# 【Android】APT

带心情去旅行 关注

17 2018.04.23 00:21:03 字数 1,814 阅读 36,582

## 介绍

APT(Annotation Processing Tool)即**注解处理器**，是一种处理注解的工具，确切的说它是javac的一个工具，它用来在**编译时**扫描和处理注解。注解处理器以**Java代码**(或者编译过的字节码)作为输入，生成.java**文件**作为输出。
简单来说就是在编译期，通过注解生成.java文件。

## 作用

使用APT的优点就是方便、简单，可以少些很多重复的代码。

用过ButterKnife、Dagger、EventBus等注解框架的同学就能感受到，利用这些框架可以少些很多代码，只要写一些注解就可以了。
其实，他们不过是通过注解，生成了一些代码。通过对APT的学习，你就会发现，他们很强~~~


## 实现

说了这么多，动手试试

**目标**
通过APT实现一个功能，通过对 `View` 变量的注解，实现 `View` 的绑定（类似于 `ButterKnife` 中的 `@BindView` ）

（参考自这里）

**创建项目**
创建Android Module命名为app
创建Java library Module命名为 apt-annotation
创建Java library Module命名为 apt-processor 依赖 apt-annotation
创建Android library Module 命名为apt-library依赖 apt-annotation、auto-service

热门故事

凤凰男的醒悟：花了半辈子孝顺父母，换来妻离子散

扎心虐文：不是所有的女主最后都会选择原谅

她闺蜜的一条朋友圈，结束了我和老公5年的婚姻

被戳穿的爱情是骗局，没被戳穿的变成了信仰

推荐阅读

APT入门
阅读 483

Android APT技术学习
阅读 119

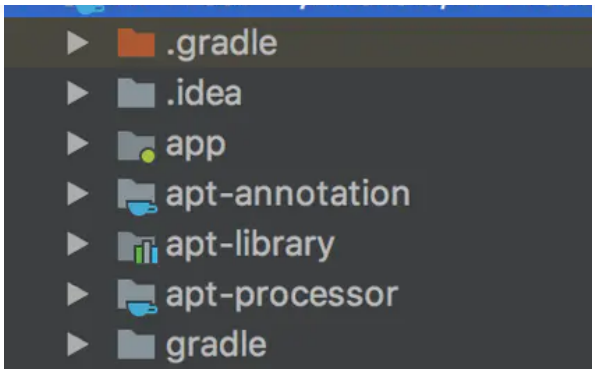
Protobuf在Android中的基本使用
阅读 595

Android Gradle 实战之自动生成DeepLink配置信息
阅读 319

1.Android架构 retrofit运行时注解（POST详解）+反射实战demo 2分...
阅读 409

# 【Android】APT

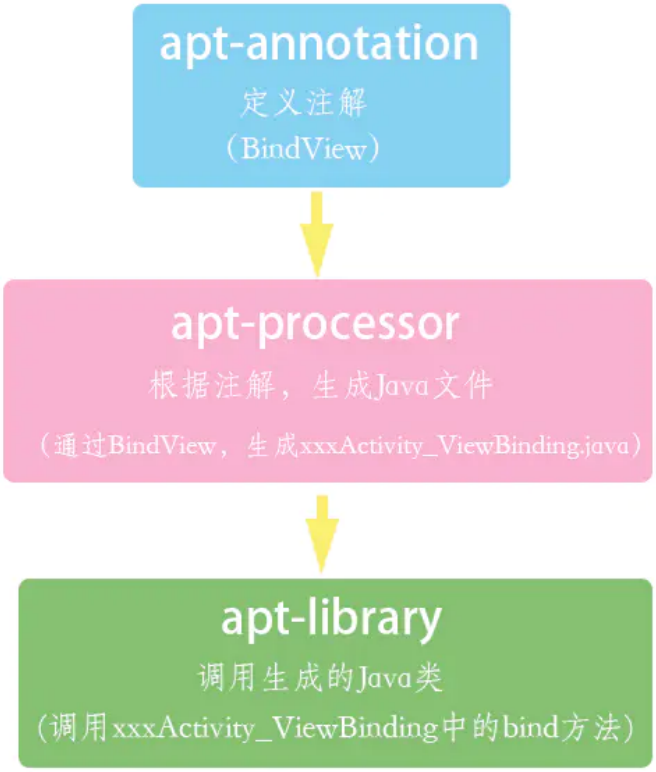
带心情去旅行 关注



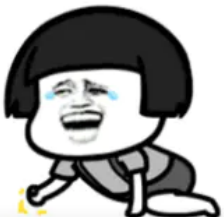

功能主要分为三个部分

- `apt-annotation`： 自定义注解，存放@BindView
- `apt-processor`： 注解处理器，根据 `apt-annotation` 中的注解，在编译期生成 `xxxActivity_ViewBinding.java` 代码
- `apt-library`： 工具类，调用 `xxxActivity_ViewBinding.java` 中的方法，实现 `View` 的绑定。

关系如下




app？app不是功能代码，只是用来验证功能的~~~



# 【Android】APT

带心情去旅行 关注

创建注解类 `BindView`

```
@Retention(RetentionPolicy.CLASS)
@Target(ElementType.FIELD)
public @interface BindView {
    int value();
}
```

`@Retention(RetentionPolicy.CLASS)`： 表示编译时注解

`@Target(ElementType.FIELD)`： 表示注解范围为类成员（构造方法、方法、成员变量）

> @Retention： 定义被保留的时间长短
> RetentionPoicy.SOURCE、RetentionPoicy.CLASS、RetentionPoicy.RUNTIME
> @Target： 定义所修饰的对象范围
> TYPE、FIELD、METHOD、PARAMETER、CONSTRUCTOR、LOCAL_VARIABLE等
> 详细内容

这里定义了运行时注解 `BindView`，其中 `value()` 用于获取对应 `View` 的 `id`。

## 2、apt-processor（注解处理器）

**(重点部分)**

在 `Module` 中添加依赖

```
dependencies {
    implementation 'com.google.auto.service:auto-service:1.0-rc2'
    // Gradle 5.0后需要再加下面这行
    // annotationProcessor  'com.google.auto.service:auto-service:1.0-rc2'
    implementation project(':apt-annotation')
}
```

> Android Studio升级到3.0以后，Gradle也随之升级到3.0。 `implementation` 替代了之前的 `compile`

创建 `BindViewProcessor`

```
@AutoService(Processor.class)
public class BindViewProcessor extends AbstractProcessor {

    private Messager mMessager;
    private Elements mElementUtils;
    private Map<String, ClassCreatorProxy> mProxyMap = new HashMap<>();

    @Override
    public synchronized void init(ProcessingEnvironment processingEnv) {
        super.init(processingEnv);
        mMessager = processingEnv.getMessager();
        mElementUtils = processingEnv.getElementUtils();
    }

    @Override
    public Set<String> getSupportedAnnotationTypes() {
        HashSet<String> supportTypes = new LinkedHashSet<>();
        supportTypes.add(BindView.class.getCanonicalName());
```

热门故事

凤凰男的醒悟：花了半辈子孝顺父母，换来妻离子散

扎心虐文：不是所有的女主最后都会选择原谅

她闺蜜的一条朋友圈，结束了我和老公5年的婚姻

被戳穿的爱情是骗局，没被戳穿的变成了信仰

推荐阅读

APT入门
阅读 483

Android APT技术学习
阅读 119

Protobuf在Android中的基本使用
阅读 595

Android Gradle 实战之自动生成DeepLink配置信息
阅读 319

1.Android架构 retrofit运行时注解（POST详解）+反射实战demo 2分...
阅读 409

智能井盖

# 【Android】APT

带心情去旅行 关注

```
    }

    @Override
    public boolean process(Set<? extends TypeElement> set, RoundEnvironment roundEnv) {
        //根据注解生成Java文件
        return false;
    }
}
```

- `init`：初始化。可以得到 `ProcessingEnviroment`， `ProcessingEnviroment` 提供很多有用的工具类 `Elements`, `Types` 和 `Filer`
- `getSupportedAnnotationTypes`：指定这个注解处理器是注册给哪个注解的，这里说明是注解 `BindView`
- `getSupportedSourceVersion`：指定使用的Java版本，通常这里返回 `SourceVersion.latestSupported()`
- `process`：可以在这里写扫描、评估和处理注解的代码，生成Java文件（**process中的代码下面详细说明**）

```
@AutoService(Processor.class)
public class BindViewProcessor extends AbstractProcessor {

    private Messager mMessager;
    private Elements mElementUtils;
    private Map<String, ClassCreatorProxy> mProxyMap = new HashMap<>();

    @Override
    public boolean process(Set<? extends TypeElement> set, RoundEnvironment roundEnvironment)
        mMessager.printMessage(Diagnostic.Kind.NOTE, "processing...");
        mProxyMap.clear();
        //得到所有的注解
        Set<? extends Element> elements = roundEnvironment.getElementsAnnotatedWith(BindView.c
        for (Element element : elements) {
            VariableElement variableElement = (VariableElement) element;
            TypeElement classElement = (TypeElement) variableElement.getEnclosingElement();
            String fullClassName = classElement.getQualifiedName().toString();
            ClassCreatorProxy proxy = mProxyMap.get(fullClassName);
            if (proxy == null) {
                proxy = new ClassCreatorProxy(mElementUtils, classElement);
                mProxyMap.put(fullClassName, proxy);
            }
            BindView bindAnnotation = variableElement.getAnnotation(BindView.class);
            int id = bindAnnotation.value();
            proxy.putElement(id, variableElement);
        }
        //通过遍历mProxyMap，创建java文件
        for (String key : mProxyMap.keySet()) {
            ClassCreatorProxy proxyInfo = mProxyMap.get(key);
            try {
                mMessager.printMessage(Diagnostic.Kind.NOTE, " --> create " + proxyInfo.getPro
                JavaFileObject jfo = processingEnv.getFiler().createSourceFile(proxyInfo.getPr
                Writer writer = jfo.openWriter();
                writer.write(proxyInfo.generateJavaCode());
                writer.flush();
                writer.close();
            } catch (IOException e) {
                mMessager.printMessage(Diagnostic.Kind.NOTE, " --> create " + proxyInfo.getPro
            }
        }

        mMessager.printMessage(Diagnostic.Kind.NOTE, "process finish ...");
        return true;
    }
}
```

通过 `roundEnvironment.getElementsAnnotatedWith(BindView.class)` 得到所有注解 `elements`，然后将 `elements` 的信息保存到 `mProxyMap` 中，最后通过 `mProxyMap` 创建对应的Java文件，其中 `mProxyMap` 是



## 热门故事

凤凰男的醒悟：花了半辈子孝顺父母，换来妻离子散

扎心虐文：不是所有的女主最后都会选择原谅

她闺蜜的一条朋友圈，结束了我和老公5年的婚姻

被戳穿的爱情是骗局，没被戳穿的变成了信仰

## 推荐阅读

APT入门
阅读 483

Android APT技术学习
阅读 119

Protobuf在Android中的基本使用
阅读 595

Android Gradle 实战之自动生成DeepLink配置信息
阅读 319

1.Android架构 retrofit运行时注解（POST详解）+反射实战demo 2分...
阅读 409

智能井盖

# 【Android】APT

带心情去旅行 关注

```java
public class ClassCreatorProxy {
    private String mBindingClassName;
    private String mPackageName;
    private TypeElement mTypeElement;
    private Map<Integer, VariableElement> mVariableElementMap = new HashMap<>();

    public ClassCreatorProxy(Elements elementUtils, TypeElement classElement) {
        this.mTypeElement = classElement;
        PackageElement packageElement = elementUtils.getPackageOf(mTypeElement);
        String packageName = packageElement.getQualifiedName().toString();
        String className = mTypeElement.getSimpleName().toString();
        this.mPackageName = packageName;
        this.mBindingClassName = className + "_ViewBinding";
    }

    public void putElement(int id, VariableElement element) {
        mVariableElementMap.put(id, element);
    }

    /**
     * 创建Java代码
     * @return
     */
    public String generateJavaCode() {
        StringBuilder builder = new StringBuilder();
        builder.append("package ").append(mPackageName).append(";\n\n");
        builder.append("import com.example.gavin.apt_library.*;\n");
        builder.append('\n');
        builder.append("public class ").append(mBindingClassName);
        builder.append(" {\n");

        generateMethods(builder);
        builder.append('\n');
        builder.append("}\n");
        return builder.toString();
    }

    /**
     * 加入Method
     * @param builder
     */
    private void generateMethods(StringBuilder builder) {
        builder.append("public void bind(" + mTypeElement.getQualifiedName() + " host ) {\n");
        for (int id : mVariableElementMap.keySet()) {
            VariableElement element = mVariableElementMap.get(id);
            String name = element.getSimpleName().toString();
            String type = element.asType().toString();
            builder.append("host." + name).append(" = ");
            builder.append("(" + type + ")(((android.app.Activity)host).findViewById( " + id +
        }
        builder.append("  }\n");
    }

    public String getProxyClassFullName()
    {
        return mPackageName + "." + mBindingClassName;
    }

    public TypeElement getTypeElement()
    {
        return mTypeElement;
    }
}
```

上面的代码主要就是从 `Elements` 、 `TypeElement` 得到想要的一些信息，如package name、Activity名、变量类型、id等，通过 `StringBuilder` 一点一点拼出 `Java` 代码，每个对象分别代表一个对应的 `.java` 文件。


## 热门故事

凤凰男的醒悟：花了半辈子孝顺父母，换来妻离子散

扎心虐文：不是所有的女主最后都会选择原谅

她闺蜜的一条朋友圈，结束了我和老公5年的婚姻

被戳穿的爱情是骗局，没被戳穿的变成了信仰

## 推荐阅读

APT入门
阅读 483

Android APT技术学习
阅读 119

Protobuf在Android中的基本使用
阅读 595

Android Gradle 实战之自动生成DeepLink配置信息
阅读 319

1.Android架构 retrofit运行时注解（POST详解）+反射实战demo 2分...
阅读 409


# 【Android】APT

带心情去旅行 关注

没想到吧！Java代码还可以这样写~~

提前看下生成的代码（不大整齐，被我格式化了）

```
public class MainActivity_ViewBinding {
    public void bind(com.example.gavin.apttest.MainActivity host) {
        host.mButton = (android.widget.Button) (((android.app.Activity) host).findViewById(213
        host.mTextView = (android.widget.TextView) (((android.app.Activity) host).findViewById
    }
}
```

**缺陷**

通过 `StringBuilder` 的方式一点一点来拼写Java代码，不但繁琐还容易写错~~

**更好的方案**

通过 `javapoet` 可以更加简单得生成这样的Java代码。(后面会说到)

> **介绍下依赖库auto-service**
>
> 在使用注解处理器需要先声明，步骤：
>
> 1、需要在 processors 库的 main 目录下新建 resources 资源文件夹；
>
> 2、在 resources文件夹下建立 META-INF/services 目录文件夹；
>
> 3、在 META-INF/services 目录文件夹下创建 javax.annotation.processing.Processor 文件；
>
> 4、在 javax.annotation.processing.Processor 文件写入注解处理器的全称，包括包路径；）
>
> 这样声明下来也太麻烦了？这就是用引入auto-service的原因。
>
> 通过auto-service中的@AutoService可以自动生成AutoService注解处理器是Google开发的，用来生成 META-INF/services/javax.annotation.processing.Processor 文件的

## 3、apt-library 工具类

完成了Processor的部分，基本快大功告成了。

在 `BindViewProcessor` 中创建了对应的 `xxxActivity_ViewBinding.java` ，我们改怎么调用？当然是**反射**啦！！！

在Module的 `build.gradle` 中添加依赖

```
dependencies {
    implementation project(':apt-annotation')
}
```

创建注解工具类 `BindViewTools`

```
public class BindViewTools {

    public static void bind(Activity activity) {

        Class clazz = activity.getClass();
        try {
            Class bindViewClass = Class.forName(clazz.getName() + "_ViewBinding");
            Method method = bindViewClass.getMethod("bind", activity.getClass());
            method.invoke(bindViewClass.newInstance(), activity);
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
```





# 【Android】APT

带心情去旅行 关注

```
        } catch (InvocationTargetException e) {
            e.printStackTrace();
        }
    }
}
```

`apt-library` 的部分就比较简单了，**通过反射**找到对应的 `ViewBinding` 类，然后**调用其中的 `bind()` 方法**完成 `View` 的绑定。

到目前为止，所有相关的代码都写完了，终于可以拿出来溜溜了

## 4、app

**依赖**

在Module的 `build.gradle` 中（Gradle>=2.2）

```
dependencies {
    implementation project(':apt-annotation')
    implementation project(':apt-library')
    annotationProcessor project(':apt-processor')
}
```

> Android Gradle 插件 2.2 版本的发布，Android Gradle 插件提供了名为 `annotationProcessor` 的功能来完全代替 `android-apt`

（若Gradle<2.2）

在Project的 `build.gradle` 中：

```
buildscript {
    dependencies {
        classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'
    }
}
```

在Module的 `buile.gradle` 中：

```
apply plugin: 'com.android.application'
apply plugin: 'com.neenbedankt.android-apt'
dependencies {
    apt project(':apt-processor')
}
```

**使用**

热门故事

凤凰男的醒悟：花了半辈子孝顺父母，换来妻离子散

扎心虐文：不是所有的女主最后都会选择原谅

她闺蜜的一条朋友圈，结束了我和老公5年的婚姻

被戳穿的爱情是骗局，没被戳穿的变成了信仰

推荐阅读

APT入门
阅读 483

Android APT技术学习
阅读 119

Protobuf在Android中的基本使用
阅读 595

Android Gradle 实战之自动生成DeepLink配置信息
阅读 319

1.Android架构 retrofit运行时注解（POST详解）+反射实战demo 2分...
阅读 409

# 【Android】APT

带心情去旅行 关注

```
    @BindView(R.id.tv)
    TextView mTextView;
    @BindView(R.id.btn)
    Button mButton;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        BindViewTools.bind(this);
        mTextView.setText("bind TextView success");
        mButton.setText("bind Button success");
    }
}
```

运行的结果想必大家都知道了，不够为了证明这个 `BindView` 的功能完成了，我还是把图贴出来

结果

## 生成的代码

上面的功能一直在完成一件事情，那就是生成Java代码，那么生成的代码在哪？
在app/build/generated/source/apt中可以找到生成的Java文件

目录

对应的代码（之前已经贴过了）：



智能井盖

# 【Android】APT

带心情去旅行 关注

```
        host.mButton = (android.widget.Button) (((android.app.Activity) host).findViewById(213
        host.mTextView = (android.widget.TextView) (((android.app.Activity) host).findViewById
    }
}
```

## 通过javapoet生成代码

上面在 `ClassCreatorProxy` 中，通过 `StringBuilder` 来生成对应的Java代码。这种做法是比较麻烦的，还有一种更优雅的方式，那就是javapoet。

先添加依赖

```
dependencies {
    implementation 'com.squareup:javapoet:1.10.0'
}
```

然后在 `ClassCreatorProxy` 中

```
public class ClassCreatorProxy {
    //省略部分代码...

    /**
     * 创建Java代码
     * @return
     */
    public TypeSpec generateJavaCode2() {
        TypeSpec bindingClass = TypeSpec.classBuilder(mBindingClassName)
                .addModifiers(Modifier.PUBLIC)
                .addMethod(generateMethods2())
                .build();
        return bindingClass;

    }

    /**
     * 加入Method
     */
    private MethodSpec generateMethods2() {
        ClassName host = ClassName.bestGuess(mTypeElement.getQualifiedName().toString());
        MethodSpec.Builder methodBuilder = MethodSpec.methodBuilder("bind")
                .addModifiers(Modifier.PUBLIC)
                .returns(void.class)
                .addParameter(host, "host");

        for (int id : mVariableElementMap.keySet()) {
            VariableElement element = mVariableElementMap.get(id);
            String name = element.getSimpleName().toString();
            String type = element.asType().toString();
            methodBuilder.addCode("host." + name + " = " + "(" + type + ")(((android.app.Activ
        }
        return methodBuilder.build();
    }


    public String getPackageName() {
        return mPackageName;
    }
}

```

最后在 `BindViewProcessor` 中

```
    @Override
    public boolean process(Set<? extends TypeElement> set, RoundEnvironment roundEnvironment)
        //省略部分代码...
        //通过javapoet生成
```



热门故事

凤凰男的醒悟：花了半辈子孝顺父母，换来妻离子散

扎心虐文：不是所有的女主最后都会选择原谅

她闺蜜的一条朋友圈，结束了我和老公5年的婚姻

被戳穿的爱情是骗局，没被戳穿的变成了信仰

推荐阅读

APT入门
阅读 483

Android APT技术学习
阅读 119

Protobuf在Android中的基本使用
阅读 595

Android Gradle 实战之自动生成DeepLink配置信息
阅读 319

1.Android架构 retrofit运行时注解（POST详解）+反射实战demo 2分...
阅读 409

# 【Android】APT

带心情去旅行 关注

```
                javaFile.writeTo(processingEnv.getFiler());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        mMessager.printMessage(Diagnostic.Kind.NOTE, "process finish ...");
        return true;
    }
```

相比用 `StringBuilder` 拼Java代码，明显简洁和很多。最后生成的代码跟之前是一样的，就不贴出来了。

javapoet详细用法

## Tips

1、如果是 `ElementType.METHOD` 类型的注解，解析 `Element` 时使用 `ExecutableElement` ，而不是 `Symbol.MethodSymbol` ，否则编译运行的时候没问题，打包的时候会报错。别问我时为什么知道的...

2、gradle升级到3.4.0以后，AutoService要这么用

```
implementation 'com.google.auto.service:auto-service:1.0-rc2'
annotationProcessor  'com.google.auto.service:auto-service:1.0-rc2'
```

## 源码

GitHub

## 参考

编译期注解之APT
详细介绍编译时注解的使用方法
Android 编译时注解-提升
Android APT及基于APT的简单应用
Android 打造编译时注解解析框架 这只是一个开始
你必须知道的APT、annotationProcessor、android-apt、Provided、自定义注解

> 以上有错误之处，感谢指出

157人点赞 >

Android开发的点滴

带心情去旅行 努力做一个喜欢钻研的人
总资产645 共写了4.4W字 获得3,996个赞 共2,660个粉丝
关注

热门故事

凤凰男的醒悟：花了半辈子孝顺父母，换来妻离子散

扎心虐文：不是所有的女主最后都会选择原谅

她闺蜜的一条朋友圈，结束了我和老公5年的婚姻

被戳穿的爱情是骗局，没被戳穿的变成了信仰

推荐阅读

APT入门
阅读 483

Android APT技术学习
阅读 119

Protobuf在Android中的基本使用
阅读 595

Android Gradle 实战之自动生成DeepLink配置信息
阅读 319

1.Android架构 retrofit运行时注解（POST详解）+反射实战demo 2分...
阅读 409

智能井盖

# 【Android】APT

带心情去旅行 关注


## 被以下专题收入，发现更多相似内容

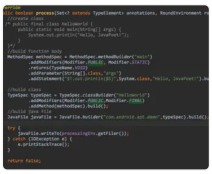
Android　Android进阶　Android笔记　值得点赞

认识虚拟机和多进程　Android...　高级汇总　展开更多

## 推荐阅读

更多精彩内容 >

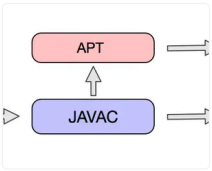
### Android APT（编译时生成代码）



摘要 APT(Annotation Processing Tool)是一种处理注解的工具,它对源代码文件进行检测找...

MarvinGuo 阅读 1,946 评论 0 赞 51

### Android之APT(Annotation Processing Tools)编译时创建Vi...




前言： 在上篇文章中讲解了通过IOS(依赖注入)的方式来为View创建对象并设置事件监听 从而简化我们的代码 方便...

明朗__ 阅读 1,073 评论 0 赞 12

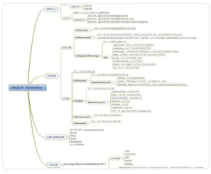
**为什么现在找工作大部分都是劳务派遣工?**


### Android APT（编译时代码生成）最佳实践

越来越多第三方库使用apt技术，如DBflow、Dagger2、ButterKnife、ActivityRoute...

ImWiki 阅读 635 评论 0 赞 2

### Java:Annotation(注解)--原理到案例



本文章涉及代码已放到github上annotation-study 1.Annotation为何而来 What：A...

zlcook 阅读 24,749 评论 14 赞 112

### 《留言》

情书并不是一张重要的纸，而是附着了无可替代的东西。 悄悄的翻开一页，看到的是意想不到的； 下一场雨目的不在于乌云，...

出雲 阅读 117 评论 0 赞 0

### SurfaceView的背景设置相关问题

最近公司产品要加一个视频的功能，所以搞的surfaceView方面的代码比较多，昨天遇到一个问题，我把视频

写下你的评论...　评论45　赞157

## 热门故事

凤凰男的醒悟：花了半辈子孝顺父母，换来妻离子散

扎心虐文：不是所有的女主最后都会选择原谅

她闺蜜的一条朋友圈，结束了我和老公5年的婚姻

被戳穿的爱情是骗局，没被戳穿的变成了信仰

## 推荐阅读

APT入门
阅读 483

Android APT技术学习
阅读 119

Protobuf在Android中的基本使用
阅读 595

Android Gradle 实战之自动生成DeepLink配置信息
阅读 319

1.Android架构 retrofit运行时注解（POST详解）+反射实战demo 2分...
阅读 409

智能井盖

# 【Android】APT

带心情去旅行 关注

自动生成签名


## 支付宝下一代功能

2015 年 8月，随着支付宝 9.0 版本的发布，支付宝开始重新梳理了定位。聚焦于生活服务平台和开放性平台，为用...

圆枢 阅读 128 评论 0 赞 0

## 微雨的黄昏

锄着草 培着花 就着酒 在微雨的黄昏 或安静的子夜 她等待着 她在等待爱情 等待她爱了一生的那个人 她的等待合着兰...

梦双眸 阅读 106 评论 1 赞 7

## 周立波也“出事”了，你怎么看？

在这个社会，对普罗大众而言，贫穷不是最大的敌人，最大的敌人是贫富不均。 当中国还有几千万贫困人口在温饱线上挣扎，当...

冰为溪水 阅读 479 评论 0 赞 0

### 热门故事

凤凰男的醒悟：花了半辈子孝顺父母，换来妻离子散

扎心虐文：不是所有的女主最后都会选择原谅

她闺蜜的一条朋友圈，结束了我和老公5年的婚姻

被戳穿的爱情是骗局，没被戳穿的变成了信仰

### 推荐阅读

APT入门
阅读 483

Android APT技术学习
阅读 119

Protobuf在Android中的基本使用
阅读 595

Android Gradle 实战之自动生成DeepLink配置信息
阅读 319

1.Android架构 retrofit运行时注解（POST详解）+反射实战demo 2分...
阅读 409

智能井盖

写下你的评论... 评论45 赞157