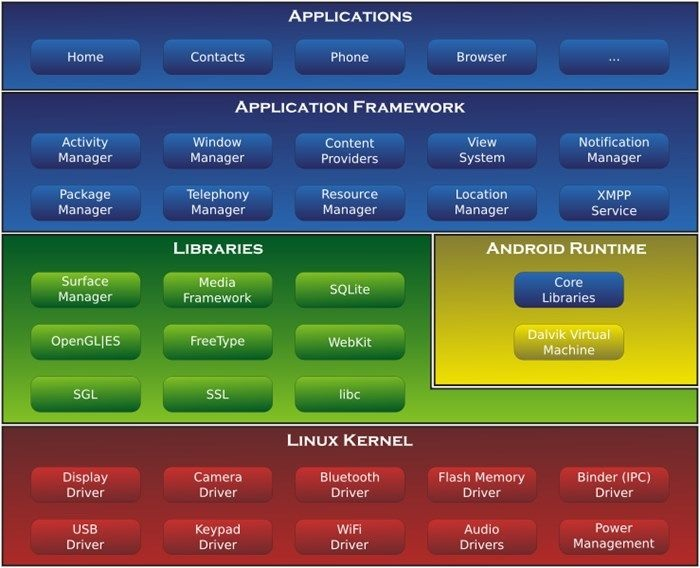
# 第一天:基础

到官网下载安装Android Studio之后,新建第一个项目HelloWorld,要运行Android项目,需要连接真机或者模拟器,新建模拟器之前需要在BIOS中打开intel虚拟化技术,然后在打开AvdManager,创建模拟器,点击运行程序.

## 1.1Android系统架构



Android分为四层架构：Linux内核层、系统运行库层、应用框架层和应用层。

1. Linux内核层

Android系统是基于Linux内核为Android设备的各种硬件提供了底层的驱动，如显示驱动、音频驱动、照相机驱动、蓝牙驱动、Wi-Fi驱动、电源管理等。

2. 系统运行库层

(1)C/C++库来为Android系统提供了主要的特性支持。如SQLite库提供了数据库的支持，OpenGL|ES库提供了3D绘图的支持，Webkit库提供了浏览器内核的支持等。

(2)Android运行时库提供了一些核心库，开发者**使用Java语言来编写Android应用**。另外，Android运行时库中还包含了Dalvik虚拟机（5.0系统后为ART运行环境），它使得每一个Android应用都能运行在独立的进程当中，并且拥有一个自己的Dalvik虚拟机实例。相较于Java虚拟机，Dalvik是专门为移动设备定制的，它针对手机内存、CPU性能有限等情况做了优化处理。

3. 应用框架层

这一层主要提供了构建应用程序时可能用到的各种API，Android自带的一些核心应用就是使用这些API完成的，开发者也可以通过使用这些API来构建自己的应用程序。

04. 应用层

所有安装在手机上的应用程序都是属于这一层的，比如系统自带的联系人、短信等程序，

或者是你从Google Play上下载的小游戏，当然还包括你自己开发的程序。

## 1.2开发特色

1.**四大组件**:活动（Activity）、服务（Service）、广播接收器（Broadcast

Receiver）和内容提供器（Content Provider）

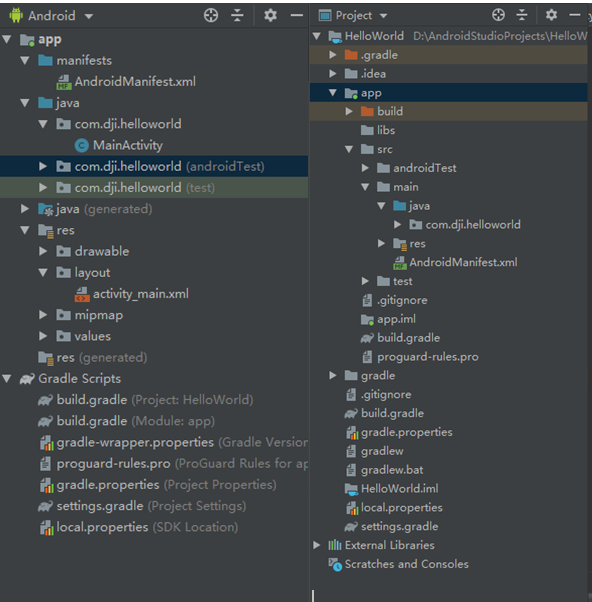
2.系统控件,也可以定制控件

3.SQLite数据库

4.多媒体服务

5.地理位置服务

## 1.3项目结构

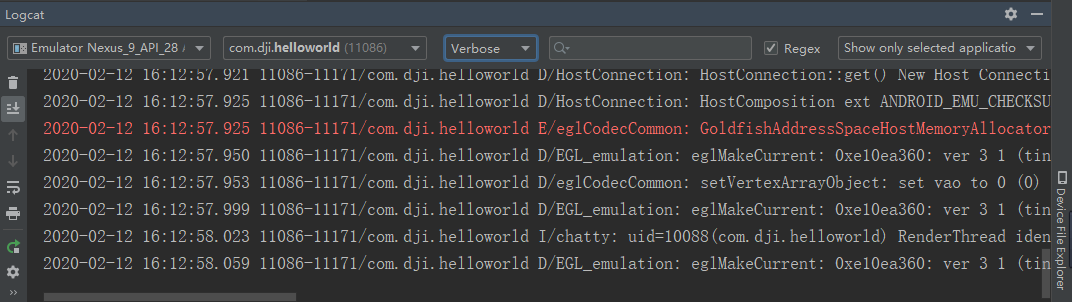


主要有两种模式下的项目结构:Android和Project

## 1.4日志

//五种日志等级  
Log.*v*(*TAG*, "verbose");  
Log.*d*(*TAG*, "debug ");  
Log.*i*(*TAG*, "info");  
Log.*w*(*TAG*, "wan");  
Log.*e*(*TAG*, "error");

同时可以使用Logcat对日志进行筛选分析.



# 第二天:活动

活动（Activity）是最容易吸引用户的地方，它是一种可以包含用户界面的组件，主要用于和用户进行交互。

## 2.1Toast的使用:

Toast.*makeText*(context,"received in MyBroadcastReceiver", Toast.*LENGTH\_SHORT*).show();

## 2.2Menu的使用:

在res目录下面添加menu目录,编写main.xml,

<?xml version="1.0" encoding="utf-8"?>  
<menu xmlns:android="http://schemas.android.com/apk/res/android">  
 <item android:id="@+id/add\_item"  
 android:title="Add"/>  
 <item android:id="@+id/remove\_item"  
 android:title="Remove"/>  
</menu>

然后在活动类中重写方法.

@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
 //getMenuInflater() 方法能够得到MenuInflater 对象，再调用它的inflate() 方法就可以给当前活动创建菜单  
 //第一个参数用于指定我们通过哪一个资源文件来创建菜单，这里当然传入R.menu.main  
 //第二个参数用于指定我们的菜单项将添加到哪一个Menu 对象  
 getMenuInflater().inflate(R.menu.*main*, menu);  
 return true;  
}

## 2.3Intent的使用

分为显式和隐式.

### 1.显式Intent

@Override  
public void onClick(View v) {  
 //显式Intent  
 Intent intent = new Intent(FirstActivity.this, SecondActivity.class);  
 startActivity(intent);  
}

### 2.隐式Intent

首先在Manifest.xml中通过在<activity> 标签下配置<intent-filter> 的内容，可以指定当前活动能够响应的action 和category

<activity android:name=".SecondActivity"  
 android:launchMode="singleInstance">  
 <intent-filter>  
<!-- 指明了当前活动可以响应 com.example.activitytest.ACTION\_START 这个action-->  
 <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>

然后在第一个活动代码中添加

@Override  
public void onClick(View v) {  
 //隐式Intent  
 Intent intent = new Intent("com.example.activitytest.ACTION\_START");  
 intent.addCategory("com.example.activitytest.MY\_CATEGORY");  
 startActivity(intent);  
}

## 2.4活动生命周期

onCreate() 。这个方法你已经看到过很多次了，每个活动中我们都重写了这个方法，它会在活动第一次被创建的时候调用。你应该在这个方法中完成活动的初始化操作，比如说加载布局、绑定事件等。

onStart() 。这个方法在活动由不可见变为可见的时候调用。

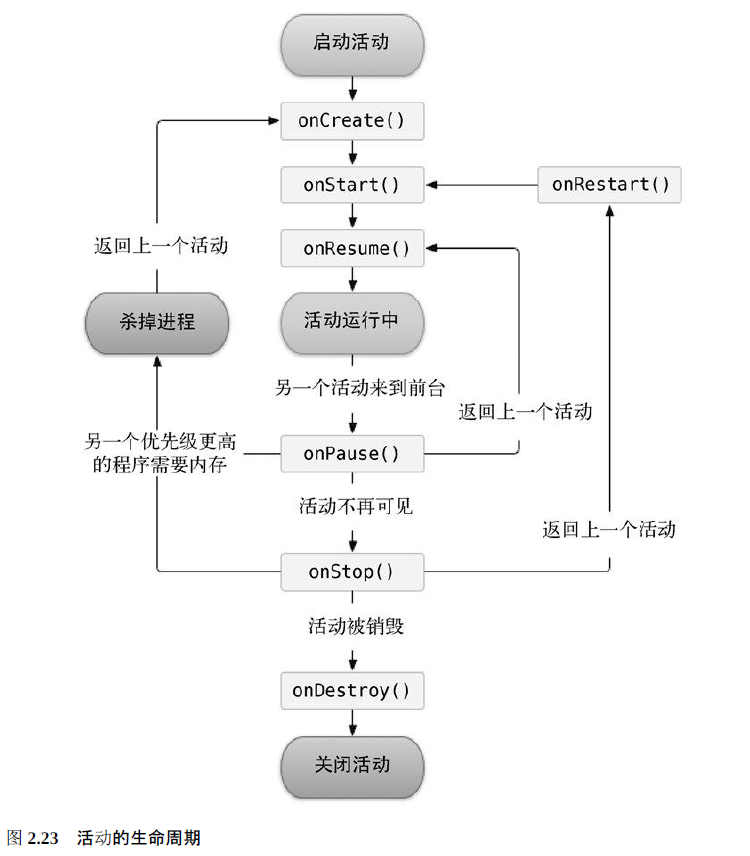
onResume() 。这个方法在活动准备好和用户进行交互的时候调用。此时的活动一定位于返回栈的栈顶，并且处于运行状态。

onPause() 。这个方法在系统准备去启动或者恢复另一个活动的时候调用。我们通常会在这个方法中将一些消耗CPU的资源释放掉，以及保存一些关键数据，但这个方法的执行速度一定要快，不然会影响到新的栈顶活动的使用。

onStop() 。这个方法在活动完全不可见的时候调用。它和onPause() 方法的主要区别在于，如果启动的新活动是一个对话框式的活动，那么onPause() 方法会得到执行，而onStop() 方法并不会执行。

onDestroy() 。这个方法在活动被销毁之前调用，之后活动的状态将变为销毁状态。

onRestart() 。这个方法在活动由停止状态变为运行状态之前调用，也就是活动被重新启动了。



临时数据保存:

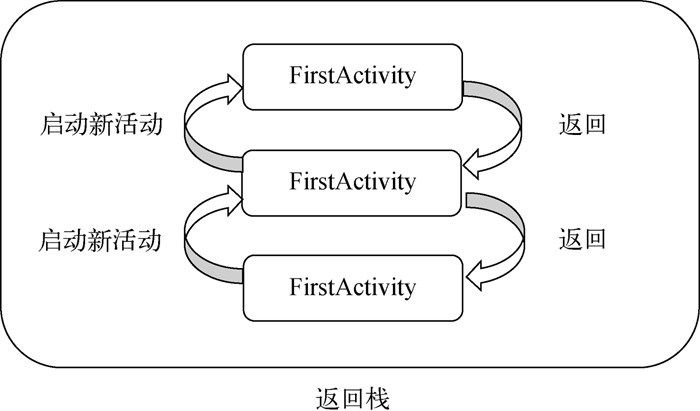
*/\*\*  
 \* 可以保证在活动被回收之前一定会被调用,  
 \* 可以通过这个方法来解决活动被回收时临时数据得不到保存的问题  
 \** ***@param*** *outState  
 \** ***@param*** *outPersistentState  
 \*/*@Override  
public void onSaveInstanceState(@NonNull Bundle outState, @NonNull PersistableBundle outPersistentState) {  
 super.onSaveInstanceState(outState, outPersistentState);  
 String tempData = "Something you just typed";  
 //将数据保存在Bundle中，可以在onCreate方法中使用  
 outState.putString("data\_key", tempData);  
}

同时,在onCreate方法中通过Bundle类型可以得到上面保存的数据:

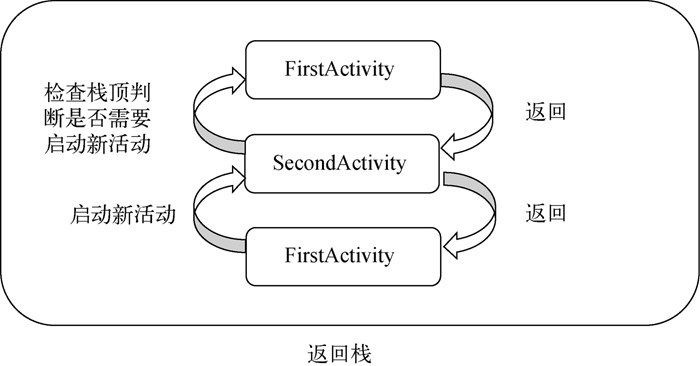
if (savedInstanceState != null) {  
 String tempData = savedInstanceState.getString("data\_key");  
 Log.*d*(*TAG*, "tempData； " + tempData);  
}

## 2.5活动的启动模式

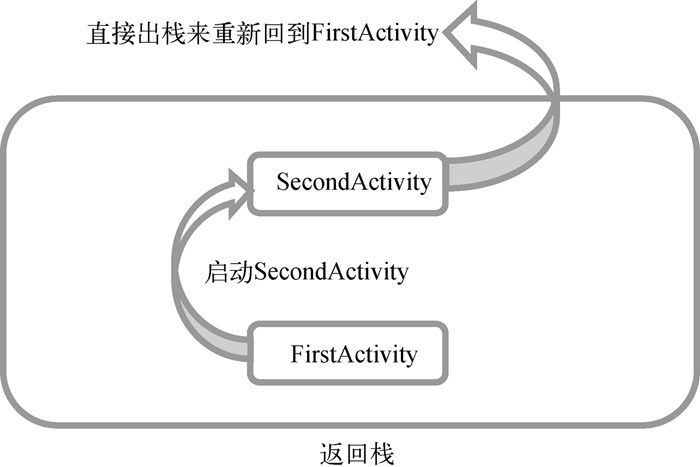
1. standard是活动**默认**的启动模式，在不进行显式指定的情况下，所有活动都会自动使用这种启动模式。在Manifest.xml的活动配置中添加android:launchMode="singleTop",其他模式配置类似,原理图为:



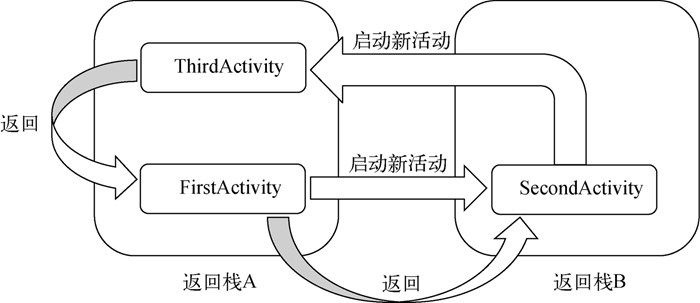
2. singleTop，在启动活动时如果发现返回栈的栈顶已经是该活动，则认为可以直接使用它，不会再创建新的活动实例。



3. singleTask，每次启动该活动时系统首先会在返回栈中检查是否存在该活动的实例，如果发现已经存在则直接使用该实例，并把在这个活动之上的所有活动统统出栈，如果没有发现就会创建一个新的活动实例.



4. singleInstance，在这种模式下会有一个单独的返回栈来管理这个活动，不管是哪个应用程序来访问这个活动，都共用的同一个返回栈，也就解决了共享活动实例的问题.类似于单例模式.



# 第三天:UI

## 3.1常用控件

### 1.TextView

<!-- android:gravity 来指定文字的对齐方式-->  
 <TextView  
 android:id="@+id/text\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:textSize="24sp"  
 android:textColor="#00ff00"  
 android:text="This is TextView" />

### 2.Button

<!--android:textAllCaps默认为true，对text内容自动转换为大写-->  
 <Button  
 android:id="@+id/button"  
 android:text="Button"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:textAllCaps="false"/>

### 3.EditText

<!--android:hint:提示内容  
android:maxLines：最大行数-->  
 <EditText  
 android:id="@+id/edit\_text"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:hint="Type something here."  
 android:maxLines="2"  
 />

### 4. ImageView

<!--图片控件-->  
 <ImageView  
 android:id="@+id/image\_view"  
 android:src="@drawable/img\_1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 />

### 5. ProgressBar

<!--默认为圆圈的进度条，  
使用style 属性可以将它指定成水平进度条,  
android:max给进度条设置最大值  
-->  
 <ProgressBar  
 android:id="@+id/progress\_bar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 style="@style/Widget.AppCompat.ProgressBar.Horizontal"  
 android:max="100"  
 />

### 6. AlertDialog

*/\*\*  
 \* 弹出对话框,AlertDialog  
 \*/*AlertDialog.Builder dialog = new AlertDialog.Builder(MainActivity.this);  
dialog.setTitle("This is a dialog");  
dialog.setMessage("Something important");  
dialog.setCancelable(false);  
dialog.setPositiveButton("OK", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 }  
});  
dialog.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
  
 }  
});  
dialog.show();

### 7. ProgressDialog

*/\*\*  
 \* ProgressDialog, 比AlertDialog多一个进度条  
 \*/*ProgressDialog progressDialog = new ProgressDialog(MainActivity.this);  
progressDialog.setTitle("This is a ProgressDialog");  
progressDialog.setMessage("Loading...");  
progressDialog.setCancelable(true);  
progressDialog.show();

## 3.2基本布局

### 1.线性布局

<!--不指定android:orientation 属性的值，默认的排列方向  
就是horizontal-->  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <Button  
 android:id="@+id/button1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="top"  
 android:text="Button 1" />  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="center\_vertical"  
 android:text="Button 2" />  
 <Button  
 android:id="@+id/button3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="bottom"  
 android:text="Button 3" />  
</LinearLayout>

其中,线性布局有按比例布局的应用:

<!--按比例布局-->  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:orientation="horizontal"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <EditText  
 android:id="@+id/input\_message"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_weight="1"  
 android:hint="Type something"  
 />  
 <Button  
 android:id="@+id/send"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Send"  
 />  
</LinearLayout>

### 2.相对布局

<!--相对布局-->  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <Button  
 android:id="@+id/button1"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentLeft="true"  
 android:layout\_alignParentTop="true"  
 android:text="Button 1" />  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentRight="true"  
 android:layout\_alignParentTop="true"  
 android:text="Button 2" />  
 <Button  
 android:id="@+id/button3"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerInParent="true"  
 android:text="Button 3" />  
 <Button  
 android:id="@+id/button4"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentLeft="true"  
 android:text="Button 4" />  
 <Button  
 android:id="@+id/button5"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentBottom="true"  
 android:layout\_alignParentRight="true"  
 android:text="Button 5" />  
</RelativeLayout>

### 3.帧布局

!--帧布局：所有的控件都会默认摆放在布局的左上角  
使用较少-->  
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent">  
 <TextView  
 android:id="@+id/text\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="left"  
 android:text="This is TextView"  
 />  
 <ImageView  
 android:id="@+id/image\_view"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_gravity="right"  
 android:src="@mipmap/ic\_launcher"  
 />  
</FrameLayout>

### 4.百分比布局:

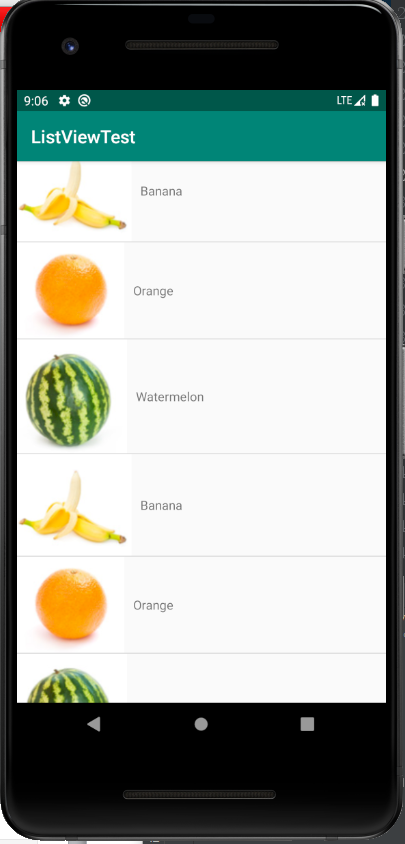
需要修改gradle文件,导入新的jar包,在build.gradle中添加:

implementation 'androidx.percentlayout:percentlayout:1.0.0'

<?xml version="1.0" encoding="utf-8"?>  
<androidx.percentlayout.widget.PercentFrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 xmlns:app="http://schemas.android.com/apk/res-auto">  
  
 <Button  
 android:id="@+id/button1"  
 android:text="Button 1"  
  
 android:layout\_gravity="left|top"  
 app:layout\_widthPercent="50%"  
 app:layout\_heightPercent="50%"  
 />  
 <Button  
 android:id="@+id/button2"  
 android:text="Button 2"  
  
 android:layout\_gravity="right|top"  
 app:layout\_widthPercent="50%"  
 app:layout\_heightPercent="50%"  
 />  
 <Button  
 android:id="@+id/button3"  
 android:text="Button 3"  
  
 android:layout\_gravity="left|bottom"  
 app:layout\_widthPercent="50%"  
 app:layout\_heightPercent="50%"  
 />  
 <Button  
 android:id="@+id/button4"  
 android:text="Button 4"  
  
 android:layout\_gravity="right|bottom"  
 app:layout\_widthPercent="50%"  
 app:layout\_heightPercent="50%"  
 />  
  
  
</androidx.percentlayout.widget.PercentFrameLayout>

## 3.3ListView

用于滚动展示数据:



使用:首先在布局文件中添加

<ListView  
 android:id="@+id/list\_view"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"/>

然后在onCreate方法中添加:

*/\*\*  
 \* 数组中的数据是无法直接传递给ListView的，我们还需要借助适配器来完成  
 \* 使用android.R.layout.simple\_list\_item\_1 作为ListView子项布局的id，  
 \* 这是一个Android内置的布局文件，里面只有一个TextView，可用于简单地显示一段文本。  
 \*/*ArrayAdapter<String> adapter = new ArrayAdapter<String>(  
 MainActivity.this, android.R.layout.*simple\_list\_item\_1*, data  
);  
ListView listView = findViewById(R.id.*list\_view*);  
listView.setAdapter(adapter);

## 3.4RecylerView

与ListView类似的效果,滚动显示数据.但更强大.

首先,在gradle中添加:

implementation 'androidx.recyclerview:recyclerview:1.0.0'

然后在布局文件中使用该控件:

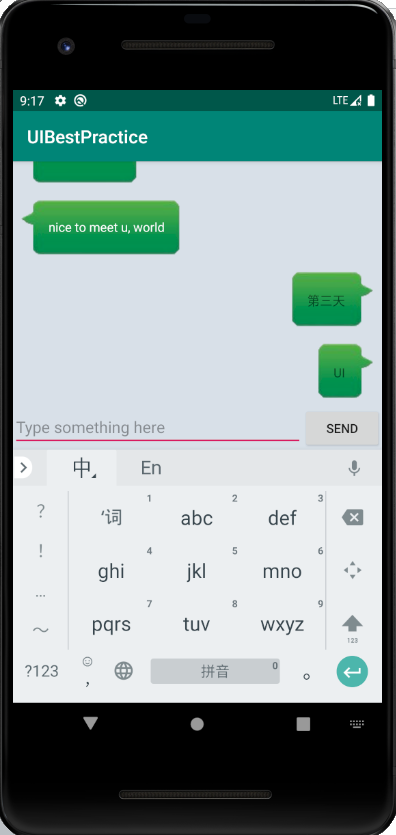
<androidx.recyclerview.widget.RecyclerView  
 android:id="@+id/recycler\_view"  
 android:layout\_height="match\_parent"  
 android:layout\_width="match\_parent" />

在onCreate方法中添加:

RecyclerView recyclerView = findViewById(R.id.*recycler\_view*);  
*/\*\*  
 \* LayoutManager用于指定RecyclerView的布局方式  
 \* 横向排列  
 \*/*/\*LinearLayoutManager layoutManager = new LinearLayoutManager(this);  
layoutManager.setOrientation(RecyclerView.HORIZONTAL);\*/  
  
*/\*\*  
 \* 一行3个，横向排列  
 \*/*StaggeredGridLayoutManager layoutManager =  
 new StaggeredGridLayoutManager(3, StaggeredGridLayoutManager.*VERTICAL*);  
recyclerView.setLayoutManager(layoutManager);  
  
//将数据传入FruitAdapter，最后调用setAdapter完成适配器设置  
FruitAdapter adapter = new FruitAdapter(fruitList);  
recyclerView.setAdapter(adapter);

## 3.5项目demo

编写项目UIBestPractice,使用RecylcerView实现聊天界面:

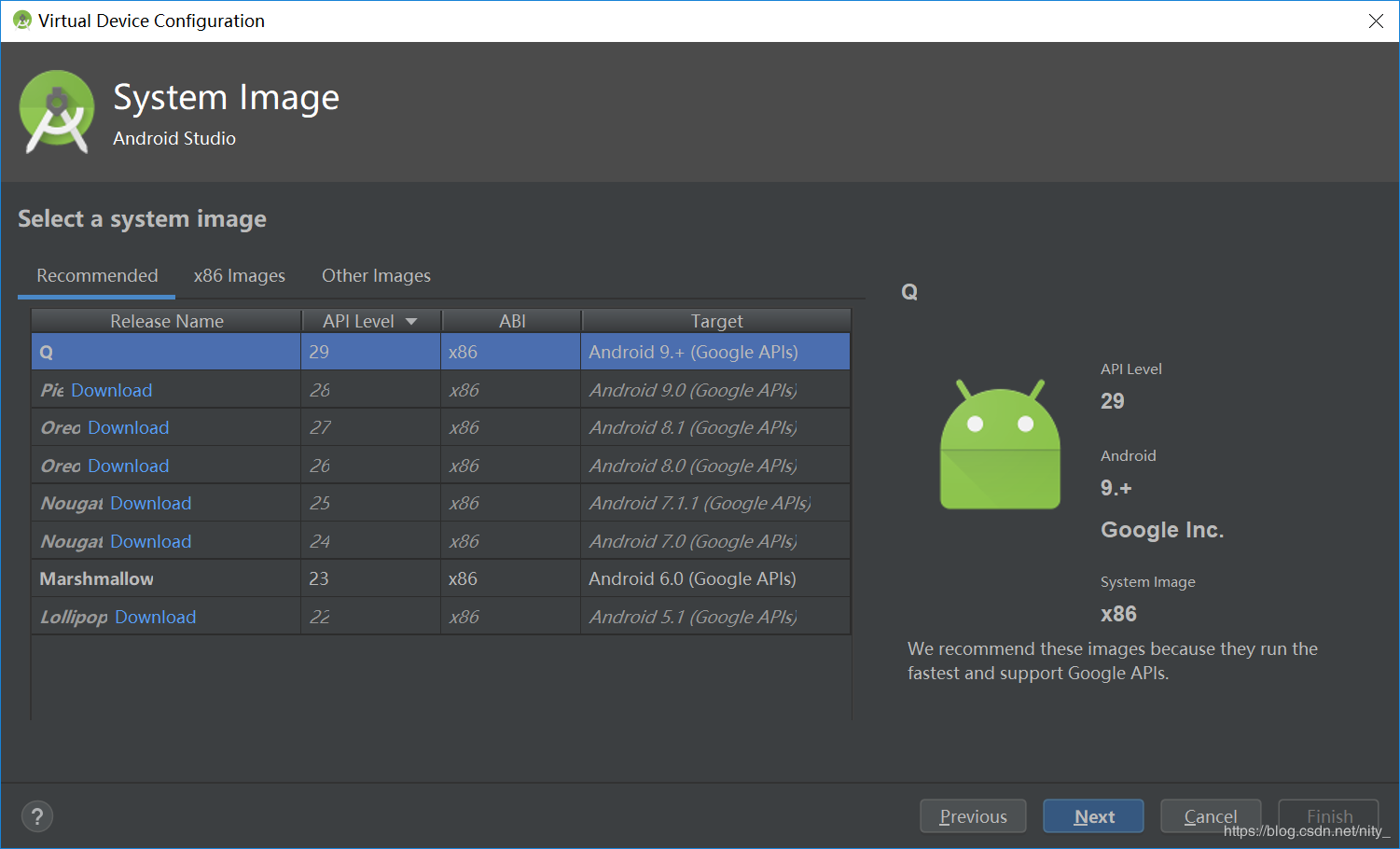


# 关于/system/bin/sh: su: not found的解决办法（安卓模拟器运行）

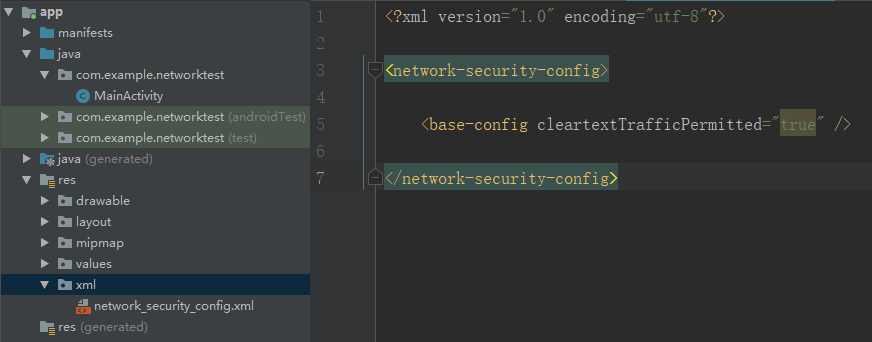
在了解关于数据库的创建时，无法在android Studio的模拟器运行的条件下运行cmd 窗口下adb shell 命令，adb shell之后su无法转到root权限,导致无法进入data/data/…原因是

Android Studio带(Google Play)的模拟器无法获得root权限安装

该换成为带(Google APIs)的模拟器即可，类似于这种



# java.net.UnknownServiceException: CLEARTEXT communication to 192.168.1.18 not permitted by network security policy



<?xml version="1.0" encoding="utf-8"?>

<network-security-config>

<base-config cleartextTrafficPermitted="true" />

</network-security-config>

* 1
* 2
* 3
* 4

