# 1.任务调度 SpringTask

## 1.1 什么是任务调度

在企业级应用中，经常会制定一些“计划任务”，即在某个时间点做某件事情，核心是以时间为关注点，即在一个特定的时间点，系统执行指定的一个操作。常见的任务调度框架有Quartz 和 SpringTask 等。

## 1.2 SpringTask 入门小 Demo

创建模块 pinyougou-task-service,引入 spring 相关依赖 dao 和 common 工程，tomcat7 端口为 9108

添加 web.xml

添加配置文件 applicationContext-task.xml ,内容如下

```
<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:p="http://www.springframework.org/schema/p"

    xmlns:context="http://www.springframework.org/schema/context"

    xmlns:task="http://www.springframework.org/schema/task"

    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
```

```
        http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd

        http://www.springframework.org/schema/task
http://www.springframework.org/schema/task/spring-task-4.2.xsd">

    <context:component-scan base-package="com.pinyougou.task"/>

    <task:annotation-driven/>

</beans>
```

创建包 com.pinyougou.task

编写类

```
@Component

public class SeckillTask {

    /**

     * 刷新秒杀商品

     */

    @Scheduled(cron="* * * * * ?")

    public void refreshSeckillGoods(){

        System.out.println("执行了任务调度"+new Date());

    }

}
```

执行后会看到控制台每秒都输出了当前时间，其中 cron 设置的为表达式，是执行的时间规则。

## 1.3 Cron 表达式

### 1.3.1 Cron 表达式格式

Cron 表达式是一个字符串，字符串以 5 或 6 个空格隔开，分为 6 或 7 个域，每一个域代表一个含义，Cron 有如下两种语法格式：

（1）Seconds Minutes Hours DayofMonth Month DayofWeek Year

（2）Seconds Minutes Hours DayofMonth Month DayofWeek

每一个域可出现的字符如下：

Seconds:可出现", - * /"四个字符，有效范围为 0-59 的整数

Minutes:可出现", - * /"四个字符，有效范围为 0-59 的整数

Hours:可出现", - * /"四个字符，有效范围为 0-23 的整数

DayofMonth:可出现", - * / ? L W C"八个字符，有效范围为 1-31 的整数

Month:可出现", - * /"四个字符，有效范围为 1-12 的整数或 JAN-DEc

DayofWeek:可出现", - * / ? L C #"四个字符，有效范围为 1-7 的整数或 SUN-SAT 两个范围。1 表示星期天，2 表示星期一， 依次类推

Year:可出现", - * /"四个字符，有效范围为 1970-2099 年

每一个域都使用数字，但还可以出现如下特殊字符，它们的含义是：

(1)*：表示匹配该域的任意值，假如在 Minutes 域使用*, 即表示每分钟都会触发事件。

(2)?:只能用在 DayofMonth 和 DayofWeek 两个域。它也匹配域的任意值，但实际不会。因为 DayofMonth 和 DayofWeek 会相互影响。例如想在每月的 20 日触发调度，不管 20 日到底是星期几，则只能使用如下写法： 13 13 15 20 * ?, 其中最后一位只能用？，而不能使用*，如果使用*表示不管星期几都会触发，实际上并不是这样。

(3)-:表示范围，例如在 Minutes 域使用 5-20，表示从 5 分到 20 分钟每分钟触发一次

(4)/：表示起始时间开始触发，然后每隔固定时间触发一次，例如在 Minutes 域使用 5/20, 则意味着 5 分钟触发一次，而 25，45 等分别触发一次.

(5),:表示列出枚举值值。例如：在 Minutes 域使用 5,20，则意味着在 5 和 20 分每分钟触发一次。

(6)L:表示最后，只能出现在 DayofWeek 和 DayofMonth 域，如果在 DayofWeek 域使用 5L,意味着在最后的一个星期四触发。

(7)W: 表示有效工作日(周一到周五),只能出现在 DayofMonth 域，系统将在离指定日期的最

近的有效工作日触发事件。例如：在 DayofMonth 使用 5W，如果 5 日是星期六，则将在最近的工作日：星期五，即 4 日触发。如果 5 日是星期天，则在 6 日(周一)触发；如果 5 日在星期一 到星期五中的一天，则就在 5 日触发。另外一点，W 的最近寻找不会跨过月份

(8)LW:这两个字符可以连用，表示在某个月最后一个工作日，即最后一个星期五。

(9)#:用于确定每个月第几个星期几，只能出现在 DayofMonth 域。例如在 4#2，表示某月的第二个星期三。

### 1.3.2 Cron 表达式例子

0 0 10,14,16 * * ? 每天上午 10 点，下午 2 点，4 点

0 0/30 9-17 * * ? 朝九晚五工作时间内每半小时

0 0 12 ? * WED 表示每个星期三中午 12 点

"0 0 12 * * ?" 每天中午 12 点触发

"0 15 10 ? * *" 每天上午 10:15 触发

"0 15 10 * * ?" 每天上午 10:15 触发

"0 15 10 * * ? *" 每天上午 10:15 触发

"0 15 10 * * ? 2005" 2005 年的每天上午 10:15 触发

"0 * 14 * * ?" 在每天下午 2 点到下午 2:59 期间的每 1 分钟触发

"0 0/5 14 * * ?" 在每天下午 2 点到下午 2:55 期间的每 5 分钟触发

"0 0/5 14,18 * * ?" 在每天下午 2 点到 2:55 期间和下午 6 点到 6:55 期间的每 5 分钟触发

"0 0-5 14 * * ?" 在每天下午 2 点到下午 2:05 期间的每 1 分钟触发

"0 10,44 14 ? 3 WED" 每年三月的星期三的下午 2:10 和 2:44 触发

"0 15 10 ? * MON-FRI" 周一至周五的上午 10:15 触发

"0 15 10 15 * ?" 每月 15 日上午 10:15 触发

"0 15 10 L * ?" 每月最后一日的上午 10:15 触发

"0 15 10 ? * 6L" 每月的最后一个星期五上午 10:15 触发

"0 15 10 ? * 6L 2002-2005" 2002 年至 2005 年的每月的最后一个星期五上午 10:15 触发

"0 15 10 ? * 6#3" 每月的第三个星期五上午 10:15 触发

## 1.4 秒杀商品列表的增量更新

每分钟执行查询秒杀商品表，将符合条件的记录并且缓存中不存在的秒杀商品存入缓存

```
    /**
     * 刷新秒杀商品
     */
    @Scheduled(cron="0 * * * * ?")
    public void refreshSeckillGoods(){
        System.out.println("执行了任务调度"+new Date());
        //查询所有的秒杀商品键集合
        List ids = new ArrayList( redisTemplate.boundHashOps("seckillGoods").keys());
        //查询正在秒杀的商品列表
        TbSeckillGoodsExample example=new TbSeckillGoodsExample();
        Criteria criteria = example.createCriteria();
        criteria.andStatusEqualTo("1");//审核通过
        criteria.andStockCountGreaterThan(0);//剩余库存大于 0
        criteria.andStartTimeLessThanOrEqualTo(new Date());//开始时间小于等于当前时间
        criteria.andEndTimeGreaterThan(new Date());//结束时间大于当前时间
        criteria.andIdNotIn(ids);//排除缓存中已经有的商品
        List<TbSeckillGoods> seckillGoodsList=
seckillGoodsMapper.selectByExample(example);
```

```
        //装入缓存
        for( TbSeckillGoods seckill:seckillGoodsList ){
            redisTemplate.boundHashOps("seckillGoods").put(seckill.getId(),
seckill);
        }
        System.out.println("将"+seckillGoodsList.size()+"条商品装入缓存");
    }
```

## 1.5 过期秒杀商品的移除

每秒中在缓存的秒杀上皮列表中查询过期的商品，发现过期同步到数据库，并在缓存中移除该秒杀商品

```
    /**
     * 移除秒杀商品
     */
    @Scheduled(cron="* * * * * ?")
    public void removeSeckillGoods(){
        System.out.println("移除秒杀商品任务在执行");
        //扫描缓存中秒杀商品列表，发现过期的移除
        List<TbSeckillGoods> seckillGoodsList =
redisTemplate.boundHashOps("seckillGoods").values();
        for( TbSeckillGoods seckill:seckillGoodsList ){
            if(seckill.getEndTime().getTime()<new Date().getTime()  ){//如果结束日期
小于当前日期，则表示过期
                seckillGoodsMapper.updateByPrimaryKey(seckill);//向数据库保存记录
    redisTemplate.boundHashOps("seckillGoods").delete(seckill.getId());//移除缓存数
```

```
据
                System.out.println("移除秒杀商品"+seckill.getId());
            }
        }
        System.out.println("移除秒杀商品任务结束");
    }
```

# 2.Maven Profile

## 2.1 什么是 MavenProfile

在我们平常的 java 开发中，会经常使用到很多配制文件（xxx.properties，xxx.xml），而当我们在本地开发（dev），测试环境测试（test），线上生产使用（product）时，需要不停的去修改这些配制文件，次数一多，相当麻烦。现在，利用 maven 的 filter 和 profile 功能，我们可实现在编译阶段简单的指定一个参数就能切换配制，提高效率，还不容易出错.

profile 可以让我们定义一系列的配置信息，然后指定其激活条件。这样我们就可以定义多个 profile，然后每个 profile 对应不同的激活条件和配置信息，从而达到不同环境使用不同配置信息的效果。

## 2.2 Maven Profile 入门

修改 pinyougou-page-web 的 pom.xml

```
<properties>
   <port>9105</port>
 </properties>
 <build>
     <plugins>
         <plugin>
```