HenCoder Plus 讲义

RxJava 3 的原理完全解析

基本用法

```
@GET("users/{username}/repos")
fun getRepos(@Path("username") username: String):
Single<List<Repo>>

...

api.getRepos("rengwuxian")
.subscribeOn(Schedulers.newThread())
.observeOn(AndroidSchedulers.mainThread())
.subscribe(object: SingleObserver<List<Repo>>() {
  override fun onSubscribe(disposable: Disposable) {
    textView.text = "正在请求"
    this@MainActivity.disposable = disposable
  }

  override fun onSuccess(repos: List<Repo>) {
    textView.text = repos[0].name
  }

  override fun onError(e: Throwable) {
    textView.text = e.message ?: e.javaClass.name
  }
});
```

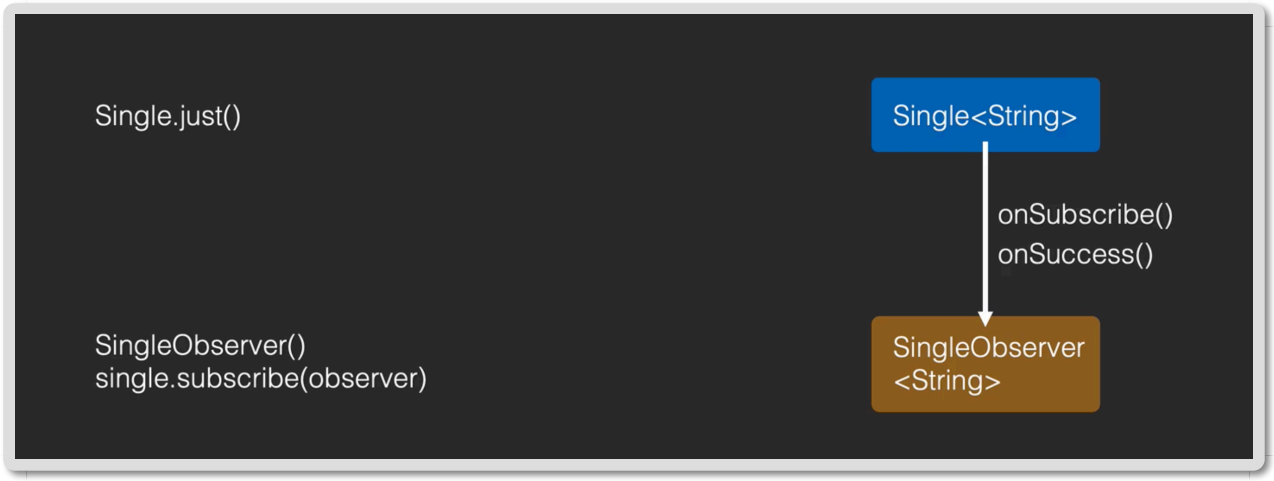
框架结构

RxJava 的整体结构是一条链，其中：

1. 链的最上游：生产者 Observable
2. 链的最下游：观察者 Observer
3. 链的中间：各个中介节点，既是下游的 Observable，又是上游的 Observer

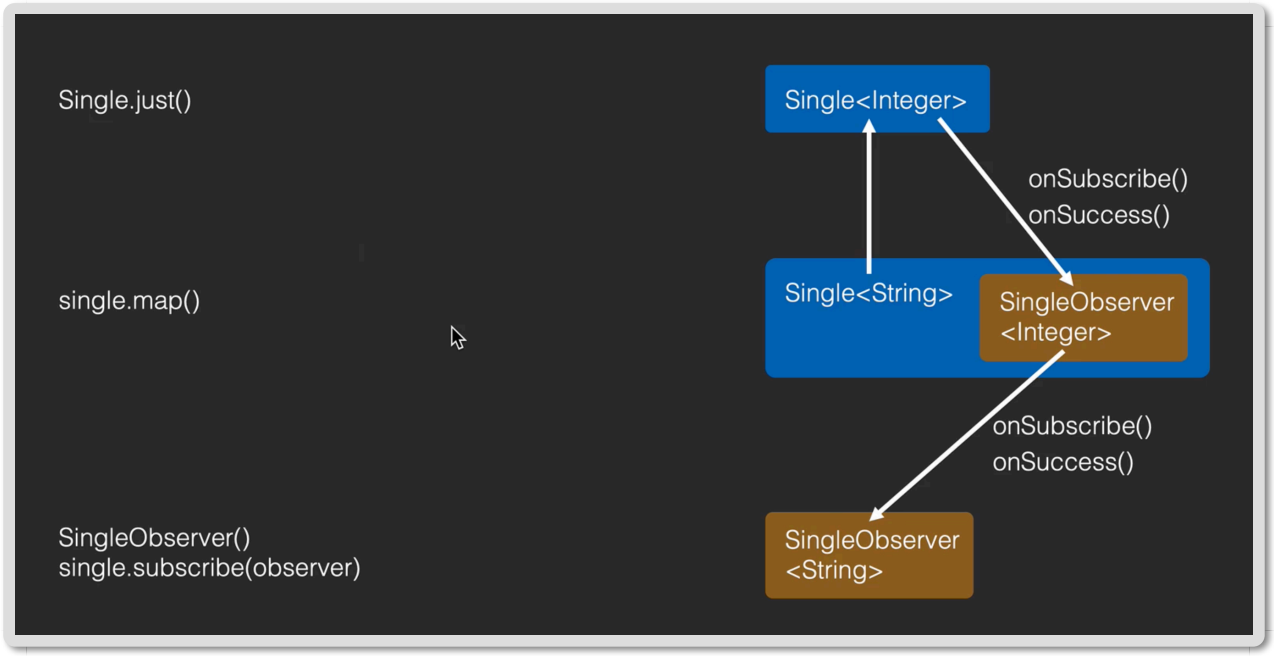
创建：

Single 为例：



操作符 Operator（map() 等等）：

1. 基于原 Observable 创建一个新的 Observable
2. Observable 内部创建一个 Observer
3. 通过定制 Observable 的 subscribeActual() 方法和 Observer 的 onXxx() 方法，来实现自己的中介角色（例如数据转换、线程切换）



Disposable：

可以通过 dispose() 方法来让上游或内部调度器（或两者都有）停止工作，达到「丢弃」的效果。

subscribeOn()

原理

在 Scheduler 指定的线程里启动 subscribe()

效果

- 切换起源 Observable 的线程；
- 当多次调用 subscribeOn() 的时候，只有最上面的会对起源 Observable 起作用。

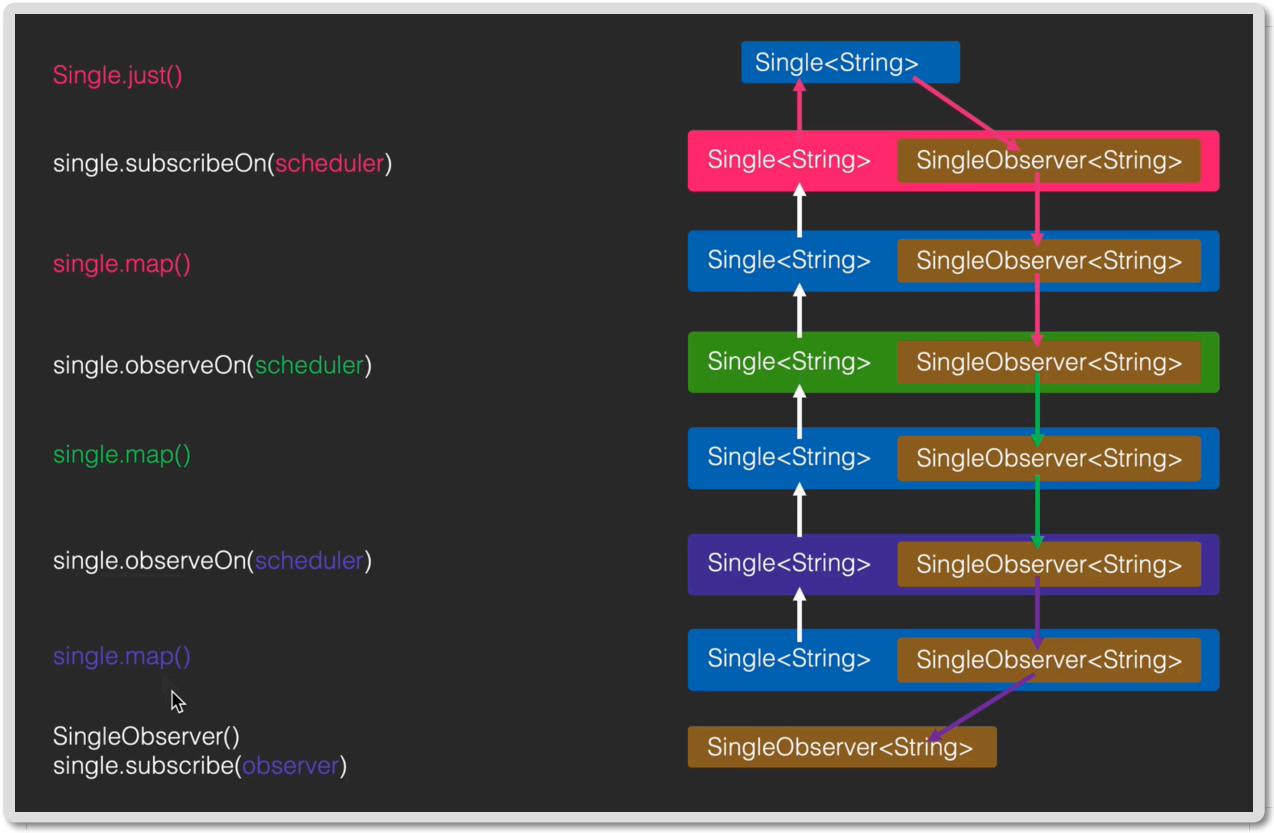
observeOn()

原理

在内部创建的 Observer 的 onNext() onError() onSuccess() 等回调方法里，通过 Scheduler 指定的线程来调用下级 Observer 的对应回调方法

效果

- 切换 observeOn() 下面的 Observer 的回调所在的线程
- 当多次调用 observeOn() 的时候，每个都会进行一次线程切换，影响范围是它下面的每个 Observer (除非又遇到新的 observeOn())



Scheduler 的原理

1. Schedulers.newThread() 和 Schedulers.io()：
    - 当 scheduleDirect() 被调用的时候，会创建一个 Worker，Worker 的内部会有一个 Executor，由 Executor 来完成实际的线程切换；
    - scheduleDirect() 还会创建出一个 Disposable 对象，交给外层的 Observer，让它能执行 dispose() 操作，取消订阅链；
    - newThread() 和 io() 的区别在于，io() 可能会对 Executor 进行重用。
2. AndroidSchedulers.mainThread()：

    通过内部的 Handler 把任务放到主线程去做。

问题和建议?

课上技术相关的问题，都可以去群里和大家讨论，对于比较通用的、有价值的问题，可以去我们的知识星球提问。

具体技术之外的问题和建议，都可以找丢物线（微信：diuwuxian），丢丢会为你解答技术以外的一切。

觉得好?

如果你觉得课程很棒，欢迎给我们好评呀！https://ke.qq.com/comment/index.html?cid=381952

> 一定要是你真的觉得好，再给我们好评。不要仅仅因为对扔物线的支持而好评（报名课程已经是你最大的支持了，再不够的话 B 站多来点三连我也很开心），另外我们也坚决不做好评返现等任何的交易。我们只希望，在课程对你有帮助的前提下，可以看到你温暖的评价。

更多内容：

- 网站：https://hencoder.com；https://kaixue.io
- 各大搜索引擎、微信公众号、微博、知乎、掘金、哔哩哔哩、YouTube、西瓜视频、抖音、快手、微视：统一账号「**扔物线**」，我会持续输出优质的技术内容，欢迎大家关注。
- 哔哩哔哩快捷传送门：https://space.bilibili.com/27559447

大家如果喜欢我们的课程，还请去扔物线的哔哩哔哩，帮我素质三连，感谢大家！