# Android迭代开发连载系列高级课

**讲师**：Anson、Mr.Hao

**适合人群：**

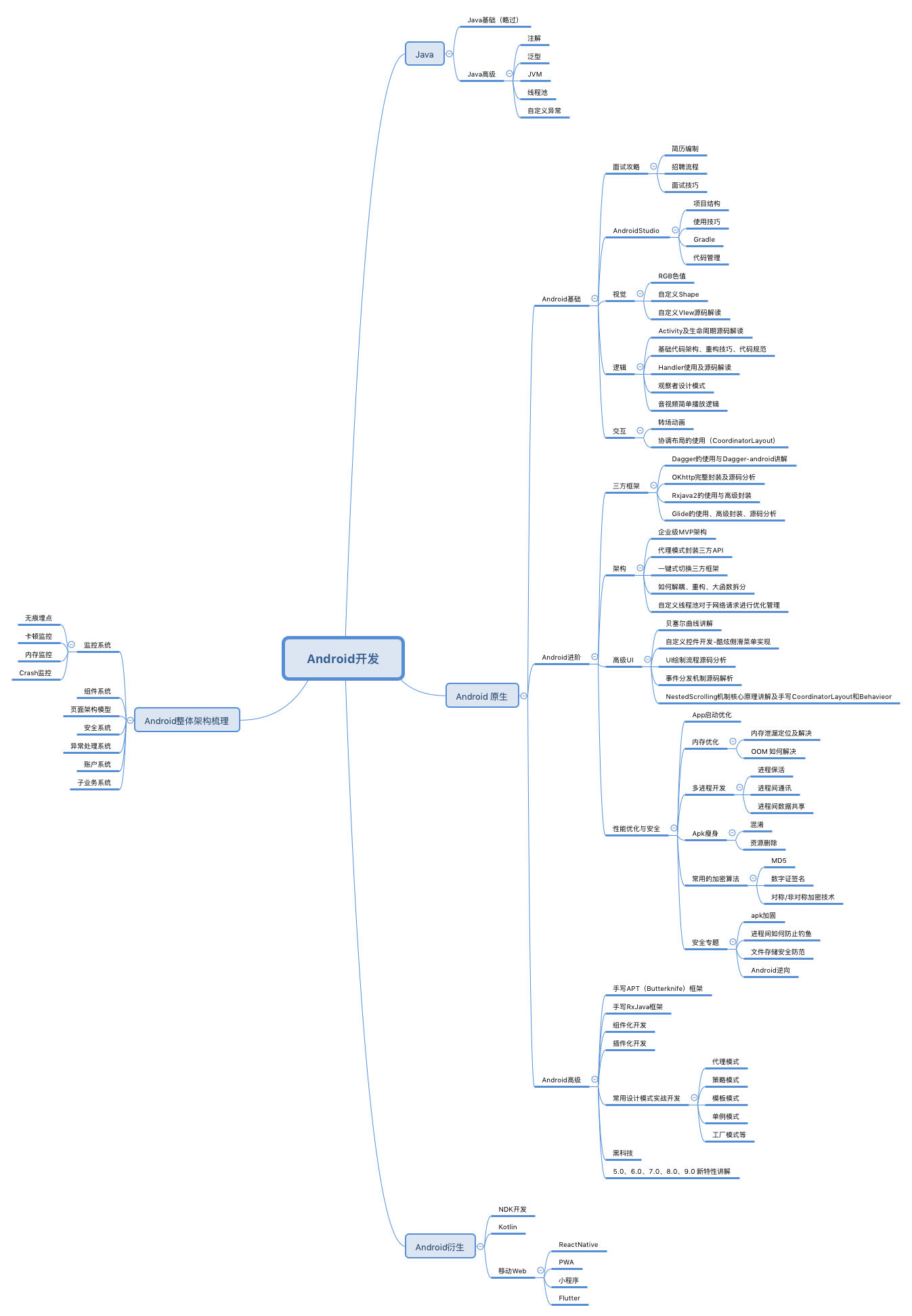
一定要有Java基础的初、中、高级Android开发工程师，如果是外行请到隔壁先把Java学会。

**课程情况：**

* **背景**：目前Android开发已经进入平缓阶段，突击培训的兄弟姐妹很难找到满意的工作，其次一些学习能力较弱的初、中开发找不到学习的群体和资料，所以我们想打造一套廉价的干货社区，为移动互联网献出自己的微薄之力。
* **内容**：从0搭建一款App，我们会根据企业的迭代内容来安排课程，不会按照固有的知识点进行讲解。
* **特色**：该套课程是连载的，也就是我们在Android开发岗位上一天，这套课程就会永久更新下去，后期会根据学员要求及新技术展开讲解。
* **售后**：我们会开放一个微信公众号，github代码仓库，qq群等，供大家学习和讨论，听取大家的意见，让我们在移动互联网的时代共同进步。

**课程大纲：**

前言：这不仅是一套开发编程学习课，更是一个Android开发小白成长到中级、高级、资深开发的人生故事，全部课程以小明第三人称为故事线，从找工作到刚进入到公司领取电脑、安装软件、从0开发一款App来阐述Android开发知识点，让我们一起见证小明的成长及开发成果。



# part1：小明的面试策略

## 简历的编制注意事项

1. 知识点写自己熟练掌握的。
2. 项目之间尽量体现差异化。
3. 简历要突出自己的能力和亮点。

## 招聘平台

猎聘网：相对招聘流程较慢、反馈慢，但精准度比较高。

Boss直聘：反馈快，时时性比较高。

拉钩网：这个可以作为辅助的工具，因为一天只能投递10次。

其他渠道：朋友内推、github、个人技术博客等。

## 如何选择公司

世界顶级公司（“FLAG”公司（Facebook, LinkedIn, Apple/Amazon, Google））：能去就去。

国内一流公司（BAT）：对于没渡过金的，能去就去。

二流老公司 (网易、新浪、携程、爱奇艺)

二流创业公司（滴滴、美团、头条）：进不了一流，可以优先考虑。看哪个事业部更适合自己。

其他公司：分两种，一种是非常有前景的，资金雄厚的完全可以去，另一种是维持现金流和试水的公司，尽量不要去。

**为什么选择公司比较重要？**

主要是影响自己人脉的积累和职业生涯的发展。

1. 周围的同事：职场上 同事是最好的老师，可以直接学到很多东西，其次品行好的同事同时在生活上会成为知心的朋友。
2. 管理流程： 正规公司的管理流程能让你了解一些管理经验
3. 领导水平： 好的领导会让你工作有动力，不会因为一些琐事影响心情。
4. 公司制度： 加薪、升职等会有公平的机会，五险一金等。
5. 知名度： 知名度越高对自己跳槽会更有帮助。

## 面试技巧

技术实力 + 情商 + “反面”

1. 不要紧张
2. 面试是需要提前做好准备的
3. 如实回答
4. 态度良好
5. 遇到自己不知道的，尽量不要说我不知道 我不会，用委婉的语气回答
6. 不要说太多废话
7. 适当反问，试探对方技术实力，了解对方和公司

群号：774288439

就这样，小明通过自己的努力进入一家创业型公司。

# part2：小明入职的第一天

## 公司组织架构

只列出跟小明工作相关的主要部门。

* 产品交互部门：产出需求、产品要具备什么功能
* 设计部门：根据产品需求设计界面
* 研发部：根据需求和界面码代码，开发功能
* 测试质量保障部：对研发人员开发功能进行测试，提出bug
* 项目管理部：保障这个需求迭代能够按期上线，管理上述部门的工时

## 环境搭建

通常公司会发Windows系统的电脑，但也不排除直接发Mac，所以本篇两个操作系统的环境搭建都跟大家介绍。

常用工具如下：

Windows：AndroidStudio（开发工具）、Git、Svn（代码管理工具），SourceTree（代码分支操作工具）、MindNode（思维导图）、Fiddler（抓包工具）、StarUML（类图、时序图等）。

Mac：AndroidStudio、Android文件传输（用于真机调试）、Charles（抓包工具）、其他同上。

准备工作：

* AndroidStudio下载：官网下载（http://www.androiddevtools.cn），国内下载（<http://www.android-studio.org>）；
* 环境变量配置：

Windows：JDK下载(http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html)，配置传送门（https://jingyan.baidu.com/article/c85b7a6414f2ee003bac95d5.html）。

Mac：如果不配置环境变量也是可以使用，只是命令行等不能用，配置传送门（https://www.jianshu.com/p/d52a8aaf03ba）。

* 项目配置：包括字体、sdk配置等一些信息可看下这篇文章（<https://www.cnblogs.com/smyhvae/p/4390905.html>）。

## 部门会议

参会人员：CTO、设计师、产品经理、PM（项目管理）、测试、后端开发、IOS、小明。

会议主题：根据现有业务开发一款安卓应用。

项目阶段：立项阶段。

会议内容：产品出需求、后台做基础架构、测试做相关流程规范、PM紧盯产品进度，小明的主要工作就是新建项目，开发基础框架。

其他：CTO说最近还会有一位Android开发人员入职，来和小明共同开发。

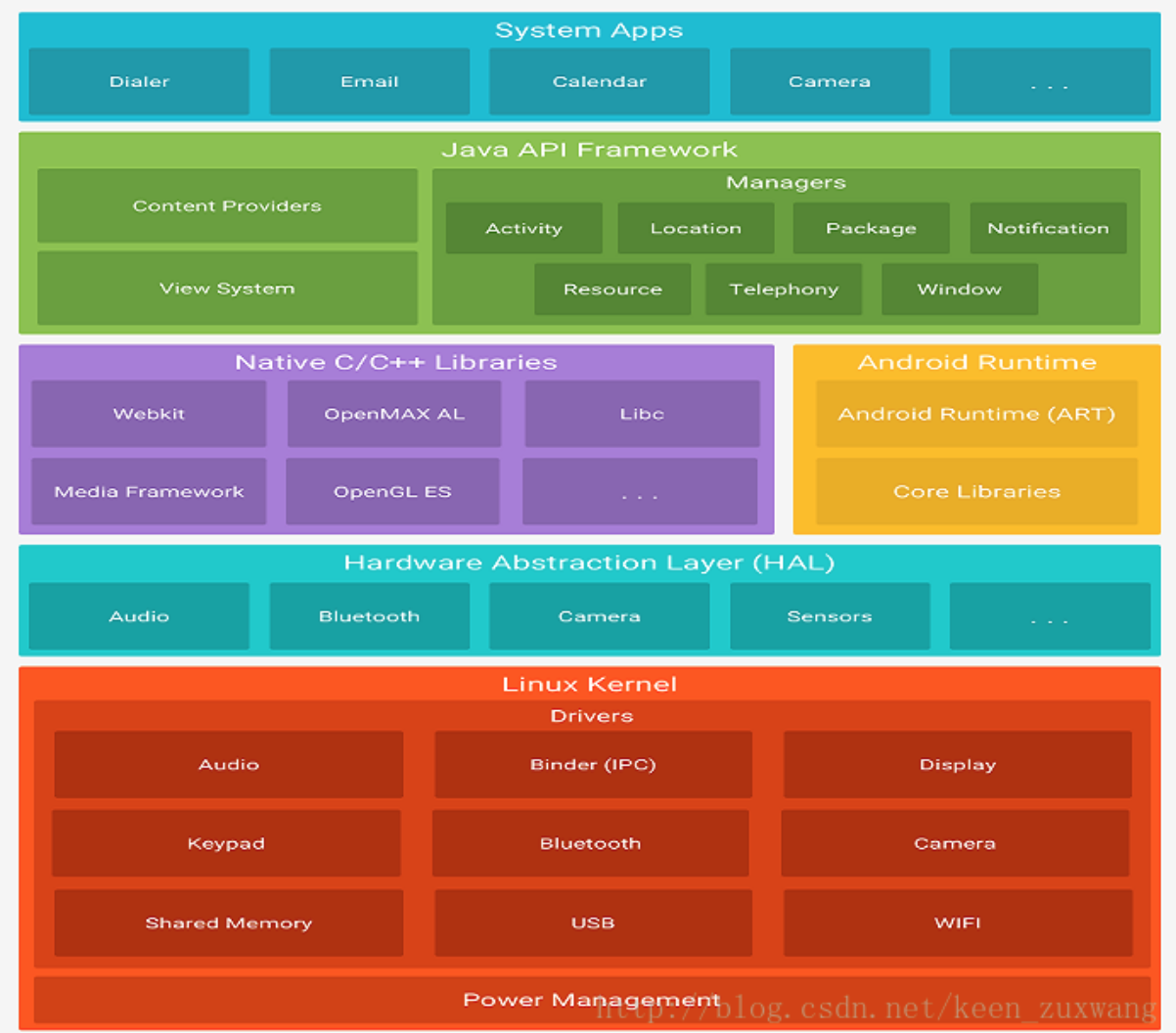
群号：774288439

# part3：项目搭建

## AndroidStudio的基本使用

1. Start a new Android Studio project
2. Application name、Company domain
3. Minimum SDK
4. MainActivity
5. 运行项目（模拟器、真机）
6. 那么问题来了？ 当前的这个界面是怎么样显示到屏幕上的？

***系统架构：*** 从早期的4层架构演变成现在的5层架构，主要是因为Linux遵循的GPL协议，底层驱动并不完全开源，基于此谷歌新增了一个硬件抽象层（HAL）。



***GUI：***展现在用户面前的界面是一个GUI（图形用户界面）系统，Android的GUI系统是基于OpenGL／EGL来实现的，OpenGL ES (OpenGL for Embedded Systems) 是 OpenGL 三维图形 API 的子集，针对手机、PDA和游戏主机等嵌入式设备而设计。

***谈界面显示必不可少的三个角色：***SurfaceFlinger（呈现视觉）、WMS（视觉管理）、ViewRoot（具体的界面效果），ViewRoot来管理具体的Layout，然后通过IWindowSession来和WMS进行通信。

***源码来看：***

Activity-mWindow 、Window – mWindowManager、WindowManagerImpl – WindowManagerGlobal 、WindowManagerGlobal - IWindowSession

## AndroidStudio的目录结构及作用

1. 左上角的视图结构：Project、Android
2. project(工作空间) 、module（项目）、gradle（编译构建工具）。

Gradle是一个基于Apache Ant和Apache Maven概念的项目自动化构建工具。它使用一种基于Groovy的特定领域语言(DSL)来声明项目设置，抛弃了基于XML的各种繁琐配置。

3、AndroidStudio快捷键：https://blog.csdn.net/xuaho0907/article/details/72026869

## 代码仓库

本课程：GitHub

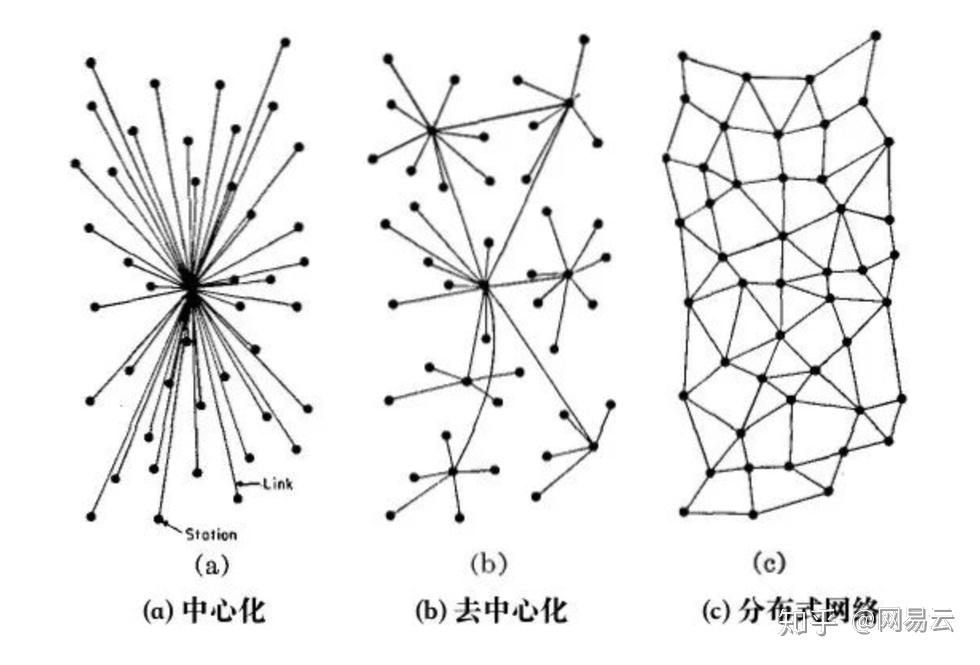
1. 代码托管平台 GitHub ，注册账号

神奇的网站，Google公司账号 Google Samples。

1. 下载git（<https://git-scm.com/downloads>）
2. git关联到Android Studio
3. github关联到Android Studio
4. 提交项目到github,建立仓库
5. 下载代码，上传代码
6. git的一些原理 ： commit push

Git是一个开源的分布式版本控制系统，可以有效、高速的处理从很小到非常大的项目版本管理。

什么是分布式？



分布式与去中心化的核心区别就是：分布式你可以知道下一个做决策的节点是谁，而去中心化，你不知道下一个做决策的节点是谁

公司：GitLab、Svn

群号：774288439

# part4：仿QQ高级特效 Splash界面

***迭代一：做一个短视频的开机画面。***

## 效果展示，需求分析：

1. 播放视频文件
2. 连续播放
3. 读秒倒计时
4. 无状态栏

## 基础原理

***编码解码：***其实就是视频压缩成文件及还原成帧图像，h264。

***封装格式：***mp4、avi等。

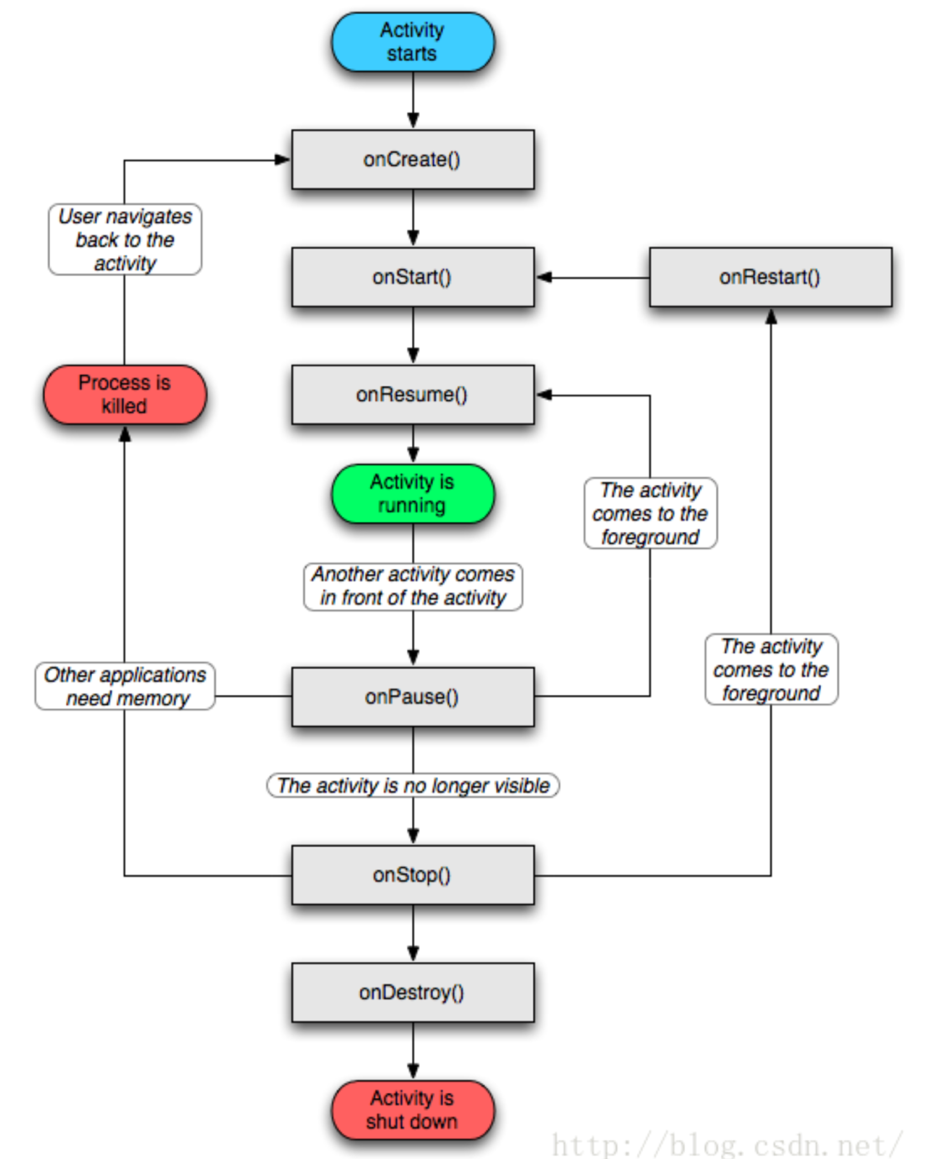
***硬解与软解：***cpu、gpu。

***关键帧：***视频压缩中比较重要的图像帧数据。

## 项目知识点

***Raw文件夹 ：***一般较大文件都存放在 Raw 和 assets 文件夹。

***Activity生命周期：***（<https://www.jianshu.com/p/fb44584daee3>）



***setContentView原理*** ：把自定义Layout加入到id 为 content 的ViewGroup中。

***findViewById：***根据xml id值来映射成view的对象。

***VideoView ：***对MediaPlayer的一层封装，只支持3gp和mp4格式的播放。

清单文件：权限 四大组件，PMS（Package Manager Service）安全检查。

状态栏、标题隐藏：修改Theme

***videoview的全屏问题：***videoview会根据视频文件的大小来改变自身的大小，所以要自定义view。

Android中View和ViewGroup是组合设计模式，23种设计模式之一。

***自定义view的几种方式：***

1、继承自View完全自定义

对于继承View类的自定义控件来说，核心的步骤分别为尺寸测量与绘制，对应的函数为onMeasure、onDraw，这是因为View的子空间是视图树的叶子节点，不需要布局。

2、继承现有控件实现特定效果

3、继承ViewGroup实现布局类

这种方式主要用于自定义子视图的排列方式时，比如下拉刷新、上拉加载等。ViewGroup为什么具有容器的作用？因为实现了两个接口 ViewParent和ViewManager，其中后者定义了addView及removeView等对子视图操作的方法,重点实现onLayout方法，这个方法主要是用于子视图的排版。

***EXACTLY模式***

该模式为精确模式。什么时候用到的是这种模式呢？当我们将layout\_width or layout\_height 设置为具体数值的时候，如100dp 或者是match\_parent 的时候。这时系统就是调用的精确模式

***AT\_MOST***

该模式为最大值模式。当我们将layout\_width or layout\_height设置为wrap\_content 的时候，系统调用的就是该模式。此时控件大小是随着子控件或者内容的变化而变化的，只要不超出父控件允许的最大范围即可

***UNSPECIFIED***

这个模式一般只有在我们自定义控件的时候才会用到，它代表没有具体的测量模式，view自己想多大就多大。

自定义计时器：用Handler的消息机制做数据的减法操作。

***TextView ：***Android系统提供处理文本的控件。

***自定义Shape ：***不自定义可以使用图片，自定义shape用xml代码来实现。

***颜色色值：***#ffffffff由#加八位数字或字母组成，前两个ff为透明度（十六进制），后面六位ffffff为颜色代码，采用RGB配色（十六进制），

***透明度参照表；***

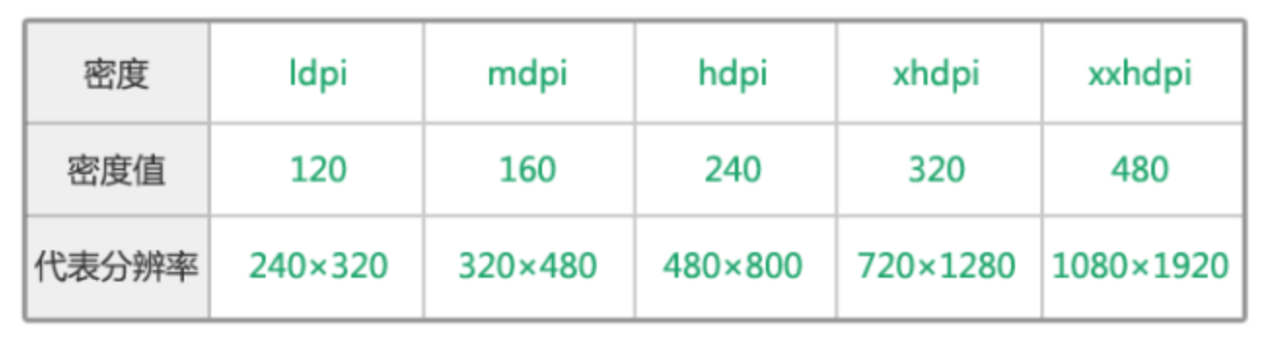
00%=FF（不透明） 5%=F2 10%=E5 15%=D8 20%=CC 25%=BF 30%=B2 35%=A5 40%=99 45%=8c 50%=7F

55%=72 60%=66 65%=59 70%=4c 75%=3F 80%=33 85%=21 90%=19 95%=0c 100%=00（全透明）

***解决硬编码：***把代码里面的中文字提出到String.xml文件，一些常量可单独提出到常量类里面，主要为了界面的整洁、规范、及维护。

***dp px sp 之间的概念：***

px：像素单位，dp：安卓距离单位，sp：安卓字体大小单位



当屏幕为mdpi时，1dp = 1px，换算单位为 dp = （dpi/160）px。

1dp永远都等于1sp。

***Bundle 的概念：***Bundle类型的数据与Map类型的数据相似，都是以key-value的形式存储数据的。

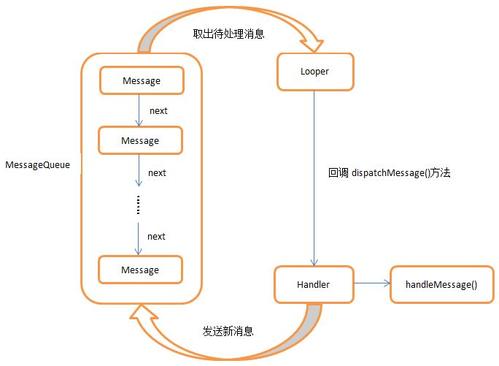
***界面跳转：intent-filter 和 Intent***

Intent中文意思指”意图”，通常Intent的隐式启动和intent-filter连用，显示启动不需要intent-filter。

IOC原理：<https://www.jianshu.com/p/f96391af0357>

观察者设计模式：观察者与被观察者。

Handler原理图：



死循环不会卡死吗？其实不然，这里就涉及到Linux pipe/epoll机制，简单说就是在主线程的MessageQueue没有消息时，便阻塞在loop的queue.next()中的nativePollOnce()方法里，此时主线程会释放CPU资源进入休眠状态，直到下个消息到达或者有事务发生，通过往pipe管道写端写入数据来唤醒主线程工作。

## 代码、分支规范

详见项目中的README.MD。

# Part5：基础项目组件集成

今天又入职了一位新同事，多人开发项目重要的是协作，那么协作包括代码、分支规范，及三方技术的选型，三方技术要考虑到每位同事的接受能力。

***前言：开源，如何采用三方库？***

1. 用的人数多
2. 大厂出品
3. 有作者还在维护
4. 对原理比较熟悉
5. 学习成本较低，易定位问题

## 项目架构

整体项目采用Dagger+MVP+OKhttp+Task，M层和P层进行双向抽象接口绑定，绑定的具体的类由Dagger进行实现，所有的网络请求用Okhttp进行，每个网络请求封装进入Task，再把相应的Task放入线程池进行维护，每个Task以Activity进行分组并命名，当Activity销毁时，把相应组的Task取消掉，这样做的好处是防止内存泄漏及偶然性的空指针。

第一步：集成Dagger、Okhttp、ButterKnife。

第二步：创建Activity、Presenter的Base类，并进行生命周期的绑定。

第三步：对之前的SplashActivity进行MVP改造。

第四步：封装Okhttp、Task、线程池。

第五步：进行整体的测试。

## MVP

1. 什么是MVP

其实我们日常开发中的Activity，Fragment和XML界面就相当于是一个 MVC 的架构模式，Activity中不仅要处理各种 UI 操作还要请求数据以及解析。这样的后果就是Activity可能会有几千行的代码，不便于维护，基于此现象及解耦的原则，才使用MVP的架构模式进行分离代码，其次MVP便于后期的代码重用，比如相同的业务逻辑，不同的ui界面等，开发MVP框架最重要的注意事项是内存泄漏。

2、内存泄漏的官方定义

内存泄漏（Memory Leak）是指程序中己动态分配的堆内存由于某种原因程序未释放或无法释放，造成系统内存的浪费，导致程序运行速度减慢甚至系统崩溃等严重后果。

3、关于内存泄漏

后面会单独出一个章节来讲解Android系统的性能调优，包括如何定位长期的内存泄漏、短期的内存泄漏，造成内存泄漏的原因分析及如何解决内存泄漏及内存溢出。

4. Java的内存管理

Android系统是用Java语言编写的，则也是遵循的Java的内存回收机制，在Java中内存有堆和栈，堆中保存的是类的对象，栈是堆内存的引用，当打开一个Activity，并在手机上退出关闭，这个时候如果Activity的对象没有在堆内存中销毁，我们就判断此时有内存泄漏，当这样的情况越来越多，这个App运行的内存越来越大，造成的后果就是越来越卡，直至造成应用崩溃无响应的后果，所以在开发Android程序时，掌握这个点是非常重要的。

参考文章：https://blog.csdn.net/johnny901114/article/details/54783106

搭建一套MVP架构框架：

\*中介者设计模式

\*Base基类的设计

\*静态代理生命周期

\*P层的安全、内存友好设计

\*APT 在 MVP 框架中应用，提高车速

5.Java的四种引用

强引用（StrongReference）：只要某个对象有强引用与之关联，JVM必定不会回收这个对象，即使在内存不足的情况下，JVM宁愿抛出OutOfMemory错误也不会回收这种对象。

软引用（SoftReference）：对于软引用关联着的对象，只有在内存不足的时候JVM才会回收该对象。

弱引用(WeakReference) : 弱引用也是用来描述非必需对象的，当JVM进行垃圾回收时，无论内存是否充足，都会回收被弱引用关联的对象。

虚引用（PhantomReference）: 在任何时候都可能被垃圾回收器回收。

## Dagger

1. 什么是Dagger：依赖注入框架，比如省略了new xxxx()实例化的过程。
2. 如何集成 ：compile 、annotationProcessor（APT）后面Mr.Hao会带大家手写一套APT框架。
3. 如何学习Dagger：上手难度不太容易，建议前期不要过度探究Dagger的实现原理及为何要这样去写，先模仿使用再去整体把握细节。
4. 如何使用：参考项目

## ButterKnife

1. 什么是ButterKnife: 减少手写findviewbyid的操作，提高开发效率，可使用As一键生成代码，后期由Mr.Hao来带领大家手写一套这样的框架。
2. 如何集成：github上面有具体的集成方式
3. 如何使用：用as plugin插件一键生成。

## Okhttp

1. 什么是okhttp？

okhttp是一个第三方类库，用于android中请求网络。这是一个开源项目,是安卓端最火热的轻量级框架,由移动支付Square公司贡献(该公司还贡献了Picasso和LeakCanary)

1. 为什么要使用OKHttp

OKHttp是Android版Http客户端。非常高效，支持SPDY、连接池、GZI和HTTP缓存，支持SPDY，可以合并多个到同一个主机的请求。

OkHttp 处理了很多网络疑难杂症：会从很多常用的连接问题中自动恢复。如果您的服务器配置了多个IP地址，当第一个IP连接失败的时候，OkHttp会自动尝试下一个IP。

OkHttp还处理了代理服务器问题和SSL握手失败问题。

1. 如何使用？

当我们使用一个三方库时，最好就是减少对其API的依赖，所以我们要对原始的OKHttp做二次封装来实现我们的业务需求，如何封装就要对其原理及一些常用的API非常熟悉，下面我们会系统学习网络这一模块，然后封装成符合我们业务的SDK。

1. 网络请求构建、网络请求发送、网络请求解析

说起网络就要先了解什么是HTTP？

功能：本地持久化数据、定制化的解析器、定制化的构建器、可自由切换网络库

1. 与Task线程池进行绑定
2. 申请权限：

## Task与线程池

设计Task与线程池是为了更好的维护网络请求的开始、暂停、取消等操作。

比如下面两个场景：

1. 当用MVP设计代码时，网络请求一般都写在P层里，当打开一个页面，这时候页面退出，网络请求还在继续，不仅造成内存泄漏更有造成空指针的危险，这个时候用自定义的Task可以Activity的生命周期进行绑定，当退出界面可把这个网络任务取消掉。
2. 当点击一个按钮会发送一次网络请求，当快速点击多次，这时会造成两个一样的网络任务并行，使用自定义的Task，可以判断是否有同一类型的网络请求，根据业务的需要去设计不同的使用场景。

本项目只采取单一的Task处理策略，当存在相同的任务时，新建立的任务会取消掉。

如何使用：TaskHelper

## Gradle配置文件

## 设计模式

#### 中介者设计模式

4个角色

第一个角色：抽象中介者->MvpPresenter （定义接口）

第二个角色：具体中介者->具体功能模块（例如：LoginPresente等…）

第三个角色：抽象同事

M层：MvpModel

V层：MvpView

第四个角色：具体同事

M层：例如->LoginModel…

V层：例如->MainActivity…

#### 代理设计模式

角色一：目标接口

角色二：目标对象

角色三：代理对象

**ps：静态代理与动态代理的区别？ 主要的区别是代理类是否提前创建存在的。**

#### 责任链设计模式

1. 使多个对象可以对请求及结果进行处理，从而使请求和结果处理者不再耦合。
2. 将请求一直传递，一直到有人处理它为止。

## 注解

重要概念：元注解、@Retention、@Target。

元注解: 元注解是一种基本注解，但是它能够应用到其它的注解上面。

@Retention：注解的保留期

- RetentionPolicy.SOURCE 注解只在源码阶段保留，在编译器进行编译时它将被丢弃忽视。

- RetentionPolicy.CLASS 注解只被保留到编译进行的时候，它并不会被加载到 JVM 中。

- RetentionPolicy.RUNTIME 注解可以保留到程序运行的时候，它会被加载进入到 JVM 中，所以在程序运行时可以获取到它们。

@Target：作用域

ElementType.ANNOTATION\_TYPE 可以给一个注解进行注解

ElementType.CONSTRUCTOR 可以给构造方法进行注解

ElementType.FIELD 可以给属性进行注解

ElementType.LOCAL\_VARIABLE 可以给局部变量进行注解

ElementType.METHOD 可以给方法进行注解

ElementType.PACKAGE 可以给一个包进行注解

ElementType.PARAMETER 可以给一个方法内的参数进行注解

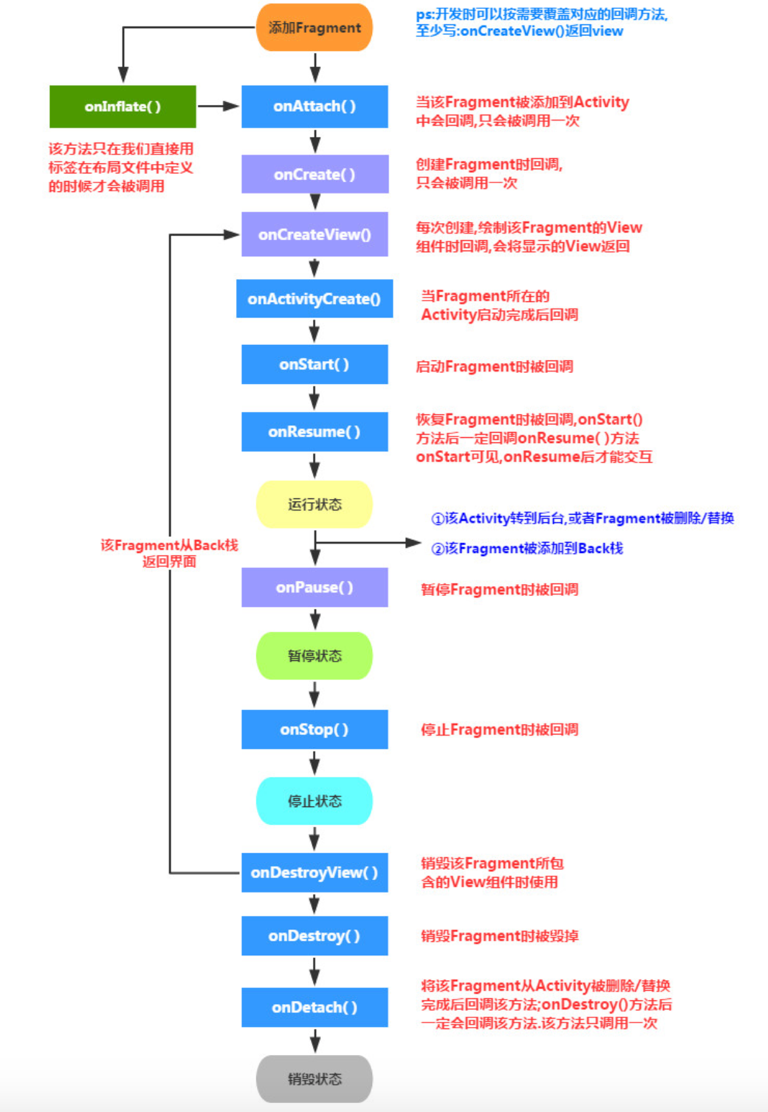
ElementType.TYPE 可以给一个类型进行注解，比如类、接口、枚举

# Part6：项目主页面动态切换效果

## 效果展示

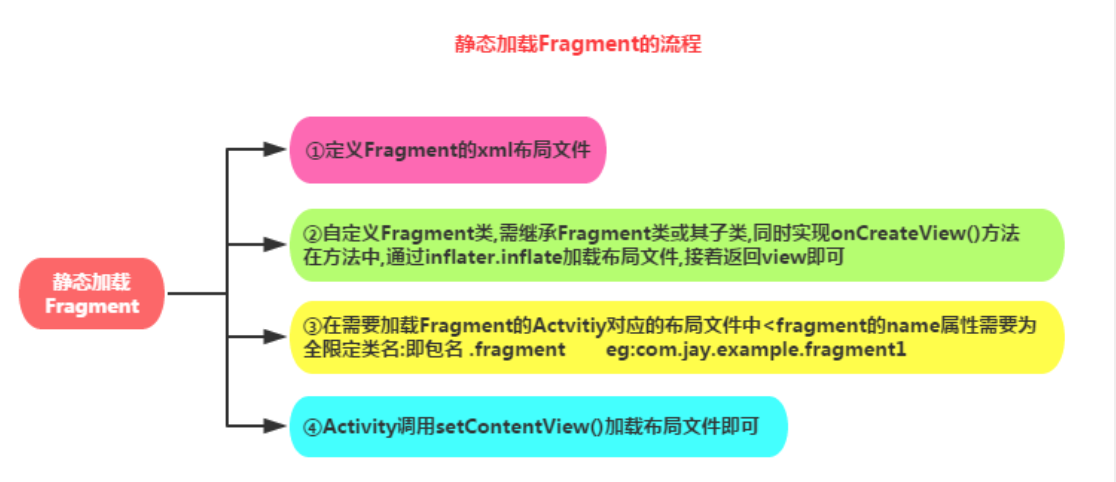
## 认识一下Fragment

Fragment是Android3.0后引入的一个新的API，他出现的初衷是为了适应大屏幕的平板电脑， 我们可以把他看成一个小型的Activity，又称Activity片段！使用Fragment 我们可以把屏幕划分成几块，然后进行分组，进行一个模块化的管理！从而可以更加方便的在 运行过程中动态地更新Activity的用户界面！另外Fragment并不能单独使用，他需要嵌套在Activity 中使用，尽管他拥有自己的生命周期，但是还是会受到宿主Activity的生命周期的影响，比如Activity 被destory销毁了，他也会跟着销毁！

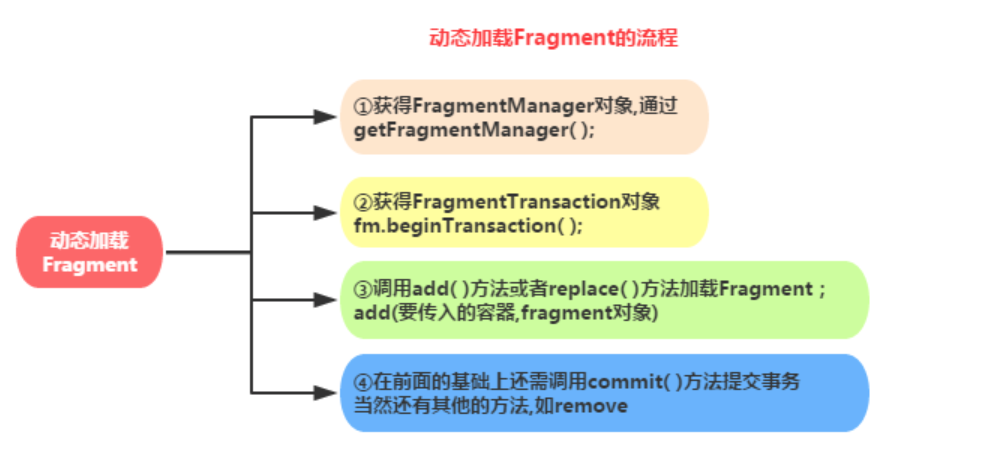


Fragment的两种加载方式

1. 静态加载



1. 动态加载



add hide show replace 中，replace 无法复用，hide和show会达到复用的效果

## 封装BaseFragment

参照之前的BaseActivity 并加上相应Fragment的生命周期。

## Design控件、RadioGroup

1. 什么是Design

全称是Android Material Design，是一套全新的控件库，里面包含AppBarLayout、CollapsingToolbarLayout、CoordinatorLayout，FloatingActionButton、NavigationView、Snackbar、TabLayout、TextInputLayout这八种控件。

2、RadioGroup 和 RadioButton常常组合起来使用，RadioButton是一种点选操作按钮，开发有时也会用TextView做替代，只是RadioButton更方便一些。

## 安卓中的动画

帧动画、补间动画、属性动画（<https://www.jianshu.com/p/420629118c10>）

## 版本控制（Git Flow）

<https://www.cnblogs.com/myqianlan/p/4195994.html>

### XML文件

选择器：selector 常用属性

（非触摸式获取焦点）state\_window\_focused 、state\_focused、state\_pressed、state\_selected

### 屏幕尺寸(drawble vs mipmap)



# Part7：协调布局CoordinatorLayout

## 协调交互原理讲解

CoordinatorLayout 是一个整体的ViewGroup，它的主要作用是协调各个子View之间的滑动交互，也就是Behavior，后期我们会详细讲一下怎么来自定义Behavior。

AppBarLayout：首先，AppBarLayout是一个LinearLayout，它的子View默认纵向排列， 可以通过一些参数控制子View的滑动行为。

AppBarLayout:对下面的子控件提供一些layout\_scrollFlags属性的配置

layout\_scrollFlags配置详细说明：

**全局滑动事件处理**

Scroll：跟着滑动的方向进行滑动，如果不配置会一直在屏幕中静态显示。

Snap：这个属性有个很明显回弹效果，当滑动距离过短时，会回弹到最近的上边缘或者下边缘。

**向下滑动事件处理**

EnterAlways:这个属性其实就是向下滚动时Scrolling View和Child View之间的滚动优先级问题，当向下滚动时，优先滑出Child View，当全部滑出时，再去处理Scrolling View的滑动事件。

EnterAlwaysCollapsed：这个属性是上个属性的附加值，这里涉及到Child View的高度和最小高度，向下滚动时，Child View先向下滚动最小高度值，然后Scrolling View开始滚动，到达边界时，Child View再向下滚动，直至显示完全。

**向上滑动事件处理**

ExitUntilCollapsed：这里也涉及到最小高度。发生向上滚动事件时，Child View向上滚动退出直至最小高度，然后Scrolling View开始滚动。也就是，Child View不会完全退出屏幕。

CollapsingToolbarLayout：实现Toolbar的折叠效果

contentScrim：折叠后的颜色

layout\_collapseMode：pin 滑动时会固定到屏幕上，不会折叠，parallax会发生折叠，并带有一定的视差特效效果，视差特效的比例由layout\_collapseParallaxMultiplier来控制。

Behavior: 处理交互的，其中最主要有两个知识点，依赖方和被依赖方。

目前用的系统的这个Behavior 被依赖的是 AppBarLayout，依赖的就是我们下面的整个布局。

## TabLayout+ViewPager+Fragment

ViewPager中的Adapter：(https://blog.csdn.net/rankun1/article/details/51842471)

主要涉及这几个类，PagerAdapter、FragmentStatePagerAdapter、FragmentPagerAdapter。

PagerAdapter：原始基类。

FragmentPagerAdapter：继承PagerAdapter，适用页数小的静态页面，所有Fragment会保存到内存当中，Fragment切换时只会走到 onDestoryView这个生命周期。

FragmentStatePagerAdapter：继承PagerAdapter，适用页数多的动态页面，Fragment切换时会走到onDetach这个生命周期。

## Recyclerview

几个重要的角色

### 1、Recycler

管理那些已经废弃了的或者从RecyclerView中分离的item view用于复用，主要用于缓存的设计。

一共四层缓存：

\* 1.一级缓存：mAttachedScrap

\* 2.二级缓存：mCacheViews

\* 3.三级缓存：mViewCacheExtension

\* 4.四级缓存：mRecyclerPool

### 2、LayoutManager

LayoutManager是一个抽象类，有3个子类：

LinearLayoutManager 线性布局管理器

GridLayoutManager 表格布局管理器

StaggeredGridLayoutManager 瀑布流布局管理器

### 3、Adapter

适配器设计模式，把相应数据Data展示到不同的 布局View 中。

### 4、高级用法

复杂条目、设置监听时机、缓存池的应用。

RecyclerView inflate 布局时 传入的三个参数 具体有什么关系？涉及系统源码

1. root 不为null attachToRoot 为 false，则子View会设置 root 的 LayoutParams。
2. root 不为null attachToRoot 为 ture，则root 会把 子view添加进来。
3. root 为 null RecycleView 会 默认给子View 生成一个默认的LayoutParams，所以会导致条目展示问题。

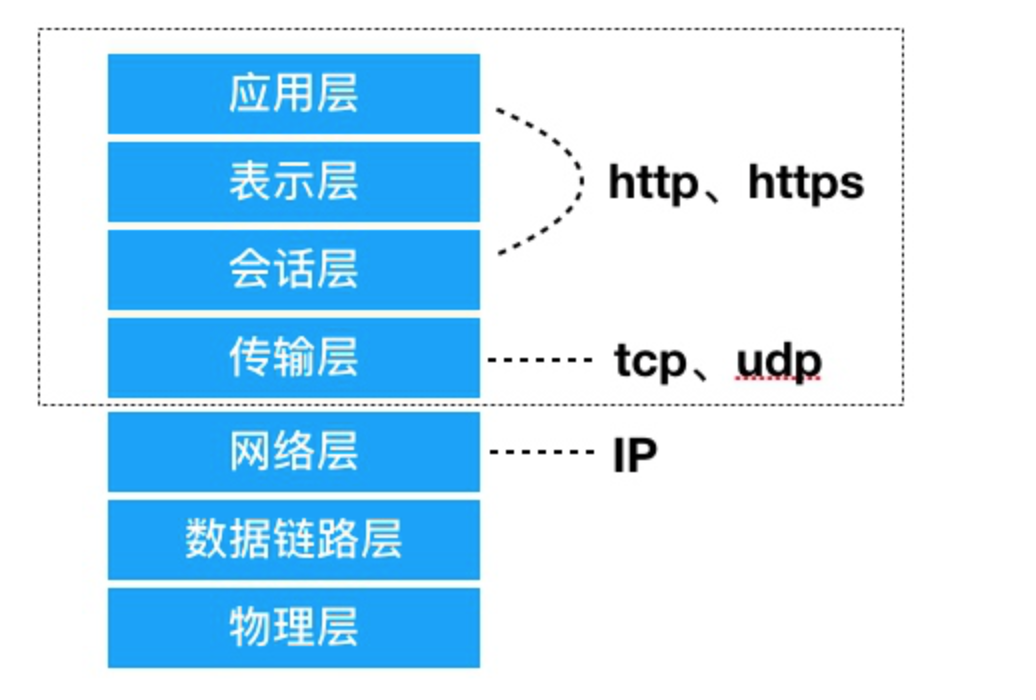
tryGetViewHolderForPositionByDeadline

## Adapter简易封装及原理

1. 使用的是RecyclerView中的Adapter
2. 自定义ViewHolder
3. SparseArray原理
4. CharSequence与String的关系 富文本

# Part8：网络

HTTP  
HTTP 是基于 TCP/IP 协议的应用层协议。它不涉及数据包（packet）传输，主要规定了客户端和服务器之间的通信格式，默认使用80端口。



TCP/IP <https://baike.baidu.com/item/TCP/IP协议>

### HTTP版本

1.0 ：HTTP/1.0 版的主要缺点是，每个TCP连接只能发送一个请求。发送数据完毕，连接就关闭，如果还要请求其他资源，就必须再新建一个连接。

TCP连接的新建成本很高，因为需要客户端和服务器三次握手，并且开始时发送速率较慢（slow start）。所以，HTTP 1.0版本的性能比较差。随着网页加载的外部资源越来越多，这个问题就愈发突出了。

为了解决这个问题，有些浏览器在请求时，用了一个非标准的Connection字段。Connection: keep-alive

1.1：1.1 版的最大变化，就是引入了持久连接（persistent connection），即TCP连接默认不关闭，可以被多个请求复用，不用声明Connection: keep-alive

虽然1.1版允许复用TCP连接，但是同一个TCP连接里面，所有的数据通信是按次序进行的。服务器只有处理完一个回应，才会进行下一个回应。要是前面的回应特别慢，后面就会有许多请求排队等着。这称为"队头堵塞"（Head-of-line blocking）。

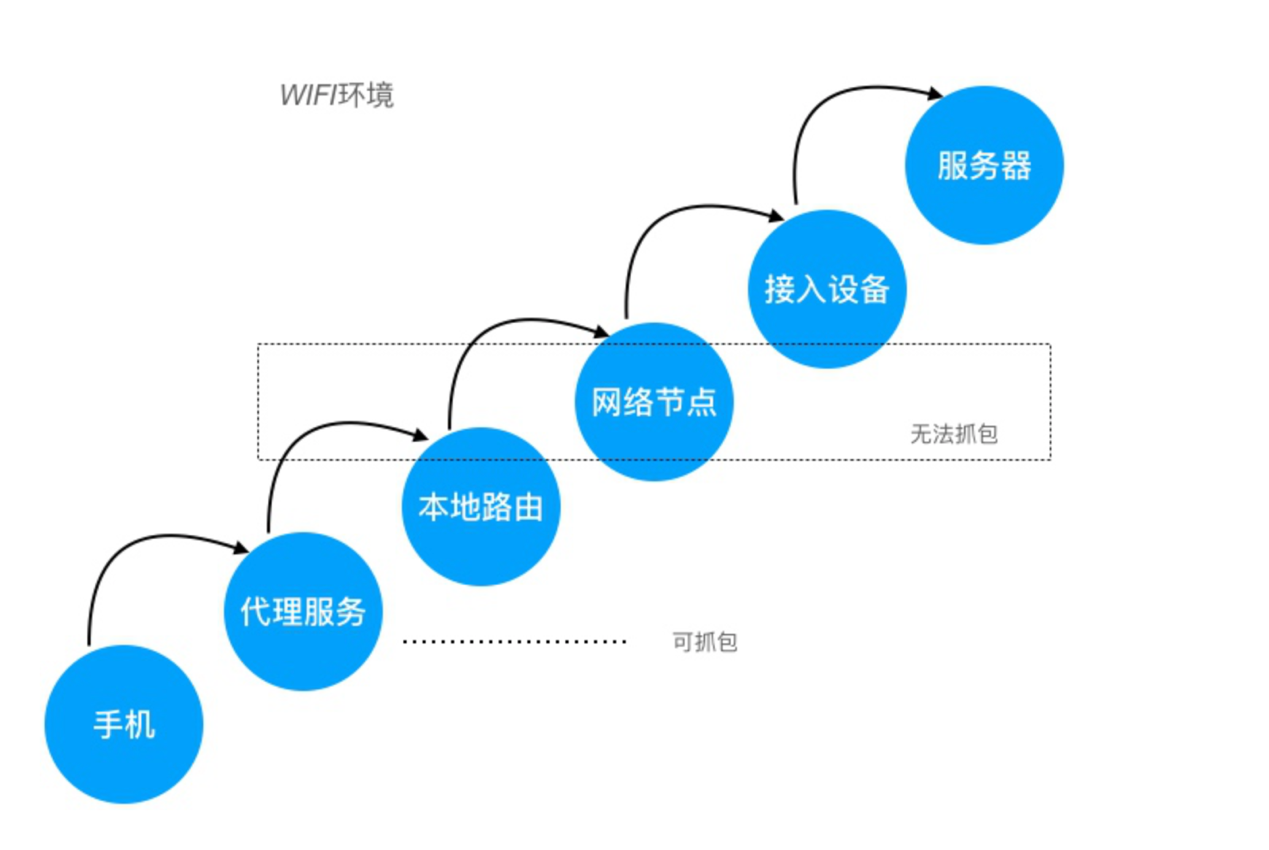
2.0：HTTP/2 复用TCP连接，在一个连接里，客户端和浏览器都可以同时发送多个请求或回应，而且不用按照顺序一一对应，这样就避免了"队头堵塞"。

举例来说，在一个TCP连接里面，服务器同时收到了A请求和B请求，于是先回应A请求，结果发现处理过程非常耗时，于是就发送A请求已经处理好的部分， 接着回应B请求，完成后，再发送A请求剩下的部分。

这样双向的、实时的通信，就叫做多工（Multiplexing）。

抓包工具的使用：Fiddler、Charles

抓包工具原理：



如何在Android 中使用网络

1. 工程项目开启网络权限

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

1. 选择网络框架

Xutils、Volley、OkHttp、Retrofit

1. 找一个网络接口
2. 什么是Json

https://baike.baidu.com/item/JSON/2462549?fr=aladdin

<http://www.bejson.com/json2javapojo/new/>

5、发送一个网络请求

集成OkHttp

### okhttp源码解析

OkHttpClient : 建议OkHttp的客户端，初始化OkHttp要使用的配置信息。

Request：请求信息封装

Call：可执行异步或者同步请求

Interceptor: OkHttp中的精华就是 拦截器，责任链设计模式。

RetryAndFollowUpInterceptor：负责错误重试及重定向。

BridgeInterceptor：负责组装请求及解析数据。

CacheInterceptor：负责读取缓存和更新缓存。

ConnectInterceptor：负责和服务器建立连接。

CallServerInterceptor：负责发送请求及接收数据。Okio 对接

### Get请求

从指定的资源请求数据。

GET 请求不应在处理敏感数据时使用

GET 请求有长度限制

例子：http://v.juhe.cn/joke/content/list.php?key=您申请的KEY&page=2&pagesize=10&sort=asc&time=1418745237

### AsyncTask



### 线程池

线程池顾名思义就是对线程进行管理，控制线程的数量及执行策略，从而减少性能消耗，使项目更好的稳定运行。

ThreadPoolExecutor 参数介绍

corePoolSize 核心线程数

maximumPoolSize 最大线程数

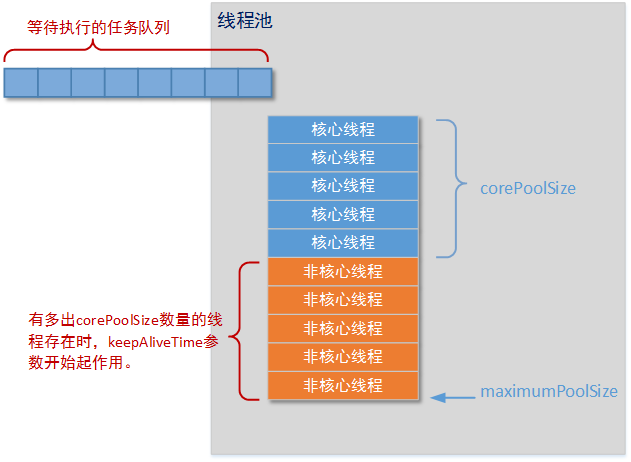
keepAliveTime 多余线程存活时间

TimeUnit 存活的单位

BlockingQueue 任务队列

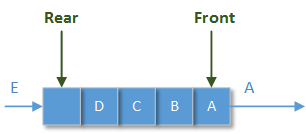
ThreadFactory 线程工厂

RejectedExecutionHandler 拒绝任务



#### 队列

队列：是一种特殊的线性结构，允许在线性结构的前端进行删除/读取操作；允许在线性结构的后端进行插入操作；这种线性结构具有“先进先出”的操作特点：



但是在实际应用中，队列中的元素有可能不是以“进入的顺序”为排序依据的。例如我们将要讲到的PriorityBlockingQueue队列。

● SynchronousQueue：

“是这样 一种阻塞队列，其中每个 put 必须等待一个 take，反之亦然。同步队列没有任何内部容量。翻译一下：这是一个内部没有任何容量的阻塞队列，任何一次插入操作的元素都要等待相对的删除/读取操作，否则进行插入操作的线程就要一直等待，反之亦然。

● ArrayBlockingQueue：

一个由数组支持的有界阻塞队列。此队列按 FIFO（先进先出）原则对元素进行排序。新元素插入到队列的尾部，队列获取操作则是从队列头部开始获得元素。这是一个典型的“有界缓存区”，固定大小的数组在其中保持生产者插入的元素和使用者提取的元素。一旦创建了这样的缓存区，就不能再增加其容量。试图向已满队列中放入元素会导致操作受阻塞；试图从空队列中提取元素将导致类似阻塞。

● LinkedBlockingQueue：

LinkedBlockingQueue是我们在ThreadPoolExecutor线程池中常用的等待队列。它可以指定容量也可以不指定容量。由于它具有“无限容量”的特性，所以我还是将它归入了无限队列的范畴（实际上任何无限容量的队列/栈都是有容量的，这个容量就是Integer.MAX\_VALUE）。

LinkedBlockingQueue的实现是基于链表结构，而不是类似ArrayBlockingQueue那样的数组。但实际使用过程中，不需要关心它的内部实现，如果指定了LinkedBlockingQueue的容量大小，那么它反映出来的使用特性就和ArrayBlockingQueue类似了。

● PriorityBlockingQueue

PriorityBlockingQueue是一个按照优先级进行内部元素排序的无限队列。存放在PriorityBlockingQueue中的元素必须实现Comparable接口，这样才能通过实现compareTo()方法进行排序。优先级最高的元素将始终排在队列的头部；PriorityBlockingQueue不会保证优先级一样的元素的排序，也不保证当前队列中除了优先级最高的元素以外的元素，随时处于正确排序的位置。

● LinkedTransferQueue

LinkedTransferQueue也是一个无限队列，它除了具有一般队列的操作特性外（先进先出），还具有一个阻塞特性：LinkedTransferQueue可以由一对生产者/消费者线程进行操作，当消费者将一个新的元素插入队列后，消费者线程将会一直等待，直到某一个消费者线程将这个元素取走，反之亦然。

#### 拒绝任务

当提交给线程池的某一个新任务无法直接被线程池中“核心线程”直接处理，又无法加入等待队列，也无法创建新的线程执行；又或者线程池已经调用shutdown()方法停止了工作；又或者线程池不是处于正常的工作状态；这时候ThreadPoolExecutor线程池会拒绝处理这个任务，触发创建ThreadPoolExecutor线程池时定义的RejectedExecutionHandler接口的实现

● CallerRunsPolicy：

这个拒绝处理器，将直接运行这个任务的run方法。但是，请注意并不是在ThreadPoolExecutor线程池中的线程中运行，而是直接调用这个任务实现的run方法。

● AbortPolicy：

这个处理器，在任务被拒绝后会创建一个RejectedExecutionException异常并抛出。这个处理过程也是ThreadPoolExecutor线程池默认的RejectedExecutionHandler实现。

● DiscardPolicy：

DiscardPolicy处理器，将会默默丢弃这个被拒绝的任务，不会抛出异常，也不会通过其他方式执行这个任务的任何一个方法，更不会出现任何的日志提示。

● DiscardOldestPolicy：

这个处理器很有意思。它会检查当前ThreadPoolExecutor线程池的等待队列。并调用队列的poll()方法，将当前处于等待队列列头的等待任务强行取出，然后再试图将当前被拒绝的任务提交到线程池执行

#### Executors

https://www.xuanyusong.com/archives/2439

CachedThreadPool（弹性缓存线程池），特点：用newCachedThreadPool()方法创建该线程池对象，创建之初里面一个线程都没有，当execute方法或submit方法向线程池提交任务时，会自动新建线程；如果线程池中有空余线程，则不会新建；这种线程池一般最多情况可以容纳几万个线程，里面的线程空余60s会被回收。

SingleThreadPool（单线程线程池），特点：池中只有一个线程，如果扔5个任务进来，那么有4个任务将排队；作用是保证任务的顺序执行。

FixedThreadPool，特点：固定池子中线程的个数。

ScheduledThreadpool（定时器线程池）

ForkJoinPool，JDK 1.7 引入，根据CPU 把任务分成小模块执行，执行后合并。

### 补充知识点

什么是泛型

泛型，即“参数化类型”。一提到参数，最熟悉的就是定义方法时有形参，然后调用此方法时传递实参。那么参数化类型怎么理解呢？顾名思义，就是将类型由原来的具体的类型参数化，类似于方法中的变量参数，此时类型也定义成参数形式（可以称之为类型形参），然后在使用/调用时传入具体的类型（类型实参）。

泛型的作用

比如大家常用的 ArrayList等，通过注入相应的参数，可承载不同的类和对象，当类型不匹配时会报错，某种程度上提高了编码的安全性。

泛型有哪些使用技巧

泛型接口、类，泛型函数，Class<T>，泛型数组，类型绑定，通配符等。

Java 关于泛型<T> 中字母的规范分类

E — Element，常用在java Collection里，如：List<E>,Iterator<E>,Set<E>

K,V — Key，Value，代表Map的键值对

N — Number，数字

T — Type，类型，如String，Integer等等

这仅仅是一个规范而已，传入其他也不会报错。

序列化与反序列化

<https://www.cnblogs.com/xdp-gacl/p/3777987.html>

HTTPS

<https://www.jb51.net/article/130874.htm>

<https://www.jianshu.com/p/4a6b731d5c60>

OKHTTP 缓存

<https://www.jianshu.com/p/647dfd41194e?utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommendation>

# part9：玩转自定义

### TabLayout

属性讲解

<https://www.jianshu.com/p/fde38f367019>

自定义控件合集

<https://github.com/opendigg/awesome-github-android-ui>

### ViewPager

<https://www.jianshu.com/p/e5abbda4a71c>

1. Fragment 嵌套问题
2. PagerAdapter 的使用
3. 预加载及懒加载
4. 动态更新
5. 内存友好设计

### 自定义Behavior

Material Design里面的CoordinatorLayout是一个非常强大的控件，它接管了child组件之间的交互。让你滑动交互使用更加方便简单，效果也更加强大，不需要像以前那样自己处理一坨什么乱七八槽的滑动,事件传递之类的处理了。

其实Behavior就是一个应用于View的观察者模式，一个View跟随着另一个View的变化而变化，或者说一个View监听另一个View。

观察者模式:

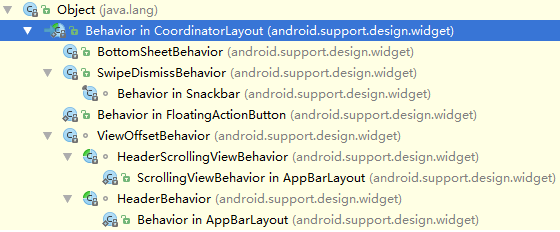
1.概述

有时被称作发布/订阅模式，观察者模式定义了一种一对多的依赖关系，让多个观察者对象同时监听某一个主题对象。这个主题对象在状态发生变化时，会通知所有观察者对象，使它们能够自动更新自己。

2. 解决的问题

　　将一个系统分割成一个一些类相互协作的类有一个不好的副作用，那就是需要维护相关对象间的一致性。我们不希望为了维持一致性而使各类紧密耦合，这样会给维护、扩展和重用都带来不便。观察者就是解决这类的耦合关系的。

在Behavior中，被观察的View 也就是事件源被称为denpendcy，而观察View，则被称为child



### Behavior中的几个重要方法

Behavior 是一个顶层抽象类，其他的一些具体行为的Behavior 都是继承自这个类。它提供了几个重要的方法：

layoutDependsOn

onDependentViewChanged

onStartNestedScroll

onNestedPreScroll

onNestedScroll

onStopNestedScroll

onNestedScrollAccepted

onNestedPreFling

onLayoutChild

### 自定义View

### 动画

1. 原生动画

①补间动画

平移:TranslateAnimation

旋转:RotateAnimation

缩放:ScaleAnimation

渐变:AlphaAnimation

小案例：给RecyclerView设置动画。

②属性动画

ObjectAnimatior: translation(x或y),rotation(x或y),scale(x或y)

ValueAnimator

ValueAnimator 类是先改变值，然后 手动赋值 给对象的属性从而实现动画；是 间接 对对象属性进行操作；

ObjectAnimator 类是先改变值，然后 自动赋值 给对象的属性从而实现动画；是 直接 对对象属性进行操作；

ObjectAnimator 类针对的是任意对象 & 任意属性值，并不是单单针对于View对象

如果需要采用ObjectAnimator 类实现动画效果，那么需要操作的对象就必须有该属性的set（） & get（）

③帧动画

AniamteDrawable

### lottie-android

github地址：<https://github.com/airbnb/lottie-android>

文档地址：<http://airbnb.io/lottie/#/android>

素材库：<https://lottiefiles.com>

### 事件分发

onTouchEvent:触摸事件的处理

dispatchTouchEvent:传递触摸事件

onInterceptTouchEvent:拦截事件传递

requestDisallowedInterceptTouchEvent(boolean disallowIntercept):请求自己的父布局不要拦截事件

1、安卓中事件的消费会 优先让子View去处理。

2、如果ViewGroup很想处理这个事件怎么办？ onInterceptTouchEvent

3、requestDisallowedInterceptTouchEvent 请求ViewGroup不要去处理这个事件。

# part10：高级进阶一

### 项目的依赖

https://www.jianshu.com/p/f34c179bc9d0

### 性能优化

1. 内存优化

JVM内存模型：

1. <https://www.jianshu.com/p/4455e4234d5c>
2. <https://www.cnblogs.com/hewenwu/p/3662529.html>

GC原理：

1. <https://blog.csdn.net/lilong329329/article/details/82222713>
2. <https://my.oschina.net/dabird/blog/1594600>
3. <https://www.cnblogs.com/jeffwongishandsome/p/talk-about-GC-and-how-to-use-GC-better.html>

内存泄漏：

# part11：手写APT框架

详见APT文档

# part12：音视频架构

### 什么是service

服务是Android四大组件之一，及其重要。服务是Android中实现程序后台运行的解决方案，他非常适合是去执行那些不需要和用户交互而且还要长期运行的任务。服务的运行不依赖于任何用户界面，即使程序被切换到后台，或者用户打开了另一个应用程序，服务仍然能够保持独立运行。不过需要注意的是，服务并不是运行在一个独立的进程当中，而是依赖于创建服务时所在的应用程序进程。当某个应用程序被杀掉时，所有依赖该进程的服务也会停止运行

### service种类

前台服务、后台服务

本地服务、远程服务

### 使用方式

1、继承

2、清单文件中注册

3、开启服务：两种方式，startService，bindService

4、关闭服务

### meta-data

<https://blog.csdn.net/yechaoa/article/details/80869987>

### MediaPlayer 原理

1、动态加载链接库

<https://www.cnblogs.com/haitaofeiyang/p/7698121.html>

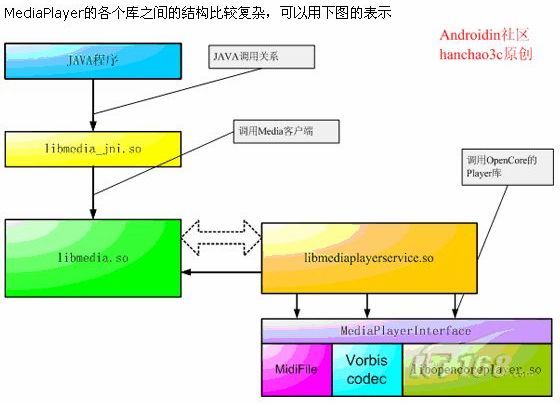
2、Native 方法

<https://baike.baidu.com/item/native/13128691>

3、Android c++ 源码查看

<http://androidxref.com/>

4、MediaPlayer架构



5、MediaPlayer弊端

兼容性、支持协议较少（如hls协议）

升级版：ExoPlayer、ijkplayer（FFmpeg）

视频地址：<https://rescdn.yishihui.com/longvideo/transcode/video/2018-11-10/2459c73737b340eba0ed5c960dba921a1541923271382.m3u8>

### SurfaceView 家族体系

1、GlSurfaceView

2、VideoView

### Surface 的特点

1、具有独立的绘图表面

2、需要在宿主窗口上挖一个洞来显示自己

3、它的UI绘制可以在单独的线程中进行，这样就可以进行负责的UI绘制，并且不会影响应用程序的主线程响应用户输入。

### Surface 的类型

SURFACE\_TYPE\_NORMAL：用RAM缓存原生数据的普通Surface   
SURFACE\_TYPE\_HARDWARE：适用于DMA(Direct memory access )引擎和硬件加速的Surface   
SURFACE\_TYPE\_GPU：适用于GPU加速的Surface   
SURFACE\_TYPE\_PUSH\_BUFFERS：表明该Surface不包含原生数据，Surface用到的数据由其他对象提供，在Camera图像预览中就使用该类型的Surface，有Camera负责提供给预览Surface数据，这样图像预览会比较流畅。如果设置这种类型则就不能调用lockCanvas来获取Canvas对象了。

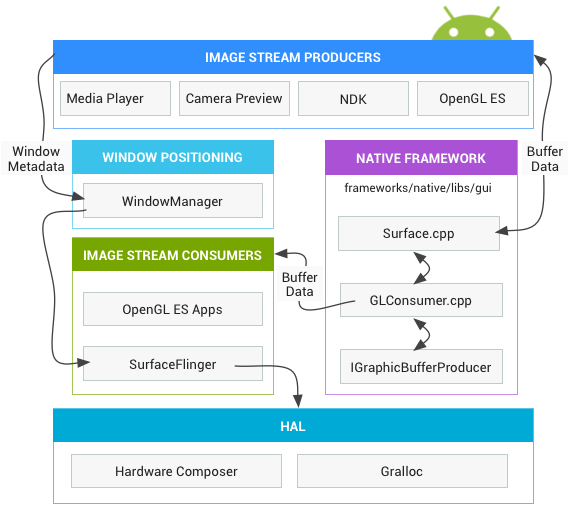
### Canvas

2D图形渲染 Skia引擎 ： SkCanvas.cpp

### GlSurfaceView

EGLSurface 和 OpenGL ES。OpenGL ES (GLES) 定义了用于与 EGL 结合使用的图形渲染 API。EGI 是一个规定如何通过操作系统创建和访问窗口的库（要绘制纹理多边形，请使用 GLES 调用；要将渲染放到屏幕上，请使用 EGL 调用）。此页还介绍了 ANativeWindow，它是 Java Surface 类的 C/C++ 等价类，用于通过原生代码创建 EGL 窗口表面。

### 安卓图形/图像 原理



### 图像流生产方

图像流生产方可以是生成图形缓冲区以供消耗的任何内容。例如 OpenGL ES、Canvas 2D 和 mediaserver 视频解码器。

### 图像流消耗方

图像流的最常见消耗方是 SurfaceFlinger，该系统服务会消耗当前可见的 Surface，并使用窗口管理器中提供的信息将它们合成到显示部分。SurfaceFlinger 是可以修改所显示部分内容的唯一服务。SurfaceFlinger 使用 OpenGL 和 Hardware Composer 来合成一组 Surface。

其他 OpenGL ES 应用也可以消耗图像流，例如相机应用会消耗相机预览图像流。非 GL 应用也可以是消耗方，例如 ImageReader 类。

# part13：进程

**-进程四要素**

有一段程序供其执行

拥有专用的系统堆栈空间

在内核存在对应进程控制块

拥有独立的用户存储空间

adb shell dumpsys meminfo 打印当前手机进程的所有信息

**-进程间数据共享**

1、广播

广播的类型

普通广播（Normal Broadcast）

系统广播（System Broadcast）

有序广播（Ordered Broadcast）

App应用内广播（Local Broadcast）

2、内容提供者



# part14：混合技术开发

### c++

也就是Android中的JNI NDK 开发，特点是 效率高、保密性、可引入优良的c++ 库。

FFmpeg

V8

OpenCV

OpenGl ES

<https://www.cnblogs.com/skyus/articles/8524408.html>

### openGL

图形图像渲染库，在混合开发中起到至关重要的作用，比如微信小程序。

微信小程序：<https://developers.weixin.qq.com/miniprogram/dev/api/canvas/RenderingContext.html>

### JNI 及 c ++ 基本语法

宏定义：#define

**JNIEXPORT ：表示可以被外部调用**

JNIEnv ：JNI 环境，可以理解 Android中的Context

C++ 和 Java通信

1、java 通讯到C++，通过Native方法 调用JNI

2、c++ 通讯到 Java，通过JNI反射到Java

导包：#include

#include<GLES2/gl2.h>

#include<GLES2/gl2ext.h>

1. ndk{
2. abiFilters **"armeabi-v7a"**,**"x86"**
3. }

# part15：混合编程

### WebView

<https://www.jianshu.com/p/3e0136c9e748>

### ReactNative

<https://reactnative.cn/>

React ：

<https://baike.baidu.com/item/react/18077599?fr=aladdin>

Node:

<http://nodejs.cn/>

npm 类似于 Android 中的 Maven 依赖，是 Node.js的一部分，俗称包管理器。

ReactNative 核心原理 包括3层：

1、JS\React 用于开发页面

2、c++ 层，负责 Js、React 和 原生进行交互和通信（JavaScript Runtime 引擎）

3、原生（Android、IOS）

严重缺陷：Android 10.0 新增了一个非常牛逼的控件，ReactNative 必须要更新SDK，有一个强依赖的特点。

### Weex

<https://weex.apache.org/zh/guide/introduction.html>

### 微信小程序

<https://developers.weixin.qq.com/miniprogram/dev/framework/>

### 总结：

**ReactNative 2015年 (跨平台框架)**

Weex 2016年 （跨平台框架）

微信小程序 2017年 （商业用途）

支付宝小程序 2018年 （商业用途）

快应用联盟 2018年 （商业用途）

…

### 跨平台开发框架实战

j2v8

### Flutter

<https://flutterchina.club>

### 跨平台技术简介

* H5+原生（Cordova、Ionic、微信小程序）
* JavaScript开发+原生渲染 （React Native、Weex、快应用）
* 自绘UI+原生(QT for mobile、Flutter)

### Hybrid（H5+原生）技术

WebView + JsBridge

混合应用的优点是动态内容是H5，web技术栈，社区及资源丰富，缺点是性能不好，对于复杂用户界面或动画，WebView不堪重任。

### JavaScript开发+原生渲染

快应用 ：<https://www.quickapp.cn>

JavaScript开发+原生渲染的方式主要优点如下：

采用Web开发技术栈，社区庞大、上手快、开发成本相对较低。

原生渲染，性能相比H5提高很多。

动态化较好，支持热更新。

不足：

渲染时需要JavaScript和原生之间通信，在有些场景如拖动可能会因为通信频繁导致卡顿。

JavaScript为脚本语言，执行时需要JIT(Just In Time)，执行效率和AOT(Ahead Of Time)代码仍有差距。

由于渲染依赖原生控件，不同平台的控件需要单独维护，并且当系统更新时，社区控件可能会滞后；除此之外，其控件系统也会受到原生UI系统限制，例如，在Android中，手势冲突消歧规则是固定的，这在使用不同人写的控件嵌套时，手势冲突问题将会变得非常棘手。

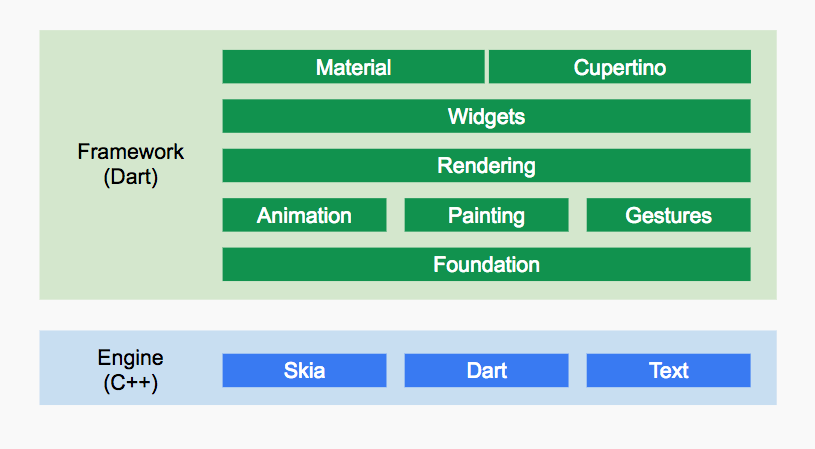
### 自绘UI+原生

* 2017 年 Google I/O 大会上，Google 首次推出了一款新的用于创建跨平台、高性能的移动应用框架——Flutter。
* 2018年2月，Flutter发布了第一个Beta版本，同年五月， 在2018年Google I/O 大会上，Flutter 更新到了 beta 3 版本。

| **技术类型** | **UI渲染方式** | **性能** | **开发效率** | **动态化** | **框架代表** |
| --- | --- | --- | --- | --- | --- |
| H5+原生 | WebView渲染 | 一般 | 高 | 支持 | Cordova、Ionic |
| JavaScript+原生渲染 | 原生控件渲染 | 好 | 中 | 支持 | RN、Weex |
| 自绘UI+原生 | 调用系统API渲染（跟Android很相似） | 好 | Flutter高, QT低 | 默认不支持 | QT、Flutter |

* 2018年6月，Flutter发布了首个预览版本。

### Flutter



**AOT** （Ahead of time）即 “提前编译

**JIT**（Just-in-time）即“即时编译”

### 硬件绘图基本原理

提到原理，我们要从屏幕显示图像的基本原理谈起。我们知道显示器（屏幕）是由一个个物理显示单元组成，每一个单元我们可以称之为一个物理像素点，而每一个像素点可以发出多种颜色，显示器成相的原理就是在不同的物理像素点上显示不同的颜色，最终构成完整的图像。

一个像素点能发出的所有颜色总数是显示器的一个重要指标，比如我们所说的1600万色的屏幕就是指一个像素点可以显示出1600万种颜色，而显示器颜色是有RGB三基色组成，所以1600万即2的24次方，即每个基本色（R、G、B）深度扩展至8 bit(位)，颜色深度越深，所能显示的色彩更加丰富靓丽。

为了更新显示画面，显示器是以固定的频率刷新（从GPU取数据），比如有一部手机屏幕的刷新频率是 60Hz。当一帧图像绘制完毕后准备绘制下一帧时，显示器会发出一个垂直同步信号（如VSync）， 60Hz的屏幕就会一秒内发出 60次这样的信号。而这个信号主要是用于同步CPU、GPU和显示器的。一般地来说，计算机系统中，CPU、GPU和显示器以一种特定的方式协作：CPU将计算好的显示内容提交给 GPU，GPU渲染后放入帧缓冲区，然后视频控制器按照同步信号从帧缓冲区取帧数据传递给显示器显示。

CPU和GPU的任务是各有偏重的，CPU主要用于基本数学和逻辑计算，而GPU主要执行和图形处理相关的复杂的数学，如矩阵变化和几何计算，GPU的主要作用就是确定最终输送给显示器的各个像素点的色值。

### Dart

2011年10月10日,Google 发布了Dart语言

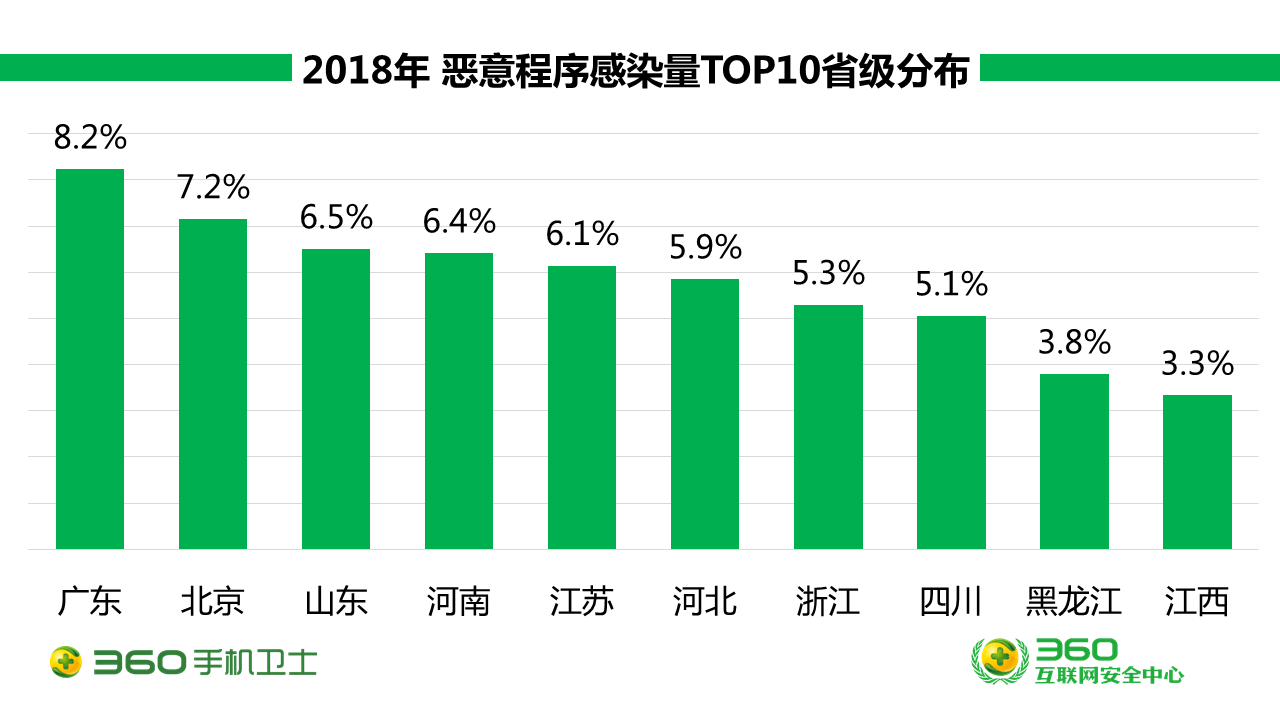
2013年11月14日，谷歌发布Dart 1.0版本

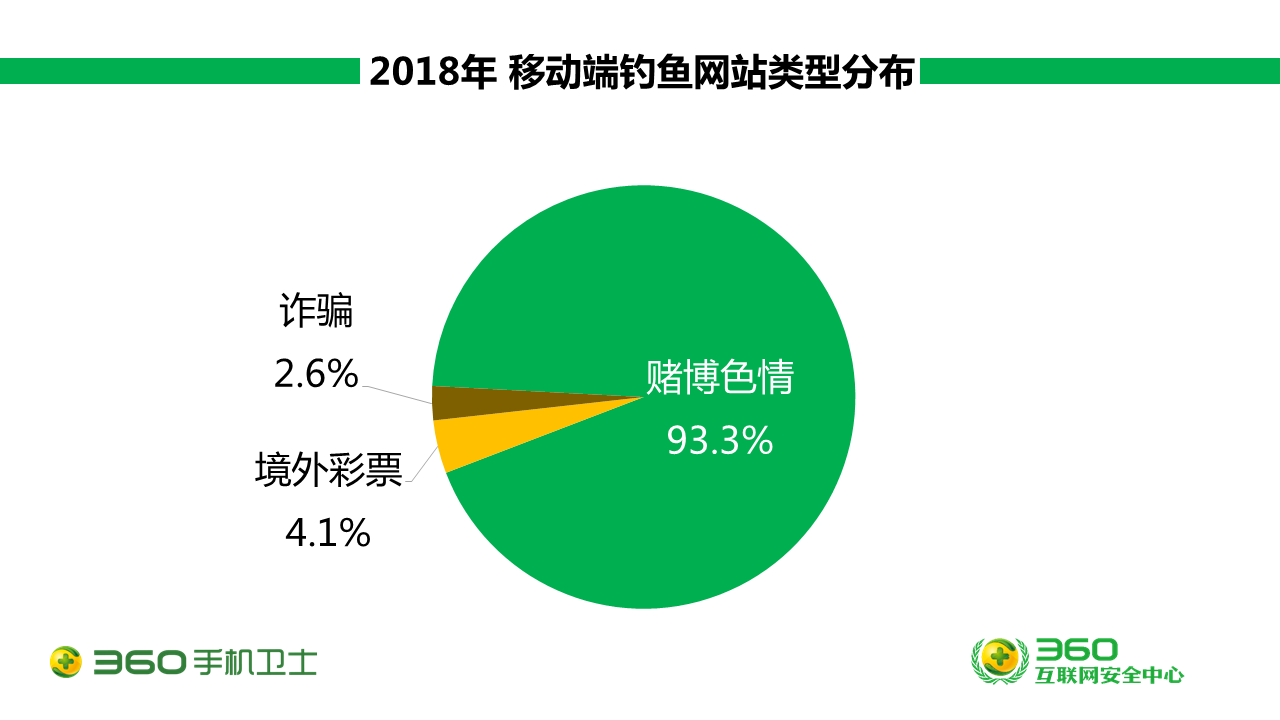
2018年8月8日，谷歌发布Dart2.0版本

最初设计Dart，是Google的一帮程序员出于对JavaScript的不满，决定自己搞一个新语言用来替换JavaScript的，所以刚开始Dart也就是用来作为浏览器脚本运行在浏览器中的。  
为了推广Dart，Google利用自己的利器Chrome让它内置了DartVM的引擎。有了运行环境，加上用户群的可观数量，Dart最初也赢得了部分前端开发者的青睐。但Google的这帮程序员万万没想到，看似并无大用的JavaScript居然因为NodeJS而焕发了第二春。此时移动互联网处于萌芽发展中，一个叫做NodeJS的开源项目借助这股移动互联的东风顺势而起，很快的烧遍了前端、烧到了后端，甚至烧到了手机应用和桌面应用，可以说是在编程界无孔不入，几乎成就了JS程序员口中的“凡是能被JS实现的，最终都要被JS实现”的愿景。前端的开发模式因此而改变，进入了一个新的天地。一些明星项目如React、React-Native、Vue等等如雨后春笋般迅速的崛起，一句话总结就是“Javascript很忙”。  
Javascript很忙，程序员更忙，Dart就这样被慢慢地遗忘在角落。加上一直以来缺少顶级的项目使用，就算再是亲儿子，得不到JS开发者的心，Google也没什么办法，在2015年的时候，Google在Chrome中默默的移除了内置的Dart引擎。  
但是Google并没有放弃Dart，Google内部用Dart编写孵化了一个移动开发框架Sky，之后又被命名为Flutter，进入了移动跨平台开发的领域。这就是Flutter的由来。

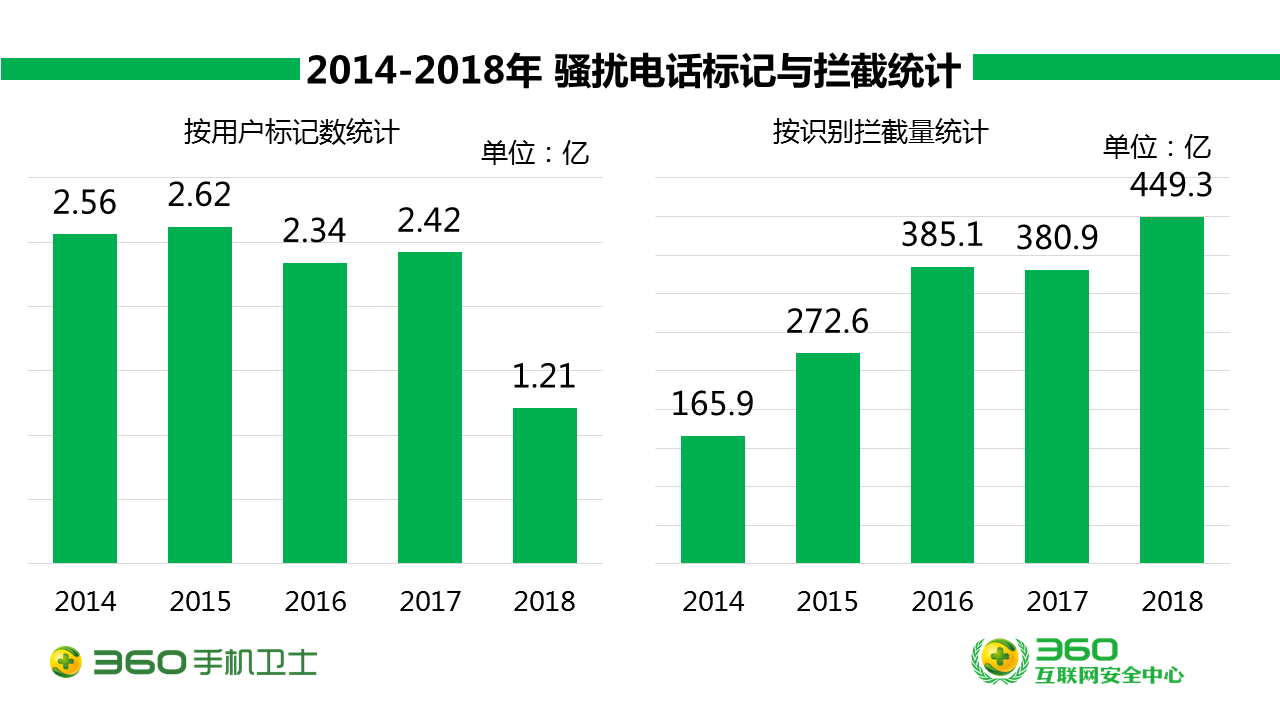
# part16：移动安全

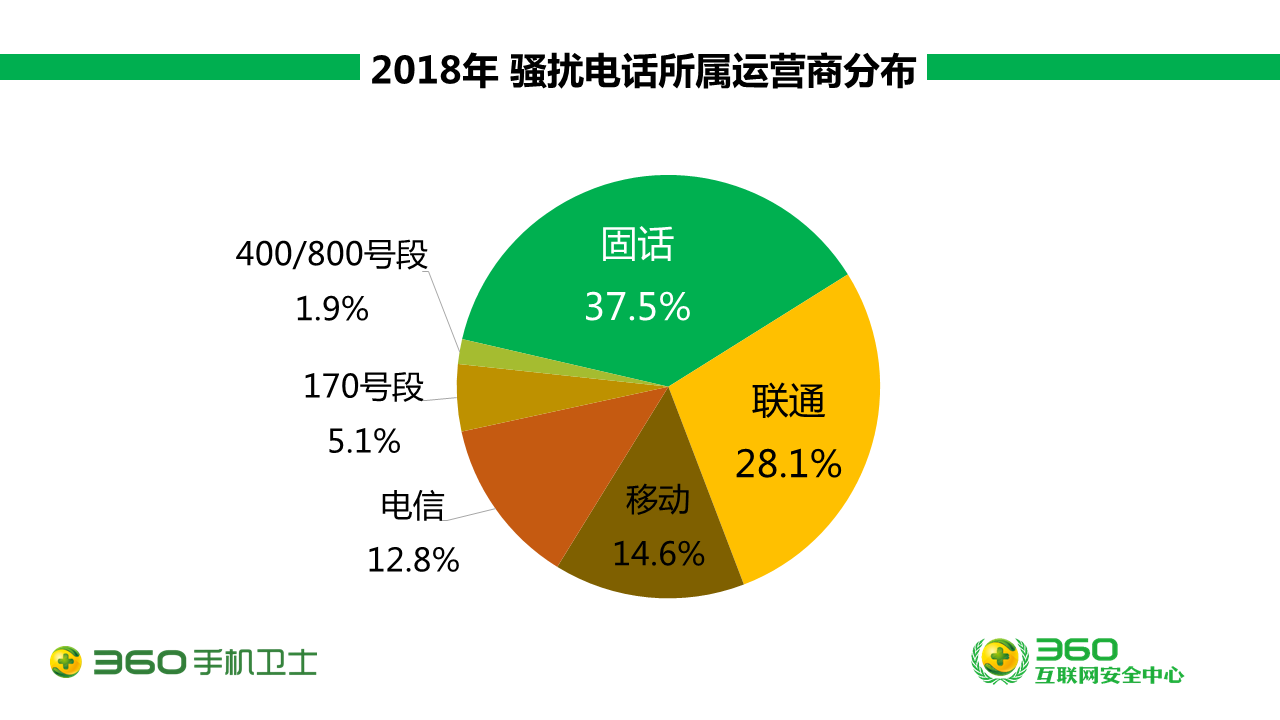
### 为什么要讲移动安全？

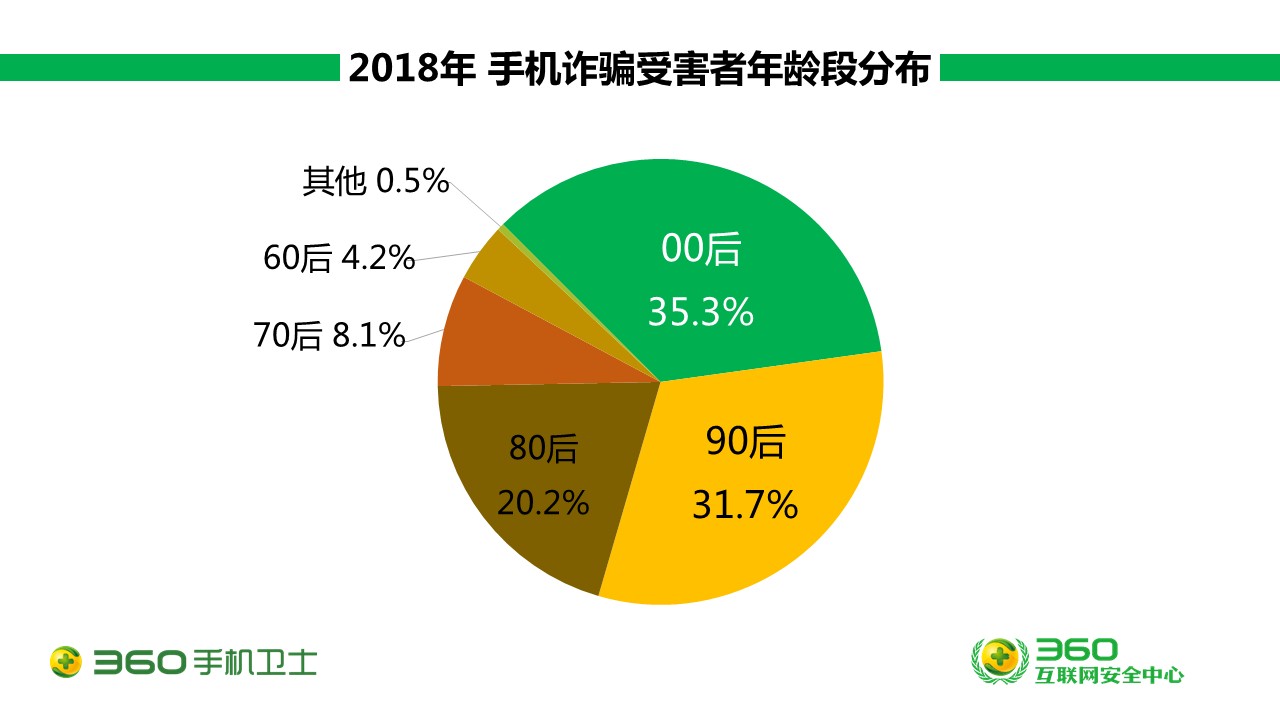
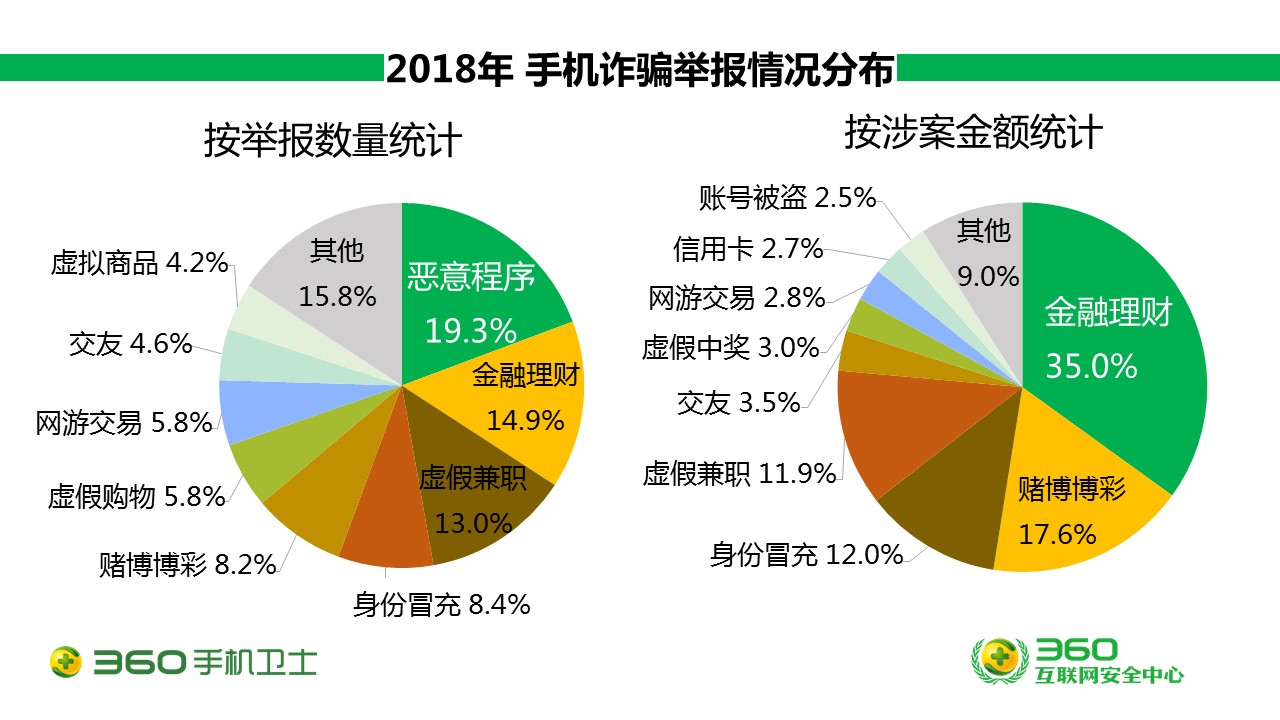
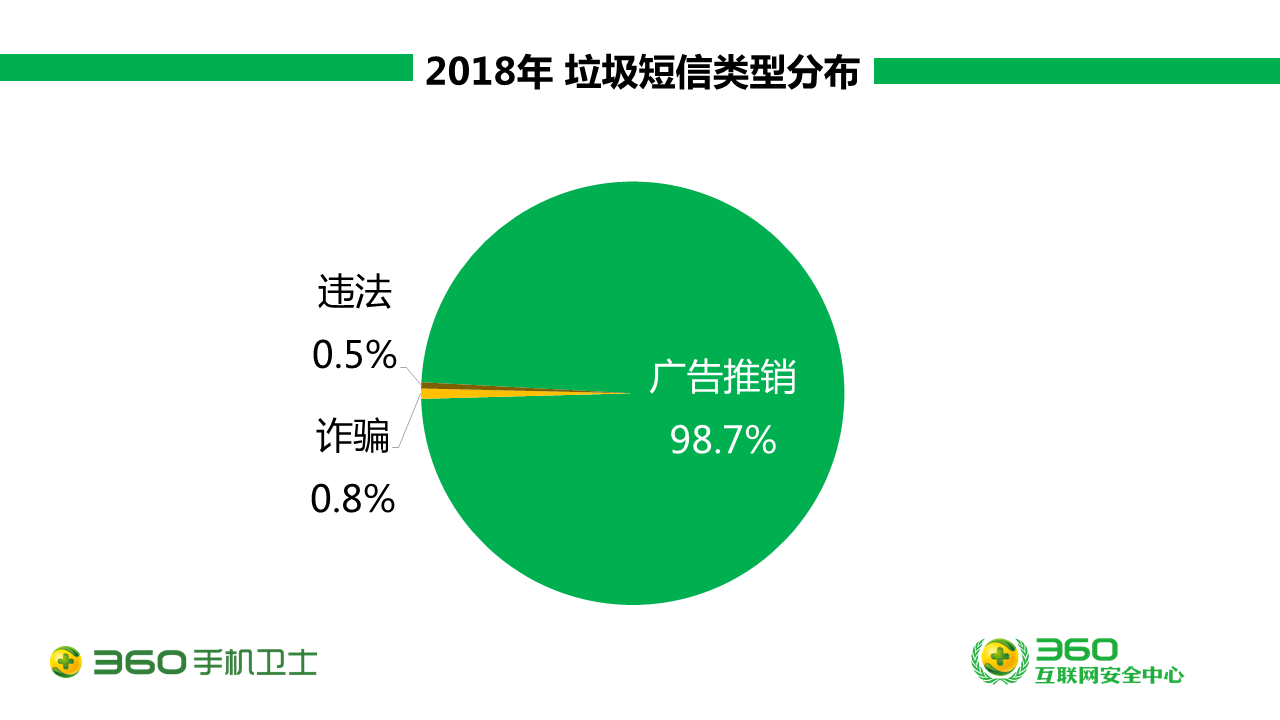




浏览器看一篇新闻，为什么会出现 “澳门赌场”、“菲律宾彩票”等图片、卡片？



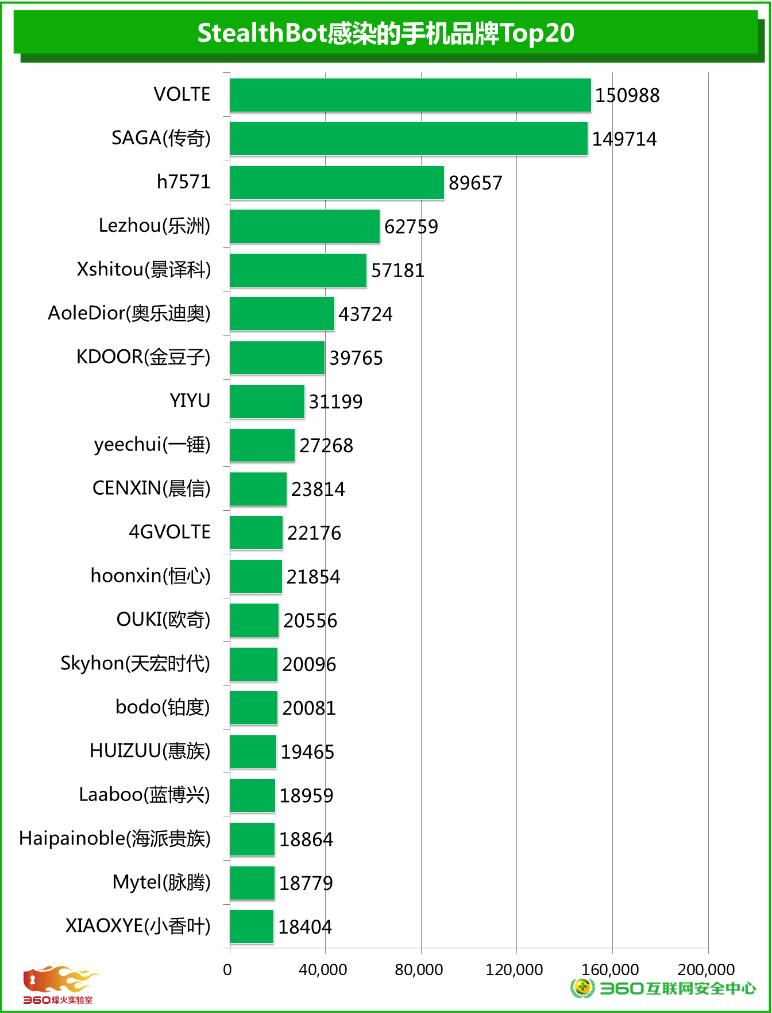




可以看下电影 巨额来电，了解电信诈骗的各个环节。

### （用户）如何防范移动安全?

1、不要选择小众品牌的手机



底层ROM漏洞是移动安全最重要的威胁，不排除手机厂商会通过内置应用恶意收集用户信息，这样用户很轻易的变成了“肉鸡”

2、不要轻易Root手机

3、不要轻易输入账号和密码

4、不要轻易授予权限

5、不要轻易连接不明wifi

sslstrip 攻击 （<https://blog.csdn.net/qq_44693903/article/details/90109631>）

kali ：黑客入门级工具集

6、http 升级 为https，防止http劫持

### （开发者）如何防范移动安全?

* **不要明文存储重要数据**

危险分析：手机root后，会随意访问任何目录文件，造成隐私数据泄露风险。

常用的加密算法：

MD5： 在线 加密解密 网址 <https://www.cmd5.com>

MD5信息摘要算法（英语：MD5 Message-Digest Algorithm），一种被广泛使用的密码散列函数。

密码散列函数：通常是指不可逆的算法。

MD5常用的场景：

1、传递重要数据

2、下载文件中作为文件名，也就是文件指纹

拓展：

1、MD5 + salt ：todayInformation + jiami

2、SHA

* 文件存储权限漏洞

标准情况下，私有目录的文件权限最后三位应为空(类似“rw-rw----”)，即除应用本身外任何人无法读写;目录则允许多一个执行位(类似“rwxrwx—r/w/rw”)。

漏洞模拟：

1、编写漏洞App 存储sp 为 MODE\_WORLD\_READABLE

2、编写黑客 App 获取 漏洞 App的数据内容。

* 网络安全

http：数据内容是明文传输的，容易被流量攻击

https：数据是加密的，采用SSL/TLS  
SSL/TLS防止的安全风险：

1）窃听风险

2）篡改风险

3）冒充风险  
SSL/TLS防止的安全风险原理：

1）加密：非对称加密+对称加密，主要解决的是窃听风险

2）校验：数字签名，主要解决的是篡改风险

3）证书：数字证书，主要解决的是冒充风险

**对称加密算法：**

加密和解密都使用同一把秘钥，这种加密方法称为对称加密，也称为单密钥加密。

常用的对称加密算法：AES、DES、3DES、TDEA、Blowfish、RC2、RC4、RC5、IDEA、SKIPJACK 等

对称加密算法常用场景：

1. 本地数据加密(例如加密 android 里 SharedPreferences 里面的某些敏感数据)

2. 网络传输:登录接口 post 请求参数加密{username=lisi,pwd=oJYa4i9VASRoxVLh75wPCg==}

3. Https

对称加密引发的安全性思考：秘钥如何保存和传输（非对称加密算法）

**非对称加密算法：**

与对称加密算法不同，非对称加密算法需要两个密钥:公钥(publickey)和私钥(privatekey)。公钥 与私钥是一对，如果用公钥对数据进行加密，只有用对应的私钥才能解密;如果用私钥对数据进行加密， 那么只有用对应的公钥才能解密。因为加密和解密使用的是两个不同的密钥，所以这种算法叫作非对称加 密算法。

常见的非对称加密算法：

RSA、Elgamal、背包算法、Rabin、D-H、ECC(椭圆曲线加密算法)等

RSA算法原理：

质因数、欧拉函数、模反元素

原理很复杂，只需要知道内部是基于分解质因数和取模操作即可。

使用步骤：

//获取 cipher 对象

Cipher cipher = Cipher.getInstance("RSA");

//通过秘钥对生成器 KeyPairGenerator 生成公钥和私钥

KeyPair keyPair = KeyPairGenerator.getInstance("RSA").generateKeyPair();

//使用公钥进行加密，私钥进行解密(也可以反过来使用)

PublicKey publicKey = keyPair.getPublic();

PrivateKey privateKey = keyPair.getPrivate();

//使用公钥初始化密码器

cipher.init(Cipher.ENCRYPT\_MODE, publicKey);

//执行加密操作

byte[] result = cipher.doFinal(content.getBytes());

//使用私钥初始化密码器

cipher.init(Cipher.DECRYPT\_MODE, privateKey);

//执行解密操作

byte[] deResult = cipher.doFinal(result);

非对称加密算法的用途：

1、身份认证

2、陌生人通信

3、秘钥交换

非对称加密一般不会单独拿来使用，他并不是为了取代对称加密而出现的，非对称加密速度比对称加密慢 很多，极端情况下会慢 1000 倍，所以一般不会用来加密大量数据，通常我们经常会将对称加密和非对称 加密两种技术联合起来使用，例如用非对称加密来给称加密里的秘钥进行加密(即秘钥交换)。

**数字签名**

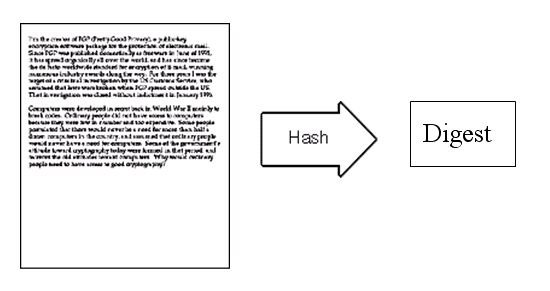
数字签名技术结合Hash算法和加密算法，来防止消息被篡改和进行身份认证。

1、发送方使用Hash算法对原文产生信息摘要，原文不变则信息摘要不变。

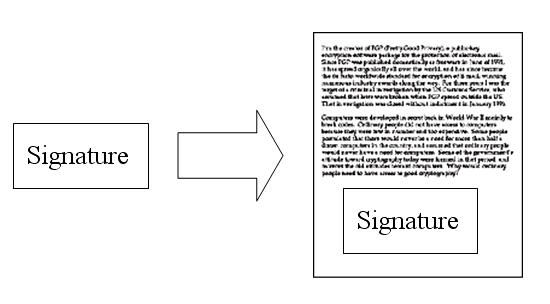
2、发送方使用自己的私钥对信息摘要加密产生数字签名，并和加密的原文一起发送。

3、接收方使用发送方的公钥解密获取数字签名

4、接收方使用Hash算法对原文计算信息摘要与解密的信息摘要比对，成功表示未篡改、未成功表示篡改。

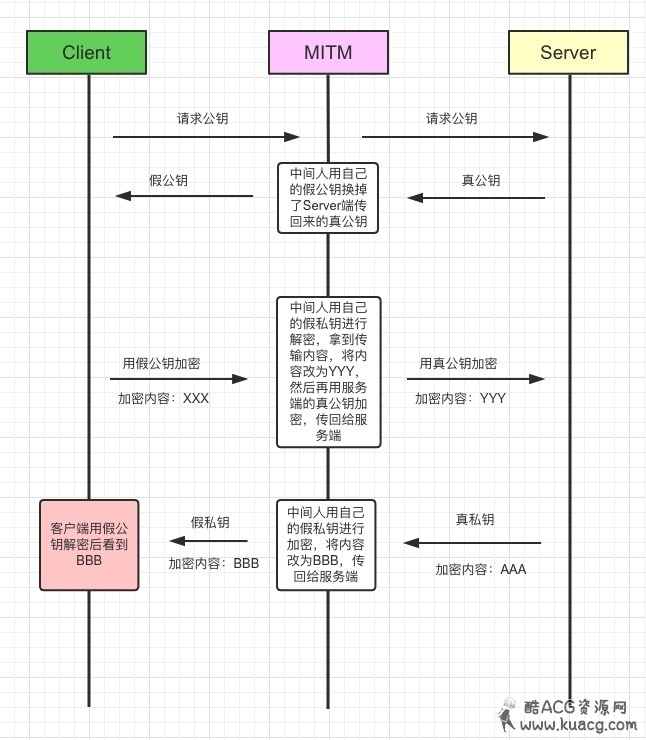






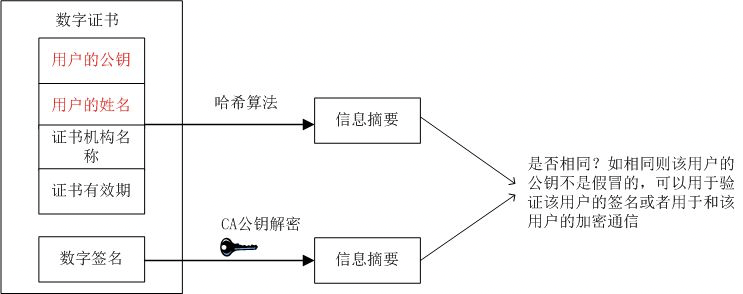


**数字证书**

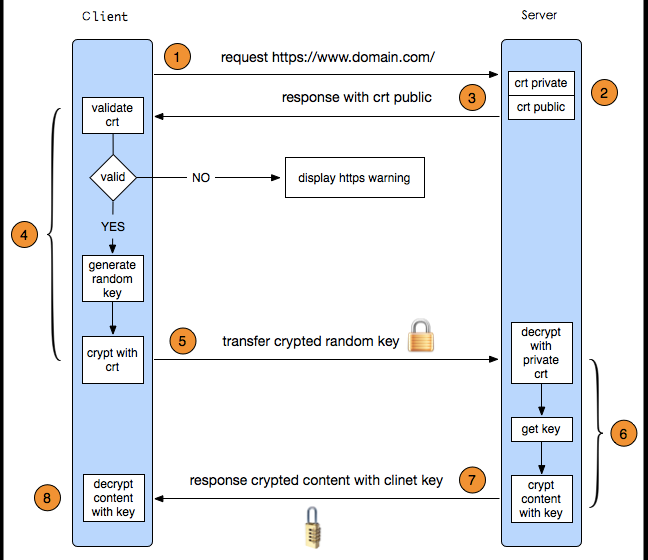


数字证书就是互联网通讯中标志通讯各方身份信息的一串数字，提供了一种在 Internet 上验证通信实体 身份的方式，数字证书不是数字身份证，而是身份认证机构盖在数字身份证上的一个章或印(或者说加在 数字身份证上的一个签名)。它是由权威机构——CA 机构，又称为证书授权(Certificate Authority)中心 发行的

数字签名一般不单独使用，基本都是用在数字证书里实现 SSL 通信协议。



数字证书是安全领域里的终极武器，SSL 通信协议里最核心的东西就是数字证书。他涉及到前面提到的 所有知识:对称加密、非对称加密、消息摘要、数字签名等。



**Android APK 打包原理**



1、AAPT（Android Asset Packaging Tool）工具，Android资源打包工具。会打包资源文件（res文件夹下的文件），并生成R.java和resources.arsc文件。

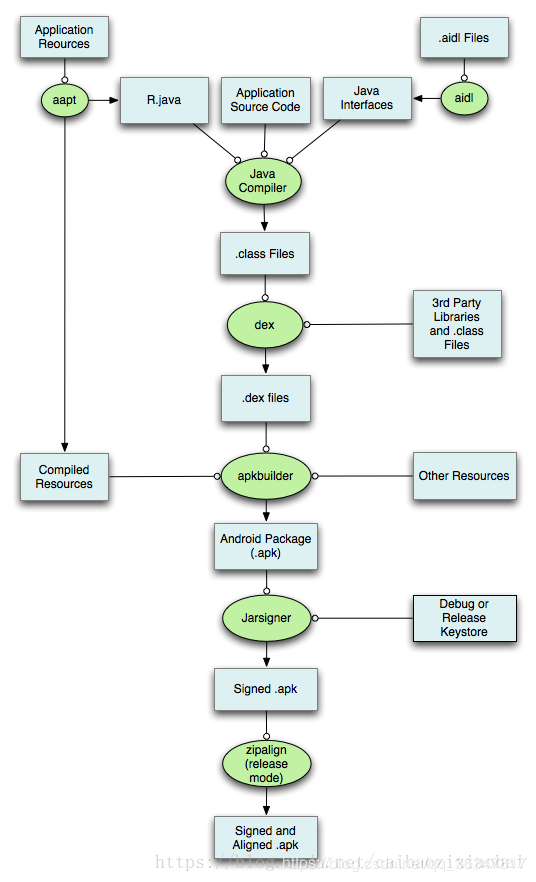
2、AIDL工具会将所有的.aidl文件编译成.java文件。

3、JAVAC工具将R.java、AIDL接口生成的java文件、应用代码java文件编译成.class文件。

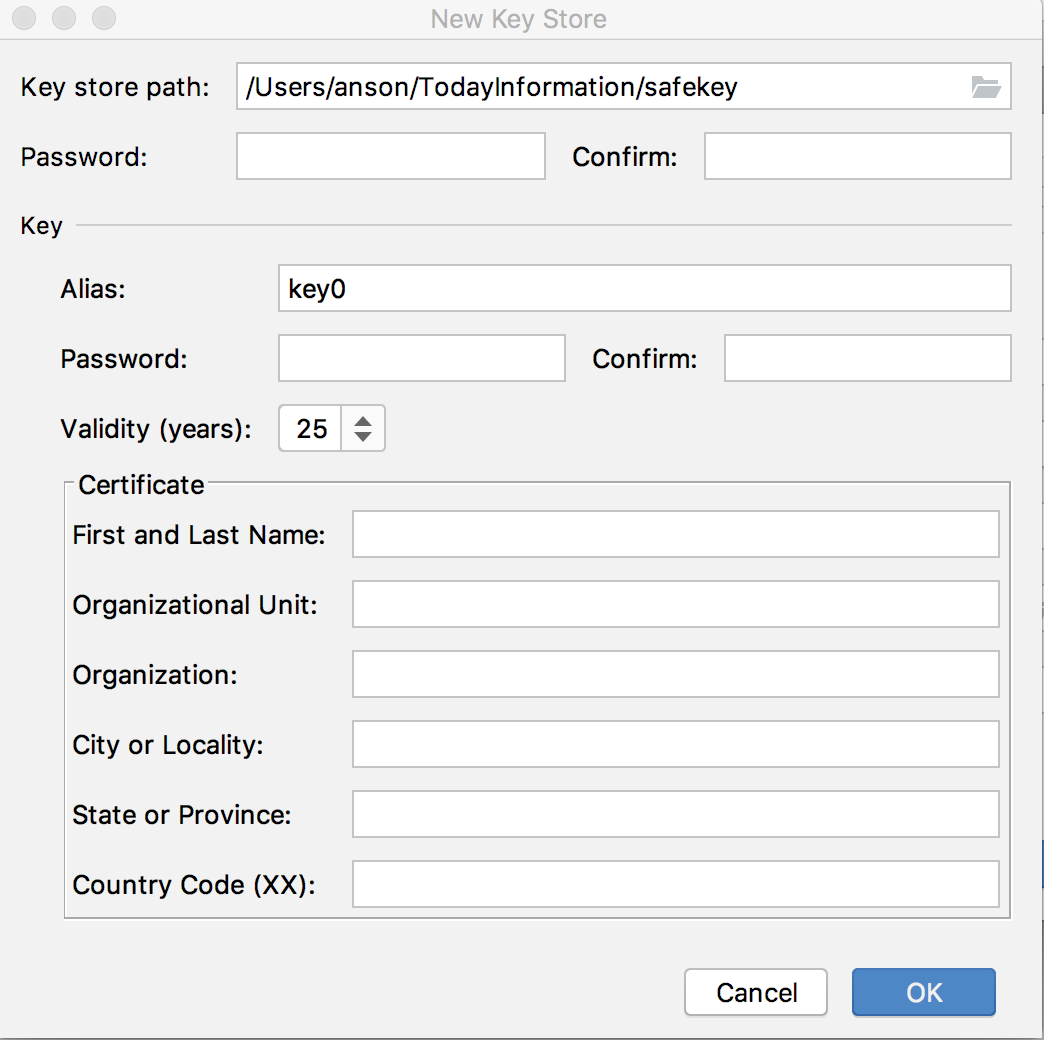
4、dx脚本将很多.class文件转换打包成一个.dex文件（为什么要转成.dex?）。

5、apkbuilder脚本将资源文件和.dex文件生成未签名的.apk文件(实际上apk就相当于一个zip文件)。

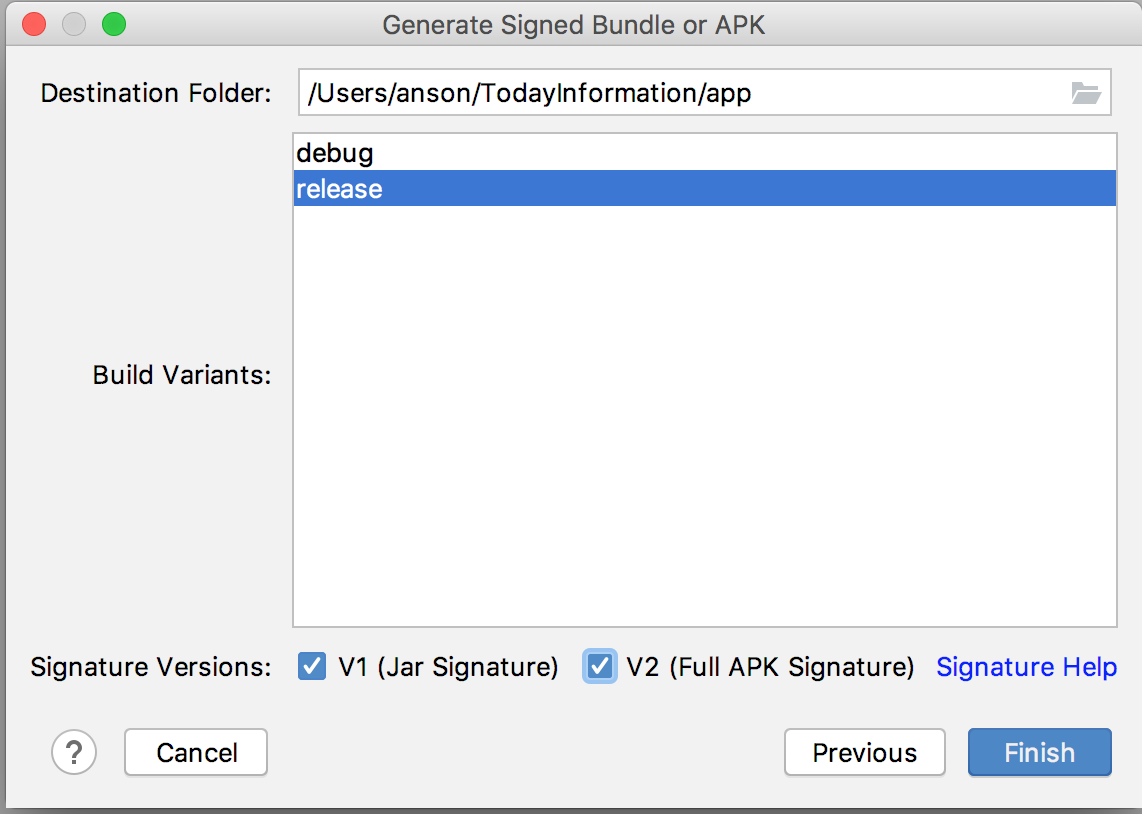
6、jarsigner对apk进行签名（安全性校验）。



生成签名apk，需要保存key store 和 认证信息。



签名版本的原则



如何查看KeyStore文件：

keytool -list -v -keystore safekey -storepass 123456

**Android APK 签名流程**

为了防止 APK 在传送的过程中被第三方篡改，Google 引入了签名机制。

签过名的 APK 文件比未签名的 APK 文件多了一个 META-AF 文件夹，包含以下三个文件。签名的信息就在这三个文件中

1、对 APK 文件夹中的文件逐一遍历进行 SHA1 （或者 SHA256）算法计算文件的消息摘要，然后进行 BASE64 编码后，作为 “SHA1-Digest” 属性的值写入到 MANIFEST.MF 文件中的一个块中。该块有一个 “Name” 属性，其值就是该文件在 APK 包中的路径。

2、计算这个 MANIFEST.MF 文件的整体 SHA1 值，再经过 BASE64 编码后，记录在 CERT.SF 主属性块（在文件头上）的 “SHA1-Digest-Manifest” 属性值值下。然后，再逐条计算 MANIFEST.MF 文件中每一个块的 SHA1，并经过 BASE64 编码后，记录在 CERT.SF 中的同名块中，属性的名字是 “SHA1-Digest” 。

Signature-Version: 1.0

Created-By: 1.0 (Android)

SHA1-Digest-Manifest: MJQyZ0dc4dv7G9nlJPAMQLwEwbU=

X-Android-APK-Signed: 2

Name: AndroidManifest.xml

SHA1-Digest: IJioMmfD693T4qnUJcPKhq9woHQ=

Name: META-INF/android.arch.core\_runtime.version

SHA1-Digest: OPQCkzMXJVPQryHeMowVNZmfRMw=

Name: META-INF/android.arch.lifecycle\_livedata-core.version

SHA1-Digest: TSBGEIW1zN2n2sraHWcuRYSO8JU=

3、把之前生成的 CERT.SF 文件， 用私钥计算出签名, 然后将签名以及包含公钥信息的数字证书一同写入 CERT.RSA 中保存。

3082 02f9 0609 2a86 4886 f70d 0107 02a0

8202 ea30 8202 e602 0101 310b 3009 0605

2b0e 0302 1a05 0030 0b06 092a 8648 86f7

0d01 0701 a082 01e1 3082 01dd 3082 0146

0201 0130 0d06 092a 8648 86f7 0d01 0105

0500 3037 3116 3014 0603 5504 030c 0d41

6e64 726f 6964 2044 6562 7567 3110 300e

0603 5504 0a0c 0741 6e64 726f 6964 310b

3009 0603 5504 0613 0255 5330 1e17 0d31

**签名校验**

1、完整性校验

如果 APK 中文件被修改，对应 MANIFEST.MF 中的 SHA1 会发生改变。

2、APK 作者身份唯一性校验

当在 Android 设备上安装 APK 包时，会从存放在 CERT.RSA 中的公钥证书中提取公钥，进行 RSA 解密来校验安装包的身份。 使用不同的 key 生成的签名信息会不同，不同的私钥对应不同的公钥，因此最大的区别是签名证书中存放的公钥会不同，所以我们可以通过提取 CERT.RSA 中的公钥来检查安装包是否被重新签名了。

**keystore和证书格式**

我们在对Apk签名时并没有直接指定私钥、公钥和数字证书，而是使用keystore文件，这些信息都包含在了keystore文件中。根据编码不同，keystore文件分为很多种，Android使用的是Java标准keystore格式JKS(Java Key Storage)，所以通过Android Studio导出的keystore文件是以.jks结尾的。

keystore使用的证书标准是X.509，X.509标准也有多种编码格式，常用的有两种：pem（Privacy Enhanced Mail）和der（Distinguished Encoding Rules）

DER（Distinguished Encoding Rules）二进制格式，所有类型的证书和私钥都可以存储为der格式。

PEM（Privacy Enhanced Mail）base64编码

从Android 7.0开始，Android支持了一套全新的V2签名机制，为什么要推出新的签名机制呢？

签名校验速度慢

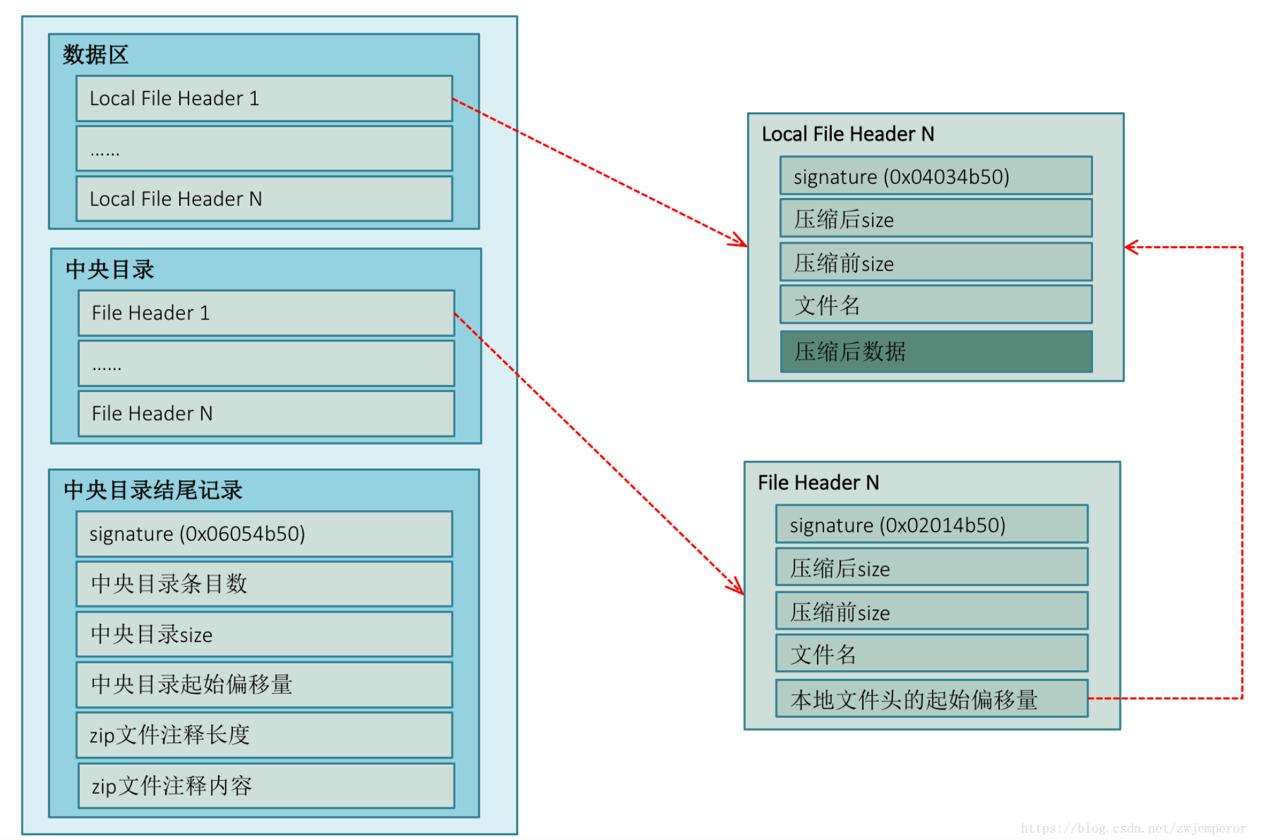
校验过程中需要对apk中所有文件进行摘要计算，在apk资源很多、性能较差的机器上签名校验会花费较长时间，导致安装速度慢；

v1签名用的 jarsigned ，v2签名用的apksigned

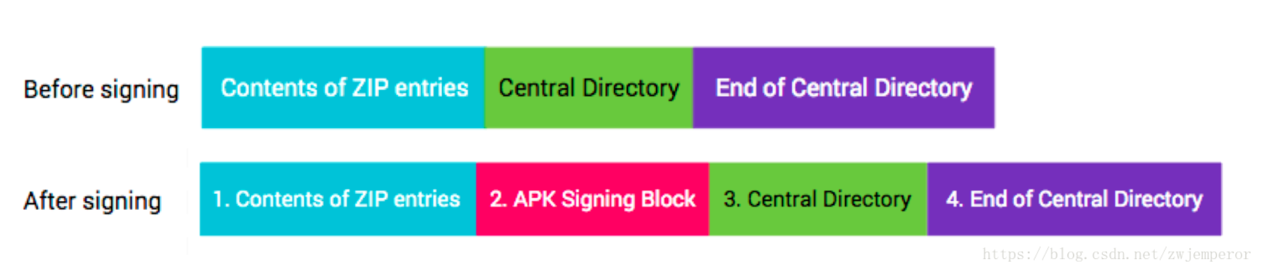
**v2签名原理**

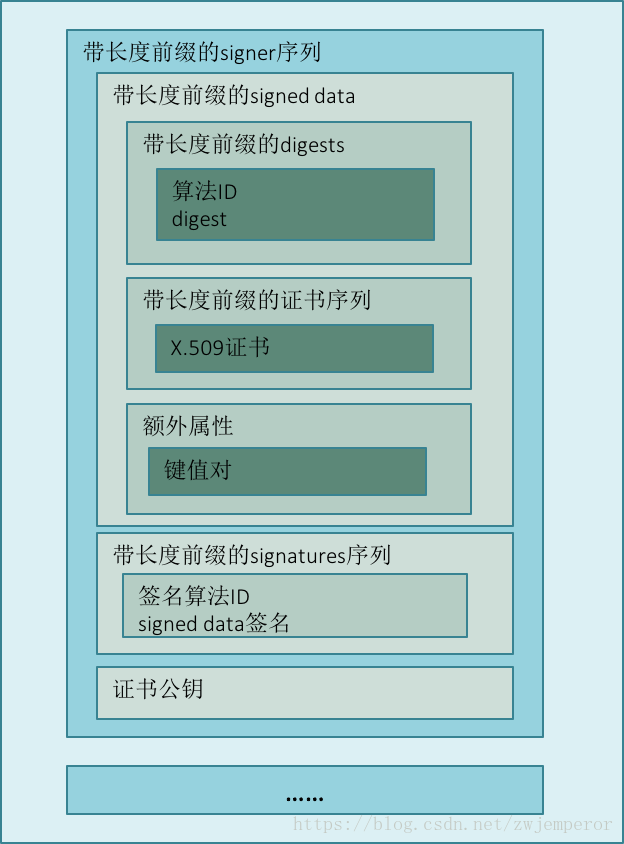
v2签名算法主要核心思想是在打包APK时，在压缩数据结构中做文章，APK实际是个ZIP结构，先了解下ZIP。

压缩可以分为无损压缩和有损压缩，有损，指的是压缩之后就无法完整还原原始信息，但是压缩率可以很高，主要应用于视频、话音等数据的压缩，因为损失了一点信息，人是很难察觉的，或者说，也没必要那么清晰照样可以看可以听；无损压缩则用于文件等等必须完整还原信息的场合，ZIP自然就是一种无损压缩，都是映射。



JAR签名是在apk文件中添加META-INF目录，即需要修改数据区、中央目录，因为添加文件后会导致中央目录大小和偏移量发生变化，还需要修改中央目录结尾记录。V2方案为加强数据完整性保证，不在数据区和中央目录中插入数据，选择在 数据区和中央目录 之间插入一个APK签名分块，从而保证了原始zip（apk）数据的完整性。





**V2签名的优势：**

1、只需进行一次hash校验，速度快；

不需要计算所有文件的摘要，以分块形式进行hash，支持并行计算。

2、除保证了APK内各文件的完整性，APK（zip包）中数据区、中央目录和中央目录结尾记录的完整性均得到了保证。

**多渠道打包**

方便后台通过大数据统计不同平台APK的表现状态。

在V2方案出现之前，快速批量打包方案有3类：

1、反编译APK后修改渠道值，再重新打包

这种方案实际上是重新签名，因有反编译、重新打包、签名的过程，速度相对后两种方案较慢；

2、将渠道信息以文件形式写入META-INF目录中(美团第一代多渠道打包工具)

因为META-INF目录是用来存放签名的，其本身无法加入签名校验中，在META-INF目录中添加文件不会破坏原有签名。此方案需同时修改zip数据区、中央目录和中央目录结尾记录；

3、将渠道信息写到zip中央目录结尾记录的comment字段中

通过前面分析zip文件结构，可以发现中央目录结尾记录最后注释字段，这部分内容在JAR签名方案中同样不在签名校验范围中，故添加注释也不会破坏原有签名。此方案只需修改中央目录结尾记录

在V2方案出现之后，因同时保证了数据区、中央目录和中央目录结尾记录的完整性，故方案2、3均不适用了。那是不是就没有快速批量打包的可能了呢？当然不是，可以从APK签名分块中着手。

APK签名分块中有一个ID-VALUE序列， 签名信息（APK 签名方案 v2 分块）只存储在ID 为 0x7109871a的ID-VALUE中，通过分析签名校验源码可以发现，其它ID-VALUE数据是未被解析的，也就是说除APK 签名方案 v2 分块外，其余ID-VALUE是不影响签名校验的。故可以定义一个新的ID-VALUE，将渠道信息写入APK签名分块中。因为V2方案只保证了第1、3、4部分和第 2 部分（APK签名分块）包含的APK 签名方案 v2分块中的 signed data 分块的完整性。新写入的ID-VALUE不受保护，所以此方案可行。实际上美团新一代渠道包生成工具Walle就是以这个方案实现的。

业务安全：

补充安全知识：