Android App设计

App设计 设计规范 MD设计

开发工具的选择

Android Studio是Google官方指定的Android开发工具，目前是1.3稳定版。Android Studio的优点就不需多说了，GitHub上大部分的Android开源库也都已迁移到Android Studio上来，在未提供jar文件时，使用Android Studio可以极为方便地集成开源库。最为重要的是Google已宣布将在年底前停止对Eclipse Android开发工具的一切支持（Google Ends Support for Android Eclipse Tools），因此请早日转移到Android Studio上来。不过也可以使用Eclipse+ADT来开发，毕竟部分公司还是用这个。

App设计风格

这一点对于一个开发者来说，貌似没有决定权，最终的决定权在产品部门手里。尽管这样，我还是会尽力说服产品部门将App设计成Material Design风格。这一点说多了都是泪啊，作为一个Android开发者，却整天开发着iOS风格的App，相信很多公司都这样，为了节省成本和时间，Android和iOS共用一套UI。举一个最常见的例子，Android App中每个页面TitleBar的左上角放一个返回按钮，这在iOS里是必须的，但Android有返回键啊，这样设计对于Android完全是多此一举。真心希望产品设计者尊重每种操作系统的风格及使用习惯，不要再设计不伦不类的产品。Material Design正好提供了一种这样的规范，自MD规范发布以来，其优雅的设计和清新的风格已吸引了大批设计者和开发者，如今MD设计不止在Android上（已有官方类库支持MD风格），甚至在CSS、HTML、JavaScript网页设计上都越来越火。因此，对于App的设计风格，Material Design当仁不让，也许你曾经错过了Android Design，请不要再错过Material Design。

相关的链接：

Material Design官网
Material Design配色模板
MD一个设计案例网站
MD风格的Andorid抽屉源码：Android-MaterialDesign-NavigationDrawer
MD风格的一个App源码（有妹子哦）：Android-MaterialDesign-DBMZ

版本支持

对于Android要支持的最低版本，可以参考各个版本的市场占有率，其实最靠谱的是根据自家App的统计数据来决定，目前我们的App最低支持2.2。以个人观点认为，虽然2.x的版本仍然有一部分用户，但其实手机更新换代特别快，为了更好的用户体验，也为了应用更新的API（很多第三方库也都有版本要求），应该提高最低支持的版本，大概3.0为宜，即API Level为11。

App框架设计

相信大家都有体会，随着功能模块的增加，App越来越大，如果没有良好的架构设计，则代码将会变得臃肿且不易维护，各功能模块的耦合度会越来越高。因此可以把App模块化，将一个完整的App划分成几个相对独立的模块，这样即可以降低模块间的耦合也利于复用。

1.网络模块

已经很少有单机版的App了吧，大部分都需要联网，从服务器请求数据，因此网络模模块必不可少。GitHub上的开源网络框架也特别多，个人认为可以使用开源框架，目前我会选`okHttp`或者`Volley`，也许以后会有更好的网络框架出现。注意如果使用开源框架，则必须要阅读其源码，必须能够驾驭它，这样就不至于当bug出现时束手无策。当然还可以自己写网络模块，目前我们的App网络模块就完全是自己写的，这样的好处是自己熟悉所写的代码，当有bug时可以迅速定位问题，同时注意处理一些联网过程中的细节，如：

（1）对HTTPS的支持、HTTPS证书的验证（目前很多做法都是默认允许所有HTTPS证书的，其实这样做是不安全的，应当真正地做证书校验）
（2）支持Wap方式上网，移动、联通、电信代理的设置
（3）支持重定向、数据压缩传输等
（4）其他值得注意的问题

自己写网络框架可以完美地处理这些细节，但时间成本比较大。如果使用开源框架，一般都没有处理这些细节，因此我们可以在第三方框架上做些修改，这样时间成本将会节省很多。

2.图片管理模块

图片也是App中不可少的元素，而且图片是占用内存的大户，因此图片管理框架特别重要，不好的图片框架容易引起内存泄露甚至导致崩溃。当然可以自己实现图片框架（目前我们也是这样做的），实现图片的下载、解码、缓存等关键环节。个人建议可以采用一些比较好的图片库，也许会比我们自己管理图片更完善和高效。我会推荐如下几个图片管理库：

`Glide`:Google的一些官方App，如Google photos都使用了，还要解释更多吗？

`Fresco`:FaceBook的开源库，功能超级强大，支持WebP、Gif、JPEG渐进显示，关键是其对图片内存的设计思想，使得图片内存开销大大减少。

`Android-Universal-Image-Loader`:在出现上述图片库之前，貌似这个最火吧，之前个人的App中也用了它。

`Picasso`:Square的开源库，据说Glide就是参考Picasso设计的。

3.本地数据库模块

也许你的App需要用到本地数据库，那么建议你采用流行的ORM框架，如`ActiveAndroid`或`greenDAO`，使用第三方库会大大方便你对sqlite的操作，个人认为在使用中我们需要注意数据库升级以及多线程并发操作数据库的问题。

4.文件管理模块

一个App，肯定会涉及到一些文件，如配置文件、图片、视频、音频、SharedPreferences文件等。我们可以提供一个全局的文件管理模块，负责文件的增、删、改、查等操作。另外还需支持文件压缩，文件的上传与下载操作，对于下载需要支持多线程并发下载、断点续传等功能。

5.组件内、组件间通信机制

对于一个App，组件通信必不可少，通信类型可以分为点对点和点对面的的通信，点对点即只有唯一的接收者可以响应消息，点对面则类似于消息广播，即所有注册过的都可以响应消息。在Android中，通常使用消息机制来实现，但消息机制的耦合度比较高。目前也有一些通信框架，如`EventBus`、`Otto`等事件总线框架，这些框架可以极大地降低组件间的耦合，但无法完美地实现点对点通信，因此建议消息机制和事件总线机制结合使用。

6.数据处理框架

其实还应该有一个数据处理框架，当发出数据请求后（走子线程），经网络模块返回数据（一般为JSON格式），JSON数据一般不能直接交给View层使用，需要解析成对应的Model，同时如有需要，还要缓存数据，因此这些流程可以抽象成一个数据处理的框架。这个框架可以认为接受数据请求的url，并将数据Model返回给Activity或Fragment。对于JSON数据解析，建议使用`fastjson`或者`GSON`，速度快且稳定，缺省值也比较完善。

7.线程调度模块

其实Android中有很多操作，如请求数据、下载图片、清除缓存等都是需要在子线程中执行的，往往很多时候都是直接起一个Thread来做了，这样做就会很乱而且线程多了将难以管理。因此可以抽象出一个线程调度模块，它维护一个线程池，如果有需要线程的话就通过线程调度模块取线程来做，这样就方便统一管理。当然第三方库中的线程操作我们将无法归到线程调度模块来管理，但其他涉及到线程的操作都应该来统一处理。

8.业务层

业务层大概就是四大组件、Fragment、View了，建议尽可能地使用原生组件，少用自定义组件，因为原生组件性能是最好的。另外建议使用`MVC`模式就好，只要设计管理好自己的逻辑，至于`MVP`、`MVVM`、`MVPR`等模式个人认为都有缺陷，总之寻求一个折中吧，有得必有失。

9.APK动态加载机制

随着App的增大，功能的扩展，很多App已经采用了APK动态加载的机制，也可以叫做插件化。由于本人没有在实际的App中应用过，所以不便发表过多评论。但这种机制个人认为很有前途，这种机制将利于App的解耦、功能扩展和局部升级。具体可以参考一个商用的解决方案：ApkPlug-移动应用模块化解决方案和一个开源的APK动态加载框架。

10.App的安全性考虑

Android App的安全问题很少有人重视，但这的确是一个很严重的问题，一些好的App经常被人破解。建议将一些核心算法等写成.so库，重要的逻辑放在服务器端，数据请求采用加密等，另外打包APK时至少要混淆代码，还可以采用APK加壳机制，总之这类的防范措施永远不嫌多。