**WdCommon**

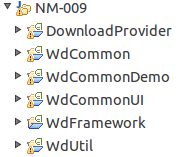
1. **框架简介**

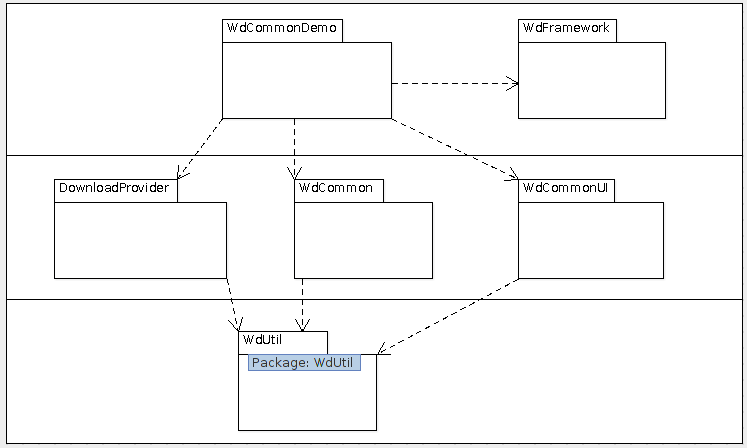
本文以NM-009下WdCommon为基础，目的是对以往项目中所用代码进行模块化整理，使得各模块之间耦合度尽量降到最低，同时又能在开始新项目时无需改动框架，仅需添加相关业务代码就能进行应用快速开发。

基于此，框架中将重要功能划分为以下几部分：

* 应用开发框架（各模块联动管理）；
* UI框架；
* 数据库管理模块；
* 网络数据交互模块；
* 异步加载图片功能；
* 缓存模块；
* 下载模块；
* 应用安装功能模块；
* util工具类；
  1. **library**

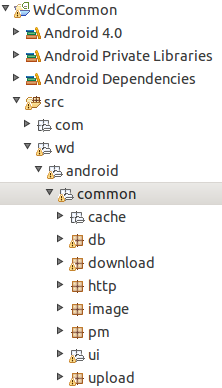
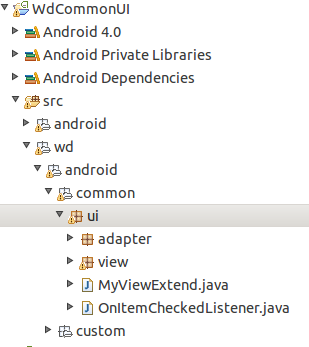
项目共分为以下几个library：

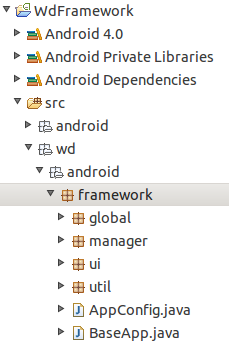


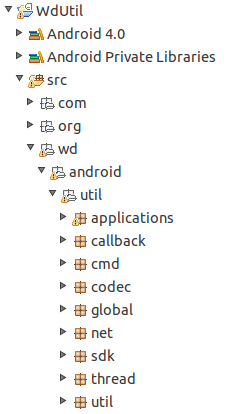
* WdUtil：基础util类，包括MyLog，IoUtil等工具类；
* WdCommonUI：自定义View；
* DownloadProvider：从系统源码中提取出的下载功能代码；
* WdCommon：包含数据库操作、网络数据交互模块、网络图片加载、缓存模块、应用安装卸载功能等；
* WdFramework：快速开发框架；
  1. 包结构图
  2. **包结构说明**

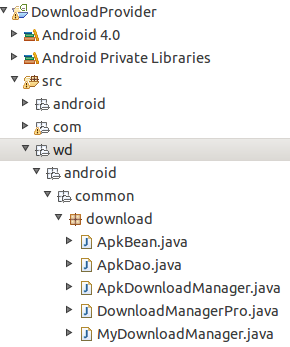
根据之前百视通多平台代码的经验，包结构不一致会导致代码同步工作很难进行，由于android应用中是以AndroidManifest.xml中所定义的包名为唯一标识，所以此处代码包名改为与项目无关的统一形式，主要分为wd.android.app、wd.android.custom、wd.android.common、wd.android.util、wd.android.common.ui、wd.android.framework。

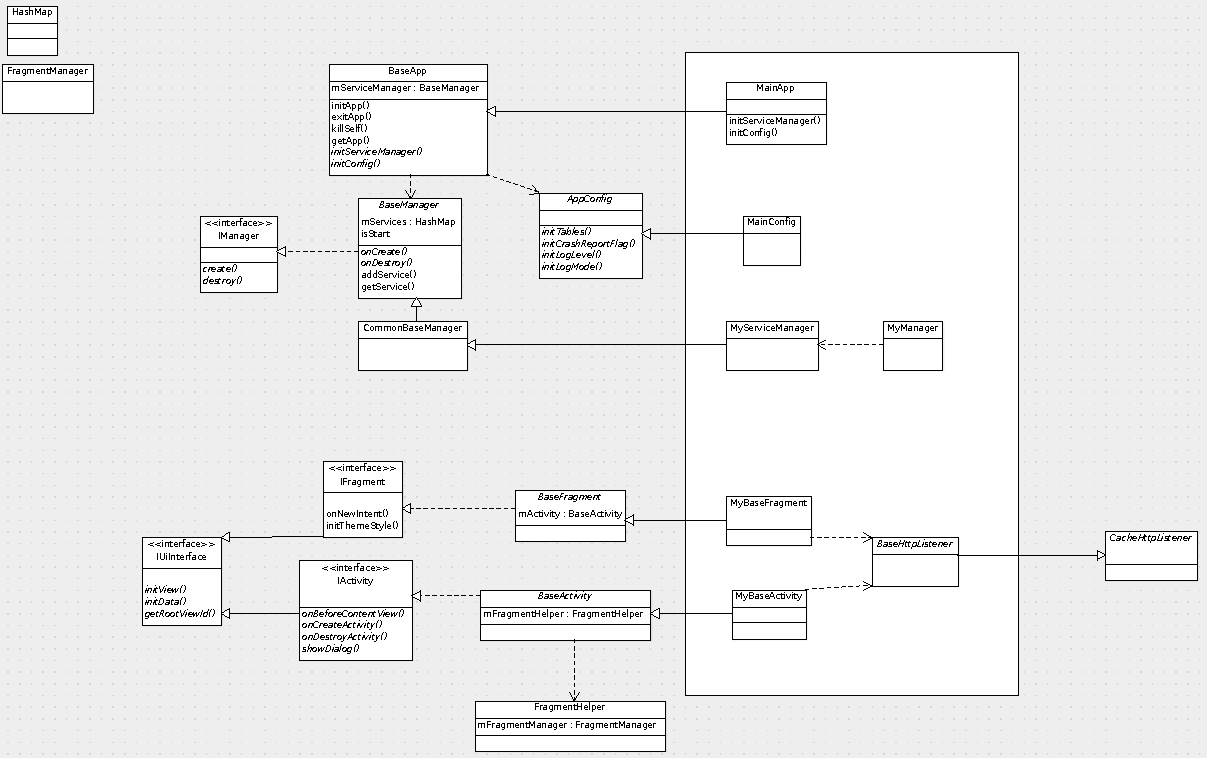
项目包结构截图如下：









1. WdFramework
   1. 类图
   2. 简介

WdFramework定义了包括BaseApp、BaseActivity、BaseFragment等应用开发过程中所需要的一些基础类。

此框架的优点如下：

* 其中BaseActivity与BaseFragment定义了统一的适配接口，这样子类在实现时就不用去关心activity和fragment的异同点；
* 引入AppConfig用于应用运行时参数配置；
* 实现Applicaiton，用以控制应用生命周期；
* 引入ServiceManager用于保存全局对象，减少单例或static属性泛滥导致的内存溢出问题；

如类图所示，IUIInterface定义了UI抽象接口，IFragment与IActivity中则加入了自有的生命周期接口。应用层开发时，实现自己的MyBaseFragment与MyBaseActivity。以BaseActivity为例，继承的子类需要实现initView、initData、getRootViewId方法，其中initView为初始化view，initData为初始化数据，getRootViewId为activity中传入的根layout。

IManager定义了模块管理接口，实现类BaseManager中用HashMap保存全局变量，应用层实现MyServiceManager，通过addService添加全局对象，而后可通过getService调用该对象。

* 1. BaseApp

采用Application保存全局变量，使其可以全局访问，并用来控制初始化与资源释放。代码中BaseApp继承自Application，主要代码为initApp()和exitApp()，以及静态方法getApp()。

在BaseApp中定义了BaseManager对象用来保存各个管理模块，由于BaseApp对象唯一，自然BaseManager对象也唯一，代码其他地方如果需要调用单例对象，只需要通过BaseManager获取到全局对象就可以直接使用了。

android不建议采用结束进程的方式退出应用，在不结束进程的场景下只需要释放BaseManager对象就可以释放资源。初始化时通过MyApp.getApp().initApp()调用，退出时调用MyApp.getApp().exitApp()。

实际开发时，在wd.android.custom包下实现子类MyApp继承BaseApp，并实现MyServiceManager继承BaseManager，实现MyConfig继承AppConfig。

* 1. BaseManager

BaseManager为统一管理模块，实现了流程相关的create和destroy方法，并定义了onCreate和onDestroy抽象方法。

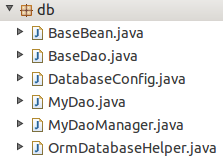
wd.android.framework.manager下还实现了CommonBaseManager，该类继承自BaseManager，定义了一些基础管理模块（ImageManager、MyDaoManager、EnvironmentInfo、GlobalData、MyPreference等）。并提供了相应的静态方法供外部直接调用，调用方式如下MyManager.getMyPreference()或者BaseManager.getService(MyPreference.class)。

实际开发时，在wd.android.custom包下定义自己的MyServiceManager继承自CommonBaseManager。

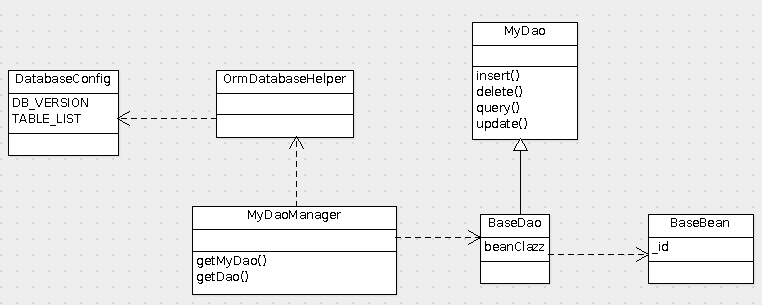
1. WdCommon
   1. 简介

WdCommon模块中集成了数据库管理模块、网络图片加载模块、网络数据交互模块、缓存模块、安装卸载模块等。

* 1. 数据库管理模块
     1. 包结构



* + 1. 类图



* + 1. 模块简介

实现子类集成自wd.android.common.db.BaseBean，每个字段中添加@DatabaseField标签，如果是数组则标签为@DatabaseField(dataType = DataType.BYTE\_ARRAY)。同时实现子类集成自wd.android.common.db.BaseDao。

如果有一些实体类（以DownloadItem为例）既需要存取数据库bean类，又需要保存其他不用持久化的数据，建议在在DownloadItem中定义DownloadBean对象。

原则上model类仅仅用于数据库存取，不允许其他多余代码。应用初始化时需要在自定义MainConfig中初始化表（DatabaseConfig.initTable()以及DatabaseConfig .initVersion()）。

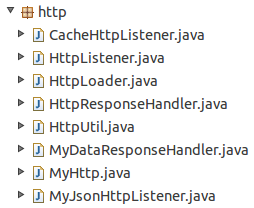
程序中调用时可通过MyDaoManager获取Dao对象，而后通过Dao对Bean进行增删查改操作。

* + 1. 其他说明

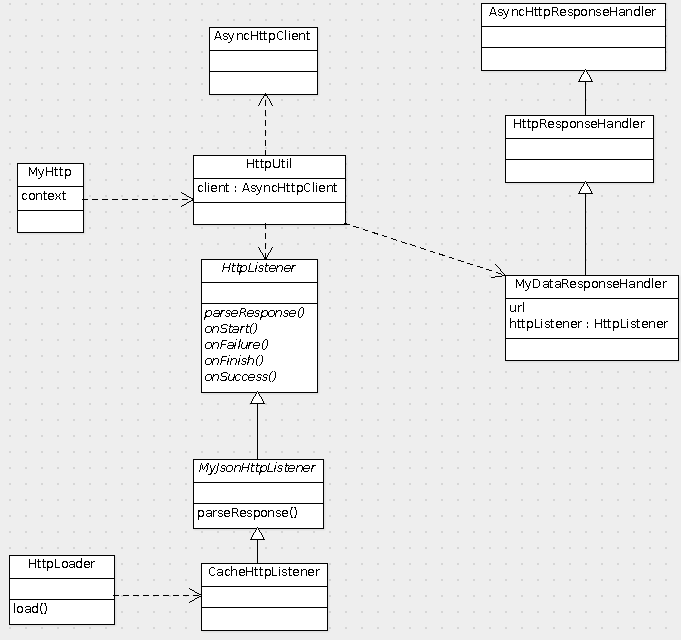
DatabaseConfig中DB\_VERSION为数据库版本。

代码中使用时，通过MyManager.getDao()获取对应的Dao接口。

* 1. 网络数据交互模块
     1. 包结构



* + 1. 类图



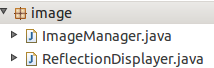
* + 1. 模块简介

在BaseActivity中创建MyHttp对象，传入Context值，请求数据时，调用MyHttp.exex()方法。

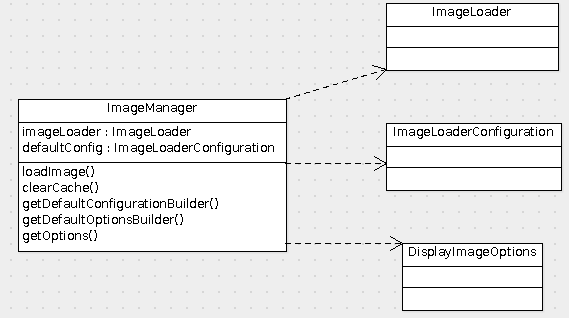
HttpUtil封装了get/post、setHeader等操作。

如果需要缓存，可以使用HttpLoader。

* 1. 网络图片加载
     1. 包结构



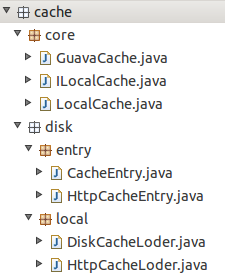
* + 1. 类图



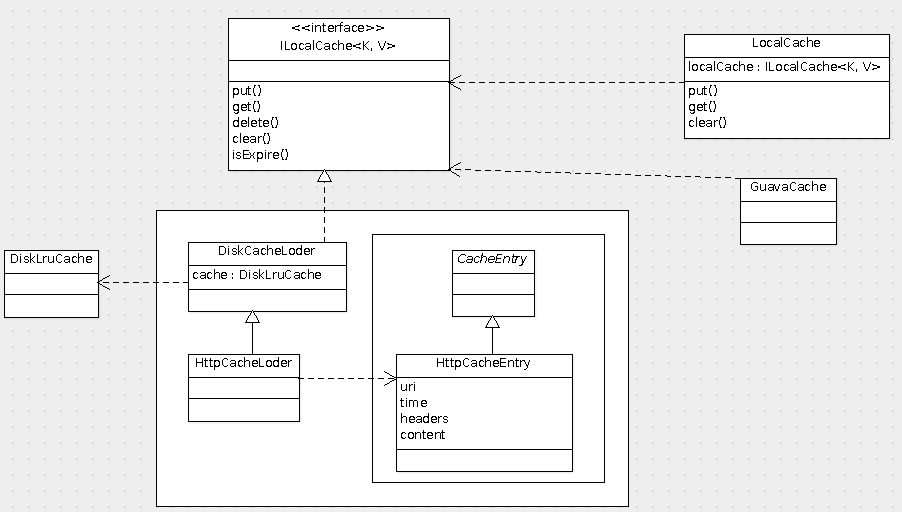
* + 1. 模块简介

调用ImageManager.load()加载网络图片，可以通过ImageLoaderConfiguration进行参数配置，DisplayImageOptions用以配置显示效果。

* 1. 缓存模块
     1. 包结构



* + 1. 类图

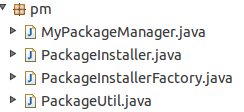


* + 1. 模块简介

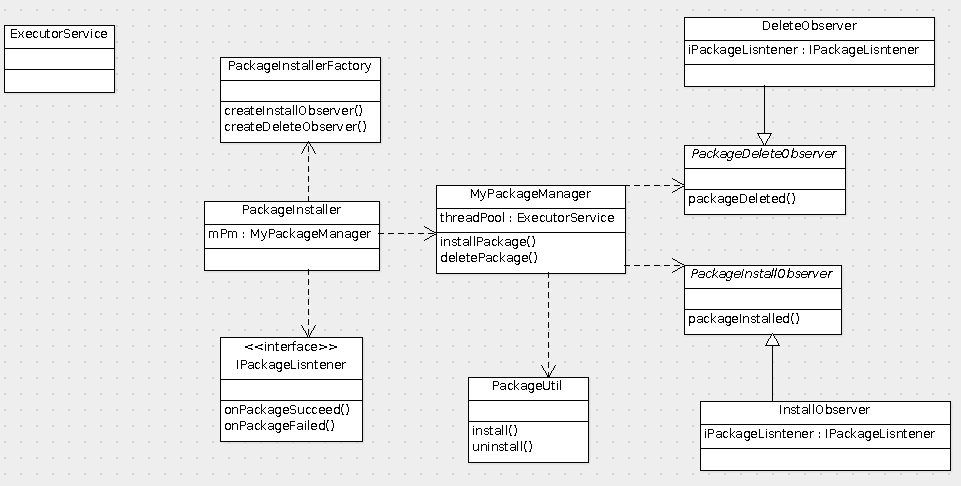
采用DiskLruCache作为缓存载体，亦可配置数据库等缓存方式。可自定义CacheEntry以及DiskCacheLoder用来缓存想要的数据（HttpCacheEntry用于缓存httpResponse数据）。LocalCache与GuavaCache为缓存操作入口，提供了常规的put、get、clear操作接口。

可自定义缓存失效时间。使用请参考HttpLoader。

* 1. 安装卸载模块
     1. 包结构



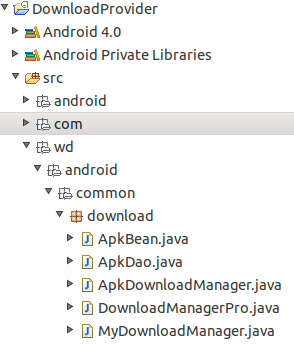
* + 1. 类图



* + 1. 模块简介

PackageUtil中定义了常规安装卸载以及静默安装卸载（需要获取root权限）操作。PackageInstallerFactory用于创建安装/卸载完成状态。应用中通过PackageInstaller对象调用其installPackage()与deletePackage()即可进行安装卸载功能。在能获取root权限时会默认进行静默安装卸载，获取不到则采用常规安装卸载方式。

* 1. 下载模块
     1. 包结构



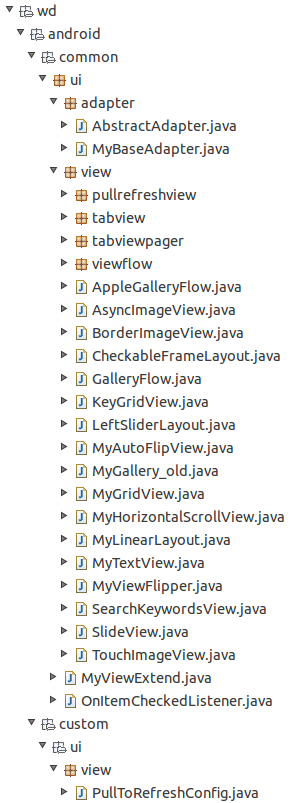
* + 1. 模块简介

基于android2.3源码提取出的下载代码。

MyDownloadManager提供了下载、暂停、删除等常规下载操作。

如果下载数据需要保存特定内容至数据库，以apk为例，自定义ApkDownloadManager继承自MyDownloadManager，实现ApkBean与ApkDao。

1. WdCommonUI
   * 1. 包结构

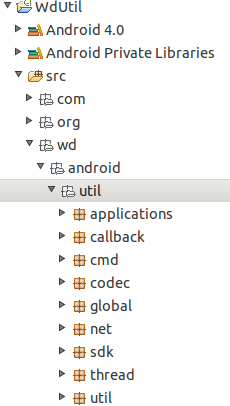


* + 1. 模块简介

包装了一些自定义view以及其他UI控件（可能会包含资源文件）。

adapter包下封装了继承自BaseAdapter的适配器基类，提供了view缓存机制。内部封装了用于显示的集合，以解决notifyDataChanged调用无效等问题。应用中使用时，实现自定义adapter继承MyBaseAdapter，实现updateView()、initViewHolder()、createRootView()等方法。

1. WdUtil
   * 1. 包结构



* + 1. 模块简介

封装了MyLog、IoUtil、Base64Util、ObjectUtil、Utils等util类。