

课程学习须知

并发编程课程大纲

- 并发编程的发展以及价值 (5月15号)
- 并发编程带来的挑战之同步锁 (5月16号)
- 并发编程带来的挑战之可见性 (5月19号)
- 并发安全性之Lock锁及原理分析 (5月22号)
- 线程阻塞唤醒wait,notify以及condition,死锁等原理分析 (5月23号)
- J.U.C并发工具集场景及原理分析 (5月26号)
- 随便聊聊ThreadLocal&ForkJoin (5月29号)
- 深度剖析阻塞队列的设计原理及实现 (5月30号)
- 并发安全的集合ConcurrentHashMap-1 (6月2号)
- 并发安全的集合ConcurrentHashMap-2 (6月5号)
- 站在架构的角度思考线程池的设计和原理 (6月6号)
- **Java8新的异步编程方式 (6月9号)**

注意：随着课程的推进，有可能会增加课时来讲解，所以如果遇到课程大纲的变化，我会及时通知！

新同学注意事项

并发编程是一个完整的体系，到目前为止已经讲了10节课了，如果前面部分内容没有听的同学，一定要记得补一下，不然越到后面越听不懂。

另外，有些同学基础比较薄弱点，对并发这块完全不熟悉，我强烈建议这些同学至少听2遍以上。因为并发编程整个体系是偏基础和底层的，和我们平时开发用的应用框架的理解完全不一样。

这就需要大家逐步去转变一些学习思维，从而更好的理解。

有部分学员反馈有问题找不到我，记得加我微信 **mic4096**，qq目前登录较少，有些消息回应不及时。

本阶段课程变化

- 新增了部分内容，如异步编程、ForkJoin等。
- 内容更加细致。
- 讲课节奏会稍微放缓，帮助大家更好的吸收。

本阶段内容的学习目标

并发编程是很多大公司面试重点考察的范围，据部分拿到高薪Offer的学员反馈，大部分面试中，并发编程很好的帮助他们拿到Offer。

所以今年，对于并发编程的内容做了调整，更加细致和全面。

大家学完之后要达到的目标。

- 很好的理解线程的本质。
- 能够灵活运用线程。
- 从原理层面理解并发编程。
- **在并发编程领域击败99%的程序员**

本周预习资料

本节课是并发编程的最后一节课，我花了12节课来讲解整个并发编程体系，几乎涵盖了97%以上内容。也就是说你去网上刷各种并发编程的面试题，你都能够清晰的回答出来。如果做不到，再去看一遍！

Callable/Future

在JDK1.5中，引入了Future的概念，它可以结合Callable接口来获得线程异步执行完成后的返回值，但是它在使用上存在一定的局限性（这个在后面会做详细介绍）。所以在JDK1.8中，引入了CompletableFuture这个组件，它在Future的基础上提供了更加丰富和完善的功能。在本章节中，会详细介绍Future/Callable的使用和原理，CompletableFuture的实际应用以及原理分析。

CompletableFuture

CompletableFuture针对Future做了改进，也就是在异步任务执行完成后，主线程如果需要依赖该任务执行结果继续后续操作时，不需再等待，而是可以直接传入一个回调对象，当异步任务执行完成后，自动调用该回调对象，相当于实现了异步回调通知功能。

除此之外，CompletableFuture还提供了非常强大的功能，比如对于回调对象的执行，可以放到非任务线程中执行，也能用任务线程执行；提供了函数式编程能力，简化了异步编程的复杂性；提供了多个CompletableFuture的组合与转化功能。

CompletableFuture方法说明

CompletableFuture提供了四个静态方法来构建一个异步事件，方法如下。

- supplyAsync(Supplier supplier)，带有返回值的异步执行方法，传入一个函数式接口，返回一个新的CompletableFuture对象。默认使用ForkJoinPool.commonPool()作为线程池执行异步任务。
- supplyAsync(Supplier supplier,Executor executor)，带有返回值的异步执行方法，多了一个Executor参数，表示使用自定义线程池来执行任务。
- runAsync(Runnable runnable)，不带返回值的异步执行方法，传入一个Runnable，返回一个新的CompletableFuture对象。默认使用ForkJoinPool.commonPool()作为线程池执行异步任务。
- runAsync(Runnable runnable,Executor executor)，不带返回值的异步执行方法，多了一个Executor参数，表示使用自定义线程池来执行任务。


主动获取执行结果

CompletableFuture类实现了Future接口，所以它开始可以像Future那样主动通过阻塞或者轮询的方式来获得执行结果。

- get()，基于阻塞的方式获取异步任务执行结果。
- get(long timeout,TimeUnit unit)，通过带有超时时间的阻塞方式获取异步执行结果。
- join()，和get()方法的作用相同，唯一不同的点在于get()方法允许被中断，也就是会抛出InterruptedException，但是join()不允许被中断。
- getNow(T valueIfAbsent)，这个方法有点特殊，如果当前任务已经执行完成，则返回执行结果，否则返回传递进去的参数valueIfAbsent。

在CompletableFuture类中还有一个比较有意思的方法complete(T value)，它表示完成完成计算，也就是把value设置为CompletableFuture的返回值并且唤醒在上述方法阻塞的线程。

CompletionStage方法及作用说明



CompletionStage表示任务执行的一个阶段，每个异步任务都会返回一个新的CompletionStage对象，我们可以针对多个CompletionStage对象进行串行、并行或者聚合的方式来进行后续下一阶段的操作，简单来说，就是实现异步任务执行后的自动回调功能。