任务执行

- 在线程中执行任务
  - 找出清晰的任务边界
  - 串行的执行任务
  - 显式的为任务创建线程
  - 无限制创建线程的不足
    - 线程生命周期的开销非常高
    - 资源消耗
    - 稳定性
- Excutor框架
  - 执行策略
    - what 在什么线程中执行任务
    - what 按照什么顺序执行(FIFO、LIFO、优先级?)
    - How many 有多少个任务可以并发执行
    - How many 有多少任务在等待执行
    - Which 如果由于过载需要拒绝一个任务,应该选哪一个?如何通知应用程序有任务被拒绝?
    - What 在执行一个任务之前,应该进行哪些动作
  - 线程池
    - 管理一组同构工作线程的资源池
    - Executors工厂方法
      - newFixedThreadPool
      - newCachedThreadPool
      - newSingleThreadExecutor
      - newScheduledThreadPool
    - Executor的生命周期
      - ExecutorService
        - shutdown

        - shutdownNow
        - isShutdown
        - isTerminated
        - awaitTermination
    - 延迟任务与周期任务
        - ScheduledThreadPoolExecutor
        - DelayQueue
- 找出可利用的并行性
    - Callable与Future
    - CompletionService
        - Executor
        - BlockingQueue
        - CompletionService
    - 为任务设置时限
        - Future.get
        - ExecutorService.invokeAll
- 小结
    - Executor框架将任务提交与执行策略解偶
    - 支持不同类型的执行策略
    - 想要分解为不同任务时获取最大的好处,必须定义清晰的任务边界
    - 通过分析发现更细粒度的并行性