# Quartz笔记

用来实现定时任务的第三方的API，主要用来实现在规定时刻执行某项功能任务的，主要的核心有JobDetail(任务的详情,schedule(任务的调度器,trigger(触发器,calendar(日历,job(接口

# 实现定时任务的步骤

## 首先要定义定时的任务，实现QuartzJobBean接口

### 1.1重写Quartz的方法executionInternal方法

在这个方法里面写自己的业务需求,还用到了Calendar这个日历类

@Override  
protected void executeInternal(JobExecutionContext context) throws JobExecutionException {  
 //获得格林威治时间  
 Calendar calendar = Calendar.*getInstance*();  
 //进行时间的运算  
 calendar.add(Calendar.*MINUTE*,-15);  
 //获得超时时间  
 Date timeOut = calendar.getTime();  
 Order order = new Order();  
 order.setStatus(6).setUpdated(new Date());  
 UpdateWrapper<Order> updateWrapper = new UpdateWrapper<>();  
 updateWrapper.eq("status",1).lt("updated",timeOut);  
 orderMapper.update(order,updateWrapper);  
 System.*out*.println("定时任务执行成功!!!!!京淘项目全部完成!!!!!!");  
}

## 2 定义配置类Config，使用@configuration,如：OrderQuartzConfig

### 2.1 在这个类的中需要定义两个Bean 一个是JobDetail，用建造者模式来构建和这个类

//定义任务详情  
@Bean  
public JobDetail orderJobDetail(){  
 return JobBuilder.*newJob*(OrderQuartz.class)  
 .withIdentity("orderQuartz")  
 .storeDurably().build();  
}

JobBuilder.newJob（这里传入的就是之前实现了的QuartzJobBean的类对象）,接着就是定义名字，还要有一个持久化的商店？

### 2.2 还有一个类就是Trigger类 ,这个类中包含一个调度器Schedule

调度器Schedule内部有定时器，如果到了执行时间，调度器就会工作，调用触发器执行自定义的工作任务

//定义触发器  
@Bean  
public Trigger orderTrigger(){  
 //表示定时任务多久执行一次!!!!  
 CronScheduleBuilder scheduleBuilder = CronScheduleBuilder.*cronSchedule*("0 0/1 \* \* \* ?");  
 return TriggerBuilder.*newTrigger*()  
 .forJob(orderJobDetail())  
 .withIdentity("orderQuartz")  
 .withSchedule(scheduleBuilder)  
 .build();  
}