# 中山大学数据科学与计算机学院本科生实验报告

## (2017 年秋季学期)

课程名称：手机应用平台开发　　任课教师： 刘宁

| | | | |
|---|---|---|---|
| 年级 | 15 级 | 专业（方向） | 软件工程 |
| 学号 | 15331236 | 姓名 | 马朝露 |
| 电话 | 15521122675 | Email | 1073627941@qq.com |
| 开始日期 | 2017/12/14 | 完成日期 | 2017/12/16 |

## 一、 实验题目

Retrofit+RxJava+OkHttp 实现网络请求

## 二、 实现内容

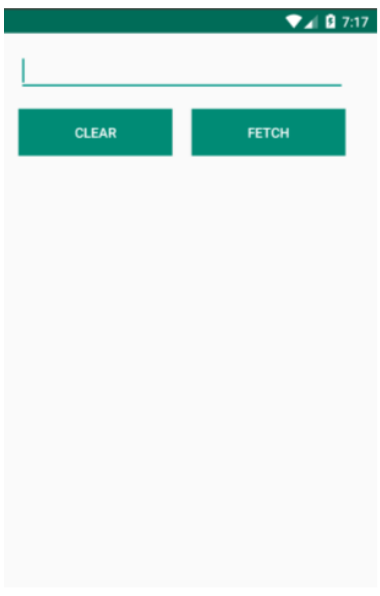
1、 实验目的

学习使用 Retrofit 实现网络请求

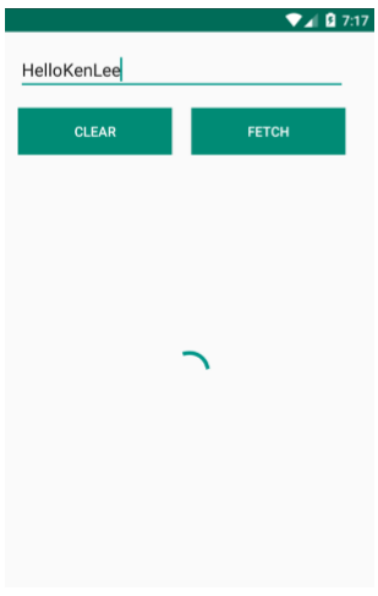
学习 RxJava 中 Observable 的使用

复习同步异步概念

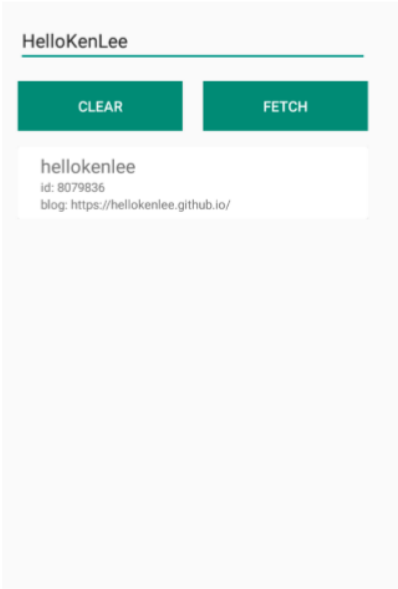
2、 实验内容



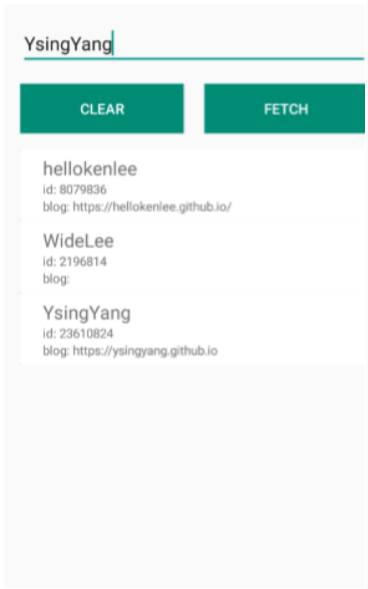

主界面



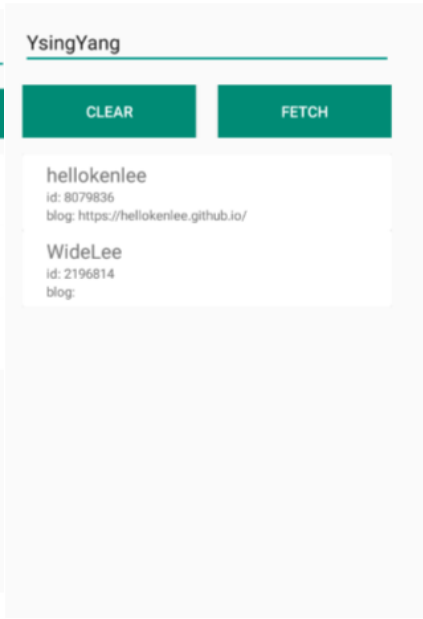

搜索用户



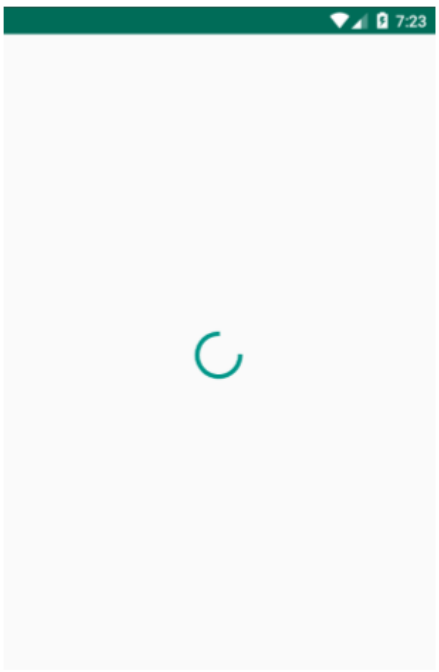

搜索结果



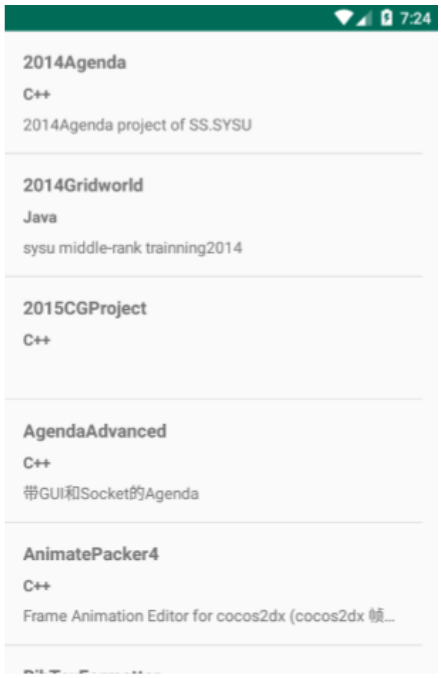

搜索结果




长按删除




点击进入个人详情页面




详情信息获取显示

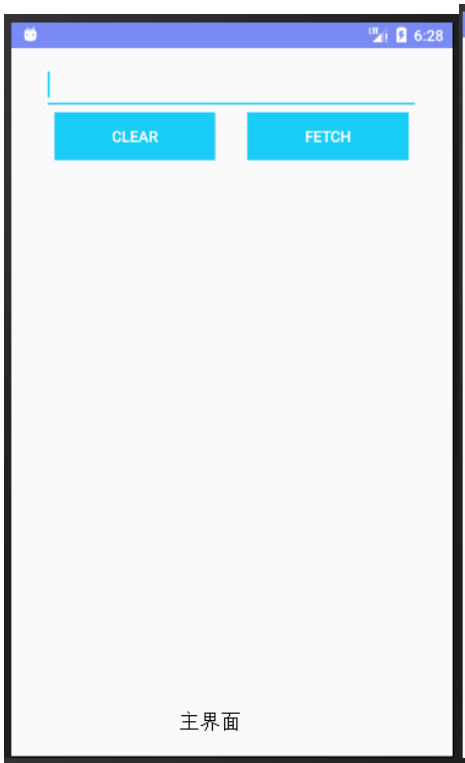
对于 User Model, 显示 id, login, blog

对于 Repository Model, 显示 name, description, language

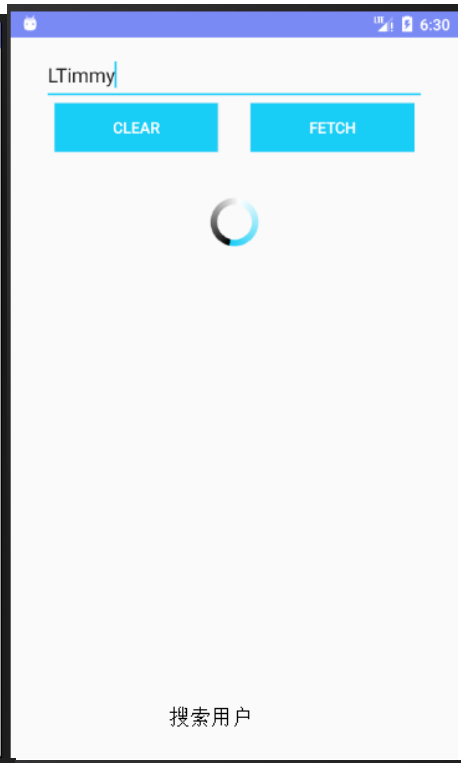
(特别注意， 如果 description 对于 1 行要用省略号代替)

## 三、 课堂实验结果

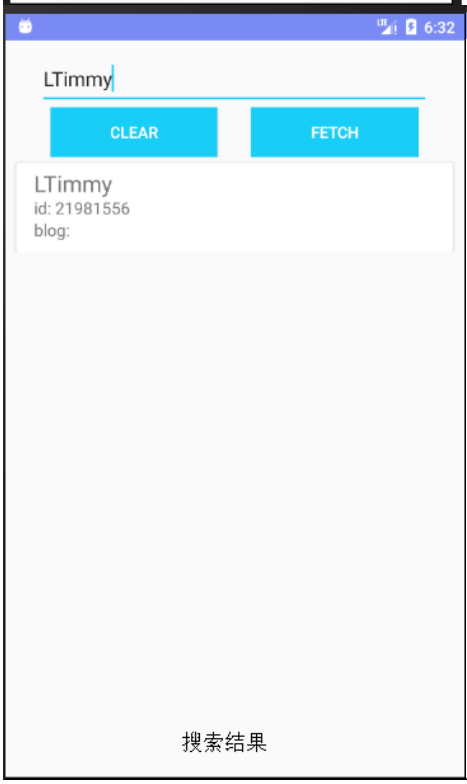
（1） 实验截图



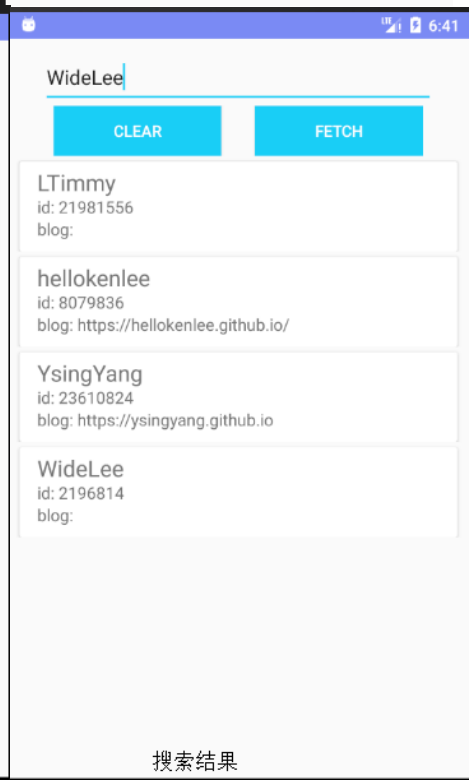

主界面



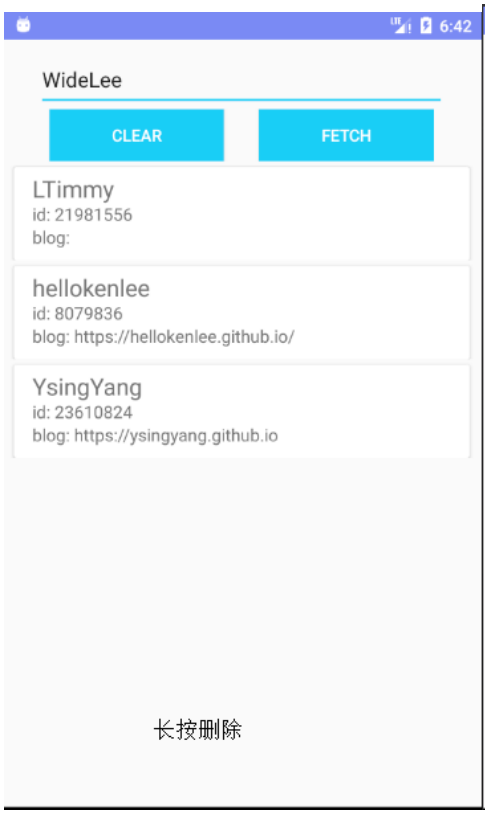

搜索用户



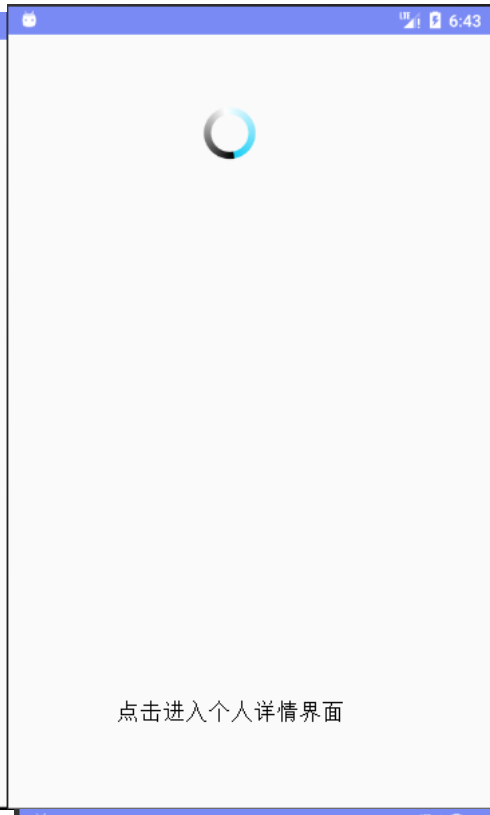

搜索结果




搜索结果




长按删除




点击进入个人详情界面


个人详情界面


框出部分超出文本框显示省略号

(2) 实验步骤以及关键代码

【1】 主界面的布局

只截取了重要部分：

```
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <android.support.v7.widget.RecyclerView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/github_list"
            >

        </android.support.v7.widget.RecyclerView>
        <ProgressBar
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/pro_bar"
            android:layout_marginTop="40dp"
            android:layout_marginLeft="165dp"
            android:indeterminateDrawable="@drawable/progress_small"
            android:visibility="invisible"
            />
    </RelativeLayout>


</LinearLayout>
```

总体使用 LinearLayout，progressbar 和 RecyclerView 部分在一起使用依赖布局，这里注意 processbar 使用的是自定义的图案，如下：

```
<?xml version="1.0" encoding="utf-8"?>
<rotate xmlns:android="http://schemas.android.com/apk/res/android"
    android:fromDegrees="0"
    android:pivotX="50%"
    android:pivotY="50%"
    android:toDegrees="360">
    <shape
        android:innerRadiusRatio="3"
        android:shape="ring"
        android:thicknessRatio="8"
        android:useLevel="false">
        <gradient
            android:centerColor="#FFFFFF"
            android:centerX="0.50"
            android:endColor="#19cef6"
            android:startColor="#000000"
            android:type="sweep"
            android:useLevel="false">
        </gradient>

    </shape>

</rotate>
```

同时也要注意 processbar 必须放到 RecyclerView 下面，否则在加载用户时它将会被 RecyclerView 的 item 覆盖。

【2】 实现 Github 类和 repository 类

```
public class Github {
    private String login;
    private String blog;
    private int id;
    public String getLogin() {return login;}
    public String getBlog() {return blog;}
    public int getId() {return id;}
}

public class Repositories {
    private String name;
    private String language;
    private String description;
    public String getName() {return name;}
    public String getLanguage() {return language;}
    public String getDescription() {return description;}

}
```

实现 github 类和 repository 类是为了使用谷歌的开源库 Gson 解析 json 数据。这样使得解析数据十分方便，只需一行代码即可。

【3】 实现 githubservice 和 reposervice

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

注意这里{}内的是我们将传入的形参，其实这里就是我们从 editText 中获得的要查询的用户

```
public interface RepoService {
    @GET("/users/{user}/repos")
    Observable<List<Repositories>> getUser(@Path("user") String user);
}
```

同理这个接口的作用是根据用户名查询 github 用户的 repository。根据 api 我们得到的是一个 list，所以根据这个接口去访问网络返回的是一个 repository 的 list

【4】 实现 RecyclerView

```
public class CardViewadpter extends RecyclerView.Adapter<CardViewadpter.ViewHolder> {

    private List<Github> data = new ArrayList<~>();
    private OnItemClickListener mOnItemClickListener;
    private OnItemLongClickListener mOnItemLongClickListener;

    public CardViewadpter( List<Github> data) { this.data = data; }
    public void addData(Github github) {
        data.add(github);
        notifyDataSetChanged();                          增
    }
    public void removeData(int position) {
        data.remove(position);
        notifyItemRemoved(position);                     删除一个
    }
    public void deleteall() {
        if (!data.isEmpty()) {
            data.clear();
            notifyDataSetChanged();                      全部删除
        }

    }
}
public  interface OnItemClickListener{
    void onClick(View view, int position);
}
public interface OnItemLongClickListener{
    void onLongClick(View view, int position);
}
public void setOnItemClickListener(OnItemClickListener onItemClickListener) {
    this.mOnItemClickListener=onItemClickListener;
}
public void setOnItemLongClickListener(OnItemLongClickListener onItemLongClickListener) {
    this.mOnItemLongClickListener=onItemLongClickListener;
}
```

实现点击 Item 接口，长按 Item 接口

```
public CardViewadpter.ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    // 绑定布局
    View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.github_item, null);
    // 创建ViewHolder
    ViewHolder viewHolder = new ViewHolder(view);
    return viewHolder;
}


@Override
public void onBindViewHolder(final ViewHolder holder, int position) {
    holder.loginview.setText(data.get(position).getLogin());
    holder.idview.setText("id: "+ Integer.toString(data.get(position).getId()));
    holder.blogview.setText("blog: "+data.get(position).getBlog());
    if(mOnItemClickListener != null){
        //为ItemView设置监听器
        holder.itemView.setOnClickListener((v) -> {
                int position = holder.getLayoutPosition(); // 1
                mOnItemClickListener.onClick(holder.itemView,position); // 2
        });
    }
    if(mOnItemLongClickListener != null){
        holder.itemView.setOnLongClickListener((v) -> {
                int position = holder.getLayoutPosition();
                mOnItemLongClickListener.onLongClick(holder.itemView,position);
                //返回true 表示消耗了事件 事件不会继续传递
                return true;
        });
    }
}
```

```
    @Override
    public int getItemCount() { return data.size(); }

    public class ViewHolder extends RecyclerView.ViewHolder {
        public TextView loginview;
        public TextView idview;
        public TextView blogview;
        public ViewHolder(View itemView) {
            super(itemView);
            loginview = (TextView)itemView.findViewById(R.id.login);
            idview = (TextView)itemView.findViewById(R.id.idv);
            blogview = (TextView)itemView.findViewById(R.id.blog);
        }
    }
}
```

这次实现的 RecyclerView 比上次作业简单，因为这次作业我直接在 adapter 中绑定了需要的布局文件，所以这次新建 adapter 时，只需传入一个 list 类型的数据，上次 adapter 需要根据 id 来获取布局文件。但是上次作业的 adapter 可以通用。

【5】 Mainactivity 网络访问

```
<uses-permission android:name="android.permission.INTERNET"/>
```

添加权限

```
private static Retrofit createRetrofit(String u) {
    return new Retrofit.Builder().baseUrl(u)
            .addConverterFactory(GsonConverterFactory.create())
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
            .client(createOkHttp())
            .build();
}
private static OkHttpClient createOkHttp() {
    OkHttpClient okHttpClient = new OkHttpClient.Builder()
            .connectTimeout(10, TimeUnit.SECONDS) // 连接超时
            .readTimeout(30, TimeUnit.SECONDS)  // 读超时
            .writeTimeout(10, TimeUnit.SECONDS) // 写超时
            .build();
    return okHttpClient;
}
```

createRetrofit 以及 OkHttpClient

```
Retrofit retrofit = createRetrofit(baseurl);
service = retrofit.create(GithubService.class);
```

new retrofit 与 service

```
fetch.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        progressBar.setVisibility(View.VISIBLE);   → 点击按钮开始进行网络搜索显示processbar
        service.getUser(msearch.getText().toString())   → 向service中传入实参，即要搜索的用户名
                .subscribeOn(Schedulers.newThread())    // 新建线程进行网络访问
                .observeOn(AndroidSchedulers.mainThread())  // 在主线程处理请求结果
                .subscribe(new Subscriber<Github>() {   → 切换回主线程进行的操作
                    @Override
                    public void onCompleted() {
                        progressBar.setVisibility(View.INVISIBLE);
                        System.out.println("完成传输");
                    }
                    @Override
                    public void onError(Throwable e) {
                        progressBar.setVisibility(View.INVISIBLE);
                        Toast.makeText(MainActivity.this, e.hashCode()
                        + "请确认你搜索的用户存在", Toast.LENGTH_SHORT).show();
                        Log.e("Github", "test"+e.getMessage());
                    }
                    @Override
                    public void onNext(Github github) {
                        progressBar.setVisibility(View.INVISIBLE);
                        cardViewadpter.addData(github);   → RecyclerView中添加item
                    }
                });
    }
});
```

【6】 Repoactivity 的 layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent">
    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        >
        <ProgressBar
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/pro_bar1"
            android:layout_marginLeft="165dp"
            android:layout_marginTop="60dp"
            android:indeterminateDrawable="@drawable/progress_small"
            android:visibility="invisible"
            />
        <ListView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/repo_list">

        </ListView>
    </RelativeLayout>

</LinearLayout>
```

这里使用的是 listview

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="100dp"
    android:id="@+id/repo_item"
    >
    <TextView
        android:layout_marginLeft="10dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/repo_name"
        android:text="Android"
        android:textStyle="bold"
        android:textSize="20dp"
        android:layout_marginTop="10dp"/>
    <TextView
        android:layout_marginLeft="10dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/language"
        android:text="Java"
        android:textStyle="bold"
        android:layout_marginTop="10dp"/>
    <TextView
        android:layout_marginLeft="10dp"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/description"
        android:text="my android project lalallalallalalalaaaaaaaaaaaaaaaaaaaaaaaaa"
        android:singleLine="true"
        android:layout_marginTop="10dp"
        />
```

用于设置只能一行，默认的是超出一行就显示省略号

repo_litem 的 layout，方框标出了如何实现超出一行显示省略号

【7】 repoactivity 的实现

```
private  List<Map<String, Object>> mDatas = new ArrayList<>();
```

存储信息的 list

```
Bundle bundle = this.getIntent().getExtras();
String name = bundle.getString("name");
```

获取从 mainactivity 传来的用户名，用于进行网络访问

```
progressBar.setVisibility(View.VISIBLE);
```

将 processbar 设为可视

```
Retrofit retrofit = createRetrofit(baseurl);
service = retrofit.create(RepoService.class);
```

new retrofit 与 service（之间要写好 createRetrofit 函数，这里就不截图了，和 mainactivity 一样）

```
service.getUser(name)
        .subscribeOn(Schedulers.newThread())    // 新建线程进行网络访问
        .observeOn(AndroidSchedulers.mainThread())  // 在主线程处理请求结果
        .subscribe(new Subscriber<List<Repositories>>() {   → 线程中返回的是一个list
            @Override
            public void onCompleted() {
                progressBar.setVisibility(View.INVISIBLE);   → 切回主线程时将processbar设为不可视
                System.out.println("完成传输");
            }
            @Override
            public void onError(Throwable e) {
                progressBar.setVisibility(View.INVISIBLE);
                Log.e("Github", "test"+e.getMessage());
            }
            @Override
            public void onNext(List<Repositories> list) {
                progressBar.setVisibility(View.INVISIBLE);
                if (list.size() == 0) {   → 判断此人是否有repository
                    Toast.makeText(RepoActivity.this,
                            "此用户没有Repository", Toast.LENGTH_SHORT).show();
                }
                for (int i = 0; i < list.size(); i++) {
                    Map<String, Object> map = new HashMap<>();
                    map.put("name",list.get(i).getName());
                    map.put("language",list.get(i).getLanguage());
                    map.put("description", list.get(i).getDescription());
                    mDatas.add(map);   → 将从线程中获得的数据在主线程中加入adapter的data中
                    adapter.notifyDataSetChanged();
                    Log.d("github", list.get(i).getName());
                    Log.d("github", list.get(i).getLanguage());
                    Log.d("github", list.get(i).getDescription());
                }
            }
```

## 四、 实验思考及感想

实验中遇到的问题：

【1】 访问网络时，搜索用户名，log.e 中总是提示不存在

解决方法：baseurl 设置时多加了一个/,去掉即可

【2】 每次只要 notifydatasetchange，就会出错。

解决方法：自定义 adapter 中的 CreateViewHolder 函数返回类型应该是 CardViewadpter.ViewHolder 而不是 ViewHolder。

【3】 Processbar 被 item 覆盖

解决方法：我使用的是相对布局，默认是后来的 view 会覆盖在前面的 view 上，所以只要将 processbar 放到 RecyclerView 上面就好了

【4】 不理解 Rxjava2 中的线程切换等等

解决方法：看 TA 给的链接

【5】 不知道如何获取一个用户 repository

解决方法：仔细看作业 PPT，发现其实 TA 有提示，网络访问返回的是一个 list，同时 repository 与 github 类似。所以舍弃将 repository 改为 list 的想法，而在 interface 中将返回类型设置为 repository 的 list

实验感想：

【1】 学习了如何使用 retrofit 访问网络，学习了如何使用 Gson 解析 json 数据，方便快捷。由于这次实验只用了一种方法访问网络，所以对其他的方法不是很了解，自己只是简单的尝试了用 httpurlconnnection 访问，其他的方法没有尝试。这次实验只使用了 get，没有用到 post 等等。安卓网络访问还有待更深入的尝试

【2】 学习了 Rxjava2，虽然完成了作业，看了博客但总觉得没有深入透彻的理解，有待继续学习。

【3】 这次实验是最后一次实验，感觉学的东西还不够，安卓还需要更深入的去学习，刘老师讲课透彻易懂，是开发课上遇到的最好的老师。同时 ta 认真负责，ta 师兄超帅，师姐超美超温柔，谢谢 ta 以及辛苦 ta 一学期的帮助与指导。