Calculator Android NDK SDK Design

deepwaterooo

2019 年 4 月 27 日

目录

1 Kotlin 在 Android 端的基本用法详解

- https://blog.csdn.net/jay100500/article/details/79090636
- 项目源码：https://github.com/jaychou2012/KotlinDemo

2 理解 AOP

- http://www.cnblogs.com/yanbincn/archive/2012/06/01/2530377.html
- 有上例子没有运行通

3 Java 注解（Annotation）原理详解

- https://blog.csdn.net/lylwo317/article/details/52163304
- HelloAnnotation 注解（接口）是一个继承了 Annotation 接口的特殊接口，而我们通过反射获取注解时，返回的是 Java 运行时生成的动态代理对象 $Proxy1，该类就是 HelloAnnotation 注解（接口）的具体实现类。

- 注解本质是一个继承了 Annotation 的特殊接口，其具体实现类是 Java 运行时生成的动态代理类。通过代理对象调用自定义注解（接口）的方法，会最终调用 AnnotationInvocationHandler 的 invoke 方法。该方法会从 memberValues 这个 Map 中索引出对应的值。而 memberValues 的来源是 Java 常量池。

4 android 小白不敢触及 NDK? 潇涧

- https://www.zhihu.com/question/40899975
- 可能没有接触过 NDK 的 Android 开发者多少都会有点这个疑惑吧，至少我曾经也有过，真的很佩服有些人能精通那么多方面的知识，可以构建出一些很牛逼很“高大上”的应用。
- 我觉得，要想消除心中的疑惑，最简单的方法就是去尝试它！设想你面前有扇门，你看着这扇门很高很笨重，感觉不容易推开，但是没准这些都只是表象，我们不去尝试的话怎么知道呢？
- 说下我的经历，我刚读研那会在三星中国研究院实习过一段日子，当时就是从零开始自己摸索 NDK，功夫不负有心，可算是和师兄一起做出了一个能够追踪人眼球在手机屏幕上的聚焦点的应用。其中，native 层的算法是师兄利用之前做好的 PC 端的算法移植出来的，而我的工作只是简单的 jni 调用而已，但是那次实习让我感觉 NDK 挺有意思的。再后来，学校某实验室有个老师需要做一个 Android 端的人脸识别应用，我在朋友的推荐下开始了自己的第二次 NDK 开发经历，花了一周时间写了一个 demo，老师很高兴，所以后来一直支持我继续做。但是，继续做下去就会发现很多知识上的不足，一方面 C++ 功底不够，另一方面 Android 功底也不深，所以基本上就是一路踩坑的过程。参加工作后，在 Flyme 系统组的时候发现之前折腾的这些 NDK 对于我理解 Android 系统的构建、Android 的核心模块原理都有不少帮助呢。
- 总而言之，我觉得你需要消除这方面的疑惑，而消除疑惑最简单的办法就是去尝试下，首先你可以看下其他几位高分答题者推荐的资源，确定下你是否感兴趣，是否有时间和精力，如果已经具备了一定的基础之后，不防看些优秀的项目，下面推荐两个：
  - 1.renard314/textfairy
    - https://github.com/renard314/textfairy
    - 这是一个 Android 端的 OCR 应用，完成度相当高，而且比较复杂，希望你能成功编过它
  - 2.MasteringOpenCV/code
    - https://github.com/MasteringOpenCV/code
    - 原书的第一个项目是和 NDK 有关的：Cartoonifier and Skin Changer for Android，将摄像头捕捉的画面动漫化。
    - 看完这个可以接着看后面的第 8 个项目：Face Recognition using Eigenfaces or Fisherfaces，使用经典的 Eigenfaces 和 Fisherfaces 来做人脸识别，这章其实不是一个 Android 应用，但是，你可以将其移植到 Android 端，加油吧！
- 我自己以前折腾的时候写过一些总结,但是年代可能有点久远,参考意义不大了,如有需要请移步阅读 Android • Hujiawei Bujidao

5 AndroidStudio 使用 OpenCV 的三种方式

- https://dongxiawu.github.io/2017/12/14/AndroidStudio%E4%BD%BF%E7%94%A8OpenCV%E7%9A%84%E4%B89%E7%A7%8D%E6%96%B9%E5%BC%8F/
- 项目源码：https://github.com/dongxiawu/OpencvDemo

6 Dagger2 原理解析、理解

- 详解 Dagger2 系列，原来 Dagger2 如此简单
  - https://juejin.im/entry/578cf2612e958a00543c45a4
- 详解 Dagger2 系列之撸码篇：横看成岭侧成峰
  - https://dreamerhome.github.io/2016/07/11/dagger%20for%20code/
- 依赖注入神器：Dagger2 详解系列
  - https://dreamerhome.github.io/2016/07/07/dagger/
- 可运行的 github 项目 (**这个项目可运行**)，根据自己的参数改编 **dagger2Demo**
  - https://github.com/luxiaoming/dagger2Demo
- Android 神兵利器 Dagger2 使用详解（一）基础使用 **Sample_dagger2**
  - https://blog.csdn.net/mq2553299/article/details/73065745
  - github 项目源码 (**这个项目可运行**):sample_dagger2 https://github.com/qingmei2/Sample_dagger2
- Android 神兵利器 Dagger2 使用详解（二）Module&Component 源码分析
  - https://blog.csdn.net/mq2553299/article/details/73136396
- Android 神兵利器 Dagger2 使用详解（三）MVP 架构下的使用
  - https://blog.csdn.net/mq2553299/article/details/73251405
- Android 神兵利器 Dagger2 使用详解（四）Scope 注解的使用及源码分析
  - https://blog.csdn.net/mq2553299/article/details/73414710
-
  -
-
  -

7 关于抓取天气以及代码实现的可能性

- OkHttp3-Android 网络请求框架常用用法介绍与实例（mob 请求天气预报）自己正在尝试使用体会的
  - https://blog.csdn.net/donkor_/article/details/53589316
- 简诗
  - https://www.jianshu.com/p/39dce598faf1
- android 开发练习三-Retrofit 一个失败的例子，还没有弄好
  - https://blog.csdn.net/fnz1111/article/details/50752562
- ACCU 天气、Okhttp、Retrofit、RxJava 的综合使用
  - https://www.jianshu.com/p/3f0f850057eb
- android 基于 MVP,RxJava2,Retrofit2 的天气 app

  - https://www.jianshu.com/p/aff298f92b76

- 真实案例出发，再谈 retrofit 封装 ANT
  - https://juejin.im/entry/57ff5c6d816dfa0056e04621
- –
- –

8 Dagger2+Retrofit+RxJava 练习请求天气的小应用

- https://www.jianshu.com/p/fac63cd4ff01
- 请求天气的小应用，练习如何使用 Dagger2+Retrofit+RxJava

8.1 gradle 配置

- project/build.gradle

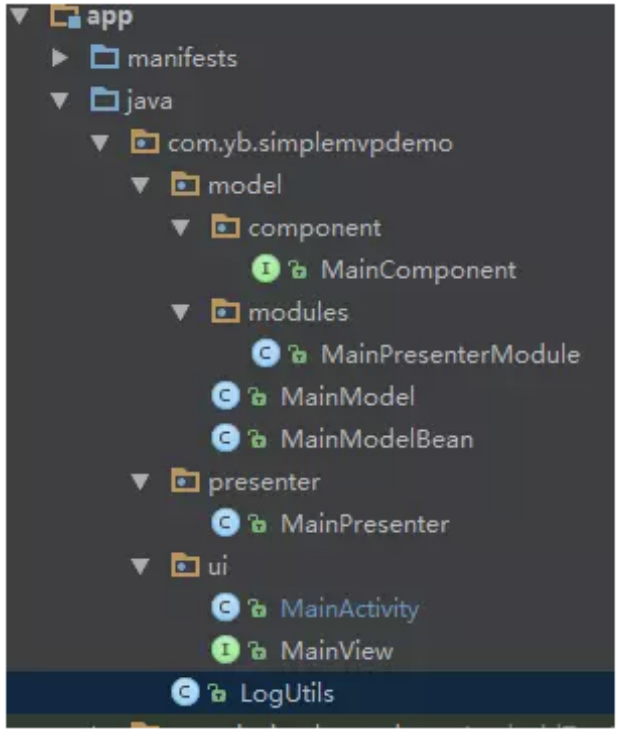
```
dependencies {
        classpath 'com.android.tools.build:gradle:2.0.0'
        classpath 'me.tatarka:gradle-retrolambda:3.2.4'
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
}
```

- project/app/build.gradle

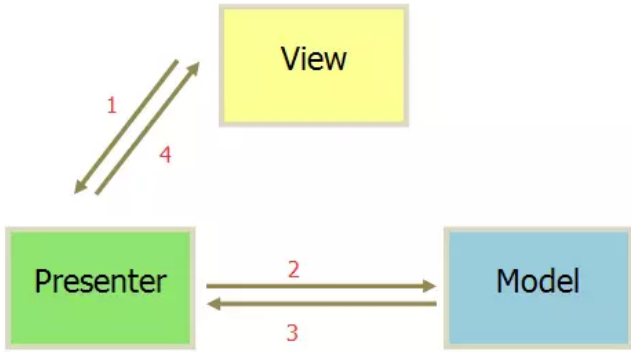
```
apply plugin: 'com.android.application'
apply plugin: 'me.tatarka.retrolambda'
apply plugin: 'com.neenbedankt.android-apt'
android {
              ......
        compileOptions {
                 sourceCompatibility JavaVersion.VERSION_1_8
                 targetCompatibility JavaVersion.VERSION_1_8
        }
              ......
}
dependencies {
        ......
```

```
14          compile 'com.jakewharton:butterknife:7.0.1'
15
16          compile 'com.squareup.retrofit2:retrofit:2.0.1'
17          compile 'com.squareup.retrofit2:converter-gson:2.0.2'
18          compile 'com.squareup.retrofit2:adapter-rxjava:2.0.0'
19
20          compile 'io.reactivex:rxandroid:1.1.0'
21          compile 'io.reactivex:rxjava:1.1.0'
22
23          compile 'com.google.dagger:dagger:2.1'
24          apt 'com.google.dagger:dagger-compiler:2.1'
25          provided 'org.glassfish:javax.annotation:10.0-b28'
26  }
```

8.2 项目结构



- mvp 流程



- MainPresenter 作为 MainActivity 和 MainModel 之间的桥梁，当 MainActivity 需要请求数据时，通知 Main-

Presenter；MainPresenter 通知 MainModel 需要数据，MainModel 通过 Retrofit 从网络中获取数据，若是请求成功，则把数据传给 MainPresenter, 若是请求失败，则把失败信息传给 MainPresenter。

- MainPresenter 从 MainModel 中获取数据信息后通知 MainActivity 更新数据。
- MainActivity 中的 MainPresenter 实例是由 Dagger2 注入的单例。

```
public class MainActivity extends AppCompatActivity implements MainView {
    @Bind(R.id.tv_displayWeather) TextView tv;
    @Bind(R.id.progressBar) ProgressBar progressBar;

    @Inject @Singleton
    public MainPresenter myPresenter; //Dagger 不能注入私有变量

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        supportRequestWindowFeature(Window.FEATURE_NO_TITLE);//去掉标题栏
        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);//ButterKnife 注入
        initView();
    }
    public void displayWeather(View view) {
        myPresenter.getData();
    }
    @Override
    public void initView() {
        MainComponent component = DaggerMainComponent.builder().
            mainPresenterModule(new MainPresenterModule(this)).build();
        component.inject(this);
    }
    @Override
    public void showData(MainModelBean mybean) {
        if(mybean =  = null){
            tv.setText(" 请求失败");
        } else{
            MainModelBean.WeatherinfoEntity bean  =  mybean.getWeatherinfo();
            String city = bean.getCity();
            String wd = bean.getWD();
            String ws = bean.getWS();
            String time = bean.getTime();
            String data = " 城市:"+city+"\n 风向:"+wd+"\n 风级:"+ws+"\n 发布时间:"+time;
            tv.setText(data);
        }
        hideProgressBar();
    }
    public void showProgressBar(){
```

```
        progressBar.setVisibility(View.VISIBLE);
    }
    public void hideProgressBar(){
        progressBar.setVisibility(View.GONE);
    }
}
```

- MainModel 的代码如下：

```
public class MainModel {

    String baseUrl="http://www.weather.com.cn/adat/sk/";
    private List<MainModelBean> list = new ArrayList<>();
    private MainPresenter mainPresenter;

    public MainModel(MainPresenter mainPresenter){
        this.mainPresenter=mainPresenter;
    }
    public void getData(){
        Retrofit retrofit=new Retrofit.Builder()
            .baseUrl(baseUrl)
            .addConverterFactory(GsonConverterFactory.create())
            .addCallAdapterFactory(RxJavaCallAdapterFactory.create())
            .build();
        WeatherService service=retrofit.create(WeatherService.class);
        service.getModelBean("101010100")
            .subscribeOn(Schedulers.io())
            .observeOn(AndroidSchedulers.mainThread())
            .subscribe(new MySubscriber());
    }

    interface WeatherService{
        @GET("{cityId}"+".html")
        Observable<MainModelBean> getModelBean(@Path("cityId") String cityId);
    }
    class MySubscriber extends Subscriber<MainModelBean>{
        @Override
        public void onCompleted() {
            mainPresenter.loadDataSuccess(list);
        }
        @Override
        public void onError(Throwable e) {
            mainPresenter.loadDataFailure();
        }
        @Override
        public void onNext(MainModelBean mainModelBean) {
```

```
            list.add(mainModelBean);
        }
    }
}
```

- MainPresenter 源码如下：

```
public class MainPresenter {
    private MainView mainView;
    public MainModel mainModel = new MainModel(this);

    @Inject @Singleton
    public MainPresenter(MainView mainView) {
        this.mainView = mainView;
    }
    public void getData(){
        mainView.showProgressBar();
        mainModel.getData();
    }
    public void loadDataSuccess(List<MainModelBean> list) {
        mainView.showData(list.get(0));
    }
    public void loadDataFailure(){
        mainView.showData(null);
    }
}
```

- 源码地址：
  - https://github.com/changeyb/SimpleMVPDemo

9 Dagger2

- 项目级别 build.gradle

```
// Top-level build file where you can add configuration options common to all sub-projects

buildscript {
    repositories {
        google()
        jcenter()

    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.3.2' // 3.3.2
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'

        // NOTE: Do not place your application dependencies here; they belong
```

```
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
        google()
        jcenter()

    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

- 应用 APP 级别 build.gradle

```
apply plugin: 'com.android.application'
//apply plugin: 'com.neenbedankt.android-apt'

android {
    compileSdkVersion 28
    buildToolsVersion '28.0.3'
    defaultConfig {
        applicationId "com.me.daggertrial"
        minSdkVersion 26
        targetSdkVersion 28
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'progua
        }
    }
}

dependencies {
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    compileOnly 'com.android.support:appcompat-v7:28.0.0'
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'com.android.support:design:28.0.0'
    implementation 'com.google.dagger:dagger:2.7'
```

```
29      annotationProcessor 'com.google.dagger:dagger-compiler:2.7'
30      implementation 'org.glassfish:javax.annotation:10.0-b28'
31      //implementation 'com.google.dagger:dagger-android:2.11'
32      //implementation 'com.google.dagger:dagger-android-support:2.11'
33      //annotationProcessor 'com.google.dagger:dagger-android-processor:2.11'
34      testImplementation 'junit:junit:4.12'
35      androidTestImplementation 'com.android.support.test:runner:1.0.2'
36      androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
37  }
```