响应式编程入门——初识 RxJava

响应式编程

响应式编程近年来大红大紫。它不是设计模式，也不是框架，而是一种编程范式。所谓编程范式，指的是某种编程语言典型的编程风格。编程范式不受语言的局限，同一种语言也可以有多种范式。常见的编程范式有，过程化编程（C 语言）、面向对象编程（C++ / Java）、函数式编程（Haskell / Scala）以及响应式编程。

响应式编程使用异步数据流进行编程。这不是什么新的概念，在和 GUI 相关的程序中早已司空见惯，各种手势事件就是数据流，你可以监听并处理这些事件。为了简单起见，不妨把它理解为观察者模式的扩展。观察者订阅一个可观察对象，之后观察者对可观察对象发出的事件或者事件序列作为响应，如图 1 响应式编程简要流程图 1 所示。这种做法的好处在于在等待可观察对象发送事件时，不会产生阻塞。

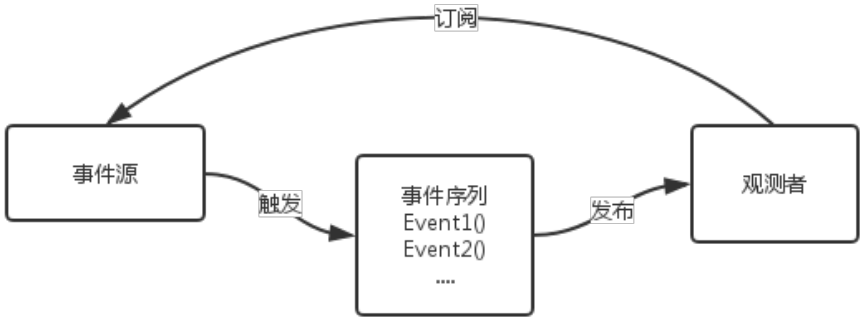

图 1 响应式编程简要流程

ReactiveX

ReactiveX 全称为 Reactive Extension，其官方定义是使用可观测的序列来组成异步的、基于事件的程序的库。换句话说，它是多个开源项目的集合，实现并扩展了多种语言的响应式编程思想。

观察者和可观察对象只是 ReactiveX 的基础，真正强大之处在于它的“X” —— 操作符。操作符可以对可观察对象发出的事件流进行变换、组合、过滤等一系列的控制，避免了在异步操作时回调函数的层层嵌套（又称“迷之缩进”、“回调地狱”），取而代之的是流式编程；提高了代码的抽象层级，所以你可以只关注定义了业务逻辑的那些相互依赖的事件，而非纠缠于大量的实现细节，代码结构也将变得简洁明了。

RxJava 是 ReactiveX 针对 Java 语言的实现，下面我们以 Android 平台的使用为例作简要介绍。

RxJava

RxJava 有四个基本概念：Observable (可观察对象)、Observer (观察者)、subscribe (订阅)、事件。Observable 和 Observer 通过 subscribe()方法实现订阅关系，从而 Observable 可以在需要的时候发出事件来通知 Observer。

图 2 展示了一个典型的处理流程。假设可观察对象由 5 个事件源组成，首先经过 map 操作符处理讲每一项转换成将其另外一种类型，然后经过 filter 操作符过滤出符合条件的子项，最后经过 subscribeOn 操作符把该事件序列指定到某个调度器上进行处理（如指定到 io 线程）。

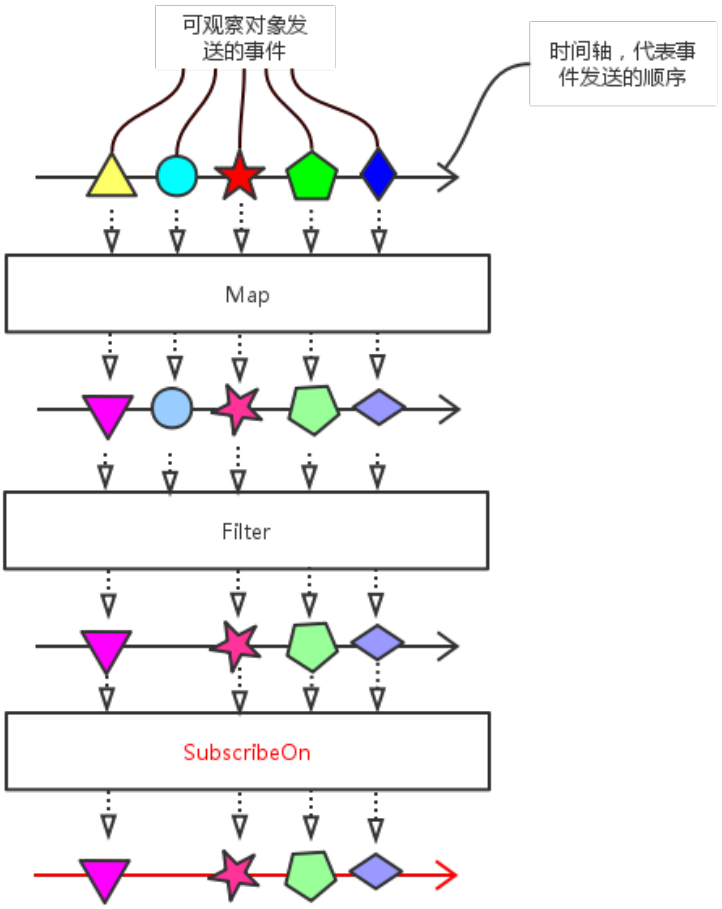

图 2 RxJava 操作符示例

下面用示例代码说明上述流程。

```
/**
* 为了节约篇幅，并让代码看起来简(zhuang)洁(bi),
* 这里使用了 lambda 表达式 (不需要完全理解，会其意即可)
* 目的：将 png 图片异步加载到内存，并通知页面更新
```

```
* 步骤:
* 1. 从目录中获取文件作为事件源
* 2. 过滤出 png 结尾的文件
* 3. 根据 file 获取 Bitmap
* 4. 将上述操作指定到 io 线程中执行
* 5. 将观察者操作指定到主线程中执行
* 6. 加载图片
*/
File[] file = getFilesFromDirectory();
Observable.from(file)
    .filter((Func1) (file) -> { file.getName().endsWith(".png") })
    .map((Func1) (file) -> { getBitmapFromFile(file) })
    .subscribeOn(Schedulers.io())
    .observeOn(AndroidSchedulers.mainThread())
    .subscribe((Action1) (bitmap) -> { load(bitmap) });
```

RxJava 相比其他的开源库而言，不太容易上手，学习曲线较为陡峭，但凭借其简洁性在 Android 领域依然广受追捧。最新的统计显示，Google play Top500 的应用程序使用 RxJava 的比例达到了 10%，与 Glide, ButterKnife 及 EventBus 等知名库的市占率相当。由此可见 RxJava 已成为 Android 程序员的必备技能，期待能早日运用到我们的项目当中。