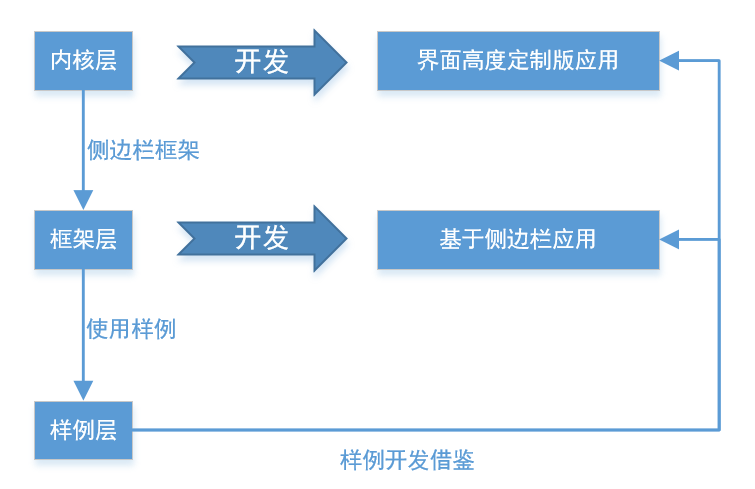
KP模板维护文档

1. KP模板层次说明

KP模板总共分为三层，用git进行了统一的管理，三层结构之间的关系如图一所示：

1. 内核层（git中是master分支）。该层主要使用了kernel（内核）库，该库主要实现了一些基于databinding、kotlin封装的MVP框架、支持JNI、支持realm数据库、支持文字图标、KP PLC读写封装、实现状态界面界面等最基础内容。（所有需要高度定制的KP应用都是基于该层开发）
2. 框架层（git中是side分支）。该层在内核层的基础上增加了侧边栏的封装，通过配置路由的方式快速在侧边栏添加子功能界面。添加应用奔溃自启机制。（所有可以复用侧边栏的项目都可以在该层基础上开发）
3. 样例层（git中的demo分支）。该层主要展现所有实现的单一功能，方便有相关功能需求时能够在样例的基础上开始集成开发。（该层只做样例展示，所有开发项目都不基于该层开发）



图一 结构关系图

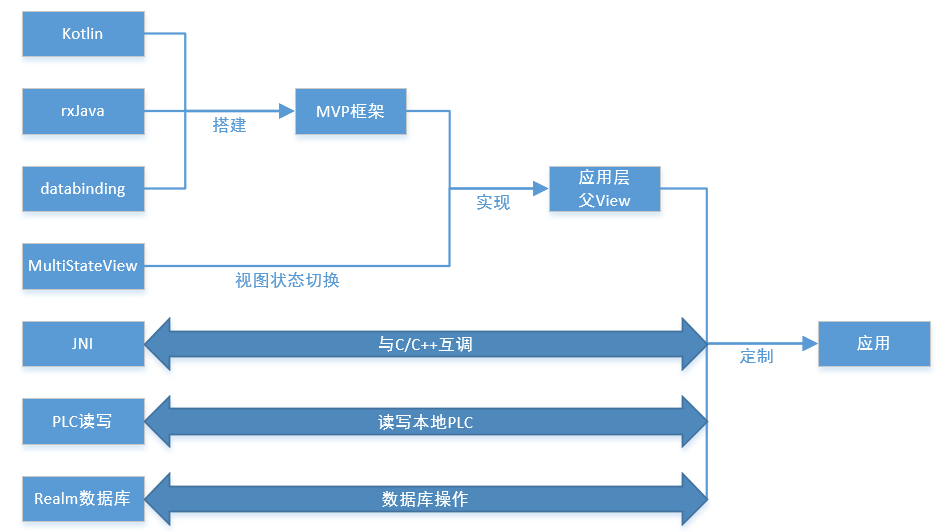
1. 内核层维护说明

该层处理的是架构层、与KP硬件密切相关、向后兼容和优化相关内容。内核层整体关系图如图二所示：

1. 该层处理了databinding、kotlin相结合的MVP模式。MVP的处理在base文件夹里面。其中BaseBindingActivity是对Activity的MVP架构组合；BaseBindingAdapter是对RecyclerView Adapter的MVP架构组合，这其中实现了一项Item的点击响应监听；BaseBindingFragment是对Fragment的MVP架构组合。这三个类中有用到一个PresenterFactory类，该类实现的功能是Presenter与Activity、Fragment绑定。

说明：另外需要特别说明的是View接口的实现是集成自IView；各功能Presenter都必须集成BasePresenter来实现相关逻辑代码。default里的EmptyPresenter是在较简单不需要使用MVP模式的界面使用。

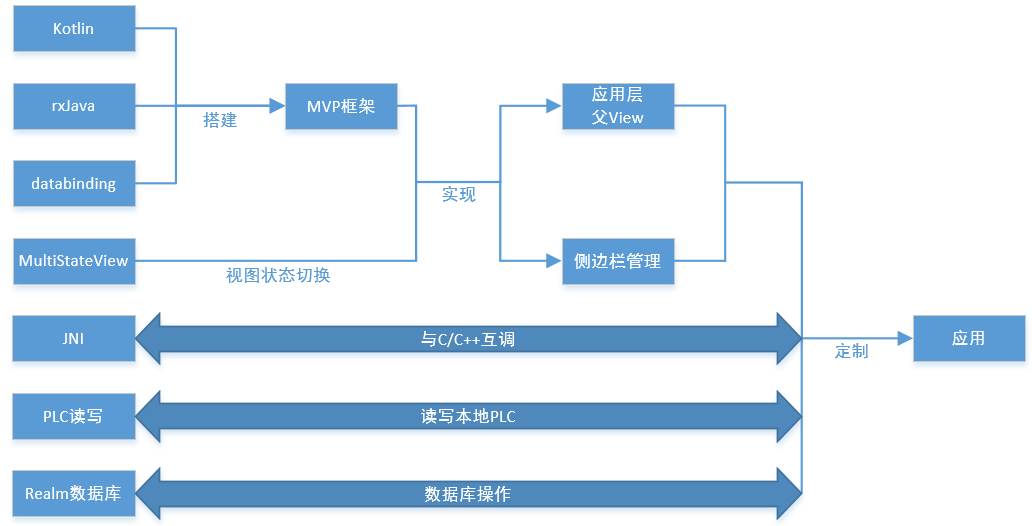
1. Kotlin支持。支持Kotlin编写Android，一是简化Android编程，用以提高开发效率；二是利用kotlin与JS的兼容方便后续对网页的支持。
2. Fragment生命周期细化。base里的BaseLazyFragment细化fragment的每个生命周期，包括首次显示、首次隐藏、再次显示、再次隐藏等。
3. Rxjava生命周期管理。在最底层实现Actvity和Fragment对RxLifecycle的支持，以便实现Rxjava生命周期的便捷管理。
4. 该层封装了界面改变相关方法。在所有需要使用时实现该接口的相关方法就可以通过view对界面状态进行控制。实现如demo中base里的IStateView。
5. KP读写PLC封装。localHttp中实现的是读写KP自带的PLC元件。
6. Realm数据库引入和基本方法封装。Utils里的RealmManager是对Realm数据列库的单例封装，封装了最基本的增、删、查、更新相关操作。
7. 文字图标支持。widget里的IconFontTextView是对文字图标的支持，并支持在布局时动态配置文字字体（ttf字体库）。
8. 不滑动ViewPager。widget里的NoScrollViewPager实现不滑动的ViewPager。
9. JNI支持。在内核层支持JNI，用以来支持C/C++。
10. KP屏幕亮度配置。在KpApplication配置是否常亮，在BaseAcitity来实现KP屏幕常亮。另外KpApplication还提供整个应用生命周期的Contex以及reaml的初始化，应用开发可以通过继承initRealm这个方法来改变Realm数据库的基本配置。



图二 内核层结构关系图

1. 框架层维护说明

该层主要是封装了侧边栏框架。通过在router 文件夹中的SideItemRouter里配置Fragment，来快速的把Fragment嵌入到侧边栏里，用以提高开发速度。框架层整体关系如图三所示。

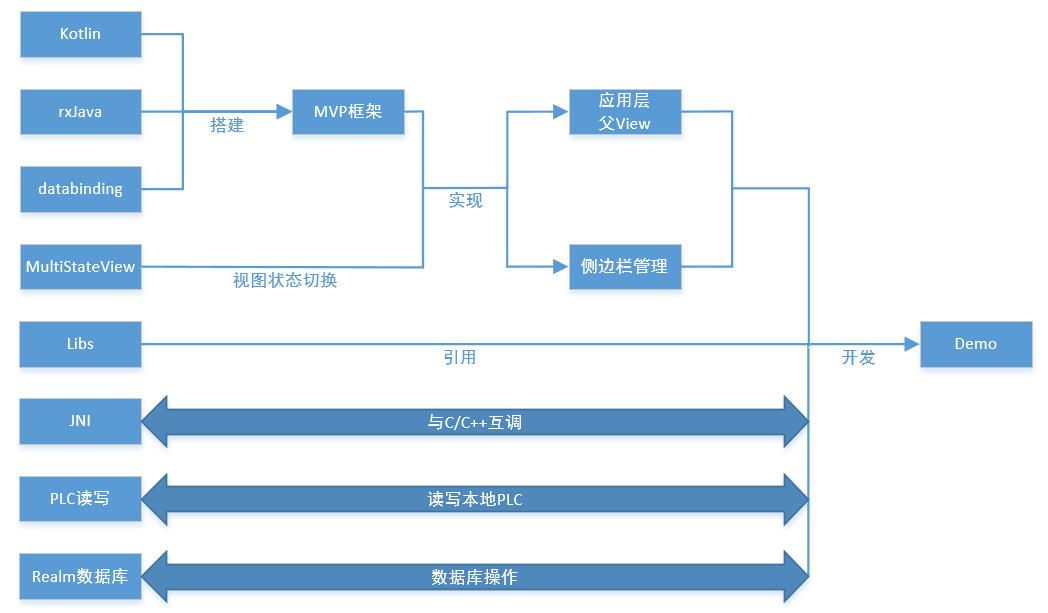


图三 框架层结构关系图

1. 样例层维护说明

样例层和框架层的处理逻辑是一样的，区别在于侧边栏，框架层侧边栏匹配的是一个Fragment枚举列表，而样例层匹配的则是一个枚举类（如：DataItemRouter），该枚举类是一个Fragment枚举列表，另外需要注意侧边栏每增加一项都需要在MainPresenter类中进行一项匹配。样例层整体关系如图四所示。

该层主要是一些使用样例，具体见《KP开发文档》。



图四 样例层结构关系图