基于 Android 端 MVP 模式和响应式网络框架的设计与实现*

李 想[1,2],特日根[1,2,3]
(1.长光卫星技术有限公司,吉林 长春 130000;2.吉林省卫星遥感应用技术重点实验室,吉林 长春 130000;
3.中国科学院长春光学精密机械与物理研究所,吉林 长春 130000)

摘 要: MVC(Model-View-Controller)模式是 Android 应用开发的传统方式(用 Activity/Fragment 表示 Controller 层,用 XML 文件表示 View 层),随着项目的页面增多,逻辑复杂度提升,将使 Activity 文件变得臃肿,代码耦合度明显提高,不利于项目后期的升级和维护。通过对传统 MVC 开发模式与主流 MVP(Model-View-Presenter)开发模式进行研究和比较,发现 MVP 开发模式能够更好地解决上述问题。同时,官方 HttpURLConnection 类对于 HTTP 网络请求的效率无法满足业务需求,而 Retrofit2+OkHttp3+RxJava2 的响应式网络请求框架具有更高的响应效率。以《长光卫星云极视》项目为背景,研究并验证 MVP 模式和 Retrofit2+OkHttp3+RxJava2 的响应式网络请求框架结合的可行性。

关键词: MVP 模式;Retrofit;OkHttp;RxJava;响应式;Android 应用开发
中图分类号: TN915 **文献标识码**: A DOI:10.16157/j.issn.0258-7998.200224

中文引用格式: 李想,特日根. 基于 Android 端 MVP 模式和响应式网络框架的设计与实现[J].电子技术应用,2021,47(2):49-53,57.
英文引用格式: Li Xiang,Te Rigen. Design and implementation of Android-based MVP mode and responsive network request framework[J]. Application of Electronic Technique,2021,47(2):49-53,57.

Design and implementation of Android-based MVP mode and responsive network request framework

Li Xiang[1,2],Te Rigen[1,2,3]
(1.Chang Guang Satellite Technology Co.,Ltd.,Changchun 130000,China;
2.Main Laboratory of Satellite Remote Sensing Technology of Jilin Province,Changchun 130000,China;
3.Changchun Institute of Optics,Fine Mechanics and Physics,Chinese Academy of Sciences,Changchun 130000,China)

Abstract: MVC(Model-View-Controller) mode is the traditional way of Android application development (Activity/Fragment is used to represent the Controller layer and XML files are used to represent the View layer). As the number of project pages increases, the logic complexity increases, and the Activity file becomes bloated and the code coupling is obvious. Improvement is not conducive to the upgrade and maintenance of the project later. By studying and comparing the traditional MVC development model and the mainstream MVP(Model-View-Presenter) development model, it is found that the MVP development model can better solve the above problems. At the same time, the efficiency of the official HttpURLConnection class for HTTP network requests can not meet business needs, and the responsive network request framework of Retrofit2+OkHttp3+RxJava2 has higher response efficiency. Based on the "Changguang Satellite Cloud Extreme Vision" project as the background, this paper studies and verifies the feasibility of combining the MVP mode with the responsive network request framework of Retrofit2+OkHttp3+RxJava2.
Key words: MVP mode;Retrofit;OkHttp;RxJava;responsive;Android application development

0 引言

在当今社会,移动端因其便携性、低功耗以及无线网的快速接入等优势,使得人们与外部世界进行网络连接更加方便而舒适。正因如此,移动端编程成为了当下最热门的计算机编程领域之一。2019 年第二季度移动端操作系统市场份额表明,Android 系统占比 77.14%,iOS 系统占比 22.83%,其余系统不及 1%,由此可知 Android 在当今手机行业起着举足轻重的作用。随着每一款应用承载的功能不断增多,其代码管理也变得更为复杂。对于 Android 应用开发来说,用 Android Studio 编译器生成 Android 项目时,其生成的 XML 文件和 Activity 文件已经对应传统 MVC(Model-View-Controller)架构模式的 View 层和 Controller 层,同时 XML 文件不能实现全部布局功

*基金项目:国家重点研发计划(2018YFB1004605);吉林省科技计划-科技创新中心项目(20180623058TC)

能，因此部分 View 层内容需交付给 Activity 文件完成。Activity 文件随着页面和业务逻辑的不断增加也会不断增大，代码间耦合度明显提高[1-3]，将对项目的升级和维护带来障碍。因此，对于大中型项目来说，MVC 架构并不可取。

对于一个常规项目，其网络请求必不可少，虽然官方提供了诸如 HttpURLConnection 类等 HTTP 请求方式，但该类在大量网络请求时，其性能较差。

针对此问题，MVP（Model－View－Presenter）＋Retrofit＋OkHttp＋RxJava 的架构应运而生，该架构能有效降低代码耦合度，使 Activity 文件的内容更加单一，网络请求和布局更新更加高效。对于整个项目而言，整体结构更加清晰，代码可维护性也得到大幅度提升。

本文通过对 MVP＋Retrofit＋OkHttp＋RxJava 的研究分析，以《长光卫星云极视》项目的登录模块为应用案例，设计并验证 Android 应用开发中 MVP 模式和 Retrofit2＋OkHttp3＋RxJava2 的响应式网络请求框架结合的方法及可行性。

1 MVP 模式

1.1 设计思想

MVP 最早出现在 IBM 公司的项目研发中，由 Model 层（负责数据修改和操作的部分）、View 层（包含所有的 UI 组件，并负责与 Presenter 层进行所有交互操作）、Presenter 层（负责所有项目的逻辑）组成[4-5]。

MVP 模式在多名开发人员协同开发及测试时体现出了更大的解耦性。使用 MVP 模式的优势包括：

（1）能将 Activity/Fragment 中的任务独立出来；

（2）能将单个复杂的多任务分割成多个简单的任务；

（3）可以分离页面与数据；

（4）促进自动化单元测试[6]。

1.2 实现过程

应用程序中 MVP 模式的顺序图如图 1 所示。在一个常规的 Android 应用中会有两个参与者：用户（使用该程序的人）和数据（存储的信息实体），以登录操作为例：

（1）用户点击按钮获取验证码；

（2）View 对象接收到用户动作，并向 Presenter 层发送委派动作，即执行 actionSendMessage()函数；

（3）如果 Presenter 层需要接口或数据库中的数据，则会向 Model 层通过 getMessage()函数发送一条检索数据的消息，在这个过程中，Presenter 层与 Model 层属于观察者与被观察者的关系；

（4）当 Model 层获取到数据时，Presenter 层将观察到 Model 层发送的事件，同时执行 returnSendMessage()函数，向 View 层发送一条消息，并将获得的验证码返回给用户，至此完成获取验证码的一次操作。

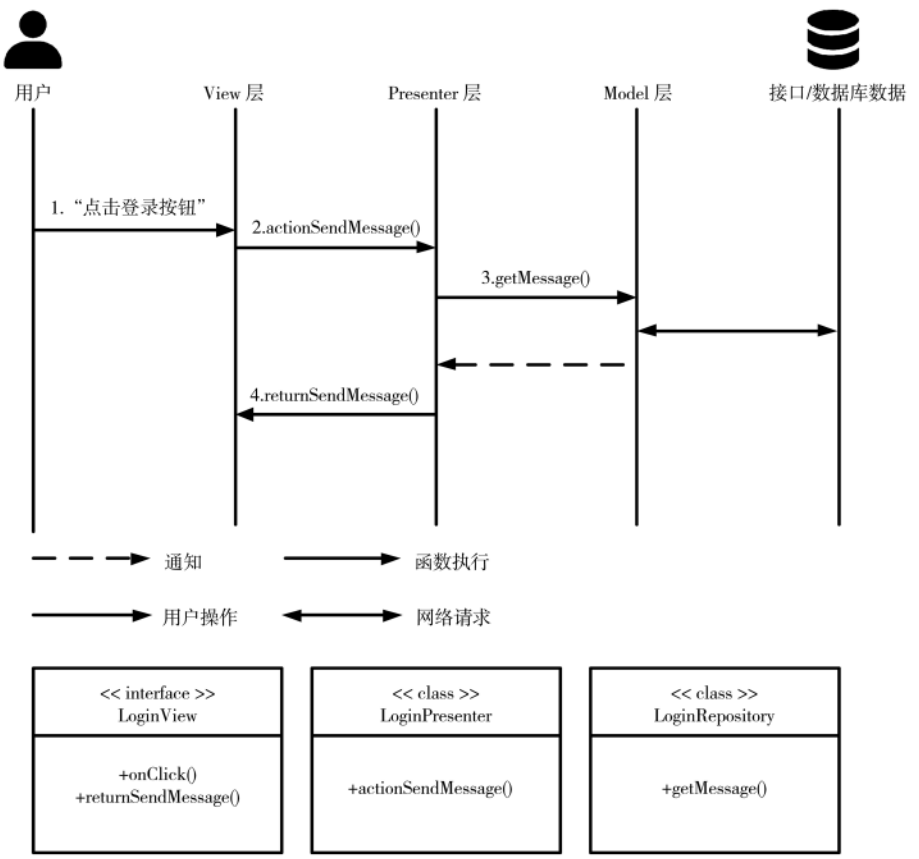

图 1 MVP 模式的顺序图

2 响应式网络请求框架

2.1 设计思想

Retrofit2+OkHttp3+RxJava2 是当下最流行的 Android 网络请求框架之一。在 Android6.0 之前，官方推荐用 HttpClient 接口来进行网络请求，后续则更改为 Java.net 下的 HttpUrlConnection 接口，但该方式仅支持 HTTP/1.0 和 HTTP/1.1，不支持 HTTP/2.0，也不支持多路复用。当遇到大量网络请求时性能较差。与 HttpUrlConnection 相比，OkHttp3 底层基于 Okio 开源库，使用了比阻塞式 IO 效率更高的 Java NIO(Non-Blocking I/O)。Retrofit2 作为基于 OkHttp 封装的 RESTful 网络请求框架，是当前耦合度最低、功能最强大的网络请求框架。Retrofit2 与 RxJava2 结合使用时，前者把请求封装进 Observable 对象，在新线程中执行 HTTP 请求，请求结束后切换到 IO 线程中执行用户的后续动作。总体来说，Retrofit2 用接口的方式进行 HTTP 网络请求，并负责请求数据以及接收返回结果，OkHttp3 负责 HTTP 请求的过程，RxJava2 负责异步以及多线程的切换。

2.2 OkHttp3

OkHttp 是 Square 公司的一款开源的网络请求库，使用 OkHttp3 进行网络请求能提高 HTTP 请求的加载速度，同时更节省带宽。OkHttp3 的高效性体现在以下三方面：

(1)允许连接同一主机的所有请求分享同一个 socket；

(2)通过响应式缓存来避免重复请求；

(3)当服务端存在多个 IP 地址时，若第一个地址连接失败，OkHttp3 会通过尝试备用 IP 地址进行静默恢复[7]。

2.3 Retrofit2

与 OkHttp 相同，Retrofit 也出自 Square 公司，Retrofit2 可以理解为一个 HTTP 网络请求的适配器，它将一个 HTTP 请求通过 Java/Kotlin 接口动态代理的方式来表达，并通过 OkHttp3 发送 HTTP 请求。Retrofit 和 OkHttp 的关系可以总结为：OkHttp 纯粹是一个 HTTP/SPDY 客户端；Retrofit 是基于 HTTP 的高级 REST 抽象。

Retrofit 框架的优势在于：

(1)使用清晰的注解方式，在最大程度上简化 URL 的拼写形式；

(2)自由度大，支持自定义 Converters 及其他业务逻辑；

(3)同时支持同步执行和异步执行；

(4)支持多种文件解析，如 GSON、JSON 和 XML 等；

(5)支持 RxJava。

2.4 RxJava2

RxJava2 是一个在 JAVA 虚拟机上使用可观测的序列组成的基于事件的异步程序库。它可以理解为是一种观察者模式，用观察者和被观察者来实现异步操作。与 Android 官方的异步操作方法相比，RxJava 的优势是简洁，当项目随着迭代变得繁琐时，RxJava 仍保持了其简洁性[8]。

2.5 实现过程

基于 Retrofit2+OkHttp3+RxJava2 的响应式网络请求框架整体流程如图 2 所示。

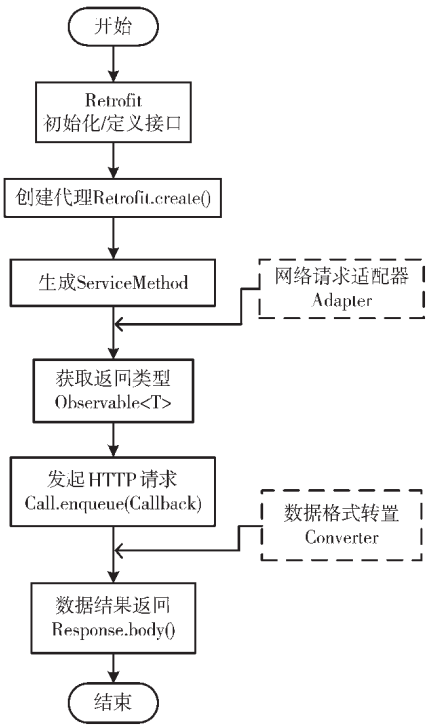

图 2 网络请求流程图

(1)导入相关的依赖，将域名传入一个 Retrofit 构造器中；

(2)通过 Retrofit.create()方法传入 Java 接口并返回一个 Call 对象（该对象默认使用 OkHttp3 作为 HTTP 请求的 Client）；

(3)RxJava2 在订阅时调用 Call.enqueue()方法来进行 HTTP 的异步请求；

(4)在网络请求获得响应后，Call 对象对返回的信息根据设置的转置模式进行转换；

(5)返回结果数据。

3 MVP 模式和网络框架在 Android 应用开发中的应用

《长光卫星云极视》项目采取了 MVP 模式以及响应式网络框架，以获取登录验证码为例，实现的目录结构如图 3 所示。

3.1 Model 层

Model 层主要的功能是从服务端获取数据，由 LoginRepository、LoginService 和 LoginServiceimpl 组成。

(1)LoginRepository

获取用户验证码是通过 LoginRepository 类中的 send-Message()方法来实现的，通过调用 Retrofit2 的 send-Message()的方法，实现 HTTP 请求，在请求成功后，数据格式

图 3 项目目录结构图

转置为 SendMessageBean 实体类对象返回，关键代码如下：

```
class LoginRepository @Inject constructor(){
    fun sendMessage(param: String):
      Observable<SendMessageBean> {
        return RetrofitFactory (BASE_URL).create(ApiService::
class.java)
      .sendMessage(param)
    }
}
```

在调用接口过程中，RetrofitFactory 类返回 Retrofit 实例并配置请求参数（Header/Url），通过 OkHttp3 对服务器端进行网络请求，关键代码如下：

```
retrofit=Retrofit.Builder()
      .baseUrl(url)
      .addConverterFactory(
            GsonConverterFactory.create())
      .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
      .client(initClient())
      .build()
```

Retrofit 实例调用接口 API 类中的对应方法 sendMessage()，并获得 HTTP 请求的 URL 进行网络请求。关键代码如下：

```
@POST("/global/mall/sendMessage.do")
@FormUrlEncoded
fun sendMessage(@Field("telephone")
    targetSentence: String) :
    Observable<SendMessageBean>
```

(2)LoginService

LoginService 是一个接口类，定义了需要实现的方法，关键代码如下：

```
interface LoginService {
    fun sendMessage(string: String):
        Observable<SendMessageBean>
}
```

(3)LoginServiceImpl

LoginServiceImpl 是 LoginService 接口的实现类，通过调用 LoginRepository 类的 sendMessage()方法传递 HTTP 请求参数，并获得请求结果的被观察者对象，关键代码如下：

```
class LoginServiceImpl @Inject constructor() : LoginService {
    @Inject
    lateinit var mLoginRepository:
        LoginRepository
    override fun sendMessage(string: String):
        Observable<SendMessageBean> {
            return
mLoginRepository.sendMessage(string)
.flatMap(Function<SendMessageBean,
ObservableSource<SendMessageBean>> {
t -> return@Function Observable.just(t)
        })
    }
}
```

3.2 View 层

View 层主要对应登录界面 LoginFragment，同时 View 层定义了 LoginView 接口，接口中包括获取短信成功及失败的回调方法，LoginPresenter 通过接口 returnSendMessage()和 returnSendMessageError()方法与 LoginFragment 进行交互，关键代码如下：

```
interface LoginView: BaseView {
    fun returnSendMessage(
    sendMessageBean: SendMessageBean)
    fun returnSendMessageError()
}
```

在 LoginFragment 中，实现 LoginView 接口，获取接口返回值进行更新 UI 等操作，关键代码如下：

```
@SuppressLint("ResourceType")
override fun returnSendMessage(
sendMessageBean: SendMessageBean) {
    if (sendMessageBean.message == "success") {
        ...
    } else {
```

```
            Toast.makeText(activity,
                ERROR,
                Toast.LENGTH_SHORT)
                .show()
        }
}

override fun returnSendMessageError() {
        Toast.makeText(activity,
            ERROR,
            Toast.LENGTH_SHORT)
        .show()
}
```

3.3 Presenter 层

Presenter 层是 View 层调用 Model 层的桥梁，LoginPresenter 持有 LoginView 的引用，在方法中通过用户在 LoginFragment 中触发按键时，调用 LoginPresenter 的对应方法，通过 LoginService 对象进行对 Model 层的访问，同时获取 HTTP 返回值的观察者对象，并将值传递给 LoginView 对象，在 LoginFragment 中进行 UI 更新。关键代码如下：

(1)LoginFragment

```
if (findTelephoneBean.meta == "success") {
        mPresenter.sendMessage(
        mTextInputEditText.text.toString())
} else {
            paramError(getString(
                R.string.login_error_3))
}
```

(2)LoginPresenter

```
class LoginPresenter  @Inject constructor(): BasePresenter<LoginView>(){
        @Inject
        lateinit var mLoginService
            : LoginService
        fun sendMessage(param: String){
            mLoginService.sendMessage
        (param).execute(object :
        BaseObserver<SendMessageBean>() {
            override fun onNext(
        t: SendMessageBean) {
                super.onNext(t)
                mView.returnSendMessage(t)
              }
        override fun onError(e: Throwable) {
            super.onError(e)
            mView.returnSendMessageError()
              }
      }, lifecycleProvider)
    }
}
```

3.4 注意事项

为了快速开发和提高代码质量，在使用 MVP+Retrofit2+OkHttp3+RxJava2 模式时，应采取 JUnit 单元测试的方式，对单个 Presenter 文件和 Service 文件编译运行。该方式无需在真机或模拟器上全局调试，同时也无需考虑其他类文件的影响。通过 JUnit 单元测试，可以快速定位到问题。

4 结论

本文中通过对 MVC 模式以及 HTTP 网络请求进行系统的分析，发现在一个中大型项目中，采用 MVC 模式，会使得充当 Controller 层的 Activity/Fragment 同时充当 View 层的角色，使得文件代码量臃肿，增加了业务逻辑的耦合度，在项目开发上乃至后期维护上都造成了很大的问题。与此同时，当项目进行复杂多次的网络请求场景时，官方提供的 HttpURLConnection 类表现出了性能低下、逻辑复杂的缺陷。

根据 MVP 模式的设计思想，将业务事件交付给 Presenter 层进行处理，使得 Model 层和 View 层做到了完全解耦，在整体项目中开发模块职责划分更明显，使逻辑代码的耦合度降低，便于后期的维护和二次开发。与此同时基于 Retrofit+OkHttp+RxJava 的响应式网络框架，在 HTTP 请求上更为高效，在业务处理上相比 HttpURLConnection 更简化。RxJava 的使用，会随着网络请求逻辑变得越来越复杂，依然保持简洁。

本文通过上述思想设计并实现了《长光卫星云极视》用户登录模块，验证了 MVP 模式与 Retrofit+OkHttp+RxJava 的网络框架在 Android 应用开发中结合的可行性，同时给出了设计思路及关键程序，并最终达到预期效果。

参考文献

[1] 倪红军.基于 MVP 模式的 Android 应用开发研究[J].电子设计工程，2018，26(11)：6-9，13.
[2] 曾露.MVP 模式在 Android 中的应用研究[J].软件，2016(6)：75-78.
[3] SYROMIATNIKOV A，WEYNS D.A journey through the land of model-view-* design patterns[C].2014 IEEE/IFIP Conference on Software Architecture(WICSA).IEEE，2014.
[4] WIKIPEDIA.Model-view-presenter[EB/OL].(2011-02-03)[2020-03-24].http://en.wikipedia.org/wiki/Modelviewpresenter.
[5] 王念桥.应用 MVP 模式改进软件架构[J].计算机时代，2012(4)：37-38.
[6] HUSTER.深入理解 MVC 与 MVP[EB/OL].(2011-03-14).[2020-03-24].https://www.cnblogs.com/seaky/archive/2011/04/06/1982533.html.

(下转第 57 页)

中文文本(包含数字),542 条英文文本。

深色框为改进前方法的定位结果,浅色框为改进后的方法定位结果,右图为对应的像素分割后的检测结果。从特征图中可以看出,本文方法对长条形的英文检测敏感,能够有效检测出长条形的英文,对曲向的英文有着不错的识别能力。在对图像进行检时,平均每张检测速度为 0.89 s,即 FPS=1.12,R=72.9,P=70.0,F=71.4。

3 结论

本文提出改进的文本检测方法在数据集表现上均超过原有方法,接近当前领先的算法精度,能够提高已有文本识别系统对自然场景下曲向文本与模糊文本的识别精度。后续结合自然语言处理和语义分割任务,又可以将所识别的文本内容、文本背景内容组合生成关于一张图片中文本的具体描述内容,使得使用者获取更多的文本信息。

参考文献

[1] Deng Dan, Liu Haifeng, Li Xuelong, et al. Pixellink: detecting scene text via instance segmentation[C]. Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[2] LONG J, SHELHAMER E, DARRELL T. Fully convolutional networks for semantic segmentation[C]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015: 3434-3440.

[3] SIMONYAN K, ZISSERMAN A. Very deep convolutional networks for large-scale image recognition[C]. International Conference on Learning Representations, 2015.

[4] Hu Jie, Shen Li, Sun Gang. Squeeze-and-excitation networks[C]. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2018: 7132-7141.

[5] Wang Xiaobing, Jiang Yingying, Luo Zhenbo, et al. Arbitrary shape scene text detection with adaptive text region representation[C]. 2019 IEEE/CVF Conference on CVPR, 2019.

[6] HOCHREITER S, SCHMIDHUBER J. Long short-term memory[J]. Neural Computation, 1997, 9(8): 1735-1780.

[7] GLOROT X, BENGIO Y. Understanding the difficulty of training deep feedforward neural networks[C]. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010: 249-256.

[8] Deng Jia, Dong Wei, SOCHER R, et al. ImageNet: a large-scale hierarchical image database[C]. IEEE Conference on Computer Vision and Pattern Recognition, 2009: 248-255.

[9] Liu Yuliang, Jin Lianwen, Zhang Shuaitao, et al. Curved scene text detection via transverse and longitudinal sequence connection[J]. Pattern Recognition, 2018, 90(6): 337-345.

[10] KARATZAS D, GOMEZ-BIGORDA L, NICOLAOU A, et al. ICDAR 2015 competition on robust reading[C]. Document Analysis and Recognition (ICDAR), 13th International Conference, IEEE, 2015: 1156-1160.

[11] Yao Cong, Bai Xiang, Liu Wenyu, et al. Detecting texts of arbitrary orientations in natural images[C]. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2012: 1083-1090.

[12] Shi Baoguang, Bai Xiang, BELONGIE S. Detecting oriented text in natural images by linking segments[C]. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017: 3482-3490.

[13] LONG S, RUAN J, ZHANG W, et al. TextSnake: a flexible representation for detecting text of arbitrary shapes[C]. European Conference on Computer Vision (ECCV), 2018: 19-35.

[14] Yang Qiangpeng, Cheng Mengli, Zhou Wenmeng, et al. IncepText: a new inception-text module with deformable PSROI pooling for multi-oriented scene text detection[C]. Twenty-Seventh International Joint Conference on Artificial Intelligence (LJCAI-18), 2018.

(收稿日期:2020-04-17)

作者简介:

陈小顺(1996-),男,硕士研究生,主要研究方向:人工智能。

王良君(1982-),通信作者,男,博士研究生,讲师,主要研究方向:图像编码、压缩感知,E-mail:ljwang0819@ujs.edu.cn。

(上接第 53 页)

[7] IMOKHAI I. OkHttp interceptors with Retrofit[EB/OL]. (2019-10-23)[2020-03-24]. https://medium.com/swlh/okhttp-interceptors-with-retrofit-2dcc322cc3f3.

[8] 扔物线.给 Android 开发者的 RxJava 详解[EB/OL].(2017-09-16).[2020-03-24].http://gank.io/post/560e15be2dca-930e00da1083.

(收稿日期:2020-03-24)

作者简介:

李想(1992-),男,硕士,助理研究员,主要研究方向:移动端研发、遥感大数据。

特日根(1987-),通信作者,男,博士,副高级研究员,主要研究方向:遥感大数据,E-mail:terigen@charmingglobe.com。

版权声明