Timer,Quartz和Spring实现作业调度Java认证考试 PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/644/2021_2022_Timer_Quar_c104_644942.htm 一、java.util.Timer 在Java中有一个任务处理类java.util.Timer，非常方便于处理由时间触发的事件任务，只需建立一个继承java.util.TimerTask的子类，重载父类的run()方法实现具体的任务，然后调用Timer的public void schedule(TimerTask task, long delay, long period)方法实现任务的调度。 但是这种方法只能实现简单的任务调度，不能满足任务调度时间比较复杂的需求。比如希望系统在每周的工作日的8：00时向系统用户给出一个提示，这种方法实现起来就困难了，还有更为复杂的任务调度时间要求。 二、Quartz OpenSymphony 的Quartz提供了一个比较完美的任务调度解决方案。 Quartz 是个开源的作业调度框架，为在 Java 应用程序中进行作业调度提供了简单却强大的机制。 Quartz中有两个基本概念：作业和触发器。作业是能够调度的可执行任务，触发器提供了对作业的调度。 1、作业 实现 org.quartz.job 接口，实现接口方法 public void execute(JobExecutionContext context) throws JobExecutionException，在这个方法实现具体的作业任务。 代码例子： java 代码 execute 方法接受一个 JobExecutionContext 对象作为参数。这个对象提供了作业实例的运行时上下文。它提供了对调度器和触发器的访问，这两者协作来启动作业以及作业的 JobDetail 对象的执行。Quartz 通过把作业的状态放在 JobDetail 对象中并让 JobDetail 构造函数启动一个作业的实例，分离了作业的执行和作业周围的状态。JobDetail 对象储存作业的侦听器、群组

、数据映射、描述以及作业的其他属性。 2、触发器 触发器可以实现对任务执行的调度。Quartz 提供了几种不同的触发器，复杂程度各不相同。 简单触发器： public void task() throws SchedulerException { // Initiate a Schedule Factory SchedulerFactory schedulerFactory = new StdSchedulerFactory(). // Retrieve a scheduler from schedule factory Scheduler scheduler = schedulerFactory.getScheduler(). // current time long ctime = System.currentTimeMillis(). // Initiate JobDetail with job name, job group, and executable job class JobDetail jobDetail = new JobDetail("jobDetail-s1", "jobDetailGroup-s1", SimpleQuartzJob.class). // Initiate SimpleTrigger with its name and group name SimpleTrigger simpleTrigger = new SimpleTrigger("simpleTrigger", "triggerGroup-s1"). // set its start up time simpleTrigger.setStartTime(new Date(ctime)). // set the interval, how often the job should run (10 seconds here) simpleTrigger.setRepeatInterval(10000). // set the number of execution of this job, set to 10 times. // It will run 10 time and exhaust. simpleTrigger.setRepeatCount(100). // set the ending time of this job. // We set it for 60 seconds from its startup time here // Even if we set its repeat count to 10, // this will stop its process after 6 repeats as it gets it endtime by then. // simpleTrigger.setEndTime(new Date(ctime 60000L)). // set priority of trigger. If not set, the default is 5 // simpleTrigger.setPriority(10). // schedule a job with JobDetail and Trigger scheduler.scheduleJob(jobDetail, simpleTrigger). // start the scheduler scheduler.start(). } 首先实例化一个 SchedulerFactory，

获得调度器。创建 JobDetail 对象时，它的构造函数要接受一个 Job 作为参数。SimpleTrigger 是一个简单的触发器。在创建对象之后，设置几个基本属性以立即调度任务，然后每 10 秒重复一次，直到作业被执行 100 次。 Cron触发器 CronTrigger 支持比 SimpleTrigger 更具体强大的调度，实现起来却不是很复杂。CronTrigger基于 cron 表达式，支持类似日历的重复间隔更为复杂的调度时间上的要求。 100Test 下载频道开通，各类考试题目直接下载。详细请访问 www.100test.com