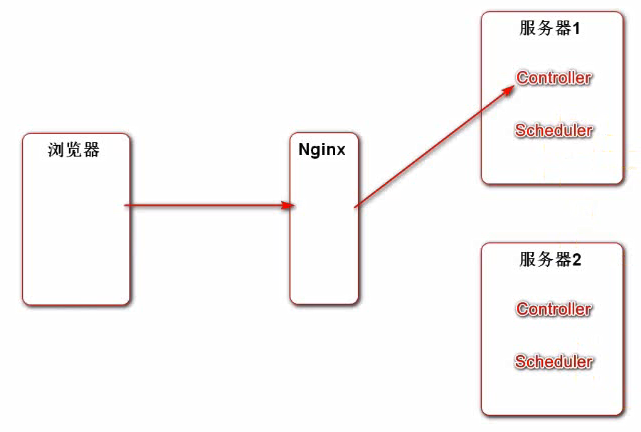
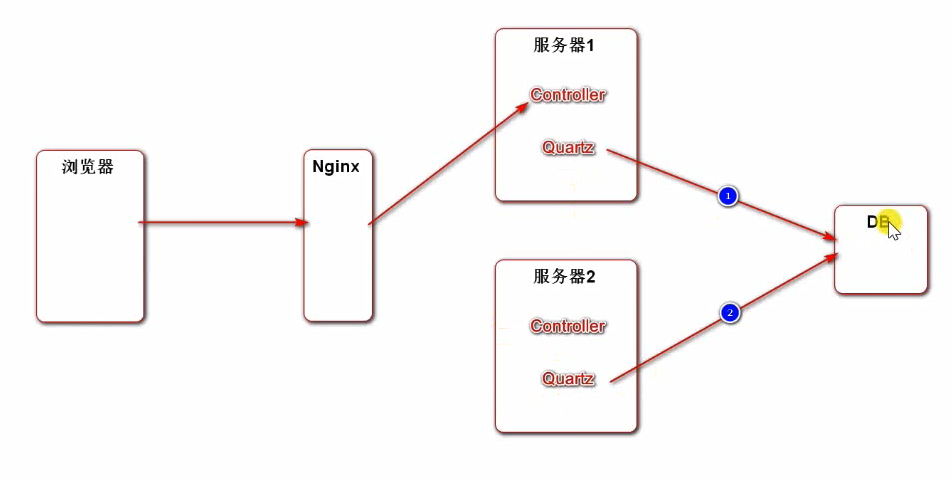


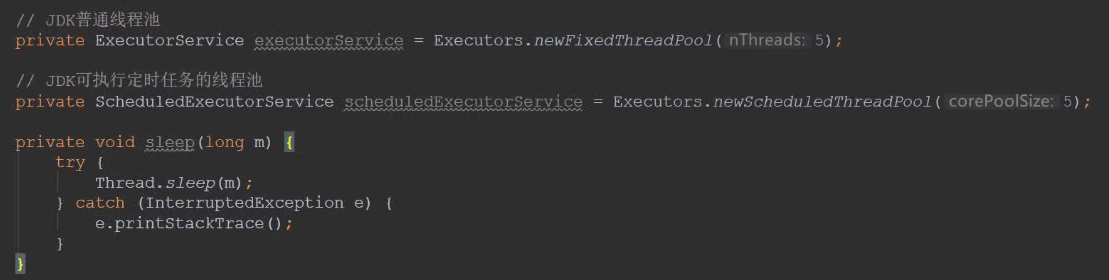
对于非定时任务，使用jdk的普通线程池和spring的普通线程池都行，但如果是定时任务，最好使用Spring Quartz工具来调用定时任务（也是一个线程池），因为在分布式的环境下，对于非定时任务，应该是在控制器里的，你通过负载均衡去调用这个执行这个任务，只有一个服务器执行，而定时任务，到达设置好的间隔时间，会自动执行，那在分布式的环境下，你执行的任务就重复了（服务器1和服务器2一块执行scheduler），可能造成一定的问题，比如清理临时文件，重复的清理就会造成问题

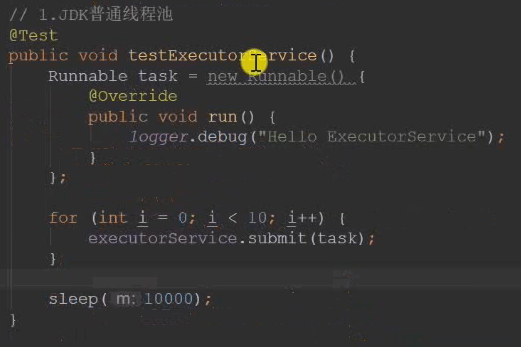


可以通过quartz来确保只有一个定时任务来执行，quartz在执行定时任务时，会从数据库中取得所需要的任务类型，名称，以及状态等等信息，当一个线程去数据库中取得相应的定时任务信息时，会先判断状态是否为运行状态，若为运行状态，说明已经有线程在执行这个任务了，则此线程就不去处理这个任务



Jdk线程池：



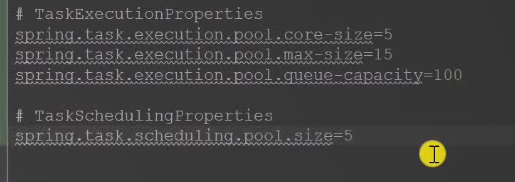


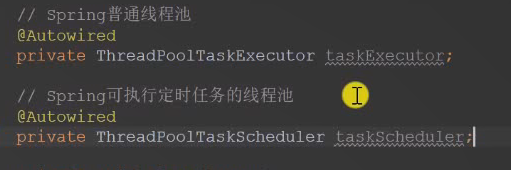
initialDelay：线程池延时时间（在程序启动后，等待相应的时间才会执行任务）

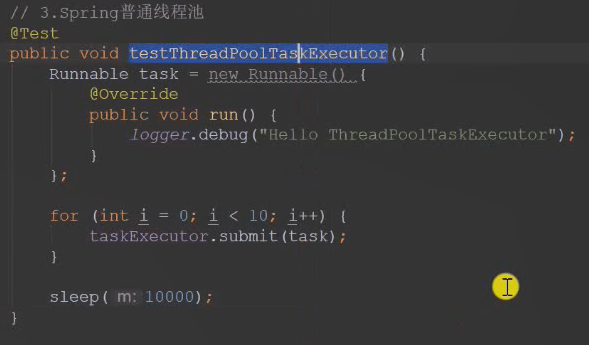
period：间隔，定时任务线程池在启动后自动执行



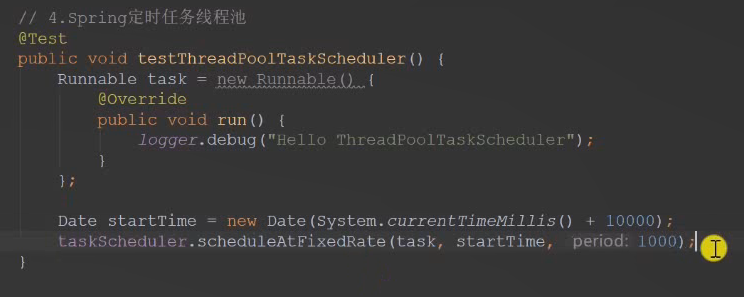
Spring：配置线程参数，

（线程池创建完毕，以后就是调用这个线程池）

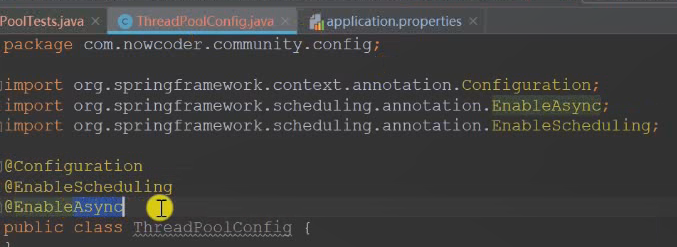




Spring的定时任务线程池，在传入任务时，第二个参数不是延迟时间，改为执行的启动时间

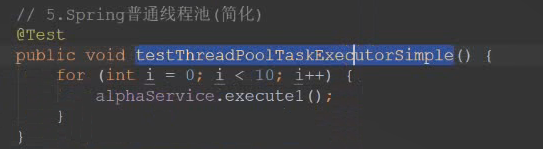


Spring的线程配置类：



Spring执行任务的方便之处：在实现bean注解的类里，在任意的方法上加上@Async注解后，此方法就相当于一个任务，在调用该方法时，spring会将其通过多线程的方式启用，按照在配置文件中配置的线程池大小





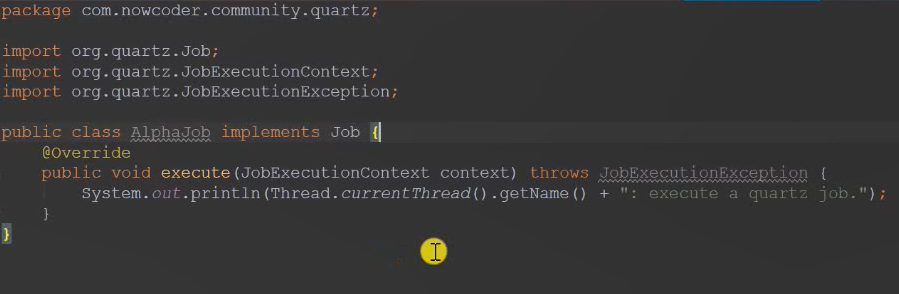
非定时任务：在实现bean注解的类里，任意方法上标注@Scheduled注解，此方法就相当于一个定时任务，不用外界调用，只要有程序启动，自动执行



Quartz分布式定时任务：

通过maven导入spring boot quarz的包，quarz的每一个定时任务所对应的主要的数据库表：job（所定义的任务）、JobDetail（job的必要描述以及相关参数配置）、Trigger（触发器 job什么时候运行、以什么样的频率反复运行），第一次运行相应的任务时，quearz会从配置的信息中读取到数据库中（立刻、自动），之后每次运行从数据库中取得相应的信息

定时任务的逻辑：

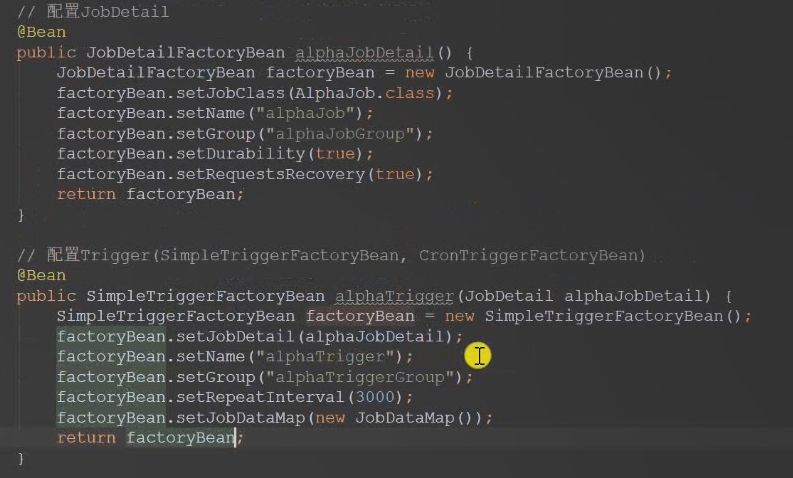
****

配置JobDetail和Trigger（在配置类中配置）：

jobDetail的相关配置（按顺序）：任务所从属于哪一类（bean的类型）、任务的名字、任务所在的组别、任务是否长久保存、任务是否可恢复

trigger的参数最好于对应的jobdetail一致，因为在有多个定时任务时，trigger在传参时会去找与参数名一致的jobDetail，Trigger的相关配置：对哪个jobDetail进行的配置、trigger的名字、trigger的组名、执行频率（间隔时间）、存储trigger状态的对象

配置完成后，在运行程序时自动执行



删除quartz任务（将数据库中对应的几个表一同删除）：传入的参数为任务名与任务所在的组名

