* 不同应用在各自独立的进程中运行。
  + 当应用以任何形式启动时，系统为其创建进程，该进程将持续运行
  + 当进程完成当前任务处于等待状态，且系统资源不足时，系统自动回收
* 主线程：在启动应用时，系统会为该应用创建一个称为“主线程”的执行线程
* 默认情况下，所有的操作都在主线程上执行（如果需要执行比较耗时的任务（如下载文件、查询数据库），可创建其他线程来处理）
* UI界面的显示和更新等操作，都是在主线程上进行，所以主线程又称UI线程
* 为保证应用有更好的响应性，需要设计任务的优先级，优先级有以下三种
  + HIGH
  + DEFAULT
  + LOW
* 在UI线程上运行的任务默认以高（HIGH）优先级运行，如果某个任务无需等待结果，也可以用低优先级

# TaskDispatcher线程框架

* 为什么需要线程管理框架？
  + 因为如果应用的业务逻辑比较复杂，可能需要创建多个线程来执行多个任务
  + 多任务与多线程的交互较为繁杂时，代码复杂难以维护。要解决此问题，开发者可以使用TaskDispatcher 来分发不同的任务
* TaskDispatcher是一个任务分发器，它是Ability分发任务的基本接口，隐藏任务所在线程的实现细节
* TaskDispatcher 具有多种实现，每种实现对应不同的任务分发器（在分发任务时可以指定任务的优先级，由同一个任务分发器分发出的任务具有相同的优先级）
* 系统提供的任务分发器有以下几种
  + GlobalTaskDispatcher 全局并发任务分发器
    - 一个应用只有一个GlobalTaskDispatcher，它在程序结束时才被销毁
  + ParallelTaskDispatcher 并发任务分发器
  + SerialTaskDispatcher 串行任务分发器
  + SpecTaskDispatcher 专有任务分发器
    - 绑定到专有线程上的任务分发器
    - 目前已有的专有任务分发器为：UITaskDispatcher，绑定到应用主线程的专有任务分发器
    - 由该分发器分发的所有的任务都在主线程上按顺序执行，它在应用程序结束时被销毁

# 任务派发方法

* TaskDispatcher框架提供了几种主要的任务派发方法
  + syncDispatcher() 同步派发任务
  + asyncDispatcher() 异步派发任务：派发任务，并立即返回，返回值是一个可用于取消任务的接口
  + delayDispatcher() 异步延迟派发任务
  + group() 成组派发任务：该组任务之间有一定的联系

# 线程间通信

* 在开发过程中，开发者经常需要在当前线程中处理下载任务等较为耗时的操作，但是又不希望当前的线程受到阻塞。此时，就可以使用EventHandler机制
* EventHandler是HarmonyOS用于处理线程间通信的一种机制
* EventHandler的主要功能是将InnerEvent事件或Runnable任务投递到其他的线程进行处理
* 使用EventHandler需要创建EventHandler的子类，并重写processEvent方法来处理事件
* 每一个EventHandler和指定的EventRunner所创建的新线程绑定，并且该新线程内部有一个事件队列。EventHandler可以投递指定的InnerEvent事件或Runnable任务到这个事件队列。EventRunner从事件队列里循环地取出事件
* 一个EventHandler只能同时与一个EventRunner绑定，一个EventRunner上可以创建多个EventHandler（主线程默认有一个EventRunner）
* EventRunner
  + EventRunner的工作模式可以分为托管模式和手动模式
  + 两种模式是在调用EventRunner的create()方法时，通过选择不同的参数来实现的（默认为托管模式）
* 约束限制
  + 在进行线程间通信的时候，EventHandler只能和EventRunner所创建的线程进行绑定，EventRunner创建时需要判断是否创建成功，只有确保获取的EventRunner实例非空时，才可以使用EventHandler绑定EventRunner