



大话企业级 Android 开发 · 第十部分

本教程说明及版权声明

- 《大话企业级 Android 开发》是国士工作室为了方便中国 Android 开发者，推动 Android 企业级应用开发，特投入大量心血撰写的书籍，并在网络上免费发布，希望为移动互联网和智能手机时代贡献绵薄之力！所有相关文档版权均属国士工作室所有。
- 本教程是由国士工作室参考官方文档，综合市面相关书籍，经过充分的吸收消化，结合开发实践的一部原创作品，为了本教程及早与广大读者同仁见面、分享，特采用定稿一部分就发布一部分的连载方式发布。读者可以在本博客获取最新内容。
- 未经国士工作室授权，禁止将此文档及其衍生作品以标准（纸质）书籍形式发行。
- 本文档受有关法律的版权保护，对本文档内容的任何未经同意的复制和抄袭行为，将导致相应的法律责任。未经国士工作室同意，任何团体及个人不能用此教程牟利，违者必究。但是：在不收取其他人费用的前提下，您可以自由传播此文档，但必须保证版权信息、文档及其自带标示的完整性。
- 如果对该文档有任何疑问或者建议，请进入官方博客 <http://www.cnblogs.com/guoshiandroid/> 留言或者直接与国士工作室联系（后附联系方式），我们会慎重参考您的建议并根据需要对本文档进行修改，以造福更多开发者！
- 《大话企业级 Android 开发》的最新及完整内容会在国士工作室官方博客定期更新，请访问国士工作室博客 <http://www.cnblogs.com/guoshiandroid/> 获取更多更新内容。



关于国土工作室

我们(国土工作室)是一支专注于 Android 平台企业级应用开发的技术团队,对娱乐多媒体应用有着深刻的理解及研发能力,致力服务于企业用户。为音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络应用向移动互联网发展提供解决方案和技术支持,为企业提供 Android 培训服务等多种业务。

我们尤其擅长于提供从 Android 客户端到服务端的一站式解决方案和技术支持,服务端可以采用 Java EE,也可以采用轻量级流行的 LAMP 技术体系。目前,研发出了比 KU6、优酷更加强大和完善的 Android 视频网站娱乐多媒体客户端软件,并在持续升级中。

目前,我们正在务实而卓有成效的与音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络服务商合作,发展迅速,渴望有志之士的加入,和我们一起为成为世界最好的 Android 软件开发和咨询、培训公司而奋斗,为移动互联网和智能手机时代贡献力量!

联系我们

电话:15711060468

Email:guoshiandroid@gmail.com

博客: <http://www.cnblogs.com/guoshiandroid/>



第四篇

“机器人”的核心组件

一个强大的“机器人”，必定要有良好的组件组装在一起，才能发挥它的作用。那么我们的 Android，都有哪些核心的组件呢？

经过小安近段时间的学习，已经掌握了 Android 的基础知识，同时也了解了各种 UI 的使用，现在已经能写一些简单的布局。可是一个应用仅仅有漂亮的布局是远远不够的，就像一个机器人仅仅有一个酷炫的外表，却没有强大的马力，就称之为强大，也是远远不行的。怎样才能丰富程序的内在呢？

需要了解 Android 的核心组件，那么就不得不从 Activity 说起。在这里，我们将其定义为“机器人”的管理员。刚接触的开发人员可能不是很明白，那么就继续往下看。

1 “机器人”的管理员——Activity

Activity 是 Android 最基本的组件之一，它就像是一个管理员。我们需要在屏幕上显示什么（通常情况下，一个 Activity 占据一个屏幕），用户将要在在这个屏幕上做什么，怎样来处理用户做出的不同操作都需要由该 Activity 来管理和调度。

下面列出 Activity 经常用到的事件：

<code>onKeyDown(int keyCode, KeyEvent event)</code>	按键按下事件
<code>onTouchEvent(MotionEvent event)</code>	点击屏幕事件
<code>onKeyUp(int keyCode, KeyEvent event)</code>	按键松开事件
<code>onTrackballEvent(MotionEvent event)</code>	轨迹球事件

那么，我们来做一个示例，了解一下 Android 的管理员 Activity 如何处理用户事件。

1. 创建项目

新建一个名为“Activity_Basic”的项目，基本信息如下：



Project name: Activity_Basic

Contents

- ☒ Create new project in workspace
- ☐ Create project from existing source
- ☒ Use default location

Location: E:/Android/MOTODEV/Activity_Basic Browse...

☐ Create project from existing sample

Samples: ApiDemos

Build Target

Target Name	Vendor	Platform	API ...
<input type="checkbox"/> Android 2.0	Android Open Source Project	2.0	5
<input type="checkbox"/> Android 2.0.1	Android Open Source Project	2.0.1	6
<input checked="" type="checkbox"/> Android 2.1	Android Open Source Project	2.1	7

Standard Android platform 2.1

Properties

Application name: 处理事件

Package name: com.sharpandroid.Activity_Basic

☒ Create Activity: EventActivity

Min SDK Version: 7

2. 编写 EventActivity.java

重写我们需要处理的事件，之后使用 Toast 显示给用户。

EventActivity.java:

```
public class EventActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
    @Override
    public boolean onKeyDown(int keyCode, KeyEvent event) {
        showInfo("按键, 按下");
        return super.onKeyDown(keyCode, event);
    }
    @Override
    public boolean onKeyUp(int keyCode, KeyEvent event) {
        showInfo("按键, 弹起");
        return super.onKeyUp(keyCode, event);
    }
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        float x = event.getX();
```



```
float y = event.getY();
showInfo("你点击的坐标为: (" + x + " : " + y + ")");
return super.onTouchEvent(event);
}

public void showInfo(String info) {
    Toast.makeText(EventActivity.this, info, Toast.LENGTH_LONG).show();
}
}
```

3. 执行程序

下面我们来看一下当我们触发相应的事件后，EventActivity 做出的反应。
当按下按键时：



当松开按键时：



当点击屏幕时：



知道了 Activity 可以对用户的操作进行处理,前面说到,一个 Activity 就是一个屏幕,它既是用户操作的屏幕,也是 Android 显示内容的屏幕。那么必须有另一个功能:显示 View。当 Activity 类被创建时,开发人员就可以通过 `setContentView()` 接口把 UI 加载到 Activity 创建的屏幕上,当然 Activity 不仅仅是可以全屏显示,也可以用其他方式实现:作为漂浮窗口(通过 `windowIsFloating` 的主题集合),或者嵌入到其它的 Activity(使用 `ActivityGroup`)。大部分的 Activity 子类都需要实现 `onCreate()` 接口。

`onCreate()` 接口是初始化 Activity 的地方,在这儿通常可以调用 `setContentView()` 设置在资源文件中定义的 UI,使用 `findViewById()` 可以获得 UI 中定义的控件。这些在前一篇都有介绍。

小安：难怪说 Activity 是 Android 的管理员呢，原来是这个样子啊。它不仅管理着 Android 应用中 View 的显示，还要处理 Android 中用户触发的各种事件。这个管理员可真够忙的啊。

大致：呵呵，可不是吗？当个管家可不是那么容易的。而且，它管的可不单单就那两样呢。你只是知道了 Activity 的一点皮毛而已，下面我就继续给你讲讲 Activity 的重要的知识和一些应用，这些非常重要，你可要仔细理解，在以后的应用开发中要合理的运行它的功能。

1.1 Activity 生命周期



初学者听到生命周期这个名词，可能会觉得这是一个比较概念性的东西，比较泛泛。就像是学编程的时候高手会告诉你面向对象多么重要，你不可能马上意识到。MVC 设计模式有多重要，你甚至可能觉得这是一个繁琐的设计模式。同样，初学者来理解 Activity 的生命周期，可能不会马上产生多大的感触，甚至会不解为什么要有生命周期这种机制。但是随着慢慢深入 Andorid 的应用开发，你会发现它的重要性。

Activity 的三种状态

Activity 有三种状态，分别是**运行状态**、**暂停状态**以及**停止状态**。

运行状态：

当 Activity 在屏幕的最前端（位于当前堆栈的顶部），它是可见的、有焦点的。可以用来进行处理用户的操作（点击、双击、长按等），那么就叫做激活或运行状态。值得注意的是，当 Activity 处于运行状态的时候，Android 会尽可能的保持它的运行，即使出现内存不足等情况，Android 也会先杀死堆栈底部的 Activity，来确保运行状态的 Activity 正常运行。

暂停状态：

在某些情况下，Activity 对用户来说，仍然是可见的，但不再拥有焦点，即用户对它的操作是没用实际意义的。在这个时候，它就是属于暂停状态。例如：当最前端的 Activity 是个透明或者没有全屏，那么下层仍然可见的 Activity 就是暂停状态。暂停的 Activity 仍然是激活的（它保留着所有的状态和成员信息并保持与 Activity 管理器的连接），但当内存不足时，可能会被杀死。

注意：也不是所有的 Activity 失去焦点就会进入暂停状态。这点在示例部分会详细说明。

停止状态：

当 Activity 完全不可见时，它就处于停止状态。它仍然保留着当前状态和成员信息。然而这些对用户来说，都是不可见的；同暂停状态一样，当系统其他地方需要内存时，它也有被杀死的可能。

生命周期事件：

Activity 状态的变化是人为操作的，而这些状态的改变，也会触发一些事件。我们且叫它生命周期事件。一共有 7 个。

`void onCreate(Bundle savedInstanceState)`

`void onStart()`

`void onRestart()`

`void onResume()`

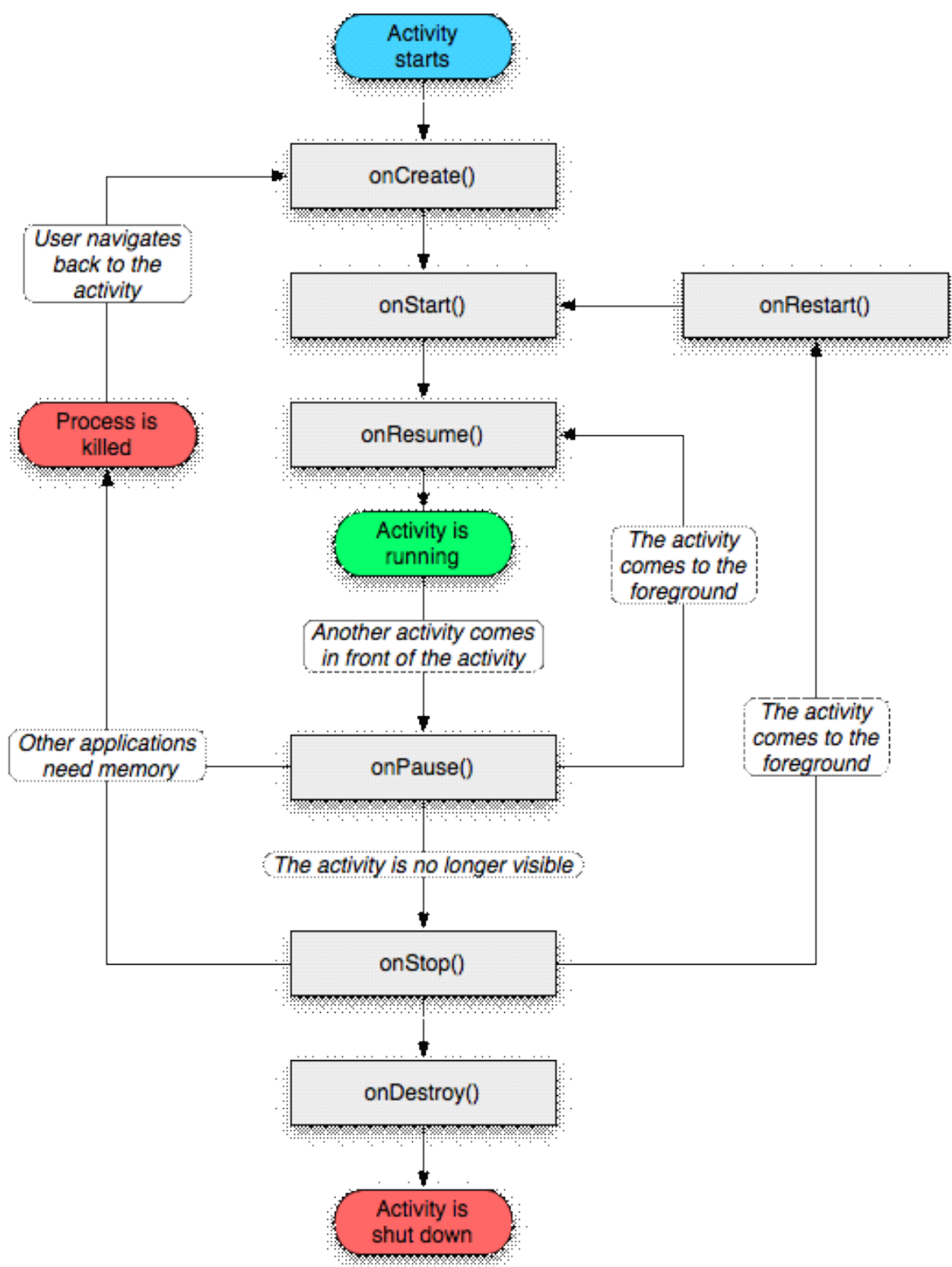
`void onPause()`

`void onStop()`

`void onDestroy()`

这些方法的作用从字面意思上都不难理解，可是这些事件都是在什么时候触发呢？

先来看看 Google 提供的官方文档关于生命周期模型的一个图示



当打开一个 Activity，如果该 Activity 实例不存在于 Activity 管理器中，就会触发 onCreate 事件。注意，Activity 的实例不是我们自己创建的，是 Android 系统自己创建的。接下来是 onStart 事件，然后是 onResume 事件，此时 Activity 就处于了运行状态。

接下来我们通过一个示例讲解一下 Activity 整个生命周期。

创建项目

创建一个名为 xx 的项目



Project name: ActivityLife

Contents

☒ Create new project

☐ Create new project using sample

☐ Create project from existing source

☒ Use default location

Location: E:\Android\MOTODEV

Browse...

Target

SDK Target	Vendor	API Level	SDK
<input type="checkbox"/> Android 2.0	Android Open Source Project	5	5
<input type="checkbox"/> Android 2.0.1	Android Open Source Project	6	6
<input checked="" type="checkbox"/> Android 2.1	Android Open Source Project	7	7

Application

Application name: User Application

Package name: com.sharpandroid.activitylife

Activity name: MainActivity

Min SDK version: 7

编写 MainActivity.java

首先要重写 7 个相应被触发的方法，以日志的形式输出相应的事件信息。然后添加两个 Button，一个用来启动新的 Activity，另一个是用来退出当前 Activity。

```
public class MainActivity extends Activity {  
    private static final String TAG = "MainActivity";  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
        Log.i(TAG, "onCreate()");  
        Button button =(Button) this.findViewById(R.id.button);
```



```
Button button1 =(Button) this.findViewById(R.id.button1);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        Intent intent = new Intent(MainActivity.this,
OtherActivity.class);
        startActivity(intent);
    }
});
button1.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        finish();
    }
});
}
@Override
protected void onDestroy() {
    Log.i(TAG, "onDestroy()");
    super.onDestroy();
}
@Override
protected void onPause() {
    Log.i(TAG, "onPause()");
    super.onPause();
}
@Override
protected void onRestart() {
    Log.i(TAG, "onRestart()");
    super.onRestart();
}
@Override
protected void onResume() {
    Log.i(TAG, "onResume()");
    super.onResume();
}
@Override
protected void onStart() {
    Log.i(TAG, "onStart()");
    super.onStart();
}
@Override
```



```
protected void onStop() {  
    Log.i(TAG, "onStop()");  
    super.onStop();  
}  
}
```

再新建一个 OtherActivity，同样重写需要触发的生命周期事件，与 MainActivity.java 相类似，并在 AndroidManifest.xml 中写入注册信息：

```
<activity android:name=".OtherActivity"  
    android:label="@string/other"  
    android:theme="@android:style/Theme.Dialog"/>
```

android:theme：设置 Activity 的主题，这里主要是为了让其达到显示暂停状态而设置。

从 Activity 开始运行，到点击【退出】按钮调用 finish() 方法结束 Activity 的整个事件调用过程。值得注意的是，在调用 finish() 之后系统会先调用 onPause()，再调用 onStop() 之后调用 onDestroy()。

在启动应用之后，当我们点击【打开新 Activity】，观看一下触发的相应事件：

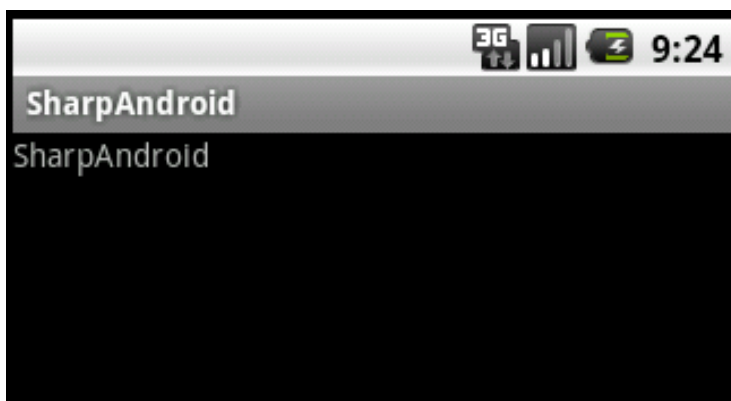
从 Logcat 控制台上看，新的 Activity 已经启动，而之前的 Activity 还处于可见状态，只是我们再去点击按钮已经没用，也就是失焦。此时，MainActivity 处于暂停状态，OtherActivity 处于运行状态。

03-30 09:13:01.765	I	539	MainActivity	onCreate()
03-30 09:13:01.784	I	539	MainActivity	onStart()
03-30 09:13:01.805	I	539	MainActivity	onResume()
03-30 09:13:05.735	I	539	MainActivity	onPause()
03-30 09:13:05.994	I	539	MainActivity	OtherActivity.onCreate()
03-30 09:13:06.017	I	539	MainActivity	OtherActivity.onStart()
03-30 09:13:06.024	I	539	MainActivity	OtherActivity.onResume()

MainActivity 转到 OtherActivity，先触发自己的 onPause() 事件而后才是 OtherActivity 的生命周期事件。



然后对应用稍作修改，先去除上面讲到的 OtherActivity 的 `android:theme` 属性，使其能全屏显示，再次运行，查看运行结果，再次点击按钮，这时已经看不到 MainActivity, 说明现在 MainActivity 已经处于停止状态，而不是此前的暂停状态。



再来查看控制台打印出的相关信息。比暂停状态多触发一个 `onStop` 事件。



03-30 09:24:17.905	I	593	MainActivity	onCreate()
03-30 09:24:17.935	I	593	MainActivity	onStart()
03-30 09:24:17.946	I	593	MainActivity	onResume()
03-30 09:24:21.015	I	593	MainActivity	onPause()
03-30 09:24:21.265	I	593	MainActivity	OtherActivity.onCreate()
03-30 09:24:21.275	I	593	MainActivity	OtherActivity.onStart()
03-30 09:24:21.285	I	593	MainActivity	OtherActivity.onResume()
03-30 09:24:22.035	I	593	MainActivity	onStop()

这里对生命周期做一个总结：

1. Activity 从创建到进入运行状态所触发的事件：
onCreate() → onStart() → onResume()
2. 当 Activity 从运行状态到停止状态所触发的事件：
onPause() → onStop()
3. 当 Activity 从停止状态到运行状态所触发的事件：
onRestart() → onStart() → onResume()
4. 当 Activity 从运行状态到暂停状态所触发的事件：
onPause()
5. 当 Activity 从暂停状态到运行状态所触发的事件：
onResume()

了解了 Android 的生命周期，我们再来看一个示例：
建立一个 Button。点击后会出现一个对话框，如图：



在这时，MainActivity 还可以看见，但我们对它进行任何操作都没有反应，即失焦。这样子我们且将其认为失焦。为什么这样认为呢？

首先先观看其打印出的生命周期过程：

03-30 10:15:59.735	I	679	MainActivity	onCreate()
03-30 10:15:59.765	I	679	MainActivity	onStart()
03-30 10:15:59.775	I	679	MainActivity	onResume()

你会发现 MainActivity 并没有触发 onPause()，也就是说它并没有进入暂停状态，这到底是怎么回事呢？这里可能会有很多人产生困惑？这样一来不是和官方解释的不一样了吗？是我们编写的错误？还是 Android 的 Bug？

其实什么都不是，只是一个理解的误区，或者说是一个思维定势。这里需要特别注意，我们一直讨论的是 Activity 生命周期，而生命周期事件我们在前面也强调了只有 Activity 与 Activity 之间的转换才会触发。而对话框并不是一个 Activity，它只是 MainActivity 的一部分。所以当点击按钮，并没有引起 Activity 生命周期的变化。

小安：哦，原来是这个样子啊，我说怎么和前面说到的怎么还自相矛盾了呢？

大致：对啊，其实本来就是是个不是问题的问题。文档说的很明确，只是我们自己在妄加揣测而已。在这里，交给你一个更简单的区分方法。运行程序前，先看看 AndroidManifest.xml 文件，到底注册了有哪些 Activity。只有它们之间



的转换才会引起生命周期事件的触发。

小安：恩，这可是个好办法。没想到生命周期看似简单，还挺不好理解的。可是，这么多的生命周期事件，到底有什么用呢？

大致：比如有些你想在应用创建初期就要处理好的，那就要放在相应的事件。或者在关闭程序你要做一些相关的数据保存，那么就在 `onStop()` 处理就好。在相应的时期做相应的事件，便于管理，也让程序的执行有条不紊。更重要的是，这样针对 Android，可以进行更好的内存管理，合理的使用和释放内存，使手机运行在最佳状态。

小安：原来是这样子啊，那么 Activity 还有没有别的用处呢？

大致：当然，一个全方面的管理员，当然要照顾到 Android 系统的方方面面。让我继续给你介绍 Activity 吧。

1.2 为应用添加新的 Activity

1.2.1 一个 Activity 实现页面转换

在很多的项目中大多都需要多个 Activity，但是也有的项目只用到一个 Activity，如果应用只有一个 Activity，它的做法无非就是通过调用 `setContentView()` 方法载入不同的 Layout 实现页面的转换。

创建项目

新建一个 Android 项目 oneActivity，如图 5-1；



Project name:

Contents

☒ Create new project in workspace

☐ Create project from existing source

☒ Use default location

Location:

☐ Create project from existing sample

Samples:

Build Target

Target Name	Vendor	Platform	AP...
<input type="checkbox"/> Android 2.0	Android Open Source Project	2.0	5
<input checked="" type="checkbox"/> Android 2.1	Android Open Source Project	2.1	7

Standard Android platform 2.1

Properties

Application name:

Package name:

☒ Create Activity:

Min SDK Version:

编写 main.xml

在 main.xml 中加入一个按钮，代码如下；

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="这是第一页"
    />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```



```
        android:text="下一页"
    ></Button>
</LinearLayout>
```

编写 two.xml

然后新建一个 Layout 文件，two.xml，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="这是第二页"
        />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="上一页"
    ></Button>
</LinearLayout>
```

编写 OneActivity.java

在 OneActivity 中，一开始加载的是 main.xml，我们单击按钮【下一页】，显示第二个界面，然后单击【上一页】，返回原来的页面，实现不同页面之间的转换效果，代码如下：

```
public class OneActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button nextButton=(Button)findViewById(R.id.next);
        nextButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
```



```
        nextLayout(); //显示下一个页面
    }
});

}

public void nextLayout() {
    setContentView(R.layout.two);
    Button upButton=(Button)findViewById(R.id.up);
    upButton.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) { //显示上一个页面
            upLayout();
        }

    });
}

public void upLayout() {
    setContentView(R.layout.main);
    Button nextButton=(Button)findViewById(R.id.next);
    nextButton.setOnClickListener(new View.OnClickListener() {

        @Override
        public void onClick(View v) {
            nextLayout(); //显示下一个页面
        }

    });
}

}
```

运行结果如图





利用 setContentView() 来转换页面有一个优点，就是不管是类变量，还是类函数都在一个 Activity 中，不需要参数的传递。

1.2.2 添加 Activity

创建项目

新建一个名为“MulActivity”的项目，如图：

Project name:

Contents

- ☒ Create new project in workspace
- ☐ Create project from existing source
- ☒ Use default location

Location:

- ☐ Create project from existing sample

Samples:

Build Target

Target Name	Vendor	Platform	AP...
<input type="checkbox"/> Android 2.0	Android Open Source Project	2.0	5
<input checked="" type="checkbox"/> Android 2.1	Android Open Source Project	2.1	7

Standard Android platform 2.1

Properties

Application name:

Package name:

☒ Create Activity:

Min SDK Version:

然后新建另一个 Activity，如下图：



Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

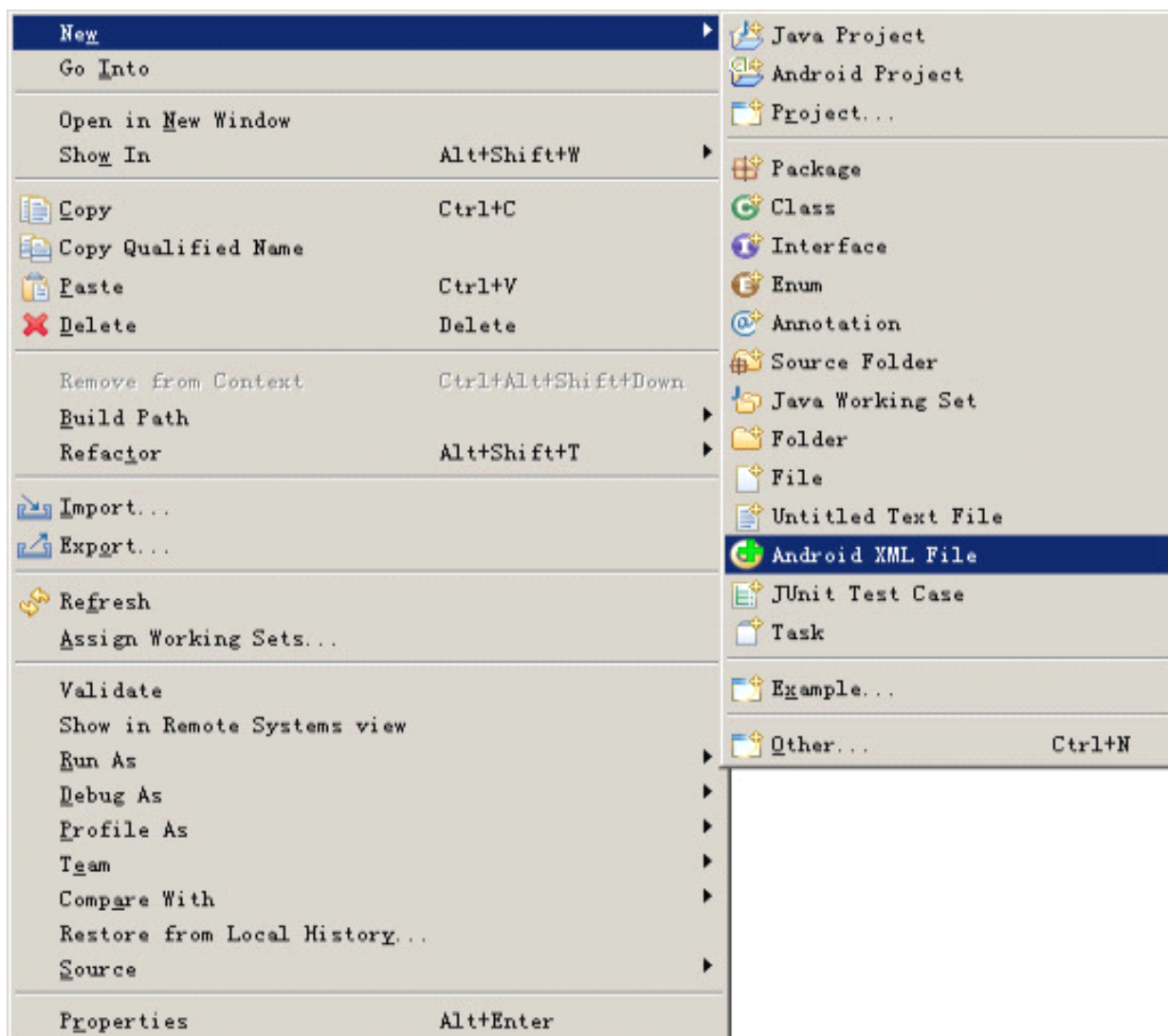
☐ Constructors from superclass

☒ Inherited abstract methods

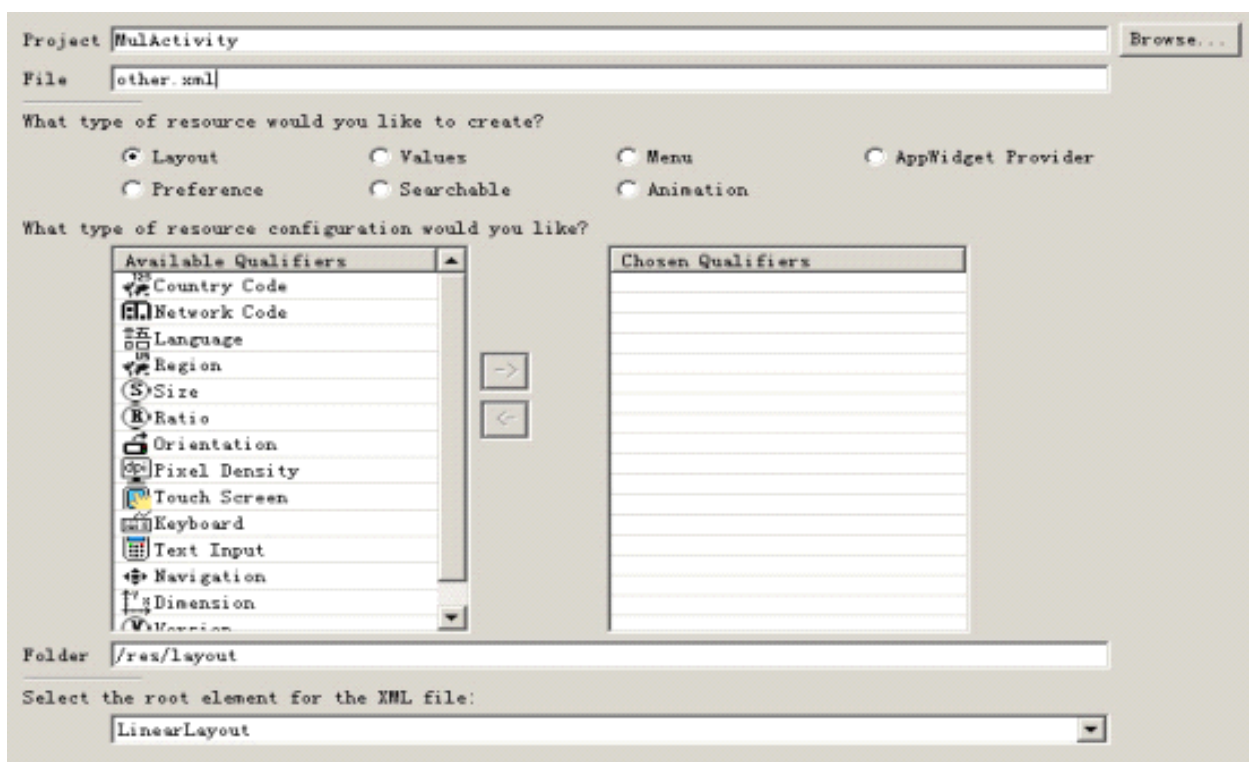
Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

打开 layout 文件夹，为 OtherActivity 新建一个界面文件 other.xml, 如图



选择新建之后，如图：



在 other.xml 中加入一个 TextView，代码如下；

other.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="第二个Activity"
    ></TextView>
</LinearLayout>
```

在 main.xml 中加入一个 Button，代码如下；

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
```




```
        android:layout_height="wrap_content"
        android:text="第一个Activity"
    />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/next"
        android:text="打开新的Activity"
    ></Button>
</LinearLayout>
```

编写 MainActivity.java

在 MainActivity 中单击按钮，打开 OtherActivity，这个时候就用了 intent(意图)，intent 在 Android 系统中非常重要，intent 用于激活组件和在组件之间传递数据，有关 intent 更深入的讲解在后面会介绍，MainActivity 的代码如下：

MainActivity.java

```
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button nextButton=(Button)findViewById(R.id.next);
        nextButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                //打开 other activity
                //Intent 意图 ,用于激活组件(Activity)的，和在组件之间传递数据的
                Intent intent = new Intent(MainActivity.this, OtherActivity.class); // 激活
                OtherActivity

            }
        });
    }
}
```

然后在 OtherActivity 中实现 onCreate() 方法，加载 other.xml，代码如下：

```
public class OtherActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
```



```
super.onCreate(savedInstanceState);  
setContentView(R.layout.other);  
}  
}
```

在 AndroidManifest.xml 文件中注册 OtherActivity，代码如下；

AndroidManifest.xml

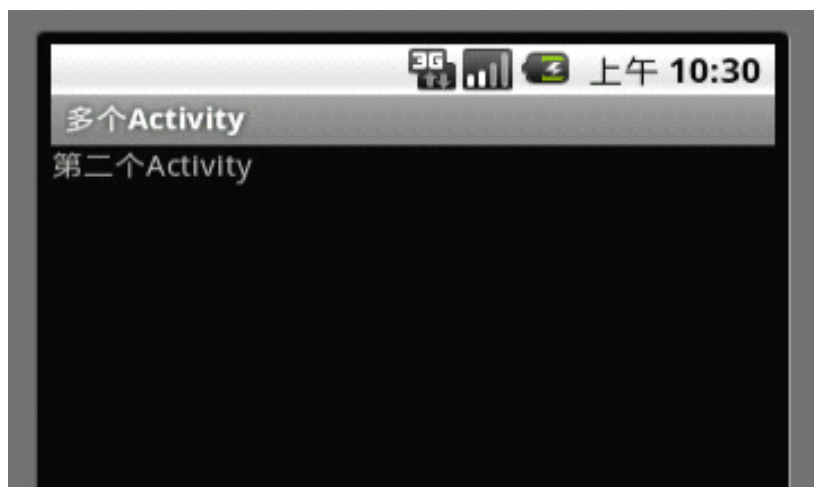
```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.sharpandroid.many"  
    android:versionCode="1"  
    android:versionName="1.0">  
    <application android:icon="@drawable/icon" android:label="@string/app_name">  
        <activity android:name=".MainActivity"  
            android:label="@string/app_name">  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN" />  
                <category android:name="android.intent.category.LAUNCHER" />  
            </intent-filter>  
        </activity>  
        <activity android:name=".OtherActivity"  
            android:label="@string/app_name">  
            </activity>  
    </application>  
    <uses-sdk android:minSdkVersion="7" />  
</manifest>
```

运行应用，结果显示如图

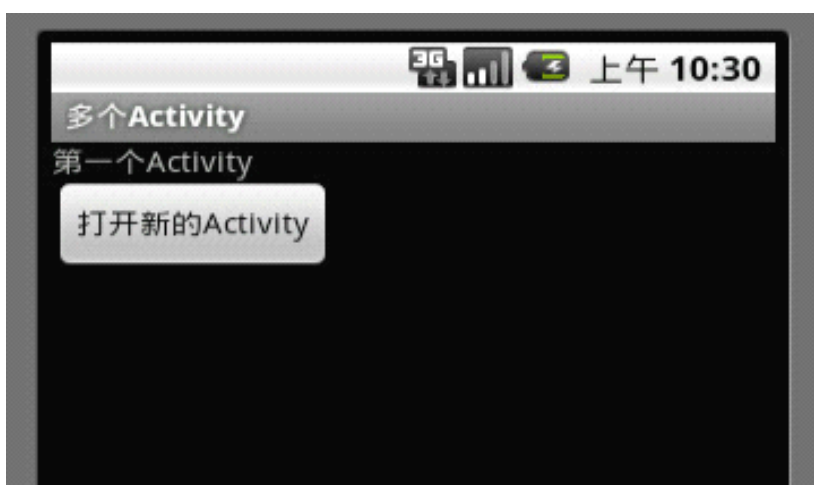




单击按钮之后，结果如图



单击手机的返回按钮，就可以返回第一个 Activity，如图



1.2.3 得到新打开 Activity 关闭后返回的数据

如果想在 Activity 中得到新打开 Activity 关闭后返回的数据，你需要使用系统提供的 `startActivityForResult(Intent intent, int requestCode)` 方法打开新的 Activity，新的 Activity 关闭后会向前面的 Activity 传回数据，为了得到传回的数据，你必须在前面的 Activity 中重写 `onActivityResult(int requestCode, int resultCode, Intent data)` 方法。

首先，在 `other.xml` 中加入一个关闭按钮，代码如下：

`Other.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
```



```

xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/other"
    android:text="第二个Activity"
></TextView>
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/close"
    android:text="关闭"
></Button>
</LinearLayout>

```

在OtherActivity关闭之前, 把要返回的数据放入到intent中, 然后调用OtherActivity的setResult()方法, 代码如下;

```

public class OtherActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.other);
        Intent intent = this getIntent(); //得到激活该组件的意图

        Bundle bundle= intent.getExtras();

        String name=bundle.getString("name");
        int age=bundle.getInt("age");
        TextView paramView = (TextView) this.findViewById(R.id.other);
        paramView.setText("名称: " + name + " 年龄: " + age); //显示数据

        Button button=(Button)findViewById(R.id.close);
        button.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent data=new Intent();
                data.putExtra("name", "王三");
                data.putExtra("age", 23); //要返回的数据
                OtherActivity.this.setResult(1, data); //设置返回码和数据, 返回码
                //可以任意
                OtherActivity.this.finish(); //关闭Activity
            }
        });
    }
}

```



```
        }  
    });  
}  
}
```

MainActivity 如何获得 OtherActivity 返回的数据呢？那就需要重写 MainActivity 的 onActivityResult() 方法。在代码编辑区的空白处点击鼠标右键，选择如图 5-18

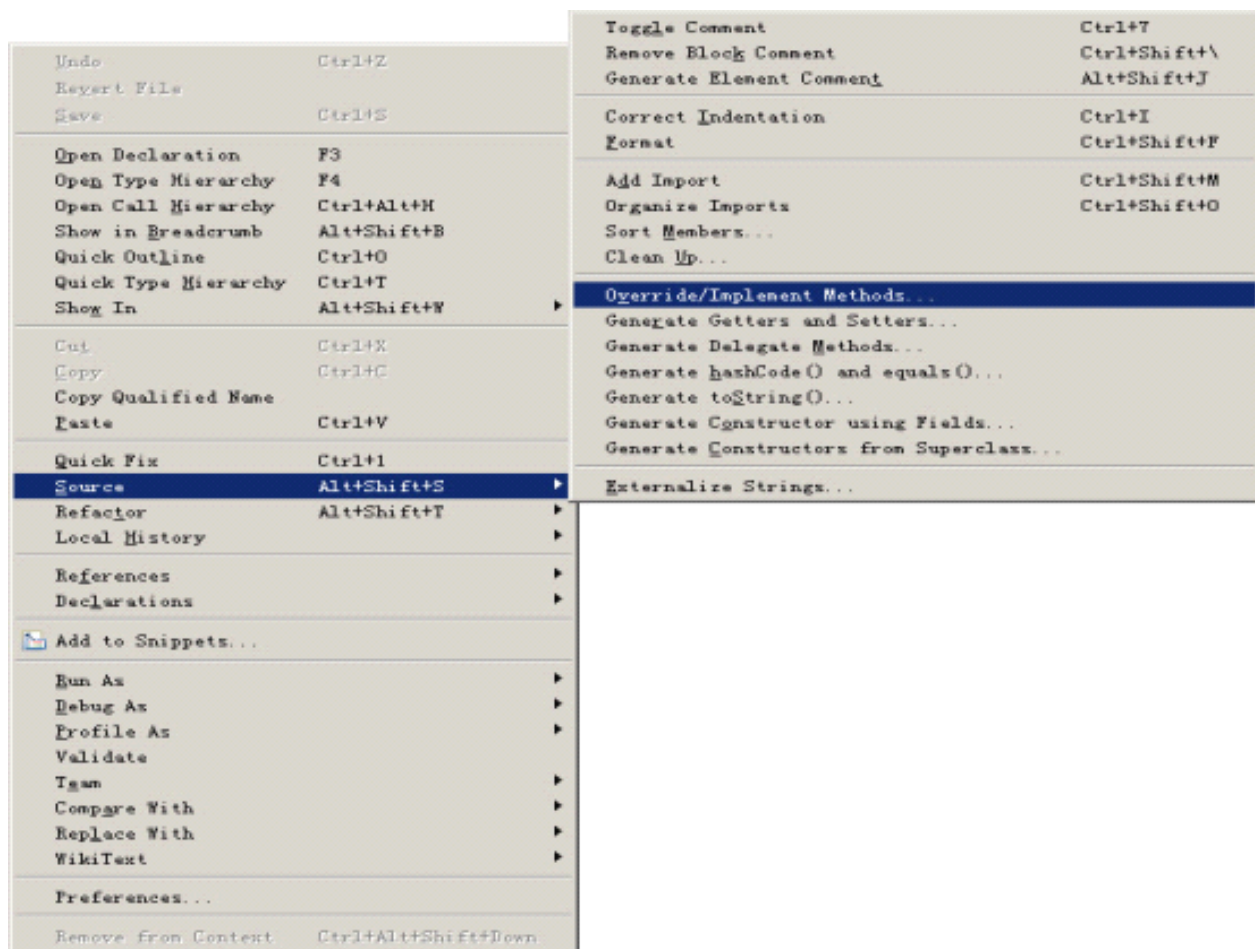


图 5-18

选择点击之后，在 onActivityResult() 方法前打上对号，如图 5-19

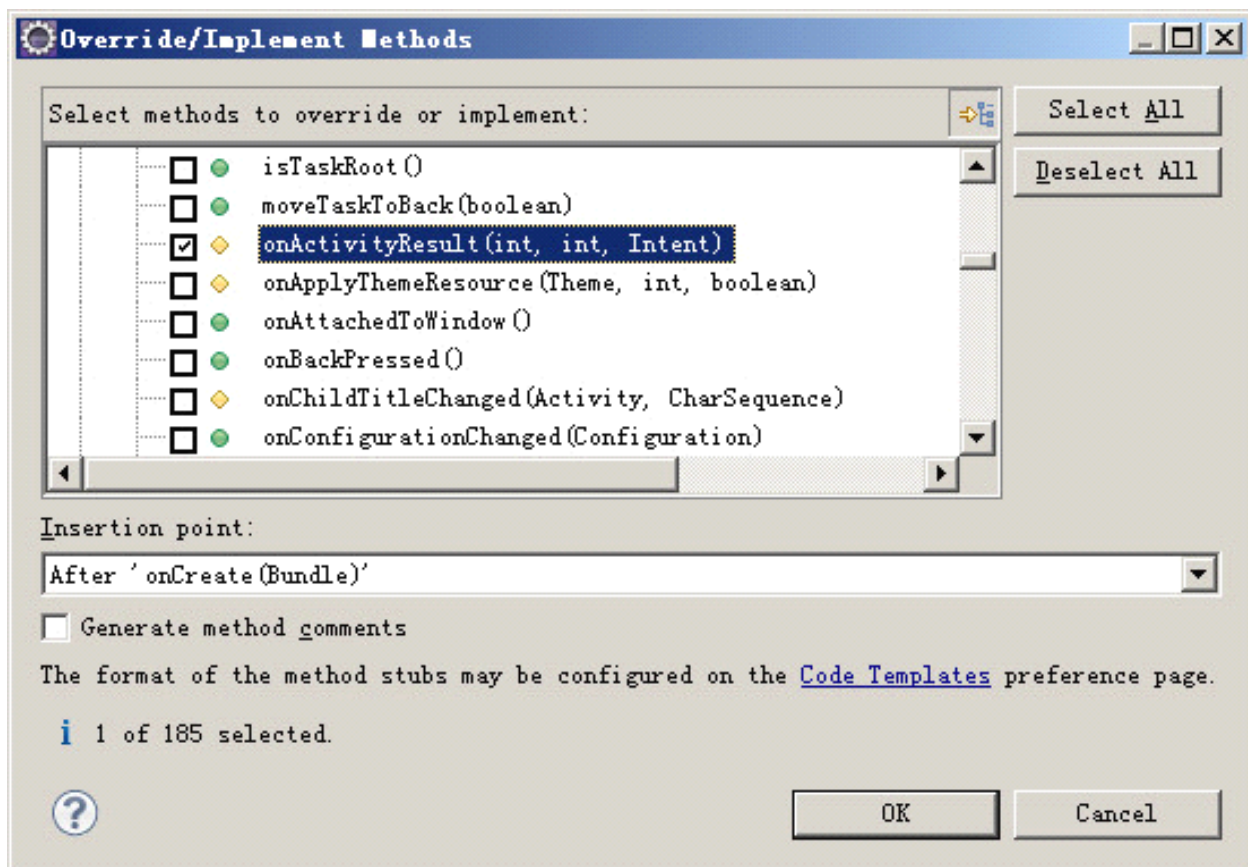


图 5-19

你会发现 onActivityResult() 方法有三个参数, 第一个参数: 打开 OtherActivity 的请求码, 第二个参数: OtherActivity 所设置的返回码, 第三个参数: OtherActivity 返回的数据, 代码如下:

MainActivity.java

```
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button nextButton=(Button)findViewById(R.id.next);
        nextButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
                OtherActivity.class); //激活OtherActivity
                intent.putExtra("name", "hou");
                intent.putExtra("age", 22);

                Bundle bundle=new Bundle();
```



```
        bundle.putString("name", "侯二");
        bundle.putInt("age", 22);
        intent.putExtras(bundle); // 附上额外的数据
        startActivityForResult(intent, 1);
        // 如果需要打开的Activity向前面Activity返回数据, 就必须使用此方法
    打开Activity
    }
    });
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    // 用提示来显示返回的信息
    Toast.makeText(MainActivity.this,
        data.getStringExtra("name") + data.getIntExtra("age", 1), 3).show();
    super.onActivityResult(requestCode, resultCode, data);
}
}
```

运行应用, 显示结果如图 5-20



图 5-20

单击按钮之后, 显示如图 5-21;



图 5-21

单击【关闭】按钮之后，显示如图：



图 5-22

通过运行结果，我们已经发现结果已经返回成功了。

1.2.4 请求码的作用

使用 `startActivityForResult(Intent intent, int requestCode)` 方法打开新的 Activity，我们需要为 `startActivityForResult()` 方法传入一个请求码(第二个参数)。请求码的值是根据业务需要由自己设定，用于标识请求来源。例如：一个 Activity 有两个按钮，点击这两个按钮都会打开同一个 Activity，不管是哪个按钮打开新 Activity，当新的 Activity 关闭后，系统都会调用前面 Activity 的 `onActivityResult(int requestCode, int resultCode, Intent data)` 方法。在 `onActivityResult()` 方法需要知道新 Activity 是由哪个按钮打开的，并且要做出相应的业务处理。如图 5-23

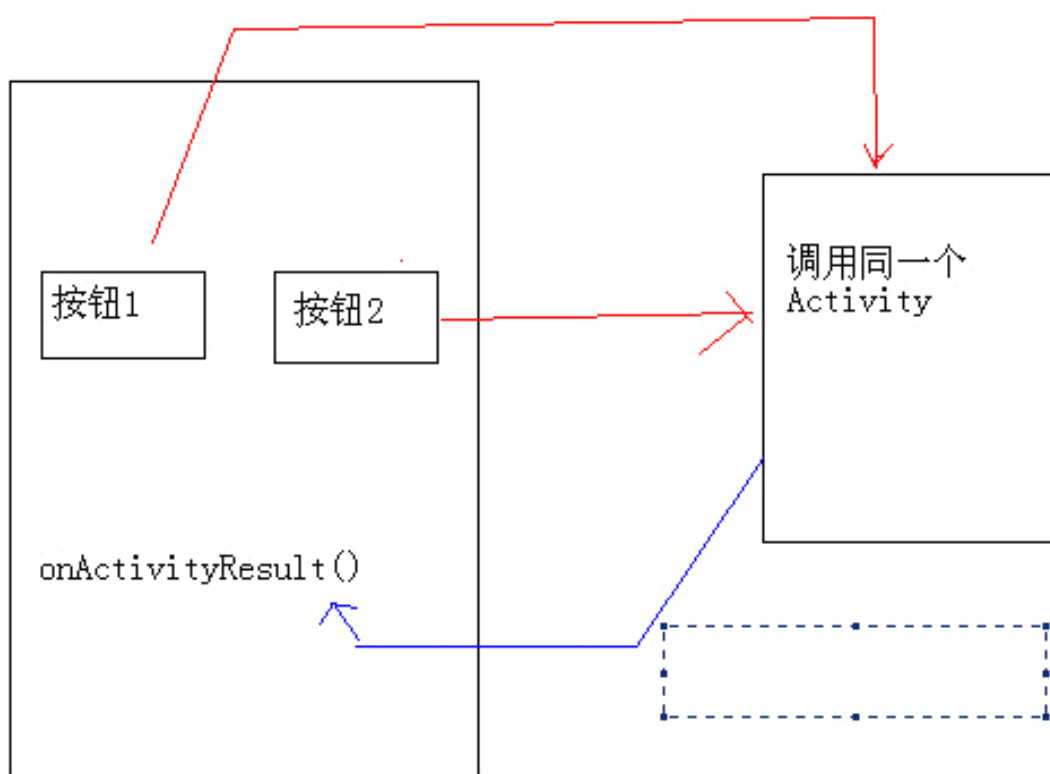


图 5-23

有关请求代码如下；

```
public void onCreate(Bundle savedInstanceState) {
    ....
    button1.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            startActivityForResult (new Intent(MainActivity.this,
NewActivity.class), 1);
        });
    button2.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {
            startActivityForResult (new Intent(MainActivity.this,
NewActivity.class), 2);
        });
    @Override protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
        switch(requestCode) {
            case 1:
                //来自按钮 1 的请求，作相应业务处理
            case 2:
                //来自按钮 2 的请求，作相应业务处理
        }
    }
}
```



1.2.5 结果码的作用

在一个 Activity 中，可能会使用 `startActivityForResult()` 方法打开多个不同的 Activity 处理不同的业务，当这些新 Activity 关闭后，系统都会调用前面 Activity 的 `onActivityResult(int requestCode, int resultCode, Intent data)` 方法。

如图 5-24

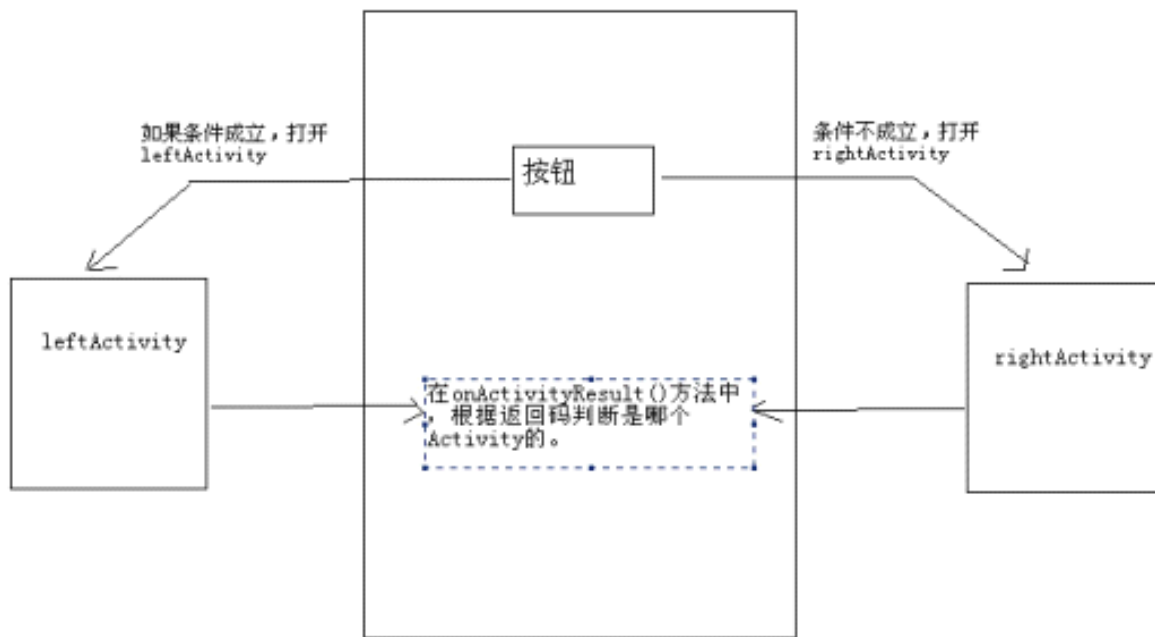


图 5-24

为了知道返回的数据来自于哪个新 Activity，在 `onActivityResult()` 方法中可以这样做（`ResultActivity` 和 `NewActivity` 为要打开的新 Activity）：

```
public class ResultActivity extends Activity {
    .....
    ResultActivity.this.setResult(1, intent);
    ResultActivity.this.finish();
}

public class NewActivity extends Activity {
    .....
    NewActivity.this.setResult(2, intent);
    NewActivity.this.finish();
}

public class MainActivity extends Activity { // 在该 Activity 会打开
ResultActivity 和 NewActivity
    @Override protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
        switch(resultCode) {
            case 1:
```



```
        // ResultActivity 的返回数据
        case 2:
            // NewActivity 的返回数据
            }
        }
    }
```

大致：怎么样，是不是觉得 Activity 的功能越来越强大了呢？

小安：对啊，它不仅能管理各个窗口，而且能使用请求码，结果码，这样就能实现更加复杂的管理了。

大致：不要惊讶的太早，这里还只是给你介绍了 Activity 的使用。要想“机器人”拥有强大的能力，还要结合其它的组件。介绍完“管理员”，下面就给你介绍一下 Android 的“邮递员” Intent 吧..

2 “机器人”的邮递员——Intent

Intent 简介

Android 基本的设计理念是鼓励减少组件间的耦合，因此 Android 提供了 Intent（意图），Intent 提供了一种通用的消息系统，它允许在你的应用程序与其它的应用程序间传递 Intent 来执行动作和产生事件。通过使用 Intent 可以激活 Android 应用的三个核心组件：活动、服务和广播接收器。

Intent 可以划分成显式意图和隐式意图。

显式意图：调用 Intent.setComponent() 或 Intent.setClass() 方法明确指定了组件名的 Intent 为显式意图，显式意图明确指定了 Intent 应该传递给哪个组件。

隐式意图：没有明确指定组件名的 Intent 为隐式意图。Android 系统会根据隐式意图中设置的动作(action)、类别(category)、数据（URI 和数据类型）找到最合适的组件来处理这个意图。代码如下：

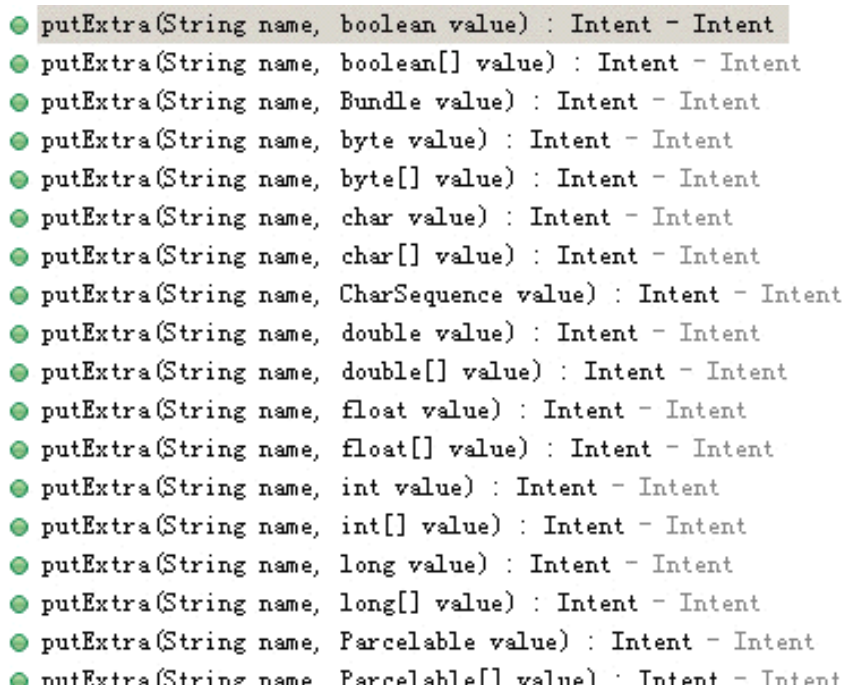
```
<intent-filter>
    <action android:name="android.intent.action.CALL" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:scheme="tel" />
</intent-filter>
<intent-filter>
    <action android:name="android.intent.action.CALL" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:mimeType="vnd.android.cursor.item/phone" />
</intent-filter>
```



对于隐式意图，Android 是怎样寻找到这个最合适的组件呢？记的前面我们在定义活动时，指定了一个 intent-filter，Intent Filter（意图过滤器）其实就是用来匹配隐式 Intent 的，当一个意图对象被一个意图过滤器进行匹配测试时，只有三个方面会被参考到：动作、数据（URI 以及数据类型）和类别。

1.2.6 不同 Activity 之间的数据传输

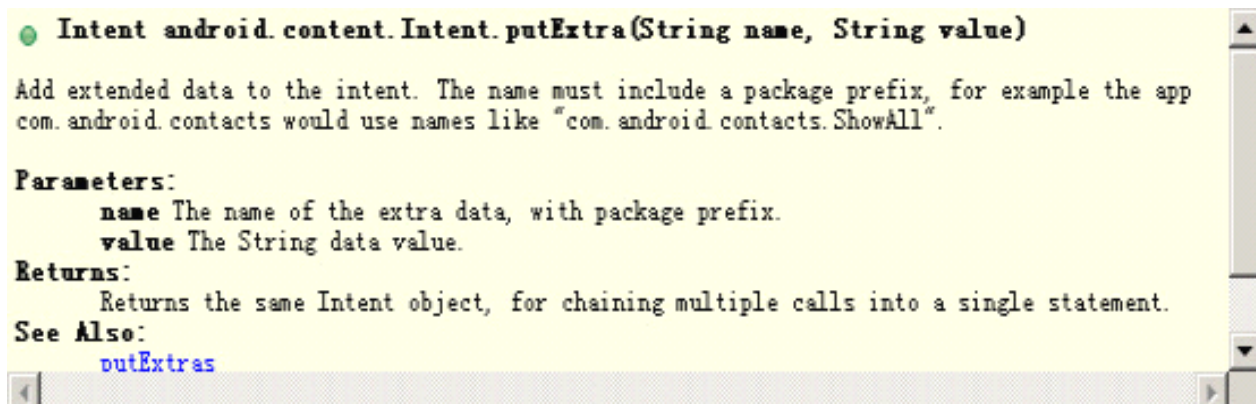
有的会问，都说它是“邮递员”了，那么它是怎样实现邮递员的功能呢？我们之前已经知道，Intent 除了激活组件还有一项作用就是传递数据。如果在打开第二个 Activity 的时候，需要将当前 Activity 中的部分信息传递过去，这个问题要怎样解决呢？这时，就用到了我们的“邮递员”intent 的 putExtra 方法，如图 5-11



```
● putExtra(String name, boolean value) : Intent - Intent
● putExtra(String name, boolean[] value) : Intent - Intent
● putExtra(String name, Bundle value) : Intent - Intent
● putExtra(String name, byte value) : Intent - Intent
● putExtra(String name, byte[] value) : Intent - Intent
● putExtra(String name, char value) : Intent - Intent
● putExtra(String name, char[] value) : Intent - Intent
● putExtra(String name, CharSequence value) : Intent - Intent
● putExtra(String name, double value) : Intent - Intent
● putExtra(String name, double[] value) : Intent - Intent
● putExtra(String name, float value) : Intent - Intent
● putExtra(String name, float[] value) : Intent - Intent
● putExtra(String name, int value) : Intent - Intent
● putExtra(String name, int[] value) : Intent - Intent
● putExtra(String name, long value) : Intent - Intent
● putExtra(String name, long[] value) : Intent - Intent
● putExtra(String name, Parcelable value) : Intent - Intent
● putExtra(String name, Parcelable[] value) : Intent - Intent
```

图 5-12

putExtra 方法几乎包括了所有的基本数据类型。



```
● Intent android.content.Intent.putExtra(String name, String value)

Add extended data to the intent. The name must include a package prefix, for example the app
com.android.contacts would use names like "com.android.contacts.ShowAll".

Parameters:
    name The name of the extra data, with package prefix.
    value The String data value.

Returns:
    Returns the same Intent object, for chaining multiple calls into a single statement.

See Also:
    putExtras
```

图 5-12

第一个参数 name：表示携带的数据的名称

第二个参数 value：表示数据的值。在这里，Intent 的作用与 JSP 中的 attribute、



parameter 相类似。不过 Intent 的作用域在 Activity 之间。

编写 MainActivity.java

传递数据的 MainActivity 的代码如下：

MainActivity.java

```
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button nextButton=(Button)findViewById(R.id.next);
        nextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //打开 other activity
                //Intent 意图 ,用于激活组件(Activity)的, 和在组件之间传递数据的
                Intent intent = new Intent(MainActivity.this,
OtherActivity.class);//所要干的一件事情是:激活OtherActivity
                intent.putExtra("name", "hou") ;
                intent.putExtra("age", 22) ;
                startActivity(intent);
            }
        });
    }
}
```

在 OtherActivity 中如何获取传递的数据呢？OtherActivity 使用 getIntent() 方法得到上个 Activity 专递过来的 intent 内容。注意：该方法获得的 intent 内容与上一个完全相同，但并不是同一个实例，只是 setIntent(Intent newIntent)进行了赋值。获得 Intent 后，根据数据的类型使用 intent 的相应方法获取数据，如图 5-13



```
● getDoubleExtra(String name, double defaultValue) : double - Intent
● getExtras() : Bundle - Intent
● getFlags() : int - Intent
● getFloatArrayExtra(String name) : float[] - Intent
● getFloatExtra(String name, float defaultValue) : float - Intent
● getIntArrayExtra(String name) : int[] - Intent
● getIntegerArrayListExtra(String name) : ArrayList<Integer> - Intent
● getIntExtra(String name, int defaultValue) : int - Intent
● getLongArrayExtra(String name) : long[] - Intent
● getLongExtra(String name, long defaultValue) : long - Intent
● getPackage() : String - Intent
● getParcelableArrayExtra(String name) : Parcelable[] - Intent
● getParcelableArrayListExtra(String name) : ArrayList<T> - Intent
● getParcelableExtra(String name) : T - Intent
● getScheme() : String - Intent
● getSerializableExtra(String name) : Serializable - Intent
● getShortArrayExtra(String name) : short[] - Intent
● getShortExtra(String name, short defaultValue) : short - Intent
● getSourceBounds() : Rect - Intent
```

图 5-13

编写 OtherActivity.java

OtherActivity 接收数据的代码如下:

```
public class OtherActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.other);
        Intent intent = this getIntent();//得到激活该组件的意图
        String name = intent.getStringExtra("name");//从意图中获取前面Activity
        传递过来的参数
        int age = intent.getIntExtra("age", 0);
        TextView paramView = (TextView) this.findViewById(R.id.other);
        paramView.setText("名称: " + name + " 年龄: " + age); //显示数据

    }
}
```

运行应用, 结果显示如图 5-8

单击按钮, 界面显示结果如图 5-14;

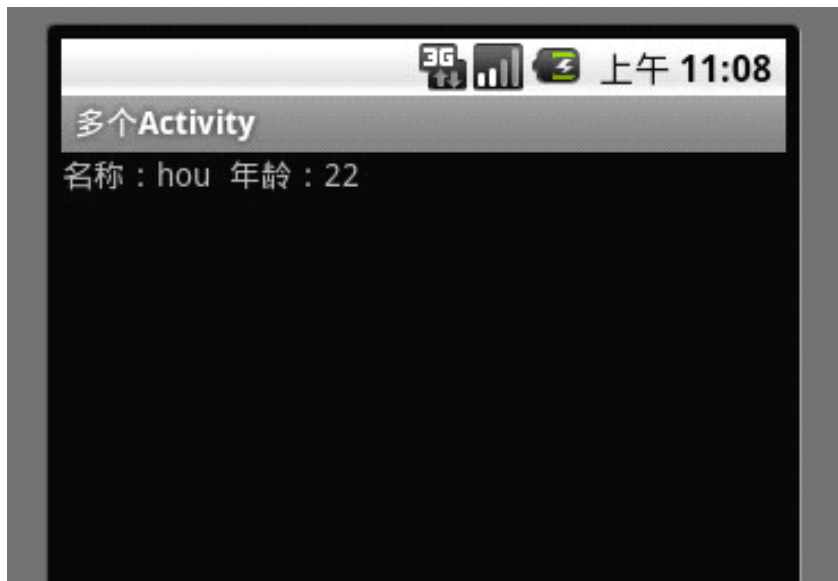


图 5-14

根据结果显示，两个 Activity 之间的数据传递成功。

小安：恩，这个邮递员使用起来还真是方便啊。那么它还有别的功能吗？

大致：当然有了，不过刚才只是给你讲了一种传递方式，不要小看我们的邮递员，它还有一个更专业的方式呢。

1.2.7 另一种传递数据的方式

Intent 还可以通过 Bundle 类携带数据，Bundle 类似 map 对象，Bundle 提供了常用类型的装填数据方法和获取方法，如图 5-15 和 5-16:

```
● notifyAll() : void - Object
● putAll(Bundle map) : void - Bundle
● putBoolean(String key, boolean value) : void - Bundle
● putBooleanArray(String key, boolean[] value) : void - Bundle
● putBundle(String key, Bundle value) : void - Bundle
● putByte(String key, byte value) : void - Bundle
● putByteArray(String key, byte[] value) : void - Bundle
● putChar(String key, char value) : void - Bundle
● putCharArray(String key, char[] value) : void - Bundle
● putCharSequence(String key, CharSequence value) : void - Bundle
● putDouble(String key, double value) : void - Bundle
● putDoubleArray(String key, double[] value) : void - Bundle
● putFloat(String key, float value) : void - Bundle
● putFloatArray(String key, float[] value) : void - Bundle
● putInt(String key, int value) : void - Bundle
● putIntArray(String key, int[] value) : void - Bundle
```

图 5-15



```
● getBoolean(String key) : boolean - Bundle
● getBoolean(String key, boolean defaultValue) : boolean - Bundle
● getBooleanArray(String key) : boolean[] - Bundle
● getBundle(String key) : Bundle - Bundle
● getBytes(String key) : byte - Bundle
● getByte(String key, byte defaultValue) : Byte - Bundle
● getByteArray(String key) : byte[] - Bundle
● getChar(String key) : char - Bundle
● getChar(String key, char defaultValue) : char - Bundle
● getCharArray(String key) : char[] - Bundle
● getCharSequence(String key) : CharSequence - Bundle
● getClass() : Class<? extends Object> - Object
● getDouble(String key) : double - Bundle
● getDouble(String key, double defaultValue) : double - Bundle
● getDoubleArray(String key) : double[] - Bundle
● getFloat(String key) : float - Bundle
● ...
```

图 5-16

编写 MainActivity.java

MainActivity 用 Bundle 携带数据的代码如下:

```
public class MainActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        Button nextButton=(Button)findViewById(R.id.next);
        nextButton.setOnClickListener(new View.OnClickListener() {

            @Override
            public void onClick(View v) {
                //打开other activity
                //用于激活组件(Activity)和在组件之间传递数据的 Intent 意图
                Intent intent = new Intent(MainActivity.this,
                OtherActivity.class);
                //激活OtherActivity
                Bundle bundle=new Bundle();
                bundle.putString("name", "侯二");
                bundle.putInt("age", 22);
                //附上额外的数据
                intent.putExtras(bundle);
                startActivity(intent);
            }
        });
    }
}
```



```
});
```

OtherActivity 获取参数的方式是和之前一样的，也可以使用 intent 的 `getExtras()` 方法获取 Bundle 对象，然后使用 `get` 类型的方法获取数据，代码如下：

编写 OtherActivity.java

```
public class OtherActivity extends Activity {  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.other);  
        Intent intent = this getIntent();//得到激活该组件的意图  
        Bundle bundle= intent.getExtras();//得到Bundle对象  
        String name=bundle.getString("name");  
        int age=bundle.getInt("age");  
        TextView paramView = (TextView) this.findViewById(R.id.other);  
        paramView.setText("名称: "+ name + " 年龄: "+ age); //显示数据  
    }  
}
```

运行应用，结果显示如图 5-8

单击按钮，界面显示结果如图 5-17



图 5-17

通过 Bundle 携带数据，也测试完成了。

1.2.8 动作测试 (Action test)

一个意图对象只能指定一个动作名称，而一个过滤器可能列举多个动作名称。如果意图对象或过滤器没有指定任何动作，结果将如下：



- 如果过滤器没有指定任何动作，那么将阻塞所有的意图，因此所有的意图都会测试失败。没有意图能够通过这个过滤器。
- 另一方面，只要过滤器包含至少一个动作，一个没有指定动作的意图对象自动通过这个测试

1.2.9 类别测试 (Category test)

对于一个能够通过类别匹配测试的意图，意图对象中的类别必须匹配过滤器中的类别。这个过滤器可以列举其他的类别，但它不能遗漏在这个意图中的任何类别。

原则上一个没有类别的意图对象应该总能够通过匹配测试，而不管过滤器里有什么。大部分情况下这个是对的。但有一个例外，Android 把所有传给 `startActivity()` 的隐式意图当作它们包含至少一个类别：`"android.intent.category.DEFAULT"` (CATEGORY_DEFAULT 常量)。因此，想要接收隐式意图的活动必须在它们的意图过滤器中包含 `"android.intent.category.DEFAULT"`。(带 `"android.intent.action.MAIN"` 和 `"android.intent.category.LAUNCHER"` 设置的过滤器是例外)

1.2.10 数据测试 (Data test)

当一个意图对象中的 URI 被用来和一个过滤器中的 URI 比较时，比较的是 URI 的各个组成部分。例如，如果过滤器仅指定了一个 `scheme`，所有该 `scheme` 的 URIs 都能够和这个过滤器相匹配；如果过滤器指定了一个 `scheme`、主机名但没有路径部分，所有具有相同 `scheme` 和主机名的 URIs 都可以和这个过滤器相匹配，而不管它们的路径；如果过滤器指定了一个 `scheme`、主机名和路径，只有具有相同 `scheme`、主机名和路径的 URIs 才可以和这个过滤器相匹配。当然，一个过滤器中的路径规格可以包含通配符，这样只需要部分匹配即可。

数据测试同时比较意图对象和过滤器中指定的 URI 和数据类型。规则如下：

- a. 一个既不包含 URI 也不包含数据类型的意图对象仅在过滤器也同样没有指定任何 URIs 和数据类型的情况下才能通过测试。
- b. 一个包含 URI 但没有数据类型的意图对象仅在它的 URI 和一个同样没有指定数据类型的过滤器里的 URI 匹配时才能通过测试。这通常发生在类似于 `mailto:` 和 `tel:` 这样的 URIs 上：它们并不引用实际数据。
- c. 一个包含数据类型但不包含 URI 的意图对象仅在这个过滤器列举了同样的数据类型而且也没有指定一个 URI 的情况下才能通过测试。
- d. 一个同时包含 URI 和数据类型（或者可从 URI 推断出数据类型）的意图对象可以通过测试，如果它的类型和过滤器中列举的类型相匹配的话。如果它的 URI 和这个过滤器中的一个 URI 相匹配或者它有一个内容 `content:` 或者文件 `file:` URI 而且这个过滤器没有指定一个 URI，那么它也能通过测试。换句话说，一个组件被假定为支持 `content:` 和 `file:` 数据如果它的过滤器仅列举了一个数据类型



1.3 意图测试

通过显示意图打开 Activity，在之前已经做过了，现在我们来做一个通过隐式意图打开 Activity，在之前的 MulActivity 项目中，新建一个 IntentActivity 类，如图 3-1

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

图 3-1

然后新建一个 Layout 文件 intent.xml，如图 3-2

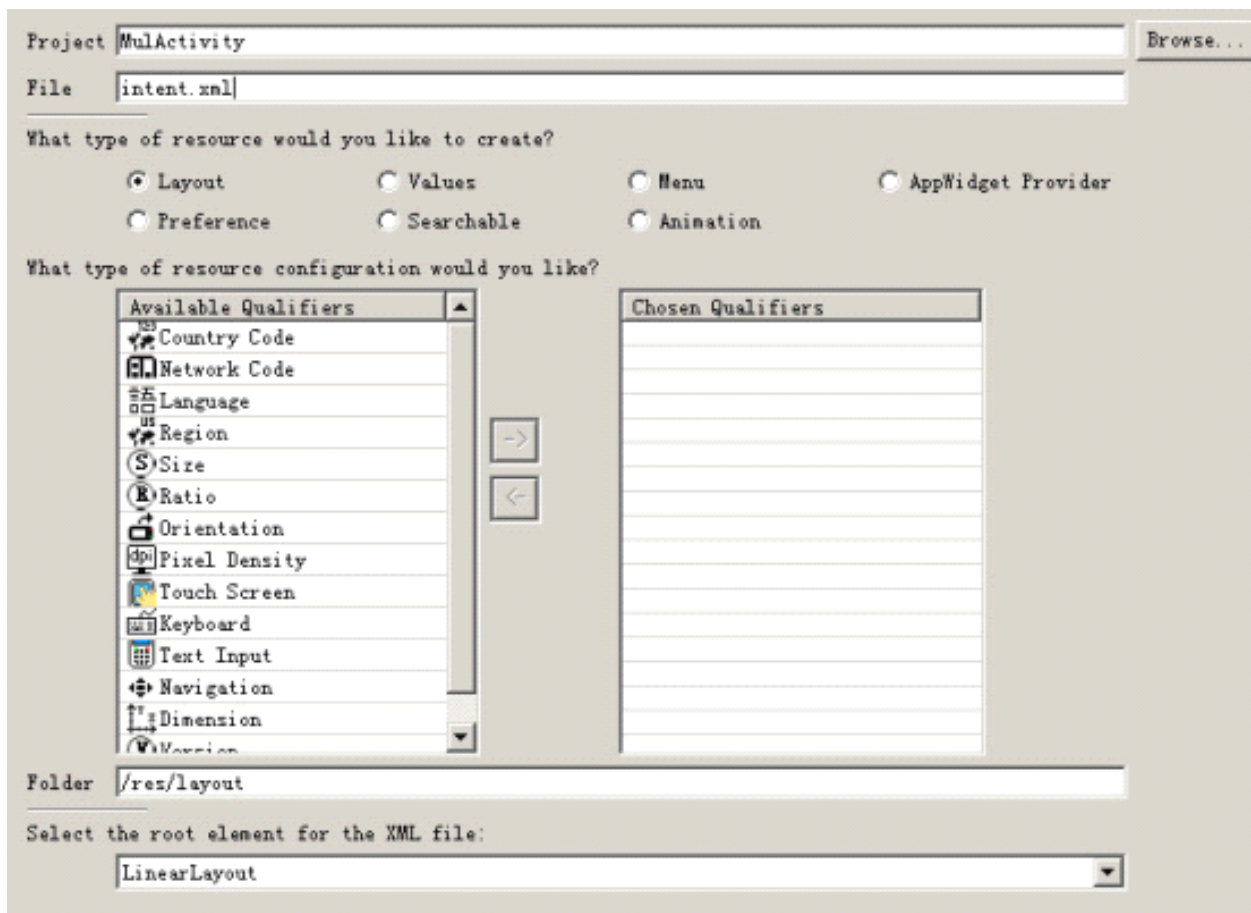


图 3-2

编写 intent.xml:

在 intent.xml 中添加一个 TextView 控件，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/intent"
        android:text="通过隐式意图打开的Activity"
    ></TextView>
</LinearLayout>
```

编写 IntentActivity.java:

在 IntentActivity 中加载 intent.xml，代码如下：

```
public class IntentActivity extends Activity {
```




```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.intent);  
}  
}
```

编写 AndroidManifest.xml:

在 AndroidManifest.xml 文件中对 IntentActivity 进行配置，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.sharpandroid.many"  
    android:versionCode="1"  
    android:versionName="1.0">  
    <application android:icon="@drawable/icon" android:label="@string/app_name">  
        <activity android:name=".MainActivity"  
            android:label="@string/app_name">  
            <intent-filter>  
                <action android:name="android.intent.action.MAIN" />  
                <category android:name="android.intent.category.LAUNCHER" />  
            </intent-filter>  
        </activity>  
        <activity android:name=".OtherActivity"  
            android:label="@string/app_name">  
        </activity>  
        <activity android:name=".IntentActivity"  
            android:label="@string/app_name">  
            <intent-filter>  
                <action android:name="intent.IntentActivity" />  
                <category android:name="android.intent.category.DEFAULT" />  
            </intent-filter>  
        </activity>  
    </application>  
    <uses-sdk android:minSdkVersion="7" />  
</manifest>
```

编写 main.xml:

在 main.xml 中加入一个新的按钮，通过隐式意图打开 IntentActivity，代码如下：

```
<?xml version="1.0" encoding="utf-8"?>  
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"
```




```
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="第一个Activity"
    />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/next"
        android:text="打开新的Activity"
    ></Button>

    <Button
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="打开IntentActivity"
        android:id="@+id/intentbutton"
    />
</LinearLayout>
```

完成MainActivity.java:

在 MainActivity 中点击按钮，通过隐式意图调用 IntentActivity, 代码如下:

```
Button button2 = (Button) this.findViewById(R.id.intentbutton);
button2.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent("intent.IntentActivity");

        startActivity(intent);
    }
});
```

运行应用，显示结果如图 3-3

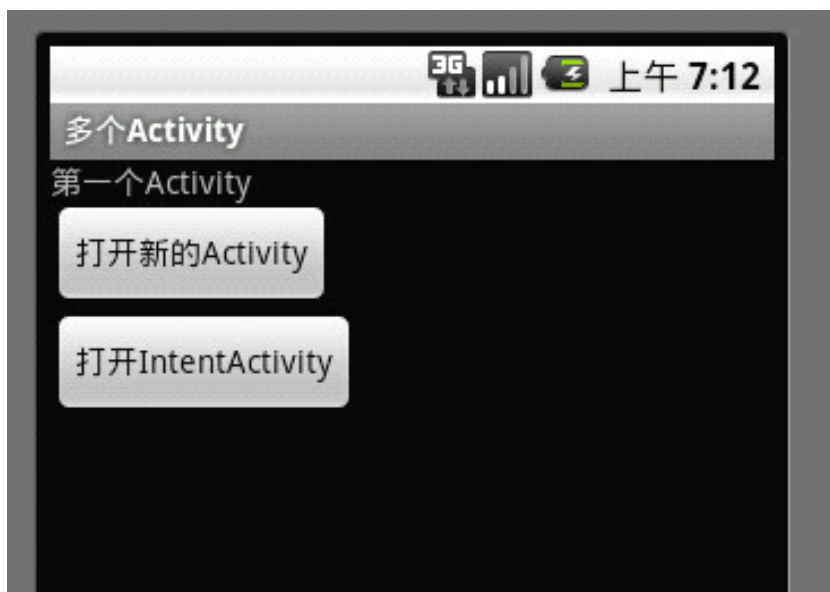


图 3-3

单击按钮【打开 IntentActivity】，显示结果为图 3-4;



图 3-4

上述通过隐式意图打开 Activity 的测试完成，还有一个知识点，在 AndroidManifest.xml 文件中对 IntentActivity 进行配置时，可以加入多个过滤器，只要意图中的动作名称和过滤器中的一个动作名称匹配就可以了。