Contents

[Install 2](#_Toc5298905)

[git 2](#_Toc5298906)

[git bash 2](#_Toc5298907)

[从github获取 3](#_Toc5298908)

[更新至github 3](#_Toc5298909)

[Toast 3](#_Toc5298910)

[default 3](#_Toc5298911)

[toast with position 3](#_Toc5298912)

[toast with picture 4](#_Toc5298913)

[user defined 4](#_Toc5298914)

[TabLayout 5](#_Toc5298915)

[Fragment 5](#_Toc5298916)

[Fragment与Activity之间的通讯 6](#_Toc5298917)

[fragment + TabLayout + ViewPager 6](#_Toc5298918)

[ViewPager + ListFragmetn 6](#_Toc5298919)

[Fonts 8](#_Toc5298920)

[https://www.cnblogs.com/plokmju/p/7608025.html 8](#_Toc5298921)

[替换单个控件字体 8](#_Toc5298922)

[修改app缺省字体 9](#_Toc5298923)

[BottomNavigationView 9](#_Toc5298924)

[RecyclerView 9](#_Toc5298925)

[Fragment中添加RecyclerView 9](#_Toc5298926)

[ListFragment 9](#_Toc5298927)

[Android Studio 10](#_Toc5298928)

[增加构造函数 10](#_Toc5298929)

[Intent 10](#_Toc5298930)

[显式Intent 10](#_Toc5298931)

[隐式Intent 10](#_Toc5298932)

[Activity的生命周期 11](#_Toc5298933)

[目录结构 11](#_Toc5298934)

[Android知识 12](#_Toc5298935)

[New()和newInstance()的区别 12](#_Toc5298936)

[This,super 12](#_Toc5298937)

[this是自身的一个对象，代表对象本身，可以理解为：指向对象本身的一个指针。 12](#_Toc5298938)

[super可以理解为是指向自己超（父）类对象的一个指针，而这个超类指的是离自己最近的一个父类。 12](#_Toc5298939)

[ADB 13](#_Toc5298940)

[Adb devices 13](#_Toc5298941)

<https://developer.android.com/guide>

https://baijiahao.baidu.com/s?id=1581222739674858335&wfr=spider&for=pc

# Install

Android SDK download link:

<http://developer.android.com/sdk/index.html>

# git

## git bash

cd c:\android\fittingChart

git init

git remote add origin <https://github.com/zxswzj/fittingChart>

# 从github获取

git stash

git reset --hard FETCH\_HEAD

git pull origin master -f

## 更新至github

Git add .

Git commit

Git push origin master

# Toast

## default

Toast toast=Toast.makeText(getApplicationContext(), "默认的Toast", Toast.LENGTH\_SHORT);

toast.show();

## toast with position

Toast toast=Toast.makeText(getApplicationContext(), "自定义显示位置的Toast", Toast.LENGTH\_SHORT);

//第一个参数：设置toast在屏幕中显示的位置。我现在的设置是居中靠顶

//第二个参数：相对于第一个参数设置toast位置的横向X轴的偏移量，正数向右偏移，负数向左偏移

//第三个参数：同的第二个参数道理一样

//如果你设置的偏移量超过了屏幕的范围，toast将在屏幕内靠近超出的那个边界显示

toast.setGravity(Gravity.TOP|Gravity.CENTER, -50, 100);

//屏幕居中显示，X轴和Y轴偏移量都是0

//toast.setGravity(Gravity.CENTER, 0, 0);

toast.show();

## toast with picture

Toast toast=Toast.makeText(getApplicationContext(), "显示带图片的toast", 3000);

toast.setGravity(Gravity.CENTER, 0, 0);

//创建图片视图对象

ImageView imageView= new ImageView(getApplicationContext());

//设置图片

imageView.setImageResource(R.drawable.ic\_launcher);

//获得toast的布局

LinearLayout toastView = (LinearLayout) toast.getView();

//设置此布局为横向的

toastView.setOrientation(LinearLayout.HORIZONTAL);

//将ImageView在加入到此布局中的第一个位置

toastView.addView(imageView, 0);

toast.show();

## user defined

//Inflater意思是充气

//LayoutInflater这个类用来实例化XML文件到其相应的视图对象的布局

LayoutInflater inflater = getLayoutInflater();

//通过制定XML文件及布局ID来填充一个视图对象

View layout = inflater.inflate(R.layout.custom2,(ViewGroup)findViewById(R.id.llToast));

ImageView image = (ImageView) layout.findViewById(R.id.tvImageToast);

//设置布局中图片视图中图片

image.setImageResource(R.drawable.ic\_launcher);

TextView title = (TextView) layout.findViewById(R.id.tvTitleToast);

//设置标题

title.setText("标题栏");

TextView text = (TextView) layout.findViewById(R.id.tvTextToast);

//设置内容

text.setText("完全自定义Toast");

Toast toast= new Toast(getApplicationContext());

toast.setGravity(Gravity.CENTER , 0, 0);

toast.setDuration(Toast.LENGTH\_LONG);

toast.setView(layout);

toast.show();

# TabLayout

|  |
| --- |
| tabLayout = view.findViewById |
| viewPager = view.findViewById |
| fragments.add(new BlankFragment()); |
| adapter = new MyFragmentPagerAdapter(getActivity().getSupportFragmentManager()); |
| adapter.addTitlesAndFragments(mTitles, fragments); |
| viewPager.setAdapter(adapter); |
| tabLayout.setupWithViewPager(viewPager); |
|  |

# Fragment

<http://hukai.me/android-training-course-in-chinese/basics/fragments/index.html>

<https://developer.android.com/guide/components/fragments>

<https://www.cnblogs.com/hixin/p/4427276.html>

## Fragment与Activity之间的通讯

因为所有的Fragment都是依附于Activity的，所以通信起来并不复杂，大概归纳为：

**a、如果你Activity中包含自己管理的Fragment的引用，可以通过引用直接访问所有的Fragment的public方法**

**b、如果Activity中未保存任何Fragment的引用，那么没关系，每个Fragment都有一个唯一的TAG或者ID,可以通过getFragmentManager.findFragmentByTag()获得任何Fragment实例，然后进行操作。**

**c、在Fragment中可以通过getActivity得到当前绑定的Activity的实例，然后进行操作。**

https://www.jianshu.com/p/6e51d40f0ad1

## fragment + TabLayout + ViewPager

<https://blog.csdn.net/qq_34773981/article/details/82022647>

## ViewPager + ListFragmetn

https://stackoverflow.com/questions/23718217/listfragment-with-pageradapter

1. 创建存储多个Fragment实例的列表
2. 创建PagerAdapter实例并关联到Viewpager中
3. 将ViewPager关联到Tablayout中
4. 根据需求改写Tablayout属性\*
5. TabLayer所在的Activit或者Fragment对应的xml文件中，添加TabLayout和ViewPage

<android.support.design.widget.TabLayout  
 android:id="@+id/tl\_tabs"  
 android:layout\_width="match\_parent"  
 android:layout\_height="40dp" />  
  
<android.support.v4.view.ViewPager  
 android:id="@+id/vp\_content"  
 android:layout\_width="match\_parent"  
 android:layout\_height="232dp" />

1. TabLayer所在的Activit或者Fragment中
   1. TabLayout和ViewPager的定义。

//TabLayout  
TabLayout tabLayout;  
ViewPager viewPager;  
List<Fragment> fragments = new ArrayList<>();  
List<String> titles = new ArrayList<>();

* 1. 添加TabLayout

tabLayout = view.findViewById(R.id.*tl\_tabs*);  
viewPager = view.findViewById(R.id.*vp\_content*);  
  
fragments.add(TabFragment.*newInstance*("a","b"));  
fragments.add(TabFragment.*newInstance*("c","b"));  
fragments.add(TabFragment.*newInstance*("d","b"));  
fragments.add(TabFragment.*newInstance*("e","b"));  
fragments.add(TabFragment.*newInstance*("f","b"));  
titles.add("head");  
titles.add("breast");  
titles.add("back");  
titles.add("shoulder");  
titles.add("arm");

* 1. ViewPage关联到TabLayout

viewPager.setAdapter(new FragmentStatePagerAdapter(getActivity().getSupportFragmentManager()) {  
 @Override  
 public Fragment getItem(int position) {  
 return fragments.get(position);  
 }  
  
 @Override  
 public int getCount() {  
 return fragments.size();  
 }  
  
 @Override  
 public void destroyItem(ViewGroup container, int position, Object object) {  
 super.destroyItem(container, position, object);

}  
  
 @Nullable  
 @Override  
 public CharSequence getPageTitle(int position) {  
  
 return titles.get(position);  
 }  
});  
  
tabLayout.setupWithViewPager(viewPager);

2019-03-20 14:19:43.030 9433-9433/com.example.fittingChart I/TabLayout: FittingFragment.FragmentActivity

2019-03-20 14:19:52.758 9433-9433/com.example.fittingChart I/TabLayout: FittingFragment.onAttach

2019-03-20 14:19:52.758 9433-9433/com.example.fittingChart I/TabLayout: FittingFragment.onCreate

I/TabLayout: FittingFragment.onCreateView

/TabLayout: FittingFragment.onActivityCreated

I/TabLayout: FittingFragment.onStart

I/TabLayout: FittingFragment.onResume

I/TabLayout: FittingFragment.onTabUnselected

I/TabLayout: FittingFragment.onTabSelected

I/TabLayout: FittingFragment.onPause

I/TabLayout: FittingFragment.onStop

I/TabLayout: FittingFragment.onStart

I/TabLayout: FittingFragment.onResume

I/TabLayout: FittingFragment.onPause

I/TabLayout: FittingFragment.onStop

I/TabLayout: FittingFragment.onDestroyView

I/TabLayout: FittingFragment.onDetach

I/TabLayout: FittingFragment.onAttach

I/TabLayout: FittingFragment.onCreate

I/TabLayout: FittingFragment.onCreateView

I/TabLayout: FittingFragment.onActivityCreated

I/TabLayout: FittingFragment.onStart

I/TabLayout: FittingFragment.onResume

。。。

# Fonts

https://fonts.google.com/

https://www.1001freefonts.com/

## <https://www.cnblogs.com/plokmju/p/7608025.html>

## 替换单个控件字体

Add ttf file in assets/fonts目录

TextView textView = (TextView) findViewById(R.id.*tv1*);  
Typeface typeface = Typeface.*createFromAsset*(getAssets(), "fonts/mangobrother.ttf");  
textView.setTypeface(typeface);

## 修改app缺省字体

# BottomNavigationView

|  |
| --- |
| BottomNavigationView navigation = findViewByID |
| BottomNavigationView.OnNavigationItemSelectedListener listener |
| navigation.setOnNavigationItemSelectedListener(listener); |
|  |
| FragmentTransaction ft = getSupportFragmentManager().beginTransaction(); |
| ft.replace(R.id.*fragment\_container*, activityFragment); |
| ft.commit(); |
|  |

# RecyclerView

<https://github.com/xinzhazha/RecyclerView>

<https://developer.android.com/guide/topics/ui/layout/recyclerview#java>

## Fragment中添加RecyclerView

<https://www.jianshu.com/p/1103d7022ea2>

<https://www.jianshu.com/p/7f79b08f5afa>

<https://github.com/astuetz/PagerSlidingTabStrip>

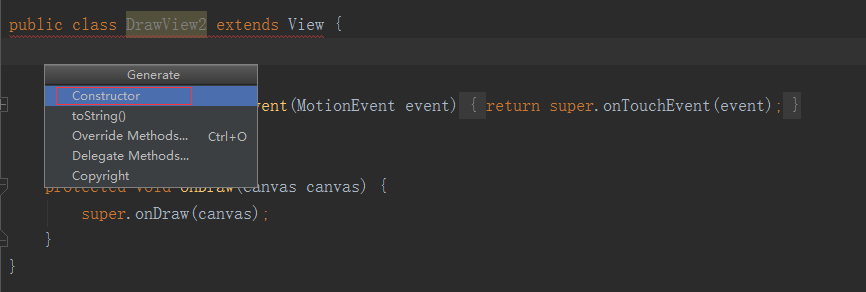
## ListFragment

https://github.com/XAVlER-S/MagicHeaderViewPager

# Android Studio

## 增加构造函数

将光标放在类里，按【Alt】+【Insert】键，插入构造函数。



# Intent

## 显式Intent

AndroidMenifest.xml add

<activity android:name=”. SecondActivity”></activity>

Intent intent = new Intent(FirstActivity.this, SecondActivity.class);

startActivity(intent);

## 隐式Intent

AndroidMenifest.xml add

<activity android:name=”. SecondActivity”></activity>

<intent.filter>

<action android:name=”com.example.activitytest.ACTION\_START” />

<category android:name=”android.intent.category.DEFAULT” />

<category android:name=” com.example.activitytest.MY\_CATEGORY” />

</intent.filter>

</activity>

Intent intent = new Intent(“com.example.activitytest.ACTION\_START”);

Intent.addCategory(“com.example.activitytest.MY\_CATEGORY”);

StartActivity(intent);

# Activity的生命周期

onSaveInstanceState在activity被回收之前被调用，可以通过Bundle来临时保存数据，在onCreate可以导入临时保存的数据

if(savedInstanceState != null) {

String str = saveInstanceState.getString(“data\_key”);

…

}

# 目录结构

* .gradle, .idea, build
  + 存放Android Studio自动生成的一些文件，无须处理
* .gitignore
  + 用来指定目录或者文件排除在版本控制之外
* Settings.gradle
  + 用于指定项目中所指定的模块
* App.libs
  + 如果项目中使用了第三方的jar包，需要将jar包放到该目录下，放在该目录下的jar包都会被自动添加到构建目录中去
* App.androidTest
  + 编写android的测试用例
* App.test
  + 编写unit test的测试用例
* App.AndroidManifest.xml
  + <activity  
     android:name=".SplashActivity"  
     android:label="@string/app\_name">  
     <intent-filter>  
     <action android:name="android.intent.action.MAIN" />  
      
     <category android:name="android.intent.category.LAUNCHER" />  
     </intent-filter>  
    </activity>
    - 表示SplashActivity是这个项目的主活动，在手机上点击应用图标，首先启动的就是这个活动
  + android:label="@string/app\_name">
    - 指定活动中标题栏的内容，标题栏是显示在活动最顶部。
* Build.gradle

repositories {

google()  
 jcenter()  
}

* + - 声明代码仓库，可以在项目中引用任何jcenter上的开源项目
* dependencies {  
   classpath 'com.android.tools.build:gradle:3.3.2'

}

* + - 声明使用该gradle插件来构建项目
* implementation 'com.android.support:appcompat-v7:28.0.0'
  + 远程依赖库，

# Android知识

## New()和newInstance()的区别

**newInstance()就是把new这个方式分解为两步，首先调用class的加载方法加载某个类，然后实例化。这样的好处是**[**显而易见**](https://www.baidu.com/s?wd=%E6%98%BE%E8%80%8C%E6%98%93%E8%A7%81&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)**的。**

**1、我们可以在调用class的静态方法forName()时获得更好的灵活性**

**2、提供给了我们降耦的手段。**

## ****This,super****

this是自身的一个对象，代表对象本身，可以理解为：指向对象本身的一个指针。

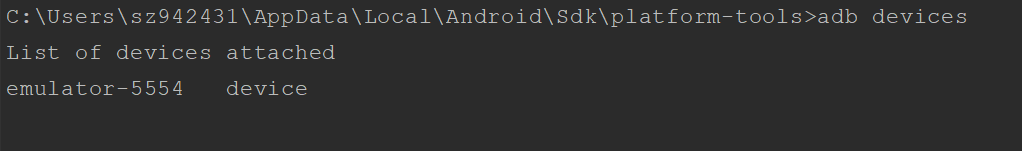
super可以理解为是指向自己超（父）类对象的一个指针，而这个超类指的是离自己最近的一个父类。

# ADB

目录：您可以在 **android\_sdk**/platform-tools/ 中找到 adb 工具

## Adb devices

列出当前连接的所有设备



要*从*模拟器或设备复制文件或目录（及其子目录），请使用

adb pull ***remote*** ***local***

要将文件文件或目录（及其子目录）复制*到*模拟器或设备，请使用

adb push ***local*** ***remote***

在上述命令中，***local*** 和 ***remote*** 指的是开发计算机（本地）和模拟器/设备实例（远程）上目标文件/目录的路径。例如：

adb push foo.txt /sdcard/foo.txt

您可以使用 shell 命令通过 adb 发出设备命令，可以进入或不进入模拟器/设备实例上的 adb 远程 shell

通过adb shell直接删除程序



Install superSU <http://os-android.liqucn.com/rj/76323.shtml>

# SQLite3

delete from fitting\_show where id=1;

MVP design pattern is a set of guidelines that if followed, decouples the code for reusability and testability. It divides the application components based on its role, called separation of concern.**MVP divides the application into three basic components:**

1. **Model**: It is responsible for handling the data part of the application.
2. **View**: It is responsible for laying out the views with specific data on the screen.
3. **Presenter**: It is a bridge that connects a Model and a View. It also acts as an instructor to the View.

**MVP lays few ground rules for the above mentioned components, as listed below:**

1. A View’s sole responsibility is to draw UI as instructed by the Presenter. It is a dumb part of the application.
2. View delegates all the user interactions to its Presenter.
3. The View never communicates with Model directly.
4. The Presenter is responsible for delegating View’s requirements to Model and instructing View with actions for specific events.
5. Model is responsible for fetching the data from server, database and file system.

*The above-mentioned principles can be implemented in a number of ways. Each developer will have its own vision of it. But in the nutshell, basic nuts and bolts are common with minor modification.*

*With great power, comes great responsibility.*

**Now, I lay down the preamble, I follow for MVP.**

1. Activity, Fragment, and a CustomView act as the View part of the application.
2. Each View has a Presenter in a one-to-one relationship.
3. View communicates with its Presenter through an interface and vice versa.
4. The Model is broken into few parts: ApiHelper, PreferenceHelper, DatabaseHelper, and FileHelper. These are all helpers to a DataManager, which in essence binds all Model parts.
5. Presenter communicates with the DataManager through an interface.
6. DataManager only serves when asked.
7. Presenter does not have access to any of the Android’s apis.

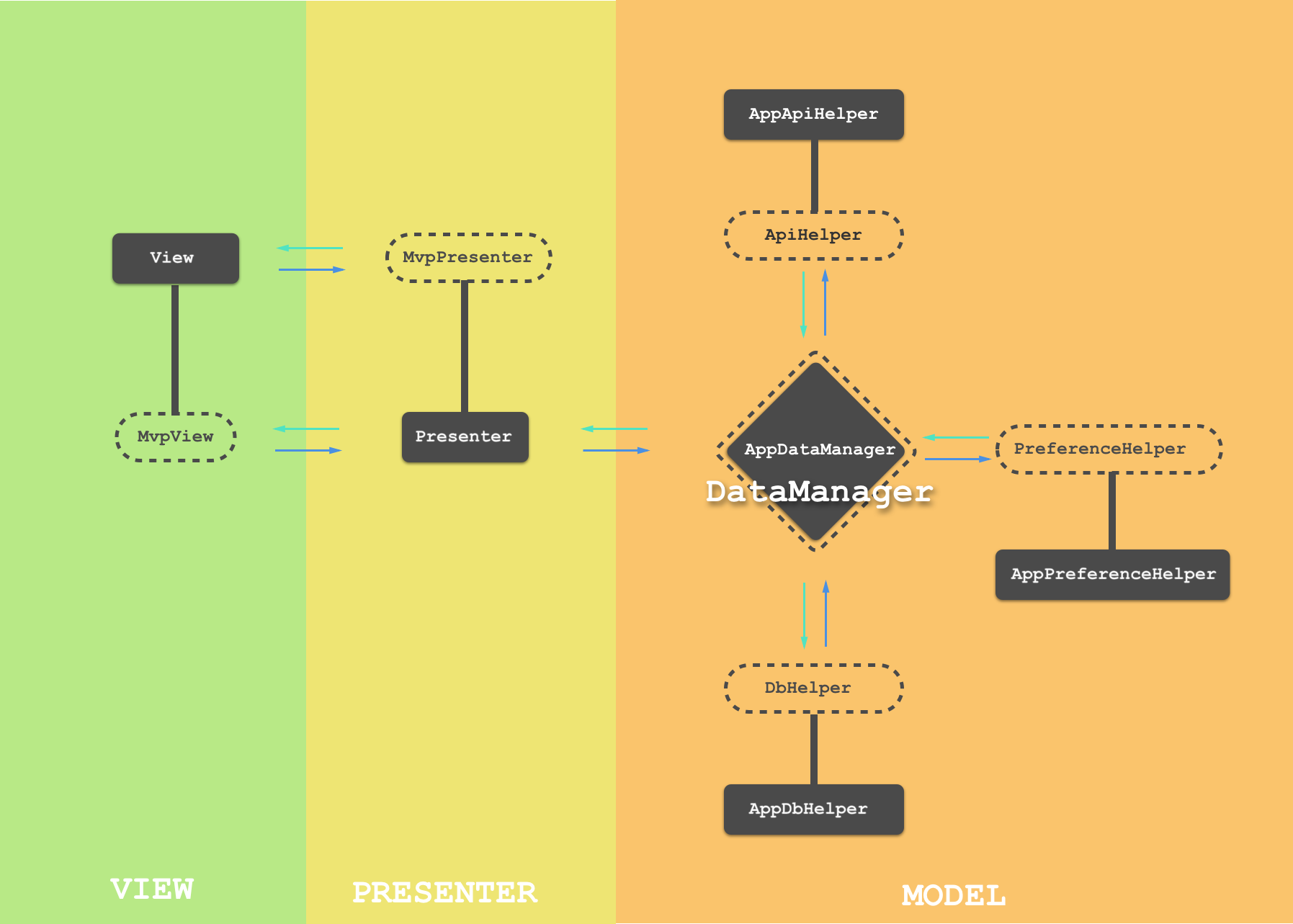
*Now, all these information can evidently be found on any blog or Android guide for MVP. Then what’s the point of this article?*

*The reason that this article is written is to solve a very important challenge with MVP.* ***How to actually implement it as an entire application?***

MVP appears to be very simple when explained with a simple Activity example but makes us feel lost when we try binding all the components of an application together.

*If you want to dive deep into a world of beautiful coding and be mesmerized then follow along with this article. It’s not a news article, so get involved with it, put on your shoes with your back straight and away from all distractions.*

#### Let’s sketch the blueprint of the architecture first.



Architecture is the first thing you should work on for any software. A carefully crafted architecture will minimize a lot of rework in future while providing the scalability ease. Most of the project today is developed in a team, hence the code readability and modularity should be taken as utmost important elements of the architecture. We also rely heavily on 3rd party libraries and we keep switching between alternatives because of use cases, bugs or support. So, our architecture should be designed with plug and play design. The interfaces for classes serves this purpose.

The blueprint of the Android architecture drawn above contains all these features and is based on the principles of MVP.

https://blog.mindorks.com/essential-guide-for-designing-your-android-app-architecture-mvp-part-1-74efaf1cda40#.lkml1yggq