CHAPTER 01

第二章 并发归集方案

并发归集方案

1 2.1 Future模式

2 2.2 CompletionService

3 2.3 CompletableFuture

2.1 Future模式

ForkJoin线程池

1 Future模式是什么？

2 Future模式时序图

3 Future模式的实现 FutureTask

4 Future模式 企业案例

Future模式是什么？

张三下馆子的故事

同步模式

1、张三去饭店吃饭，点完餐之后就站在出餐口等着，什么时候饭做好了再拿走用餐。

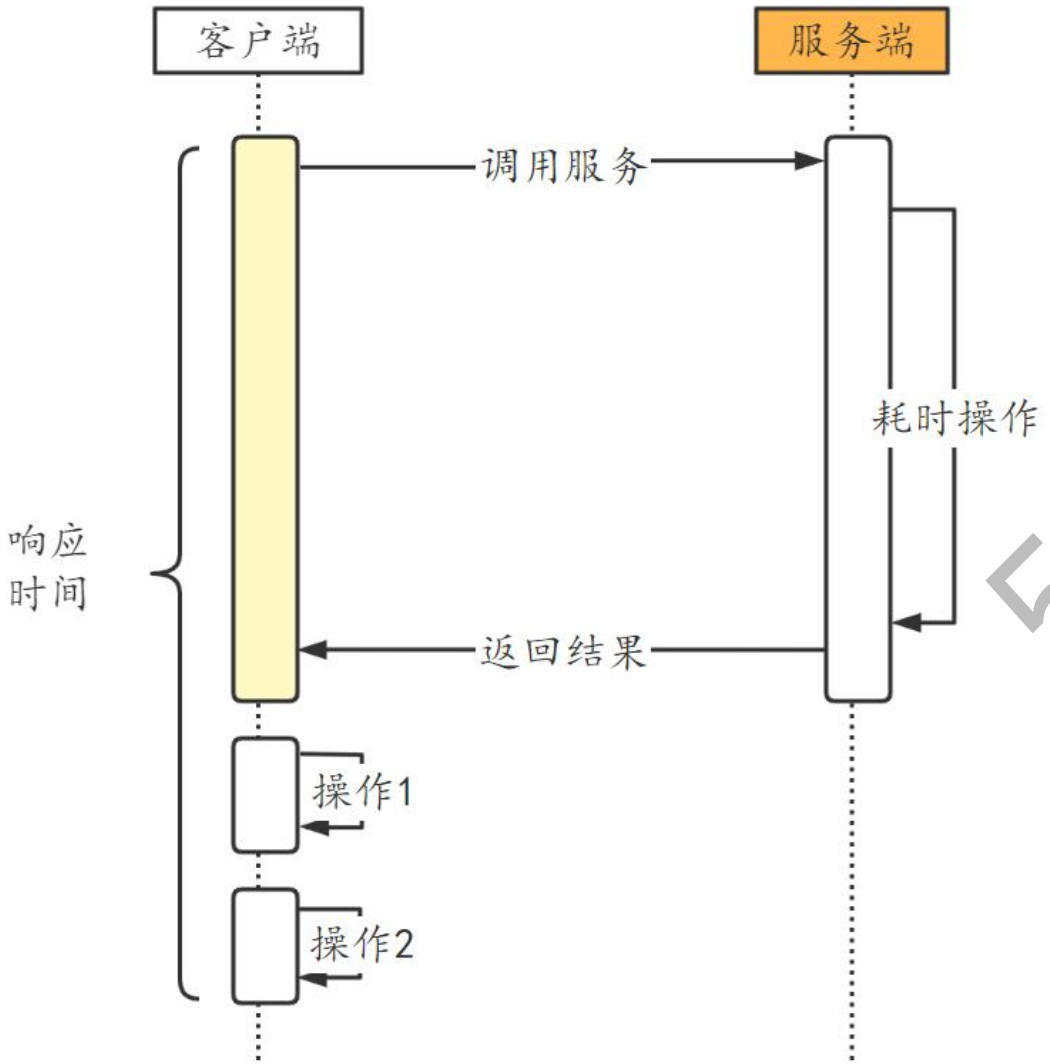
Future模式

2、张三去饭店吃饭，点完餐之后，服务员给了他一张小票。然后张三就隔壁奶茶店买了一杯奶茶，20分钟之后才回来。然后他拿出小票，到出餐口取餐。如果饭做好了就拿走用餐。如果饭还没做好则原地等待。

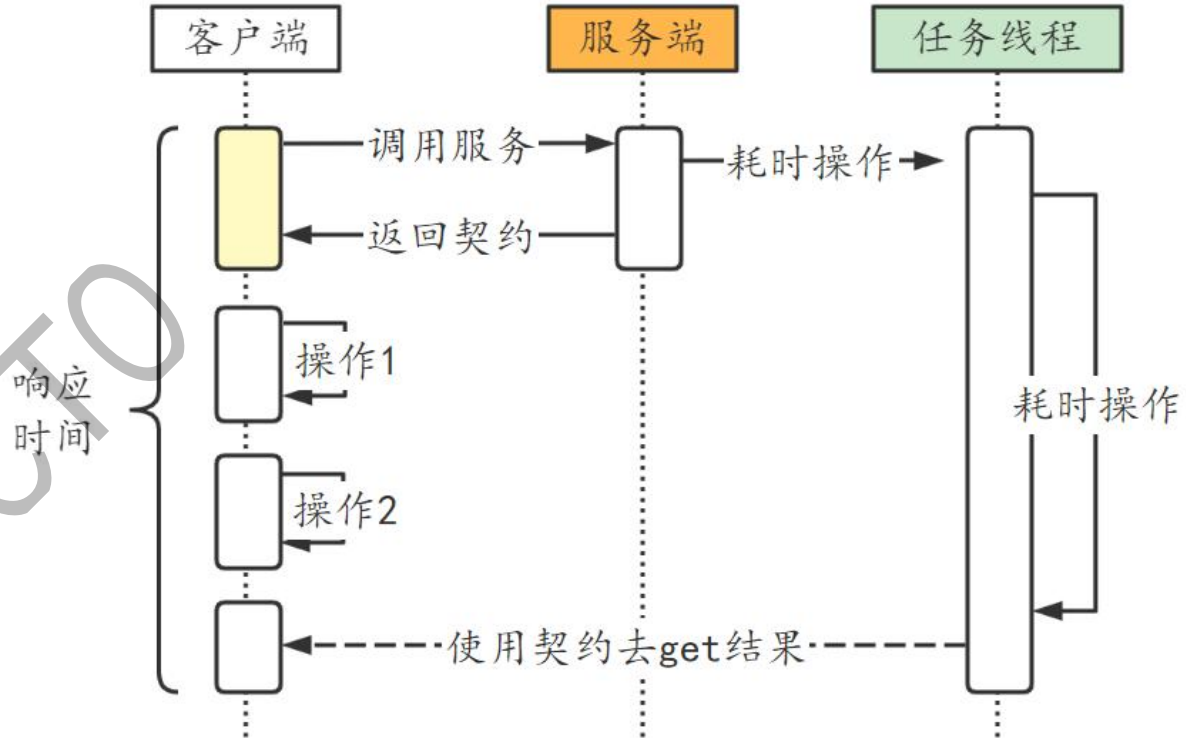
回调模式

3、张三去饭店吃饭，点完餐之后，然后张三就隔壁奶茶店买了一杯奶茶，又回到饭店坐着刷手机。什么时候饭做好了，服务员通知他。

Future模式时序图



常见的同步模式



Future异步模式

Future模式的实现 FutureTask

FutureTask实现了Future模式，提供最简单的Future实现。

弊端十分明显，要想获得异步的执行结果，只能轮询或者阻塞。

最佳实践

适合前后端同步类操作，缩短响应时间

让耗时的操作先使用FutureTask异步执行，然后去执行其他的同步操作。

最后再将FutureTask的异步执行的结果进行组装。

Future模式 企业案例

要求：实时返回营销员的业绩信息，最大限度缩短响应时间。

背景：订单总数和最后一笔订单的时间可以直接查询数据库获得，而实时佣金、单笔最高佣金、推荐佣金需要调用佣金系统的不同服务接口。

方案：思考一下？

2.2 CompletionService

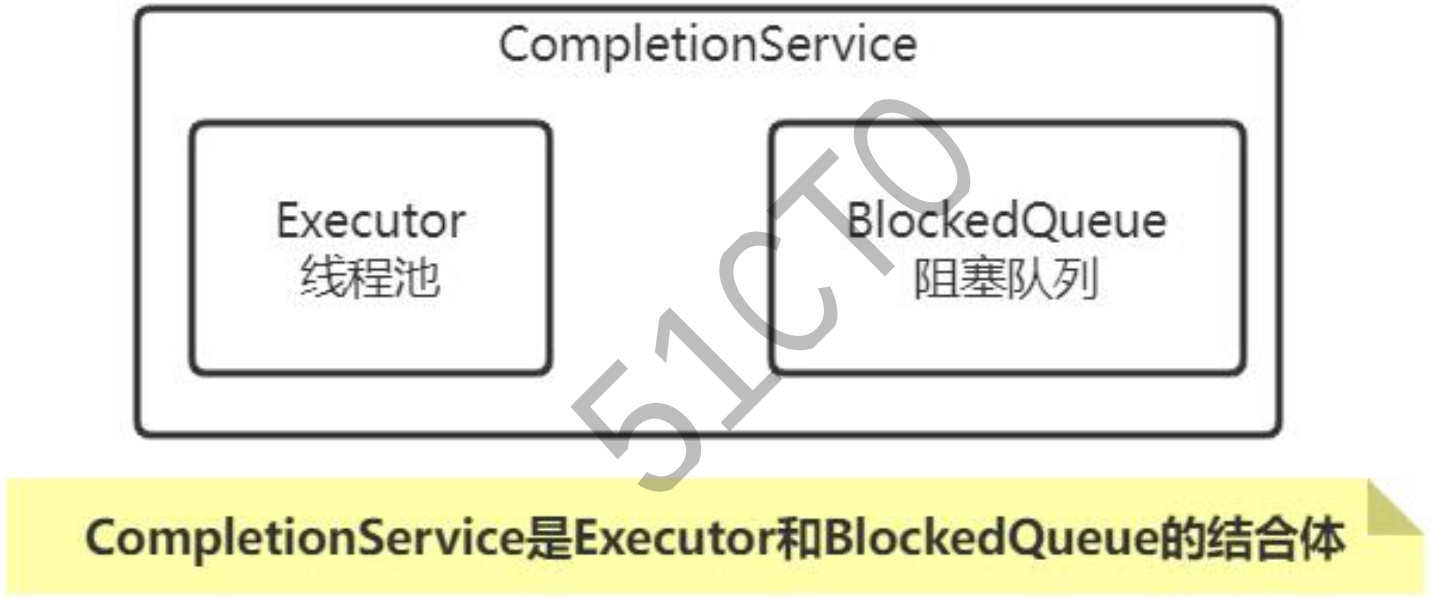
CompletionService

1 CompletionService 是什么

2 CompletionService 原理

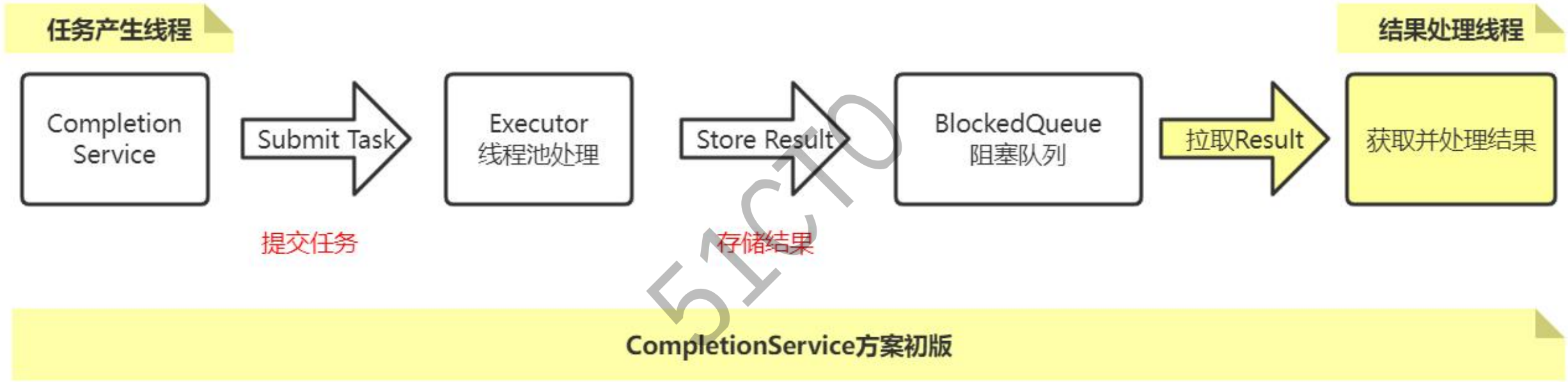
3 CompletionService 企业案例

CompletionService 是什么



★ CompletionService = 线程池 + 阻塞队列

CompletionService 原理



- 多个任务提交线程池处理，结果存储在阻塞队列中，谁先处理完就先获取谁的结果。

CompletionService 企业案例

订单编号	金额	完成时间	当前状态	结果顺序
A1001	100	12:31:33	处理成功	6
A1002	203	12:20:30	处理失败	3
A1003	214	12:10:22	处理成功	1
A1004	223	12:11:34	处理成功	2
A1005	10	12:34:44	处理失败	7
A1006	89	12:23:33	处理成功	5
A1007	423	12:22:11	处理失败	4
A1008	99	12:35:32	处理成功	8

需求：并行处理所有订单，每个订单对应多笔工单，因此每个订单的处理时长不确定。

要求：

1、实时获取订单处理进度
2、实时计算成功失败的笔数和金额
3、所有任务处理完成后发送邮件提醒
4、计算任务总耗时

方案： 思考一下？

2.3 CompletableFuture

CompletableFuture

1 为什么要使用 CompletableFuture

2 CompletableFuture 经典场景

3 CompletableFuture 核心方法

为什么要使用CompletableFuture

张三下馆子的故事

同步模式

1、张三去饭店吃饭，点完餐之后就站在出餐口等着，什么时候饭做好了再拿走用餐。

Future模式

2、张三去饭店吃饭，点完餐之后，服务员给了他一张小票。然后张三就隔壁奶茶店买了一杯奶茶，20分钟之后才回来。然后他拿出小票，到出餐口取餐。如果饭做好了就拿走用餐。如果饭还没做好则原地等待。

最佳解决方案

回调模式

3、张三去饭店吃饭，点完餐之后，然后张三就隔壁奶茶店买了一杯奶茶，又回到饭店坐着刷手机。什么时候饭做好了，服务员通知他。

为什么要使用CompletableFuture

缺点

Future -> FutureTask -> CompletionService 只能阻塞式的获取结果。

优点

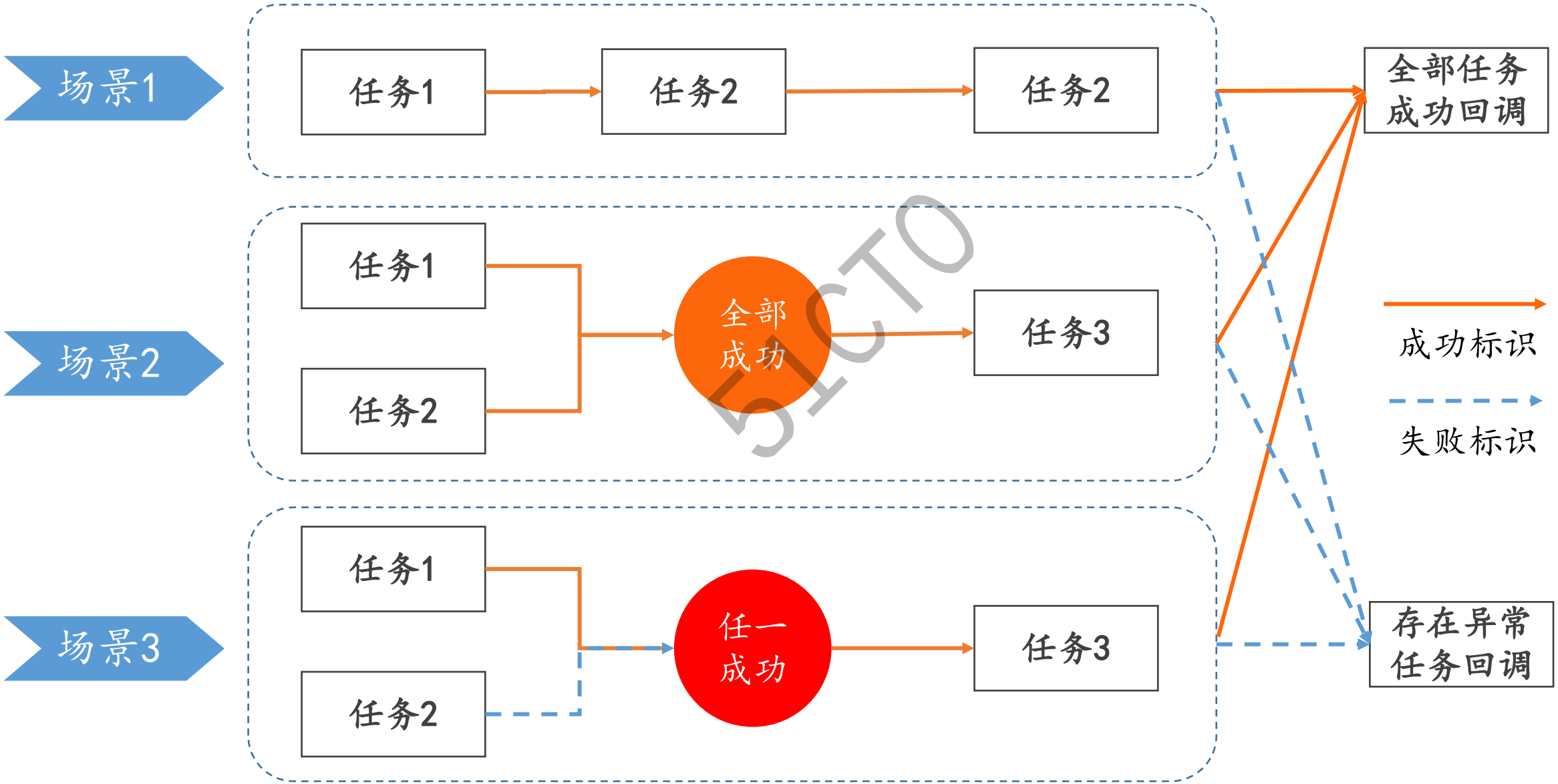
java.util.concurrent.CompletableFuture类

1、CompletableFuture 支持监听与异步回调

2、CompletableFuture 支持成功、失败聚合

3、CompletableFuture 支持串行、并行、组合逻辑

CompletableFuture 经典场景



CompletableFuture 核心方法

实例化	获取结果	异常捕获	后续操作	组合操作
supplyAsync runAsync 构造函数	get getNow	exceptionally	whenComplete whenCompleteAsync handle handleAsync thenApply thenApplyAsync thenAccept thenAcceptAsync	thenCombine thenCombineAsync thenCombine thenCombineAsync

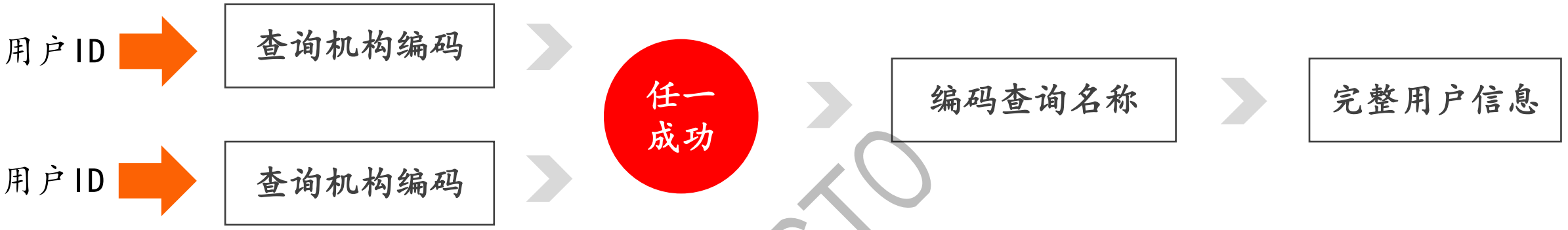
CompletableFuture 企业案例

1、根据用户ID，到组织机构服务中获取用户所在的机构编码。

2、根据第1步获取的机构编码再去查询机构名称。

3、将所有基本信息组装在一起，获得完整的用户信息。

CompletableFuture 企业案例



1、企业内有多个系统，都可以查询用户所属的机构编码，但是稳定性未知，为了保证高可用，可同时去两个服务去查询机构代码。

2、哪一个服务先拿到机构代码，都立即继续执行后续操作。使用第1步获取的编码，去查询机构名称。

CompletableFuture 企业案例

1、同时使用用户ID去两个系统查询权限编码、账户编码。

2、只有当两个查询都成功的时候，才进行后续操纵。

3、使用第1步返回的权限编码和账户编码，去查询账户余额。