## Spring异步任务处理，@Async的配置和使用

参考博客：<https://blog.csdn.net/clementad/article/details/47403185>

这个注解用于标注某个方法或某个类里面的所有方法都是需要异步处理的。被注解的方法被调用的时候，会在新线程中执行，而调用它的方法会在原来的线程中执行。这样可以避免阻塞、以及保证任务的实时性。适用于处理log、发送邮件、短信……等。

**注解的应用范围：**

* 类：表示这个类中的所有方法都是异步的
* 方法：表示这个方法是异步的，如果类也注解了，则以这个方法的注解为准

**相关的配置：**

**<task:annotation-driven />配置：**

* **executor**：指定一个缺省的executor给@Async使用。

例子：

<task:annotation-driven executor="asyncExecutor" />

**<task:executor />配置参数：**

* **id**：当配置多个executor时，被@Async("id")指定使用；也被作为线程名的前缀。
* **pool-size**：
  + **core size**：最小的线程数，缺省：1
  + **max size**：最大的线程数，缺省：Integer.MAX\_VALUE
* **queue-capacity**：当最小的线程数已经被占用满后，新的任务会被放进queue里面，当这个queue的capacity也被占满之后，pool里面会创建新线程处理这个任务，直到总线程数达到了max size，这时系统会拒绝这个任务并抛出TaskRejectedException异常（缺省配置的情况下，可以通过rejection-policy来决定如何处理这种情况）。缺省值为：Integer.MAX\_VALUE
* **keep-alive**：超过core size的那些线程，任务完成后，再经过这个时长（秒）会被结束掉
* **rejection-policy**：当pool已经达到max size的时候，如何处理新任务
  + ABORT（缺省）：抛出TaskRejectedException异常，然后不执行
  + DISCARD：不执行，也不抛出异常
  + DISCARD\_OLDEST：丢弃queue中最旧的那个任务
  + CALLER\_RUNS：不在新线程中执行任务，而是有调用者所在的线程来执行

<!-- 缺省的异步任务线程池 -->

<task:annotation-driven executor="asyncExecutor" />

<task:executor id="asyncExecutor" pool-size="100-10000" queue-capacity="10" />

<!-- 处理log的线程池 -->

<task:executor id="logExecutor" pool-size="15-1000" queue-capacity="5" keep-alive="5"/>

@Override

@Async("logExecutor") //如果不指定名字，会使用缺省的“asyncExecutor”

public void saveUserOpLog(TabUserOpLog tabUserOpLog) {

userOpLogDAO.insertTabUserOpLog(tabUserOpLog);

}