第五章

全局大喇叭--详解广播机制

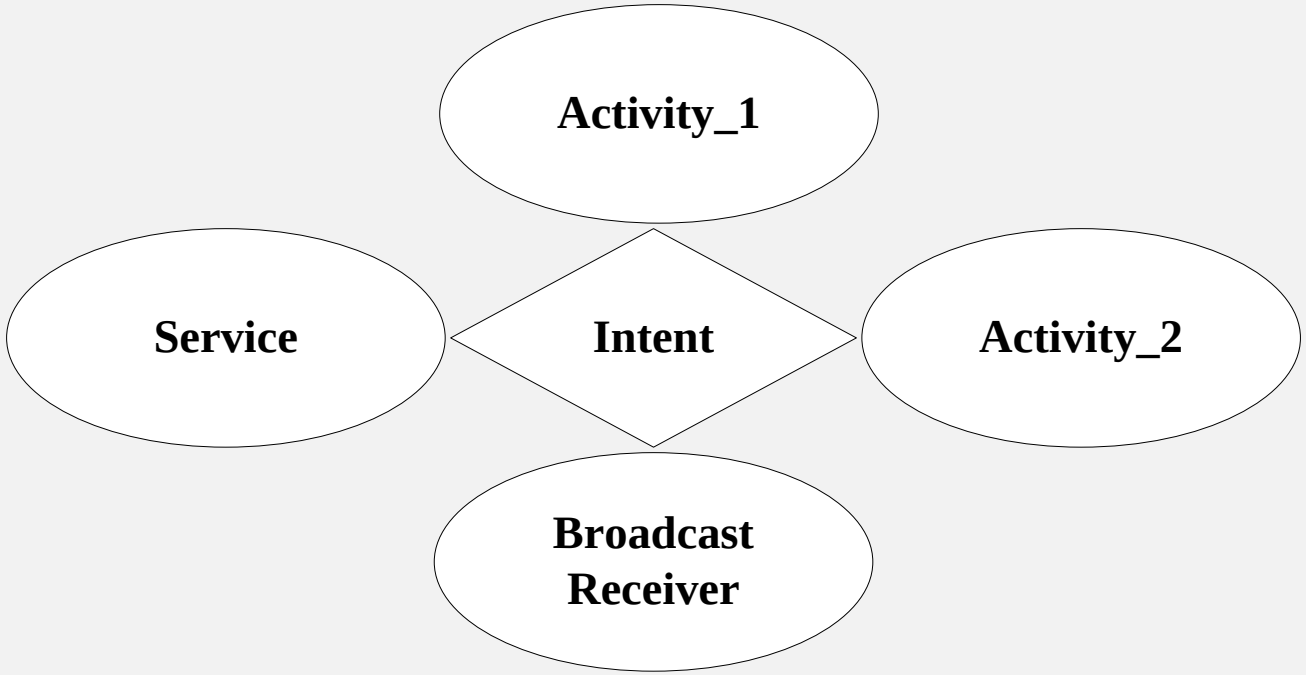
主讲：王海

本章目标

- 掌握Activity的创建及生命周期方法
- 能够访问Android中的各种资源
- 理解AndroidManifest.xml清单文件
- 掌握Android应用程序生命周期
- 掌握Application类及生命周期事件

Intent原理及分类

- Intent是Android应用内不同组件之间的通讯载体
- 使用Intent可以激活Android的三个核心组件：
  - Activity
  - Service
  - BroadcastReceiver



使用Intent启动Activity、Service和BroadcastReceiver三大组件

- 启动Activity：通常需要调用startActivity(Intent intent)或startActivityForResult(Intent intent,int requestCode)方法，其中Intent参数用于封装目标Activity所需信息
- 启动Service：通常需要调用startService(Intent intent)或bindService(Intent intent,ServiceConnection conn ,int flags)方法，其中Intent参数用于封装目标Service所需信息
- 触发BroadcastReceiver时，调用sendBroadcast(Intent intent)方法来发送广播信息，其中Intent参数用于封装目标BroadcastReceiver所需信息

Intent意图类

- 显式Intent——明确指定需要启动或触发组件的类名
- 隐式Intent——指定了需要启动或触发的组件应满足的条件

Intent属性

- Component组件
- Action动作
- Category类别
- Data数据
- Type数据类型
- Extras扩展信息
- Flags标志位

Component组件

- Component组件为目标组件，需要接受一个ComponentName对象
- ComponentName对象的构造方法有以下几种方式：
  - ✓ ComponentName(String pkg,String className)
  - ✓ ComponentName(Context context,String className)
  - ✓ ComponentName(Context context,Class<?> className)
- Intent还可以指定待启动组件的包名和类名：
  - ✓ setClass(Context ctx,Class<?> cls)
  - ✓ setClassName(Context ctx,String className)
  - ✓ setClassName(String pkg,String className)

Component组件

- 创建ComponentName对象

```
Intent intent =new Intent();
ComponentName component = new ComponentName(MainActivity.this
                          ,SecondActivity.class);
intent.setComponent(component);
startActivity(intent);
```

- 使用setClass()方法指定待启动组件

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

Context对象

Class对象

- 使用Intent()构造方法指定启动组件

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

Action动作

- Action是一个字符串，用于描述一个Android应用程序的组件
- 启动Activity的系统标准Action

Action常量	字符串	描述
ACTION_SEND	android.intent.action.SEND	启动一个Activity，该Activity会发送Intent中指定的数据。接收人需要由解析的Activity来选择。
ACTION_SENDTO	android.intent.action.SENDTO	启动一个Activity来向Intent的数据URI所指定的联系人发送一条消息。
ACTION_ANSWER	android.intent.action.ANSWER	打开一个处理来电的Activity，通常这个动作是由本地电话拨号程序处理
ACTION_INSERT	android.intent.action.INSERT	打开一个子Activity能在Intent的数据URI指定的游标处插入新项的Activity。当作为子Activity调用时，应该返回一个指向新插入项的URI
ACTION_DELETE	android.intent.action.DELETE	启动一个Activity，允许删除Intent的数据URI中指定的数据
ACTION_ALL_APPS	android.intent.action. ACTION_ALL_APPS	打开一个列出所有已安装应用程序的Activity，通常此操作由启动器处理
ACTION_SEARCH	android.intent.action.SEARCH	通常用于启动特定的搜索Activity。

标准Action常量

- ACTION_BOOT_COMPLETED：系统启动完成广播
- ACTION_TIME_CHANGED：时间改变广播
- ACTION_DATE_CHANGED：日期改变广播
- ACTION_TIME_TICK：每分钟改变一次时间
- ACTION_TIMEZONE_CHANGED：时区改变广播
- ACTION_BATTERY_LOW：电量低广播
- ACTION_PACKAGE_ADDED：添加包广播
- ACTION_PACKAGE_REMOVED：删除包广播

Category类别

- Category属性用来描述动作的类别

Category常量	字符串	描述
CATEGORY_DEFAULT	android.intent.category.DEFAULT	默认的Category
CATEGORY_BROWSABLE	android.intent.category.BROWSABLE	指定Activity能被浏览器安全调用
CATEGORY_TAB	android.intent.category.TAB	指定Activity能作为TabActivity的Tab页
CATEGORY_LAUNCHER	android.intent.category.LAUNCHER	Activity显示顶级程序列表中
CATEGORY_INFO	android.intent.category.INFO	用于提供包信息
CATEGORY_HOME	android.intent.category.HOME	设置该Activity随系统启动而运行
CATEGORY_PREFERENCE	android.intent.category.PREFERENCE	该Activity是参数面板
CATEGORY_TEST	android.intent.category.TEST	该Activity是一个测试
CATEGORY_CAR_DOCK	android.intent.category.ANSWER	指定手机被插入汽车底座时运行该Activity
CATEGORY_DESK_DOCK	android.intent.category.CAR_DOCK	指定手机被插入桌面底座时运行该Activity
CATEGORY_CAR_MODE	android.intent.category.CAR_MODE	设置该Activity可以在车载环境下使用

Data数据

- Data属性通常用与Action属性结合使用
- 为Intent提供可操作的数据
- Data属性接收URI对象
  - 【语法】

```
scheme://host:port/path
```

  - 【示例】URI字符串

```
http://www.baidu.com
```

Type数据类型

- Data属性与Type属性之间能够相互覆盖：
  - ✓ 如果为Intent先设置Data属性，再设置Type属性，那么Type属性将会覆盖Data属性
  - ✓ 如果为Intent先设置Type属性，再设置Data属性，那么Data属性将会覆盖Type属性
  - ✓ 如果希望Intent既有Data属性也有Type属性，应该调用Intent的setDataAndType()方法

Extras扩展信息

- Extras属性是一个Bundle对象
- 用于在多个Activity之间交换数据
- Extras属性的使用过程
  - 使用Extras属性

```
Bundle bundle= new Bundle();
bundle.putString("test", "this is a test");
Intent intent = new
Intent(MainActivity.this,SecondActivity.class);
intent.putExtras(bundle);
startActivity(intent);
```

  - 通过getExtras()方法获得Bundle对象并进行取值

```
Bundle bundle = this.getIntent().getExtras();
String test = bundle.getString("test");
```

Flags标志位

- Flag属性用于为Intent添加额外的控制标志
- 通过Intent的addFlags()方法为Intent添加控制标志
- 常用的Flag值：
  - ✓ FLAG_ACTIVITY_CLEAR_TOP
  - ✓ FLAG_ACTIVITY_NEW_TASK
  - ✓ FLAG_ACTIVITY_NO_HISTORY
  - ✓ FLAG_ACTIVITY_SINGLE_TOP

使用Intent启动Activity

- 通过调用Context的startActivity()方法可以创建并显示目标Activity

```
startActivity(myIntent);
```

- startActivity()方法会查找并启动一个与Intent参数相匹配的Activity
- startActivityForResult()方法启动Activity并跟踪子Activity的反馈

显式Intent启动Activity

- 通过Intent来显式地指定要打开的Activity

```
Intent intent = new Intent(MyActivity.this,
MyOtherActivity.class);
startActivity(intent);
```

隐式Intent启动Activity

- 使匿名的应用程序组件响应动作请求
- 隐式Intent启动Activity

```
Intent intent =new Intent(Intent.ACTION_DIAL, Uri.parse("tel:555-2368"));
startActivity(intent);
```

- 构建一个隐式的Intent时，需要指定一个所要执行的动作
- 通过向Intent添加Extra的方式来向目标Activity发送额外的数据
- 当多个Activity都能够执行指定的动作时，会向用户呈现各种选项供用户手动选择

使用Intent来启动内置应用程序

- 启动浏览器

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

- 启动地图

```
Intent i=new Intent(Intent.ACTION_VIEW,
      Uri.parse("geo:25.04692437135412,121.516178395967 8"));
startActivity(i);
```

- 打电话

```
Intent i=new Intent(Intent.ACTION_DIAL, Uri.parse("tel:+1234567"));
startActivity(i);
```

- 发送电子邮件

```
Intent i=new Intent(Intent.ACTION_SENDTO,
Uri.parse("mailto:qst@163.com"));
startActivity(i);
```

使用Intent的resolveActivity()方法进行确认

```
//创建隐式Intent来启动新的Activity
Intent intent =new Intent(Intent.ACTION_DIAL， Uri.parse("tel:555-2368"));
//检查这个Activity是否存在
PackageManager pm = getPackageManager();
ComponentName cn = intent.resolveActivity(pm);
if (cn == null) {
    //如果这个Activity不存在则指向the Google Play Store
    Uri marketUri =Uri.parse("market://search?q=pname:com.myapp.packagename");
    Intent marketIntent = new Intent(Intent.ACTION_VIEW).setData(marketUri);
    //如果在Google Play Store中有则下载，否则报错。
    if (marketIntent.resolveActivity(pm) != null){
        startActivity(marketIntent);
        }else{
            Log.d(TAG， "Market client not available.");
        }
}else{
    startActivity(intent);
}
```

传递数据给其他Activity

- 通过Intent的putExtra()或putExtras()方法可以向目标Activity传递数据
- 使用putExtras()方法批量传递数据

```
Intent intent = new Intent();
Bundle bundle = new Bundle();//该类用作携带数据
bundle.putString("name","中华文明");
bundle.putString("address","青岛");
intent.putExtras(bundle);
```

- 使用putExtra()方法单个传递数据

```
Intent intent = new Intent();
intent.putExtra("name", "中华文明");
```

从Activity返回数据

- 使用startActivityForResult()方法启动一个Activity，新启动的Activity在关闭时可以向原Activity返回数据
- 启动一个目标Activity
  - 显示启动Activity

```
Intent intent = new Intent(this,
MyOtherActivity.class);
startActivityForResult(intent, 1);
```

  - 隐式启动Activity

```
Uri uri = Uri.parse("content://contacts/people");
Intent intent = new Intent(Intent.ACTION_PICK, uri);
startActivityForResult(intent, 2);
```

从Activity返回数据

- 从目标Activity中返回数据
  - 在目标Activity中调用finish()方法之前，通过setResult()方法向原Activity返回一个结果
  - setResult()方法是一个重载方法，其形式如下：
    - ✓ setResult(int resultCode)——设置传递到上一个界面的数据

```
setResult(Activity.RESULT_CANCELED);
```

    - ✓ setResult(int resultCode, Intent data) ——设置传递到上一个界面的数据

```
Intent result = new Intent(Intent.ACTION_PICK,
selectedHorse);
setResult(Activity.RESULT_OK, result);
```

处理从目标Activity返回的数据

- 重写onActivityResult()方法处理从目标Activity返回的结果

```
onActivityResult(int requestCode,  int resultCode,  Intent data));
```

  - ✓ requestCode是在启动目标Activity时所使用的请求码
  - ✓ resultCode表示从目标Activity返回的状态码
  - ✓ data是状态码对应的返回数据

Intent Filter过滤器

- 意图的过滤器， 用于描述指定的组件可以处理哪些意图
- 在Intent Filter中可以包含Intent对象的三个属性：
  - ACTION
  - DATA
  - CATEGORY
- 在AndroidManifest.xml中配置<intent-filter>元素

```
<intent-filter>
    <action android:name="com.example.project.SHOW_CURRENT" />
    <action android:name="com.example.project.SHOW_RECENT" />
    <action android:name="com.example.project.SHOW_PENDING" />
    ......
 </intent-filter>
```

Intent Filter过滤器

- 当Intent对象或者过滤器没有指定<action>时：
  - 如果一个Intent过滤器没有指定任何<action>，则不会匹配任何Intent，即所有的Intent都不会通过此测试；
  - 如果一个Intent对象没有指定任何<action>，而相应的过滤器中有至少一个<action>时将自动通过此测试。
- Category测试

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

Intent Filter过滤器

- Data测试

```
<intent-filter>
    <data android:mimeType="video/mpeg" android:scheme="http://" ... />
    <data android:mimeType="audio/mpeg" android:scheme="http://" ... />
    ......
</intent-filter>
```

- 每个<data >元素可以指定URI和data type属性
- URI属性由schema、host、port和path组成
  - **【语法】**

```
schema://host:port/path
```

  - **【示例】URI**

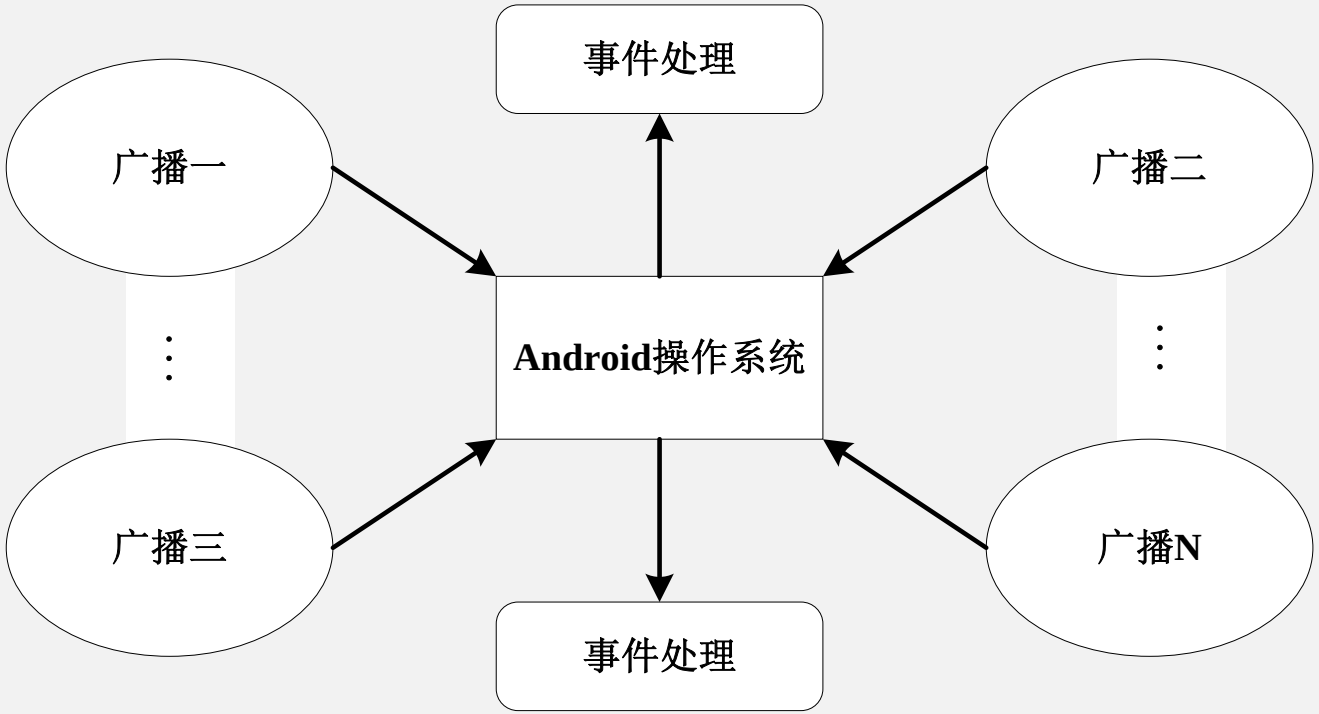
```
content://com.example.project:200/folder/subfolder/etc
```

BroadCastReceiver

- BroadcastReceiver是广播接收器，用于接收系统和应用中的广播
- BroadcastReceiver是一种对广播进行过滤接收并响应的组件
- 自身并不提供用户图形界面
- 本质上就是一个全局监听器，用于监听系统全局的广播消息

广播接收机制

- 电池的使用状态、电话的接收和短信的接收等都会产生一个广播
- 广播接收机制变相采用了事件处理机制



广播和接收Intent的步骤

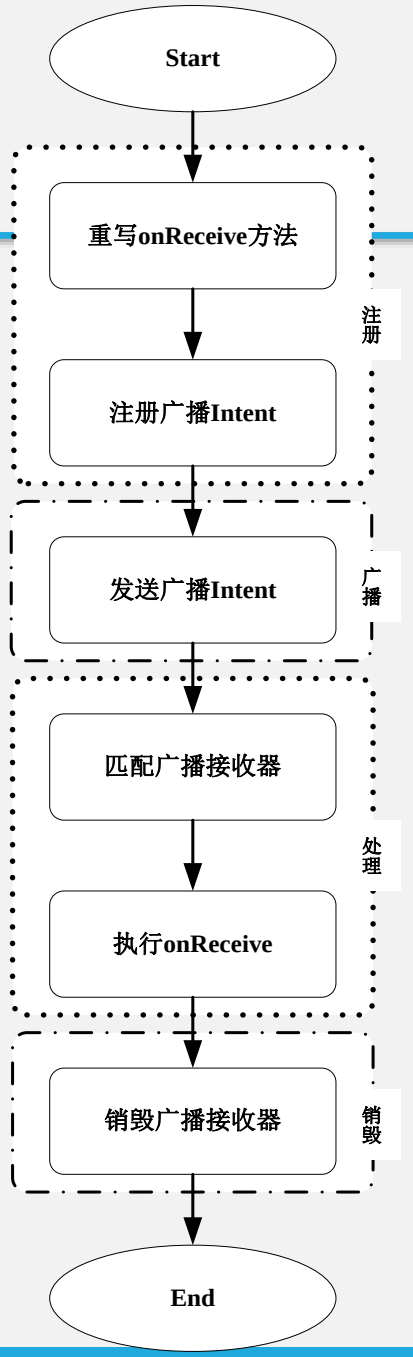
① 定义BroadCaseReceiver广播接收器

② 注册BroadCaseReceiver广播接收器

③ 发送广播

④ 执行

⑤ 销毁



使用BroadcaseReceiver步骤

① 定义一个BroadcaseReceiver的子类，并重写onReceive()方法，在接收到广播后进行相应的逻辑处理

② 在AndroidManifest.xml文件中注册广播接收器对象，并指明触发BroadcastReceiver事件的条件

③ 在AndroidManifest.xml中添加接收和发送短信权限

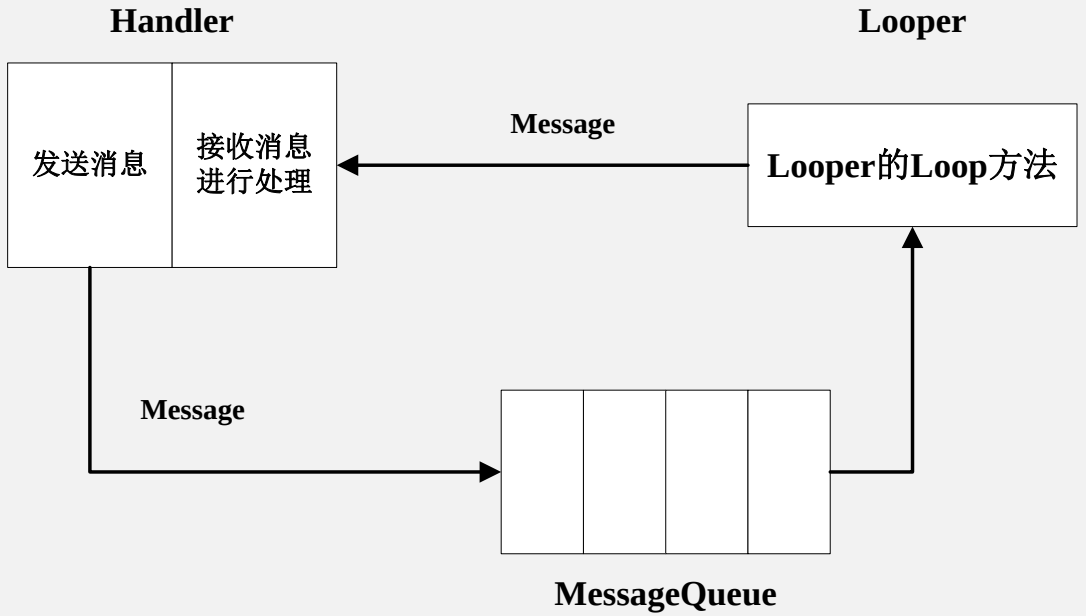
Handler消息传递机制

- Handler常用方法

方法	功能描述
handleMessage(Message msg)	通过重写该方法来处理消息
hasMessage(int what)	检查消息队列中是否包含what所指定的消息
hasMessage(int what，Object object)	检查队列中是否有指定的what和指定对象的消息
obtainMessage()	用于获取消息，具有多个重载方法
sendEmptyMessage(int what)	用于发送空消息
sendEmptyMessageDelayed(int what,long delayMillis)	用于在指定的时间之后发送空消息
sendMessage(Message msg)	立即发送消息
sendMessageDelayed(Message msg，long delayMillis)	用于在指定的时间之后发送空消息

Handler的工作机制

- 配合Handler工作的其他组件：
  - android.os.Message——用于封装线程之间传递的消息
  - android.os.MessageQueue——用于负责接收并处理Handler发送过来的消息
  - android.os.Looper——负责消息队列的管理



Looper类静态方法

- Looper.prepare()
  - ✓ 在线程中保存一个Looper实例，其中保存一个MessageQueue对象
  - ✓ Looper.prepare()方法在每个线程中只能调用一次，否则会抛出异常
  - ✓ 在一个线程中只会存在一个MessageQueue
- Looper.loop()
  - ✓ 当前线程通过无限循环的方式，不断从MessageQueue队列中读取消息
  - ✓ 回调Message.target.dispatchMessage(msg)方法将消息分配给Handler对象并进行处理
  - ✓ Message的target属性为所关联的Handler对象

AsyncTask类

- AsyncTask的语法格式：

```
public abstract class AsyncTask<Params,Progress,Result>
```

  - ✓ Params是启动任务执行的输入参数
  - ✓ Progress是后台任务执行的进度
  - ✓ Result是后台计算结果的类型

- 执行异步任务步骤：
  1. execute(Params... params)
  2. onPreExecute()
  3. doInBackground
  4. onProgressUpdate(Progress... values)
  5. onPostExecute(Result result)

使用AsyncTask注意事项

- 异步任务的实例必须在UI线程中创建
- execute(Params... params)方法必须在UI线程中调用
- 不能手动调用onPreExecute()、doInBackground() 等方法
- 不能在doInBackground(Params... params)方法中更改UI组件的信息
- 每个任务实例只能执行一次，当执行第二次时将会抛出异常

本章总结

- Intent是Android中的一个消息传递机制，提供了一种通用的消息系统，可以激活Android应用的三个核心组件：Activity、Service和BroadcastReceiver
- Intent对象是一个信息的捆绑包，包含了接收该Intent的组件所需要的信息
- 广播是一种广泛运用的在应用程序之间传输信息的机制。而BroadcastReceiver是对发送出来的广播进行过滤接收并响应的一类组件
- Intent类提供了许多Action，包括标准的Activity动作、标准的Broadcast动作、标准的类别动作和标准的附加数据动作
- Handler类的作用包括：在新启动的线程中发送消息和在主线程中获取和处理消息
- AsyncTask使得创建异步任务变得更加简单，不再需要编写任务线程和Handler实例也可完成相同的任务