使用 RxJava 从多种来源中加载数据

原创 2016-06-29 伯乐专栏/小小鱼 安卓应用频道

> 英文：Dan Lew
> 来源：伯乐在线 - 我是鱼小小鱼
> 链接：http://android.jobbole.com/83513/

假设我需要从网络上查询一些数据。显然我可以在每次需要时都从网络上查询，但是将数据缓存在硬盘和内存中将更加高效。

具体点说，我想要这么做：

1.只有在需要从服务器更新数据的时候，才通过网络查询。

2.其他情况则从通过缓存之前查询的数据，实现快速的读取。

接下来，我将通过使用 RxJava 来实现上面的想法。

基本模式

为每一个数据源（网络、磁盘和内存）准备一个Observable<Data> ，然后我们就可以通过使用concat() 和 first() 这两种操作来构建一个简单的解决方案。

concat() 可以把多个Observables的结果链接起来。而first() 可以取出一个序列中的第一项。因此，你可以使用concat().first()来取出多个数据源所组成序列的第一项。

让我们通过代码来看一下：

```
// Our sources (left as an exercise for the reader)
Observable<Data> memory = ...;
Observable<Data> disk = ...;
Observable<Data> network = ...;

// Retrieve the first source with data
Observable<Data> source = Observable
```

```
    .concat(memory, disk, network)
    .first();
```

这个模式的关键点在于，concat()只会在需要时才去订阅每个Observable。如果数据已经被缓存了，first() 就会停止查询，即不再去较慢的数据源中查询。换种更具体的说法，如果内存中返回了数据结果，我们就不必再去磁盘或者网络中查询了。相反，如果内存和磁盘都不返回结果，就会触发一次新的网络请求。

请注意，在concat()中不同Observable的放置顺序是非常重要的，因为在查询时，是按照指定的顺序进行的。

数据存储

接下来，我们需要对查询到的数据进行储存。如果你不把网络查询的数据结果储存到磁盘，也不把磁盘查询的结果缓存到内存中，那之前的工作就白瞎啦！我们在上面写的代码会一直通过网络查询数据。

我的解决方案是：每当一个数据源返回结果，就把该结果储存或者缓存。

```
Observable<Data> networkWithSave = network.doOnNext(data -> {
  saveToDisk(data);
  cacheInMemory(data);
});

Observable<Data> diskWithCache = disk.doOnNext(data -> {
  cacheInMemory(data);
});
```

现在，当你使用networkWithSave 和 diskWithCache加载数据时，这些数据都会同时被储存起来。

采用这种策略还有一个好处，networkWithSave/diskWithCache 可以在任何地方使用，而不仅限在我们的多数据源模式中。

数据陈旧

不幸的是，我们目前的数据存储方案效率有点高的过头了。每次都会返回相同的数据，而这些数据都不知道是几年前的了。注意，在服务器更新数据后，我们是需要通过网络查询新数据的。

这个问题的解决办法存在于first()中，可以通过它来进行过滤操作，即拒绝返回没有价值的数据。

```
Observable<Data> source = Observable
  .concat(memory, diskWithCache, networkWithSave)
  .first(data -> data.isUpToDate());<br>
```

这样一来，我们就只会得到序列中最快且最新的数据。如果某个数据源中的数据不是最新的，那么就会在它后面的数据源中查询，直到找到最新的数据。

first() vs. takeFirst()

除了使用first()，我们还有一种选择，即takeFirst()。

两种方法间的区别是当所有数据源都无法返回有效数据时，first()会抛出NoSuchElementException() 异常，而takeFirst()会正常结束，不抛出异常。

至于具体使用哪一个方法，主要取决于你是否需要显式地处理数据的缺失。

代码样例

下面的链接是上述代码的完整实现：https://github.com/dlew/rxjava-multiple-sources-sample

如果你想要学习真实的案例，可以进入the Gfycat app，在这里使用了文章中介绍的模式以获取Gfycat的元数据。所涉及的代码并没有用到文章中介绍的所有能力（因为并不需要），但是它向我们展示了concat().first()的基本使用。

阅读原文