App架构设计经验谈:展示层的设计

2016-03-20 安卓应用频道

(点击上方公众号，可快速关注)

> 作者：Keegan小钢
> 网址：http://keeganlee.me/post/architecture/20160222

系列文章：

App架构设计经验谈:接口的设计
App架构设计经验谈:技术选型
App架构设计经验谈:数据层的设计
App架构设计经验谈:业务层的设计

三层架构中，数据层和业务层都已经做过了简单的分享，最后，就剩下展示层了。本篇就给各位分享下我在展示层设计方面的一些经验心得。

展示层是三层架构中最复杂的一层了，需要考虑的包括但不限于界面布局、屏幕适配、文字大小、颜色、图片资源、提示信息、动画等等。展示层也是变化最频繁的一个层面，每天改得最多的就是界面了。因此，展示层也是最容易变得混乱不堪的一个层面。一个良好的展示层，应该有较好的可读性、健壮性、维护性、扩展性。

三原则

我在Android项目重构之路:界面篇中提到过三个原则，要设计好展示层，至少需要遵循好这三条基本的原则：

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。

关于这三个原则详细的解说，界面篇已经讲过的，我这里就不再重复。在此，我只做些补

充。

关于规范，Android方面，我已经分享过一套Android技术积累:开发规范，主要分为书写规范、命名规范、注释规范三部分。iOS方面，苹果已经有一套Coding Guidelines，主要属于命名方面的规范。当我们制定自己的开发规范时，首先就要遵守苹果的这份规范，在此基础上再加上自己的规范。

最重要的不是开发规范的制定，而是开发规范的执行。如果没有按照开发规范去执行，那开发规范就等于形同虚设，那代码混乱的问题依然得不到解决。

另外，Android系统本身已经对资源进行了很好的分离，字符串、颜色值、尺寸大小、图片、动画等等都用不同的xml文件定义。而iOS系统在这方面就逊色很多，只能自己实现，其中一种实现方案就是通过plist文件的方式实现和Android一样的机制。

工程结构

工程结构其实就是模块的划分，无非分为两类：按业务划分或按组件划分。
比如一个电商App，可能会有首页、附近、分类、我的四大模块，工程结构也根据这四大模块进行划分，Android可能就分为了四个模块包：

- com.domain.home 首页
- com.domain.nearby 附近
- com.domain.category 分类
- com.domain.user 我的

同样的，iOS则分为四个分组：home、nearby、category、user。

之后，每个模块下相应的页面就放入相应的模块。那么，问题来了，商品详情页应该属于哪个模块呢？首页会跳转到商品详情页，附近也会跳转到商品详情页，分类也会跳转到商品详情页，用户查看订单时也能跳转到商品详情页。有些页面，并不能很明显的区分出属于哪个模块的。我接手过的，按业务划分的二手项目中（即不是由我搭建的项目），我要找一个页面时，我认为应该属于A模块的，但在A模块却找不到，问了同事才知道在B模块。类似的情况出现过很多次，而且不止出现在我身上，对业务不熟悉的开发人员都会出现这个问题。而且，对业务不熟悉的开发人员开发新的页面或功能时，如果对业务理解不深，划分出错，那也将成为问题，其他人员要找该页面时更难找到了。

因此，我更喜欢按组件划分的工程结构，因为组件每个人都懂，不管对业务熟不熟悉，查找起来都明显方便很多。Android按组件划分大致如下：

- com.domain.activities 存放所有的Activity
- com.domain.fragments 存放所有的Fragment
- com.domain.adapters 存放所有的Adapter
- com.domain.services 存放所有的Service
- com.domain.views 存放所有的自定义View
- com.domain.utils 存放所有的工具类

iOS的分组则大致如下：

- controllers 存放所有ViewController
- cells 存放所有Cell，包括TableViewCell和CollectionViewCell
- views 存放所有自定义控件或对系统控件的扩展
- utils 存放所有的工具类

基类的定义

Android的Activity、Fragment、Adapter，iOS的ViewController，分别定义一个基类，将大部分通用的变量和方法定义和封装好，将减少很多工作量，而且有了统一的设置，也会减少代码的混乱。比如我在Android项目重构之路:实现篇中提到的KBaseActivity和KBaseAdapter的实现就是例子，当然还可以抽离出更多变量和方法。

每个Activity的onCreate()方法，一般分为三步：

1. 变量的初始化；
2. View的初始化；
3. 加载数据。

因此，其实可以将onCreate()方法拆分成三个方法：

1. initVariables()
2. initViews()
3. loadData()

在基类中将这三个方法定义为抽象方法，由子类去实现，这样，子类就不需要实现onCreate()方法了，只要实现更细化的上述三个方法即可。

iOS的ViewController也是同样的方式，这里就不重复了。

写在最后

自此，该系列的文章暂时就完结了，方法论比较多，很少涉及到具体的实现。因为具体实现的方案很多，而且还要结合实际项目，无法说哪个方案好哪个方案差。但方法论大部分是想通的，所以，本系列主要讲方法论。