中山大学移动信息工程学院本科生实验报告

（2017 年秋季学期）

课程名称：移动应用开发　　任课教师：

年级	15	专业（方向）	移动互联网
学号	15352155	姓名	赖贤城
电话	13727024851	Email	754578682@qq.com
开始日期	2017/12/21	完成日期	2017/12/21

一、 实验题目

Retrofit+RxJava+OkHttp 实现网络请求

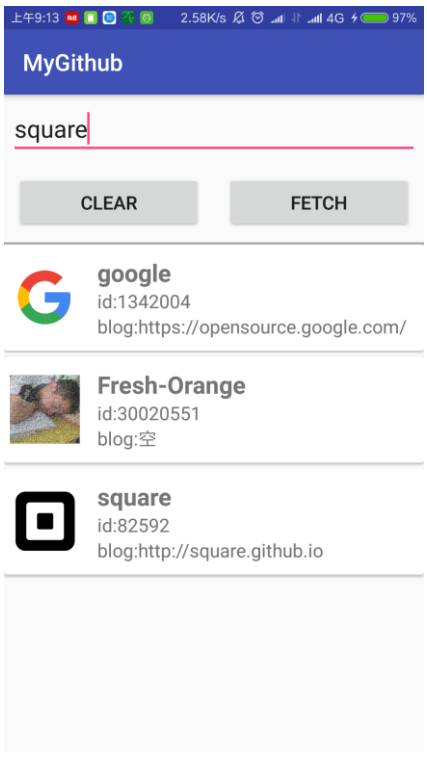
二、 实现内容

使用 Retrofit 实现网络请求，在本次实验中是请求 github 的数据

学习 RxJava 中 Observable 的使用，在本次实验中是配合 Retrofit使用

三、 课堂实验结果

（1） 实验截图



（2） 实验步骤以及关键代码

1. 实现获取用户和仓库接口

- ***用户的获取接口***

通过查询 github 的 api 可以看到获取单个用户的方法是 bse_url+“users/user”这样的形式，它返回的 json 里面包含了 login，id，blog 等等信息。

据此，我们可以先定义出一个 user 的 model 以供待会 json 解析的时候可以填充数据

```
public class User {
    private String login;
    private String blog;
    private String id;
    private String avatar_url;

    public String getLogin() { return login; }

    public String getBlog() { return blog; }

    public String getId() { return id; }

    public String getAvatar_url() { return avatar_url; }
}
```

然后创建对应的获取 user 的接口，以供待会 retrofit 据此生成代理对象，这里 @GET 中填入的是根据 api 的规定，api 就是 bse_url+“users/user”，这里的 user 是可变的，其值是根据 getUser 函数的参数来动态填入的

```
public interface UserService {
    @GET("/users/{user}")
    Observable<User> getUser(@Path("user") String user);
}
```

添加 gson 转换器以及 RxJava 适配器，并根据上一步的接口生成一个 UserService 代理对象

```
static public UserService getUserService(){
    Retrofit retrofit =
            new Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
            .build();
    return retrofit.create(UserService.class);
}
```

- ***仓库的获取接口***

同样的，查看 api 可以看到，获取用户的仓库的方法是 bse_url+“users/user/repos”这样的形式，它返回的 json 是一个数组的形式，每个数组元素里面包含了 name，language，description 等等信息。

据此，我们可以先定义出一个 repos 的 model 以供待会 json 解析的时候填充数据

```
public class Repos {
    private String name;
    private String description;
    private String language;

    public String getName() { return name; }

    public String getDescription() { return description; }

    public String getLanguage() { return language; }
}
```

然后创建对应的获取 repos 的接口，以供待会 retrofit 据此生成代理对象，注意因为返回的是 json 数组，所以这里的泛型填入的是 List 类型

```
public interface ReposService {
    @GET("/users/{user}/repos")
    Observable<List<Repos>> getReposList(@Path("user") String user);
}
```

添加 gson 转换器以及 RxJava 适配器，并根据上一步的接口生成一个 ReposService 代理对象

```
static public ReposService getReposService(){
    Retrofit retrofit =
            new Retrofit.Builder()
                    .baseUrl(BASE_URL)
                    .addConverterFactory(GsonConverterFactory.create())
                    .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
                    .build();
    return retrofit.create(ReposService.class);
}
```

2. 实现用户获取界面及逻辑

基本的布局就是 button+editview+RecyclerView 而已，这里不展开布局的细节，与以前不同的是增加了 progressBar 用以表示等待数据状态

1) RecyclerView

与之前的一样，adapter 留出事件接口，在 activity 中实现，点击事件跳转到展示仓库（repos）的 activity（并且带过去用户名），长按则是删除该项

```
recyclerView = findViewById(R.id.rc_items);
recyItemClickListener itemClickListener = new recyItemClickListener() {
    @Override
    public void onClick(int position) {
        Intent i = new Intent( packageContext: MainActivity.this, ReposActivity.class); 点击事件
        i.putExtra( name: "USER", userList.get(position).getLogin());
        startActivity(i);
    }
    @Override
    public void onLongClick(int position) {
        userList.remove(position);                                                    长按事件
        userAdapter.notifyDataSetChanged();
    }
};
userAdapter = new UserAdapter( context: this, userList, itemClickListener);
recyclerView.setLayoutManager(new LinearLayoutManager( context: this));
recyclerView.setAdapter(userAdapter);
```

虽然 RecyclerView 的适配器不展开讲，但是 item 布局还是要讲一下，这次使用了 cardView 作为每一个数据项的根布局。cardView 可以通过设置 Z 轴高度以及圆角的大小实现卡片的效果

```
<android.support.v7.widget.CardView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:foreground = "?attr/selectableItemBackground"
    android:clickable="true"
    android:focusable="true"
    android:layout_marginBottom="5dp"
    app:elevation="5dp">
```

2) 点击获取用户

- 首先是将环形进度条显示出来表示当前状态是获取数据的状态

- 然后通过前面定义好的 Service 调用对应的获取用户的函数，该函数返回的是一个 rxJava 的 Observable 对象，通过这个对象我们可以定义它的请求操作的线程为“新线程”，最后回调是在主线程，这样就可以把耗时操作与主线程分离开并且最后还是能在主线程更新界面
- 最后定义三个回调函数，由于 onCompleted() 和 onError() 二者是互斥的，所以两个地方都要定义进度条的隐藏操作，重要的是 onNext()，它在成功获取到数据的时候回调并且将数据作为参数传入，很明显，得到新的用户数据的时候要将其填入 RecyclerView 中进行显示

```
btFetch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        pbWait.setVisibility(View.VISIBLE);
        String user_name = etInput.getText().toString();
        ServicePool
                .getUserService()
                .getUser(user_name)
                .subscribeOn(Schedulers.newThread())
                .observeOn(AndroidSchedulers.mainThread())
                .subscribe(new Subscriber<User>() {
                    @Override
                    public void onCompleted() {
                        pbWait.setVisibility(View.GONE);
                    }

                    @Override
                    public void onError(Throwable e) {
                        pbWait.setVisibility(View.GONE);
                        Toast.makeText( context: MainActivity.this,  text: "请确认搜索的用户存在",Toast.LENGTH_SHORT).show();
                    }

                    @Override
                    public void onNext(User user) {
                        userList.add(user);
                        userAdapter.notifyDataSetChanged();
                    }
                });
    }
});
```

网络请求等操作放在新线程

最后的回调运行在主线程

3. 实现用户的仓库页面

这里的界面就很简单，直接一个 RecyclerView 就可以了，onCreate 的时候获取到主页面传过来的用户名

```
setContentView(R.layout.activity_repos);
String userName = getIntent().getStringExtra( name: "USER");
```

- ●用这个用户名去调用前面已经实现好的获取仓库的接口
- ●定义它的请求操作的线程为“新线程”，最后回调是在主线程，成功获取到数据（onNext()）之后将数据填入 RecyclerView 中显示出来
- ●由于 onCompleted() 和 onError() 二者是互斥的，所以两个地方都要定义进度条的隐藏操作

```
ServicePool
        .getReposService()
        .getReposList(userName)
        .subscribeOn(Schedulers.newThread())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new Subscriber<List<Repos>>() {
            @Override
            public void onCompleted() {
                pbWait.setVisibility(View.GONE);
            }

            @Override
            public void onError(Throwable e) {
                pbWait.setVisibility(View.GONE);
                Toast.makeText( context: ReposActivity.this,  text: "获取出错",Toast.LENGTH_SHORT).show();
            }

            @Override
            public void onNext(List<Repos> repos) {
                reposList.addAll(repos);
                reposAdapter.notifyDataSetChanged();
            }
        });
```

耗时操作在新线程

最后回调是在主线程

（3） 实验遇到困难以及解决思路

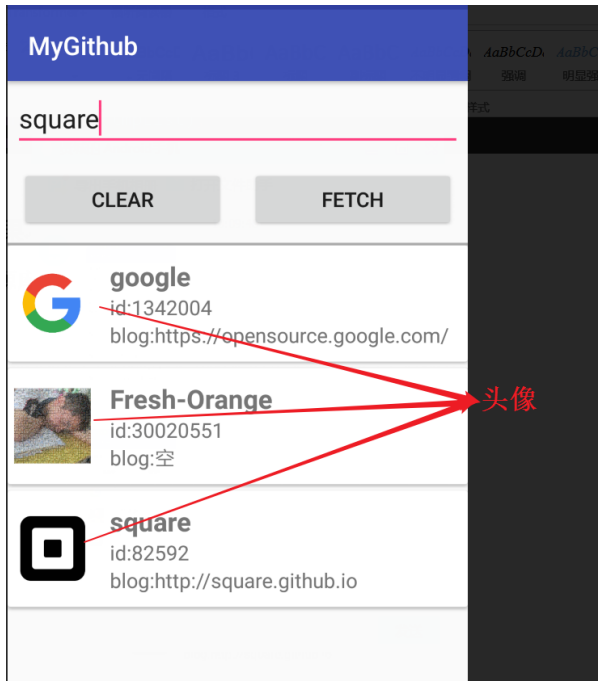
问题：实验上没遇到什么问题，就是不懂 retrofit 使用注解就能完成 url 拼接的原理？

解决：通过学习注解以及自己对 retrofit 的理解我认为这里的注解应该是运行时 retrofit 通过反射机制提取出注解然后根据注解将参数值与注解的地址接合，最后再将其接到 base_url 后面，就构成了符合 RESTful 的 api 的 url

```
ArrayList<BirthdayBean> tmpList = db.queryAll();
birthdayBeans.clear();
birthdayBeans.addAll(tmpList);
birthAdapter.notifyDataSetChanged();
```

四、 课后实验结果

使用 glide 开源库，将 user 信息中的头像显示出来



1. 用户的 model 中添加头像这一项，这样 gson 就会将得到的 json 里面的头像的信息填入 user 对象中

```
public class User {
    private String login;
    private String blog;
    private String id;
    private String avatar_url;

    public String getLogin() { return login; }

    public String getBlog() { return blog; }

    public String getId() { return id; }

    public String getAvatar_url() { return avatar_url; }
}
```

2. 然后在 RecyclerView 的 adapter 中的 onBindViewHolder 函数中使用 glide 将获得的头像的 url 设置进入头像对应的 imageview 里面，接下去的事，glide 就会帮忙搞定的！

```java
@Override
public void onBindViewHolder(final UsersVH holder, int position) {
    //直接在这里设置数据
    User user = userList.get(position);
    String login = user.getLogin();
    String id = "id:"+ user.getId();
    String blog = "blog:"+ (TextUtils.isEmpty(user.getBlog()) ? "空" : user.getBlog());
    String avatar_url = userList.get(position).getAvatar_url();
    holder.tvLogin.setText(login);
    holder.tvId.setText(id);
    holder.tvBlog.setText(blog);
    Glide.with(context).load(avatar_url).into(holder.avatar);
```

五、 实验思考及感想

1. 一些优秀的开源库真的很方便，很值得深入去学习。例如在没学 rxJava 之前，异步的操作经常要写得要不就是很乱要不就是耦合度太高，稍微修改就很难受，而 rxJava 却能在逻辑越来越复杂的情况下也能保持一样的简洁。 虽然这次实验我们只用到了 rxJava 最基础的部分，但我已经很喜欢这个库了！
2. 为了理解 retrofit 的运行机制，这次学习了 Java 中的注解，发现原来 Java 也可以那么灵活，使用贴注解然后通过反射提取注解的技术还能做出很多有意思的事情，有时间一定深入去学学

作业要求：

1. 命名要求: 学号_姓名_实验编号，例如 15330000_林 XX_lab1。
2. 实验报告提交格式为 pdf。
3. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。