<http://dwf07223.blog.51cto.com/8712758/1557145>

注解@Scheduled 可以作为一个触发源添加到一个方法中，例如，以下的方法将以一个固定延迟时间5秒钟调用一次执行，这个周期是以上一个调用任务的**完成时间**为基准，在上一个任务完成之后，5s后再次执行：

|  |  |
| --- | --- |
| 1  2  3  4 | @Scheduled(fixedDelay=5000)  public void doSomething() {      // something that should execute periodically  } |

    如果需要以固定速率执行，只要将注解中指定的属性名称改成fixedRate即可，以下方法将以一个固定速率5s来调用一次执行，这个周期是以上一个任务**开始时间**为基准，从上一任务开始执行后5s再次调用：

|  |  |
| --- | --- |
| 1  2  3  4 | @Scheduled(fixedRate=5000)  public void doSomething() {      // something that should execute periodically  } |

    对于固定延迟和固定速率的任务，可以指定一个初始延迟表示该方法在第一被调用执行之前等待的毫秒数：

|  |  |
| --- | --- |
| 1  2  3  4 | @Scheduled(initialDelay=1000, fixedRate=5000)  public void doSomething() {      // something that should execute periodically  } |

    如果简单的定期调度不能满足，那么cron表达式提供了可能。例如，下面的方法将只会在工作日执行：

|  |  |
| --- | --- |
| 1  2  3  4 | @Scheduled(cron="\*/5 \* \* \* \* MON-FRI")  public void doSomething() {      // something that should execute on weekdays only  } |

    还可以通过使用zone属性来指定cron表达式被调用的时区。

**注意：**

　　1、spring的注解@Scheduled  需要写在实现方法上；

　　2、定时器的任务方法不能有返回值（如果有返回值，spring初始化的时候会告诉你有个错误、需要设定一个proxytargetclass的某个值为true），不能指向任何的参数；

　　3、如果该方法需要与应用程序上下文的其他对象进行交互，通常是通过依赖注入来实现；

　　4、实现类上要有组件的注解@Component。

参考文档：<http://docs.spring.io/spring/docs/current/spring-framework-reference/htmlsingle/#scheduling-annotation-support-scheduled>