# 异步任务

1. 线程的四种使用方式
   1. 1. 继承Thread类
   2. 2. 实现Runnable接口
   3. 3. 实现Callable接口 + FutureTask
      1. 可以返回结果，但是不能控制线程资源
   4. 4. 线程池
      1. 如果一个项目中来一个就创建线程，来一个就创建一个，内存本身是有限的。会导致内存崩溃，所以引入线程池来控制线程资源，性能稳定
      2. 日常开发中也是会使用线程池的
2. 线程池的七个参数
   1. corePoolSize[5]核心线程数
      1. 相当于就是new5个Thread等待任务来启动，线程池创建好之后就准备好的线程数（不会销毁的，除非设置了某个参数名字忘了）
   2. maximumPoolSize[200] 最大线程数,控制资源的一个玩意
      1. 就算来了100万请求，也只能有200个正在运行
   3. keepAliveTime：存活时间
      1. 要是当前的线程数大于（最大线程数-核心线程数）的数量，就会释放空闲的线程
   4. unit：时间单位
   5. workQueue：阻塞队列
      1. 假设现在有1000请求，最大线程数为500，那么剩下的500就会被放进队列中，一旦线程空闲，就会去队列中取出新的任务执行
   6. threadFactory：线程的创建工厂
   7. handler：拒绝策略
      1. 如果队列满了，就会按照指定的拒绝策略进行执行
3. 线程池的执行流程
   1. 1. 首先线程池创建，准备好core个核心线程数，准备接受任务啦
   2. 2.如果core满了，再进来的就会放到阻塞队列里去，空闲的core就会自己去阻塞队列获取任务执行
   3. 3. 阻塞队列满了的话，就会开新线程去执行，但是只能开到max指定的数量
   4. 4. max满了就用拒绝策略拒绝任务
   5. 5. max都执行完了，有很多空闲线程就在指定的存活时间之后，释放叼这些线程
4. 为什么要使用线程池
   1. 降低资源的消耗：它不需要就是频繁的创建和销毁线程了
   2. 提高响应速度：就拿单核CPU来说，并发执行其实都是假象，只有抢到CPU时间片的才能真正的执行，线程池就刚刚好限制了线程个数嘛
5. 异步任务的好处
   1. 假设一个接口里有ABC三个方法，C必须需要A的结果才能执行，那么B只能干等着，如果用了异步任务那么，就可以B先执行
   2. 使用异步任务：需要自己写一个线程池配置类，然后将那些参数配置到yml中，然后CompleTableFutrue(实现了Futrue所以可以获取结果)有四个静态方法去异步操作，其中runXXX的是无结果的，supplyXXX的是可以获取返回结果的，最后需要有个get方法，就是等他们全部执行完