[1.环境搭建，开发相关 2](#_Toc28078643)

[1.1背景相关与系统架构分析 2](#_Toc28078644)

[1.1.1系统特性： 2](#_Toc28078645)

[1.1.2架构的简单理解： 2](#_Toc28078646)

[1.2 开发环境搭建 2](#_Toc28078647)

[1.2.1 环境搭建 2](#_Toc28078648)

[1.2.2相关术语的解析 2](#_Toc28078649)

[1.2.3 打包 2](#_Toc28078650)

[1.3 Git版本管理 2](#_Toc28078651)

[1.3.1 Git是什么 2](#_Toc28078652)

[1.3.2 创建本地仓库与代码托管 2](#_Toc28078653)

[1.4 界面原型设计 2](#_Toc28078654)

[1.5 工程相关解析 2](#_Toc28078655)

[1.5.1 res 2](#_Toc28078656)

[1.5.2 java 2](#_Toc28078657)

[1.6 Android程序签名打包 2](#_Toc28078658)

[1.6.1 什么是签名，有什么用 2](#_Toc28078659)

[1.6.2 如何签名 2](#_Toc28078660)

[1.7 反编译apk获得代码与资源 2](#_Toc28078661)

[2.常用UI控件与事件处理机制 2](#_Toc28078662)

[3.Android四大组件和Intent组件 2](#_Toc28078663)

[4.Fragment 2](#_Toc28078664)

[5.数据存储 2](#_Toc28078665)

[6.网络编程 2](#_Toc28078666)

[7.绘图与动画 2](#_Toc28078667)

[8.多媒体开发 2](#_Toc28078668)

[9.系统API，地图定位 2](#_Toc28078669)

[10.项目实战 2](#_Toc28078670)

# 1.环境搭建，开发相关

## 1.1背景相关与系统架构分析

我们可能有这样的一个疑问： 那么多的系统版本，我们开发的时候要针对哪个版本进行开发？ 这就是作为一个Android必须面对的Android的"碎片化"问题了，而这个问题又分为两个： ①系统碎片化：我们开发App时可能需要做到低版本兼容，比如，最低兼容至2.3版本；由于各种Rom定制的盛行，国人都喜欢对原生系统做一些更改，这导致了在原生系统上可行，而在定制Rom上不可行的问题，比如相机调用~ ②屏幕碎片化：市面上各种各样屏幕尺寸的手机，4.3寸，4.5寸，4.7寸，5.0寸，5.3寸...等等，除了手机外，还有Android平板，所以开发时我们可能要处理这个屏幕适配的问题，当然，刚学我们并不需要去考虑这些复杂的东西，后续实际开发我们再来深究！

### 1.1.1系统特性：

**1.** 应用程序框架支持组件的重用与替换（app发布时遵守了框架的约定，其他app也可以使用该模块）

**2.**Dalvik虚拟机:专门为移动设备优化 -集成的浏览器:开源的WebKit引擎

**3.**SQLite结构化的数据存储

**4.**优化的图形库,多媒体支持,GSM电话技术,蓝牙等

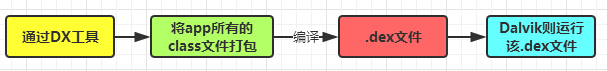
**5.**采用软件叠层方式构建

### 1.1.2架构的简单理解：

1.Application(应用程序层) 我们一般说的应用层的开发就是在这个层次上进行的，当然包括了系统内置的一组应用程序，使用的是Java语言

2.Application Framework(应用程序框架层) 无论系统内置或者我们自己编写的App，都需要使用到这层，比如我们想弄来电黑名单，自动挂断电话，我们就需要用到电话管理(TelephonyManager) 通过该层我们就可以很轻松的实现挂断操作，而不需要关心底层实现

3.Libraries(库) + Android Runtime(Android运行时) Android给我们提供了一组C/C++库，为平台的不同组件所使用，比如媒体框架；而Android Runtime则由Android核心库集 + Dalvik虚拟机构成，Dalvik虚拟机是针对移动设备的虚拟机，它的特点:不需要很快的CPU计算速度和大量的内存空间;而每个App都单独地运行在单独的Dalvik虚拟机内每个app对于一条Dalvik进程）而他的简单运行流程如：



4.Linux内核 这里就是涉及底层驱动的东西了，一些系统服务，比如安全性，内存管理以及进程管理等

## 1.2 开发环境搭建

### 1.2.1 环境搭建

略

### 1.2.2相关术语的解析

**1.**AVD：(android virtual machine):安卓虚拟设备,就是安卓的模拟器

**2.**ADT：(android development tools)安卓开发工具

**3.**SDK：(software development kit)软件开发工具包,就是安卓系统,平台架构等的工具集合

**4.**DDMS：(dalvik debug monitor service)安卓调试工具

**5.**adb：安卓调试桥,在sdk的platform-tools目录下,功能很多,命令行必备

**6.**DX工具：将.class转换成.dex文件

**7.**AAPT：(android asset packing tool),安卓资源打包工具

**8.**R.java文件：由aapt工具根据App中的资源文件自动生成,可以理解为资源字典

**9.**AndroidManifest.xml：app包名 + 组件声明 + 程序兼容的最低版本 + 所需权限等程序的配置文件

### 1.2.3 打包

略

## 1.3 Git版本管理

### 1.3.1 Git是什么

一个分布式版本控制系统，和SVN类似，但远比SVN强大的一个版本控制系统 ①Git可以方便的在本地进行版本管理，如同你本地有一个版本管理服务器一样 我们可以选择在合适的时间将本地版本推送到统一的版本管理服务器 ②Git每次会提取整个代码仓库的完整镜像，相当于对整个代码仓库都进行了一次备份，这样计时版本服务器除了问题，我们可以直接采用本地仓库恢复！结合本地版本管理功能，远程版本管理服务器出问题了，我们依然能继续写自己的代码，当他恢复的时候我们再提交我们的本地版本！ Git研发初期是为了更好的管理Linux内核，不过现在已经广泛应用于各种项目中！

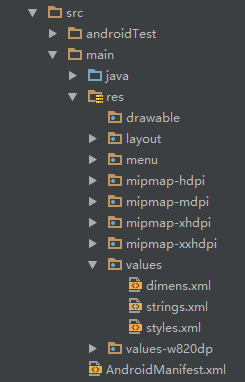
### 1.3.2 创建本地仓库与代码托管

略

## 1.4 界面原型设计

我们常说用户体验用户体验，用户使用你的软件，第一个会接触的是什么？没错，图形化界面(GUI)，简称UI，对于用户而言，最直观，给用户留下第一印像的是往往是程序的界面，而非功能！人，总喜欢美的东西，对吧？ 假如一样的功能，决定用户取向的，往往是UI！精美的UI！由此，一个产品的UI是非常重要的，而产品的界面原型设计一般是由公司的产品经理+美工来完成的，需求分析 ->界面原型设计 ->我们来写代码！可能你觉得界面原型对我们而言并不没什么作用，但假如你以后想自己开发App呢？又或者你升做产品经理呢？嘿嘿！世事无绝对，以后的事，谁知道呢？

## 1.5 工程相关解析



### 1.5.1 res

说到这个**res**目录，另外还有提下这个**assets**目录，虽然这里没有，但是我们可以自己创建，两者的区别在于是否前者下所有的资源文件都会在R.java文件下生成对应的资源id，而后者并不会；前者我们可以直接通过资源id访问到对应的资源；而后者则需要我们通过AssetManager以二进制流的形式来读取！对了，这个R文件可以理解为字典，res下每个资源都都会在这里生成一个唯一的id！存放我们各种资源文件的地方，有图片，字符串，动画，音频等，还有各种形式的XML文件

**1.图片**

**drawable**：存放各种位图文件，(.png，.jpg，.9png，.gif等)除此之外可能是一些其他的drawable类型的XML文件

**mipmap-hdpi**：高分辨率，一般我们把图片丢这里

**mipmap-mdpi**：中等分辨率，很少，除非兼容的的手机很旧

**mipmap-xhdpi**：超高分辨率，手机屏幕材质越来越好，以后估计会慢慢往这里过渡

**mipmap-xxhdpi**：超超高分辨率，这个在高端机上有所体现

**2.布局**

**layout**：该目录下存放的就是我们的布局文件，另外在一些特定的机型上，我们做屏幕适配，比如480\*320这样的手机，我们会另外创建一套布局，就行：layout-480x320这样的文件夹！

**3.菜单**

**menu**：在以前有物理菜单按钮，即menu键的手机上，用的较多，现在用的并不多，菜单项相关的资源xml可在这里编写，不知道谷歌会不会出新的东西来替代菜单了~

**4. values**

**demens.xml**：定义尺寸资源

**string.xml**：定义字符串资源

**styles.xml**：定义样式资源

**colors.xml**：定义颜色资源

**arrays.xml**：定义数组资源

**attrs.xml**：自定义控件时用的较多，自定义控件的属性！

theme主题文件，和styles很相似，但是会对整个应用中的Actvitiy或指定Activity起作用，一般是改变窗口外观的！可在Java代码中通过setTheme使用，或者在Androidmanifest.xml中为<application...>添加theme的属性！

PS:你可能看到过这样的values目录：values-w820dp，values-v11等，前者w代表平板设备，820dp代表屏幕宽度；而v11这样代表在API(11)，即android 3.0后才会用到的！

**5. raw**

用于存放各种原生资源(音频，视频，一些XML文件等)，我们可以通过openRawResource(int id)来获得资源的二进制流！其实和Assets差不多，不过这里面的资源会在R文件那里生成一个资源id而已

**6.动画**

属性动画和补间动画：

animator：存放属性动画的XML文件

**anim**：存放补间动画的XML文件

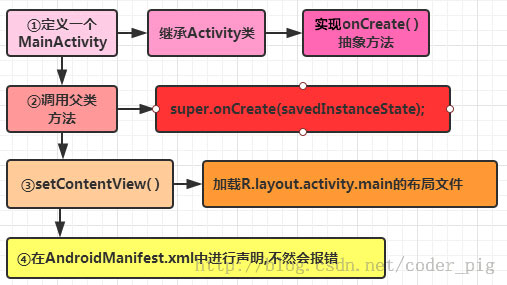
嗯，知道有什么资源，接下来就来了解该怎么用了： 前面也说了，我们所有的资源文件都会在R.java文件下生成一个资源id，我们可以通过这个资源id来完成资源的访问.

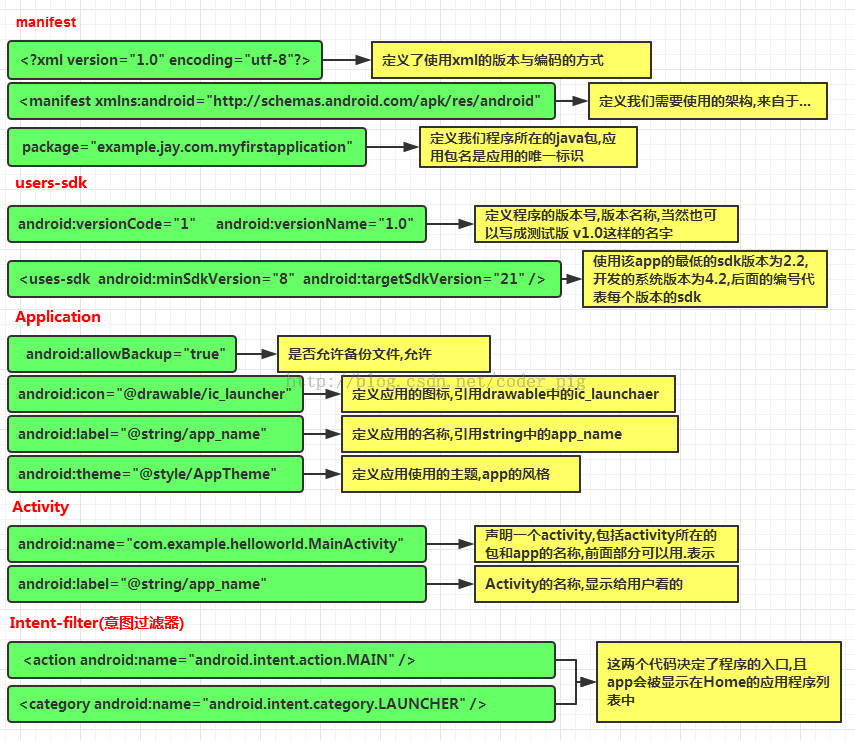
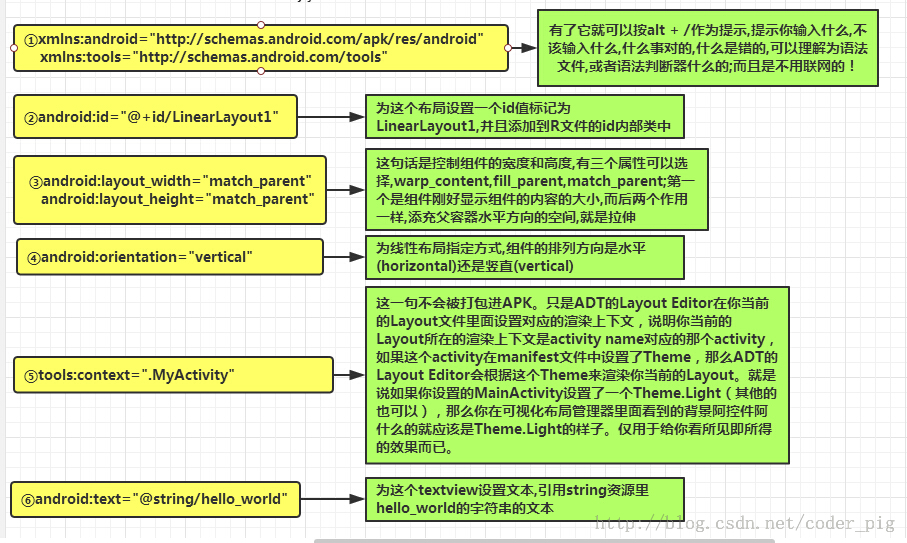
例如：R.string.app\_name

### 1.5.2 java

java：我们写Java代码的地方，业务功能都在这里实现

java文件与布局文件以及AndroidMainfest.xml的关系





## 1.6 Android程序签名打包

### 1.6.1 什么是签名，有什么用

1.应用程序升级：如果你希望用户无缝升级到新的版本，那么你必须用同一个证书进行签名。这是由于只有以同一个证书签名，系统才会允许安装升级的应用程序。如果你采用了不同的证书，那么系统会要求你的应用程序采用不同的包名称，在这种情况下相当于安装了一个全新的应用程序。如果想升级应用程序，签名证书要相同，包名称要相同！

2.应用程序模块化： Android系统可以允许同一个证书签名的多个应用程序在一个进程里运行，系统实际把他们作为一个单个的应用程序，此时就可以把我们的应用程序以模块的方式进行部署，而用户可以独立的升级其中的一个模块。

3.代码或者数据共享： Android提供了基于签名的权限机制，那么一个应用程序就可以为另一个以相同证书签名的应用程序公开自己的功能。以同一个证书对多个应用程序进行签名，利用基于签名的权限检查，你就可以在应用程序间以安全的方式共享代码和数据了。 不同的应用程序之间，想共享数据，或者共享代码，那么要让他们运行在同一个进程中，而且要让他们用相同的证书签名。 ————上述内容摘自:android 为什么需要签名

### 1.6.2 如何签名

略

## 1.7 反编译apk获得代码与资源

反编译Apk"，看上去好像好像很高端的样子，其实不然，就是通过某些反编译软件，对我们的APK进行反编译，从而获取程序的源代码，图片，XML资源等文件；不知道你有没有这样做过，看到一个别人的一个APP界面做得很精美，或者你看上别人的图片素材，简单点的，我们可以下载别人的APK，然后改下后缀名，改成xxx.zip，然后解压。但是，这种方法，获得的只会是一些.png，或者.jpg这样的位图文件资源，如果是xml类的资源，打开我们会发现是乱码，并且假如我们想看APK程序的Java代码，也是行不通的，因为他们都打被打包到classes.dex文件中！ 但是反编译可以解决你的需要~另外，切勿拿反编译来做违法的事，比如把人家的APK重新打包后使用自己的签名然后发布到相关市场...另外，我们是参考别人的代码，而不是完全拷贝！！！切记！！

这里推荐一个工具傻瓜化反编译：**Android killer**

# 2.常用UI控件与事件处理机制

如果你读到了这里说明你已经对安卓系统有一个整体的认识了，下面开始第二部分，也就是系统自带的UI。

# 3.Android四大组件和Intent组件

# 4.Fragment

# 5.数据存储

# 6.网络编程

# 7.绘图与动画

# 8.多媒体开发

# 9.系统API，地图定位

# 10.项目实战