deepwaterooo deepwaterooMe – I am the same GitHub account person

deepwaterooo

September 5, 2022

Contents

1 要求 1

2 主要思路 2

3 下载图片并保存到本地：rxjava 2.x+retrofit 通过动态 url 保存网络图片到本地 3

4 关于图片的处理：不仅要下载，下载后还需要自动保存到数据库 4

5 用 Retrofit+Rxjava 上传图片支持多张图片的上传 6

1 要求

- Overview
  - Build an **employee directory app** that shows a list of employees from the provided endpoint.
  - The app should display a list (or any kind of **collection view**!) which shows all the employees returned from the JSON endpoint described below.
  - Each item in the view should contain a **summary of the employee**, including their **photo, name, and team** at minimum. You may add more information to the summary if you want, or **sort employees in any fashion** you'd like –sort and group by name, team, etc.
  - There should be some UX to reload the employee list from within the app at any time. The UX can be done in any way you want: **a button, pull-to-refresh**, etc.
  - If there is any additional UI/UX you would like to add, feel free to do so! We only ask that you please **do not build any more screens** than this list. Do not worry about building custom controls or UI elements –using **system-provided, standard elements** is totally fine.
  - Be sure to **appropriately handle the normal variety of errors when querying an endpoint**. The app should **display useful loading, empty, and error states** where appropriate. **If images fail to load, displaying a placeholder** is fine.
  - One extra thing we ask is that you please ensure you **do not use more network bandwidth than necessary –load expensive resources such as photos on-demand only**.

  - The **employee list should not be persisted to disk**. You can reload it from the network **on each app launch and when refresh is requested** —but no more often than that unintentionally. (Android developers in particular should take care **not to make redundant network calls** when the **phone is rotated, or when memory is low**).
  - **Images**, however, should **be cached on disk** so as to not waste device bandwidth. You may use an **open source image caching solution**, or write your own caching. Do not rely upon HTTP caching for image caching.
  - Note that photos at a given URL will never change. Once one is loaded, you do not need to reload the photo. If an employee's photo changes, they will be given a new photo URL.
  - Tests should be provided for the app. We do not expect 100% code coverage, so please use your best judgment for what should be tested. We're also interested only in unit tests. Feel free to skip snapshot or app tests.

- MVVM: 需要数据驱动，viewModel 里定义一个状态变量，来标记当前的活动状态
  - If any employee is malformed, it is fine to invalidate the entire list of employees in the response - there is no need to exclude only malformed employees.
  - If there are no employees to show, the app should present an **empty state** view instead of an empty list.

2 主要思路

- 这是一个看似要求极其简单，实则考验的知识点和深度有着相当的跨度的小项目。
- 它们一定挑都要挑我出差到 WSU 的一个星期里来考验我，因为他们就是想要去打败一个人。呵呵，真正想要打败一个人，谈何容易，就凭这???
- Retrofit + RxJava: 好像是更合适的，可以用注解，并且用得更为广泛
  - 搜索关键字：Retrofit + OkHttp +RxJava 网络库构建
  - **OkHttp**: 网络请求处理, 主要是在应用启动的时候，什么时机开始发布和调用网络请求。所以这个可以不用了，大家都喜欢新的更好用的库

- **图片本地缓存**: 第三方库找一个，还是用 AndroidX 的 Room
  - 我 **现在数据库的问题**是：我 **缓存保存了员工数据进数据库**，但是这里说得很清楚了，**不用保存员工数据，只保存每个员工 id 所对应的图片就可以了**
- 现在的难点：不知道怎么定义图片数据库，同时以 OkHTTP respnose 回来的连接起来
- 应用的 **启动优化**：重中之重，需要借助这个小应用弄懂弄清楚，**不知道如何拆解网络请求的步骤，什么时候加载，初始化之类的？**以达到较好的启动优化
-
- **MVVM 设计**：只有一个页面，相对就简单方便多了。工作中的案例是使用 MVVM 但自己编辑逻辑处理信号下发，与数据驱动的 UI 更新，没有实现双向数据绑定的；可是这里感觉 **双向数据绑定**更简单，会有哪些可能的问题呢？这里基本可以当作不需要双向，因为一个 UI 按钮要求刷新是唯一的 UI 需求；更多的只是需要时候的数据往 UI 加载更新；所以 **可以简单使用观察者模式，UI 观察数据的变化**就可以了

- **图片的加载与处理**：用样可以使用么第三方库 **glide**
- **图片的加载与处理**：用样可以使用么第三方库 **CircularImageView**
- **AndroidX RecyclerView** 的使用：选择相对更为高效和方便管理的库和数据结构来使用
- **Constraint Layout vs Coordinate Layout**：暂时先用任何简单的 layout 先能运行起一个大致的框架来，再进一步优化
- 我丢掉了的文件呀，我写过的项目呀，不是在进 Lucid 之前写得好好的一个项目，现在源码全丢了。。。。。该死的 GitHub. . . ..

3 下载图片并保存到本地：rxjava 2.x+retrofit 通过动态 url 保存网络图片到本地

```
// HttpManager 类：就是一个通过单例模式实现的类，获取 retrofit 的一个实例来调用 NetApi 接口内声明的方法，此处只写关键的一部分，别的
public <T> T getHttpApi(Class<T> service) {
    Retrofit retrofit = new Retrofit.Builder()
        .baseUrl(BASE_URl)
        .client(getClient())
        .addConverterFactory(GsonConverterFactory.create())
        .addCallAdapterFactory(RxJava2CallAdapterFactory.create())
        .build();

    return retrofit.create(service);
}
// BASE_URl 是你定义的域名比如：http://www.xxxx.com:8080 之类的

// NetApi 接口：
@GET
@Streaming
Observable<ResponseBody> downloadImg(@Url String  imgUrl);

// 注意注解：
// @GET 后面不加任何东西，平时的都是 @GET("api/getuserinfo") 之类的和上面的那个 BASE_URl 拼接起来生成 url：
// http://www.xxxx.com:8080/api/getuserinfo? 请求条件 =xx
// 然后去请求，这里采用 @Url 注解的方式就不用那么麻烦了
// @Url 此处是动态 url 即网络图片的 url，需要从外部传入，如度娘图标 url：
// https://www.baidu.com/img/superlogo_c4d7df0a003d3db9b65e9ef0fe6da1ec.png
// 用字符串的形式传入即可

// Presenter 类 ： 发起网络请求把得到的图片二进制流转化为 bitmap 对象，再通过 bitmap 对象保存到本地指定目录下
/**
 * 指定线程下载文件 (异步)，非阻塞式下载
 * @param url       图片 url
 * @param savePatch 下载文件保存目录
 * @param fileName  文件名称 (不带后缀)
 */
public void downloadFile(String url, final String savePatch, final String fileName) {
    HttpManager.getInstance().getHttpApi(NetApi.class)
        .downloadImg(url)
        .subscribeOn(Schedulers.io())
        .observeOn(Schedulers.newThread())
        .subscribe(new DisposableObserver<ResponseBody>() {
                @Override
                    public void onNext(ResponseBody responseBody) {
                    Bitmap bitmap = null;
                    byte[] bys;
                    try {
                        bys = responseBody.bytes();
                        bitmap = BitmapFactory.decodeByteArray(bys, 0, bys.length);

                        try {
                            FileUtils.saveImg(bitmap, savePatch, fileName);
                            String savePath = savePatch + File.separator + fileName + ".jpg";
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
```

```
                    if (bitmap != null) {
                        bitmap.recycle();
                    }
                }
                @Override
                    public void onError(Throwable e) {
                    //你的处理
                }
                @Override
                    public void onComplete() {
                    //你的处理
                }
            });
}
// decodeByteArray 是 BitmapFactory 内的方法，把二进制流转化为 bitmap，需要导入系统包：
// import android.graphics.BitmapFactory;

// FileUtils 类：IO 操作，把图片保存到本地：
/**
 * 保存图片到 SD 卡
 * @param bm          图片 bitmap 对象
 * @param floderPath 下载文件保存目录
 * @param fileName    文件名称 (不带后缀)
 */
public static void saveImg(Bitmap bm, String floderPath, String fileName) throws IOException {
    //如果不保存在 sd 下面下面这几行可以不加
    if (!Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)) {
        Log.e("SD 卡异常");
        return;
    }
    File folder = new File(floderPath);
    if (!folder.exists()) {
        folder.mkdirs();
    }
    String savePath = folder.getPath() + File.separator + fileName + ".jpg";
    File file = new File(savePath);
    BufferedOutputStream bos = new BufferedOutputStream(new FileOutputStream(file));
    bm.compress(Bitmap.CompressFormat.JPEG, 80, bos);
    Log.d(savePath + " 保存成功");
    bos.flush();
    bos.close();
}
// 在你的 service 或者 activity 中调用：
mPresenter.downloadFile("https://www.baidu.com/img/superlogo_c4d7df0a003d3db9b65e9ef0fe6da1ec.png", Environment.getExternal
```

4 关于图片的处理：不仅要下载，下载后还需要自动保存到数据库

- https://blog.csdn.net/ANDROID_WangWeiDa/article/details/62284675
- 主要源码参考如下：

```
/**
 * 观察者
 */
Observer<String> observer = new Observer<String>() {
    @Override
    public void onCompleted() {
        Log.e("TAG", "oncompleted()");
    }
    @Override
    public void onError(Throwable e) {
        Log.e("TAG", "onError()");
    }
    @Override
    public void onNext(String s) {
        Log.e("TAG", "onNext()" + s);
    }
};
// 或者创建观察者的实现类：Subscriber
/**
 * 观察者 (观察者的实现类)
 */
Subscriber<String> subscriber = new Subscriber<String>() {
    @Override
    public void onCompleted() {
```

```
        Log.e("TAG", "oncompleted()");
    }
    @Override
    public void onError(Throwable e) {
        Log.e("TAG", "onError()");
    }
    @Override
    public void onNext(String s) {
        Log.e("TAG", "onNext()" + s);
    }
};
// 可以说，两者的效果是一样的。
// 接着创建可观察者（被观察者）Observable

/**
 * 可观察者（被观察者）
 */
Observable observale = Observable.create(new Observable.OnSubscribe<String>() {
        @Override
        public void call(Subscriber<? super String> subscriber) {
            subscriber.onNext("Hello");
            subscriber.onNext("My name is Avater!");
            subscriber.onCompleted();
        }
    });
// 好了，到此已经创建完毕，接着在 onCreate 方法中进行简单的调用：
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    observale.subscribeOn(Schedulers.io())  //订阅在 io 线程（非主线程），不会阻塞主线程
        .observeOn(AndroidSchedulers.mainThread())  //在主线程中观察
        .subscribe(observer);   //进行订阅关系
}
// Log:
// 03-15 12:06:45.837 2952-2952/com.avater.myapplication E/TAG: onNext()Hello
// 03-15 12:06:45.847 2952-2952/com.avater.myapplication E/TAG: onNext()My name is Avater!
// 03-15 12:06:45.847 2952-2952/com.avater.myapplication E/TAG: oncompleted()
// 是不是很快？是不是很懵逼？哈哈，这就对了，毕竟入门嘛，多实战，多理解！
// 下面附上一个使用 Rxjava 下载图片的例子：

private ImageView imageView;
private String url = "https://ss0.bdstatic.com/5aV1bjqh_Q23odCf/static/superman/img/logo/bd_logo1_31bdc765.png";

/**
 * 图片观察者
 */
Observer<Bitmap> bitmapOberver = new Observer<Bitmap>() {
    @Override
    public void onCompleted() {

    }
    @Override
    public void onError(Throwable e) {
        Toast.makeText(MainActivity.this, " 图片下载失败", Toast.LENGTH_SHORT).show();
    }
    @Override
    public void onNext(Bitmap bitmap) {
        imageView.setImageBitmap(bitmap);
    }
};

/**
 * 可观察者（被观察者）
 */
Observable<Bitmap> bitmapObservable = Observable.create(new Observable.OnSubscribe<Bitmap>() {
        @Override
        public void call(Subscriber<? super Bitmap> subscriber) {
            URL net;
            HttpURLConnection conn = null;
            InputStream inputStream = null;
            Bitmap bitmap = null;
            try {
                net = new URL(url);
                conn = (HttpURLConnection) net.openConnection();
                inputStream = conn.getInputStream();
                bitmap = BitmapFactory.decodeStream(inputStream);
            } catch (MalformedURLException e) {
                e.printStackTrace();
```

```
            } catch (IOException e) {
                e.printStackTrace();
            } finally {
                conn.disconnect();
                try {
                    inputStream.close();
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
            subscriber.onNext(bitmap);
        }
    });
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    imageView = (ImageView) findViewById(R.id.imageview);

    bitmapObservable.subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(bitmapOberver);
}
```

5 用 Retrofit+Rxjava 上传图片支持多张图片的上传

```
// 1. 这是一个接口
@POST
Observable<ResponseBody> Image(@Url String url, @HeaderMap Map<String,Object> headermap,@Body MultipartBody body);
// 第一个是上传一个 第二个是上传多个

// 下面这个是一个 Retrofit 封装好的工具类
public class Retrofits{
    private MyApiService myApiService;
    public Retrofits() {
        HttpLoggingInterceptor loggingInterceptor =new HttpLoggingInterceptor();
        loggingInterceptor.setLevel(HttpLoggingInterceptor.Level.BODY);
        OkHttpClient okHttpClient =new OkHttpClient.Builder()
            .readTimeout(20,TimeUnit.SECONDS)
            .connectTimeout(20,TimeUnit.SECONDS)
            .writeTimeout(20,TimeUnit.SECONDS)
            .addInterceptor(loggingInterceptor)
            .retryOnConnectionFailure(true)
            .build();
        Retrofit retrofit =new Retrofit.Builder()
            .addConverterFactory(GsonConverterFactory.create())
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
//                  存放的头文件
            .baseUrl(Contacts.BASE_URL)
            .client(okHttpClient)
            .build();
        myApiService =retrofit.create(MyApiService.class);
    }
    public static  Retrofits getInstance(){
        return RetroHolder.OK_UTIL;
    }
    static class RetroHolder{
        private static final Retrofits OK_UTIL =new Retrofits ();
    }
    /**
     * 封装一个上传图片
     */
    public OkUtil image(String murl,Map<String,Object> headermap,Map<String,Object> map,List<Object> list){
        MultipartBody.Builder builder = new MultipartBody.Builder().setType(MultipartBody.FORM);
        if (list.size()==1) {
            for (int i = 0; i < list.size(); i++) {
                File file = new File((String) list.get(i));
                builder.addFormDataPart("image", file.getName(),RequestBody.create(MediaType.parse("multipart/octet-stream"
            }
        }
        myApiService.Image(murl,headermap,builder.build())
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(observer);
        return Retrofits.getInstance();
    }
```

```
    /**
     * 多个图片的上传
     */
    public OkUtil pinglun(String murl,Map<String,Object> headermap,Map<String,Object> map,List<Object> list){
        MultipartBody.Builder builder = new MultipartBody.Builder().setType(MultipartBody.FORM);
        builder.addFormDataPart("commodityId",String.valueOf(map.get("commodityId")));
        if(!String.valueOf(map.get("orderId")).equals("")){
            builder.addFormDataPart("orderId",String.valueOf(map.get("orderId")));
        }
        builder.addFormDataPart("content",String.valueOf(map.get("content")));
        if (list.size()!=0) {
            for (int i = 1; i < list.size(); i++) {
                File file = new File((String) list.get(i));
                builder.addFormDataPart("image", file.getName(),RequestBody.create(MediaType.parse("multipart/octet-stream"
            }
        }
        myApiService.Image(murl,headermap,builder.build())
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(observer);
        return Retrofits.getInstance();
    }
//    重写一个观察者模式
    private Observer observer =new Observer<ResponseBody>(){
        @Override
        public void onCompleted() {
        }
        @Override
        public void onError(Throwable e) {
            if(httpListener!=null){
                httpListener.onError(e.getMessage());
            }
        }
        @Override
        public void onNext(ResponseBody responseBody) {
            if(httpListener !=null){
                try {
                    httpListener.onSuccess(responseBody.string());
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }
    };
    public interface HttpListener{
        void onSuccess(String gsonstr);
        void onError(String error);
    }
    private HttpListener httpListener;
    public void setHttpListener(HttpListener listener){
        this.httpListener =listener;
    }
}

// 一个方法把得到的图片路径 变为 String 类型
public String getFilePath(String fileName, int requestCode, Intent data) {
    if (requestCode == 1) {
        return fileName;
    } else if (requestCode == 0) {
        Uri uri = data.getData();
        String[] proj = {MediaStore.Images.Media.DATA};
        Cursor actualimagecursor = managedQuery(uri, proj, null, null, null);
        int actual_image_column_index = actualimagecursor
            .getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
        actualimagecursor.moveToFirst();
        String img_path = actualimagecursor
            .getString(actual_image_column_index);
        // 4.0 以上平台会自动关闭 cursor, 所以加上版本判断,OK
        if (Build.VERSION.SDK_INT < Build.VERSION_CODES.ICE_CREAM_SANDWICH)
            actualimagecursor.close();
        return img_path;
    }
    return null;
}

// 一个打开图库的方法
Intent intent1 = new Intent(Intent.ACTION_PICK);
intent1.setType("image/*");
startActivityForResult(intent1,0);
```

```java
// 重写一个回调方法
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(data==null){
        return;
    }
    if(requestCode==0){
        String filePath = getFilePath(null,requestCode,data);
        /**
         * 这里是用的一个图片的上传
         */
        Map<String, Object> map = new HashMap<>();
        List<Object> list =new ArrayList<>();
        list.add(filePath);
        pressent.image(Contacts.UploadYourHead, headermap, map,list,Register.class);
    }
}
```