封面

第一财经APP
项目重构建议

Nov 25, 2015

现在第一财经APP存在的问题（从技术的角度看）

1.整个项目结构不明朗；
2.整个项目没有按照各个模块独立出来；
3.整个项目对于常量的管理很混乱；
4.整个APP运行起来的时候内存没有优化好有卡顿的现象（偶现闪退）；
5.整个项目命名不规范；
6.在布局文件中直接写数字、字符串、color没有写到values中去；
7.动画卡顿，跳转衔接不流畅；
8.对于style的运用不到位；
9.APP中有些图标在高分辨率的手机上运行起来很模糊；
10.有些界面在低分辨率手机上显示不全；
11.APP布局重用不够，重复布局代码太多；
12.布局代码冗长过度重绘严重。

怎样的APP才是一款好的APP
从不同的角度判断一款好的APP
从用户的角度来说
能够解决用户实际问题；
产品结构简单，信息架构清晰，便捷容易上手；
交互优雅，有整体的风格；
核心功能突出，次要功能含蓄；
运行足够流畅，不会经常崩溃和访问速度慢；
有合理的提示以及过度动画；
产品有趣味性，互动性。

开发的角度来说
整个项目具有可扩展性的明朗的项目架构（项目架构的搭建）；
运行足够流畅，不会经常崩溃和访问速度慢（内存优化）；
适配不同的手机设备包括不同分辨率和不同内存大小的手机设备（APP的兼容性）；
有合理的提示以及过度动画（优秀的衔接动画）；
减少用户没必要的操作，替用户做某些事儿（保存用户的痕迹）。

项目架构的根本原则

项目架构的根本原则:**高内聚低耦合**

内聚：一个模块内各个元素彼此结合的紧密程度，高内聚就是一个模块内各个元素彼此结合的紧密程度高。
所谓高内聚是指一个软件模块是由相关性很强的代码组成，只负责一项任务，也就是常说的单一责任原则。

耦合：一个软件结构内不同模块之间互连程度的度量(耦合性也叫块间联系。指软件系统结构中各模块间相互联系紧密程度的一种度量。模块之间联系越紧密，其耦合性就越强，模块的独立性则越差，模块间耦合的高低取决于模块间接口的复杂性，调用的方式以及传递的信息。)

项目架构层级

项目分为了四个层级：

1.模型层、

模型层定义了所有的模型；

2.接口层、

接口层封装了服务器提供的API；

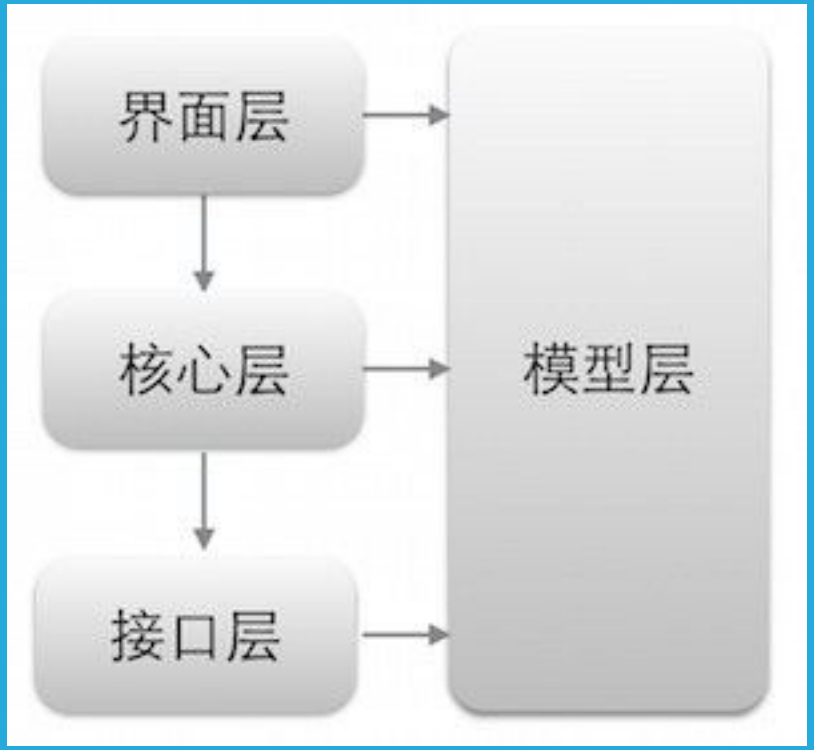
3.核心层、

核心层处理所有业务逻辑；

4.界面层。

界面层就处理界面的展示。

几个层级之间的关系如下图所示：



接口层

接口层封装了网络底层的API，并提供给核心层调用。

核心层

核心层介于接口层和界面层之间，主要处理业务逻辑，集中做数据处理。向上，给界面层提供数据处理的接口，称为Action；向下，调用接口层向服务器请求数据。

界面层

界面层处于最上层，其核心就是负责界面的展示。
因为公司有为不同商户定制不同app的需求，因此，这里就需要建立多个app的界面，这是一个很繁琐的事情。

模型层

模型层横跨所有层级，封装了所有数据实体类，基本上也是跟json的obj数据一致的，在接口层会将obj转化为相应的实体类，再通过Action传到界面层。

界面篇

将项目分为了四个层级：模型层、接口层、核心层、界面层。其中，最上层的界面，是变化最频繁的一个层面，也是最复杂最容易出问题的一个层面，如果规划不好，很容易做着做着，又乱成一团了。

要规划好界面层，至少应该遵循几条基本的原则：

1.保持规范性：定义好开发规范，包括书写规范、命名规范、注释规范等，并按照规范严格执行；

2.保持单一性：布局就只做布局，内容就只做内容，各自分离好；每个方法、每个类，也只做一件事情；

3.保持简洁性：保持代码和结构的简洁，每个方法，每个类，每个包，每个文件，都不要塞太多代码或资源，感觉多了就应该拆分（每个方法完成独立的任务；每个类都只代表独立的对象抽象；每个文件负责独立的模块）。

如何优化内存总结

可以从四个方面着手

首先是减小对象的内存占用，

其次是内存对象的重复利用，

然后是避免对象的内存泄露，

最后是内存使用策略优化。

减小对象的内存占用

1）使用更加轻量的数据结构

2）避免在Android里面使用Enum

3）减小Bitmap对象的内存占用

4）使用更小的图片

内存对象的重复利用

1）复用系统自带的资源

2）注意在ListView/GridView的优化

3）Bitmap对象的复用

避免对象的内存泄露

1）注意Activity的泄漏

2）考虑使用Application Context而不是Activity Context

3）注意临时Bitmap对象的及时回收

4）注意监听器的注销

5）注意缓存容器中的对象泄漏

6）注意WebView的泄漏

7）注意Cursor对象是否及时关闭

内存使用策略优化

1）谨慎使用large heap

2）综合考虑设备内存阈值与其他因素设计合适的缓存大小

3）onLowMemory()与onTrimMemory()

4）资源文件需要选择合适的文件夹进行存放

5）Try catch某些大内存分配的操作

6）谨慎使用static对象

7）特别留意单例对象中不合理的持有

8）珍惜Services资源

9）优化布局层次，减少内存消耗

10）谨慎使用“抽象”编程

……

Android屏幕适配

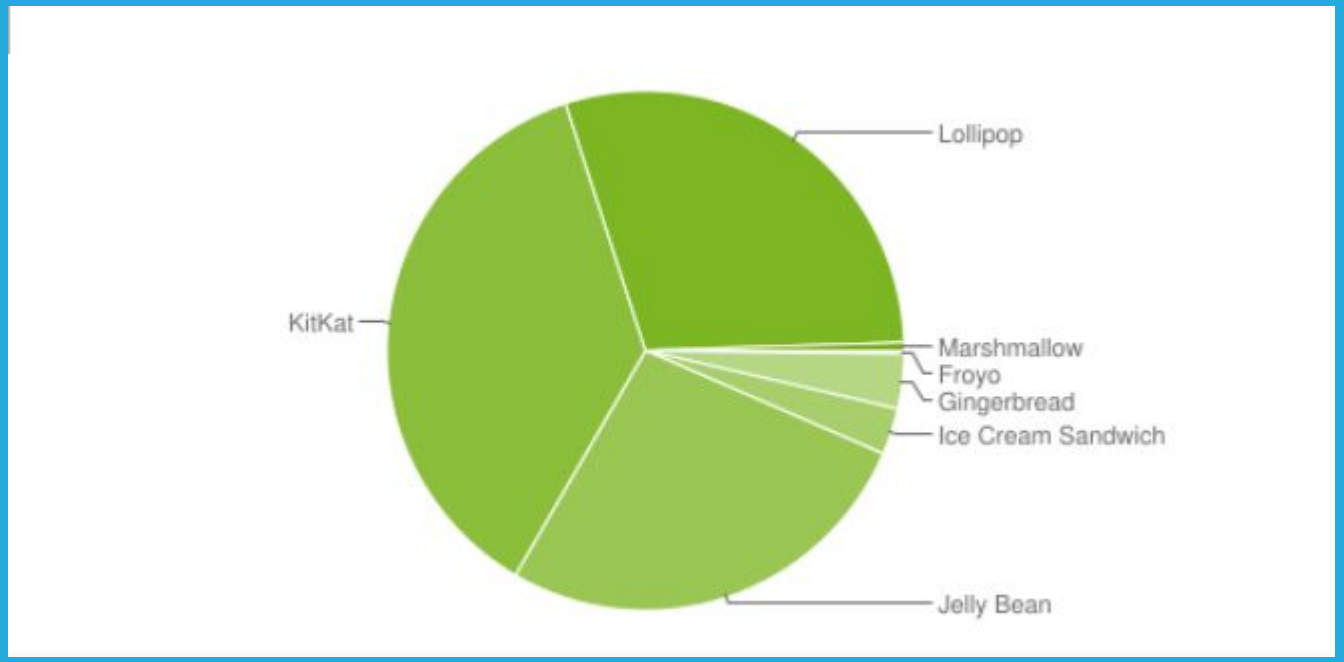
屏幕相关概念

屏幕像素px
屏幕大小size
屏幕密度dpi
屏幕方向
分辨率
独立密度像素DIP
DPI与DIP的换算
点9图片

为什么会出现Android屏幕适配问题

Android使用版本的多样化

2.2	Froyo	8	0.2%
2.3.3 - 2.3.7	Gingerbread	10	3.4%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.9%
4.1.x	Jelly Bean	16	10.0%
4.2.x		17	13.0%
4.3		18	3.9%
4.4	KitKat	19	36.6%
5.0	Lollipop	21	16.3%
5.1		22	13.2%
6.0	Marshmallow	23	0.5%



由以上图表分析得现在使用Android4.0以上的用户比重占到了将近96%

但是这96%的占有率并不集中在某一版本之上，而是分布于从api15到23的不同版本上这就造成了我们写Android应用是需要去适配不同版本，是其原因之一。

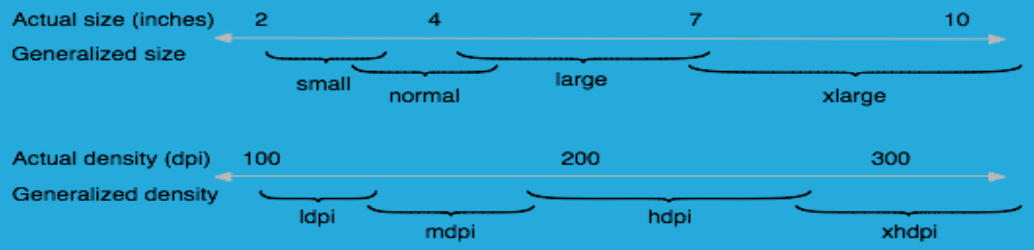
Android设备屏幕尺寸和屏幕密度的多样化

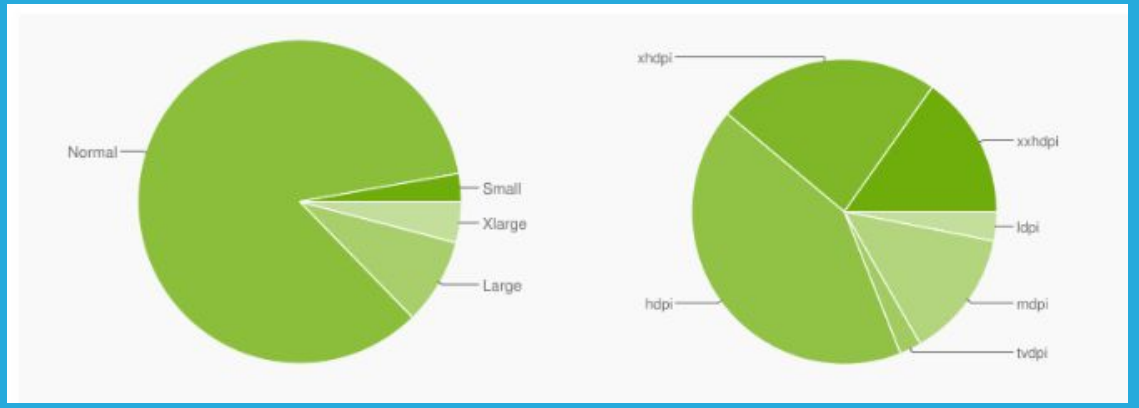
以下图表提供有关具不同屏幕大小和密度占有比例。
Android官方将Android设备的屏幕尺寸和屏幕密度人为的划分为不同的等级。
屏幕分辨率等级：

- xlarge screens are at least 960dp x 720dp
- large screens are at least 640dp x 480dp
- normal screens are at least 470dp x 320dp
- small screens are at least 426dp x 320dp

屏幕密度等级：

- ldpi (low) ~120dpi
- mdpi (medium) ~160dpi
- hdpi (high) ~240dpi
- xhdpi (extra-high) ~320dpi
- xxhdpi (extra-extra-high) ~480dpi
- xxxhdpi (extra-extra-extra-high) ~640dpi





	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	2.8%						**2.8%**
Normal		5.7%	0.1%	41.2%	22.3%	15.2%	**84.5%**
Large	0.3%	4.8%	2.2%	0.6%	0.7%		**8.6%**
Xlarge		3.1%		0.3%	0.7%		**4.1%**
Total	**3.1%**	**13.6%**	**2.3%**	**42.1%**	**23.7%**	**15.2%**	

由以上数据分析得屏幕分辨率在470dp x 320dp以上的设备占到了将近97%，屏幕密度在240dpi以上的设备占到了将近80%，但是他们同样分布不均，从而导致了我们的Android应用需要去适配不同的屏幕分辨率和不同的屏幕密度的设备

Android屏幕适配建议综合

屏幕适配的宗旨：

Android屏幕适配不只是让你的应用布局能正常显示，

而且还要给用户带来精妙，舒适的布局享受。

Android屏幕适配建议综合

屏幕适配建议

1.编写程序前应该首先定位运行的最低版本号，以及所支持的屏幕尺寸，以及是否支持竖屏，同时还要考虑到程序扩展性（特指对于以后更高分辨率手机出现后程序的适应性），同时需要该产品主体使用设备和设备版本。
2.在具体写代码时不要使用px单位去定义布局；
3.在具体写代码时使用dp单位去定义布局；
4.在具体写代码时使用sp单位去定义字体的大小；
5.建议在使用位图资源时首先考虑能否使用.9.png图片；
6.建议在使用位图资源时需要考虑是否需要制作多张图，如果制作一张图是否能支持大多数手机的显示；
7.尽量将点9图片放置在高像素密度资源文件夹中，这样即使在低像素密度手机上显示时会先对图片进行缩小再进行局部拉伸，但是在低像素密度手机上运行应用时，所有使用点9图片的地方都会对图片进行一次计算缩放，影响性能；
8.要理解drawable-ldpi、drawable-mdpi、drawable-hdpi、drawable-xhdpi、drawable-xxhdpi这几个图片资源的文件夹对应的像素密度为

drawable-ldpi	120dpi
drawable-mdpi	160dpi
drawable-hdpi	240dpi
drawable-xhdpi	320dpi

9.同时需要考虑如果只放置一张图片资源时尽量制作高分辨率的图片；

10.如果只放置一张图片资源时尽量制作高分辨率的图片的情况下需要考

虑在底屏幕密度的设备上是否能完美的显示；

11.要深入理解Android设备在加载图片资源时的顺序；

12.要防止图片放置在低密度文件夹中，而要在高像素密度设备上显示时，其会先进行放大，然后再显示，这样就会导致高像素密度设备上显示模糊这种情况的发生；

13.适配不同屏幕尺寸的解决办法建议多使用 "wrap_content" 和 "match_parent"；

14.建议多使用RelativeLayout（相对布局）；

15.对于特定屏幕尺寸建议使用尺寸标示符；

16.对于支持平板设备的APP布局建议使用最小宽度标识符；

17.对于支持平板设备的APP布局建议建议使用横竖标识符；

适配不同屏幕密度的解决办法建议提供多种可供选择的位图.

以上内容从项目架构，项目的优化以及适配三个方面对今后的开发做出了技术方面的建议。

一个优秀的app光从技术方面肯定是不够，它往往需要从产品到UI设计到美工到技术到后期的运营共同的协同作业。

一个优秀的app往往在很多的细节能打动用户（举例，微信的开机图；网易头条新闻）

提出建议
第一：希望能有主题色；
第二：设计图尽量规范；
第三：popupwindow toast 等对话框尽量精细；
第四：以后能否考虑个性化新闻推荐，收集用户行为，判断用户分类；
第五：加强与用户互动；
第六：注重用户数据的反馈。
第七：材料设计（Android 5.0最新）

谢谢观看