Android移动开发

Android Mobile Application Development

第6讲 组件通信

吴以凡

计算机学院 一教505

yfwu@hdu.edu.cn

Intent

Intent的作用

- Intent提供了应用程序间的交互机制，可被看成不同组件之间的信使。
- Android根据Intent的描述，负责找到对应的组件，将 Intent传递给调用的组件，并完成组件的调用。
- 通过Intent，可实现同一或不同应用程序中的组件之间运行时动态绑定。

Intent的作用

- Intent可以用来启动:
  - Activity（活动）
    - Context.startActivity(), Context.startActivityForResult()
  - Services（服务）
    - Context.startService(), Context.bindService()
  - BroadcastReceiver（广播接收器)
    - Context.sendBroadcast()

Intent属性

- Intent类的属性包括：
  - Component(组件)
  - Action（行为）
  - Data（数据）
  - Category（分类）
  - Extras（扩展信息）
  - Flags（标志）

Intent属性

- Component(组件)
  - 处理Intent组件名称
    - 如：com.hdu.helloworld.MainActivity
  - 设置与获取该属性的方法
    - setComponent()、getComponent()
- Action（行为）
  - 指定Intent要完成的动作
  - Intent类定义了Action常量
    - 如：android.intent.action.ACTION_CALL
  - 设置与获取该属性的方法
    - setAction()、getAction()

Intent属性

- Data（数据）
  - 由两部分组成
    - URI：Universal Resource Identifier 统一资源标识符
    - MIME：Multipurpose Internet Mail Extensions
  - 内容格式由Action属性决定
    - 如：ACTION_CALL：电话号码、ACTION_VIEW：网址
  - 常用的Data属性格式：
    - file:
    - content:
    - smsto:
    - http:
    - mailto:
    - tel:

Intent属性

- Categroy（分类）
  - 处理Intent的组件的分类信息，是对Action的补充
  - 一个Intent对象可以包含多个Category属性
  - 设置与获取该属性的方法
    - addCategory()、removeCategory()、getCategory()
- Extras（扩展信息）
  - 用于传递目标组件所需的额外数据
  - 设置与获取该属性的方法
    - putExtras()、getExtras()

Intent属性

- Flags（标志）
  - 指示Android系统如何去启动一个活动、及启动后如何处理
  - Intent类（android.content.Intent）定义了若干个Flags
    - FLAG_ACTIVITY_NEW_TASK
    - FLAG_ACTIVITY_CLEAR_TOP
    - FLAG_ACTIVITY_SINGLE_TOP
    - FLAG_ACTIVITY_CLEAR_WHEN_TASK_RESET
    - FLAG_ACTIVITY_RESET_TASK_IF_NEEDED

Intent启动机制

- Intent是一种同一或不同应用程序中的组件之间调用的机制
- Android根据Intent的描述找到对应的组件，并将Intent传递给调用的组件
- 按照Intent的处理方式，Intent可以分为显式Intent和隐式Intent两类。因此Intent的启动可分为：
  - 显式启动
  - 隐式启动

显式启动

- 显式Intent是一种明确被启动的组件名字的Intent，通常通过Intent下面的方式指定处理该Intent的组件，这样Android系统就能根据指定的名字启动对应的组件
  - setComponent()
  - setClassName()
  - setClass()

```
Intent intent = new Intent();
intent.setClass(MainActivity.this, SupplActivity.class);
startActivity(intent);
```

隐式启动

- 隐式Intent没有指定被启动的组件名字
- 虽然隐式Intent中没有明确被启动的组件，但Android操作系统会根据隐式Intent中包含的Action(动作)属性、Category(类别)属性和Data属性(数据)，使用Intent Filter查找最合适处理该意图的组件

隐式Intent示例

- Action属性和Data属性可以在大多数情况下表达一个完整的意图
  - ACTION_VIEW content://contacts/people/1 -- 显示编号为1的人相关信息
  - ACTION_DIAL content://contacts/people/1 -- 给编号为1的人打电话
  - ACTION_VIEW tel:123 -- 将号码123显示
  - ACTION_DIAL tel:123 -- 拨打号码123
  - ACTION_EDIT content://contacts/people/1 -- 编辑编号为1的联系人
- Category属性给意图添加一些约束，使意图更加精确
  - 主Activity都需要能够接受一个Category为CATEGORY_LAUNCHER、Action为ACTION_Main的意图

```
<activity android:name=".MainActivity" android:label="主活动" >
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

Intent Filter

- Intent Filter （意图过滤器）提供了Intent过滤的功能。通过Intent Filter，可以依据Action(动作)属性、Category(类别)属性和Data属性(数据)来过滤Intent
- Intent Filter定义了接受Intent的能力，这种能力表示活动(Activity)、服务（Service）、广播接收者(BroadcastReceiver)等组件能够接受的Intent
- 每个活动(Activty)、服务（Service）、广播接收者(BroadcastReceiver)可以有一个或多个Intent Filter

生成Intent Filter

- Android提供了两种生成Intent Filter方式
  - 通过IntentFilter类生成，该类的常用方法包括：
    - addAction(String action)、getAction (int index)
    - addCategory(String category)、hasCategory(String category)
- 通过应用程序的清单文件AndroidManifest.xml定义<intent-filter>生成

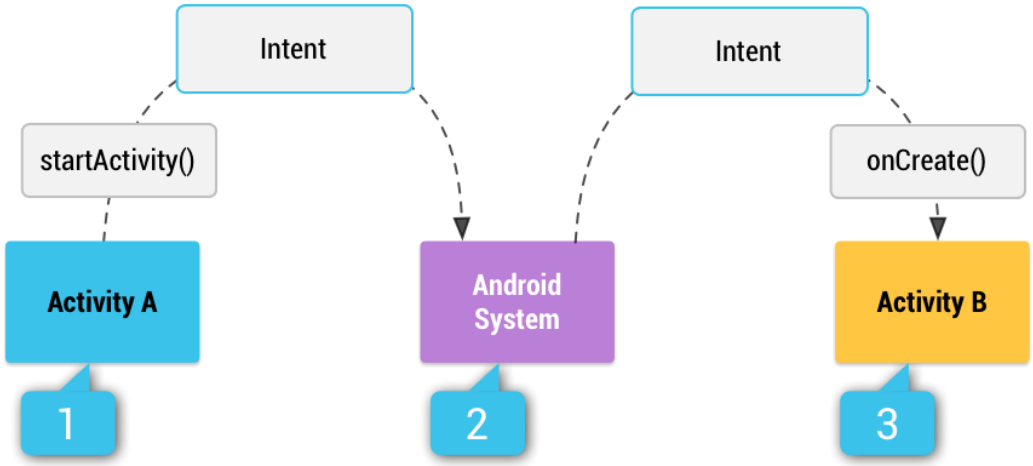
```
<intent-filter>
  <action android:name="android.intent.ACTION_BATTERY_LOW" />
  <action android:name="android.intent.ACTION_BATTERY_OKAY" />
  <action android:name="android.intent.ACTION_POWER_CONNECTED" />
  <action android:name="android.intent.ACTION_POWER_DISCONNECTED" />
</intent-filter>
```

Intent Filter的过滤测试

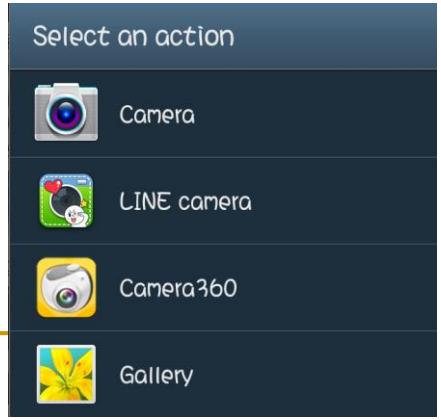
- Intent Filter被用于匹配隐式Intent
- 当匹配一个Intent意图对象时，该Intent需要经历以下方面的测试：
  - 动作测试 <action … />
  - 类别测试 <category … />
  - 数据测试 <data … />

使用Intent Filter实现隐式启动

- [1] 活动A创建隐式Intent，并传递给startActivity()
- [2] Android操作系统搜索所有app，寻找一个可以匹配该Intent的Intent Filter
- [3] 若找到该Intent Filter，系统启动包含该Intent Filter的app的主活动(Activity B)，并将Intent传递给它的onCreate()



* 若找到多个Intent Filter可匹配，系统显示对话框让用户选择



常用的Intent Action

- Intent中指定了Intent要完成的动作（最多指定一个Action），通常是一个字符串。在Intent类里面定义了一些常用的Action常量属性：
  - ACTION_CALL　　发起一个电话应用
  - ACTION_EDIT　　显示数据以供用户编辑
  - ACTION_MAIN　　初始化操作，这个操作既没有输入也没有输出
  - ACTION_VIEW　　显示数据给用户
  - ACTION_DIAL　　显示打电话的界面
  - ACTION_SEND　　发送短信

常用的Intent Action

- ACTION_SCREEN_ON　　打开移动设备屏幕
- ACTION_SYNC　　同步服务器与移动设备之间的数据
- ACTION_BATTERY_LOW　　警告设备电量低
- ACTION_TIMEZONE_CHANGED 移动设备时区发生变化
- ACTION_GET_CONTENT　　获取内容
- ACTION_ANSWER　　应答电话
- ACTION_GTALK_SERVICE_CONNECTED Gtalk已建立连接
- ACTION_GTALK_SERVICE_DISCONNECTED　Gtalk已断开连接
- ACTION_INPUT_METHOD_CHANGED　　改变输入法
- ACTION_PACKAGE_INSTALL　下载并且完成安装

Intent Action典型应用-浏览网页

- 使用ACTION_VIEW浏览网页：

```
String linkString = “http://www.baidu.com”; //定义被浏览网页的地址
Uri myUri = Uri.parse(linkString);//转换成URI格式
Intent myIntent = new Intent(Intent.ACTION_VIEW, myUri);//生成Intent对象
startActivity(myIntent);//启动浏览网页应用
```

Intent Action典型应用-搜索

- 使用ACTION_WEB_SEARCH 启动google的内容搜索

```
String searchString = “Android 程序设计”; //定义被搜索的内容
Intent myIntent = new Intent();//生成Intent对象
myIntent.setAction(Intent.ACTION_WEB_SEARCH);
myIntent.putExtra(SearchManager.QUERY, searchString);//将搜索的内容放到Intent中
startActivity(myIntent);//启动内容搜索
```

Intent Action典型应用-打电话

- 使用ACTION_DIAL打电话

```
String phoneString= “tel:+16327100001; //定义被叫号码
Uri myUri = Uri.parse(phoneString);//将电话号码转换成URI格式
Intent myIntent = new Intent(Intent. ACTION_DIAL, myUri);//生成Intent对象
startActivity(myIntent);//启动打电话应用
```

Intent Action典型应用-打开录音设备

- 使用Media.RECORD_SOUND_ACTION打开录音设备

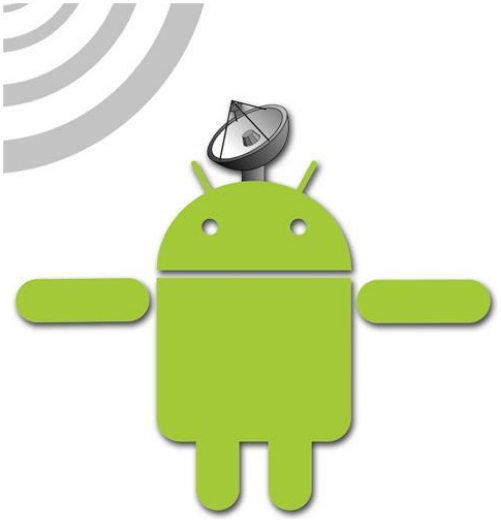
```
Intent myIntent = new Intent(Media.RECORD_SOUND_ACTION);
startActivity(myIntent);
```

Intent Action典型应用-打开地图

- 使用ACTION_VIEW打开地图

```
String locationString= “geo:38.899533,-77.036476”; //定义经纬度
Uri myUri = Uri.parse(locationString);//转换成URI格式
Intent myIntent = new Intent(Intent.ACTION_VIEW, myUri);//生成Intent对象
startActivity(myIntent);//在地图应用中打开Intent中指定的经纬度
```

Broadcast



广播Broadcast简介

- 在Android系统中，有一些操作完成以后，会发送广播。如果某个程序接收到这个广播，就会做出相应的处理。
  - 广播可以被多个应用程序所接收，也可以不被任何应用程序所接受
- BroadcastReceiver是负责对发送出来的广播进行过滤接收并响应的一类组件
  - BroadcastReceiver和事件处理机制类似，不同的是广播处理机制是系统级别的，而事件处理机制用于应用程序组件级别的

Broadcast中的Intent

- Andorid使用Broadcast机制来把Intent传递给多个应用和多个Activity
  - 例：手机电量低时，广播给所有当前运行的Activity，这些Activity响应该广播（如：进入省电模式）

实现Broadcast的步骤

- Broadcast机制的实现包含四个步骤
  - 第一步需要注册相应的BroadcastReceiver，广播接收器是接收广播消息并对消息作出反应的组件，如电量较低时的通知信息
  - 第二步发送广播，这个过程将消息内容和用于过滤的信息封装起来，并广播给BroadcastReceiver
    - 把要发送和用于过滤的信息装入一个Intent对象，然后通过调用Context.sendBroadcast()、sendOrderBroadcast()或者sendStickyBroadcast()方法将Intent对象以广播的方式发送出去
  - 第三步满足条件的BroadcastReceiver执行onReceive()方法
    - 所有已经注册的BroadcastReceiver会检查注册时的IntentFilter是否与发送的Intent相匹配，若匹配则会调用BroadcastReceiver的onReceive()方法
  - 最后执行onReceive()方法，销毁广播接收器

BroadcastReceiver创建

- 继承BroadcastReceiver基础类，并重写onReceive()

```
public class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        ...
    }
}
```

BroadcastReceiver注册

- 根据IntentFilter注册广播Intent
  - Java注册（动态注册）

```
IntentFilter myfilter = new IntentFilter("android.provider.Telephony.SMS_RECEIVED");
MyReceiver myreceiver = new MyReceiver();
Context.registerReceiver(myreceiver, myfilter);  // 注册
Context.unregisterReceiver(myreceiver);  // 注销
```

  - XML注册（静态注册）

```
<receiver android:name=".MyReceiver">
   <intent-filter>
      <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
   </intent-filter>
</receiver>
```

Broadcast Intent 广播

- 3种广播Intent的方式
  - Context.sendBroadcast
    - 满足条件的BroadcastReceiver都会执行onReceive()方法；不严格保证执行顺序
  - Context.sendOrderBroadcast
    - 满足条件的BroadcastReceiver都会执行onReceive()方法；保证执行顺序，根据IntentFilter设置的优先级顺序执行
  - Context.sendStickyBroadcast (API 21 (5.0) 后作废)
    - “粘着”功能：保存发送的Intent

Broadcast Intent 接收

- BroadcastReceiver收到广播的Intent，对Intent进行判断，如果该BroadcastReceiver满足条件，执行onReceive()方法
- 所有包含匹配的IntentFilter的BroadcastReceiver都会被激活
- 只有BroadcastReceiver能接收Broadcast Intent，Activity和Service不能
  - Activity只能接收startActivity()传递的Intent
  - Service只能接收startService()传递的Intent

Broadcast Intent 接收

- 例：手机电量低时，系统发送一个广播，该广播的Action为ACTION_BATTERY_LOW，收到该广播后，所有包含相匹配的IntentFilter的BroadcastReceiver会执行onReceive()的处理代码，进入节电模式

```
public void onReceive(Context mycontext, Intent myintent) {
   if (myintent.getAction().equals(Intent. ACTION_BATTERY_LOW) {
      // 添加低电量的处理代码，如关闭WIFI和GPS以节点模式运行
   }
}
```

BroadcastReceiver销毁

- 每次广播消息到来时，都会创建BroadcastReceiver实例并执行onReceive()方法；onReceive()方法执行完后，BroadcastReceiver实例被销毁
- onReceive()方法包含的代码必须简洁快速

实例：静态注册方式

```
// 事件处理发送广播
send_btn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
      Intent intent=new Intent();
      intent.setAction("MyBroadcast");
      //发送广播
      MainActivity.this.sendBroadcast(intent);
    }
});
```

```
//注册广播接收器
<receiver android:name=".MyReceiver">
      <intent-filter>
          <action android:name="MyBroadcast" />
      </intent-filter>
</receiver>
```

```
// 自定义广播接收器
public class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // 收到广播时，显示一个通知
        Toast.makeText(context,"广播接收成功",Toast.LENGTH_SHORT).show();
    }
}
```

* Android 8.0+静态注册自定义广播失效

实例：动态注册方式

```
//事件处理发送广播
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        myReceiver = new MyReceiver();
        intentFilter = new IntentFilter();
        intentFilter.addAction("android.provider.Telephony.SMS_RECEIVED");
        //代码动态注册广播
        MainActivity.this.registerReceiver(myReceiver, intentFilter);
    }
});
```


```
//自定义广播接收器
public class MyReceiver extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context,"收到短信",Toast.LENGTH_SHORT).show();
    }
}
```

```
//添加权限
<uses-permission android:name="android.permission.RECEIVE_SMS" />
```