第三章 组件通信与广播消息

北京邮电大学 计算机学院
刘伟
w.liu@foxmail.com

本章重点

- 了解使用Intent进行组件通信的原理
- 掌握使用Intent启动Activity的方法
- 掌握获取Activity返回值的方法
- 了解Intent过滤器的原理与匹配机制
- 掌握发送和接收广播消息的方法

3.1 Intent简介

- Intent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
  - Intent也可称为一个在不同组件之间传递的消息，这个消息在到达接收组件后，接收组件会执行相关的动作
  - Intent为Activity、Service和BroadcastReceiver等组件提供交互能力
- Intent的用途
  - Android应用程序可以包含一个或多个Activity，一般需要指定一个程序启动时显示的Activity
    - 启动Activity和Service
    - 在Android系统上发布广播消息
  - 广播消息可以是接收到特定数据或消息，也可以是手机的信号变化或电池的电量过低等信息

3.1.1 启动Activity

- 在Android系统中，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， Android系统根据Intent的动作和数据来决定启动哪一个Activity，也就是说在隐式启动时，Intent中只包含需要执行的动作和所包含的数据，而无需指明具体启动哪一个Activity，选择权有Android系统和最终用户来决定

3.1 Intent简介

3.1.1 启动Activity

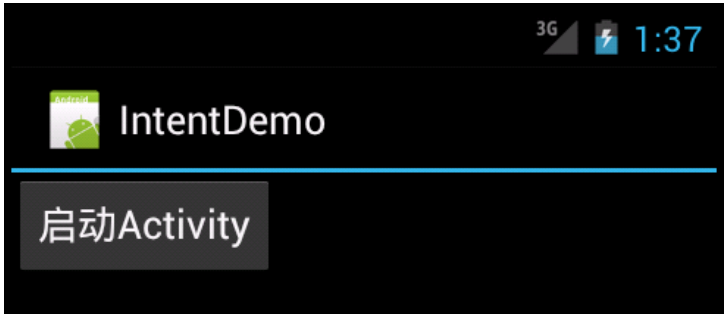
- 显式启动
  - 使用Intent显式启动Activity
  - 创建一个Intent
  - 指定当前的应用程序上下文以及要启动的Activity
  - 把创建好的这个Intent作为参数传递给startActivity()方法

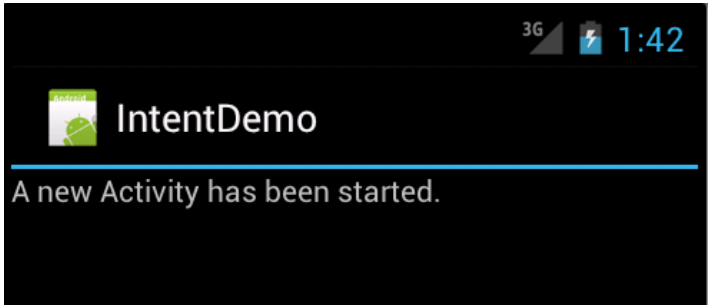
```
1.Intent intent = new Intent(IntentDemo.this, ActivityToStart.class);
2.startActivity(intent);
```

3.1 Intent简介

3.1.1 启动Activity

- 显式启动
  - 下面用IntentDemo示例说明如何使用Intent启动新的Activity。IntentDemo示例包含两个Activity，分别是IntentDemoActivity和NewActivity。
  - 程序默认启动的Activity是IntentDemo，在用户点击“启动Activity”按钮后，程序启动的Activity是NewActivity





3.1 Intent简介

3.1.1 启动Activity

- 显式启动
  - 在IntentDemo示例中使用了两个Activity，因此需要在AndroidManifest.xml文件中注册这两个Activity。注册Activity应使用<activity>标签，嵌套在<application>标签内部。

3.1 Intent简介

- 3.1.1 启动Activity
- 显式启动
  - AndroidManifest.xml文件代码如下

```
1.<?xml version="1.0" encoding="utf-8"?>
•<manifest xmlns:android="http://schemas.android.com/apk/res/android"
•     package="edu.bupt.IntentDemo"
1.    android:versionCode="1"
•     android:versionName="1.0">
1.    <application android:icon="@drawable/icon" android:label="@string/app_name">
2.          <activity android:name=".IntentDemo"
3.                android:label="@string/app_name">
4.                <intent-filter>
•                       <action android:name="android.intent.action.MAIN" />
•                       <category android:name="android.intent.category.LAUNCHER" />
1.                </intent-filter>
•           </activity>
1.          <activity android:name=".NewActivity"
2.                      android:label="@string/app_name">
3.          </activity>
4.    </application>
•     <uses-sdk android:minSdkVersion="14" />
•</manifest>
```

3.1 Intent简介

- 显式启动
  - Android应用程序中，用户使用的每个组件都必须在AndroidManifest.xml文件中的<application>节点内定义。在上面的代码中，<application>节点下共有两个<activity>节点，分别代表应用程序中所使用的两个Activity，IntentDemoActivity和NewActivity。

3.1 Intent简介

3.1.1 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);
•button.setOnClickListener(new OnClickListener(){
•                public void onClick(View view){
•                                Intent intent = new Intent(IntentDemoActivity.this,
NewActivity.class);
1.                               startActivity(intent);
2.               }
3.});
```

3.1 Intent简介

3.1.1 启动Activity

- 显式启动
  - 在点击事件的处理函数中，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。

3.1 Intent简介

3.1.1 启动Activity

- 隐式启动
  - 隐式启动的好处在于不需要指明需要启动哪一个Activity，而由Android系统来决定，这样有利于降低组件之间的耦合度。
  - 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。
  - 因此，这种方式强调了Android组件的可复用性。

3.1 Intent简介

3.1.1 启动Activity

- 隐式启动
  - 如果程序开发人员希望启动一个浏览器，查看指定的网页内容，却不能确定具体应该启动哪一个Activity，此时则可以使用Intent的隐式启动方式，由Android系统在程序运行时决定具体启动哪一个应用程序的Activity来接收这个Intent。
  - 程序开发人员可以将浏览动作和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。

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

3.1 Intent简介

3.1.1 启动Activity

- 隐式启动
  - Intent的动作是Intent.ACTION_VIEW，数据是Web地址，使用Uri.parse(urlString)方法，可以简单的把一个字符串解释成Uri对象。Android系统在匹配Intent时，首先根据动作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.google.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。

3.1 Intent简介

3.1.1 启动Activity

- 隐式启动
  - Intent的语法如下：

```
1.Intent intent = new Intent(Intent.ACTION_VIEW, Uri.parse(urlString));
```

3.1 Intent简介

3.1.1 启动Activity

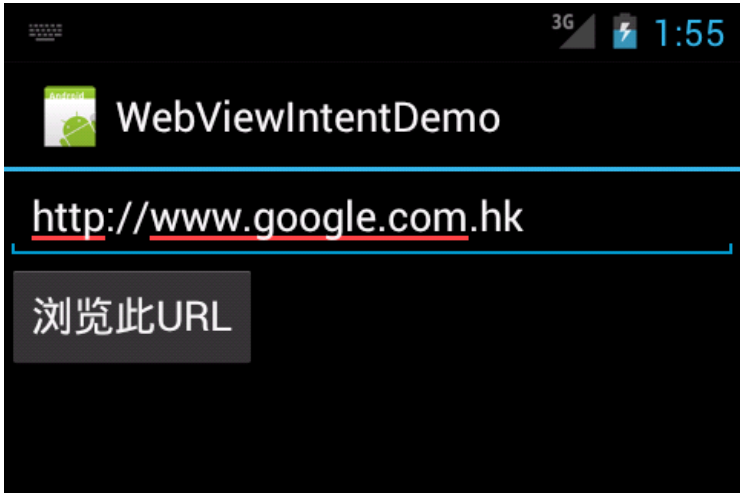
- 隐式启动
  - Intent构造函数的第1个参数是Intent需要执行的动作，Android系统支持的常见动作字符串常量可以参考表。第2个参数是URI，表示需要传递的数据。

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

3.1 Intent简介

3.1.1 启动Activity

- 隐式启动
  - WebViewIntentDemo示例说明了如何隐式启动Activity，用户界面



3.1 Intent简介

3.1.1 启动Activity

- 隐式启动
  - 当用户在文本框中输入Web地址后，通过点击“浏览此URL”按钮，程序根据用户输入的Web地址生成一个Intent，并以隐式启动的方式调用Android内置的Web浏览器，并打开指定的Web页面。本例输入的Web地址http://www.google.com.hk，打开页面后的效果如图

3.1 Intent简介

3.1.2 获取Activity返回值

- 在上一小节IntentDemo示例中，通过startActivity(Intent)方法启动Activity，启动后的两个Activity之间相互独立，没有任何的关联。在很多情况下，后启动的Activity是为了让用户对特定信息进行选择，在后启动的Activity关闭时，这些信息是需要返回给先前启动的Activity。
- 后启动的Activity称为为“子Activity”，先启动的Activity称为“父Activity”。
- 如果需要将子Activity的信息返回给父Activity，则可以使用Sub-Activity的方式去启动子Activity。

3.1 Intent简介

3.1.2 获取Activity返回值

- 获取子Activity的返回值，一般可以分为以下三个步骤：
  - 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；
  - 下面详细介绍每一个步骤的过程和代码实现。

3.1 Intent简介

- **3.1.2 获取Activity返回值**
  - 以Sub-Activity的方式启动子Activity
    - 以Sub-Activity方式启动子Activity，需要调用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

3.1 Intent简介

3.1.2 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;
•Intent intent = new Intent(this, SubActivity1.class);
•startActivityForResult(intent, SUBACTIVITY1);
```

  - 隐式启动子Activity的代码如下

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

3.1 Intent简介

3.1.2 获取Activity返回值

- 设置子Activity的返回值
  - 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。setResult()函数有两个参数，一个是结果码，一个是返回值。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。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。

3.1 Intent简介

3.1.2 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);
•Intent result = new Intent(null, data);
•result.putExtra("address", "JD Street");
•setResult(RESULT_OK, result);
1.finish();
```

3.1 Intent简介

3.1.2 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。
  - 如果需要在父Activity中处理子Activity的返回值，则重载此函数即可。
  - onActivityResult()函数的语法如下：

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

  - 其中第1个参数requestCode是请求码，用来判断第3个参数是哪一个子Activity的返回值；resultCode用于表示子Activity的数据返回状态；Data是子Activity的返回数据，返回数据类型是Intent。根据返回数据的用途不同，Uri数据的协议则不同，也可以使用Extra方法返回一些原始类型的数据。

3.1 Intent简介

3.1.2 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;
•private static final int SUBACTIVITY2 = 2;
•@Override
1.public void onActivityResult(int requestCode, int resultCode, Intent data){
•       Super.onActivityResult(requestCode, resultCode, data);
1.      switch(requestCode){
2.              case SUBACTIVITY1:
3.                      if (resultCode == Activity.RESULT_OK){
4.                              Uri uriData = data.getData();
•                       }else if (resultCode == Activity.RESULT_CANCEL){
•                       }
1.              break;
•               case SUBACTIVITY2:
1.                      if (resultCode == Activity.RESULT_OK){
2.                              Uri uriData = data.getData();
3.                      }
4.              break;
•       }
•}
```

3.1 Intent简介

3.1.2 获取Activity返回值

- 在父Activity中获取返回值
  - 代码的第1行和第2行是两个子Activity的请求码，在第7行对请求码进行匹配。
  - 代码第9行和第11行对结果码进行判断，如果返回的结果码是Activity.RESULT_OK，则在代码的第10行使用getData()函数获取Intent中的Uri数据；
  - 如果返回的结果码是Activity.RESULT_CANCELED，则放弃所有操作。

3.1 Intent简介

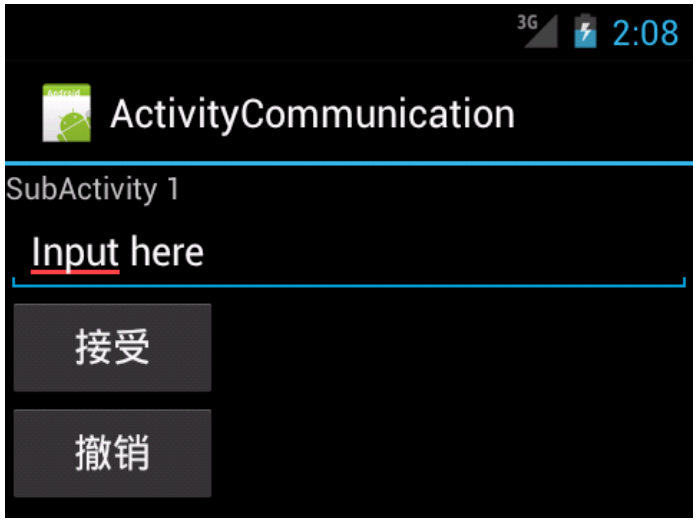
3.1.2 获取Activity返回值

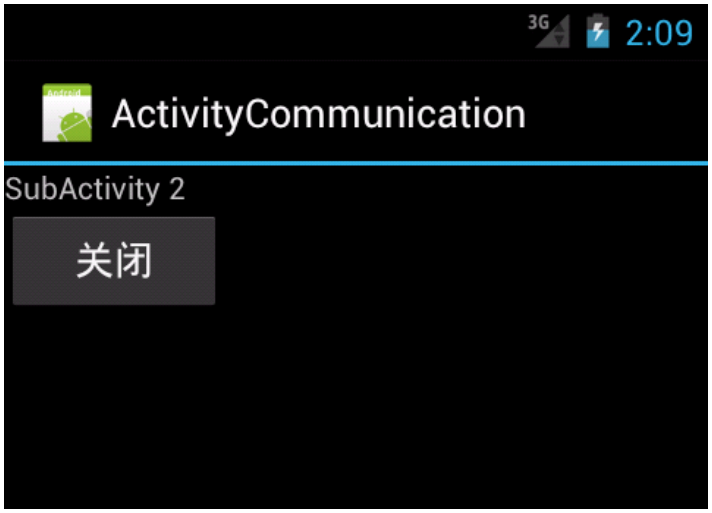
- 在父Activity中获取返回值
  - ActivityCommunication示例说明了如何以Sub-Activity方式启动子Activity，以及如何使用Intent进行组件间通信。

3.1 Intent简介

3.1.2 获取Activity返回值

- 在父Activity中获取返回值
  - 当用户点击“启动Activity1”和“启动Activity2”按钮时，程序将分别启动子SubActivity1和SubActivity2。





3.1 Intent简介

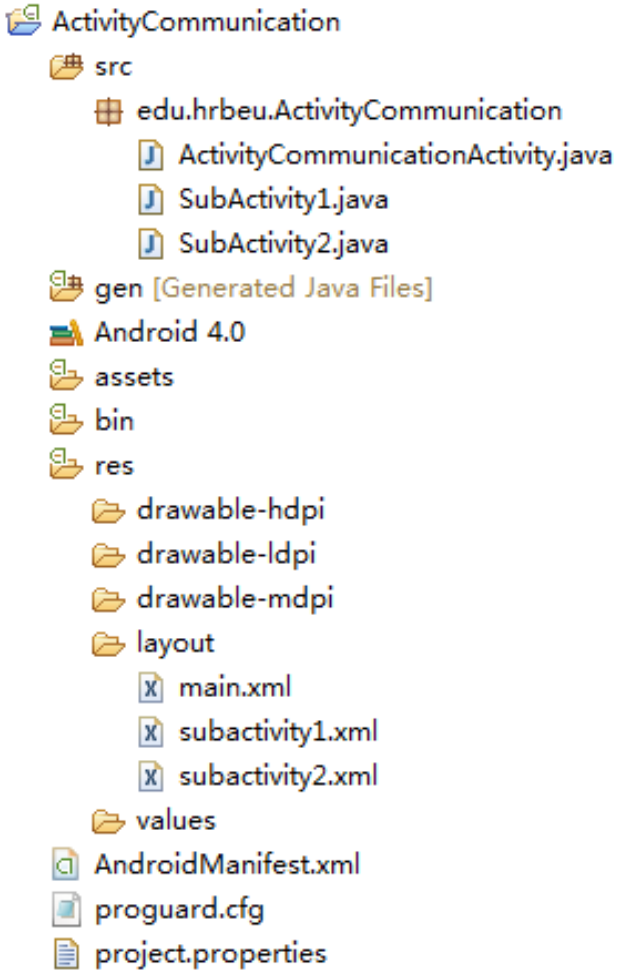
3.1.2 获取Activity返回值

- 在父Activity中获取返回值
  - SubActivity1提供了一个输入框，以及“接受”和“撤销”两个按钮。如果在输入框中输入信息后点击“接受”按钮，程序会把输入框中的信息传递给其父Activity，并在父Activity的界面上显示。
  - 如果用户点击“撤销”按钮，则程序不会向父Activity传递任何信息。
  - SubActivity2主要是为了说明如何在父Activity中处理多个子Activity，因此仅提供了用于关闭SubActivity2的“关闭”按钮。

3.1 Intent简介

3.1.2 获取Activity返回值

- 在父Activity中获取返回值
  - ActivityCommunication示例的文件结构
    - 父Activity的代码在ActivityCommunication.java文件中，界面布局在main.xml中；两个子Activity的代码分别在SubActivity1.java和SubActivity2.java文件中，界面布局分别在subactivity1.xml和subactivity2.xml中。



3.1 Intent简介

3.1.2 获取Activity返回值

- 在父Activity中获取返回值
  - ActivityCommunicationActivity.java文件的核心代码如下

```
1   public class ActivityCommunicationActivity extends Activity {
2       private static final int SUBACTIVITY1 = 1;
3       private static final int SUBACTIVITY2 = 2;
4       TextView textView;
5       @Override
6       public void onCreate(Bundle savedInstanceState) {
7           super.onCreate(savedInstanceState);
8           setContentView(R.layout.main);
9           textView = (TextView)findViewById(R.id.textShow);
10          final Button btn1 = (Button)findViewById(R.id.btn1);
11          final Button btn2 = (Button)findViewById(R.id.btn2);
12
13          btn1.setOnClickListener(new OnClickListener(){
14              public void onClick(View view){
15              Intent intent = new Intent(ActivityCommunication.this,
    SubActivity1.class);
16              startActivityForResult(intent, SUBACTIVITY1);
17              }
```

3.1 Intent简介

3.1.2 获取Activity返回值

- 在父Activity中获取返回值

```
18                      });
19
20                      btn2.setOnClickListener(new OnClickListener(){
21                              public void onClick(View view){
22                              Intent intent = new
    Intent(ActivityCommunication.this, SubActivity2.class);
23                              startActivityForResult(intent, SUBACTIVITY2);
24                              }
25                      });
26          }
27
28          @Override
29          protected void onActivityResult(int requestCode, int resultCode, Intent
    data) {
30                      super.onActivityResult(requestCode, resultCode, data);
31
```

3.1 Intent简介

3.1.2 获取Activity返回值

- 在父Activity中获取返回值

```
32                      switch(requestCode){
33                      case SUBACTIVITY1:
34                              if (resultCode == RESULT_OK){
35                                      Uri uriData = data.getData();
36
        textView.setText(uriData.toString());
37                              }
38                              break;
39                      case SUBACTIVITY2:
40                              break;
41                      }
42      }43             }
```

3.1 Intent简介

3.1.2 获取Activity返回值

- 在父Activity中获取返回值
  - 在代码的第2行和第3行分别定义了两个子Activity的请求码。
  - 在代码的第16行和第23行以Sub-Activity的方式分别启动两个子Activity。
  - 代码第29行是子Activity关闭后的返回值处理函数，其中requestCode是子Activity返回的请求码，与第2行和第3行定义的两个请求码相匹配；resultCode是结果码，
  - 在代码第32行对结果码进行判断，如果等于RESULT_OK，在第35行代码获取子Activity返回值中的数据；data是返回值，子Activity需要返回的数据就保存在data中。

3.1 Intent简介

3.1.2 获取Activity返回值

- 在父Activity中获取返回值
  - SubActivity1.java的核心代码如下：

```
1.public class SubActivity1 extends Activity {
•                   @Override
•                   public void onCreate(Bundle savedInstanceState) {
1.                              super.onCreate(savedInstanceState);
•                               setContentView(R.layout.subactivity1);
•                               final EditText editText = (EditText)findViewById(R.id.edit);
1.                              Button btnOK = (Button)findViewById(R.id.btn_ok);
2.                              Button btnCancel = (Button)findViewById(R.id.btn_cancel);
3.
•                               btnOK.setOnClickListener(new OnClickListener(){
•                                               public void onClick(View view){
1.                                                      String uriString = editText.getText().toString();
•                                                       Uri data = Uri.parse(uriString);
•                                                       Intent result = new Intent(null, data);
1.                                                      setResult(RESULT_OK, result);
2.                                                      finish();
3.                                              }
•                               });
•
1.                              btnCancel.setOnClickListener(new OnClickListener(){
•                                               public void onClick(View view){
•                                                       setResult(RESULT_CANCELED, null);
1.                                                      finish();
2.                                              }
3.                              });
•                   }
•}
```

3.1 Intent简介

3.1.2 获取Activity返回值

- 在父Activity中获取返回值
  - 代码第13行将EditText控件的内容作为数据保存在Uri中，并在第14行代码中构造Intent。
  - 在第15行代码中，RESUIT_OK作为结果码，通过调用setResult()函数，将result设定为返回值。
  - 最后在代码第16行调用finish()函数关闭当前的子Activity

3.1 Intent简介

3.1.2 获取Activity返回值

- 在父Activity中获取返回值
  - SubActivity2.java的核心代码：

```
1.public class SubActivity2 extends Activity {
•       @Override
•       public void onCreate(Bundle savedInstanceState) {
1.          super.onCreate(savedInstanceState);
•           setContentView(R.layout.subactivity2);
1.
2.          Button btnReturn = (Button)findViewById(R.id.btn_return);
3.          btnReturn.setOnClickListener(new OnClickListener(){
4.              public void onClick(View view){
•                   setResult(RESULT_CANCELED, null);
•                   finish();
1.              }
•           });
1.      }
2.}
```

3.1 Intent简介

3.1.2 获取Activity返回值

- 在父Activity中获取返回值
  - 在SubActivity2的代码中，第10行的setResult()函数仅设置了结果码，第2个参数为null，表示没有数据需要传递给父Activity。

3.2 Intent过滤器

- **Intent解析**
  - 隐式启动Activity时，并没有在Intent中指明Activity所在的类，因此，Android系统一定存在某种匹配机制，使Android系统能够根据Intent中的数据信息，找到需要启动的Activity。这种匹配机制是依靠Android系统中的Intent过滤器（Intent Filter）来实现的。

3.2 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。Intent过滤器可以匹配数据类型、路径和协议，还可以确定多个匹配项顺序的优先级（Priority）。应用程序的Activity、Service和BroadcastReceiver组件都可以注册Intent过滤器。这样，这些组件在特定的数据格式上则可以产生相应的动作。

3.2 Intent过滤器

- **Intent解析**
  - 为了使组件能够注册Intent过滤器，通常在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。当然，也可以在程序代码中动态地为组件设置Intent过滤器。<intent-filter>节点支持<action>标签、<category>标签和<data>标签，分别用来定义Intent过滤器的“动作”、“类别”和“数据”。<intent-filter>节点支持的标签和属性说明参考表

3.2 Intent过滤器

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

3.2 Intent过滤器

- **Intent解析**
  - <category>标签用来指定Intent过滤器的服务方式，每个Intent过滤器可以定义多个<category>标签，程序开发人员可以使用自定义的类别，或使用Android系统提供的类别。Android系统提供的类别可以参考表

3.2 Intent过滤器

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

3.2 Intent过滤器

- **Intent解析**
  - 这种Intent到Intent过滤器的映射过程称为“Intent解析”。Intent解析可以在所有的组件中，找到一个可以与请求的Intent达成最佳匹配的Intent过滤器。Android系统中Intent解析的匹配规则如下：

3.2 Intent过滤器

- **Intent解析**
  - （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。
  - IntentResolutionDemo示例说明了如何在AndroidManifest.xml文件中注册Intent过滤器，以及如何设置<intent-filter>节点属性来捕获指定的Intent。

3.2 Intent过滤器

- **Intent解析**
  - AndroidManifest.xml的完整代码如下

```
1.<?xml version="1.0" encoding="utf-8"?>
•<manifest xmlns:android="http://schemas.android.com/apk/res/android"
•                    package="IntentResolutionDemo"
1.                   android:versionCode="1"
•                    android:versionName="1.0">
•                    <application android:icon="@drawable/icon" android:label="@string/app_name">
•                                        <activity android:name=".IntentResolutionDemo"
1.                                                                  android:label="@string/app_name">
2.                                                  <intent-filter>
•                                                                   <action android:name="android.intent.action.MAIN" />
•                                                                   <category android:name="android.intent.category.LAUNCHER"
/>
1.                                                  </intent-filter>
•                                        </activity>
•                                        <activity android:name=".ActivityToStart"
•                                                                   android:label="@string/app_name">
1.                                                  <intent-filter>
2.                                                                  <action android:name="android.intent.action.VIEW" />
•                                                                   <category android:name="android.intent.category.DEFAULT" />
•                                                                   <data android:scheme="schemodemo" android:host="edu.bupt"
/>
1.                                                  </intent-filter>
•                                        </activity>
•                    </application>
•                    <uses-sdk android:minSdkVersion="14" />
1.</manifest>
```

3.2 Intent过滤器

Intent解析

- 在代码的第7行和第14行分别定义了两个Activity。第9行到第12行是第1个Activity的Intent过滤器，动作是android.intent.action.MAIN，类别是android.intent.category.LAUNCHER，由此可知，这个Activity是应用程序启动后显示的缺省用户界面。
- 第16行到第20行是第2个Activity的Intent过滤器，过滤器的动作是android.intent.action.VIEW，表示根据Uri协议，以浏览的方式启动相应的Activity；类别是android.intent.category.DEFAULT，表示数据的默认动作；数据的协议部分是android:scheme="schemodemo"，数据的主机名称部分是android:host="edu.bupt"。

3.2 Intent过滤器

- **Intent解析**
  - 在IntentResolutionDemo.java文件中，定义了一个Intent用来启动另一个Activity，这个Intent与Activity设置的Intent过滤器是完全匹配的。

    IntentResolutionDemo.java文件中Intent实例化和启动Activity的代码如下

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

3.2 Intent过滤器

- **Intent解析**
  - 代码第1行所定义的Intent，动作为Intent.ACTION_VIEW，与Intent过滤器的动作android.intent.action.VIEW匹配；
  - Uri是“schemodemo://edu.bupt/path”，其中的协议部分为“schemodemo”，主机名部分为“edu.bupt”，也与Intent过滤器定义的数据要求完全匹配。
  - 因此，代码第1行定义的Intent，在Android系统与Intent过滤器列表进行匹配时，会与AndroidManifest.xml文件中ActivityToStart定义的Intent过滤器完全匹配。

3.2 Intent过滤器

- AndroidManifest.xml文件中每个组件的<intent-filter>都被解析成一个Intent过滤器对象。当应用程序安装到Android系统时，所有的组件和Intent过滤器都会注册到Android系统中。
- 这样，Android系统便可以将任何一个Intent请求通过Intent过滤器映射到相应的组件上。

3.3 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，例如网络连接变化、电池电量变化、接收到短信或系统设置变化等。
- 如果应用程序注册了BroadcastReceiver，则可以接收到指定的广播消息。
- 使用Intent发送广播消息非常简单，只需创建一个Intent，并调用sendBroadcast()函数就可把Intent携带的信息广播出去。
- 但需要注意的是，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。
- 如果要在Intent传递额外数据，可以用Intent的putExtra()方法。下面的代码构造用于广播消息的Intent，并添加了额外的数据，然后调用sendBroadcast()发送广播消息：

3.3 广播消息

- sendBroadcast()代码

```
1.String UNIQUE_STRING = "edu.bupt.BroadcastReceiverDemo";
•Intent intent = new Intent(UNIQUE_STRING);
•intent.putExtra("key1", "value1");
•intent.putExtra("key2", "value2");
1.sendBroadcast(intent);
```

- BroadcastReceiver用于监听广播消息，可以在AndroidManifest.xml文件或在代码中注册一个BroadcastReceiver，并使用Intent过滤器指定要处理的广播消息。

3.3 广播消息

onReceive()方法

- 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 {
•               @Override
•               public void onReceive(Context context, Intent intent) {
•                       //TODO: React to the Intent received.
1.              }
2.}
```

- 当Android系统接收到与注册BroadcastReceiver匹配的广播消息时，Android系统会自动调用这个BroadcastReceiver接收广播消息。在BroadcastReceiver接收到与之匹配的广播消息后，onReceive()方法会被调用，但onReceive()方法必须要在5秒钟执行完毕，否则Android系统会认为该组件失去响应，并提示用户强行关闭该组件。

3.3 广播消息

- **BroadcastReceiverDemo示例**
  - BroadcastReceiverDemo示例说明了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。

    BroadcastReceiverDemo示例的界面如图所示，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，示例内部的BroadcastReceiver将接收这个广播消息，并显示在用户界面的下方。

3.3 广播消息

- BroadcastReceiverDemo示例

  用户界面

3.3 广播消息

BroadcastReceiverDemo示例

- BroadcastReceiverDemo.java文件中包含发送广播消息的代码，其关键代码如下

```
1.<?xml version="1.0" encoding="utf-8"?>
•<manifest xmlns:android="http://schemas.android.com/apk/res/android"
•                package="edu.bupt.BroadcastReceiverDemo"
1.               android:versionCode="1"
•                android:versionName="1.0">
•                <application android:icon="@drawable/icon" android:label="@string/app_name">
•                        <activity android:name=".BroadcastReceiverDemo"
1.                                                      android:label="@string/app_name">
2.                                              <intent-filter>
•                                                       <action android:name="android.intent.action.MAIN" />
•                                                       <category
android:name="android.intent.category.LAUNCHER" />
1.                                              </intent-filter>
•                        </activity>
•                        <receiver android:name=".MyBroadcastReceiver">
•                                               <intent-filter>
1.                                                      <action android:name="edu.bupt.BroadcastReceiverDemo"
/>
2.                                              </intent-filter>
•                        </receiver>
•                </application>
1.               <uses-sdk android:minSdkVersion="14" />
•</manifest>
```

3.3 广播消息

- **BroadcastReceiverDemo示例**
  - 在代码的第14行中创建了一个<receiver>节点，在第15行中声明了Intent过滤器的动作为“edu.bupt.BroadcastReceiverDemo”，
  - 这与BroadcastReceiverDemo.java文件中Intent的动作相一致，表明这个BroadcastReceiver可以接收动作为“edu.bupt.BroadcastReceiverDemo”的广播消息

3.3 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 {
•               @Override
•               public void onReceive(Context context, Intent intent) {
•                       String msg = intent.getStringExtra("message");
1.                      Toast.makeText(context, msg, Toast.LENGTH_SHORT).show();
2.              }
3.}
```

  - 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：

- 简述Intent的定义和用途。
- 简述Intent过滤器的定义和功能。
- 简述Intent解析的匹配规则。
- 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。