.NET 多线程

- 异步 Asynchronous (21)
- 常见任务 (19)
- 线程 Thread
  - 基本概念
    - 什么是线程?
      - 线程是操作系统中能够独立运行的最小单位，也是程序中能够并发执行的一段指令序列
      - 线程是进程的一部分，一个进程可以包含多个线程，这些线程共享进程的资源
      - 进程有入口线程，也可以创建更多的线程
    - 为什么要多线程?
      - 批量重复任务希望同时进行（比如对于数组中的每个元素都进行相同且耗时的操作）
      - 多个不同任务希望同时进行，互不干扰（比如有多个后台线程需要做轮询等操作）
    - 什么是线程池?
      - 一组预先创建的线程，可以被重复使用来执行多个任务
      - 避免频繁地创建和销毁线程，从而减少了线程创建和销毁的开销，提高了系统的性能和效率
      - 异步编程默认使用线程池
    - 什么是线程安全?
      - 线程安全
        - 多个线程访问共享资源时，对共享资源的访问不会导致数据不一致或不可预期的结果
      - 同步机制
        - 用于协调和控制多个线程之间执行顺序和互斥访问共享资源
        - 确保线程按照特定的顺序执行，避免竞态条件和数据不一致的问题
      - 原子操作
        - 在执行过程中不会被中断的操作。不可分割，要么完全执行，要么完全不执行，没有中间状态
        - 在多线程环境下，原子操作能够保证数据的一致性和可靠性，避免出现竞态条件和数据竞争的问题
    - 常用实现方式
      - 线程
      - 线程池
      - 异步编程
      - 考虑一下自带方法?
        - Parallel
          - For、ForEach、Invoke
        - PLINQ
          - AsParallel、AsSequential
          - AsOrdered
  - 线程的创建
    - 创建 Thread 实例，并传入 ThreadStart 委托
      - 还可以配置线程，如是否为后台线程
    - 调用 Thread.Start 方法，还可以传参
  - 线程的终止
    - 调用 Thread.Join 方法，等待线程的结束
    - 调用 Thread.Interrupt 方法，中断线程的执行
      - 会在相应线程中抛出 ThreadInterruptedException
      - 如果线程中包含一个 while(true) 循环，那么需要保证包含等待方法，如IO操作，Thread.Sleep 等
    - 不能用 Abort?
      - 使用 Abort 方法来强制终止线程可能导致一些严重的问题，包括资源泄漏和不可预测的行为
      - 较新版本的 .NET 中如果使用这个方法，会报 PlatformNotSupportedException
      - 推荐使用 Thread.Interrupt 或 CancellationToken
  - 线程的挂起与恢复
    - Thread.Suspend 以及 Thread.Resume
    - 较新版本的 .NET 中，这两个方法已经被标记为 Obsolete，且调用会报错
    - 推荐使用锁、信号量等方式实现这一逻辑
  - 线程的超时
    - Join 方法拥有 Timeout 入参，并会在超时后返回 false
    - 可以在这种情况下考虑使用 Interrupt 或 CancellationToken 的方式终止一个线程
- 线程安全与同步机制 Thread-Safety
  - 原子操作
    - Interlocked
  - 锁与信号量
    - lock & Monitor
    - Mutex
    - Semaphore
    - WaitHandle
      - ManualResetEvent
      - AutoResetEvent
    - ReaderWriterLock
  - 轻量型
    - SemaphoreSlim
    - ManualResetEventSlim
    - ReaderWriterLockSlim
  - 不要自己造轮子!
    - 线程安全的单例：Lazy (1)
    - 线程安全的集合类型：ConcurrentBag、ConcurrentStack、ConcurrentQueue、ConcurrentDictionary
    - 阻塞集合：BlockingCollection
    - 通道：Channel
    - 原子操作：Interlocked
    - 周期任务：PeriodicTimer