# AndLang框架V1.0

方便Android开发者快速搭建项目框架

目的：开发过程中，经常View与Data混杂在一起，项目复杂化，代码量增加时，逻辑难以阅读，影响迭代开发；AndLang目的是将View与Data进行分开，这样开发者能够将更多的精力用在业务逻辑处理，提高开发效率，代码逻辑清晰。同时整合常用开发包到AndLang框架中，方便Android开发者快速搭建项目框架。

原理：运用java中的kvo模式(observer和observable),对model数据变化进行监控，控制view显示。

已整合开发包：butterknife (注入包)，glide(图片加载库，支持gif图)，okhttp(目前流行的网络请求库)，gson(gson数据处理库,使用新版本2.8.2)。

已整合的方法：Adapter和ViewHolder优化，PreferencesUtil(本地化存储工具)，

ActivityUtil（Activity管理工具），ButtonUtil（防button连点工具）,DateUtil（日期工具），FileUtil（针对android7.0以上文件访问位置工具），IDCardUtil（身份证号校验工具），ImageUtil（图片工具），RegexUtil（输入校验工具），

StatusBarUtils（状态栏自定义工具），ToastUtil（Toast显示工具）,

PermissionsCheckerUtil（Android6.0以上权限检测工具）,NetUtil(网络状态检测工具),DigitUtil(数字格式化工具)

模块及任务：一个Activity或Fragment，一个Presenter， 一个ViewModel。

Activity 和 Fragment 负责设置布局，View注入，事件注册，初始化Presenter，特殊View处理。

Presenter 负责创建ViewModel实例，响应事件处理，控制ViewModel字段间接控制View，网络数据请求，网络请求结果处理。

ViewModel 负责数据存放。

使用方法：

1.通过添加lib工程方式引入到项目中；

2.在项目的根gradle中添加

classpath 'com.neenbedankt.gradle.plugins:android-apt:1.8'

classpath 'com.jakewharton:butterknife-gradle-plugin:8.4.0'

ext{

isBuildModule=false;

}

3.使用butterknife注入时，需要在每个module中的gradle中进行配置引入；

4.创建Application,继承BaseLangApplication；

5.创建Model类继承BaseLangViewModel；

6.创建Presenter类继承BaseLangPresenter；

7.创建的Activity 继承 BaseLangActivity<对应Presenter类名>；使用WebView组件，需要继承BaseLangWebActivity;

8.说明Activity中方法执行顺序：

initView();

initPresenter();

initData();

已整合的View：

一.自定义标题栏:

1.AndroidManifest.xml中application标签样式设置成android:theme="@style/AppTheme.LangNoActionBar"

2.布局文件线性布局include lang\_common\_top.xml

<include layout="@layout/lang\_common\_top"/>

3.activity的initView方法中使用initTitleBar(false,"标题");标题栏颜色和状态栏颜色自行修改

二.自定义网络加载等待动画:

1.布局文件相对布局include lang\_common\_loading.xml

<include layout="@layout/lang\_common\_loading"/>

2.activity的initView方法中使用initLoading(); 动画可自行修改

三.webview加载整合：

1.无需创建布局文件，直接继承BaseLangWebActivity；

BaseLangWebActivity中自行修改webview配置

四.自定义Dialog

1.创建BaseLangDialog对象,实现BaseLangDialogInterface接口；

后续。。。

类说明：

BaseLangViewModel

|  |  |
| --- | --- |
| 方法名 | **说明** |
| notifyData(Object tag) | 通知观察者 Activity刷新UI，tag为标识字段， |
|  |  |

BaseLangPresenter

|  |  |
| --- | --- |
| 方法名 | **说明** |
| BaseLangPresenter(activity,modelclass) | 构造函数，modelclass 为BaseLangViewModel子类 |
| BaseLangPresenter(fragment,activity,modelclass) | 构造函数，modelclass 为BaseLangViewModel子类 |
| initModel | BaseLangPresenter的子类必须实现此方法，进行数据初始化，网络请求等操作 |

BaseLangUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| isEmpty(String str) | 判断字符串是否为空 |
| syncCookie(Context context,String url,String cookie) | 为某个域名 设置cookie |
| getUrlParma(String url, String key) | 获取url中的参数 |

VersionUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| getVersionName(Context context) | 获取版本名称 |
| getVersionCode(Context context) | 获取版本号 |
|  |  |

PreferencesUtil 使用BaseLangApplication的子类获取PreferenceUtil单例

Application.*getInstance*().getSpUtil()

|  |  |
| --- | --- |
| 方法名 | **说明** |
| putInt(Context context, String name, int value) | 存储int数据 |
| putString(Context context, String name, String value) | 存储string数据 |
| putBoolean(Context context,String name, Boolean value) | 存储boolean数据 |
| putLong(Context context, String name, Long value) | 存储long数据 |
| putFloat(Context context , String name,Float value) | 存储float数据 |
| getInt(Context context, String name) | 获取int数据 |
| getString(Context context, String name) | 获取string数据 |
| getBoolean(Context context,String name) | 获取boolean数据 |
| getLong(Context context, String name) | 获取long数据 |
| getFloat(Context context , String name) | 获取float数据 |
|  |  |

ActivityUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| getInstance | 获取工具单例 |
| pushOneActivity(Activity actvity) | 把一个activity压入管理栈中  BaseLangActivity中oncreate已调用 |
| getLastActivity() | 获取管理栈中顶部的activity |
| finishActivity(Class<?> cls) | 结束指定类的activity |
| popOneActivity(Activity activity) | 把一个activity从管理栈中移除  BaseLangActivity中ondestory已调用 |
| exitApplication() | 将管理栈中所有activity全部finish，并退出app |
| getActivitySize() | 获取管理栈的activity数量 |
| start(Activity activity, Intent i) | 从右到左动画启动activity |
| start2(Activity activity, Intent i) | 淡入启动activity |
| startResult(Activity activity, Intent i, int requestCode) | startForResult方式 从右到左动画启动activity |
| exit(Activity activity) | 关闭软键盘，从右退出activity |
| exitResult(Activity activity, Intent i, int resultCode) | setResult方式 从右退出activity |

ButtonUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| isFastDoubleClick(int btn) | btn标识符 ，间隔3s |
| isFastDoubleClick2() | 间隔3s |
| isFastDoubleClick3() | 间隔5s |

DateUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| datetime(String format) | 获得指定格式的日期时间字符串 |
| String timeStamp2Date(String seconds, String format) | 时间戳转换成日期格式字符串 |
| ... | ... |

FileUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| file2Uri(Context ctx, File file) | 根据文件转换成对应的Uri |
| getFileFromUri(Context ctx, Uri uri) | 根据Uri转换成对应的文件 |
|  |  |

IDCardUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| IDCardValidate(String IDStr) | 检验身份证号是否有效 |
|  |  |
|  |  |

ImageUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| calculateInSampleSize(BitmapFactory.Options options,int reqWidth, int reqHeight) | 计算图片的缩放值 |
| readPictureDegree(String path) | 获取图片角度 |
| rotateBitmap(Bitmap bitmap, int degress) | 旋转图片角度 |
| getFileimg(Uri uri, Activity activity) | 获取路径为uri的文件 |
| zoomImg(String img, int newWidth, int newHeight) | img 为图片路径 缩放图片 |
| zoomImg(Bitmap bm, int newWidth, int newHeight) | 缩放图片 |
| addWatermark(Bitmap src, Bitmap watermark) | 添加水印 |
| saveBitmap(Bitmap bitmap, String parentDir, String fileName) | 将内存中的图片保存到本地 |
| saveBitmap(Bitmap bitmap, File imageFile) | 将内存中的图片保存到本地 |
| getUri(android.content.Intent intent, ContentResolver cr) | 解决小米手机上获取图片路径为null的情况 |
| getViewBitmap(View addViewContent) | View转换成图片bitmap |
| compressImage(String filePath, String targetPath, int quality) | 将图片从filePath路径复制到targetPath路径，图片质量为quality |
| getSmallBitmap(String filePath,int wight,int height) | 根据路径获得图片信息并按比例压缩 |
|  |  |
|  |  |

RegexUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| checkTSZF(String str) | 是否存在特殊字符 |
| checkNumber(String str) | 是否是纯数字 |
| checkWord(String str) | 是否是字母 |
| boolean isPhoneNumber(String str) | 判断是不是手机号 |

StatusBarUtils

|  |  |
| --- | --- |
| 方法名 | **说明** |
| setWindowStatusBarColor(Activity activity, int colorResId) | 修改状态栏颜色 |
| setTextColorStatusBar(Activity activity, boolean isDark) | 状态栏字体颜色 isDark=true 黑色，isDark=false 白色 |
| FlymeSetStatusBarLightMode(Window window, boolean dark) | 设置状态栏图标为深色和魅族特定的文字风格 |
| MIUISetStatusBarLightMode(Window window, boolean dark) | 设置状态栏字体图标为深色，需要MIUIV6以上 |
| getStatusHeight(Context context) | 获取状态栏高度 默认4.4以下返回状态栏高度为0 |
| translateStatusBar(Activity activity) | 状态栏透明 |
|  |  |

ToastUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| show(Context context, String info) | 显示toast |
| … | ... |

PermissionsCheckerUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| lacksPermissions(Context context,String... permissions) | permissions 权限列表，需要申请的权限请自行百度 google |
| … | ... |

NetUtil

|  |  |
| --- | --- |
| 方法名 | **说明** |
| isNetworkAvailable(Activity activity) | 判断当前网络状态是否为连接状态  BaseLangActivity中onresume已调用 |
| … | ... |