucky(table=”表名”)**

**主键**

**@Lucky(id=”主键属性”)**

**外键**

**@Lucky(key={”外键属性1”,……,”外键属性n”})**

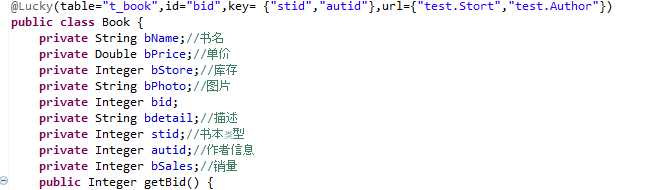
**外键的关联表**

**@Lucky(url={”外键1对应的类的全路径”,……,” 外键n对应的类的全路径”})**

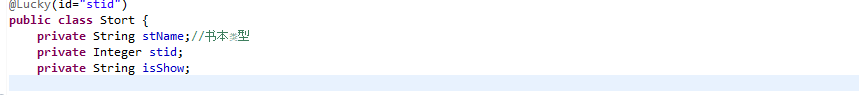
注意：table可以不写，不写时默认表名为类名

key和url 必须成对出现并且相互对应（值得一提的是这了两个属性都是数组哦！）

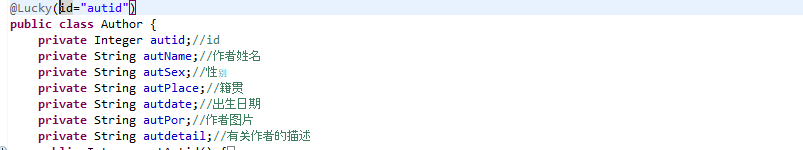
**Book类**



**Stort类**



**Author类**



### @Cascade注解

顾名思义，这个就是告诉Lucky级联信息的注解

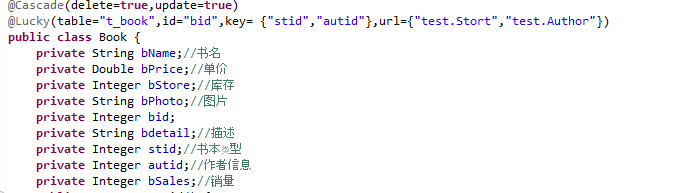
**级联删除**

**@Cascsde(delete=true)**

**级联更新**

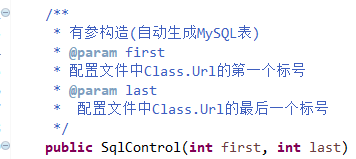
**@Cascsde(update=true)**

**Book类**



### 使用SqlControl有参构造建表

做好如上配置后Lucky就知道了建表的所有信息了，那么我们如何使用建表机制帮助我们建表呢？这个问题的答案很简单，我们只需要使用SqlControl类的有参构造就可以了。还记得之前的整数n吗？它就是构造器的参数！



要使用Lucky的建表机制创建如上Book,Stort,Authr这三张表，我们可以这样做：  
***SqlControl sql=new SqlControl(1,3);//使用有参构造获得对象***

***类加载器在创建这个类对象的同时建表机制就会启动帮我们建表。***

## Lucky的缓存机制

### 缓存配置

要使用Lucky的缓存机制必须在配置文件中进行配置:

Cache=true;



### 缓存原理

我们想要查询、增加、删除、修改数据时就必须向数据库发送SQL语句。但仔细想想，未必每次查询都要向数据库发送SQL语句，例如如果我们第二次查询的结果与第一次相同时，我们就没有必要去发送SQL语句，这样不就可以大大降低性能的消耗了？

开启缓存机制后Lucky就能帮助我们实现以上的功能，当我们第一次向数据库发送SQL时，Lucky会将查询结果存入缓存，当我们再次查询时，Lucky会首先去缓存中查找对应结果，如果找到就直接返回结果，找不到才会向数据库发送SQL。