# 案例背景

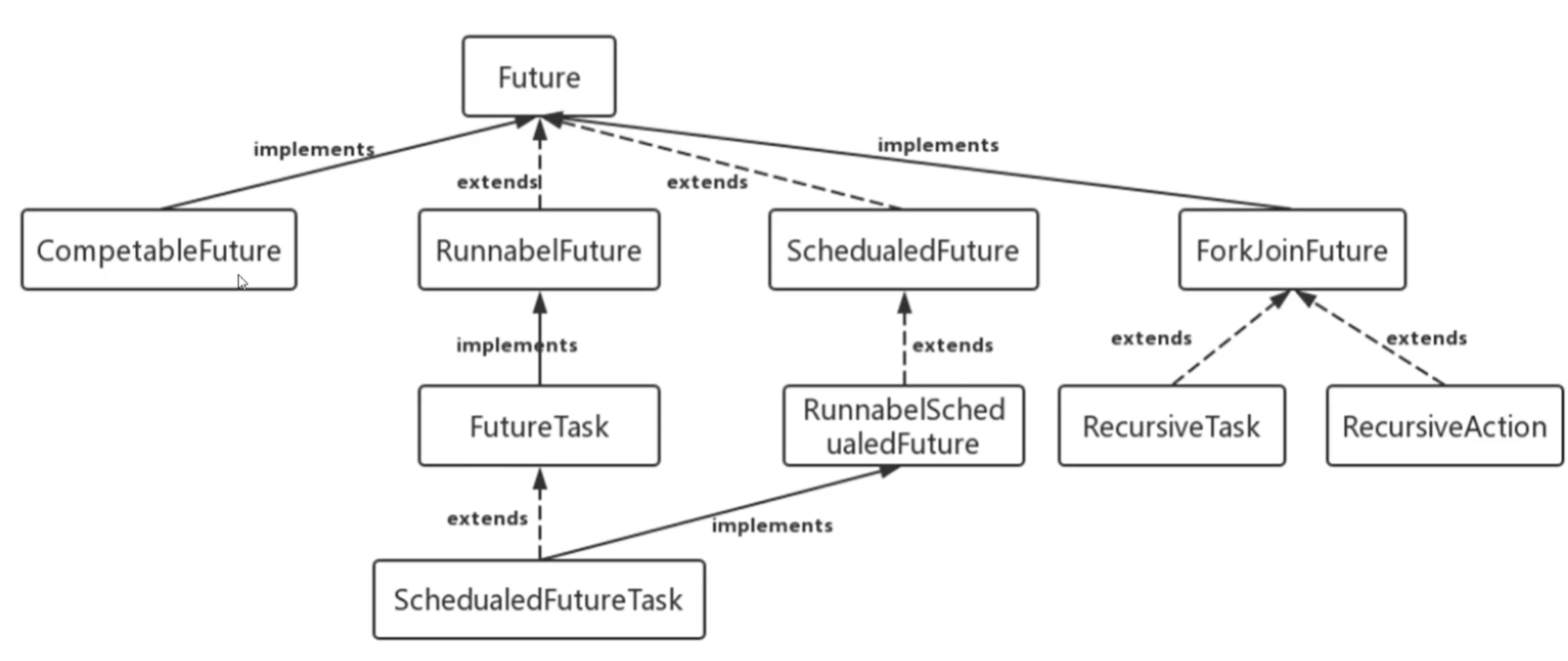
大型电商公司的支付聚合服务都有这类的场景:

1. 调用校验服务校验待生成的订单是否合法
2. 订单服务生成订单(校验服务和订单服务没有依赖关系)
3. 调用1和2,支付服务实现支付核心的功能
4. 结合步骤1至3完成支付服务的聚合调用

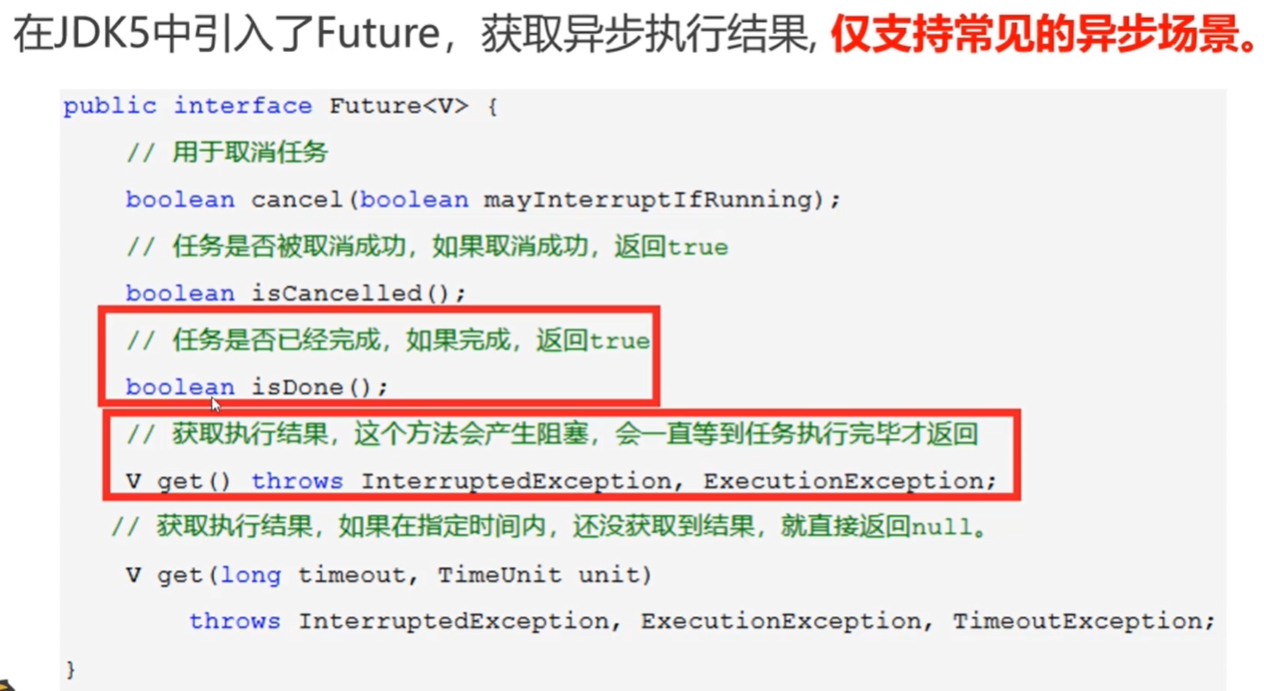
假如步骤1的耗时5秒,步骤2的耗时3秒,步骤3的耗时2秒,如果你是架构师,要求:

1. 请实现微服务的同步调用
2. 请实现微服务的异步调用(使用CompletableFuture实现)
3. 比较1和2的性能.

# Future类图



# Future的不足



Future直接表述多个Future结果之间的依赖性,有一定的缺陷:

1. 将两个异步计算合并为一个(第二个异步计算依赖于第一个的结果),这个用Future不太好实现.
2. 等待Future集合中的所有的任务都完成
3. 仅等待Future集合中最快结束的任务完成,并返回它的结果

# CompletionStage

用于异步执行中的阶段处理.

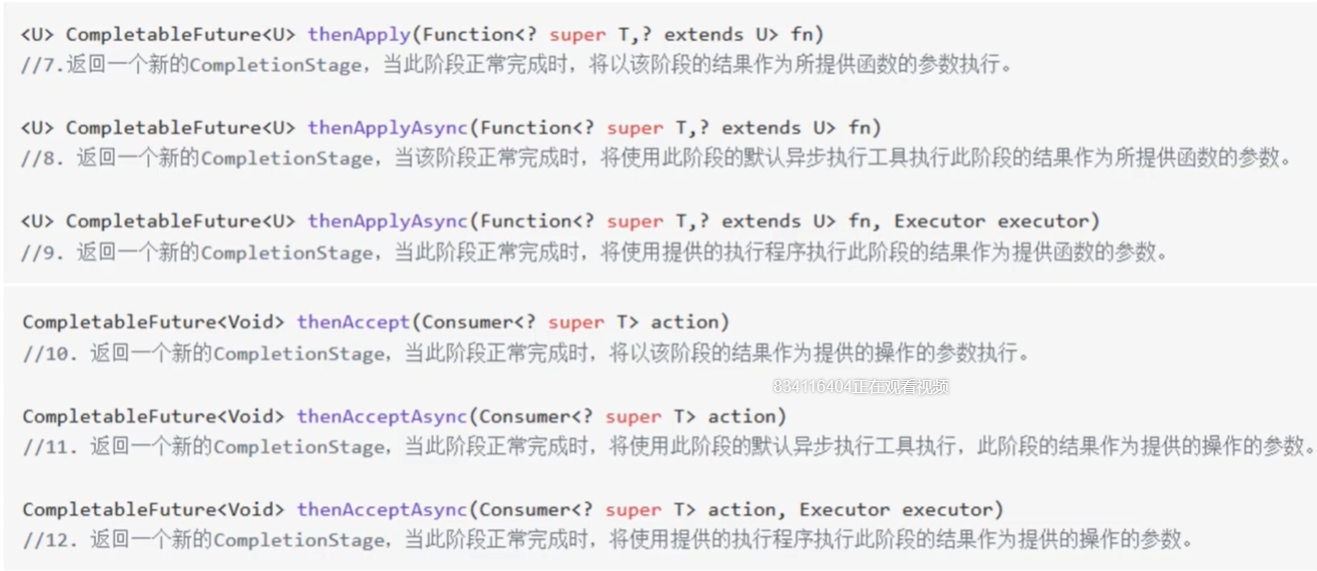
CompletionStage定义了一组接口用于在一个阶段执行结束之后,要么继续执行下一个阶段,要么对结果进行转换产生新的结果等等

# 核心用法

## 提交任务执行\获取执行结果



## 异步结果的处理



## 合并结果





# 代码使用方式

执行Test这里类就可以看到结果了.