Android UI develop

范心仪-字节跳动 Android 工程师

fanxinyi.1008@bytedance.com

ByteDance 字节跳动

今天会学到什么?

- Activity基础
- 了解如何编写View布局
- 掌握高级UI组件RecyclerView的使用
- 实现一个抖音热搜榜列表

Activity

https://developer.android.google.cn/guide/components/activities/

Activity

- 什么是 Activity
- Activity 的基本用法
- Activity 之间的跳转
- Activity 的生命周期
- Activity 的启动模式

1.1 什么是 Activity

- 概念

  1. Android的四大组件之一

  2. 用户操作的可视化界面

     https://developer.android.com/guide/components/activities/?hl=zh-CN

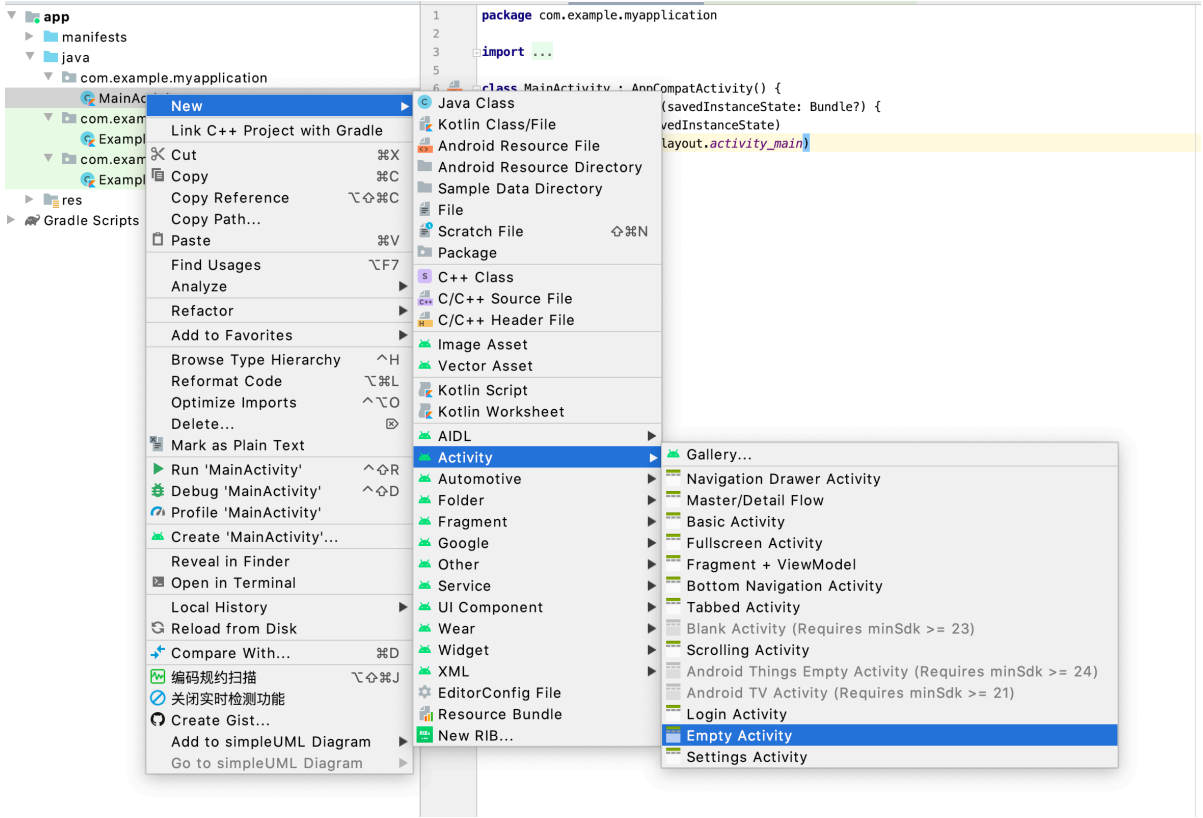
- 打开Activity

  1. 创建

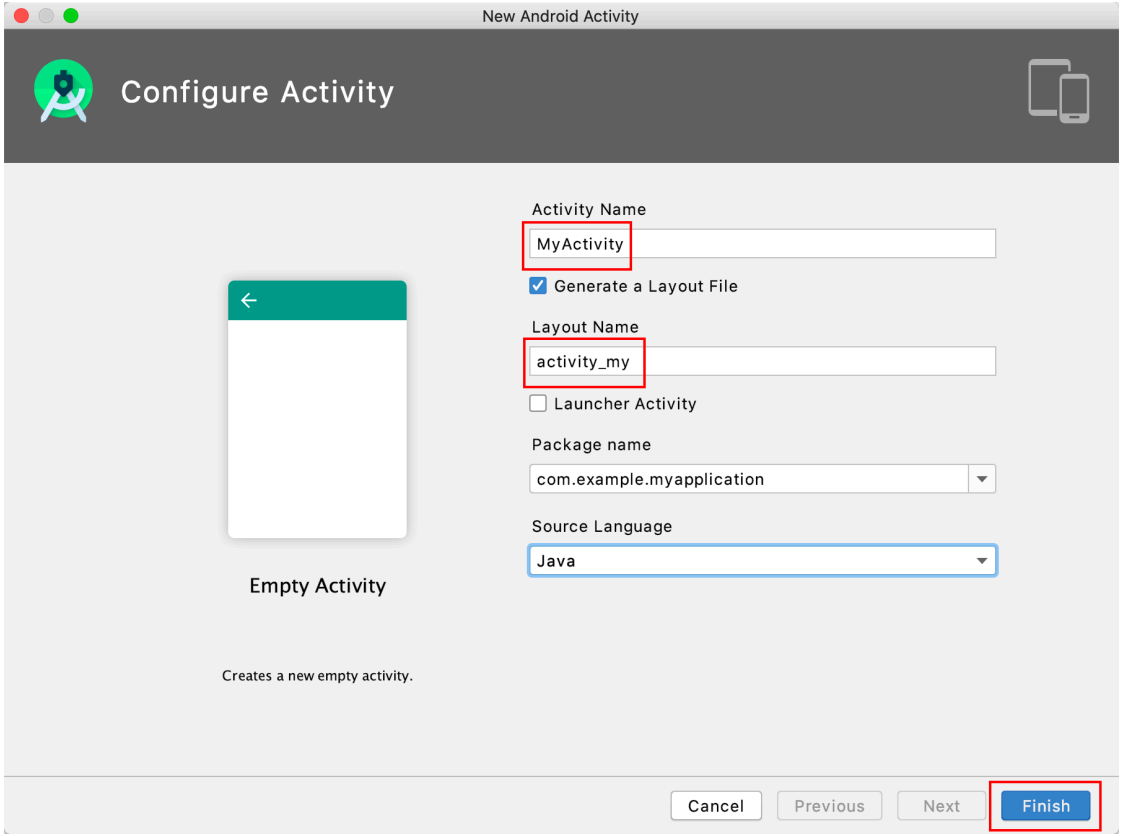
  2. 注册

  3. 启动

1.2 Activity 的基本用法-创建



1.2 Activity 的基本用法-创建



1.2 Activity 的基本用法-创建

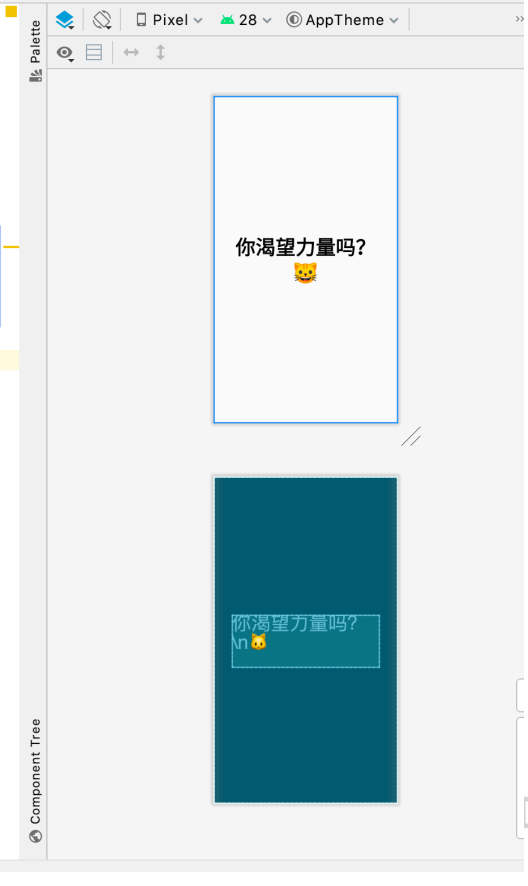
1.2 Activity 的基本用法 — MyActivity.java

MyActivity.java

```
1   package com.example.chapter2;
2
3   import ...
6
7   public class MyActivity extends AppCompatActivity {
8
9       @Override
10      protected void onCreate(Bundle savedInstanceState) {
11          super.onCreate(savedInstanceState);
12          setContentView(R.layout.activity_my);
13      }
14  }
```

1.2 Activity 的基本用法 — activity_my.xml

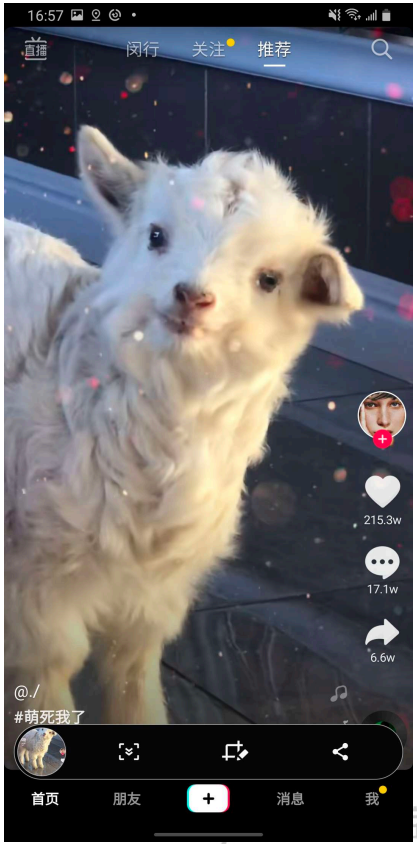
```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center">


    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="你渴望力量吗? \n😺"
        android:textSize="45sp"
        android:textColor="#000000"
        android:textStyle="bold"
        android:gravity="center"/>

</LinearLayout>
```



1.2 Activity 的基本用法 — AndroidManifest.xml

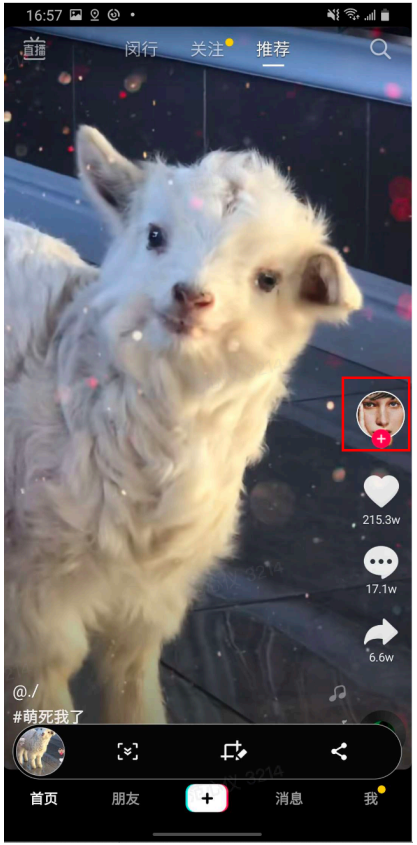
activity_my.xml | MyActivity.java | AndroidManifest.xml

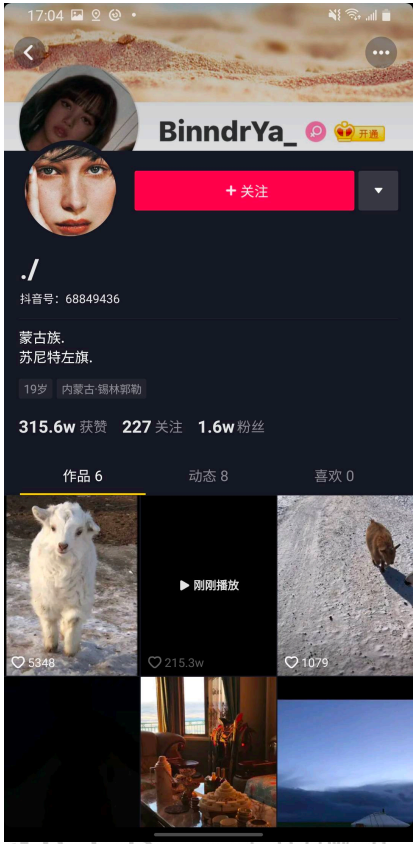
```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MyActivity"></activity>
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```



1.3 Activity 之间的跳转

- 显示 Intent
- 隐式 Intent
- 向下一个 Activity 传递数据





https://developer.android.com/guide/components/intents-filters?hl=zh_cn

1.3 Activity 之间的跳转 — 显式 Intent

```
Intent intent = new Intent( packageContext: MainActivity.this, MyActivity.class);
startActivity(intent);
```

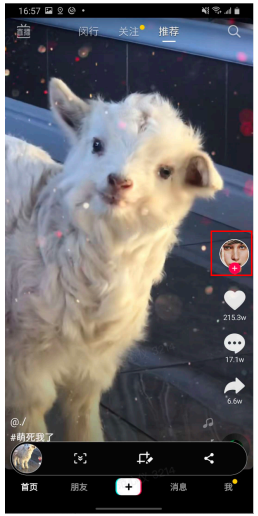
1.3 Activity 之间的跳转 – 向下一个 Activity 传递数据

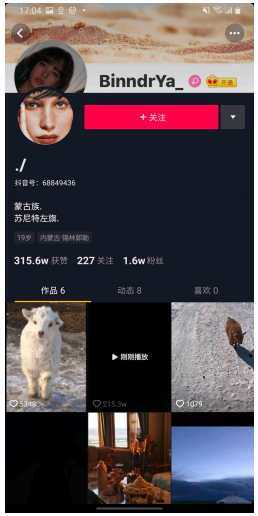
❑ 传递数据

```
Intent intent = new Intent( packageContext: MainActivity.this, MyActivity.class);
intent.putExtra( name: "userId", value: 123456);
startActivity(intent);
```

❑ 获取数据

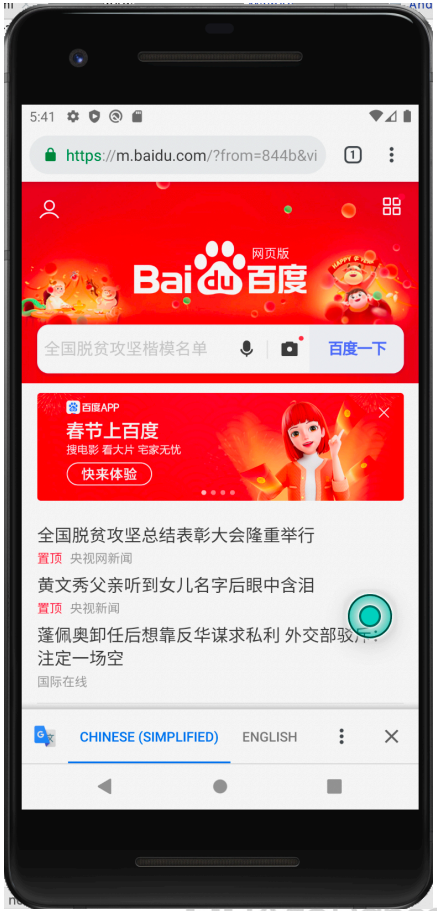
```
Long extra = getIntent().getLongExtra( name: "userId", defaultValue: 0);
```





1.3 Activity 之间的跳转 — 隐式 Intent

```
Intent intent = new Intent(Intent.ACTION_VIEW);
intent.setData(Uri.parse("http://www.baidu.com"));
startActivity(intent);
```



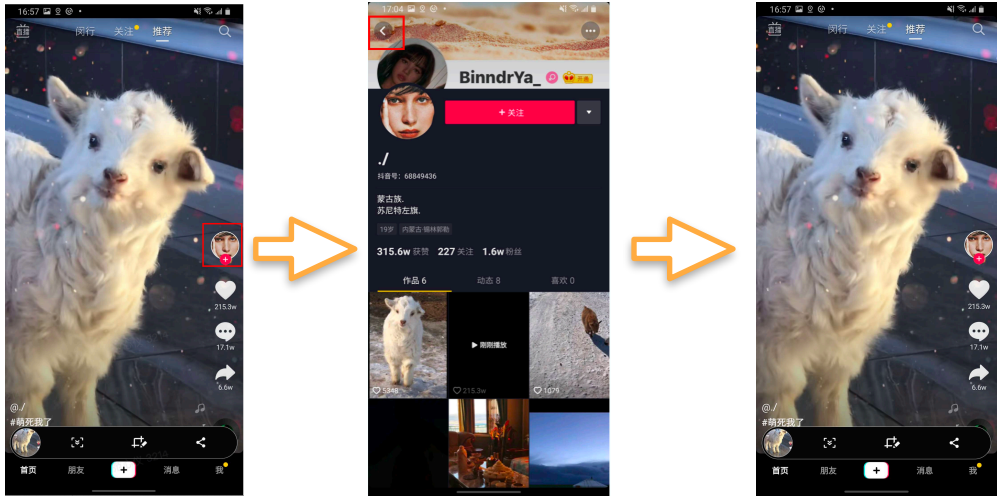
1.4 Activity 的生命周期

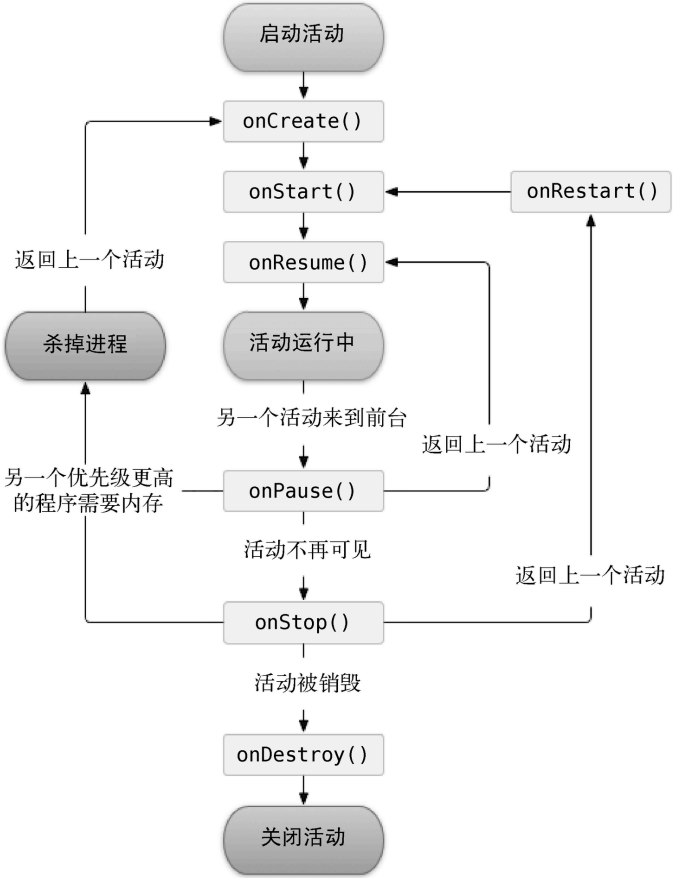
https://developer.android.com/guide/components/activities/activity-lifecycle?hl=zh-cn

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }
    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be "paused").
    }
    @Override
    protected void onStop() {
        super.onStop();
        // The activity is no longer visible (it is now "stopped")
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        // The activity is about to be destroyed.
    }
}
```

1.4 Activity 的生命周期

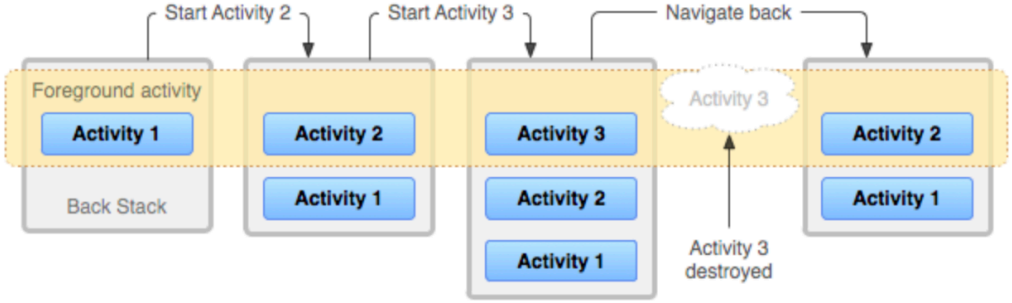
- 完整生存期: onCreate() - onDestroy()
- 可见生存期: onStart() - onStop()
- 前台生存期: onResume() - onPause()

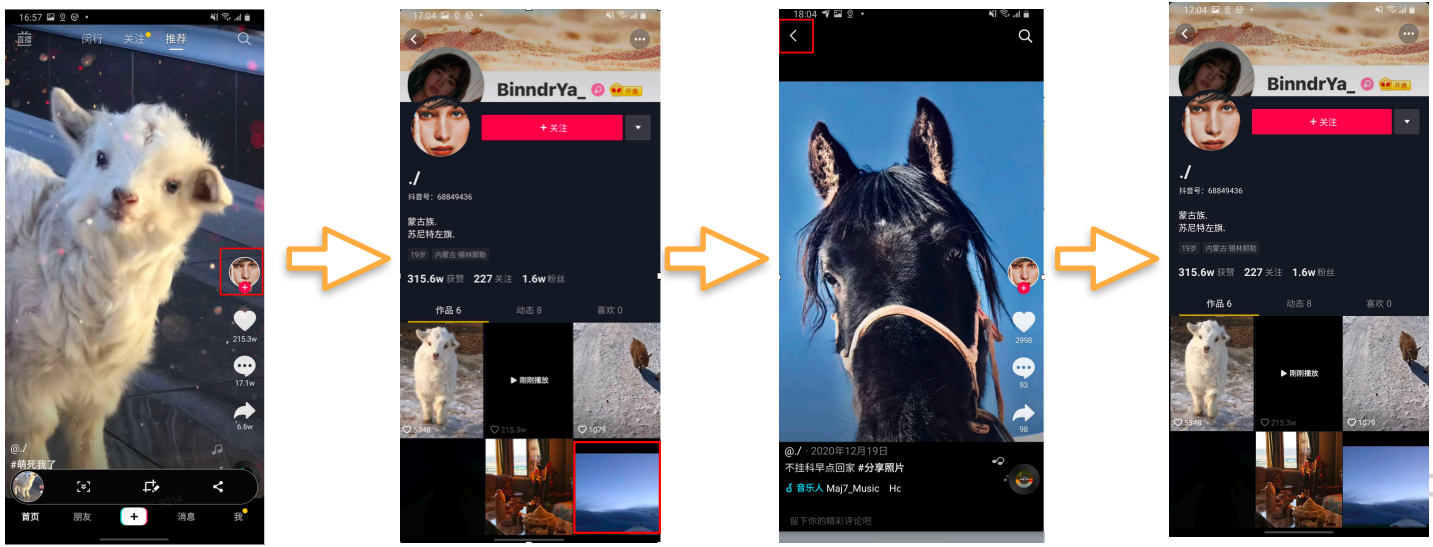




1.5 Activity 返回栈

瞅瞅更多：https://developer.android.com/guide/components/activities/tasks-and-back-stack?hl=zh-cn





1.5 Activity 启动模式

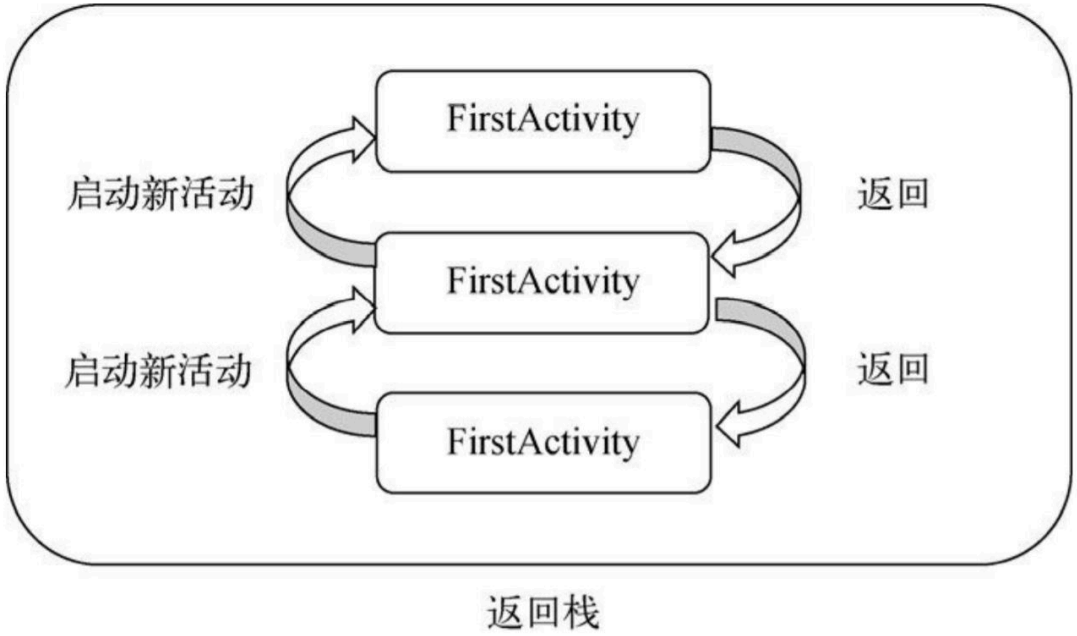
- standard
- singleTop
- singleTask
- singleInstance

瞅瞅更多：https://inthecheesefactory.com/blog/understand-android-activity-launchmode/en

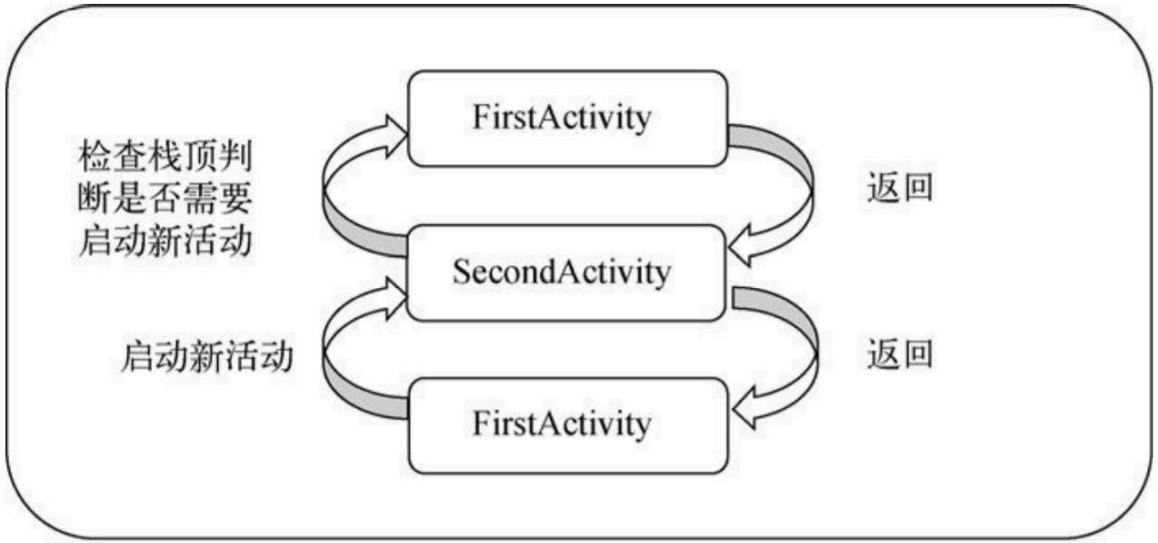
1.5 Activity 启动模式

```
<activity android:name=".MyActivity"
    android:launchMode="standard">
    <intent-filter>
        <action android:name="com.example.chapter2.ACTION_START"/>

        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
```

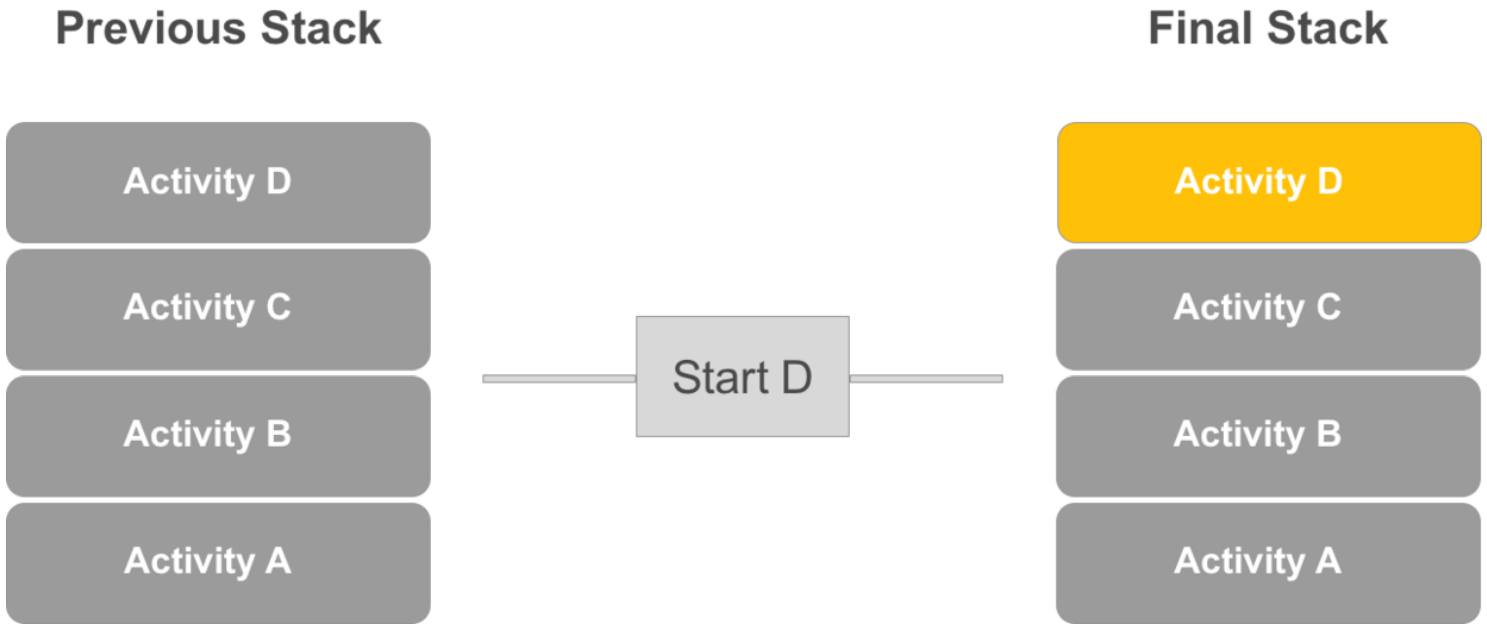
1.5 Activity 启动模式 — standard



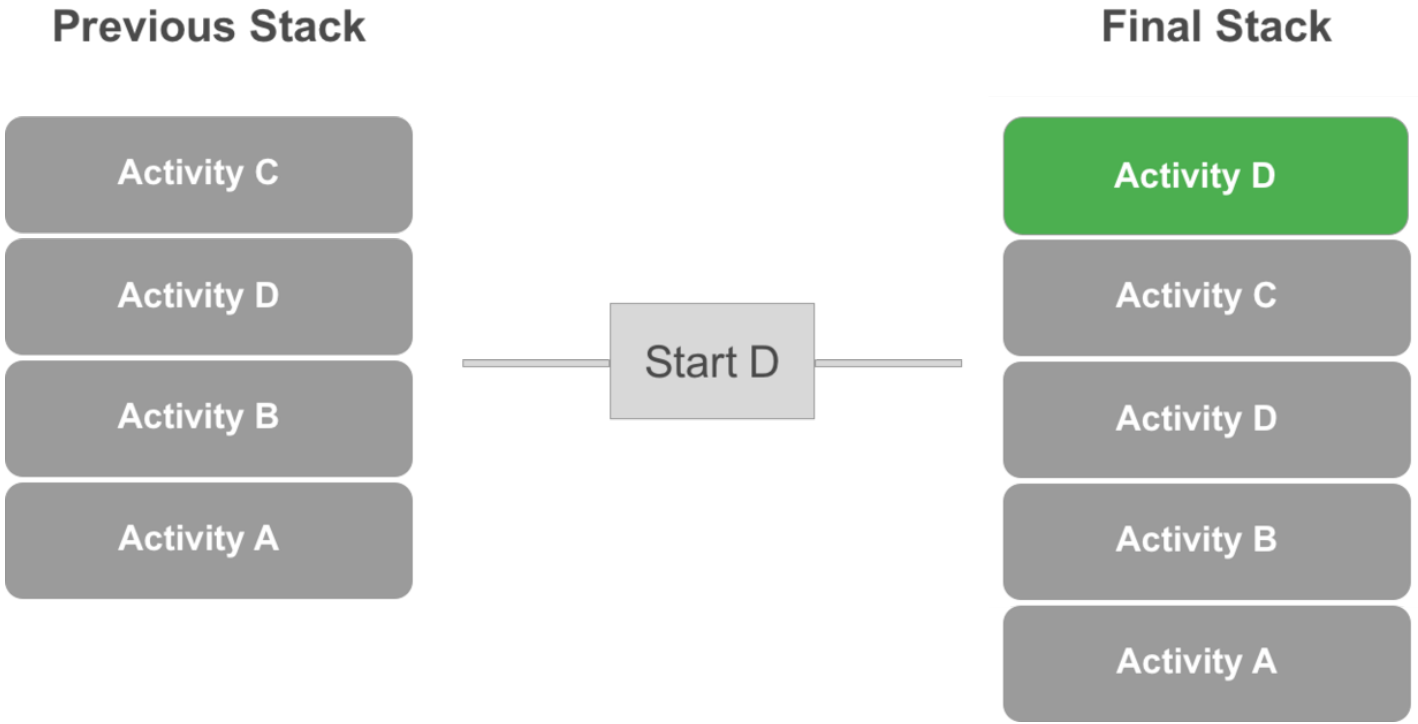
1.5 Activity 启动模式 — singleTop



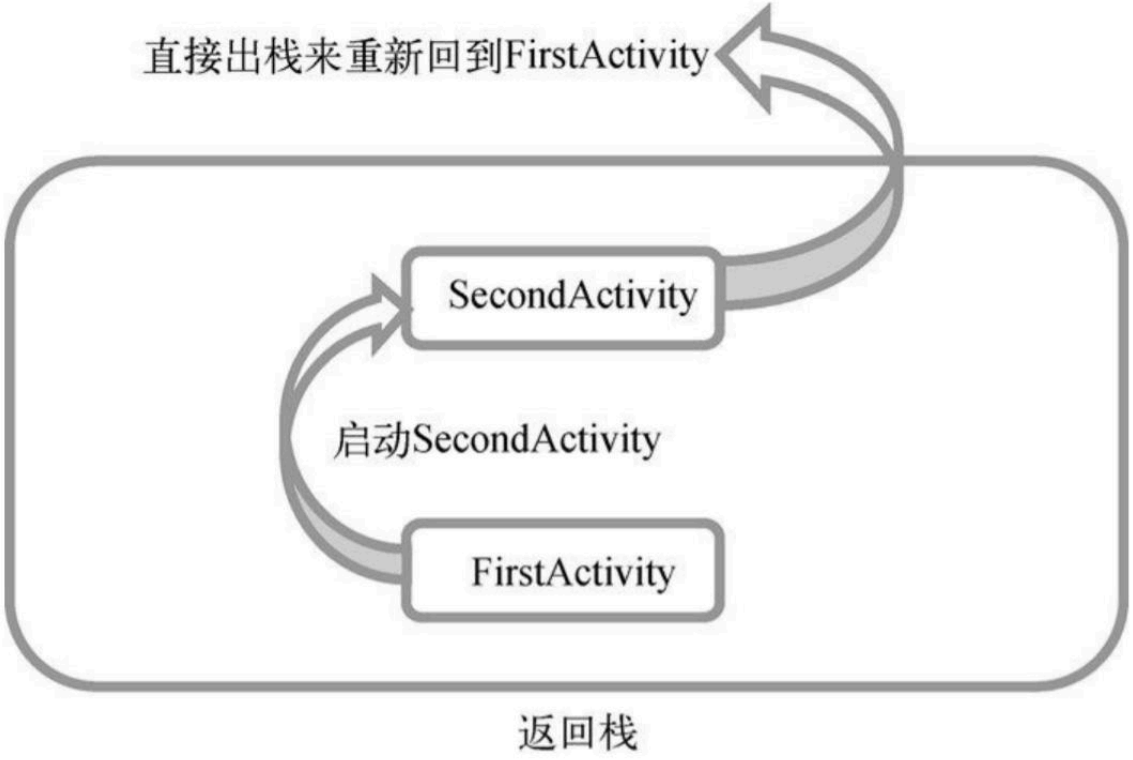
1.5 Activity 启动模式 — singleTop



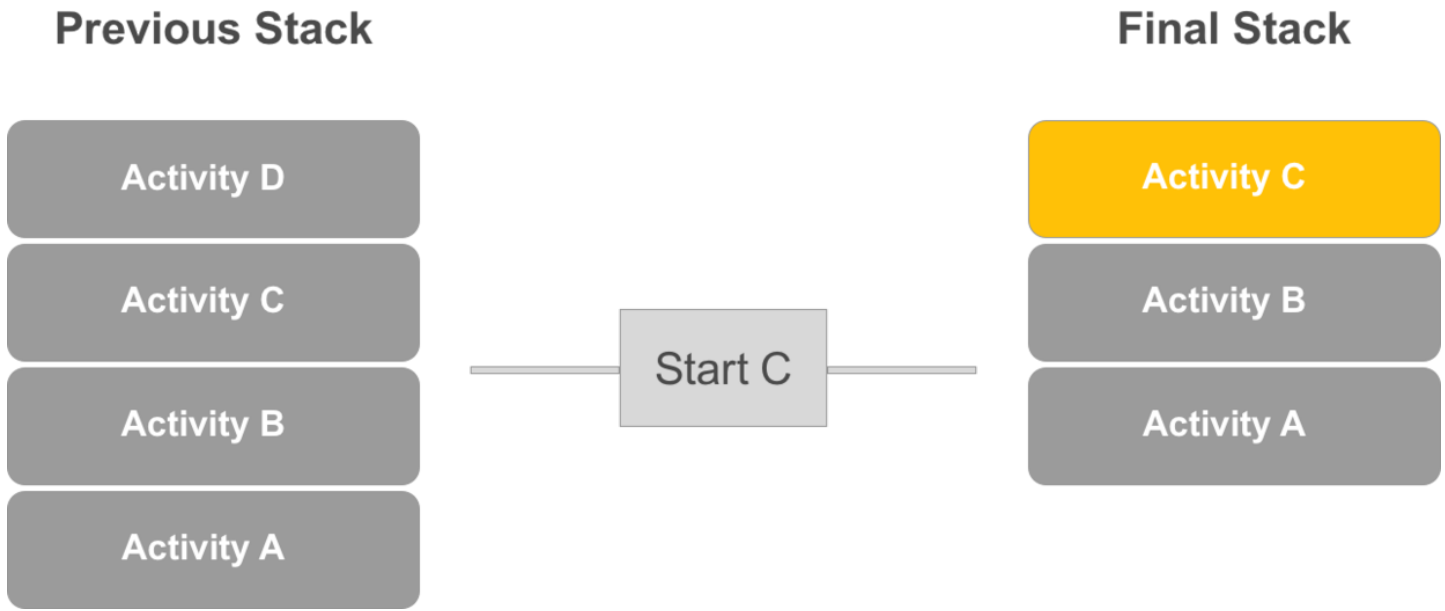
1.5 Activity 启动模式 — singleTop



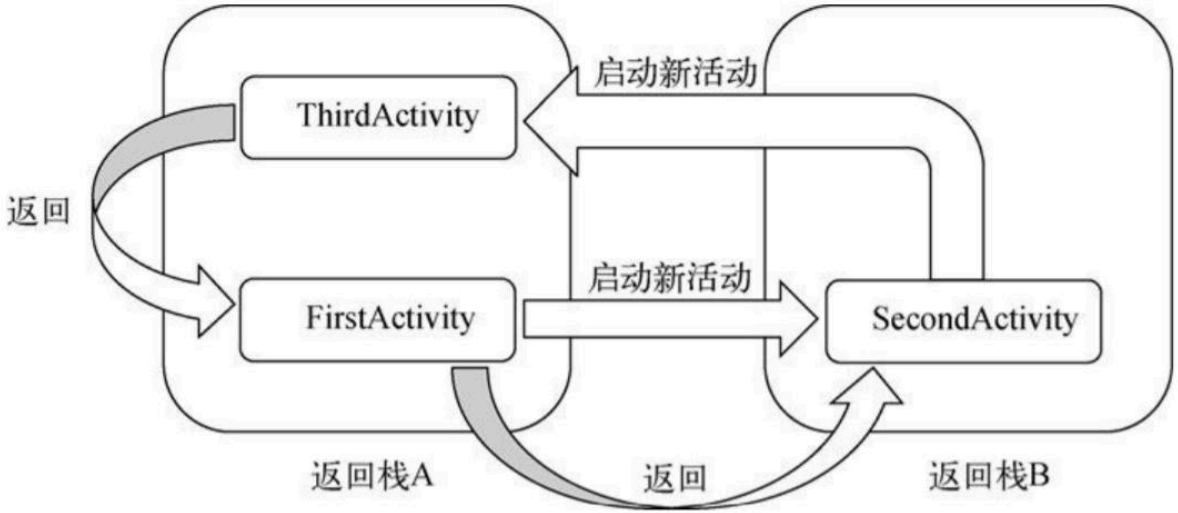
1.5 Activity 启动模式 — singleTask



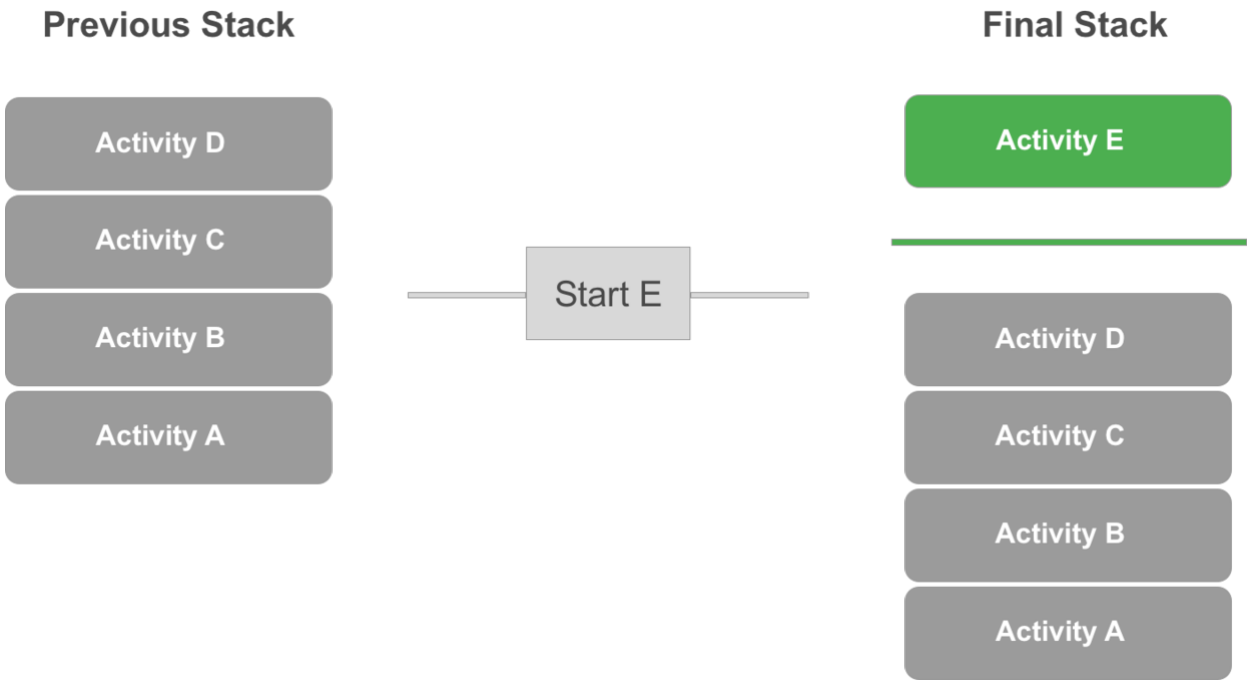
1.5 Activity 启动模式 — singleTask



1.5 Activity 启动模式 — singleInstance



1.5 Activity 启动模式 — singleInstance



UI

https://developer.android.com/guide/topics/ui/overview.html?hl=zh-cn

UI

- 如何编写程序界面
- 常用控件
- 基本布局

2.1 常用控件

- UI控件

TextView、ImageView、Button、ProgressBar、ScrollView、Toast

- UI布局

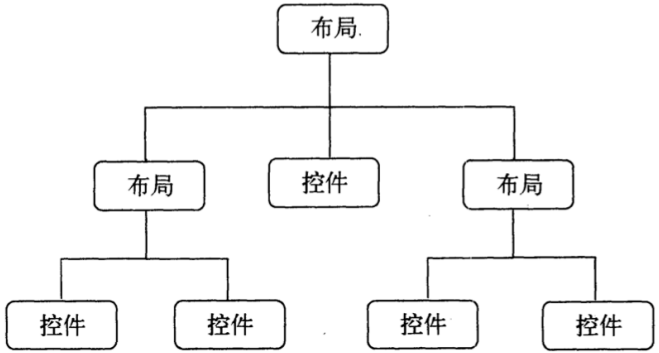
LinearLayout 、 RelativeLayout、FrameLayout

想看一共有哪些控件可以看design

瞅瞅更多：http://hukai.me/android-training-course-in-chinese/basics/firstapp/building-ui.html

2.1 如何编写程序界面

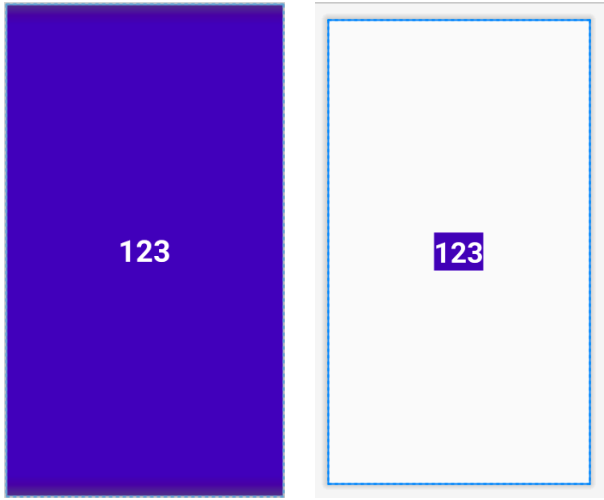
```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World" />

</FrameLayout>
```



如何编写程序界面 — wrap_content 和 match_parent

- layout_width：宽度
- layout_height：高度

- wrap_content：自身内容一样长
- match_parent：和父组件一样长

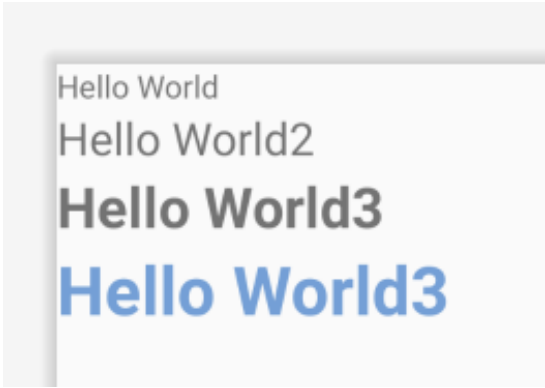


2.2 常用控件

- TextView
- EditText
- ImageView
- AlertDialog

2.2 常用控件 — TextView

```xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World2"
        android:textSize="20sp" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World3"
        android:textSize="25sp"
        android:textStyle="bold" />

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World3"
        android:textSize="30sp"
        android:textStyle="bold"
        android:textColor="#699fdb" />

</LinearLayout>
```



2.2 常用控件 — TextView

android:text	设置文本
android:textColor	设置字体颜色
android:textSize	设置字体大小
android:gravity	设置文本位置，如设置成“center”，文本将居中显示。
android:maxLength	限制显示的文本长度，超出部分不显示。
android:lines	设置文本的行数，设置两行就显示两行，即使第二行没有数据。
android:maxLines	设置文本的最大显示行数，与width或者layout_width结合使用，超出部分自动换行，超出行数将不显示。
android:minLines	设置文本的最小行数，与lines类似。

常用控件 — EditText


```xml
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="请输入内容"/>

</FrameLayout>
```

常用控件 — EditText

❑ 监听输入内容变化

```
mEditView.addTextChangedListener(new TextWatcher() {
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
        Log.i(TAG,  msg: "beforeTextChanged: " + s);
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        Log.i(TAG,  msg: "onTextChanged: " + s);
    }

    @Override
    public void afterTextChanged(Editable s) {
        Log.i(TAG,  msg: "afterTextChanged: " + s);
    }
});
```

2.2 常用控件 — ImageView

android:src

```
<ImageView
    android:id="@+id/image"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/icon_search" />
```

setImageResource

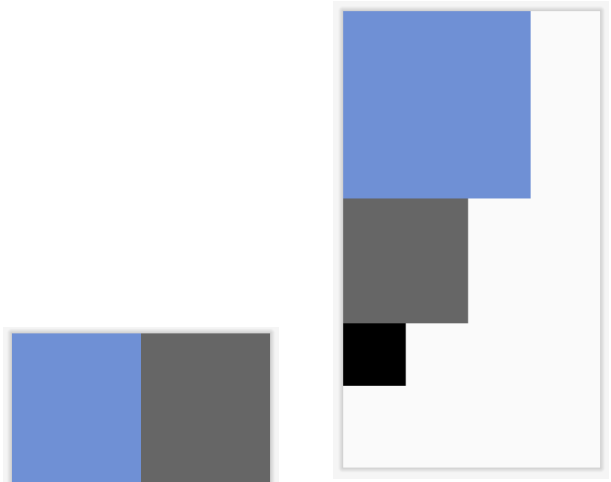
```
mImageView.setImageResource(R.drawable.icon_search);
```

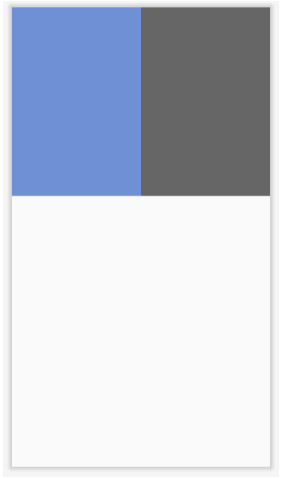
2.3 基本布局

- FrameLayout
- LinearLayout
- RelativeLayout
- ScrollView

2.3 基本布局 — LinearLayout

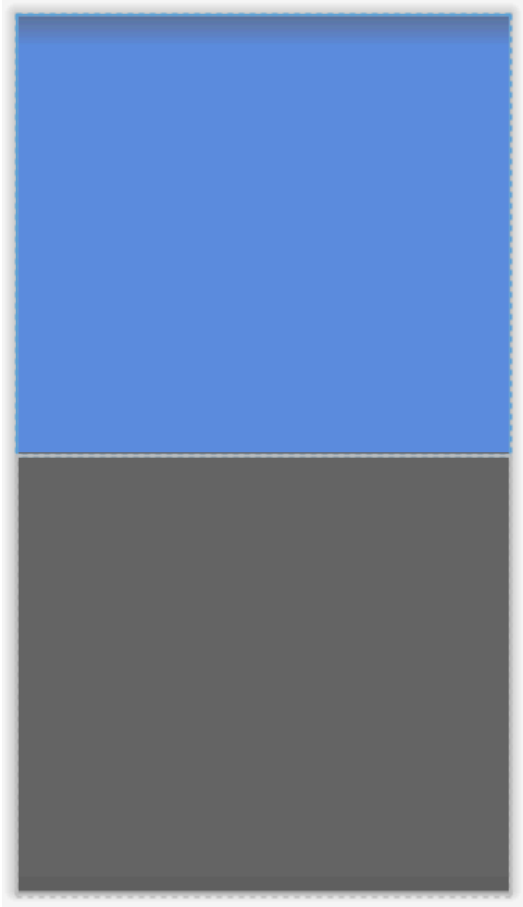
- 线性布局
- 支持水平和垂直布局
- 支持设置布局权重





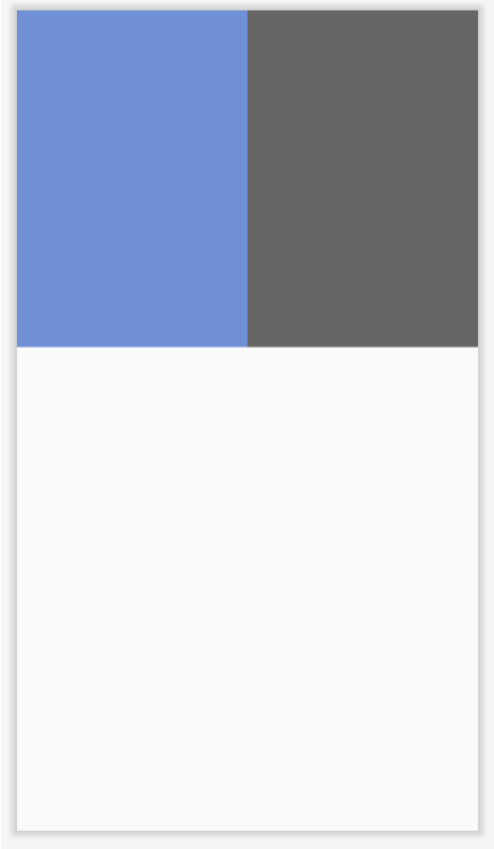
2.3 基本布局 — LinearLayout

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <View
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:background="#688fdb" />

    <View
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:background="#666666" />

</LinearLayout>
```



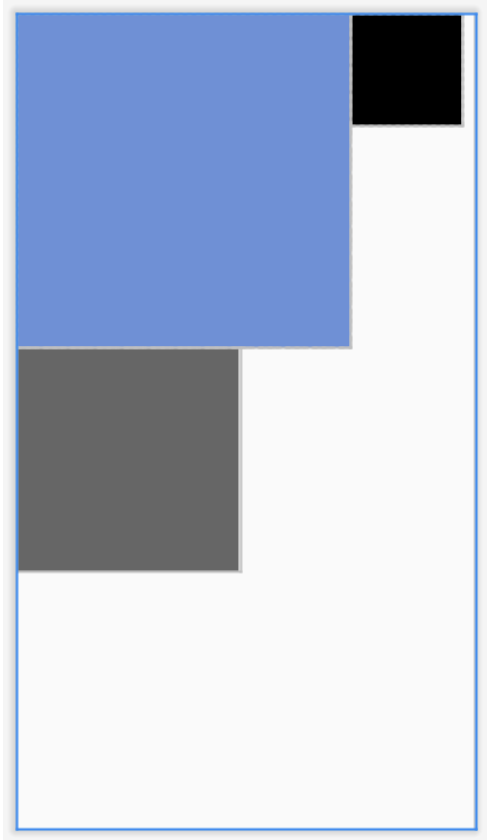
2.3 基本布局 — LinearLayout

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">

    <View
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:background="#688fdb" />

    <View
        android:layout_weight="1"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:background="#666666" />

</LinearLayout>
```



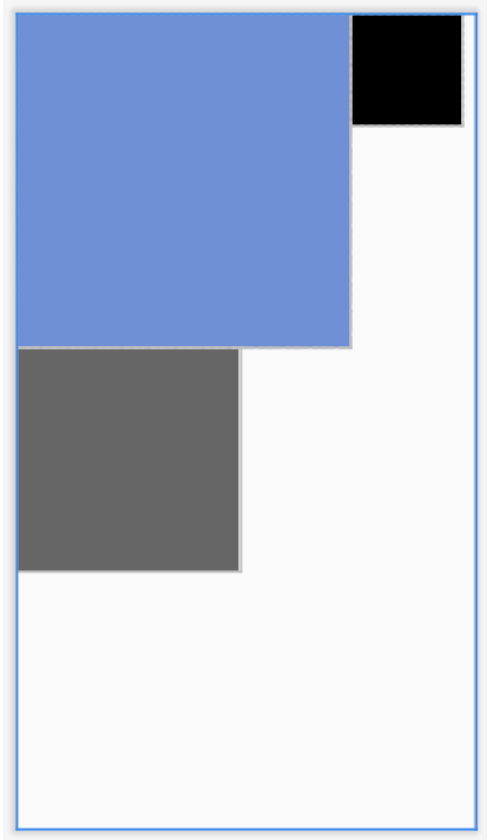
2.3 基本布局 — RelativeLayout

- 相对布局
- 相对兄弟 View 布局
- 相对父视图布局



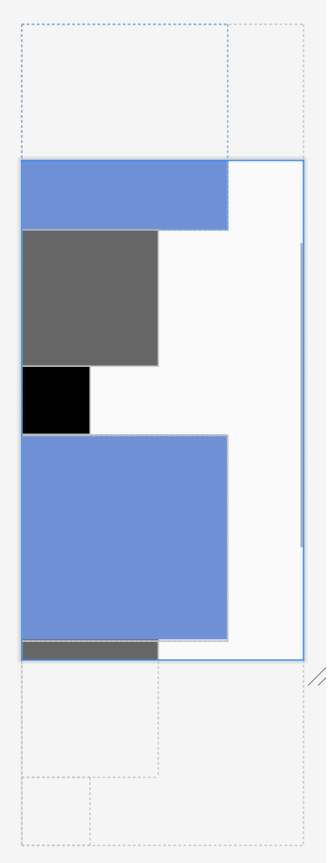
2.3 基本布局 — RelativeLayout

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <View
        android:id="@+id/view0"
        android:layout_width="300dp"
        android:layout_height="300dp"
        android:background="#688fdb" />

    <View
        android:layout_width="200dp"
        android:layout_height="200dp"
        android:layout_below="@id/view0"
        android:background="#666666" />

    <View
        android:layout_width="100dp"
        android:layout_height="100dp"
        android:layout_toRightOf="@id/view0"
        android:background="#000" />

</RelativeLayout>
```



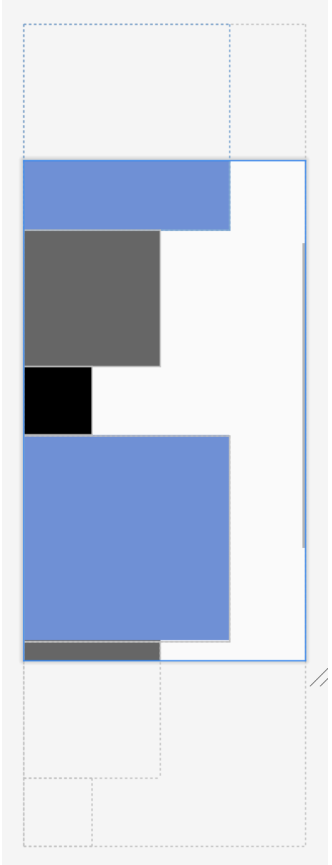
2.3 基本布局 — ScrollView

- 滚动布局
- 直接子 View 只能有一个



2.3 基本布局 — ScrollView

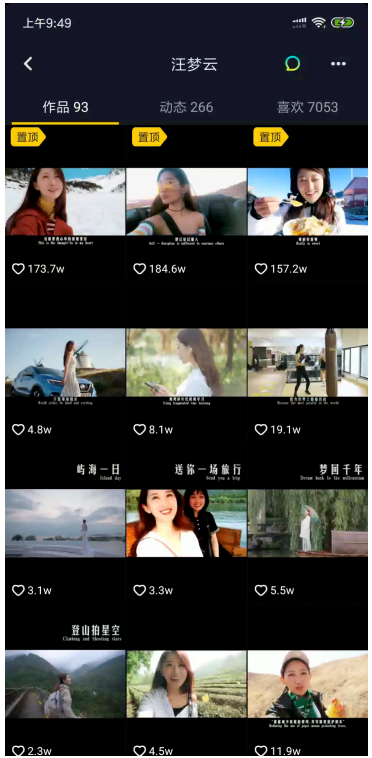
```xml
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <View
            android:layout_width="300dp"
            android:layout_height="300dp"
            android:background="#688fdb" />

        <View
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:background="#666666" />

        <View
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:background="#000" />

        <View
            android:layout_width="300dp"
            android:layout_height="300dp"
            android:background="#688fdb" />

        <View
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:background="#666666" />

        <View
            android:layout_width="100dp"
            android:layout_height="100dp"
            android:background="#000" />

    </LinearLayout>

</ScrollView>
```



RecyclerView

https://developer.android.com/guide/topics/ui/layout/recyclerview

RecyclerView



3.1 Why RecyclerView ?

- 之前学过的知识想做出一个抖音的消息页面可以吗?

答：可以，如何做？（ScrollView+N个Linearlayout 。。。）

- 上面的做法有何不妥?

答：数据过多时，内存有限，性能欠佳，直观感受->滑动很卡,严重丢帧！！！，甚至崩溃

- 更优解?

RecyclerView 应运而生

3.2 RecyclerView 定义

- 什么是 RecycleView?

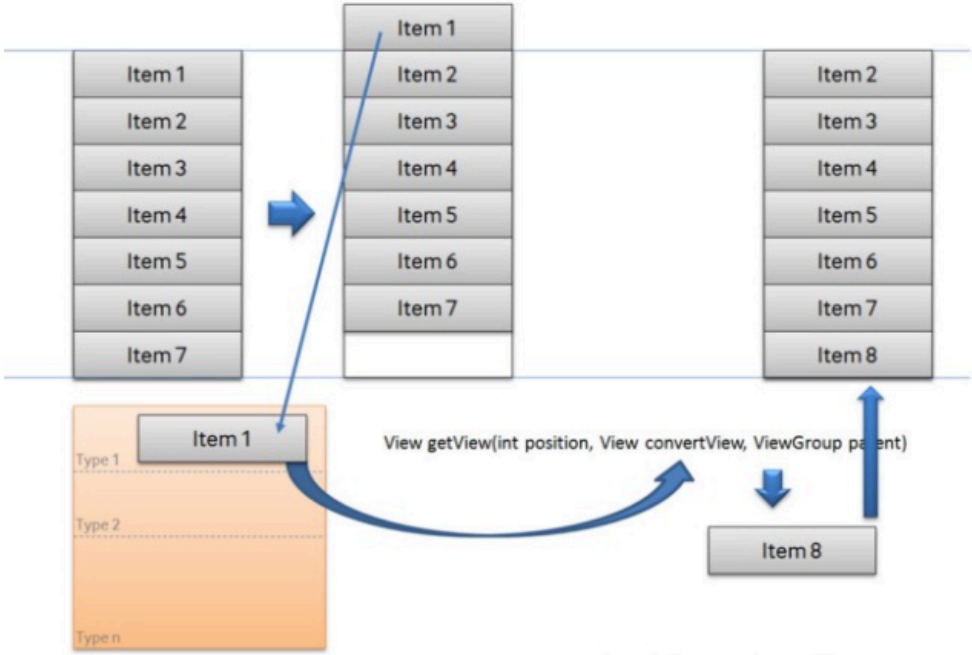
1.强大的滚动控件

2.展示大量的数据

3.提供数据和item布局，动态生成列表

4. 缓存-重用机制，提升性能

回收元素，重用视图



https://developer.android.com/guide/topics/ui/layout/recyclerview?hl=zh-cn

3.3 RecyclerView — 基本用法

- 添加依赖到build.gradle文件

```
implementation 'com.android.support:recyclerview-v7:28.0.0'
```

3.3 RecyclerView — 基本用法

- 在布局文件中添加RecyclerView控件

activity_search.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</LinearLayout>
```

3.3 RecyclerView — 基本用法

- 创建Item的布局文件


```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/andr
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="80dp"
    android:paddingLeft="16dp"
    android:paddingRight="16dp">

    <TextView
        android:id="@+id/tv_index"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="18sp"
        android:textStyle="bold|italic"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tv_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/white"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toRightOf="@+id/tv_index"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tv_hot"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@android:color/darker_gray"
        android:textSize="14sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3.3 RecyclerView — 基本用法

- 继承ViewHolder
- 编写自己的ViewHolder

设计item的外观和点击行为

ViewHolder作用：

避免频繁的inflate view调用

减少findViewById()的使用

从而达到节省资源提高运行效率的目的

```
@Override
public MyAdapter.MyViewHolder onCreateViewHolder(ViewGroup parent,
                                                 int viewType) {
    return new MyViewHolder(LayoutInflater.from(parent.getContext())
            .inflate(R.layout.recycler_item, parent,  attachToRoot: false));
}
```

```
public static class MyViewHolder extends RecyclerView.ViewHolder {
    private TextView tvIndex;
    private TextView tvTitle;
    private TextView tvHot;
    private View contentView;


    public MyViewHolder(View v) {
        super(v);
        contentView = v;
        tvIndex = v.findViewById(R.id.tv_index);
        tvTitle = v.findViewById(R.id.tv_title);
        tvHot = v.findViewById(R.id.tv_hot);
    }

    public void onBind(int position, TestData data) {
        tvIndex.setText(new StringBuilder().append(position).append(".  ").toString())
        tvTitle.setText(data.title);
        tvHot.setText(data.hot);
        if (position < 3) {
            tvIndex.setTextColor(Color.parseColor( colorString: "#FFD700"));
        } else {
            tvIndex.setTextColor(Color.parseColor( colorString: "#FFFFFF"));
        }
    }

    public void setOnClickListener(View.OnClickListener listener) {
        if (listener != null) {
            contentView.setOnClickListener(listener);
        }
    }

    public void setOnLongClickListener(View.OnLongClickListener listener) {
        if (listener != null) {
            contentView.setOnLongClickListener(listener);
        }
    }
```

3.3 RecyclerView — 基本用法

- 创建Adapter，将视图绑定到数据

1. 数据集
2. getItemCount

   获取数据集大小
3. onCreateViewHolder

   需要提供类型为viewType的

   新ViewHolder时会回调
4. onBindViewHolder

   当展示该position的item时会回调

   ViewHolder持有该position的view

```
public class MyAdapter extends RecyclerView.Adapter<MyAdapter.MyViewHolder> {
    private List<TestData> mDataset = new ArrayList<>();  1
    private IOnItemClickListener mItemClickListener;

    @Override
    public MyAdapter.MyViewHolder onCreateViewHolder(ViewGroup parent,    2
                                                     int viewType) {
        return new MyViewHolder(LayoutInflater.from(parent.getContext())
                .inflate(R.layout.recycler_item, parent,  attachToRoot: false));
    }

    @Override
    public void onBindViewHolder(MyViewHolder holder, final int position) {  3
        holder.onBind(position, mDataset.get(position));
        holder.setOnClickListener((v) → {
                if (mItemClickListener != null) {
                    mItemClickListener.onItemCLick(position, mDataset.get(position));
                }
        });
        holder.setOnLongClickListener((v) → {
                if (mItemClickListener != null) {
                    mItemClickListener.onItemLongCLick(position, mDataset.get(position));
                }
                return false;
        });
    }

    // Return the size of your dataset (invoked by the layout manager)
    @Override
    public int getItemCount() { return mDataset.size(); }  4
```

3.3 RecyclerView — 基本用法

- 1.利用findViewById获取RecyclerView实例
- 2.指定LayoutManager布局管理器的类别
- 3.指定Adapter
- 4.设置数据源

布局管理器

控制item的布局方式，横向、竖向以及瀑布式

- LinearLayoutManager 线性布局管理器
- GridLayoutManager 网格布局管理器
- StaggeredGridLayoutManager 错列网格布局管理器

```
private void initView() {
    //获取实例
    recyclerView = findViewById(R.id.recycler);
    //更改数据时不会变更宽高
    recyclerView.setHasFixedSize(true);
    //创建线性布局管理器
    layoutManager = new LinearLayoutManager( context: this);
    //创建格网布局管理器
    gridLayoutManager = new GridLayoutManager( context: this,  spanCount: 2);
    //设置布局管理器
    recyclerView.setLayoutManager(layoutManager);
    //创建Adapter
    mAdapter = new MyAdapter(TestDataSet.getData());
    //设置Adapter每个item的点击事件
    mAdapter.setOnItemClickListener(this);
    //设置Adapter
    recyclerView.setAdapter(mAdapter);
    //分割线
    LinearItemDecoration itemDecoration = new LinearItemDecoration(Color.BLUE);
      recyclerView.addItemDecoration(itemDecoration);
    recyclerView.addItemDecoration(new DividerItemDecoration( context: this, LinearLayoutManager.VERTICAL));
    //动画
    DefaultItemAnimator animator = new DefaultItemAnimator();
    animator.setAddDuration(3000);
    recyclerView.setItemAnimator(animator);
}
```

3.4 热搜榜列表页

THANKS