Spring quartz

博客分类：

- 批处理

springquartz批处理

本文摘自

http://www.blogjava.net/mikechen/archive/2012/07/03/382090.html

由于本人看到这篇文章很多帮助，欣喜之下，转之，希望更多人看到，谢谢作者！

一：quartz简介

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，定时调度器，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

二：quartz spring配置详解

- **为什么不适用java.util.Timer结合java.util.TimerTask**

1.主要的原因，适用不方便，特别是制定具体的年月日时分的时间，而quartz使用类似linux上的cron配置，很方便的配置每隔时间执行触发。

2.其次性能的原因，使用jdk自带的Timer不具备多线程，而quartz采用线程池，性能上比timer高出很多。

- **详解quartz在spring里面的配置**

在spring里主要分为两种使用方式：第一种，也是目前使用最多的方式，spring提供的MethodInvokingJobDetailFactoryBean代理类，通过雷利类直接调用任务类的某个函数；第二种,程序里实现quartz接口，quartz通过该接口进行调度。

主要讲解通过spring提供的代理类MethodInvokingJobDetailFactoryBean

1.业务逻辑类:业务逻辑是独立的，本身就与quartz解耦的，并没有深入进去，这对业务来讲是很好的一个方式。

```
public class  TestJobTask{

    /**
     *业务逻辑处理
     */
      public void   service(){
         /**业务逻辑*/

…

…

…
..
      }

}
```

2.增加一个线程池

```
<!-- 线程执行器配置，用于任务注册 -->
<bean id="executor"
class="org.springframework.scheduling.concurrent.ThreadPoolTaskExecutor">
 <property name="corePoolSize" value="10" />
 <property name="maxPoolSize" value="100" />
 <property name="queueCapacity" value="500" />
</bean>
```

3.定义业务逻辑类

```
<!-- 业务对象 -->
<bean id="testJobTask" class="com.mike.scheduling.TestJobTask" />
```

4.增加quartz调用业务逻辑

```
    <!-- 调度业务 -->
<bean id="jobDetail"
class="org.springframework.scheduling.quartz.MethodInvokingJobDetailFactoryBean">
 <property name="targetObject" ref="testJobTask" />
 <property name="targetMethod" value="service" />
</bean>
```

5.增加调用的触发器，触发的时间，有两种方式：

第一种触发时间，采用类似linux的cron，配置时间的表示发出丰富

```
  <bean id="cronTrigger"
class="org.springframework.scheduling.quartz.CronTriggerBean">
 <property name="jobDetail" ref="jobDetail" />
 <property name="cronExpression" value="10 0/1 * * * ?" />
</bean>
```

Cron表达式"10 */1 * * * ?"意为：从10秒开始，每1分钟执行一次

第二种，采用比较简话的方式，申明延迟时间和间隔时间

```
  <bean id="taskTrigger"
class="org.springframework.scheduling.quartz.SimpleTriggerBean">
 <property name="jobDetail" ref="jobDetail" />
 <property name="startDelay" value="10000" />
 <property name="repeatInterval" value="60000" />
</bean>
```

延迟10秒启动，然后每隔1分钟执行一次

6.开始调用

```
    <!-- 设置调度 -->
<bean class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
 <property name="triggers">
  <list>
   <ref bean="cronTrigger" />
  </list>
 </property>
 <property name="taskExecutor" ref="executor" />
</bean>
```

7.结束：启动容器即可，已经将spring和quartz结合完毕。

Cron常用的表达式

"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

三：quartz原理

根据上面spring的配置，我们就比较清楚quartz的内部情况，下面我们主要详解配置涉及到的每个点

1.我们先从最后一个步骤看起，SchedulerFactoryBean，scheduler的工厂实现，里面可以生产出对应的多个jobDetail和trigger，每个jobDetail对应trigger代表一个任务

Quartz的SchedulerFactory是标准的工厂类，不太适合在Spring环境下使用。此外，为了保证Scheduler能够感知 Spring容器的生命周期，完成自动启动和关闭的操作，必须让Scheduler和Spring容器的生命周期相关联。以便在Spring容器启动后， Scheduler自动开始工作，而在Spring容器关闭前，自动关闭Scheduler。为此，Spring提供SchedulerFactoryBean，这个FactoryBean大致拥有以下的功能：

1)以更具Bean风格的方式为Scheduler提供配置信息；
2)让Scheduler和Spring容器的生命周期建立关联，相生相息；
3)通过属性配置部分或全部代替Quartz自身的配置文件。

2.jobDetail,表示一个可执行的业务调用

3.trigger:调度的时间计划，什么时候，每隔多少时间可执行等时间计划

4.ThreadPoolTaskExecutor，线程池，用来并行执行每个对应的job，提高效率，这也是上面提到不推荐使用jdk自身timer的一个很重要的原因