特别说明

使用 Quartz 调度器

Quartz 调度器为调度工作提供了更丰富的支持。和 Java 定时器一样，可以使用 Quartz 来每隔多少毫秒执行一个工作。但 Quartz 比 Java Timer 更先进之处在于它允许你调度一个工作在某个特定的时间或日期执行。

创建一个类来定义工作

定义 Quartz 工作的第一步是创建一个类来定义工作。要做到这一点，你需要从 Spring 的 QuartzJobBean 中派生子类，如程序清单 7.3 所示：

程序清单 7.3 定义一个 Quartz 工作 public class EmailReportJob extends QuartzJobBean {

```
public EmailReportJob() {}


    protected void executeInternal(JobExecutionContext context)

     throws JobExecutionException {

courseService.sendCourseEnrollmentReport();
    }


        private CourseService courseService;

        public void setCourseService(CourseService courseService) {

            this.courseService = courseService;

        }

}

<bean id="reportJob"

            class="org.springframework.scheduling.quartz.JobDetailBean"
>

        <property name="jobClass">

            <value>xxx.xxx.EmailReportJob</value>

        </property>
```

```
<property name="jobDataAsMap">
    <map>
     <entry key="courseService">
        <ref bean="courseService"/>
     </entry>
        </map>
        </property>
</bean>
```

值得注意的是，在这里你并没有直接声明一个 EmailReportJob Bean，而是声明了一个 JobDetailBean。这是使用 Quartz 时的一个特点。
JobDetailBean 是 Quartz 的 org.quartz.JobDetail 的子类，它要求通过 jobClass 属性来设置一个 Job 对象。

使用 Quartz 的 JobDetail 中的另一个特别之处是 EmailReportJob 的 courseService 属性是间接设置的。
JobDetail 的 jobDataAsMap 属性接受一个 java.util.Map，其中包含了需要设置给 jobClass 的各种属性。
在这里，这个 map 包含了一个指向 courseService Bean 的引用，它的键值为 courseService。
当 JobDetailBean 实例化时，它会将 courseService Bean 注入到 EmailReportJob 的 courseService 属性中。

调度工作

现在工作已经被定义好了，接下来你需要调度这个工作。
Quartz 的 org.quartz.Trigger 类描述了何时及以怎样的频度运行一个 Quartz 工作。
Spring 提供了两个触发器，SimpleTriggerBean 和 CronTriggerBean。你应该使用哪个触发器？让我们分别考察一下这两个触发器，首先从 SimpleTriggerBean 开始。

SimpleTriggerBean 与 ScheduledTimerTask 类似。你可以用它来指定一个工作应该以怎样的频度运行，以及（可选地）在第一次运行工作之前应该等待多久。例如，要调度报表工作每 24 小时运行一次，第一次在 1 小时之后开始运行，可以按照以下方式进行声明：

```
<bean id="simpleReportTrigger"
```

```
        class="org.springframework.scheduling.quartz.SimpleTriggerBean
">
    <property name="jobDetail">
      <ref bean="reportJob"/>
    </property>
    <property name="startDelay">
      <value>3600000</value>
    </property>
    <property name="repeatInterval">
      <value>86400000</value>
    </property>
</bean>
```

属性 jobDetail 装配了将要被调度的工作，在这个例子中是 reportJob Bean。属性 repeatInterval 告诉触发器以怎样的频度运行这个工作（以毫秒作为单位）。这里，我们设置它为 86400000，因此每隔 24 小时它会被触发一次。你也可以选择设置 startDelay 属性来延迟工作的第一次执行。我们设置它为 3600000，因此在第一次触发之前它会等待 1 小时。

调度一个 cron 工作

尽管你可能认为 SimpleTriggerBean 适用于大多数应用，但它仍然不能满足发送注册报表邮件的需求。正如 ScheduledTimerTask，你只能指定工作执行的频度，而不能准确指定它于何时运行。因此，你无法使用 SimpleTriggerBean 在每天早晨 6:00 给课程主任发送注册报表邮件。

然而，CronTriggerBean 允许你更精确地控制任务的运行时间。如果你对 Unix 的 cron 工具很熟悉，则会觉得 CronTriggerBean 很亲切。你不是定义工作的执行频度，而是指定工作的准确运行时间（和日期）。例如，要在每天早上 6:00 运行报表工作，可以按照以下方式声明一个 CronTriggerBean：

```
<bean id="cronReportTrigger"
        class="org.springframework.scheduling.quartz.CronTriggerBean">
```

```
<property name="jobDetail">
  <ref bean="reportJob"/>
</property>
<property name="cronExpression">
  <value>0 0 6 * * ?</value>
    </property>
</bean>
```

和 SimpleTriggerBean 一样，jobDetail 属性告诉触发器调度哪个工作。这里我们又一次装配了一个 reportJob Bean。属性 cronExpression 告诉触发器何时触发。属性 cronExpression 告诉触发器何时触发。如果你不熟悉 cron，这个属性可能看上去有点神秘，因此让我们进一步考察一下这个属性。

一个 cron 表达式有至少 6 个（也可能是 7 个）由空格分隔的时间元素。从左至右，这些元素的定义如下：

1. 秒（0－59）

2. 分钟（0－59）

3. 小时（0－23）

4. 月份中的日期（1－31）

5. 月份（1－12 或 JAN－DEC）

6. 星期中的日期（1－7 或 SUN－SAT）

7. 年份（1970－2099）

每一个元素都可以显式地规定一个值（如 6），一个区间（如 9-12），一个列表（如 9，11，13）或一个通配符（如*）。“月份中的日期”和“星期中的日期”这两个元素是互斥的，因此应该通过设置一个问号（？）来表明你不想设置的那个字段。

一些cron表达式的例子

表 达 式	意 义
0 0 10,14,16 * * ?	每天上午10点，下午2点和下午4点
0 0,15,30,45 * 1-10 * ?	每月前10天每隔15分钟
30 0 0 1 1 ? 2012	在2012年1月1日午夜过30秒时
0 0 8-5 ? * MON-FRI	每个工作日的工作时间

3，启动工作

Spring 的 SchedulerFactoryBean 是 Quartz 中与 TimerFactoryBean 等价的类。按照如下方式在 Spring 配置文件中声明它：

```
<bean
class="org.springframework.scheduling.quartz.SchedulerFactoryBean">

    <property name="triggers">

      <list>

        <ref bean="cronReportTrigger"/>

<ref bean="simpleReportTrigger"/>

      </list>

    </property>

</bean>
```

属性 triggers 接受一组触发器。由于目前只有一个触发器，因此只需简单地装配一个包含 cronReportTrigger Bean 的一个引用的列表即可。

现在，你已经实现了调度发送注册报表邮件的需求。但在这个过程中，你做了一些额外的工作。在开始新的话题之前，首先让我们看一下如何通过更简单一些的方式调度报表邮件。

附表：

"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