2021

第6章 组件通信与广播消息

复旦大学 陈辰

学习目标

AIMS

01 了解使用Intent进行组件通信的原理

02 掌握使用Intent启动Activity的方法

03 掌握获取Activity返回值的方法

04 了解Intent过滤器的原理与匹配机制

05 掌握发送和接收广播消息的方法

6.1 Intent简介

- **Intent是一种轻量级的消息传递机制**
  - 是一个动作的完整描述，包含了动作的产生组件、接收组件和传递的数据信息
  - 用于组件之间数据交换
    - Activity、Service和BroadcastReceiver的数据交互
    - 启动Activity和Service
  - 发送广播消息
    - 应用程序广播消息
    - 系统广播消息
      - 手机的信号变化
      - 电池的电量过低

6.1 Intent简介

- **6.1.1 启动Activity**
  - 应用程序一般都有多个Activity，Intent可以实现不同Activity之间的切换和数据传递
  - 启动Activity方式
    - 显式启动
      - 必须在Intent中指明启动的Activity所在的类
    - 隐式启动
      - 根据Intent的动作和数据来决定启动哪一个Activity
      - 选择权有Android系统和最终用户来决定

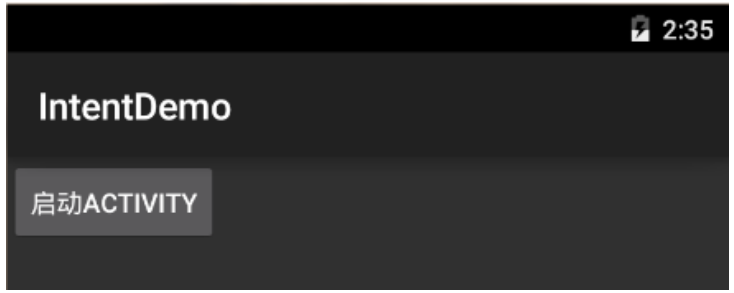
6.1 Intent简介

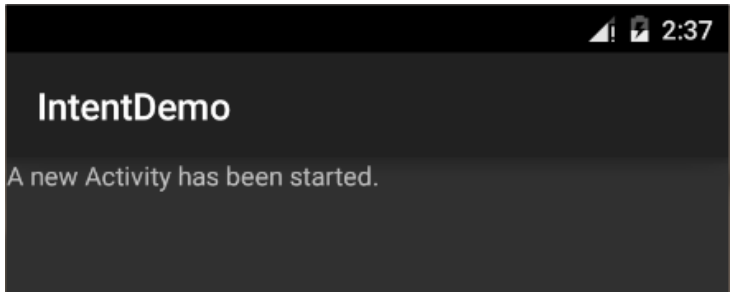
- **6.1.1 启动Activity**
  - 显式启动
    - 创建一个Intent
    - 指定当前的应用程序上下文，以及要启动的Activity
    - 把创建好的Intent作为参数传递给startActivity()方法

```
1. Intent intent = new Intent(IntentDemo.this, ActivityToStart.class);
2. startActivity(intent);
```

6.1 Intent简介

- **6.1.1 启动Activity**
  - 显式启动
    - 用IntentDemo示例说明如何使用Intent启动新的Activity
      - 示例中有两个Activity：IntentDemoActivity和NewActivity。
      - 程序默认启动的Activity是IntentDemo
      - 在点击“启动Activity”按钮后，启动的NewActivity





6.1 Intent简介

• 6.1.1 启动Activity

- 显式启动
  - 务必在AndroidManifest.xml文件中注册这两个Activity
  - AndroidManifest.xml文件代码如下:

```
1.  <?xml version="1.0" encoding="utf-8"?>
2.  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3.  package="edu.hrbeu.IntentDemo"
4.  android:versionCode="1"
5.  android:versionName="1.0">
6.  <application android:icon="@drawable/icon" android:label="@string/app_name">
7.  <activity android:name=".IntentDemo"
8.  android:label="@string/app_name">
9.  <intent-filter>
10. <action android:name="android.intent.action.MAIN" />
11. <category android:name="android.intent.category.LAUNCHER" />
12. </intent-filter>
13. </activity>
14. <activity android:name=".NewActivity"
15. android:label="@string/app_name">
16. </activity>
17. </application>
18. <uses-sdk android:minSdkVersion="14" />
19. </manifest>
```

6.1 Intent简介

• 6.1.1 启动Activity

- 显式启动
  - 在IntentDemoActivity.java文件中，包含了使用Intent启动Activity的核心代码：

```
1.  Button button = (Button)findViewById(R.id.btn);
2.  button.setOnClickListener(new OnClickListener(){
3.  public void onClick(View view){
4.  Intent intent = new Intent(IntentDemoActivity.this, NewActivity.class);
5.  startActivity(intent);
6.  }
7.  });
```

Intent构造函数的第1个参数是应用程序上下文，在这里就是IntentDemoActivity
第2个参数是接收Intent的目标组件，这里使用的是显式启动方式，直接指明了需要启动的Activity

6.1 Intent简介

- **6.1.1 启动Activity**
  - 隐式启动
    - 不需要指明需要启动哪一个Activity，而由Android系统和用户来决定具体启动哪个Activity
    - 有利于降低组件之间的耦合度
    - 隐式启动的原理：
      - 选择隐式启动Activity，Android系统会在程序运行时解析Intent，并根据一定的规则对Intent和Activity进行匹配，使Intent上的动作、数据与Activity完全吻合
      - 匹配的组件可以是程序本身的Activity，也可以是Android系统内置的Activity，还可以是第三方应用程序提供的Activity
      - 这种方式强调了Android组件的可复用性

6.1 Intent简介

• 6.1.1 启动Activity

- 隐式启动
  - 使用Intent隐式启动浏览器，查看指定的网页内容
    - 可将“浏览动作”和“Web地址”作为参数传递给Intent
    - Android系统则通过匹配动作和数据格式，找到最适合于此动作和数据格式的组件
      - 在匹配Intent时，根据动作Intent.ACTION_VIEW得知需要启动具备浏览功能的Activity，但具体是浏览电话号码还是浏览网页，还需要根据URI的数据类型来做最后判断
      - 因为数据提供的是Web地址"http://www.baidu.com"，所以最终可以判定Intent需要启动具有网页浏览功能的Activity
      - 在缺省情况下，Android系统会调用内置的Web浏览器

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

6.1 Intent简介

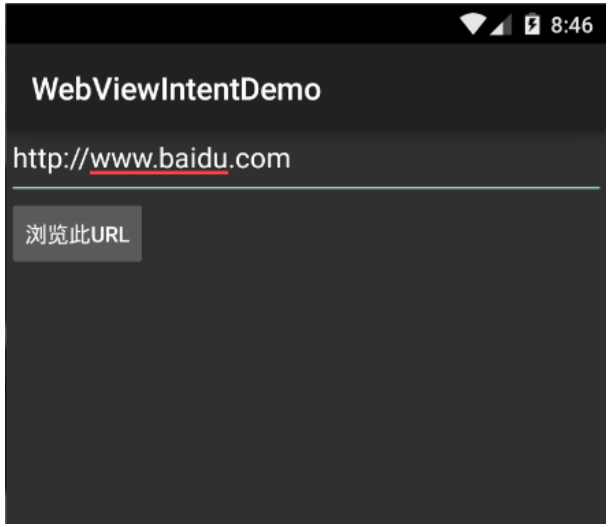
• 6.1.1 启动Activity

- 隐式启动
  - Android系统支持的“动作字符串常量”

动作	*说明*
ACTION_ANSWER	*打开接听电话的Activity，默认为Android内置的拨号界面*
ACTION_CALL	*打开拨号盘界面并拨打电话，使用Uri中的数字部分作为电话号码*
ACTION_DELETE	*打开一个Activity，对所提供的数据进行删除操作*
ACTION_DIAL	*打开内置拨号界面，显示Uri中提供的电话号码*
ACTION_EDIT	*打开一个Activity，对所提供的数据进行编辑操作*
ACTION_INSERT	*打开一个Activity，在提供数据的当前位置插入新项*
ACTION_PICK	*启动一个子Activity，从提供的数据列表中选取一项*
ACTION_SEARCH	*启动一个Activity，执行搜索动作*
ACTION_SENDTO	*启动一个Activity，向数据提供的联系人发送信息*
ACTION_SEND	*启动一个可以发送数据的Activity*
ACTION_VIEW	*最常用的动作，对以Uri方式传送的数据，根据Uri协议部分以最佳方式启动相应的Activity进行处理。对于http:address将打开浏览器查看；对于tel:address将打开拨号界面并呼叫指定的电话号码*
ACTION_WEB_SEARCH	*打开一个Activity，对提供的数据进行Web搜索*

6.1 Intent简介

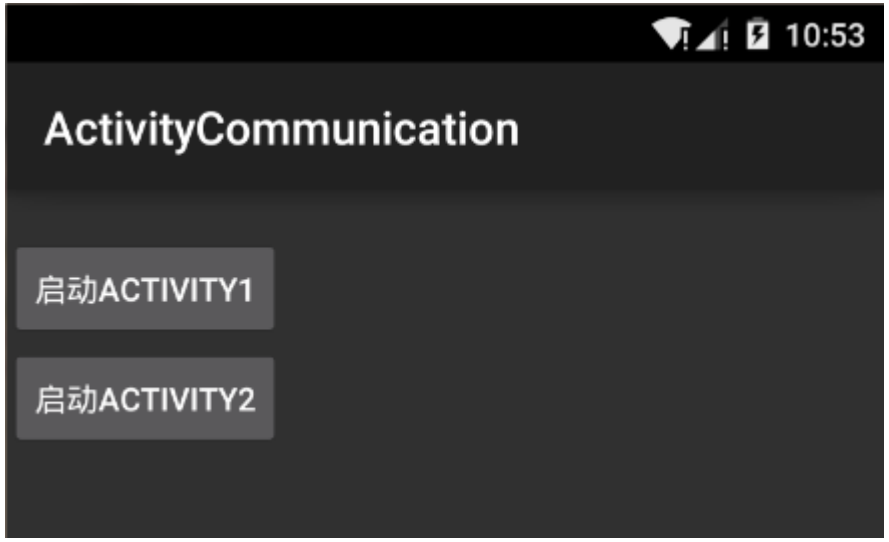
- **6.1.1 启动Activity**
  - 隐式启动
    - WebViewIntentDemo示例说明了如何隐式启动Activity

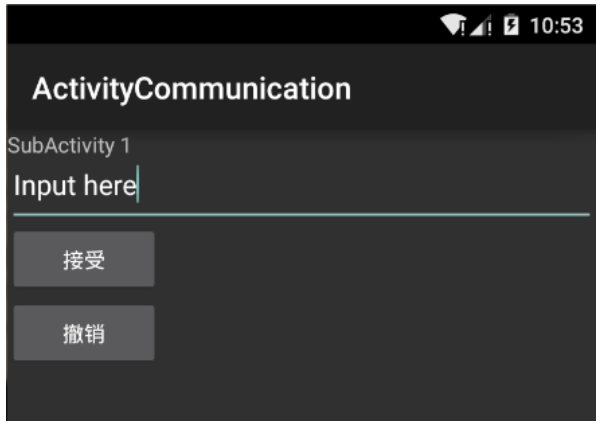


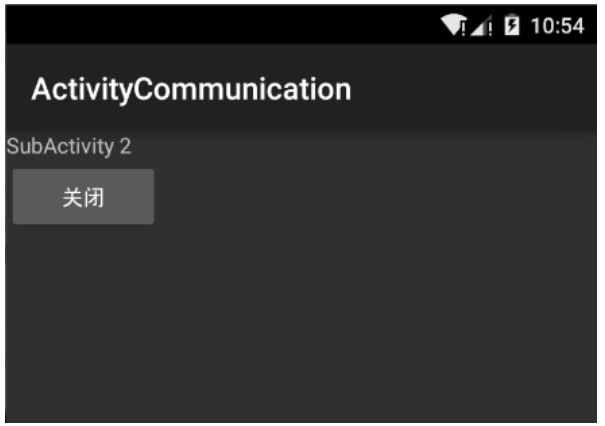

```
1.  String urlString = editText.getText().toString();
2.  Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(urlString));
3.  startActivity(intent);
```

6.1 Intent简介

- **6.1.2 获取Activity返回值**
  - 如何将“子Activity”的数据返回给“父Activity”?
    - 使用Sub-Activity的方式去启动子Activity
  - ActivityCommunication示例







6.1 Intent简介

- **6.1.2 获取Activity返回值**
  - 获取子Activity的返回值，一般分为三个步骤:
    - 以Sub-Activity的方式启动子Activity
    - 设置子Activity的返回值
    - 在父Activity中获取返回值

6.1 Intent简介

• 6.1.2 获取Activity返回值

- 以Sub-Activity的方式启动子Activity
  - 显式启动子Activity的代码如下:

```
1. int SUBACTIVITY1 = 1;
2. Intent intent = new Intent(this, SubActivity1.class);
3. startActivityForResult(intent, SUBACTIVITY1);
```

  - 隐式启动子Activity的代码如下:

```
1. int SUBACTIVITY2 = 2;
2. Uri uri = Uri.parse("content://contacts/people");
3. Intent intent = new Intent(Intent.ACTION_PICK, uri);
4. startActivityForResult(intent, SUBACTIVITY2);
```

startActivityForResult(Intent, requestCode)函数，参数Intent用于决定启动哪个Activity，参数requestCode是请求码。因为所有子Activity返回时，父Activity都调用相同的处理函数，因此父Activity使用requestCode来确定数据是哪一个子Activity返回的

6.1 Intent简介

• 6.1.2 获取Activity返回值

- 设置子Activity的返回值
  - 以下代码说明如何在子Activity中设置返回值:

```
1.  Uri data = Uri.parse("tel:" + tel_number);
2.  Intent result = new Intent(null, data);
3.  result.putExtra("address", "JD Street");
4.  setResult(RESULT_OK, result);
5.  finish();
```

1. 在子Activity调用finish()函数关闭前，调用setResult()函数设定需要返回给父Activity的数据。
2. setResult()函数有两个参数，“结果码”和“返回值”。结果码表明了子Activity的返回状态，通常为Activity.RESULT_OK（正常返回数据）或者Activity.RESULT_CANCELED（取消返回数据），也可以是自定义的结果码，结果码均为整数类型。
3. 返回值封装在Intent中，也就是说子Activity通过Intent将需要返回的数据传递给父Activity。数据主要以Uri形式返回给父Activity，此外还可以附加一些额外信息，这些额外信息用Extra的集合表示。

6.1 Intent简介

• 6.1.2 获取Activity返回值

- 在父Activity中获取返回值
  - 当子Activity关闭后，父Activity会调用onActivityResult()函数，用了获取子Activity的返回值。
  - onActivityResult()函数的语法如下：

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

    - 第1个参数requestCode是“请求码”，用来判断是哪一个子Activity的返回值
    - 第2个参数resultCode用于表示子Activity的数据返回状态
    - 第3个参数Data是子Activity的返回数据，返回数据类型是Intent
      - Uri uriData = data.getData();

6.1 Intent简介

- **6.1.2 获取Activity返回值**
  - 在父Activity中获取返回值
    - 以下代码说明如何在父Activity中处理子Activity的返回值:

```
1.  private static final int SUBACTIVITY1 = 1;
2.  private static final int SUBACTIVITY2 = 2;
3.  @Override
4.  public void onActivityResult(int requestCode, int resultCode, Intent data){
5.    Super.onActivityResult(requestCode, resultCode, data);
6.    switch(requestCode){
7.    case SUBACTIVITY1:
8.    if (resultCode == Activity.RESULT_OK){
9.    Uri uriData = data.getData();
10.   }else if (resultCode == Activity.RESULT_CANCEL){
11.   }
12.   break;
13.   case SUBACTIVITY2:
14.   if (resultCode == Activity.RESULT_OK){
15.   Uri uriData = data.getData();
16.   }
17.   break;
18.   }
19. }
```

6.2 Intent过滤器

- **Intent解析**
  - Android系统一定存在一种**匹配机制**，使Android系统能够根据Intent中的数据信息，找到需要启动的组件
  - 这种匹配机制是依靠Android系统中的**Intent过滤器**（Intent Filter）来实现的

6.2 Intent过滤器

- **Intent解析**
  - Intent过滤器
    - 是一种根据Intent中的动作（Action）、类别（Categorie）和数据（Data）等内容，对适合接收该Intent的组件进行匹配和筛选的机制
    - 可以匹配数据类型、路径和协议，还可以确定多个匹配项顺序的优先级（Priority）
  - 注册Intent过滤器的组件
    - Activity
    - Service
    - BroadcastReceiver

6.2 Intent过滤器

- **Intent解析**
  - AndroidManifest.xml文件代码：

```
1.  <activity android:name=".IntentResolutionDemo"
2.  android:label="@string/app_name">
3.  <intent-filter>
4.  <action android:name="android.intent.action.MAIN" />
5.  <category android:name="android.intent.category.LAUNCHER" />
6.  </intent-filter>
7.  </activity>
8.
9.  <activity android:name=".ActivityToStart"
10. android:label="@string/app_name">
11. <intent-filter>
12. <action android:name="android.intent.action.VIEW" />
13. <category android:name="android.intent.category.DEFAULT" />
14. <data android:scheme="schemodemo" android:host="edu.hrbeu" />
15. </intent-filter>
16. </activity>
```

Intent启动Activity代码：

```
1.  Intent intent = new Intent(Intent.ACTION_VIEW,
    Uri.parse("schemodemo://edu.hrbeu/path"));
2.  startActivity(intent);
```

6.2 Intent过滤器

- **Intent解析**
  - 组件注册Intent过滤器，在AndroidManifest.xml文件的各个组件下定义<intent-filter>节点，然后在<intent-filter>节点中声明该组件所支持的动作、执行的环境和数据格式等信息
  - <intent-filter>节点中
    - <action>标签：Intent过滤器的“动作”
    - <category>标签：Intent过滤器的“类别”
    - <data>标签：Intent过滤器的“数据”

6.2 Intent过滤器

<intent-filter>节点支持的标签和属性说明:

标签	属性	说明
<action>	**android:name**	指定组件所能响应的动作，用字符串表示，通常由Java类名和包的完全限定名构成
<category>	**android:category**	指定以何种方式去服务Intent请求的动作
<data>	**Android:host**	指定一个有效的主机名
	android:mimetype	指定组件能处理的数据类型
	android:path	有效的URI路径名
	android:port	主机的有效端口号
	android:scheme	所需要的特定协议

6.2 Intent过滤器

- **Intent解析**
  - **<category>标签用来指定Intent过滤器的服务方式**
  - **Android系统提供的类别：**

值	说明
ALTERNATIVE	**Intent数据默认动作的一个可替换的执行方法**
SELECTED_ALTERNATIVE	**和ALTERNATIVE类似，但替换的执行方法不是指定的，而是被解析出来的**
BROWSABLE	**声明Activity可以由浏览器启动**
DEFAULT	**为Intent过滤器中定义的数据提供默认动作**
HOME	**设备启动后显示的第一个Activity**
LAUNCHER	**在应用程序启动时首先被显示**

6.2 Intent过滤器

- **Intent解析**
  - Intent到Intent过滤器的映射
  - 匹配规则：

    （1） Android系统把所有应用程序包中的Intent过滤器集合在一起，形成一个完整的Intent过滤器列表

    （2） 在Intent与Intent过滤器进行匹配时，Android系统会将列表中所有Intent过滤器的“动作”和“类别”与Intent进行匹配，任何不匹配的Intent过滤器都将被过滤掉。没有指定“动作”的Intent过滤器可以匹配任何的Intent，但是没有指定“类别”的Intent过滤器只能匹配没有“类别”的Intent

    （3） 把Intent数据Uri的每个子部与Intent过滤器的<data>标签中的属性进行匹配，如果<data>标签指定了协议、主机名、路径名或MIME类型，那么这些属性都要与Intent的Uri数据部分进行匹配，任何不匹配的Intent过滤器均被过滤掉

    （4） 如果Intent过滤器的匹配结果多于一个，则可以根据在<intent-filter>标签中定义的优先级标签来对Intent过滤器进行排序，优先级最高的Intent过滤器将被选择

6.3 广播消息

- **广播消息**
  - Intent的另一种用途是发送广播消息
    - 应用程序和Android系统都可以使用Intent发送广播消息
    - 广播消息的内容可以与应用程序密切相关的数据信息，也可以Android的系统信息
      - 网络连接变化
      - 电池电量变化
      - 接收到短信
      - 系统设置变化

6.3 广播消息

- **使用Intent发送广播消息非常简单**
  - 只需创建一个Intent
  - 并调用sendBroadcast()函数把Intent携带的信息广播出去

```
1.  String UNIQUE_STRING = "edu.hrbeu.BroadcastReceiverDemo";
2.  Intent intent = new Intent(UNIQUE_STRING);
3.  intent.putExtra("key1", "value1");
4.  intent.putExtra("key2", "value2");
5.  sendBroadcast(intent);
```

  - 在构造Intent时必须定义一个全局唯一的字符串，用来标识其要执行的动作，通常使用应用程序包的名称
  - 要在Intent传递额外数据，可以用Intent的putExtra()方法

6.3 广播消息

- **接收消息**

（1）在AndroidManifest.xml文件中注册BroadcastReceiver

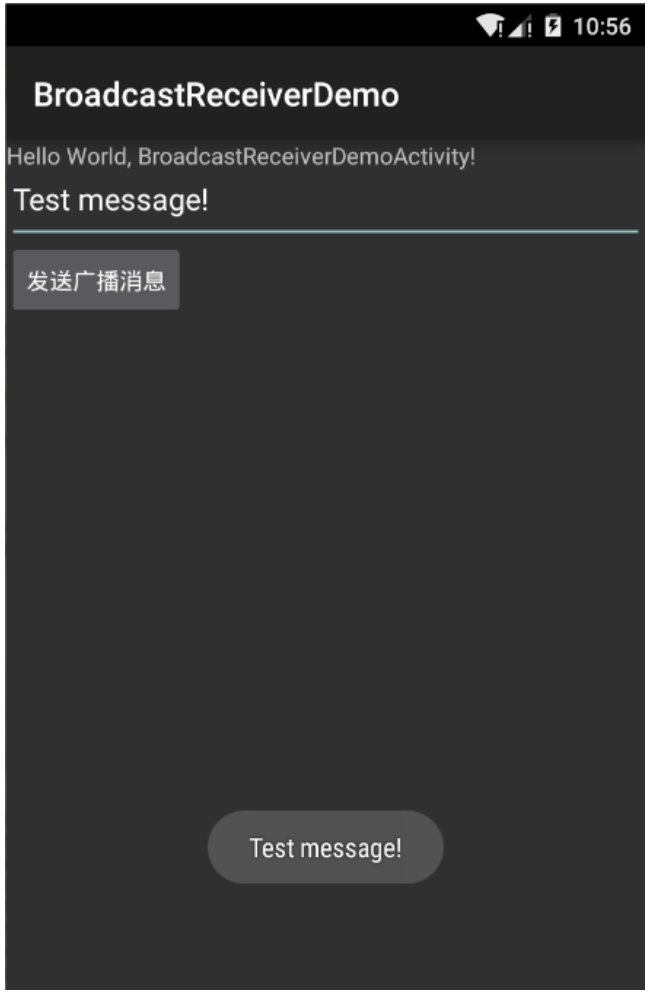
```
1.  <receiver android:name=".MyBroadcastReceiver">
2.      <intent-filter>
3.      <action android:name="edu.hrbeu.BroadcastReceiverDemo" />
4.      </intent-filter>
5.      </receiver>
```

（2）创建BroadcastReceiver需继承BroadcastReceiver类，并重载onReceive()方法。示例代码如下：

```
1.  public class MyBroadcastReceiver extends BroadcastReceiver {
2.  @Override
3.  public void onReceive(Context context, Intent intent) {
4.  //TODO: React to the Intent received.
5.  }
6.  }
```

6.3 广播消息

- **BroadcastReceiverDemo示例**
  - 说明如何在应用程序中注册BroadcastReceiver组件，并指定接收广播消息的类型
  - 在点击“发生广播消息”按钮后，EditText控件中内容将以广播消息的形式发生出去，内部的BroadcastReceiver将接收这个广播消息，并显示在用户界面的下方



6.3 广播消息

- **BroadcastReceiverDemo示例**
  - MyBroadcastReceiver.java文件创建了一个自定义的BroadcastReceiver，其核心代码如下：

```
1.  public class MyBroadcastReceiver extends BroadcastReceiver {
2.  @Override
3.  public void onReceive(Context context, Intent intent) {
4.  String msg = intent.getStringExtra("message");
5.  Toast.makeText(context, msg, Toast.LENGTH_SHORT).show();
6.  }
7.  }
```

  - 代码第1行首先继承了BroadcastReceiver类，并在第3行重载了onReveive()函数。当接收到AndroidManifest.xml文件定义的广播消息后，程序将自动调用onReveive()函数进行消息处理
  - 代码第4行通过调用getStringExtra()函数，从Intent中获取标识为message的字符串数据，并使用Toast()函数将信息显示在界面

习题：

1.简述Intent的定义和用途。

2.简述Intent过滤器的定义和功能。

3.简述Intent解析的匹配规则。

4.编程实现具有“登录”按钮的主界面，点击“登录”按钮后打开一个新的Activity，新打开的Activity上面有输入用户名和密码的控件，在用户关闭这个Activity后，将用户名和密码传递到主界面的Activity中。

THANKS

谢　　谢　　观　　看