优点：  
工具类：复用性强，大大减少了开发时间。  
：网络请求类（RequestModel）,网络请求使用了官方（volley），这也是一个不错的网络请求框架，请求中使用了post请求，安全性好。请求成功和失败回调接口都做了相应的封装，复用性高！  
：Toast：这是一个很好的提示框，给予用户很好的提示体验，给予用户简单明了的提示。在本项目中，封装好了一个自定义的toast方法，比默认弹出的效果好多了！  
：file工具类：文件的创建，sd卡的检测，缓存都做好了相应的封装！  
还有很多实惠的工具类，logger、ui、view。。。等等很多都值得去使用。  
在本项目中，自定义了一个MyApplication，里面封装了好多全局使用的方法。字段等等，在项目中的任何地方皆可以使用，这种思维方式值得学习！  
在项目中，创建了一个activity的基类，很多activity公用的功能什么的都做好了，只需继承它，许多功能就已经是实现了。  
功能性的接口都定义在一个xml文件中，扩展性强。假如某个接口变了或者需要新添加一些，直接去这个xml文件中去修改，并不需要去繁大的代码中寻找！  
适配性：总的来说还是不错的，各个不同分辨率的dp、px、drawable都在相应的xml文件中定义好了，由于Android手机的机型多，这种方法也不失为一个好的适配方式。  
内存优化性还好，比如一些游标、字段、handler的释放，图片的加载等，减少了可能出现的内存溢出，内存泄漏等。  
整体风格比较统一，不会让人有乱七八糟的感觉！  
缺点：  
布局的优化，在这方面并不好，布局的层叠性高，许多功能在布局中完成的，许多不需要嵌套的可以直接去掉。总的来说就是嵌套层次过深，可能会导致栈溢出，页面加载的时间过长。  
解决方式：（1）尽量使用merge标签，就是合并、融合的意思。使用它可以有效的将某些符合条件的多余的层级优化掉。  
viewstub：其实就是一个轻量级的页面，我们通常使用它来做预加载处理，来改善页面加载速度和提高流畅性，ViewStub本身不会占用层级，它最终会被它指定的层级取代。  
Include：制造这个标签纯碎是为了布局重用，在项目中通常会存在一些布局公用的部分，比如自定义标题啊等等。  
 2、许多可能为空的字段没有进行非空的判断，可能下标越界的没有做判断。这样最大的坏处就是会导致程序崩溃，在这一方面的需要重视起来，为了程序能够正常的运行起来，还是不要偷懒了，该写则写，无需犹豫！  
许多地方并没有注释，个人认为注释还是挺重要的，好的注释就是代表了应用的阅读性，别人看你的代码也方便，花费的时间也会变少，至少后期维护或者添加功能的时候，有了，一眼就能看出那个模块的功能，大大减少了开发周期，程序员的开发精力！  
Bug也算是多的，由于没有注释，花费的时间相对来说也是多一些，主要的一些逻辑的判断，代码的扩展性一般般  
Handler的使用较多，如果handler没有处理好，很容易就会导致内存泄漏，久而久之就会出现内存的溢出，程序的崩溃。