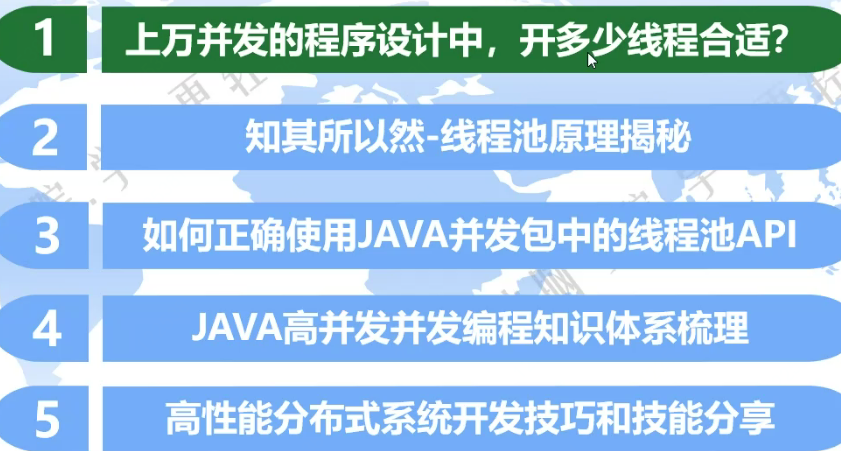
Java并发编程之多线程



问题1：用多线程的目的是什么？

**充分利用CPU资源，并发做多件事。**

问题2：单核CPU机器上适不适合多线程?

适合，如果是单线程，线程中需要等待IO时，此时CPU就空闲出来了。

问题3：线程什么时候回让出CPU?<面试题目>

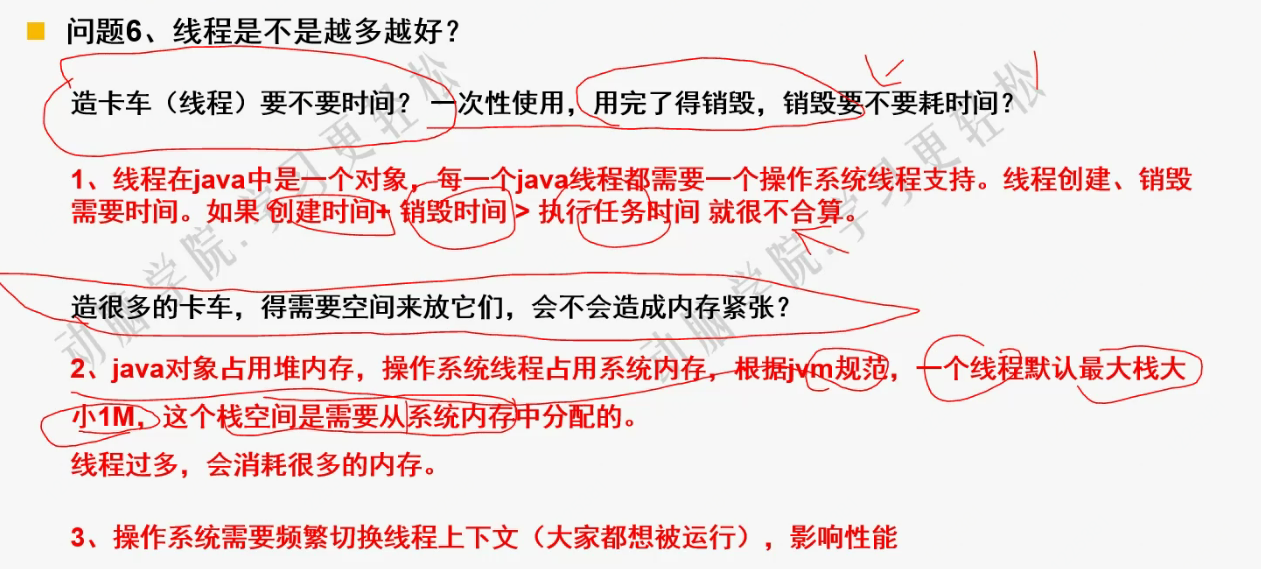
阻塞时wait、await、等待IO、sleep、yield、结束了。

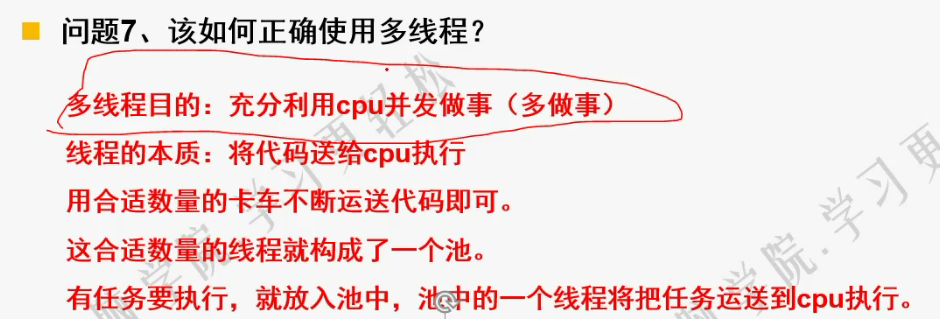
sleep：直接睡觉，不再等待；

yield：放弃当前的执行权或放弃排队，继续等待调度，有可能再次进入队列中。

join的区别。

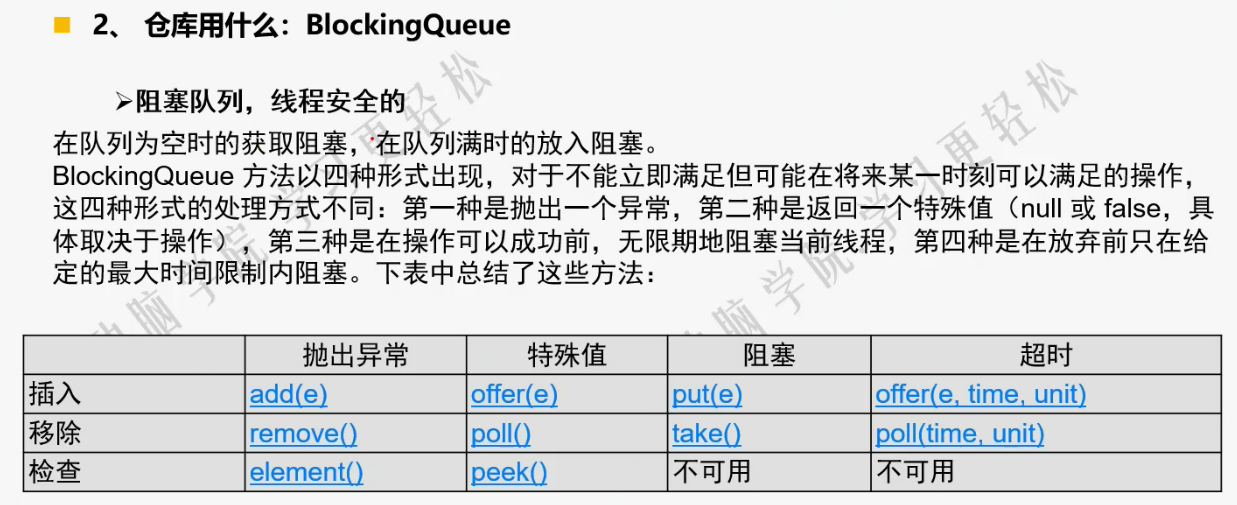








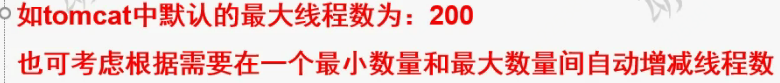




问题8：如何确定合适数量的线程？

1. 如果是计算型任务：CPU数量的1-2倍；
2. 如果是IO型任务：则需要多一些线程，要根据具体的IO阻塞时间长度进行考量决定。

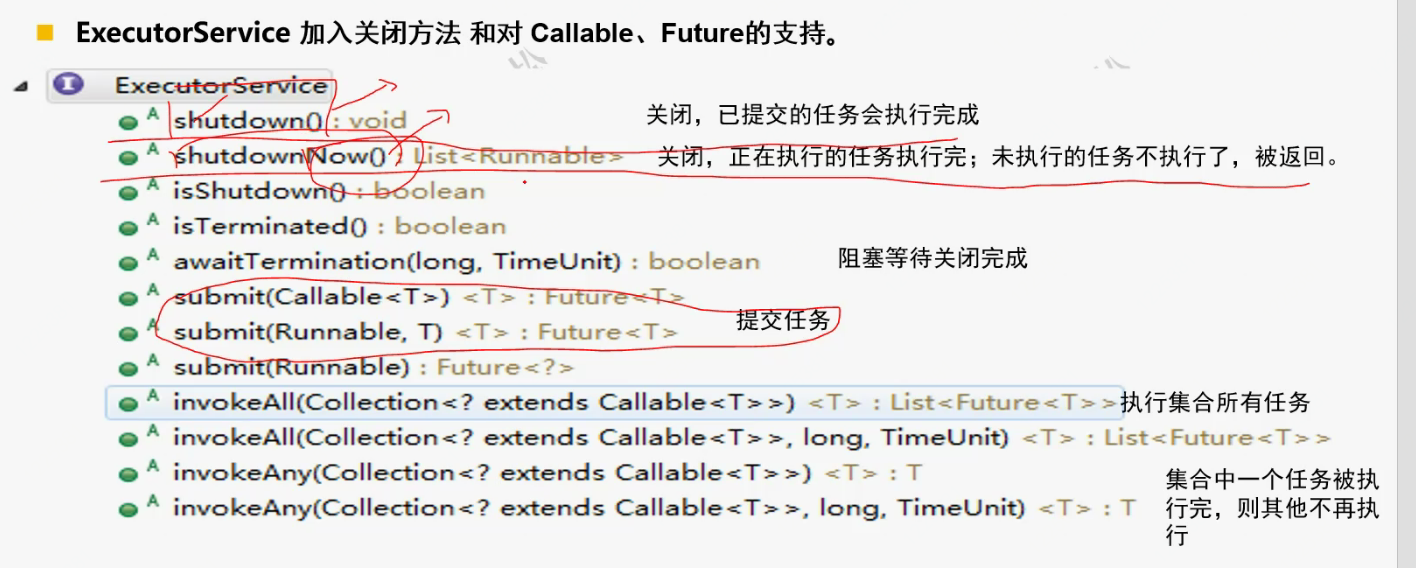
如tomcat中默认的最大线程数为200；





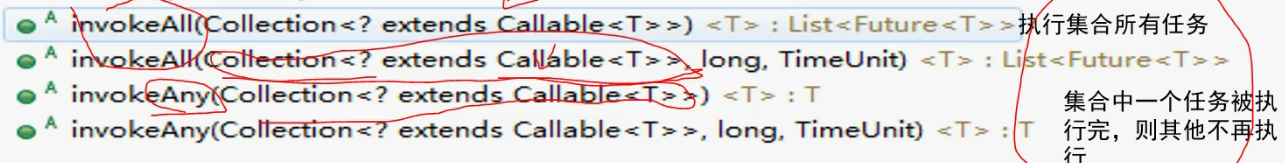


**学习下Spring的定时任务功能。。。**



注意区分shutdown与shutdownNow方法的区别：

isTerminated只有首先调用了shutdown或shutdownNow才会返回true。

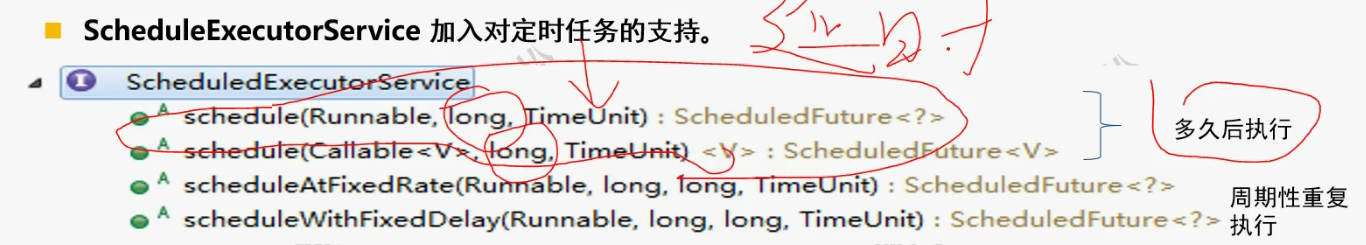


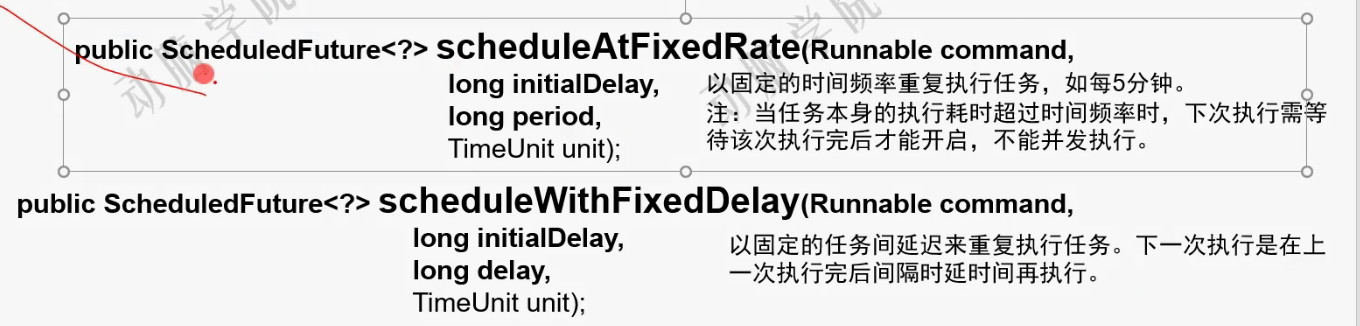
invokeAll表示执行所有，invokeAny执行一个。

execute与submit的区别：

submit有返回值，返回一个Future对象，可以方便处理异常；

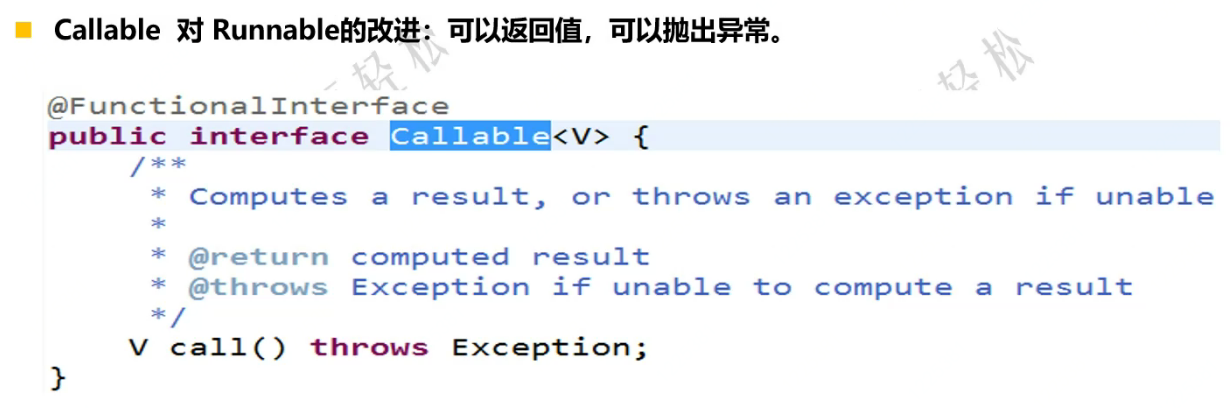
execute没有返回值，不方便处理异常。

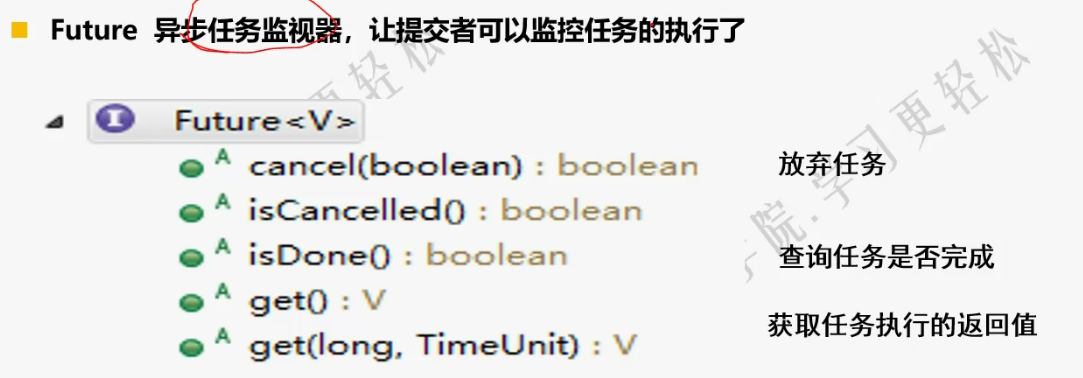




上一个开始到下一个开始，时间间隔是固定的period，即scheduleAtFixedRate

上一个结束到下一个开始，时间间隔是固定的delay，即scheduleWithFixedDelay





对Future的理解：

异步任务的监视器，形象生动。

线程池对任务的调度是异步的，一个任务被调度后，返回一个Future对象；

Future可以通过isDone判断是否执行完毕，利用get方法获取返回值；

get方法是阻塞方法，因此，如果一直无法返回结果，可以调度cancel取消该任务的执行。



阻塞队列的选择：

ArrayBlockingQueue

小于核心数，就新建；如果大于核心线程数，但是阻塞队列没有满，就塞入阻塞队列；

只有阻塞对列满了，才会创建新的线程。

LinkedBlockingQueue

阻塞队列无限大，因此maxPoolSize失效。

SynchronousQueue

使用要谨慎。 来一个任务就创建一个线程执行。

DelayQueue

所有元素都必须实现Delayed接口，传进去的任务，都要先实现。接到任务的时候，首先先入队，只有达到了延时时间才执行。主要用于ScheduledThreadPool。

