



# Android应用开发

主讲教师 胡鑫

[hithuxin@hit.edu.cn](mailto:hithuxin@hit.edu.cn)

## 第6章 Intent

1. [Intent简介](#)
2. [使用Intent启动Activity](#)
3. [Activity间数据传递](#)

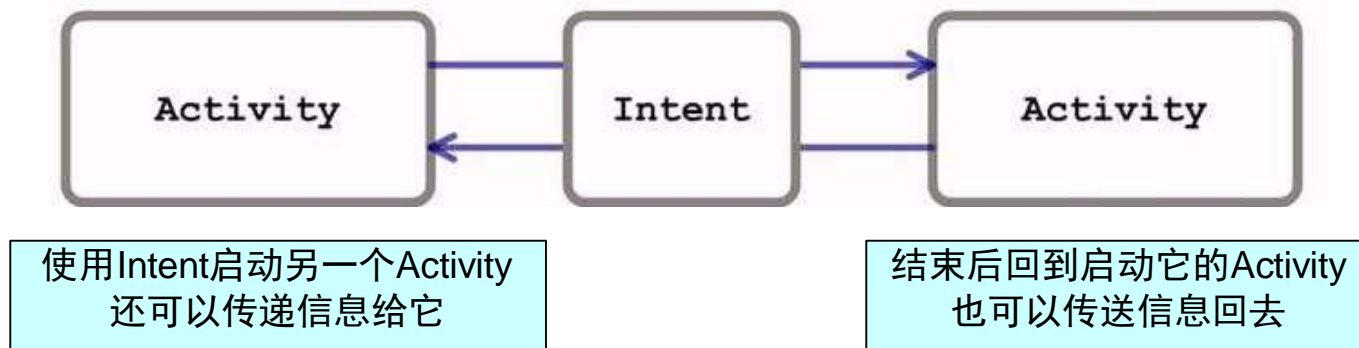
## 解决问题：

如何启动其他的Activity并实现它们之间的单/双向通信



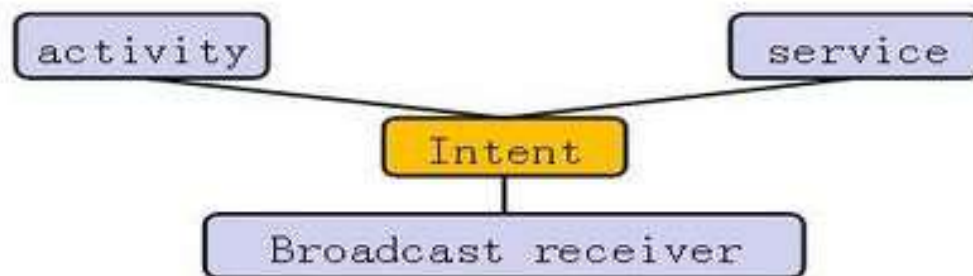
## 6.1 Intent简介

- Intent是一种**组件之间消息传递机制**，它是一个动作的完整描述：包含了动作产生组件、接收组件和传递的数据信息。



## 6.1 Intent简介

- Intent主要用途：启动Activity、Service，在Android系统上发布Broadcast消息。



## 6.2 使用Intent启动Activity

■ 显示启动

■ 隐式启动

## 1. 显示启动



A端

```
Intent intent = new Intent(A_Activity.this, B_Activity.class);  
startActivity(intent);    //A启动B
```

启动Activity

当前的Activity

要启动的Activity(一定要用class)

■ 显示启动：在Intent中指明要启动的Activity类。

## 2. 隐式启动

- 无需指明具体启动哪一个Activity，而由Android系统根据Intent的动作和数据来决定启动哪一个Activity。
- 例如：希望启动一个浏览器，却不知道具体应该启动哪一个Activity，此时则可以使用Intent的隐式启动，由Android系统决定启动哪一个Activity来接收这个Intent。
- 隐式启动的可以是Android系统内置的Activity，也可能是程序本身的Activity，还可是第三方应用程序的Activity。




## 隐式启动示例1

### ■ 启动浏览器打开一个网址：

方式1：

```
Intent intent = new Intent( Intent.ACTION_VIEW,  
                           Uri.parse("http://www.163.com") );  
startActivity(intent);
```



方式2：

```
Intent intent = new Intent( );  
intent.setAction(Intent.ACTION_VIEW);  
intent.setData(Uri.parse("http://www.163.com"));  
startActivity(intent);
```

## 隐式启动示例2

### ■ 打开播放器播放音乐：

```
Intent intent = new Intent( Intent.ACTION_VIEW );
```

```
String path = "/sdcard/mp3/flower.mp3";
```

SDCard文件路径

```
Uri uri = Uri.parse("file://" + path);
```

或者"/storage/emulated/0/"

```
intent.setDataAndType(uri, "audio/*");
```

将得到这样一个Uri:  
file:///sdcard/mp3/flower.mp3

```
startActivity(intent);
```

设置数据和类型

提醒：添加读取权限

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

## 播放音乐例子在7.0报错问题的解决

```
FATAL EXCEPTION: main
```

```
Process: com.example.test, PID: 1853
```

```
android.os.FileUriExposedException: file:///sdcard/Music/flower.mp3 exposed beyond app through Intent.getData()
```

- **报错：** FileUriExposedException: ...exposed beyond app through Intent.getData()
- **原因：** 按照Android N的要求，如果 file://格式的Uri的Intent离开应用将导致 FileUriExposedException 异常。若要在应用间共享文件，应发送content://格式的Uri，并授予Uri临时访问权限，实现此方法的是FileProvider类。

## 解决方法：不推荐

onCreate()中添加如下代码：即去掉安全策略

```
//判断是否是AndroidN以及更高的版本 N=24
if ( Build.VERSION.SDK_INT >= Build.VERSION_CODES.N ) {
    StrictMode.VmPolicy.Builder builder =
        new StrictMode.VmPolicy.Builder();
    StrictMode.setVmPolicy(builder.build());
}
```

其他代码不变

## 常见的Intent动作

动作	说明
Intent.ACTION_VIEW	为用户显示数据
<p>示例：</p> <p>打开浏览器： <code>Intent intent = new Intent( Intent.ACTION_VIEW, Uri.parse("http://www.163.com") );</code></p> <p>打开所有联系人： <code>Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("content://contacts/people"));</code></p> <p>打开id=1联系人： (如果打不开，则添加权限，见后)</p> <p><code>Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse("content://contacts/people/1"));</code></p>	
Intent.ACTION_DIAL	打开拨号界面（如果模拟器没有拨号程序则无法使用）
<p>示例：</p> <p>拨打10086： <code>Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:10086"));</code></p>	
Intent.ACTION_CALL	直接拨打电话(权限问题后页有详解)
<p>示例：</p> <p><code>Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:18908643860"));</code></p>	

## 常见的Intent动作

动作	说明
Intent.ACTION_SENDTO	调用发送短信
示例： Intent intent = new Intent(Intent.ACTION_SENDTO, Uri.parse("smsto:18908643860"));	
Intent.ACTION_EDIT	编辑
示例：编辑id=1的联系人 Intent intent = new Intent(Intent.ACTION_EDIT, Uri.parse("content://contacts/people/1"));	
Intent.ACTION_WEB_SEARCH	用google搜索内容
示例： Intent intent = new Intent(Intent.ACTION_WEB_SEARCH); intent.putExtra( SearchManager.QUERY, "武科大" ); //传送数据	

## 隐式启动示例3：程序拨打电话例子

(1) 在AndroidManifest.xml添加权限：

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

(2) 核心程序代码：

```
Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:18908643860"));  
startActivity(intent);
```

(3) 权限申请 (6.0以后)：代码见后

## Android 6.0(API23)以后的权限问题解决

以拨打电话为例  
6.0真机测试

```
Button btn1 = (Button) findViewById(R.id.btn_test);
btn1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:18908643860"));
        startActivity(intent);

    }
});
```



直接输入上述代码会提示要  
添加permission check

检查授权

```
if (ActivityCompat.checkSelfPermission(this, Manifest.permission.CALL_PHONE) !=
    PackageManager.PERMISSION_GRANTED) {

    //如果没有获得系统授权
    //用requestPermissions()方法来请求系统授权
    return;
}
```



检测是否经过系统授权(运行时会弹出授权消息框, 要选择允许权限)

```
if (ActivityCompat.checkSelfPermission( MainActivity.this, Manifest.permission.CALL_PHONE)
    != PackageManager.PERMISSION_GRANTED ) {

    ActivityCompat.requestPermissions(MainActivity.this,
        new String[] { Manifest.permission.CALL_PHONE }, 123);

    return;
}

Intent intent = new Intent(Intent.ACTION_CALL, Uri.parse("tel:18908643860"));
startActivity(intent);
```

自定义int类型的REQUEST\_CODE值

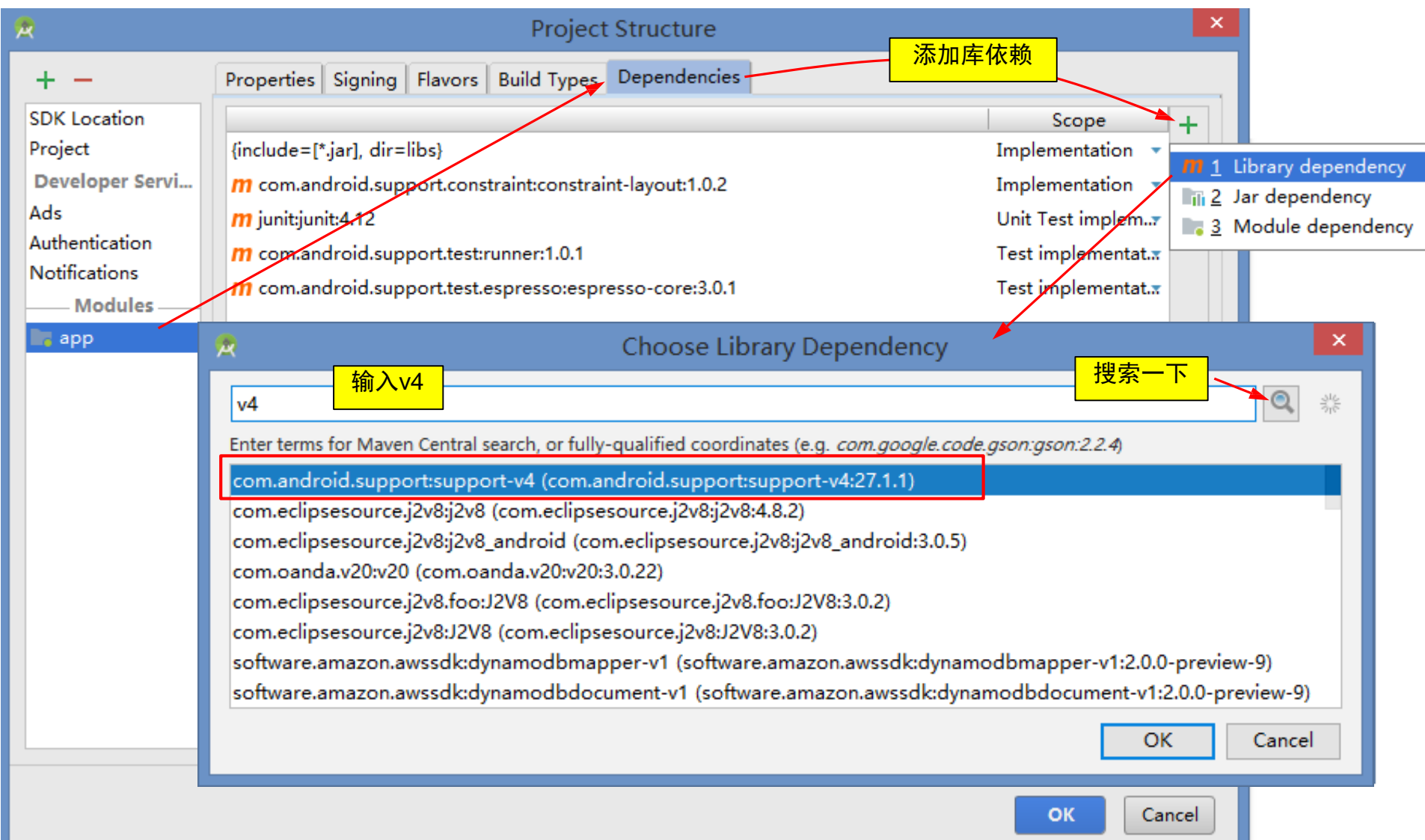
备注: 如果是其他权限, 用法同上, 换个权限值即可

备注:

```
import android.Manifest;
import android.support.v4.app.ActivityCompat
```

注: ActivityCompat类需要com.android.support:support-v4依赖包的支持 (添加依赖的方法见后)

添加v4依赖：File → Project Structure...



build.gradle(Module:app) 文件

```
apply plugin: 'com.android.application'
```

```
android {  
    compileSdkVersion 27  
    buildToolsVersion "27.0.3"  
    defaultConfig {  
        applicationId "com.example.he  
        minSdkVersion 21  
        targetSdkVersion 27  
        versionCode 1  
        versionName "1.0"  
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
    }  
    buildTypes {  
        release {  
            minifyEnabled false  
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
        }  
    }  
}
```

提醒:

- (1) 添加新依赖时要取消选中gradle的offline work选项!
- (2) v4的版本要与compileSdkVersion版本一致 (高会报错)

版本要一致

```
dependencies {  
    implementation fileTree(include: ['*.jar'], dir: 'libs')  
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.1'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'  
    implementation 'com.android.support:support-v4:27.1.1'  
}
```

新添加的v4依赖包

[【返回】](#)

## 6.3 Activity间数据传递

使用Intent启动另一个Activity组件，也可以传递信息给它



结束后回到启动它的Activity，也可以传送信息回去

## 6.3 Activity间数据传递

■ 单向传递数据

■ 双向传递数据

## 1. 单向传递数据



### ■ A 利用Bundle给 B 传递数据：★★★

A端

```
Intent intent = new Intent(A_Activity.this, B_Activity.class);  
Bundle bundle=new Bundle();  
bundle.putString("name", "wustzz"); //给Bundle添加key-value值对  
intent.putExtras(bundle); //为intent设置bundle  
startActivity(intent); //启动B
```

B端

```
Intent intent= getIntent(); //获取传递过来的intent  
Bundle bundle=intent.getExtras(); //取出intent中的bundle  
String name=bundle.getString("name"); //取出key对应的value
```

## 关于Bundle

key	value
key	value
...	...

- Bundle类是一个存储和管理key-value值对的类，多应用于Activity之间相互传递值。

- 用法示例：

(1) 新建一个bundle对象：`Bundle bundle=new Bundle();`

(2) 在bundle中添加一个key-value对：

```
bundle.putString("name", "wustzz");
```

key

value (这里是String数据)

(3) 从bundle中取出value：

```
bundle.getString("name");
```

Bundle类有很多的put方法，如：

putInt(), putDouble(), putIntArray()

对应的get方法，如：

getInt(), getDouble(), getIntArray()

## Bundle类的一些方法

- `clear()`: 清除此Bundle映射中的所有保存的数据。
- `clone()`: 克隆当前Bundle
- `containsKey(String key)`: 返回指定key的值
- **`getString(String key)`: 返回指定key的字符**
- `isEmpty()`: 如果这个捆绑映射为空, 则返回true
- **`putString(String key, String value)`: 插入一个给定key的字符串值**
- `remove(String key)`: 移除指定key的值



## Intent类的putExtras()、getExtras()方法

- putExtras(Bundle): 往Intent中添加一个Bundle对象
- getExtras(): 从Intent中取出Bundle对象

## 单向传递数据示例1 -- 传递普通数据

HelloAndroid

发送端: RegisterActivity

用户名 请输入用户名

手机号 11位数字

密 码 8位小写字母

性 别 ☐男 ☐女

爱 好 ☐篮球 ☐足球

专 业 计算机科学

照 片 

提醒注意: 所有元素都必填,以避免出现空对象导致错误闪退

```
Intent intent=new Intent(RegisterActivity.this,WelcomeActivity.class);  
Bundle bundle=new Bundle();  
bundle.putString("username",username.getText().toString());  
bundle.putString("sex",sex.getText().toString());  
bundle.putString("photo_uri",photo_uri.toString());  
intent.putExtras(bundle);  
startActivity(intent);
```

备注: 关于photo\_uri值

- (1) 先在RegisterActivity类中定义成员 Uri photo\_uri;
- (2) 然后在获取相册图像的onActivityResult代码中添加:  
photo\_uri=data.getData();


## 单向传递数据示例1 -- 传递普通数据

接收端: WelcomeActivity

HelloAndroid

用户名: wustzz

性别: 男



onCreate()代码

```
Intent intent=getIntent();
Bundle bundle=intent.getExtras();
String u=bundle.getString("username");
String sex=bundle.getString("sex");
String photo_uri=bundle.getString("photo_uri");
```

//显示一下

```
TextView tv_username= (TextView)findViewById(R.id.tv_username);
tv_username.setText("用户名: "+u);
TextView tv_sex= (TextView)findViewById(R.id.tv_sex);
tv_sex.setText("性别: "+sex);
ImageView iv_photo= (ImageView)findViewById(R.id.iv_photo);
iv_photo.setImageURI( Uri.parse(photo_uri) );
```

将Uri格式的字符串转为Uri对象

Uri: 用来标识资源的逻辑位置  
(远程或本地的可用资源)



## 单向传递数据示例2 -- 传递对象数据

- 针对前例，如何用一个user对象传递数据：

```
public class User {  
    private String username;  
  
    private String sex;  
    private String photo_uri;  
  
    //构造函数  
    public User(String username, String sex, String photo_uri) {  
        this.username = username;  
        this.sex = sex;  
        this.photo_uri = photo_uri;  
    }  
  
    //以下省略getter、setter方法...  
}
```

自定义的  
User类

## 关键点在于Bundle如何传递对象数据

- Bundle可以传递对象，但前提是这个对象需要序列化。
- Bundle的putSerializable()方法，可以存储已经序列化的对象数据(仍然是Key-Value形式); 
- 接收数据时Bundle用getSerializable()方法，获得数据需要强制转化一下原来的对象类型。 

=====什么是序列化=====

序列化是一种用来处理对象流的机制，以解决如网络传播、磁盘读写等对对象流读写操作时所引发的问题。

## 具体过程

(1) 先将User类序列化(直接实现**Serializable**接口即可)

```
public class User implements  
Serializable
```

```
{ private String username;  
  private String sex;  
  private String photo_uri;
```

```
  //构造函数
```


```
  public User(String username, String sex, String photo_uri) {  
    this.username = username;  
    this.sex = sex;  
    this.photo_uri = photo_uri;  
  }
```

```
  //以下省略getter、setter方法...
```

```
}
```

标识一个类的  
对象可以  
被序列化

## (2) 发送端：RegisterActivity主要代码



```
Intent intent=new Intent(RegisterActivity.this,WelcomeActivity.class);
Bundle bundle=new Bundle();
User user=new User(username.getText().toString(),
                    sex.getText().toString(),
                    photo_uri.toString() );
bundle.putSerializable("user",user); //存放序列化对象
intent.putExtras(bundle);
startActivity(intent);
```

提醒注意：所有元素都必填,避免出现空对象导致错误闪退

## (3) 接收端：WelcomeActivity主要代码

```
Intent intent=this getIntent();
```

```
Bundle bundle=intent.getExtras();
```



```
User user= (User)bundle.getSerializable("user"); //获取序列化对象(需强转)
```

```
后面再用getter方法获取属性值去显示：如u.getUsername()
```

```
.....
```

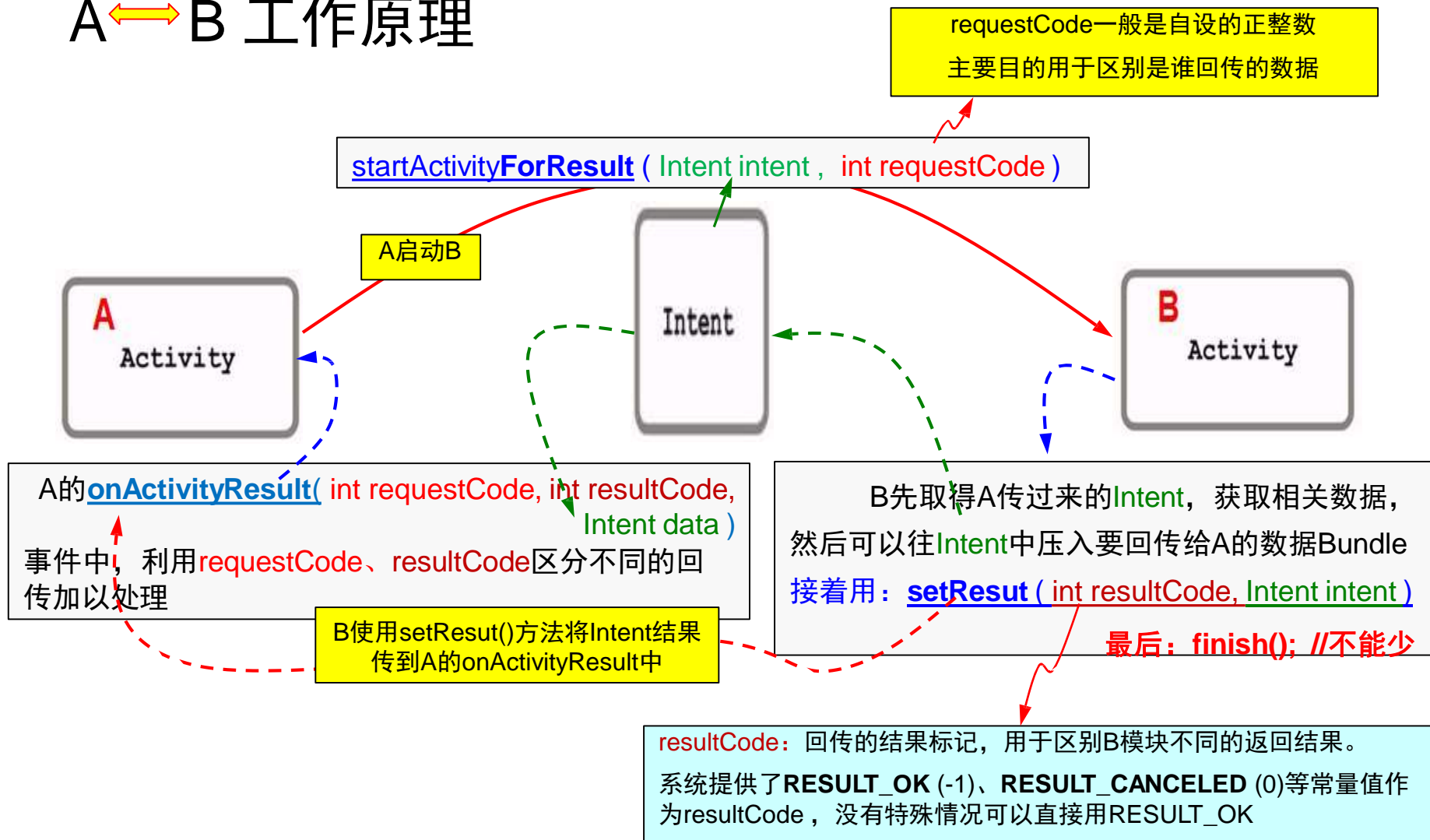


## 2. 双向传递数据

重点看 B 如何返回数据给 A



## A ↔ B 工作原理



## B 返回数据给 A 的核心代码框架 ★★★

**A端**

```
Intent intent = new Intent(A_Activity.this, B_Activity.class);
```

```
//如有需要intent可以给B传递Bundle(略)
```

```
startActivityForResult(intent, 100);
```

自设requestCode=100

**B端**

```
Intent intent=getIntent(); //取得A传过来的Intent
```

```
//如有需要处理从A传过来的Bundle(略)
```

```
Bundle bundle2=new Bundle(); //如需要则新建要传回的Bundle
```

```
bundle2.putString("key值", "value值");
```

```
intent.putExtras(bundle2);
```

```
setResult(RESULT_OK, intent);
```

resultCode回传一个RESULT\_OK(-1)标记

```
finish(); //必须的，用于关闭B端，返回A端
```

## B 返回数据给 A 的基本代码框架(续)

A端

@Override

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    super.onActivityResult(requestCode, resultCode, data);  
    if(requestCode==100){  
        if (resultCode==RESULT_OK) {  
            //处理B回传的数据  
            Bundle b=data.getExtras();  
            String str=b.getString("key值");  
            ...  
        }  
        if (resultCode==其他标记值) {  
            ...  
        }  
    }  
}
```

B中回传的Intent

如有必要可以  
是使用switch

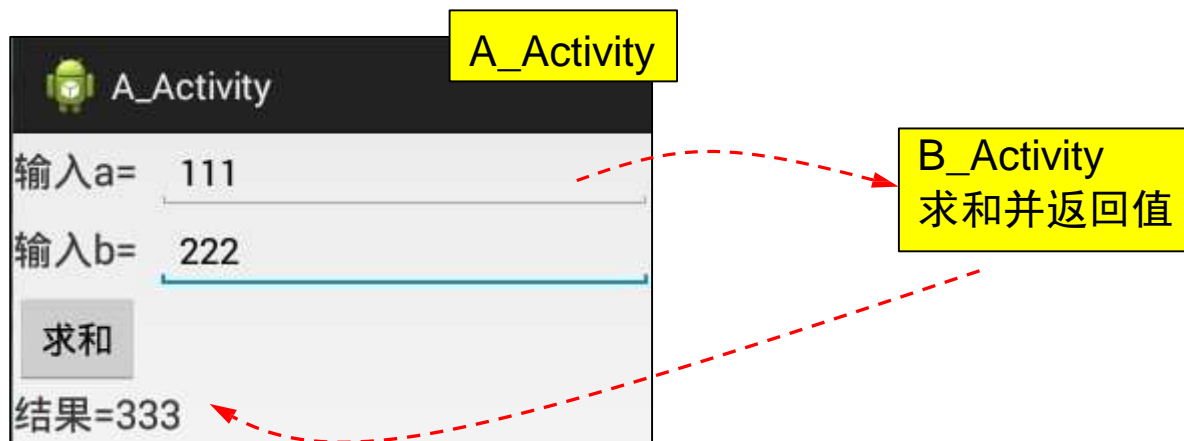
表明是B的回传

B回传的某个结果标记

B中传回的其他一个标记(如果有的话)

## 双向传递数据示例1

- A\_Activity输入两个数，B\_Activity求和并返回值



## A\_Activity主要代码1

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ...
    Button btn=(Button)findViewById(R.id.button1);
    btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            EditText et1=(EditText)findViewById(R.id.editText1);
            EditText et2=(EditText)findViewById(R.id.editText2);
            String a=et1.getText().toString();
            String b=et2.getText().toString();
            Intent intent=new Intent(A_Activity.this,B_Activity.class);
            Bundle bundle=new Bundle();
            bundle.putString("a", a);
            bundle.putString("b", b);
            intent.putExtras(bundle);
            startActivityForResult(intent, 100); //关键语句
        }
    });
}
```

## B\_Activity主要代码

@Override

**protected void onCreate(Bundle savedInstanceState) {**

...

Intent intent=getIntent();

Bundle bundle=intent.getExtras();

String a=bundle.getString("a");

String b=bundle.getString("b");

int sum=Integer.parseInt(a)+Integer.parseInt(b);

bundle.putInt("sum", sum); //换成int数据

intent.putExtras(bundle);

**setResult(RESULT\_OK, intent);** //关键语句

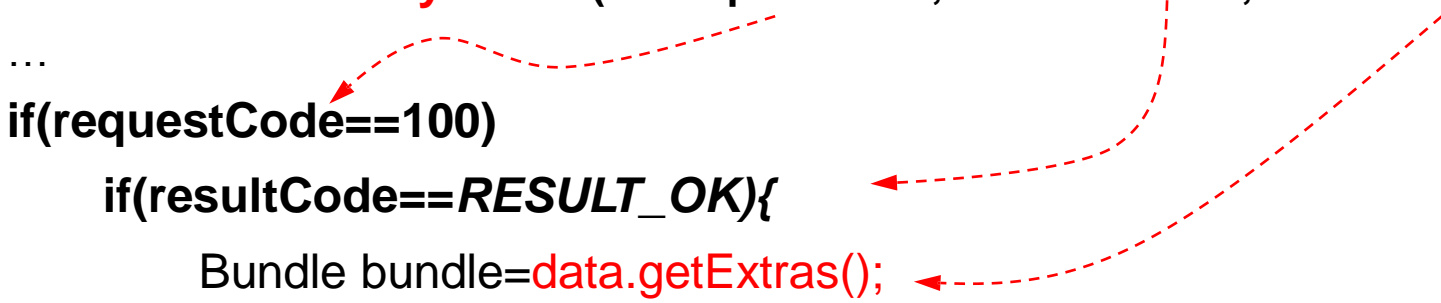
**finish();** //关键语句

}

## A\_Activity主要代码2 -- 关键事件onActivityResult

@Override

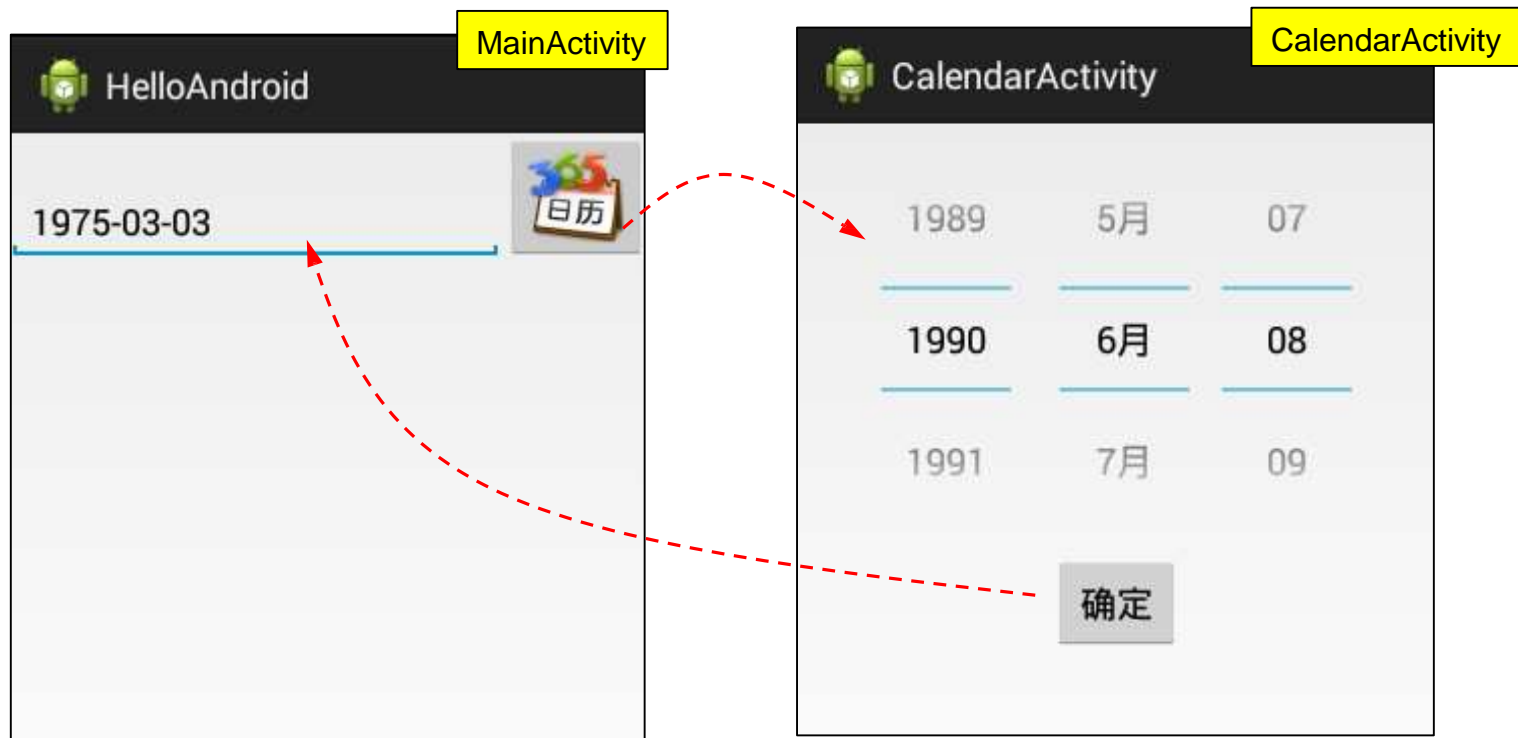
```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
    ...  
    if(requestCode==100)  
        if(resultCode==RESULT_OK){  
            Bundle bundle=data.getExtras();  
            int s=bundle.getInt("sum");  
            TextView tv=(TextView)findViewById(R.id.textView3);  
            tv.setText("结果="+s);  
        }  
    }  
}
```





## 双向传递数据示例2 -- 自学

- 将DatePicker控件单独作为一个activity。



## MainActivity主要代码1

```
ImageButton imgBt=(ImageButton)findViewById(R.id.imageButton1);
imgBt.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        EditText et=(EditText)findViewById(R.id.editText1);
        Intent intent=new Intent(MainActivity.this,CalendarActivity.class);
        Bundle bundle=new Bundle();
        bundle.putString("date", et.getText().toString());
        intent.putExtras(bundle);
        startActivityForResult(intent,100);
    }
});
```

将文本框录入的日期传到CalendarActivity中显示

## MainActivity主要代码2

```
@Override
```

```
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
    super.onActivityResult(requestCode, resultCode, data);
```

```
    if(requestCode==100){
```

```
        if(resultCode==RESULT_OK){
```

```
            Bundle bundle=data.getExtras();
```

```
            EditText et=(EditText)findViewById(R.id.editText1);
```

```
            et.setText(bundle.getInt("year")+"-"+bundle.getInt("month")+"-"+bundle.getInt("day"));
```

```
        }
```

```
    }//end if
```

```
}
```

接收CalendarActivity  
回传的日期值并显示

## CalendarActivity主要代码1

```
final Intent intent=this getIntent();  
Bundle bundle=intent.getExtras();  
String sdate=bundle.getString("date");
```

取出从MainActivity传来的日期串

```
Date date=null;
```

```
try{
```

```
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
```

```
    date=(Date)sdf.parse(sdate);
```

将日期串转换为指定格式的Date类型(年/月/日)

```
}catch(Exception e){
```

```
    Toast.makeText(CalendarActivity.this, "传入的日期格式不正确", Toast.LENGTH_SHORT).show();
```

```
}
```

```
Calendar cal=Calendar.getInstance(); //此时cal值为当前系统日期
```

Java不推荐使用Date类型  
建议用Calendar对象

```
if(date!=null){
```

```
    cal.setTime(date);
```

用Date值设置Calendar

这里使用Calendar目的是利用  
Calendar来初始化DatePicker

```
}
```

```
final DatePicker dp=(DatePicker)findViewById(R.id.datePicker1);
```

```
dp.setCalendarViewShown(false);
```

## CalendarActivity主要代码2

```
dp.init(cal.get(Calendar.YEAR), cal.get(Calendar.MONTH),  
        cal.get(Calendar.DAY_OF_MONTH), null);
```

将Calendar的年月日取出  
来初始化DatePicker

```
Button bt=(Button)findViewById(R.id.button1);  
bt.setOnClickListener(new View.OnClickListener() {
```

```
@Override
```

```
public void onClick(View arg0) {
```

```
Bundle bundle=new Bundle();  
bundle.putInt("year", dp.getYear());  
bundle.putInt("month", dp.getMonth()+1);  
bundle.putInt("day", dp.getDayOfMonth());  
intent.putExtras(bundle);  
setResult(RESULT_OK, intent);  
finish(); //不能少
```

将DatePicker选定的日期  
回传给MainActivity

```
}
```

```
});
```

【完】