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：郑贵锋

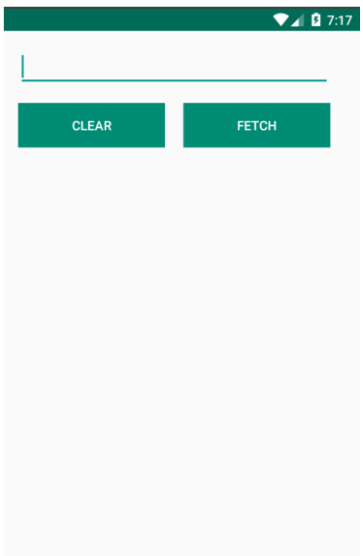
年级	15 级	专业（方向）	软件工程(移动)
学号	15352133	姓名	黄少豪
电话	13727024545	Email	328730316@qq.com
开始日期	2017.12.22	完成日期	2017.12.23

一、 实验题目

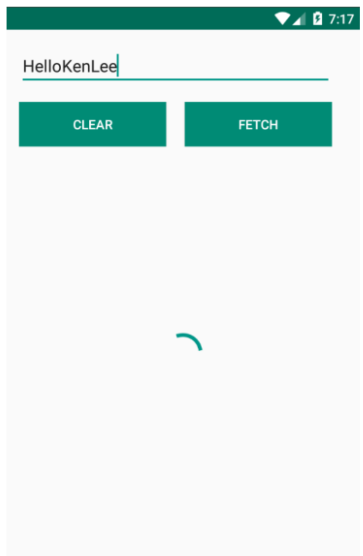
Retrofit+RxJava+OkHttp 实现网络请求

二、 实现内容

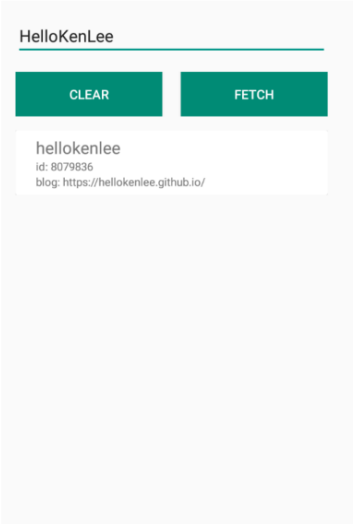
1. 在输入框输入内容，点击 FETCH 按钮可从 github 获取以输入框内容作为 login 值的 json 文件，将 json 文件中的 login，id，blog 作为信息显示在主界面的列表中。
2. 点击 CLEAR 按钮可将主界面列表清空
3. 长按列表项目可删除列表项目
4. 点击列表项目可进入该用户的 repos 信息，以列表形式显示每个 repo 的 name，language，description 信息



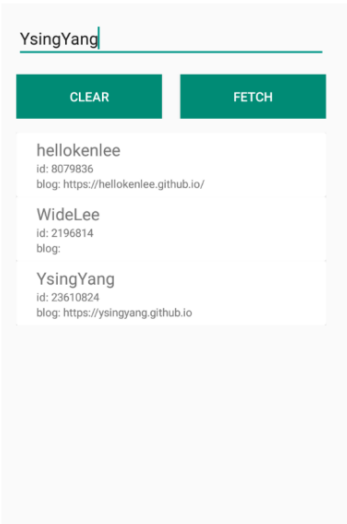
主界面



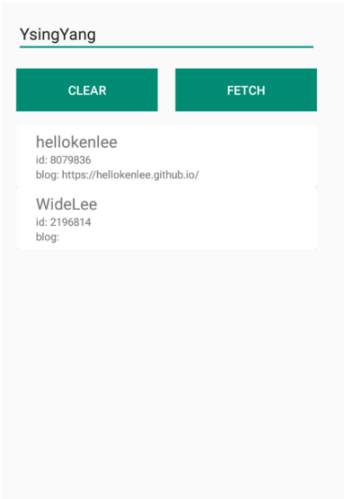
搜索用户



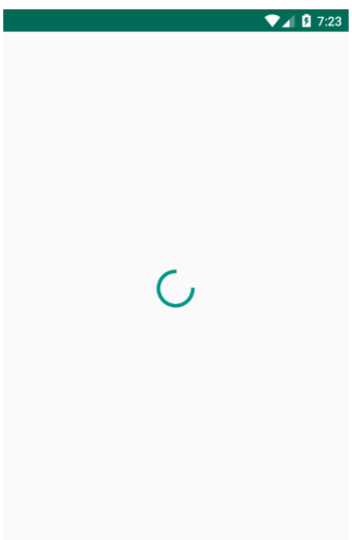
搜索结果



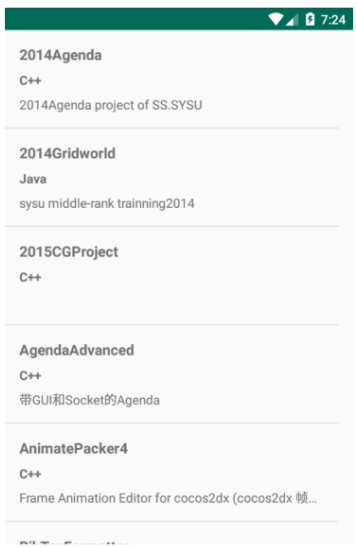
搜索结果



长按删除



点击进入个人详情页面

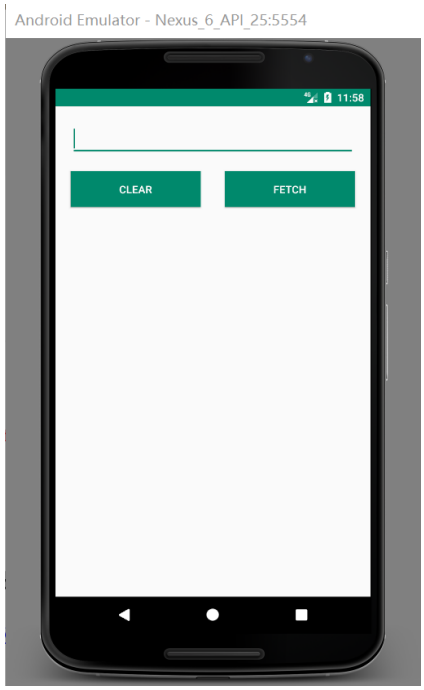


详情信息获取显示

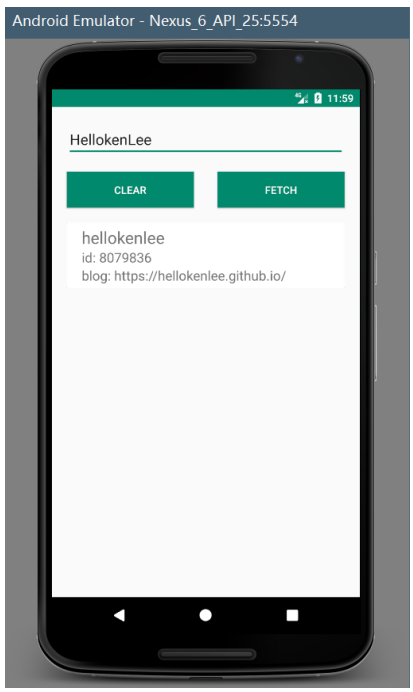
三、 课堂实验结果

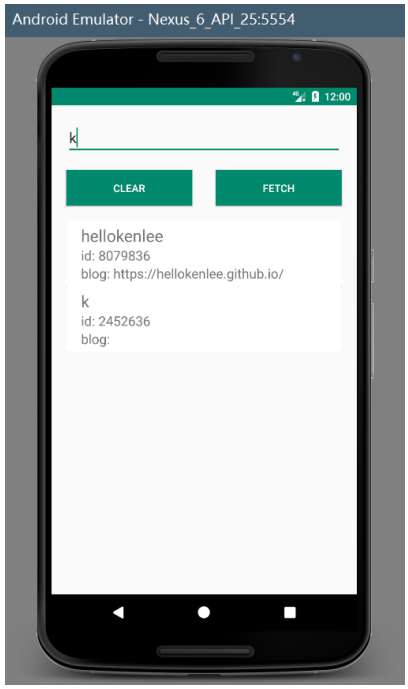
（1） 实验截图

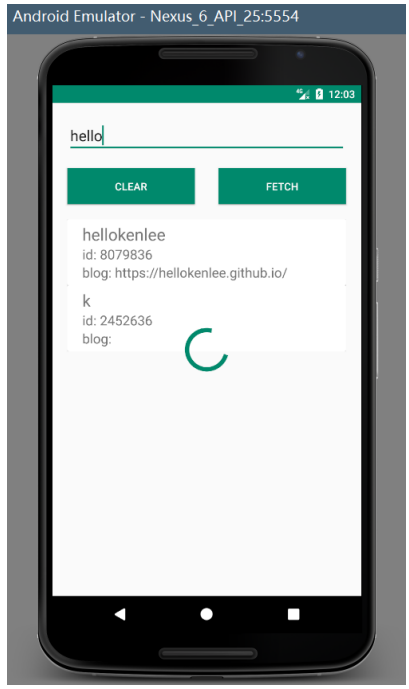
1. 主界面



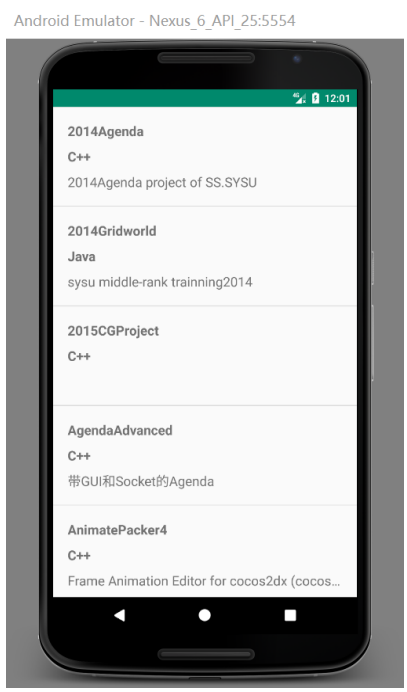
2. 输入内容，点击 FETCH 按钮



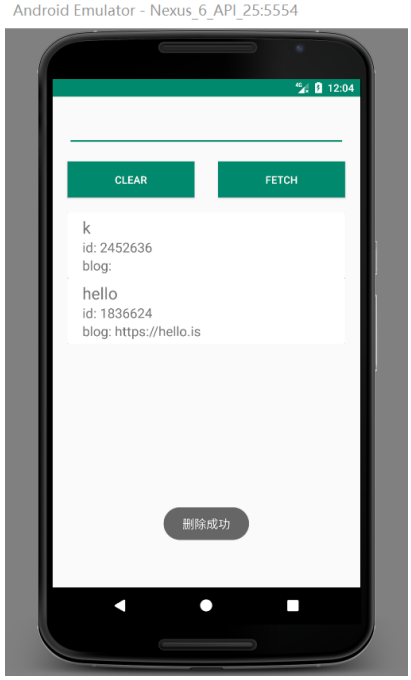




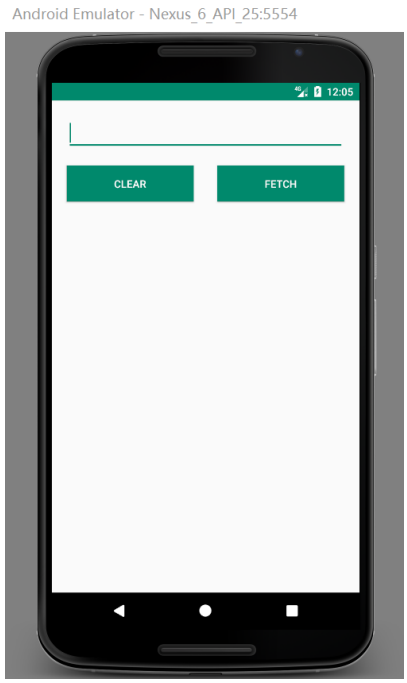
3. 单击第一项进入 repos 信息界面



4. 长按删除第一项



5. 点击 CLEAR 按钮清空列表



（2） 实验步骤以及关键代码

1. 主界面 UI 的搭建

```
<android.support.v7.widget.RecyclerView
    android:id="@+id/activity_main_list"
    android:layout_width="0px"
    android:layout_height="wrap_content"
    app:layout_constraintLeft_toLeftOf="@id/activity_main_search"
    app:layout_constraintRight_toRightOf="@id/activity_main_search"
    app:layout_constraintTop_toBottomOf="@id/activity_main_clear"
    android:layout_marginTop="20dp">
</android.support.v7.widget.RecyclerView>

<ProgressBar
    android:id="@+id/acticity_main_progressbar"
    android:layout_width="70dp"
    android:layout_height="70dp"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    android:visibility="invisible"/>
```

主要是两个 button 下的 RecyclerView 以及 ProgressBar

2. 给 FETCH 按钮设置点击事件

```
String user = content.getText().toString().trim();
//输入框内容
GithubService service = ServiceFactory.createService();
//新建一个网络请求服务
progressBar.setVisibility(View.VISIBLE);
//设置等待条可见
service.getUser(user)
        .subscribeOn(Schedulers.newThread())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new Subscriber<Github>() {
            @Override
            public void onNext(Github github) {
                Map<String, String> map = new HashMap<String, String>();
                map.put("login", github.getLogin());
                map.put("blog", "blog: " + github.getBlog());
                map.put("id", "id: " + github.getId());
                list.add(map);
                cardAdapter.notifyDataSetChanged();
                progressBar.setVisibility(View.INVISIBLE);
            }
```

基本步骤是(1)获取输入框输入内容；(2)创建一个网络请求；(3)设置等待条可见；(4)创建一个线程发送网络请求，请求响应在本线程操作；(5)在主线程获得网络请求响应，根据 Github 模型过滤得到需要的数据，将这些数据存入列表中；(6)更新列表内容，设置等待条不可见。

3. 给列表项添加点击事件，点击后进入 repos 界面；添加长按事件，长按后删除列表项。

```
cardAdapter.setOnItemClickListener(new CardAdapter.OnItemClickListener() {
    @Override
    public void onClick(int position) {
        if (cardAdapter.tag == false) return;
        Intent intent = new Intent(MainActivity.this, ReposActicity.class);
        intent.putExtra("name", list.get(position).get("login"));
        startActivity(intent);
    }

    @Override
    public void onLongClick(int position) {
        if (cardAdapter.tag == false) return;
        cardAdapter.tag = false;
        list.remove(position);
        cardAdapter.notifyDataSetChanged();
        Toast.makeText(MainActivity.this, "删除成功", Toast.LENGTH_SHORT).show();
        cardAdapter.tag = true;
    }
});
```

点击后，跳转到 repos 界面，将 login 值传递给 repos 界面让 repos 界面可以据此构造 url 获取数据。

4. Repos 界面是一个 ListView，ListView 使用 SimpleAdapter 显示数据

```
<ListView
    android:id="@+id/activity_repos_list"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
</ListView>
```

5. Repos 界面根据 MainActivity 界面传递过来的 login 值构造 url 获取 repos 信息。

```
GithubService service = ServiceFactory.createService();
progressBar.setVisibility(View.VISIBLE);
service.getRepos(name)
        .subscribeOn(Schedulers.newThread())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new Subscriber<List<Repos>>() {
            @Override
            public void onNext(List<Repos> repos) {
                for (Repos repo : repos){
                    Map<String, String> map = new HashMap<String, String>();
                    map.put("name", repo.getName());
                    map.put("description", repo.getDescription());
                    map.put("language", repo.getLanguage());
                    list.add(map);
                    simpleAdapter.notifyDataSetChanged();
                    progressBar.setVisibility(View.INVISIBLE);
                }
            }
```

因为 github 返回的 repos 的 json 数据是一个 List，所以获取数据时需要以 List<Repos>的形式接受，然后再枚举遍历这个 List 获得所需数据。

6. GithubService 接口定义

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

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

上面的 url 是获取主界面 user 信息的，下面的 url 是获取 repos 界面数据的。

7. Github 的 model 类型与 Repos 的 model 类型

```
public class Github {
    private String login;
    private String blog;
    private String id;

    public String getLogin() { return login; }

    public String getBlog() { return blog; }

    public String getId() { return id; }
}
```

```
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; }
}
```

（3） 实验遇到困难以及解决思路

1. 手机访问 url 出现 404 错误。

一开始使用模拟器测试，模拟器是可以上网的，但是运行应用时却没有数据显示出来。查看 debug 信息发现是网络请求 404。后来发现原来是没有在 AndroidManifest.xml 文件中加上网络访问权限，加上网络访问权限九就解决问题了。

```
<uses-permission android:name="android.permission.INTERNET"/>
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="lab9_code"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
```

2. 访问 github 的 user 数据没有问题，但是访问 user 的 repos 信息时却出错。

主界面可以获取 github 上的 user 信息，但是点击进入 user 的 repos 信息界面时却没有数据显示出来。后来发现原因是使用 github 的 api 获取 user 信息时，返回的 json 文件只有一个键值对集合。而使用 github 的 api 获取 user 的 repos 数据时，返回的 json 文件有若干个键值对集合，这些键值对集合构成了一个 List。于是解决方法是修改 GithubService 的 getRepos 的返回对象类型，使其返回的对象类型是一个 List。然后在收到网络请求的响应时，遍历这个 List 将所有键值对集合的所需数据提取出来。

```java
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; }
}
```

```java
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; }
}
```

```java
GithubService service = ServiceFactory.createService();
progressBar.setVisibility(View.VISIBLE);
service.getRepos(name)
        .subscribeOn(Schedulers.newThread())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new Subscriber<List<Repos>>() {
            @Override
            public void onNext(List<Repos> repos) {
                for (Repos repo : repos){
                    Map<String, String> map = new HashMap<String, String>();
                    map.put("name", repo.getName());
                    map.put("description", repo.getDescription());
                    map.put("language", repo.getLanguage());
                    list.add(map);
                    simpleAdapter.notifyDataSetChanged();
                    progressBar.setVisibility(View.INVISIBLE);
                }
            }
```

四、 课后实验结果

五、 实验思考及感想

在代码中创建 Retrofit 的语句如下：

```
private static Retrofit createRetrofit(String baseUrl) {
    return new Retrofit.Builder()
            .baseUrl(baseUrl)
            .addConverterFactory(GsonConverterFactory.create())
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
            .client(createOkHttp())
            .build();
}
```

我的理解是这样的：Retrofit 对象会使用创建的 OkHttp 对象发起网络请求，网络请求的 url 以传入的 baseUrl 作为基地址。Retrofit 对象会使用设置的“格式变换工厂”将返回的请求响应数据转换为我们自己定义的数据模型。这里我们是使用 GsonConverterFactory 将返回的 json 数据转换成我们自己定义的数据模型对象。而将 addCallAdapterFactory 设置为 RxJavaCallAdapterFactory 后，就可以使用 RxJava 来处理 Retrofit 的网络数据交互了。创建好 Retrofit 对象后，我们还需要给这个对象设置好资源路径与响应数据的数据转换模型，这里是使用一个 service 接口实现的。代码如下：

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

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

当给 Retrofit 对象的 create 方法传入这个接口时，会得到一个 service 接口对象。当调用这个接口的某个方法时，需要传入一个字符串，传入的字符串会替换方法上方 GET 注解中花括号包住的字符串(包括花括号)得到一个 url。这个 url 和上面提到的 Retrofit 对象的 baseUrl 共同构成一个资源 url。方法会返回一个 Observable 对象，响应的数据类型会被上面提到的 Retrofit 对象通过 GsonConverter 对象转换成尖括号内指示的数据类型。之后我们就可以使用 RxJava 的链式法则来处理这个 Observable 对象，使发送网络请求，接受网络请求响应，处理请求响应数据这些操作沿着代码链一气呵成。