Android程序设计

事件和活动

2019.5.12

isszym　sysu.edu.cn

内容

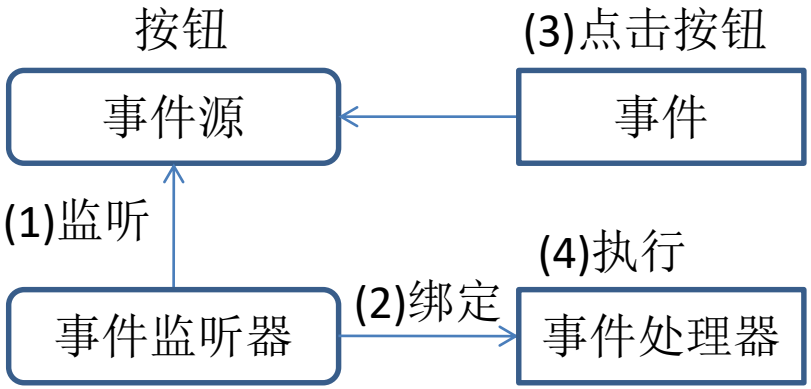
事件(Event)
启动Activity
用Bundle对象传递数据
Intent
Activity的加载方式
Activity的生命周期
Android 应用程序的状态
拨打电话
Fragment
PreferenceFragment
附录1、安卓权限设置

事件(Event)

OnTouch

- 事件处理机制

对事件源(Event Source)绑定一个事件监听器(Event Listener)，一旦事件(Event)发生，事件监听器就会执行事件处理器(Event Handler)。例如：点击（事件）鼠标（事件源）后执行一段程序（事件处理器）。



四种方法
- 把事件处理器直接绑定到标签上
- 使用监听类（内部类）作为事件监听器
- 使用匿名类作为事件监听器
- 把Activity实例作为事件监听器

- 项目：NewEvent

本项目展示了四种事件绑定方法：

(1)把事件处理器直接在布局文件中绑定

```
android:onClick="clickHandler1"

public void clickHandler1(View source) {  ...  }
```

(2)把接口OnClickListener的子类MyClickListener的一个实例设置为事件监听器。

```
btn.setOnClickListener(new MyClickListener());

class MyClickListener implements OnClickListener {
    @Override
    public void onClick(View v){ ... }
}
```

(3)与上面不同的是试了用匿名子类的一个实例。

```
btn.setOnClickListener(new OnClickListener(){
     @Override
     public void onClick(View v){ ... }
});
```

(4)把Activity的实例作为接口OnClickListener的子类的实例。

```
public class MainActivity extends AppCompatActivity implements OnClickListener {
    @Override
    public void onClick(View v){ ... }
   protected void onCreate(){ ... btn.setOnClickListener(this);  }
}
```

MainActivity.java

```
public class MainActivity extends AppCompatActivity
     implements OnClickListener {
    TextView tv;
    @Override
    public void onClick(View v){
        tv.setText("Button4 is clicked!");
    }
    // 把事件处理器直接绑定到标签上
    public void clickHandler1(View source) {
        Toast.makeText(this, "Button1 is clicked!",
            Toast.LENGTH_SHORT).show();
    }

    class MyClickListener implements OnClickListener {
      @Override
      public void onClick(View v){
         Toast.makeText(MainActivity.this,
               "Button2 is clicked!",
              Toast.LENGTH_SHORT).show();
        }
    }
```

```
@Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv = (TextView) findViewById(R.id.textView);
        // 使用监听类作为事件监听器
        Button btn2 = (Button) findViewById(R.id.button2);
        btn2.setOnClickListener(new MyClickListener());

        // 使用匿名子类作为事件监听器
        Button btn3 = (Button) findViewById(R.id.button3);
        btn3.setOnClickListener(new OnClickListener(){
            @Override
            public void onClick(View v){
                Toast.makeText(MainActivity.this,
                        "Button3 is clicked!",
                        Toast.LENGTH_SHORT).show();
            }
        });
       // 把Activity本身作为监听类
        Button btn4 = (Button) findViewById(R.id.button4);
        btn4.setOnClickListener(this);
    }

}
```

activity_main.xml

```
<RelativeLayout xmlns:android
   =http://schemas.android.com/apk/res/android
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:id="@+id/activity_main">
  <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello World!"
      android:id="@+id/textView" />
  <Button
      android:text="Button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="33dp"
      android:id="@+id/button1"
      android:onClick="clickHandler1"
      android:layout_below="@+id/textView"
      android:layout_alignParentStart="true"
  />
  <Button
      android:text="Button2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginTop="34dp"
      android:id="@+id/button2"
      android:layout_below="@+id/button1"
      android:layout_alignParentStart="true"
  />
  <Button
      android:text="Button3"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:id="@+id/button3"
      android:layout_centerVertical="true"
      android:layout_alignParentStart="true"
  />
  <Button
      android:text="Button4"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_below="@+id/button3"
      android:layout_alignParentStart="true"
      android:layout_marginTop="42dp"
      android:id="@+id/button4"
   />
</RelativeLayout>
```

启动Activity

- **如何启动另一个Activity?**
  - **startActivity(Intent intent)**
    用条件查找并启动另一个Acitivity，并可以带数据过去。
  - **startActivityForResult(Intent intent, int requestCode)**
    用请求码启动另一个Acitivity，重写方法onActivityResult()获取返回的结果。

- **启动条件有哪些?**
  - **指明Activity所在包的包名和它的类名。同一个app可以直接用Activity的类启动。**
  - **给出Activity的过滤条件。**

- **如何结束Activity?**
  - **finish()**
    结束当前的Activity。
  - **finishActivity(int requestCode)**
    结束用请求码启动的Activity。该Aciivity不能用finish()直接结束，因为会造成取不到返回值.

- **Activity的属性(在AndoidManifest.xml中)**

```
<activity
    android:name=".MainActivity"                    类名（省略了包名）
    android:icon="@mipmap/ic_launcher"              图标
    android:label="@string/app_name"                标签
    android:exported="true"                         是否允许被其他应用程序调用
    android:launchMode="singleTop">                 加载模式(后面有详细介绍)
    <intent-filter>
        <action android:name="com.example.startact.THIRDACT" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

  - intent-filter定义被其它Activity启动的条件，action给出了主要条件，category给出附加条件，实际上就是给出要匹配的字符串。也就是说，只要给出了这些字符串就可以启动这个Activity。
  - exported主要用于表示是否允许其他应用调用当前Activity。如果包含有intent-filter，其取值默认为true; 没有intent-filter默认为false。

- **项目名：NewStartActivity**

主Activity采用三种方法启动另外两个Activity，还可以启动另一个app的Activity：

（1）用Activity类启动第二个Activity

```
Intent intent = new Intent(this,SecondActivity.class);
intent.putExtra("parm","*Hello From Activity 1!");//参数
startActivity(intent);
```

（2）用Action启动第二个Activity

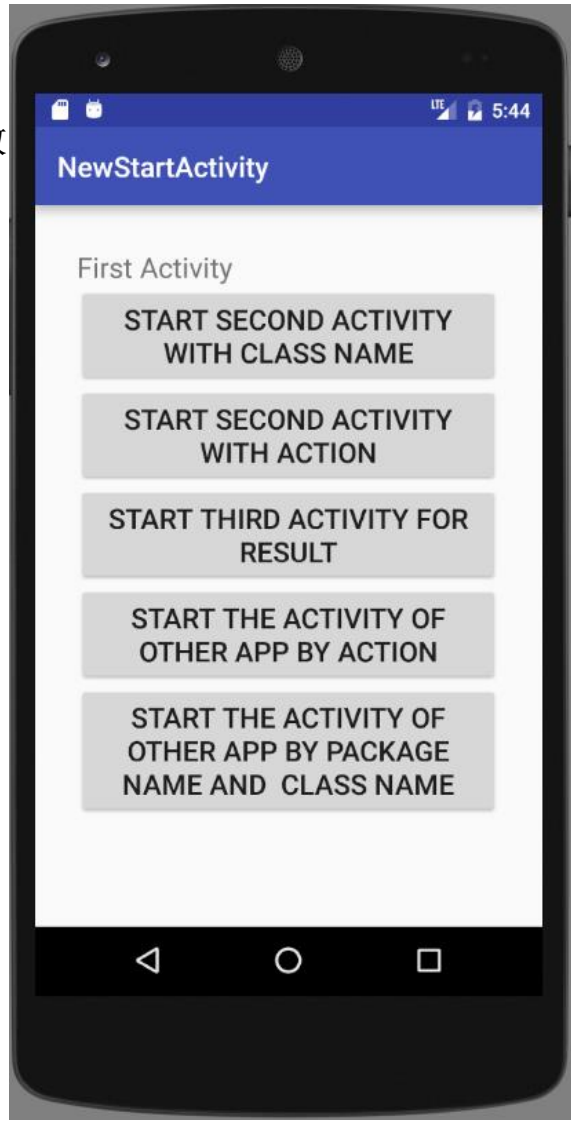
```
Intent intent = new Intent();
intent.putExtra("parm","**Hello From Activity 1!");
intent.setAction("com.example.startact.SECONDACT");
startActivity(intent);
```

简化：第1句改为下面格式可以省略第3句

```
Intent intent = new Intent("com.example.startact.SECONDACT");
```

（3）用Action启动第三个Activity，并要求返回值

```
Intent intent = new Intent("com.example.startact.THIRDACT");
intent.putExtra("parm","***Hello From Activity 1!");
startActivityForResult(intent,1);
```



（4）用Action启动另一个app的Activity。

```
Intent intent  = new Intent("com.example.startedact.MAIN");    // 参数：Action
intent.putExtra("parm","****Hello From Activity 1!");
startActivity(intent);
```

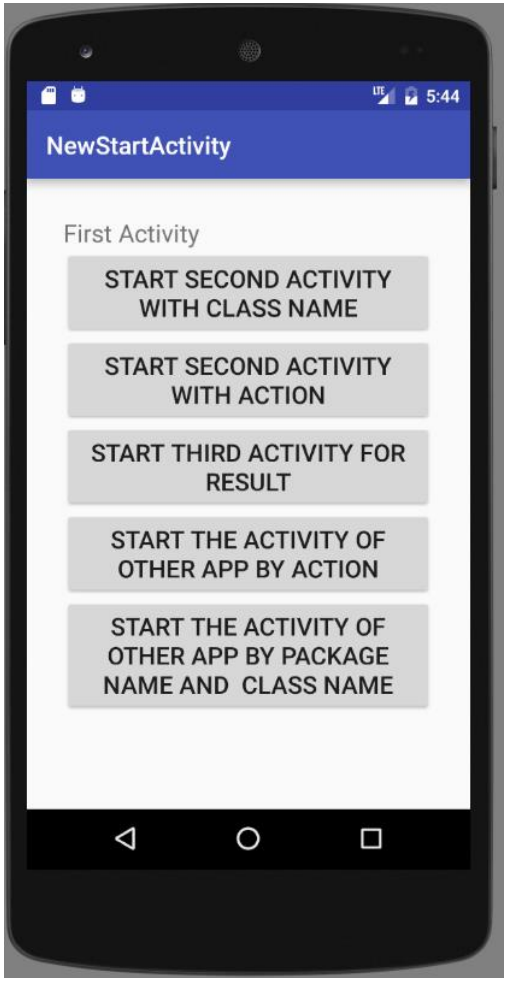
（5）包名和类名启动另一个app的Activity。

```
Intent intent = new Intent();
intent.setClassName("com.example.isszym.newstartedactivity",
                    "com.example.isszym.newstartedactivity.MainActivity");
intent.putExtra("parm","*****Hello From Activity 1!");
startActivity(intent);
```

或

```
Intent intent = new Intent();
ComponentName cn = new ComponentName("com.example.isszym.newstartedactivity",
                           "com.example.isszym.newstartedactivity.MainActivity");
intent.setComponent(cn);
intent.putExtra("parm","*****Hello From Activity 1!");
startActivity(intent);
```

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
    public void clickHandler1(View source) {
        Intent intent = new Intent(this,SecondActivity.class);
        intent.putExtra("parm","*Hello From Activity 1!");
        startActivity(intent);
    }
    public void clickHandler2(View source) {
        Intent intent = new Intent();
        intent.putExtra("parm","**Hello From Activity 1!");
        intent.setAction("com.example.startact.SECONDACT");
        startActivity(intent);
    }
    public void clickHandler3(View source) {
        Intent intent = new Intent();
        intent.putExtra("parm","***Hello From Activity 1!");
        intent.setAction("com.example.startact.THIRDACT");
        startActivityForResult(intent,1);
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent intent){
        if(requestCode==1 && resultCode==2){
            TextView tv = (TextView) findViewById(R.id.infoView);
            tv.setText(intent.getStringExtra("res"));
            finishActivity(1); //结束以启动码启动的Activity
        }
    }
```

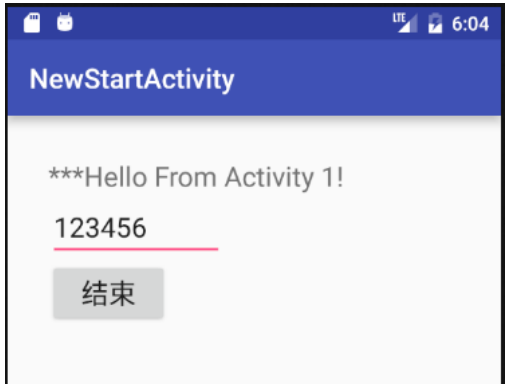


```
    public void clickHandler4(View source) {
        Intent intent  = new Intent("com.example.startedact.MAIN");
        intent.putExtra("parm","****Hello From Activity 1!");
        startActivity(intent);
    }
    public void clickHandler5(View source) {
        Intent intent = new Intent();
        intent.setClassName("com.example.isszym.newstartedactivity",
                "com.example.isszym.newstartedactivity.MainActivity");
        intent.putExtra("parm","*****Hello From Activity 1!");
        startActivity(intent);
    }
}
```

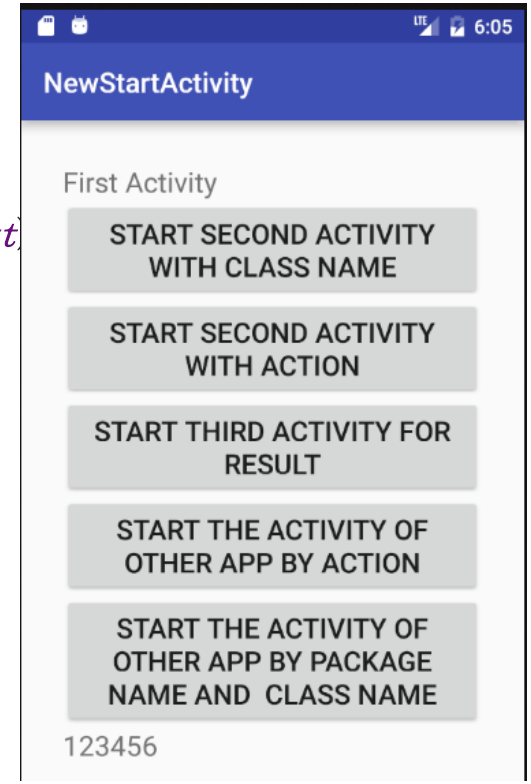
```
public class SecondActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_second);
        Intent intent = getIntent();
        String parm= intent.getStringExtra("parm");
        TextView tv = (TextView) findViewById(R.id.textView);
        tv.setText(parm);
    }

    public void clickHandler(View source) {
        finish();
    }
}
```

在MainActivity中，点击第一个按钮得到右上的图，点击结束按钮会回到MainActivity，点击第二按钮得到右下的图，点击结束按钮会回到MainActivity 。

```
public class ThirdActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_third);
        Intent intent = getIntent();
        String parm= intent.getStringExtra("parm");
        TextView tv = (TextView) findViewById(R.id.textView);
        tv.setText(parm);
        Button bn3 = (Button) findViewById(R.id.btn1);
        bn3.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View source) {
                Intent intent = getIntent();
                EditText et = (EditText) findViewById(R.id.editText
                intent.putExtra("res",et.getText().toString());
                setResult(2,intent);  //response Code
                finish();
            }
        });
    }
}
```



点击结束按钮得到下图



activity_main.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"
    android:padding="30dp">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:textSize="20sp"
        android:text="First Activity" />
    <Button
        android:text="Start second activity with class name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:onClick="clickHandler1"/>
    <Button
        android:text="Start second activity with action"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:onClick="clickHandler2"/>
```

```
    <Button
        android:text="Start third activity for result"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:onClick="clickHandler3"/>
    <Button
        android:text="Start the activity of other app by action"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:onClick="clickHandler4"/>
    <Button
        android:text="Start the activity of other app by package name and class name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:onClick="clickHandler5"/>
    <TextView
        android:text=""
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:id="@+id/infoView" />
</LinearLayout>
```

activity_second.xml

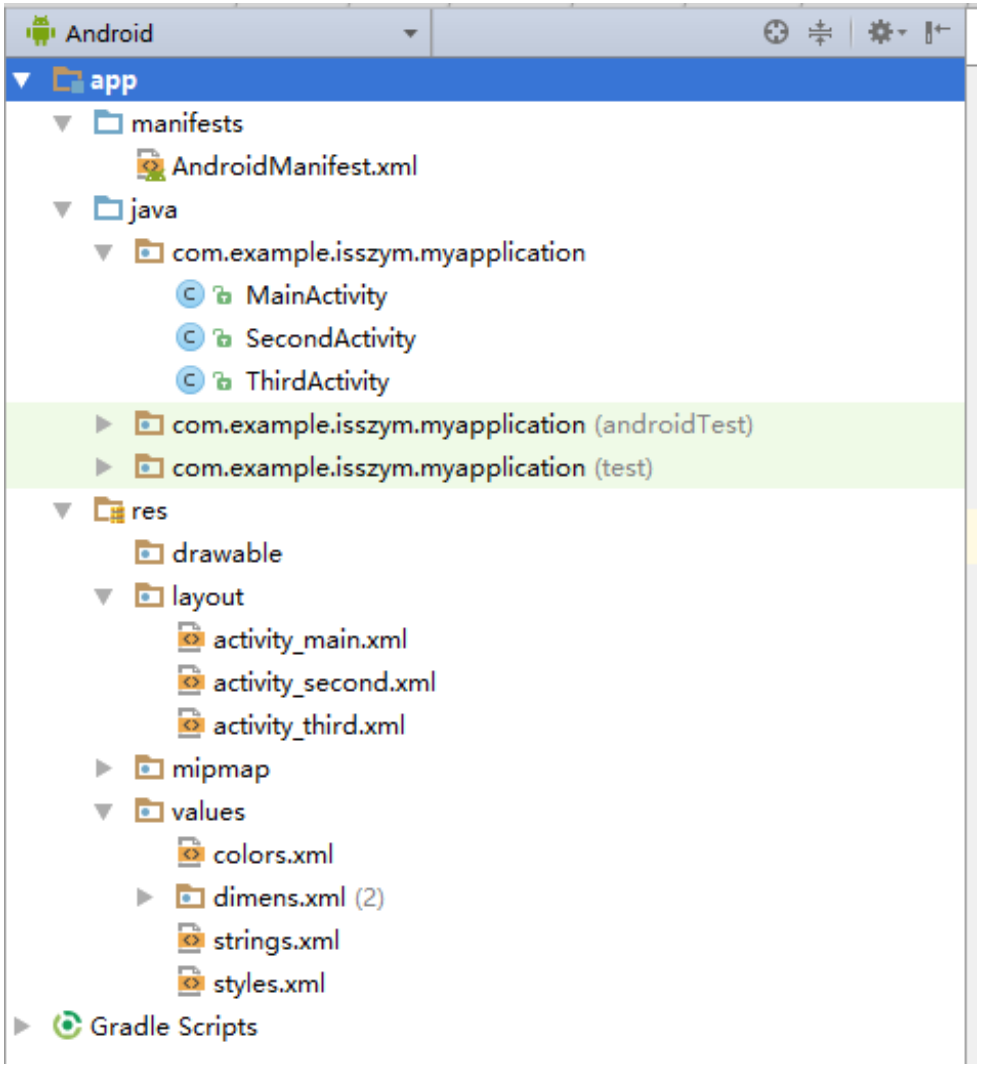
```
?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="30dp">
    <TextView
        android:text="Second Activity"
        android:textSize="20sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:text=""
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:id="@+id/textView" />
    <Button
        android:text="结束"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="35dp"
        android:textSize="20sp"
        android:onClick="clickHandler"  />
</LinearLayout>
```

activity_third.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"
    android:padding="30dp">
    <TextView
        android:text="Third Activity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:id="@+id/textView"/>
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:id="@+id/editText"
        android:hint="输入返回结果"/>
    <Button
        android:text="结束"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="20sp"
        android:id="@+id/btn1"/>
</LinearLayout>
```

AndroidManifest.xml

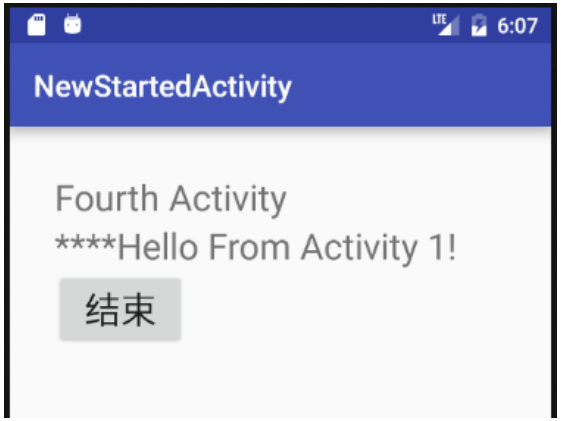
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.isszym.myapplication">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="NewStartActivity"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".SecondActivity">
            <intent-filter>
                <action android:name="com.example.startact.SECONDACT" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
        <activity android:name=".ThirdActivity">
            <intent-filter>
                <action android:name="com.example.startact.THIRDACT" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```


Android
app
manifests
AndroidManifest.xml
java
com.example.isszym.myapplication
MainActivity
SecondActivity
ThirdActivity
com.example.isszym.myapplication (androidTest)
com.example.isszym.myapplication (test)
res
drawable
layout
activity_main.xml
activity_second.xml
activity_third.xml
mipmap
values
colors.xml
dimens.xml (2)
strings.xml
styles.xml
Gradle Scripts

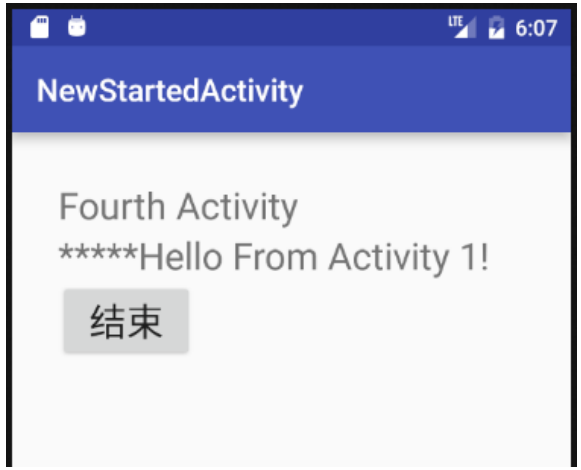
- 项目名：**NewStartedActivity**

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Intent intent = getIntent();
        String parm= intent.getStringExtra("parm");
        TextView tv = (TextView) findViewById(R.id.textView);
        tv.setText(parm);
    }

    public void clickHandler(View source) {
        finish();
    }
}
```

点击NewStartActivity的第四个按钮得到：



点击NewStartActivity的第五个按钮得到：



activity_main.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"
    android:padding="30dp">
    <TextView
        android:text="Fourth Activity"
        android:textSize="24sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <TextView
        android:text=""
        android:textSize="24sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/textView" />
    <Button
        android:text="结束"
        android:textSize="24sp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="clickHandler" />
</LinearLayout>
```

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.isszym.newstartedactivity">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="NewStartedActivity"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter>
                <action android:name="com.example.startedact.MAIN" />
                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

用Bundle对象传递数据

- Activity之间是通过Bundle来传递数据的，传递的数据可以是Int、Float、String、Array、ArrayList、Object等。
- 在一个Activity中用putExtra(key, value)设置intent的bundle对象中：

```
Intent intent = new Intent();
Bundle bundle = new Bundle();
bundle.putString("key1","value1");
intent.putExtras(bundle);
intent.setAction("com.example.startact.SECONDACT");
startActivity(intent);
```

简化:`intent.putExtra("key1","value1");`

- 在另一个Activity中用getXxxExtra(key)把值从intent的bundle对象中取出：

```
Intent intent = getIntent();
Bundle bundle = intent.getExtras();
String value = bundle.getString("key1");
```

简化为

```
Intent intent = getIntent();
String value = intent.getStringExtra("key1");
```

- **项目名: NewBundleTest**

展示了如何把一个对象从一个Activity传送给另一个Activity。

设置intent的bundle数据

```
Person p = new Person(name.getText().toString(),
          passwd.getText().toString(), gender);
Intent intent=new Intent(this, Main2Activity.class);
Bundle bundle = new Bundle();
bundle.putSerializable("person", p);
intent.putExtras(bundle);
startActivity(intent);
```

获取intent的bundle数据

```
// 获取启动该Result的Intent
Intent intent = getIntent();
// 通过Intent取出它所携带的Bundle数据包中的数据
Person p = (Person)
                intent.getSerializableExtra("person");
name.setText("您的用户名为：" + p.getName());
passwd.setText("您的密码为：" + p.getPass());
gender.setText("您的性别为：" + p.getGender());
```

Person. java

```
import java.io.Serializable;
public class Person implements Serializable{
   private static final long serialVersionUID = 1L;
   private Integer id;
   private String name;
   private String pass;
   private String gender;
   public Person(){}
   public Person(String name, String pass, String gender) {
      this.name = name;
      this.pass = pass;
      this.gender = gender;
   }
   public Integer getId(){ return id; }
   public void setId(Integer id){this.id = id;}
   public String getName(){ return name; }
   public void setName(String name){this.name = name;}
   public String getPass(){return pass;}
   public void setPass(String pass){this.pass = pass;}
   public String getGender(){ return gender; }
   public void setGender(String gender){ this.gender = gender;}

}
```

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button bn = (Button) findViewById(R.id.bn);
        bn.setOnClickListener(new View.OnClickListener() {
          public void onClick(View v) {
            EditText name = (EditText)findViewById(R.id.name);
            EditText passwd = (EditText)findViewById(R.id.passwd);
            RadioButton male = (RadioButton) findViewById(R.id.male);
            String gender = male.isChecked() ? "男 " : "女";
            Person p = new Person(name.getText().toString(), passwd
                        .getText().toString(), gender);
            Intent intent=new Intent(this, Main2Activity.class);
            Bundle bundle = new Bundle();
            bundle.putSerializable("person", p);
            intent.putExtras(bundle);
            startActivity(intent);
          }
        });
    }
}
```

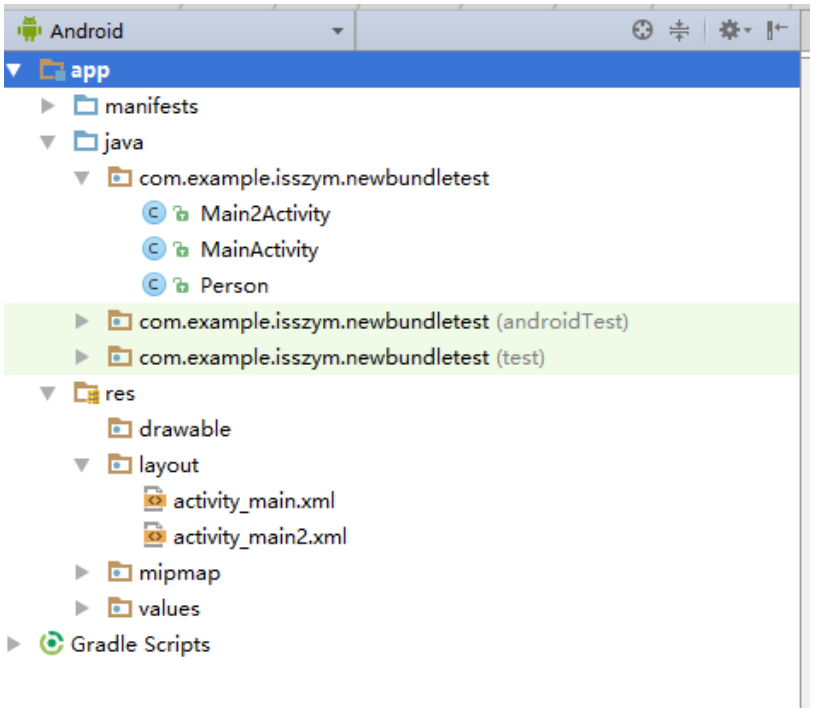
可以简化为

```
intent.putExtras("person", p);
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android
="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
    >
<TextView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="请输入您的注册信息"
    android:textSize="20sp"
    />
<TableRow>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="用户名 : "
        android:textSize="16sp"
        />
    <!-- 定义一个EditText，用于收集用户的帐号 -->
    <EditText
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="请填写想注册的帐号"
        android:selectAllOnFocus="true"
        />
</TableRow>
<TableRow>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="密码 : "
        android:textSize="16sp"
        />
    <!-- 用于收集用户的密码 -->
    <EditText
        android:id="@+id/passwd"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:password="true"
        android:selectAllOnFocus="true"
        />
</TableRow>
<TableRow>
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="性别 : "
        android:textSize="16sp"
        />
```

```
    <!-- 定义一组单选框，用于收集用户注册的性别 -->
    <RadioGroup
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        >
        <RadioButton
            android:id="@+id/male"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="男"
            android:textSize="16sp"
            />
        <RadioButton
            android:id="@+id/female"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="女"
            android:textSize="16sp"
            />
    </RadioGroup>
  </TableRow>
  <Button
    android:id="@+id/bn"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="注册"
    android:textSize="16sp"
    />
</TableLayout>
```



```java
public class Main2Activity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        TextView name
                = (TextView) findViewById(R.id.name);
        TextView passwd
                = (TextView) findViewById(R.id.passwd);
        TextView gender
                = (TextView) findViewById(R.id.gender);
        // 获取启动该Result的Intent
        Intent intent = getIntent();
        // 通过Intent取出它所携带的Bundle数据包中的数据
        Person p = (Person)
                intent.getSerializableExtra("person");
        name.setText("您的用户名为："
                + p.getName());
        passwd.setText("您的密码为：" + p.getPass());
        gender.setText("您的性别为：" + p.getGender());
    }
}
```

activity_main2.xml

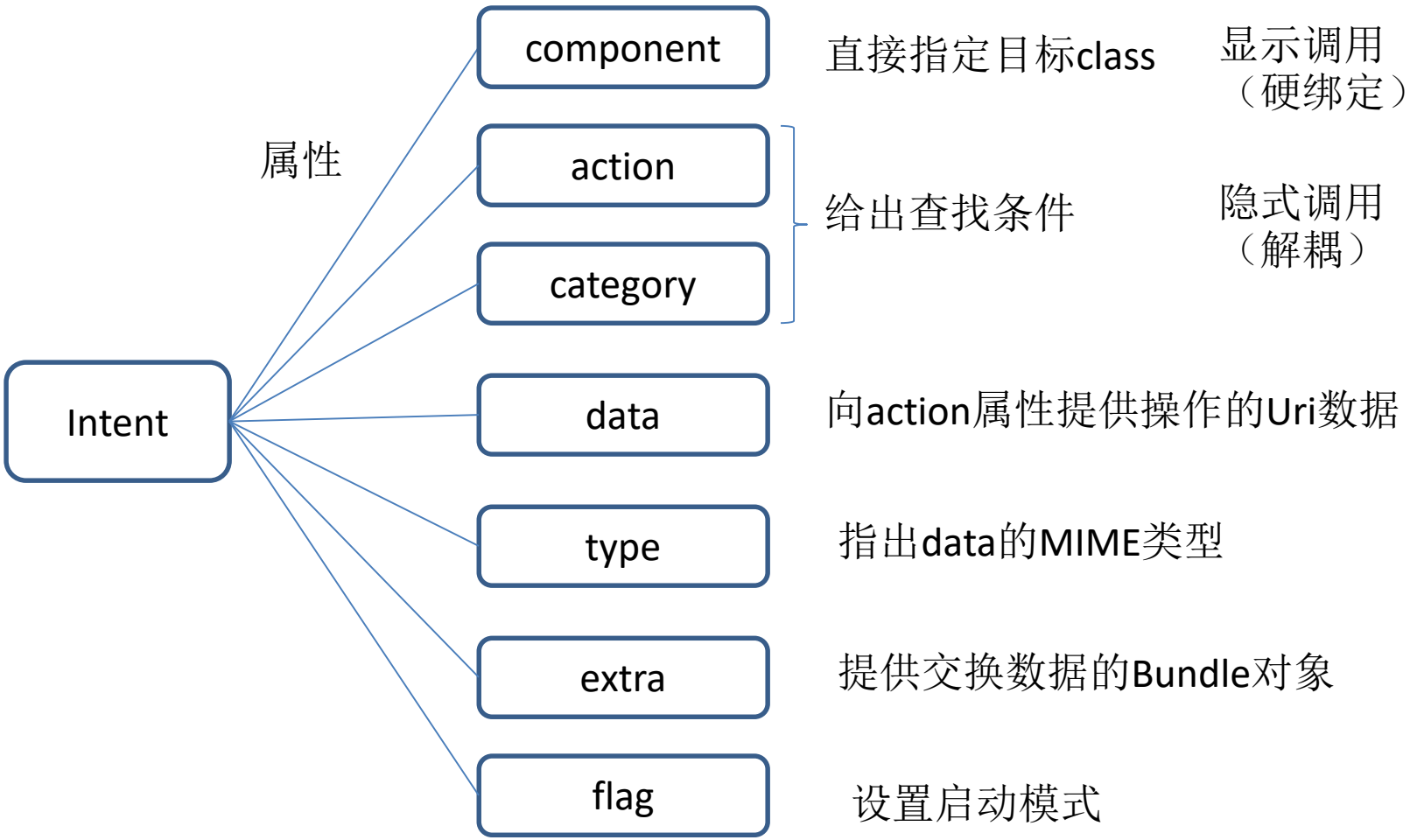
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    >
    <!-- 定义三个TextView，用于显示用户输入的数据 -->
    <TextView
        android:id="@+id/name"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        />
    <TextView
        android:id="@+id/passwd"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        />
    <TextView
        android:id="@+id/gender"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        />
</LinearLayout>
```

AndroidManifest.xml

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

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

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Main2Activity"></activity>
    </application>

</manifest>
```

Intent

- Intent表示启动的“意图”，安卓系统根据Intent的属性来启动不同的组件。



```
public class Intent  implements Parcelable, Cloneable {
    private String mAction;
    private Uri mData;
    private String mType;
    private String mPackage;
    private ComponentName mComponent;
    private int mFlags;
    private ArraySet<String> mCategories;
    private Bundle mExtras;
    public Intent(Intent o) {
        this.mAction = o.mAction;
        this.mData = o.mData;
        ...
    }
    public Intent(String action) {
        setAction(action);
    }
    public Intent(String action, Uri uri) {
        setAction(action);
        mData = uri;
    }
    public Intent(Context packageContext, Class<?> cls) {
        mComponent = new ComponentName(packageContext, cls);
    }
    public Intent(String action, Uri uri, Context packageContext, Class<?> cls) {
        setAction(action);
        mData = uri;
        mComponent = new ComponentName(packageContext, cls);
    }
    ......
}
```

- **Component属性**

component属性(mComponent)用来设置要访问的组件(Activity等)的包名和类名，用于访问相同或不同的应用程序的组件。 mComponent是ComponentName的实例。

构造器：ComponentName(String pkg, String className)
ComponentName(Context pkg, String className)
ComponentName(Context pkg, Class<?> cls)

ComponentName.setClass方法：
setClass(String pkg, String className)
setClass(Context pkg, String className)
setClass(Context pkg, Class<?> cls)

```
ComponentName comp = new ComponentName(CurActitivy.this, Activity2.class);
Intent intent  = new Intent();
Intent.setComponent(comp);
startActivity(intent);
```

可以简化为

```
Intent intent  = new Intent(CurActitivy.this, Activity2.class);
startActivity(intent);
```

在Activity2中可以取到Intent中带来的Component：

```
ComponentName comp = getIntent().getComponent();
Toast.makeText(this, comp.getPackageName()+comp.getClassName(),
                    Toast.LENGTH_SHORT).show();
```

```
public class Intent  implements Parcelable, Cloneable {
    private ComponentName mComponent;
    public Intent setComponent(ComponentName component) {
        mComponent = component;
        return this;
    }
    public Intent setClassName(Context packageContext, String className) {
        mComponent = new ComponentName(packageContext, className);
        return this;
    }
    public Intent setClassName(String packageName, String className) {
        mComponent = new ComponentName(packageName, className);
        return this;
    }
    public Intent setClass(Context packageContext, Class<?> cls) {
        mComponent = new ComponentName(packageContext, cls);
        return this;
    }
}
```

- **action属性和category属性**

action属性(mAction)指出要做什么“动作”，category属性(mCategaries)用于指明该动作的附加信息。这个动作只是给出一个字符串，由系统通过查找所有组件(Activity等)的intent-filter去查找匹配的组件。

```
Intent intent  = new Intent();
Intent.setAction("com.group.action.ADDRBK_ACTION");
startActivity(intent);
```

或简化为

```
Intent intent  = new Intent("com.group.action.ADDRBK_ACTION");
startActivity(intent);
```

被启动的Actitivity：

```
<activity android:name=".AddrBook"><intent-filter>
    <action android:name="com.group.action.ADDRBK_ACTION" />
    <category android:name="android.intent.category.DEFAULT" />
</intent-filter></activity>
```

```
Intent intent  = new Intent();
Intent.setAction("com.group.action.ADDRBK_ACTION");
Intent.addCategory("com.group.action.ADDRBK_CATEGORY");
startActivity(intent);
```

被启动的Activity：

```
<activity android:name=".AddrBook"><intent-filter>
    <action android:name="com.group.action.ADDRBK_ACTION" />
    <category android:name=" com.group.action.ADDRBK_CATEGORY" />
</intent-filter></activity>
```

在被启动的Activity中获得action和category：

```
String action=getIntent().getAction();
Set<String> categories=getIntent().getCategory();
```

* AndroidManifest.xml可以包含多个<intent-filter>，一个<intent-filter>可以包含0~N个<action>元素、0~N个<categroy>元素、0~1个<data>元素。

* action名可以任意字符串，最好采用反转的公司域名以防重名。

```
public class Intent  implements Parcelable, Cloneable {
    private String mAction;
    private ArraySet<String> mCategories;

    public String getAction() {
        return mAction;
    }

    public Intent setAction(String action) {
        mAction = action != null ? action.intern() : null;
        return this;
    }

   public Intent addCategory(String category) {
        if (mCategories == null) {
            mCategories = new ArraySet<String>();
        }
        mCategories.add(category.intern());
        return this;
    }

    public void removeCategory(String category) {
        if (mCategories != null) {
            mCategories.remove(category);
            if (mCategories.size() == 0) {
                mCategories = null;
            }
        }
    }
    ......
}
```

- **data**属性

  data属性用于提供的一个Uri对象。

  type属性用于指定data属性所指定的Uri对应的MIME类型，也可以自己定义，格式是abc/xyz。

  设置type属性和data属性：

```
intent.setType("abc/xyz");
intent.setData(Uri.parse("zhang://www.group.com:8080/test"));
```

  上面设置只有最后的一个起作用，下面设置可以同时起作用：

```
intent.setDataAndType(Uri.parse("zhang://www.group.com:8080/test"), "abc/xyz");
```

  <data>可以作为intent-filter子元素用于匹配：

```
<data android:mimeType=""
      android:scheme=""
      android:host=""
      android:port=""
      android:path=""
      android:pathPrefix=""    //匹配path的前缀
      android:pathPattern=""   //匹配path的模板
 />
```

```
public class Intent  implements Parcelable, Cloneable {
    private Uri mData;
    private String mType;
    public Intent setData(Uri data) {
        mData = data;
        mType = null;
        return this;
    }
    public Intent setType(String type) {
        mData = null;
        mType = type;
        return this;
    }

    public Intent setDataAndType(Uri data, String type) {
        mData = data;
        mType = type;
        return this;
    }
    ......
}
```

- **extra属性**

Intent的extra属性用于在多个Activity之间进行数据交换，是Bundle类的实例，可以保存若干key-value对。Bundle类的详细介绍见“Bundle”节。

Intent的方法

```
putExtra(Bundle data)                    // 向Intent设置数据包
Bundle getExtra()                        // 从Intent取出数据包
putExtra(String key, Xxx value)          // 向extra中存入key-value对
getXxxExtra(String key)                  // 从extra中取出key的value(Xxx类型)
putExtra(String key, Serializable data)  //存入一个可序列化对象
getSerializableExtra(String key, Serializable data)  //取出可序列化对象
putSssArrayListExtra(String name, ArrayList<Integer> value)
ArrayList<Sss> getIntegerArrayListExtra(String name)
```

* **Xxx**可以是一种简单类型：Short、Int、Long、Float、Double、Char、String、CharSequence；也可以是数组TttArray（Ttt为简单类型），例如，getIntArrayExtra(key)返回Int[] 。
* **Sss**可以是Integer、String、CharSequence。
* 串行化的data是一个Java对象。

```
public class Intent  implements Parcelable, Cloneable {
    private Bundle mExtras;
    public Intent putExtra(String name, int value) {
        if (mExtras == null) {        mExtras = new Bundle();      }
        mExtras.putInt(name, value);     return this;
    }
    public Bundle getExtras() {
       return (mExtras != null) ? new Bundle(mExtras)  : null;
    }
    public Intent putExtras(Bundle extras) {
       if (mExtras == null) { mExtras = new Bundle(); }
       mExtras.putAll(extras);   return this; // putAll把extras的所有key-value克隆到mExtras中
    }
    public Intent putExtra(String name, int value) {
      if (mExtras == null) {    mExtras = new Bundle();    }
      mExtras.putInt(name, value);
      return this;
    }
    public int getIntExtra(String name, int defaultValue) {
         return mExtras == null ? defaultValue : mExtras.getInt(name, defaultValue);
    }

    public Intent putExtra(String name, String value) {
        if (mExtras == null) {   mExtras = new Bundle();   }
        mExtras.putString(name, value);     return this;
    }

    public String getStringExtra(String name) {
        return mExtras == null ? null : mExtras.getString(name);
    }
```

```
    public Intent putExtra(String name, Serializable value) {
        if (mExtras == null) {
            mExtras = new Bundle();
        }
        mExtras.putSerializable(name, value);        return this;
    }
    public Serializable getSerializableExtra(String name) {
        return mExtras == null ? null : mExtras.getSerializable(name);
    }
    public Intent putStringArrayListExtra(String name, ArrayList<String> value) {
        if (mExtras == null) {
            mExtras = new Bundle();
        }
        mExtras.putStringArrayList(name, value);          return this;
    }
    public ArrayList<String> getStringArrayListExtra(String name) {
        return mExtras == null ? null : mExtras.getStringArrayList(name);
    }
    public void removeExtra(String name) {
        if (mExtras != null) {
            mExtras.remove(name);
            if (mExtras.size() == 0) {
                mExtras = null;                                  * putExtra()使用了重载(Overload)
            }
        }
    }
    ......
}
```

- Bundle类的作用是通过key-value保存一批数据，用于在Activity或Service之间传递参数。它的功能主要通过继承BaseBundle获得对mMap的操作。

```
public final class Bundle extends BaseBundle implements Cloneable, Parcelable {
    ...
}
```

- mMap是ArrayMap<String, Object>类型,其中的Object可以是以下类别：
（1）基本类型：Char、Byte、Short、Int、Long、Float、Double、CharSequence、String。
（2）数组：int[]、String[]等。
（3）ArrayList：ArrayList<Integer>、ArrayList<String>\ArrayList<CharSequence>
（4）串行化对象：Serialize

- mMap的方法有：putXxx()和getXxx()。Parcelable接口有两个主要方法：writeToParcel(Parcel out, int flags)和readFromParcel(Parcel in)。主要用于把mAction、mData等数据打包在一起。Cloneable接口目前还没有方法。
- Bundle类的putAll(Bundle bundle)方法利用mMap.putAll()方法实现，用于拷贝bundle中的所有的key-value。

```
public class BaseBundle {
    ArrayMap<String, Object> mMap = null;
    public int size() {
        unparcel();    // 同步方法 （每个方法都作为第一个语句，后面略）
        return mMap.size();
    }
    public boolean isEmpty() {   return mMap.isEmpty();   }
    public void clear() {   mMap.clear();   }
    public boolean containsKey(String key) {   return mMap.containsKey(key);  }
    public Object get(String key) { return mMap.get(key);  }
    public void remove(String key) {    mMap.remove(key);   }
    public Set<String> keySet() {  return mMap.keySet();  }

    public void putInt(String key, int value) { mMap.put(key, value);   }
    public void putString(String key, String value){mMap.put(key, value);}
    public void putStringArray(String key, String[] value) {  mMap.put(key, value);   }
    void putStringArrayList(String key, ArrayList<String> value){ mMap.put(key, value);  }
    void putSerializable(String key, Serializable value) {  mMap.put(key, value);   }

    public int getInt(String key) { return getInt(key, 0);   } //  0为默认值
    public String getString(String key) { return mMap.get(key);}
    public String[] getStringArray(@Nullable String key) {  return (String[]) mMap.get(key);}
    ArrayList<String> getStringArrayList(String key){return (ArrayList<String>) Map.get(key);}
    Serializable getSerializable(@Nullable String key) {  return mMap.get(key);   }
    ...
}
```

* 这里做了简化：很多参数和返回值都带@Nullable提示，表示可为Null值；删除了一些错误处理(try).

- **flag属性**

Intent可调用addFlags()方法来为Intent添加控制标记，设置启动模式。

```
Intent intent = new Intent(MainActivity.this, MainActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
```

部分可设置的启动属性如下：

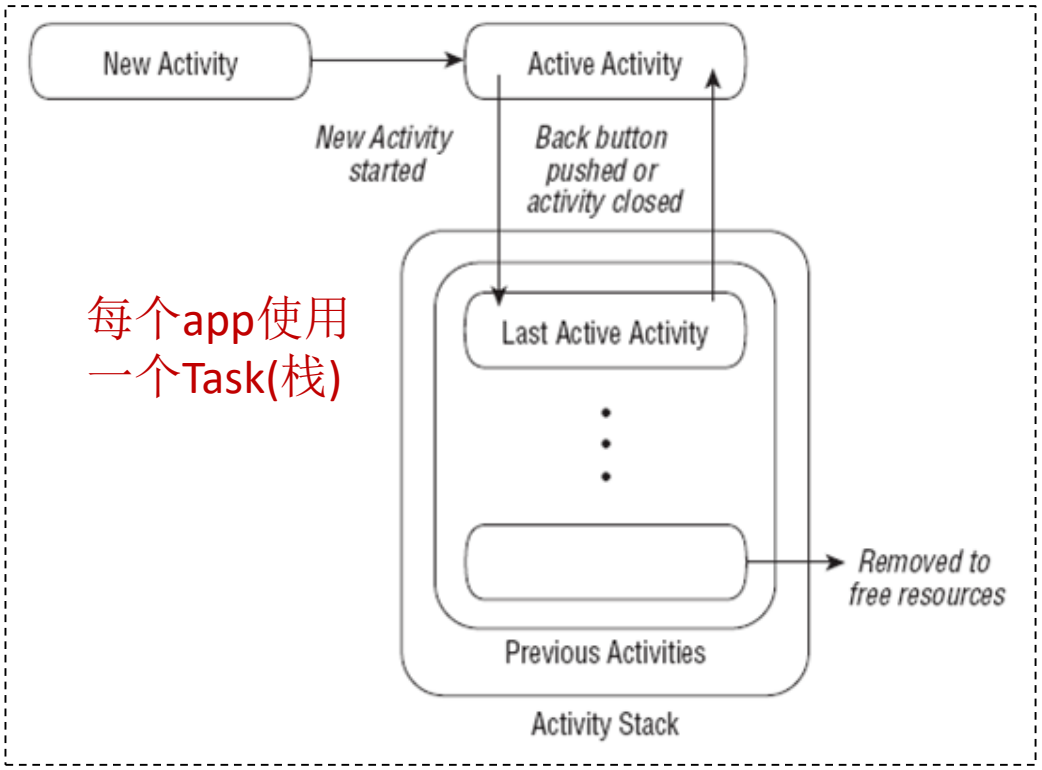
- FLAG_ACTIVITY_NEW_TASK
- FLAG_ACTIVITY_SINGLE_TOP
- FLAG_ACTIVITY_CLEAR_TOP
- FLAG_ACTIVITY_REORDER_TO_FRONT
- FLAG_ACTIVITY_BROUGHT_TO_FRONT
- FLAG_ACTIVITY_NO_HISTORY

参考“Task和Activity的加载方式”

```
public class Intent  implements Parcelable, Cloneable {
    private int mFlags;

    public Intent setFlags(int flags) {
        mFlags = flags;
        return this;
    }

    public Intent addFlags(int flags) {
        mFlags |= flags;
        return this;
    }

    public int getFlags() {
        return mFlags;
    }
    ......
}
```

Activity的加载方式

- 启动一个应用，系统就会为之创建一个Task。 Task是一个具有栈结构的容器，可以放置多个Activity实例，当前Activity总是位于Task栈的栈顶。
- 当一个Activity用默认模式启动另一个Activity时，前者会被压入当前Task栈而后者产生一个新实例压入当前Task栈的栈顶，当用户按下**后退键**或退出当前Activity，后者从Task栈弹出，前者又显示在幕前。如果启动的是其他程序中的Activity时也是一样的，好像它们属于同一个应用。
- Task和Task之间是互相独立的。当我们运行一个app时，按下**Home键**回到主屏，启动另一个应用，之前的Task会被转移到后台，新的Task被转移到前台，其栈顶的Activity也会显示到幕前。
- 如果此时按下**Home键**回到主屏，再选择之前的应用，之前的Task会被转移到前台，如果这时用户再做**后退**等动作，就是针对该Task内部进行操作了。



- **Activity的四种加载模式**

（1）standard模式（默认）

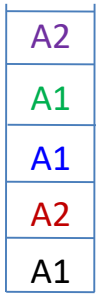
安卓会为目标Activity创建一个新实例，并把它加入当前Task栈中。

```
<activity android:name=".MainActivity"
          android:launchMode="standard"
          android:taskAffinity="abc">
    <intent-filter>
        ...
    </intent-filter>
</activity>
```

（2）singleTop模式

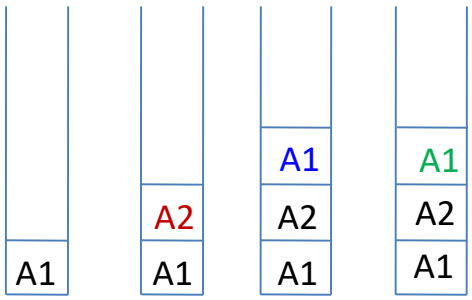
与standard模式相同，只是如果目标Activity已经位于Task的栈顶，则直接使用它，而不会创建一个新实例。

A1->A2->A1->A1->A2



A1 A2均使用standard模式启动(默认)

A1 -> A2 -> A1 -> A1



A1 A2均使用singleTop模式启动

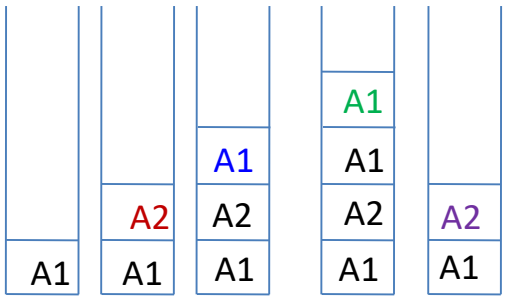
（3）singleTask模式

一个Acitivity在同一个Task最多只有一个实例。当目标Activity被启动时，如果task中不存在该Activity的实例，则创建它的一个实例；如果已经在Task中，则弹出它之上的所有Activity，直到它处于栈顶；如果已经在栈顶，则与singleTop相同，直接使用它。

（4）singleInstance模式

如果将要启动的Activity不存在，系统将会先创建一个全新的Task，再创建目标Activity实例并将其放入此Task中且单独使用。如果将要启动的Activity已存在，那么无论它位于哪个应用程序系统都会把该Activity所在的Task转到前台，从而使该Activity显示出来。整个系统只会包含该Activity的一个实例。

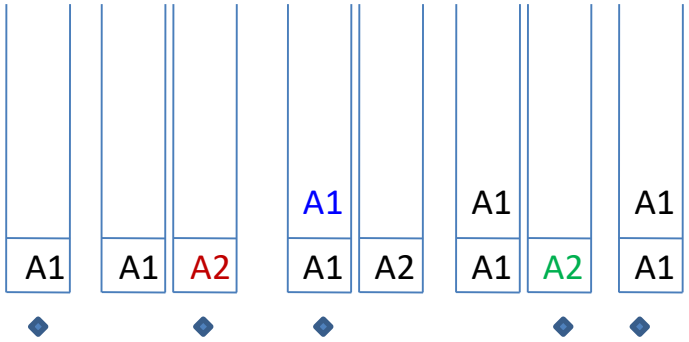
A1 -> A2 -> A1 -> A1 ->A2



如果在栈内则弹至栈顶使用，否则产生新实例

A1使用标准模式启动
A2使用singleTask模式启动

A1 -> A2 -> A1 -> A2 ->按返回键



整个系统只有一个实例，且用一个Task单独保存。

(◆表示前台)

A1使用标准模式启动
A2使用singleInstance启动

- **采用flag属性设置启动模式**

```
Intent intent = new Intent(MainActivity.this, MainActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
startActivity(intent);
```

- **部分flag属性取值：**
  - FLAG_ACTIVITY_NEW_TASK：如果要启动的Activity的affinity和当前任务的affinity相同，则会把它放入到现有任务当中，如果不同则会去创建一个新的Task。
  - FLAG_ACTIVITY_SINGLE_TOP：相当于Activity加载模式中的singleTop。
  - FLAG_ACTIVITY_CLEAR_TOP：相当于加载模式中的singleTask。
  - FLAG_ACTIVITY_REORDER_TO_FRONT：如果activity在Task中存在，拿到最顶端，不会启动新的Activity。如果Task栈包含A、B、C、D四个Activity（都是用standard启动），D用本模式启动B，得到A、C、D、B。
  - FLAG_ACTIVITY_BROUGHT_TO_FRONT：如果activity在Task栈存在并且是以这个模式的，则会被拿到最顶端,不会启动新的Activity，如果以标准模式启动，则会启动一个新的Activity。如果Task栈包含的A、B、C、D四个Activity都是用standard启动的，D用本模式启动B，得到A、B、C、D、B。如果A以本模式启动B，其他都是standard启动，D再以标准模式启动B时得到A、C、D、B。
  - FLAG_ACTIVITY_NO_HISTORY：用本模式启动新Activity时一旦离开当前Activity，它就不会存在于栈中。如果Task栈包含A、B、C、D四个Activity（都是用standard启动），D用本模式启动E，得到A、B、C、E。

- **项目名：NewActivityMode**

本项目有三个Activity分别采用standard模式、singleTop和singleTask，每个Activity都可以启动它本身的实例和另外两种Activity的实例。每次启动一个Acitvity实例，该Acitvity中的静态变量count都会加1。

```
// standard（默认） 每次启动新增一个实例
public class MainActivity extends AppCompatActivity {
    static int count = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        count++;
        TextView tv1=(TextView)findViewById(R.id.tv1);
        tv1.setText("计数："+count);
    }
    public void clickHandler1(View vw){
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }
    public void clickHandler2(View vw){
        Intent intent = new Intent(this, Main2Activity.class);
        startActivity(intent);
    }
    public void clickHandler3(View vw){
        Intent intent = new Intent(this, Main3Activity.class);
        startActivity(intent);
    }
}
```

```java
// lauchMode="singleTop" 每次启动看Task的顶部是否是该Activity的实例，
//          是则不新增，直接用他，如果不是则新增一个
public class Main2Activity extends AppCompatActivity {
    static int count = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        count++;
        TextView tv1=(TextView)findViewById(R.id.tv1);
        tv1.setText("计数："+count);
    }
    public void clickHandler1(View vw){
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }
    public void clickHandler2(View vw){
        Intent intent = new Intent(this, Main2Activity.class);
        startActivity(intent);
    }
    public void clickHandler3(View vw){
        Intent intent = new Intent(this, Main3Activity.class);
        startActivity(intent);
    }
}
```

```java
// launchMode="singleTask" 每次启动在Task中找实例，
//          找到弹至顶部，没找到新增一个
public class Main3Activity extends AppCompatActivity {
    static int count = 0;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);
        count++;
        TextView tv1=(TextView)findViewById(R.id.tv1);
        tv1.setText("计数："+count);
    }
    public void clickHandler1(View vw){
        Intent intent = new Intent(this, MainActivity.class);
        startActivity(intent);
    }
    public void clickHandler2(View vw){
        Intent intent = new Intent(this, Main2Activity.class);
        startActivity(intent);
    }
    public void clickHandler3(View vw){
        Intent intent = new Intent(this, Main3Activity.class);
        startActivity(intent);
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center">
    <TextView
        ...
        android:text="Activity1"
        android:id="@+id/textView" />
    <TextView
        ...
        android:text="计数：0"
        android:id="@+id/tv1"/>
    <Button
        ...
        android:text="启动Activity1(standard)*"
        android:id="@+id/btn1"
        android:onClick="clickHandler1"/>
    <Button
        ...
        android:text="启动Activity2（singleTop）"
        android:id="@+id/button"
        android:onClick="clickHandler2" />
    <Button
        ...
        android:text="启动Activity3（SingleTask）"
        android:id="@+id/button2"
        android:onClick="clickHandler3" />
</LinearLayout>
```

（1）控件省略了

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:textSize="20sp"
```

（2）activity_main2.xml和activity_main3.xml和本xml基本相同，不同点有两个，属性text的值的*分别打在第二个和第三个Button，第一个TextView的属性text的值分别为Activity2和Activity3。

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.isszym.newactivitymode">
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".MainActivity"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".Main2Activity"
                  android:label="singleTop"
                  android:launchMode="singleTop"/>
        <activity android:name=".Main3Activity"
                  android:label="singleTask"
                  android:launchMode="singleTask"/>
    </application>
</manifest>
```

启动

A1=> A1=> A3=> A3 => A1=> A1=>

不变

A2=> A1=> A2=> A2=> Back=>

不变

A3=> Back=> Back=> Back=>

栈顶只有一个A2

上一个A3弹至栈顶

- A1=> 点击“启动Activity1按钮”进入下个画面
- Back 按回退键

Activity的生命周期

活动状态：前台，可见，具有焦点可以响应用户操作

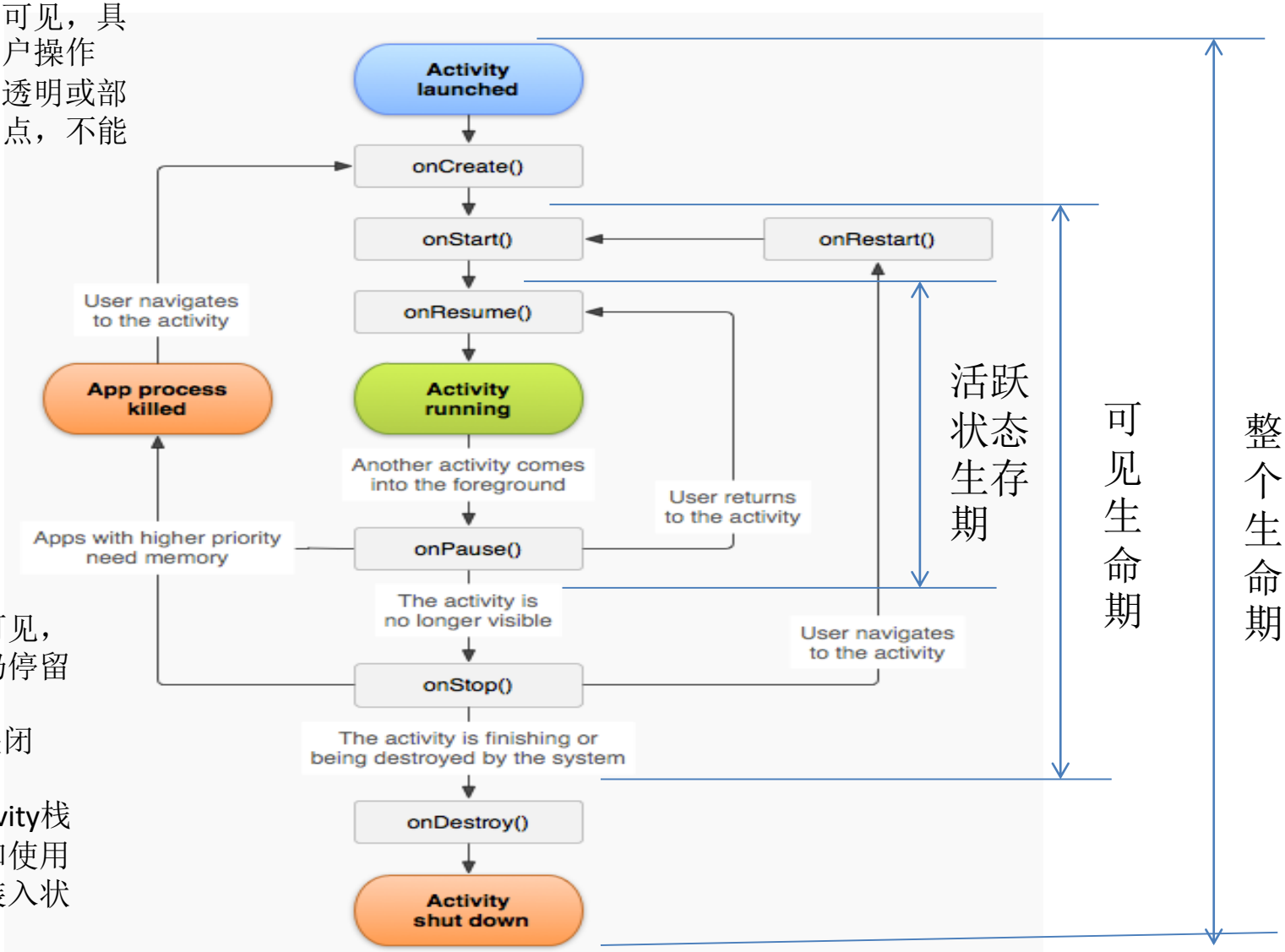
暂停状态：可见，透明或部分被遮蔽，没有焦点，不能响应用户操作

为了释放资源

停止状态：完全不可见，保存了状态信息，仍停留在内存中

退出或关闭

非活动状态：从Activity栈中移出了，在显示和使用前需要重新启动（装入状态）。



例子： 状态回调函数(1)　　项目名：ActivityStates

```
public class MainActivity extends AppCompatActivity {
    final String TAG = "测试";
    // 在整个生命期的开始时调用
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Log.d(TAG,"onCreate");
        // 初始化Activity和扩张(inflate) UI.
        setContentView(R.layout.activity_main);
    }
    // 在onCreate完成之后调用，用于恢复UI状态
    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        // 恢复savedInstanceState保存的UI状态.
        // 整个bundle也会传送给onCreate.
        // 只有它在最后可见后被系统杀死时才被调用。横竖变化时也会被调用
        Log.d(TAG,"onRestoreInstanceState");
    }
    // 在Activity进程的下一个可见生命期之前被调用
    @Override
    public void onRestart(){
        super.onRestart();
        // 知道在这个进程内本Activity已经可见时装载需要的变化.
        Log.d(TAG,"onRestart");
    }
```

```
// 在可见生命期的起始时调用（第一次可见）.
@Override
public void onStart(){
    super.onStart();
    // 进行Activity可见时所需的UI改变.
    Log.d(TAG,"onStart");
}
// 在活跃生存期的开始处调用
@Override
public void onResume(){
    super.onResume();
    // 重启Activity所需的任何暂停的UI更新，线程，进程
    // 它们在Activity不活动时被挂起了
    Log.d(TAG,"onResume");
}
// 在活跃生命期结束时用于保存UI状态改变
@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    // 把UI状态改变保存到参数savedInstanceState中.
    // 这个bundle在进程被杀死后再重启时会被传递给
    // onCreate and onRestoreInstanceState
    Log.d(TAG,"onSaveInstanceState");
    super.onSaveInstanceState(savedInstanceState);
}
```

```
    // 在活跃生生命期结束时被调用.
    @Override
    public void onPause(){
        // 挂起UI更新、线程、CPU密集型线程，它们在Activity不在前台活跃时
        // 不需要被更新
        Log.d(TAG,"onPause");
        super.onPause();
    }
    // 在可见生命期结束时被调用
    @Override
    public void onStop(){
        // 挂起剩余的UI更新、线程或进程.它们在Activity不可见时不被需要
        // 当在这次调用之后本进程被杀死时保持所有的编辑或状态改变
        Log.d(TAG,"onStop");
        super.onStop();
    }
    // 在整个生命期结束时的某个时候被调用
    @Override
    public void onDestroy(){
        // 清除任何占用的资源，例如，结束线程，关闭数据库连接等
        Log.d(TAG,"onDestroy");
        super.onDestroy();
    }
}
```

第一次运行	D/测试: onCreate D/测试: onStart D/测试: onResume
按Home键	D/测试: onPause D/测试: onSaveInstanceState D/测试: onStop
再次运行	D/测试: onRestart D/测试: onStart D/测试: onResume
按Back键	D/测试: onStop D/测试: onDestroy
再次运行	D/测试: onCreate D/测试: onStart D/测试: onResume
竖到横变换	D/测试: onPause D/测试: onSaveInstanceState D/测试: onStop D/测试: onDestroy
横到竖变换	D/测试: onCreate D/测试: onStart D/测试: onRestoreInstanceState D/测试: onResume

例子：状态回调函数(2)　　项目名：ActivityRestore

```
public class MainActivity extends AppCompatActivity {
        private static final String TAG = "测试";
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            // If an instance of this activity had previously stopped, we can
            // get the original text it started with.
            if(null != savedInstanceState)
            {
                int IntTest = savedInstanceState.getInt("IntTest");
                String StrTest = savedInstanceState.getString("StrTest");
                Log.d(TAG, "onCreate get the savedInstanceState+IntTest ="
                           +IntTest+"+StrTest="+StrTest);
            }
            setContentView(R.layout.activity_main);
            Log.d(TAG, "onCreate");
        }
```

```
    @Override
    public void onSaveInstanceState(Bundle savedInstanceState)  {
        // Save away the original text, so we still have it if the activity
        // needs to be killed while paused.
        savedInstanceState.putInt("intTest", 100);
        savedInstanceState.putString("strTest", "Hello, world! ");
        super.onSaveInstanceState(savedInstanceState);
        Log.d(TAG, "onSaveInstanceState");
    }
    @Override
    public void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);
        int intTest = savedInstanceState.getInt("intTest");
        String strTest = savedInstanceState.getString("strTest");
        Log.d(TAG, "onRestoreInstanceState: intTest ="+intTest+", strTest="+strTest);
    }
}
```

第一次运行　　D/测试: onCreate

按Home键后再运行　　D/测试: onSaveInstanceState

按Home键后再运行　　D/测试: onSaveInstanceState

按Back键后再运行　　D/测试: onCreate

横竖变化后

D/测试: onSaveInstanceState

D/测试: onCreate get the savedInstanceState+IntTest =0+StrTest=null
D/测试: onCreate
D/测试: onRestoreInstanceState：intTest =100，strTest=Hello，world！

Android 应用程序的状态

- 每个Android应用程序都是通过它们自己的进程运行的，每一个进程都运行在独立的Dalvik实例中。每个应用程序的内存管理和进程管理都是由运行时专门处理的。
- 与其它应用程序不同，Android应用程序不能控制它们自己的生命周期，系统有权根据优先级终止它们并释放它们占用的资源，它们只能默默地监听自己状态的改变并作出反应。

活动的Activity　OnReceive的Broadcast Receiver　正在执行的Service

1、活动进程	正在与用户交互的应用程序进程	关键优先级
2、可见进程	部分被遮蔽不能响应用户操作的应用程序进程	高优先级
3、启动服务进程	启动了Service的进程	
4、后台进程	不可见也没运行任何Service的Activity进程	低优先级
5、空进程	已结束但是仍保留在内存以防再次运行的应用程序进程	

```
import android.app.Application;
import android.content.res.Configuration;
public class MyApplication extends Application {
  private static MyApplication singleton;

  public static MyApplication getInstance() {
    return singleton; // 返回应用程序实例
  }
  @Override
  public final void onCreate() { // 创建应用程序时调用
    super.onCreate();
    singleton = this; // 这是单件模式，还可以初始化状态变量和共享资源
  }
  @Override
  public final void onLowMemory() {//只在后台进程终止或前台程序缺乏资源时调用
    super.onLowMemory();      // 应用程序通过重写它来清空缓存和释放不必要的资源
  }
  @Override
  public final void onTrimMemory(int level) {// 在当前应用程序决定减少内存
    super.onTrimMemory(level); // 开销时使用，通常在它进入后台时调用。level
  }
  @Override
  public final void onConfigurationChanged(Configuration newConfig) {
    super.onConfigurationChanged(newConfig); // 当配置改变时
  }
}
```

拨打电话

通过Action取值android.intent.action.DIAL可以访问系统提供的拨号app，Intent的data还可以给出要拨打的电话。

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    // Intent.ACTION_DIAL="android.intent.action.DIAL"
    public void clickHandler1(View source) {
        Intent intent = new Intent(Intent.ACTION_DIAL);
        startActivity(intent);
    }
    public void clickHandler2(View source) {
        Intent intent = new Intent(Intent.ACTION_DIAL); // "android.intent.action.DIAL"
        Uri data = Uri.parse("tel:" + "13600015411");
        intent.setData(data);
        startActivity(intent);
    }
}
```

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.isszym.dialphone.MainActivity">
    <Button
        android:text="拨号1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="34dp"
        android:layout_marginStart="34dp"
        android:layout_marginTop="69dp"
        android:onClick="clickHandler1"
        android:id="@+id/button" />
    <Button
        android:text="拨号2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@+id/button"
        android:layout_toRightOf="@+id/button"
        android:layout_toEndOf="@+id/button"
        android:layout_marginLeft="36dp"
        android:layout_marginStart="36dp"
        android:onClick="clickHandler2"
        android:id="@+id/button2" />
</RelativeLayout>
```

Fragment

- 概述

从安卓3.0开始增加了Fragment（碎片），其设计目的是当屏幕变大时可以放置更多的组件，使UI界面可以同时适用于手机和平板电脑 。

例如，要显示一篇分为很多节的长文，可以用两个Fragment放在一个Activity上显示文章页，左边的显示节标题列表，右边的显示当前节的内容，点击节标题会显示这一节的内容。还有一个Activity只显示所有节标题，点击后转到文章页。如果是手机，文章页隐藏标题列表的Fragment。

在Activity运行过程中，可以添加、移除或者替换Fragment。一个Activity上可以放置多个Fragment，一个Fragment可以在多个Activity上使用。

Fragment具有与Activity类似的使用方法，可以响应自己的输入事件，并且有自己的生命周期。它们的生命周期也会直接被其所属的宿主activity的生命周期所影响。

- 自定义Fragment类

通过继承Fragment类可以自定义Fragment类，利用回调函数onCreateView创建视图：（1）利用布局填充器得到一个布局；（2）修改布局的内容并返回该布局。

```
public class MyFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater,
                             ViewGroup container, Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fg_content,container,false);
        TextView txt_content = (TextView) view.findViewById(R.id.txt_content);
        Bundle bundle=this.getArguments();

        String content = bundle.getString("data");
        txt_content.setText(content);
        return view;
    }
}
```

- 把MyFragment实例加入Activity

把Fragment加入当前Activity的步骤：

（1）先从当前Activity得到片段管理器FragmentManager

（2）用该管理器启动片段处理事务；

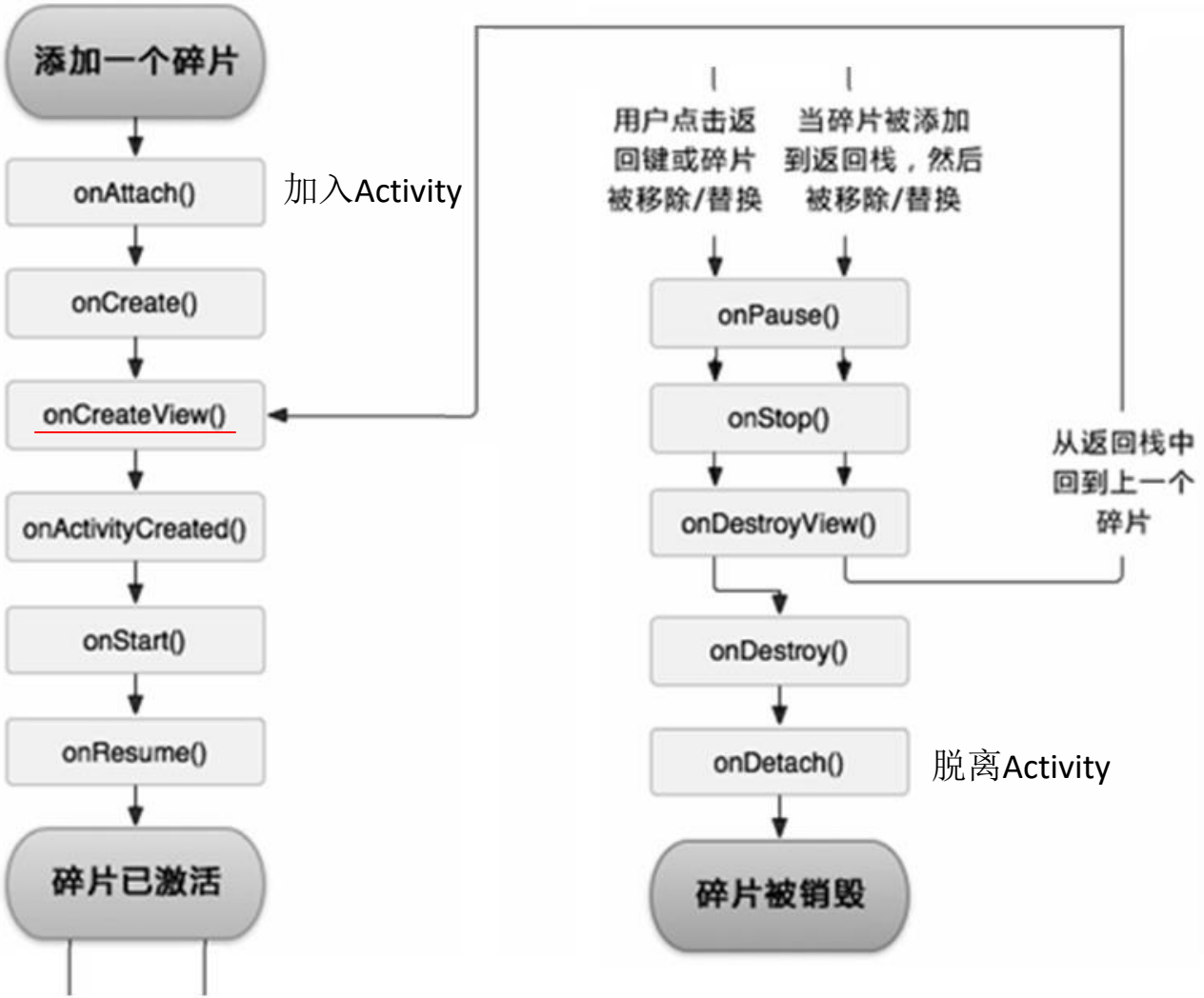
（3）创建新的MyFragment实例，设置要传递的参数；

（4）把该实例加入到Activity的一个布局中；

（5）把该实例提交给片段管理器。

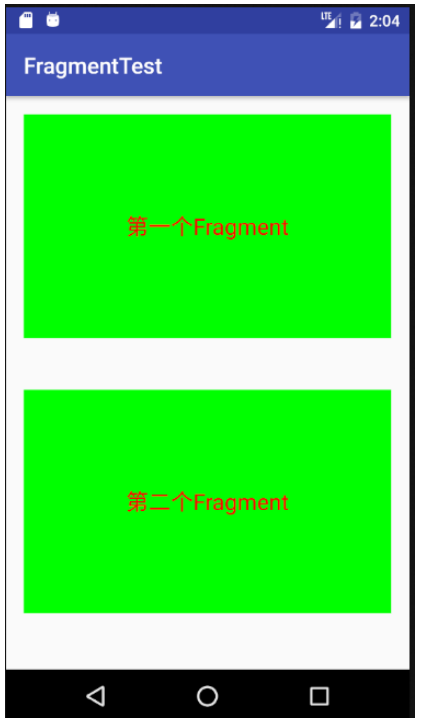
```
fManager = getFragmentManager();
FragmentTransaction fTransaction = fManager.beginTransaction();
MyFragment fg1 = new MyFragment();
Bundle bundle = new Bundle();
bundle.putString("data", "第一个Fragment");
fg1.setArguments(bundle);
fTransaction.add(R.id.content1,fg1);
fTransaction.commit();
```

- Fragment的生命周期

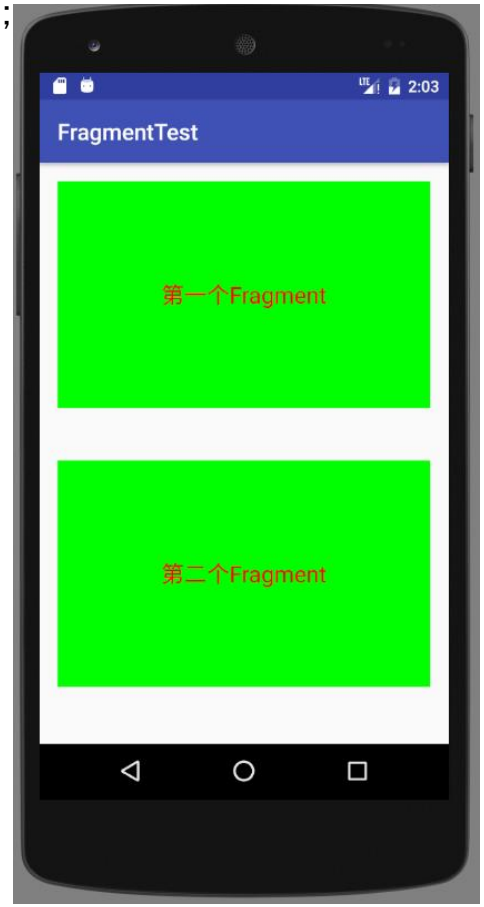


- 实例1、FragmentTest

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        FragmentManager fManager = getFragmentManager();
        FragmentTransaction fTransaction = fManager.beginTransaction();
        MyFragment fg1 = new MyFragment();
        Bundle bundle = new Bundle();
        bundle.putString("data", "第一个Fragment");
        fg1.setArguments(bundle);
        fTransaction.add(R.id.content1,fg1);

        MyFragment fg2 = new MyFragment();
        bundle = new Bundle();
        bundle.putString("data", "第二个Fragment");
        fg2.setArguments(bundle);
        fTransaction.add(R.id.content2,fg2);
        fTransaction.commit();
    }
}
```



```
public class MyFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                                    Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fg_content,container,false);
        TextView txt_content
            = (TextView) view.findViewById(R.id.txt_content);
        Bundle bundle=this.getArguments();
        String content = bundle.getString("data");
        txt_content.setText(content);
        return view;
    }
}
```



activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.isszym.fragmenttest.MainActivity">
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:id="@+id/content2"
        android:layout_marginBottom="34dp"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">
    </FrameLayout>
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:id="@+id/content1"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentStart="true">
    </FrameLayout>
</RelativeLayout>
```

fg_content.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="#0F0">

    <TextView
        android:id="@+id/txt_content"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:text="呵呵"
        android:textColor="#F00"
        android:textSize="20sp"/>
</LinearLayout>
```

AndroidManifest.xml

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <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>
```

PreferenceFragment

- 安卓采用PreferenceFragment保存app的选项设置，使它们在每次启动后生效。实现方法是以一个列表结构显示选项对象的层级关系，并把它们选项设置自动保存为SharedPreferences。
- 选项布局主要依赖选项的xml文件，其根节点为PreferenceScreen，下一层为分类节点PreferenceCategory，每个PreferenceCategory可以放置若干个选项设置节点，例如，多选节点ListPreference，单选节点CheckBoxPreference，文本节点EditTextPreference。

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <PreferenceCategory android:title="First Category">
        <ListPreference ...    />
   </PreferenceCategory>
   <PreferenceCategory android:title="Second Category">
        <EditTextPreference .../>
        <CheckBoxPreference ... />
    </PreferenceCategory>
</PreferenceScreen>
```

- 每个节点的主要属性：

android:key	唯一标识符，PreferenceManager可以用findPreference获取指定的preference。
android:title	选项界面显示的大标题
android:summary	摘要，标题下面的文字，字体较小。
android:entries	弹出的对话框的列表显示内容，用于多选。
android:entryValues	多选时的选项值，与android:entries相对应。
android:defaultValue	当对应值不存在时的默认值
android:dialogTitle	弹出的对话框中的标题信息

```
<ListPreference
    android:key="list_key"
    android:defaultValue="list key default value"
    android:title="list title"
    android:summary="list_summary"
    android:entries="@array/list_preference"
    android:entryValues="@array/list_preference"
    android:dialogTitle="list_dialog_title" />
```

•通过sharedPreferences保存在/data/data/<packagename>/shared_prefs/目录下）。简单的说就是此处是数据库中的值。上面的android：entries是展现给用户的列表的值

- 实现

（1）通过addPreferencesFromResource加入*选项XML文件*并显示选项界面。
（2）List和EditText选项需要弹出框进行修改，CheckBox选项直接在界面修改。
（3）由于修改List和EditText选项后不会自动改变Fragment中的显示，需要通过SharedPreferences获取选项值进行修改。

```
EditTextPreference mEtPreference = (EditTextPreference)findPreference("edittext_key");
SharedPreferences sharedPreferences = getPreferenceScreen().getSharedPreferences();
mEtPreference.setSummary(sharedPreferences.getString("edittext_key", "linc"));
```

（4）为了捕捉选项修改事件，需要在当前Fragment中注册onSharedPreferenceChanged事件。

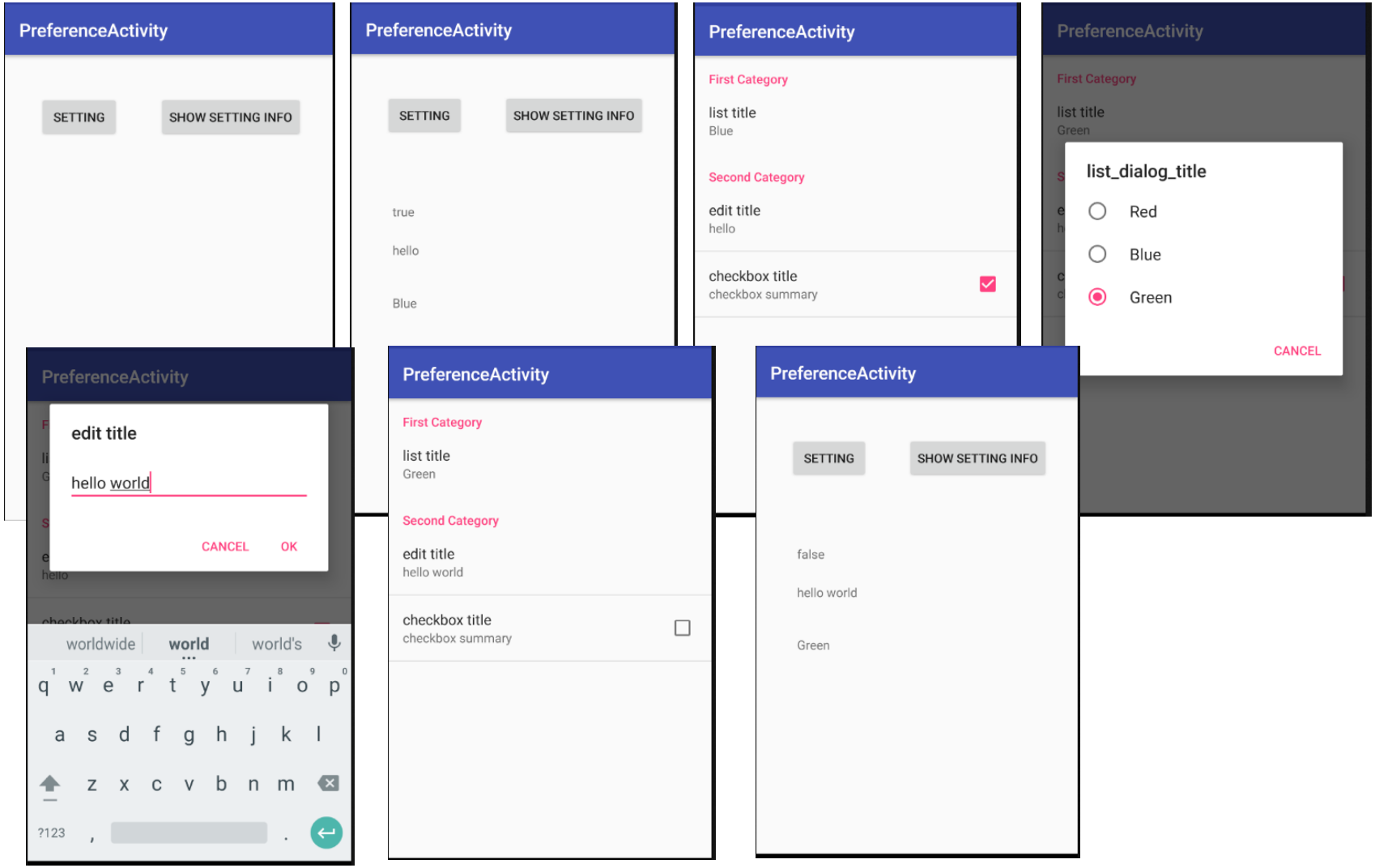
```
sharedPreferences.registerOnSharedPreferenceChangeListener(this);
```

（5）任何时候都可以通过上下文环境的PreferenceManager用getDefaultSharedPreferences方法获得选项设置值。

```
SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(this);
boolean chk = settings.getBoolean("checkbox_key", false);
```

- 实例1、项目PreferenceActivity

先显示设置信息(Show Setting Info)，然后进入设置界面(setting)逐个修改选项值后再显示选项设置值。



```
public class MainActivity extends AppCompatActivity {
    private Button btnSetting,btnShow;  private TextView tvCheckout,tvList,tvEditText;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);     setContentView(R.layout.activity_main);
        initView();
    }
    private void initView() {
        btnSetting = (Button)findViewById(R.id.btn_setting);
        btnShow = (Button)findViewById(R.id.btn_show);
        btnSetting.setOnClickListener(buttonListener);
        btnShow.setOnClickListener(buttonListener);
        tvCheckout = (TextView)findViewById(R.id.tv_checkout);
        tvList = (TextView)findViewById(R.id.tv_list);
        tvEditText = (TextView)findViewById(R.id.tv_edittext);
    }
    private OnClickListener buttonListener = new OnClickListener() {
        @Override
        public void onClick(View v) {
            switch(v.getId()) {
            case R.id.btn_setting:
                startActivity(new Intent(MainActivity.this,PreferenceActivity.class)); break;
            case R.id.btn_show:
                showSettingInfo();   break;
            }
        }
    };
    private void showSettingInfo() {
        SharedPreferences settings = PreferenceManager.getDefaultSharedPreferences(this);
        tvCheckout.setText(settings.getBoolean("checkbox_key", false)+"");
        tvEditText.setText(settings.getString("edittext_key", ""));
        tvList.setText(settings.getString("list_key", ""));
    }
}
```

preferences.xml

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android" >
    <PreferenceCategory android:title="First Category">
        <ListPreference
            android:key="list_key"
            android:defaultValue="list key default value"
            android:title="list title"
            android:summary="list_summary"
            android:entries="@array/list_preference"
            android:entryValues="@array/list_preference"
            android:dialogTitle="list_dialog_title" />
   </PreferenceCategory>
   <PreferenceCategory android:title="Second Category">
        <EditTextPreference
            android:key="edittext_key"
            android:defaultValue="edit default value"
            android:summary="edit summary"
            android:title="edit title" />
        <CheckBoxPreference
            android:key="checkbox_key"
            android:defaultValue="checkbox default value"
            android:summary="checkbox summary"
            android:title="checkbox title"
            />
    </PreferenceCategory>
</PreferenceScreen>
```

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" >
    <TextView
        android:id="@+id/tv_checkout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" />
    <Button
        android:id="@+id/btn_setting"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Setting" />
    <TextView
        android:id="@+id/tv_edittext"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" />
    <TextView
        android:id="@+id/tv_list"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" />
    <Button
        android:id="@+id/btn_show"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Setting Info"  />
</RelativeLayout>
```

*删除了定位信息

activity_preference.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:id="@+id/ly_content">
    </FrameLayout>
</RelativeLayout>
```

res/values/arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="list_preference">
        <item>Red</item>
        <item>Blue</item>
        <item>Green</item>
    </string-array>
</resources>
```

```
public class PreferenceActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        setContentView(R.layout.activity_preference);
        FragmentManager fManager = getFragmentManager();
        FragmentTransaction fTransaction = fManager.beginTransaction();
        SettingFragment fg = new SettingFragment();
        Bundle bundle = new Bundle();
        bundle.putString("data", "第一个Fragment");
        fg.setArguments(bundle);
        fTransaction.add(R.id.ly_content, fg);
        fTransaction.commit();
    }
}
```

```
public class SettingFragment extends PreferenceFragment
                                    implements OnSharedPreferenceChangeListener {
    private EditTextPreference mEtPreference;
    private ListPreference mListPreference;
    private CheckBoxPreference mCheckPreference;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
        initPreferences();
        Bundle bundle=this.getArguments();
        String content = bundle.getString("data");
        Log.d("测试",content);
    }
    private void initPreferences() {
       mEtPreference = (EditTextPreference)findPreference("edittext_key");
       mListPreference = (ListPreference)findPreference("list_key");
       mCheckPreference = (CheckBoxPreference)findPreference("checkbox_key");
    }
    @Override
    public void onResume() {
        super.onResume();

        // Setup the initial values
        SharedPreferences sharedPreferences = getPreferenceScreen().getSharedPreferences();
        mListPreference.setSummary(sharedPreferences.getString("list_key", ""));
        mEtPreference.setSummary(sharedPreferences.getString("edittext_key", "linc"));
        // Set up a listener whenever a key changes
        sharedPreferences.registerOnSharedPreferenceChangeListener(this);
    }
```

```
    @Override
    public void onPause() {
        super.onPause();
        // Unregister the listener whenever a key changes
        getPreferenceScreen().getSharedPreferences().unregisterOnSharedPreferenceChangeListener(thi
    }
    @Override
    public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String key) {
        if (key.equals("edittext_key")) {
            mEtPreference.setSummary(
                    sharedPreferences.getString(key, "20"));
        } else if(key.equals("list_key")) {
            mListPreference.setSummary(sharedPreferences.getString(key, ""));
        }
    }
}
```

附录1、安卓权限设置

参考

下面是安卓权限码：

权限日历	READ_CALENDAR
日历	WRITE_CALENDAR
相机	CAMERA
联系人	READ_CONTACTS
联系人	WRITE_CONTACTS
联系人	GET_ACCOUNTS
麦克风	RECORD_AUDIO
电话	READ_PHONE_STATE
电话	CALL_PHONE
电话	READ_CALL_LOG
电话	WRITE_CALL_LOG
电话	ADD_VOICEMAIL
电话	USE_SIP
电话	PROCESS_OUTGOING_CALLS
传感器	BODY_SENSORS
短信	SEND_SMS
短信	RECEIVE_SMS
短信	READ_SMS
短信	RECEIVE_WAP_PUSH
短信	RECEIVE_MMS
存储	READ_EXTERNAL_STORAGE
存储	WRITE_EXTERNAL_STORAGE
位置	ACCESS_FINE_LOCATION
位置	ACCESS_COARSE_LOCATION

下面以WRITE_EXTERNAL_STORAGE和READ_EXTERNAL_STORAGE为例说明使用方法

（1）在清单文件(AndroidManifest.xml)中的application元素之外加入：

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
```

（2）在Android 6.0后要加入如下代码：

```
// 判断是否已经赋予权限
if (ContextCompat.checkSelfPermission(this, Manifest.permission.WRITE_EXTERNAL_STORAGE)
        != PackageManager.PERMISSION_GRANTED) {
    // 如果应用之前请求过此权限但用户拒绝了请求，此方法将返回 true。
    if (ActivityCompat.shouldShowRequestPermissionRationale(this,
                        Manifest.permission.WRITE_EXTERNAL_STORAGE)) {
        // 这里可以写个对话框之类的项向用户解释为什么要申请权限，并在对话框的确认键后
        // 再次申请权限.它在用户选择"不再询问"的情况下返回false
    }else{
        // 申请权限，字符串数组内是一个或多个要申请的权限，
        //     1是申请权限结果的返回参数，在onRequestPermissionsResult可以得知申请结果
        ActivityCompat.requestPermissions(this,
                        new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE,
                                Manifest.permission.READ_EXTERNAL_STORAGE}, 1);
    }
}
```

（3）可以用下面的代码判定权限是否申请成功：

```
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    if (requestCode == 1) {
        for (int i = 0; i < permissions.length; i++) {
            if (grantResults[i] == PERMISSION_GRANTED) {
                Toast.makeText(this, "" + "权限" + permissions[i]
                        + "申请成功", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(this, "" + "权限" + permissions[i]
                        + "申请失败", Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```