第 3 章

Activity

学习目标

- 掌握 Activity 的生命周期。
- 掌握 Activity 的常用方法。
- 掌握显式和隐式意图的使用。
- 掌握 Activity 的启动方式。
- 掌握 Activity 中的数据传递方式。

在 Android 系统中，用户与程序的交互是通过 Activity 完成的，同时 Activity 也是 Android 四大组件中使用最多的一个，本章将详细讲解有关 Activity 的知识。

3.1 Activity 基础

3.1.1 认识 Activity

Activity 的中文意思是“活动”，它是 Android 应用中负责与用户交互的组件。相当于 Swing 编程中的 JFrame 控件，与其不同的是，JFrame 本身可以设置布局管理器，不断地向其添加组件，而 Activity 只能通过 setContentView(View)来显示布局文件中已经定义的组件。

在应用程序中，Activity 就像一个界面管理员，用户在界面上的操作是通过 Activity 来管理的，下面是 Activity 的常用事件。

- OnKeyDown(int keyCode,KeyEvent event)：按键按下事件。
- OnKeyUp(int keyCode,KeyEvent event)：按键松开事件。
- OnTouchEvent(MotionEvent event)：单击屏幕事件。

当用户按下手机界面上的按键时，就会触发 Activity 中对应的事件 OnKeyDown()来响应用户的操作。3.1.2 节会通过具体实例讲解 Activity 的常用事件。

3.1.2 如何创建 Activity

创建一个 Activity 的具体步骤如下。

(1) 定义一个类继承自 android.app.Activity 或其子类,如图 3-1 所示。

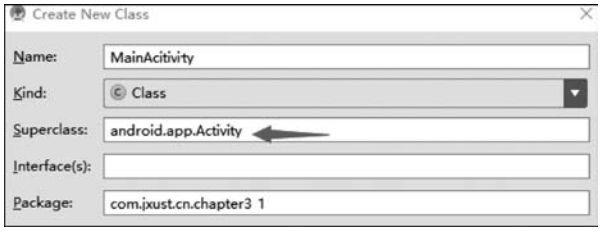

图 3-1 创建 Activity

(2) 在 res/layout 目录下创建一个 xml 文件,用于创建 Activity 的布局。

(3) 在 app/manifests 目录下的 AndroidManifest.xml 清单文件中注册 Activity,如图 3-2 所示。

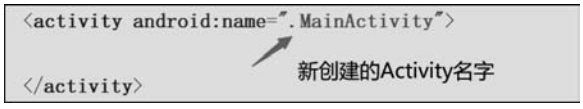

图 3-2 Activity 注册

(4) 重写 Activity 的 onCreate()方法,并在该方法中使用 setContentView()加载指定的布局文件。新创建的 Activity 的具体代码如下:

```
package com.jxust.cn.chapter3_1;
import android.app.Activity;
import android.os.Bundle;
public class MainActivity extends Activity {
//项目的入口 Activity
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        //设置 Activity 显示的布局
        setContentView(R.layout.activity_main);
    }
}
```

接下来通过具体的例子讲解 3.1.1 节中几个 Activity 的常用事件,具体的 Activity 代码如下:

```
public class MainActivity extends Activity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
    }
    //响应按键按下事件
    public boolean onKeyDown(int keyCode, KeyEvent event){
  Toast.makeText(this,"按键已经按下了!",Toast.LENGTH_SHORT).show();
        return super.onKeyDown(keyCode, event);
    }
    //响应按键松开事件
    public boolean onKeyUp(int keyCode, KeyEvent event){
      Toast.makeText(this,"按键松开了!",Toast.LENGTH_SHORT).show();
        return super.onKeyDown(keyCode, event);
    }
    //响应屏幕触摸操作
    public boolean onTouchEvent( MotionEvent event){
      Toast.makeText(this,"触摸了屏幕!",Toast.LENGTH_SHORT).show();
        return super.onTouchEvent(event);
    }
}
```

运行效果如图 3-3 所示。

图 3-3　Activity 常用事件


视频讲解

3.1.3　Activity 的生命周期

每一个 Android 应用程序在运行时，对于底层的 Linux Kernel 而言都是一个单独的进程，但是对于 Android 系统而言，因为局限于手机画面的大小与使用的考虑，不能把每一个运行中的应用程序窗口都显示出来。所以通常手机系统的界面一次仅显示一个应用程序窗口，Android 使用了 Activity 的概念来表示界面。

Activity 的生命周期分为 3 种状态，分别是运行状态、暂停状态和停止状态。下面将详

细介绍这 3 种状态。

- 运行状态：当 Activity 在屏幕最前端的时候，它是有焦点的、可见的，可以供用户进行单击、长按等操作，这种状态称为运行状态。
- 暂停状态：在一些情况下，最上层的 Activity 没有完全覆盖屏幕，这时候被覆盖的 Activity 仍然对用户可见，并且存活。但当内存不足时，这个暂停状态的 Activity 可能会被杀死。
- 停止状态：当 Activity 完全不可见时，它就处于了停止状态，但仍然保留着当前状态和成员信息，当系统内存不足时，这个 Activity 就很容易被杀死。

Activity 从一种状态变到另一种状态时会经过一系列 Activity 类的方法。常用的回调方法如下。

- onCreate(Bundle savedInstanceState)：该方法在 Activity 的实例被 Android 系统创建后第一个被调用。通常在该方法中设置显示屏幕的布局、初始化数据、设置控件被单击的事件响应代码。
- onStart()：在 Activity 可见时执行。
- onRestart()：回到最上边的界面，再次可见时执行。
- onResume()：Activity 获取焦点时执行。
- onPause()：Activity 失去焦点时执行。
- onStop()：用户不可见，进入后台时执行。
- onDestroy()：Activity 销毁时执行。

为了更好地理解 Activity 的生命周期以及在 Activity 不同状态切换时所调用的方法，接下来将通过 Google 公司提供的一个 Activity 生命周期图来更生动地展示，如图 3-4 所示。

从图 3-4 中可以看出，Activity 在从启动到关闭的过程中，会依次执行 onCreate()→onStart()→onResume()→onPause()→onStop()→onDestroy()方法。如果进程被杀死，则会重新执行 onCreate()方法。

为了更好地掌握 Activity 的生命周期中方法的执行过程，接下来将通过具体的例子来展现方法的执行顺序。

先创建一个布局界面，其中包含一个按钮，用来跳转到另一个 Activity 中使用，布局代码如下：

```
<?xml version = "1.0" encoding = "utf - 8"?>
< LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
     xmlns:app = "http://schemas.android.com/apk/res - auto"
     xmlns:tools = "http://schemas.android.com/tools"
     android:layout_width = "match_parent"
     android:layout_height = "match_parent"
     tools:context = ".MainActivity">
     < Button
          android:id = "@ + id/button"
          android:layout_width = "wrap_content"
          android:layout_height = "wrap_content"
          android:layout_weight = "1"
          android:text = "跳转到第二个 Activity" />
</LinearLayout >
```

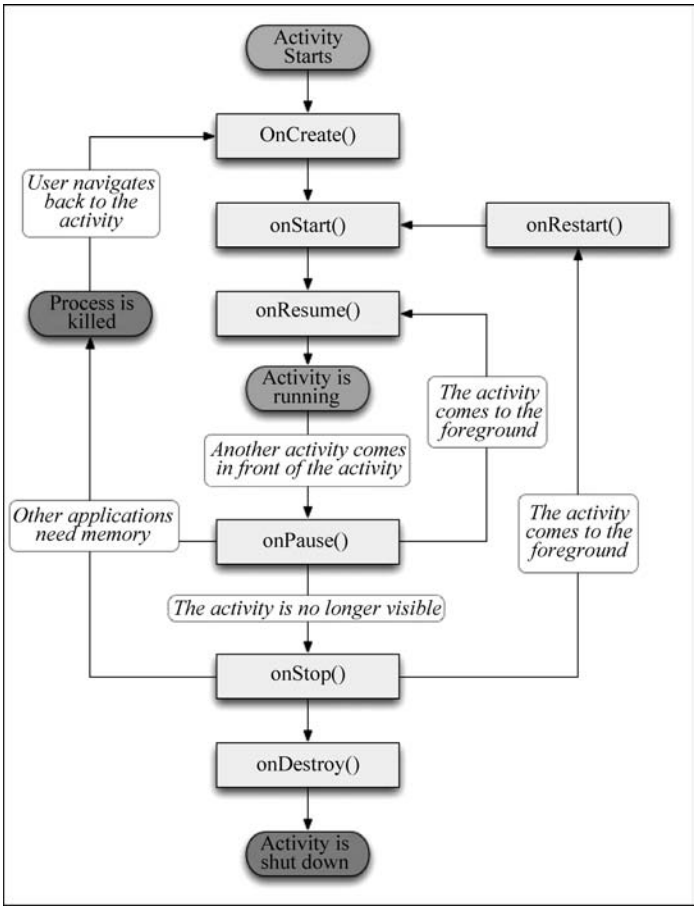

图 3-4　Activity 的生命周期图

第一个 Activity 的代码如下：

```
public class MainActivity extends Activity {
    //Activity1 创建时调用的方法
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.i("Activity1","onCreate()");
        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
           Intent intent = new Intent(MainActivity.this,SecondActivity.class);
            startActivity(intent);
            }});
    }
    //Activity1 可见时调用的方法
    @Override
```

```
    protected void onStart() {
        super.onStart();
        Log.i("Activity1","onStart()");
}
    @Override
    protected void onRestart() {
        super.onRestart();
        Log.i("Activity1","onReStart()");
    }
    //Activity1 获取到焦点时调用的方法
    @Override
    protected void onResume() {
        super.onResume();
        Log.i("Activity1","onResume()");
    }
    //Activity1 失去焦点时调用的方法
    @Override
    protected void onPause() {
        super.onPause();
        Log.i("Activity1","onPause()");
    }
    //Activity1 不可见时调用的方法
    @Override
    protected void onStop() {
        super.onStop();
        Log.i("Activity1","onStop()");
    }
    //Activity1 被销毁时调用的方法
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.i("Activity1","onDestroy()");
    }
}
```

第二个 Activity 中的代码和使用的布局代码与第一个类似。

上面的代码写好以后，在 AndroidManifest.xml 中注册创建 Activity。完成上述步骤以后，运行界面如图 3-5 所示。

图 3-5 Activity1 界面

使用 Log 来打印日志信息，在 Log 窗口打印的 Activity1 生命周期的执行方法顺序如图 3-6 所示。

```
11-20 16:00:45.702 6942-6942/? I/Activity1: onCreate()
11-20 16:00:45.707 6942-6942/? I/Activity1: onStart()
11-20 16:00:45.707 6942-6942/? I/Activity1: onResume()
```

图 3-6 Activity1 的生命周期

从图 3-6 中的日志信息可以看出启动 Activity1 依次执行了 onCreate()、onStart()、onResume()方法，这是 Activity 从创建到可供用户操作的过程。

接下来单击第一个布局界面的按钮跳转到第二个 Activity，出现如图 3-7 所示的界面。

图 3-7　Activity2 界面

与此同时，Log 打印的日志信息如图 3-8 所示。

从图 3-8 中的日志信息可以看出，当跳转到 Activity2 的时候，Activity1 会失去焦点，然后执行 onPause()方法，此时 Activity2 会依次执行 onCreate()、onStart()、onResume()方法。这时 Activity1 会执行 onStop()方法。

```
11-20 16:02:57.284 6942-6942/com.example.chapter3_shengtime I/Activity1: onPause()
11-20 16:02:57.315 6942-6942/com.example.chapter3_shengtime I/Activity2: onCreate()
11-20 16:02:57.321 6942-6942/com.example.chapter3_shengtime I/Activity2: onStart()
11-20 16:02:57.323 6942-6942/com.example.chapter3_shengtime I/Activity2: onResume()
11-20 16:02:57.957 6942-6942/com.example.chapter3_shengtime I/Activity1: onStop()
```

图 3-8　Activity1 跳转到 Activity2 的生命周期

接下来从第二个界面返回到第一个界面，此时 Log 打印的日志信息如图 3-9 所示。

```
11-20 16:04:48.331 6942-6942/com.example.chapter3_shengtime I/Activity2: onPause()
11-20 16:04:48.337 6942-6942/com.example.chapter3_shengtime I/Activity1: onRestart()
11-20 16:04:48.338 6942-6942/com.example.chapter3_shengtime I/Activity1: onStart()
11-20 16:04:48.339 6942-6942/com.example.chapter3_shengtime I/Activity1: onResume()
11-20 16:04:48.879 6942-6942/com.example.chapter3_shengtime I/Activity2: onStop()
11-20 16:04:48.879 6942-6942/com.example.chapter3_shengtime I/Activity2: onDestroy()
```

图 3-9　Activity2 跳转到 Activity1 的生命周期

从图 3-9 中的日志信息可以看出，从 Activity2 再次返回到 Activity1 时，Activity2 会先执行 onPause()方法，然后 Activity1 会依次执行 onRestart()方法、onStart()方法、onResume()方法，随后 Activity2 执行 onStop()方法和 onDestroy()方法。如果退出应用程序，则 Activity1 会执行 onStop()方法，然后执行 onDestroy()方法。


视频讲解

3.1.4　Activity 中的单击事件

3.1.3 节学习了 Activity 中的生命周期，可以看到，从第一个界面到第二个界面使用了按钮的单击事件。在 Android 中 View 的单击事件共有 4 种，接下来详细讲解这 4 种方式。

第一种为在布局文件中设置按钮的 onClick 属性为其指定 Activity 中的方法，代码如下：

```
<!-- 布局文件中添加单击事件为其指定方法名 -->
android:onClick = "click"

Activity 中的方法:
    public void click(View view){
```

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

第二种是创建内部类的方式，创建一个内部类实现 OnClickListener 接口并重写 onClick()方法，在方法中写入单击事件的逻辑。这一种方法不常用，所以不做详细介绍。

第三种就是主类实现 OnClickListener 接口，然后重写 onClick()方法，并通过 switch()语句判断哪个按钮被单击，具体的代码如下：

```
Public class MainActivity extends Activity implements View.OnClickListener {
    register = (Button)findViewById(R.id.register);
    register.setOnClickListener(this);
    @Override
    public void onClick(View view) {
        switch (view.getId()){
            case R.id.register:
            break;
        }
    }
}
```

这里需要注意的是“register.setOnClickListener(this);”方法中这个 this 代表的是该 Activity 的引用，由于 Activity 实现了 OnClickListener 接口，所以这里就代表了 OnClickListener 的引用，在方法中传入 this，就代表该控件绑定了单击事件的接口。

第四种是匿名内部类的方式，适合按钮比较少的情况下使用。这种方式可以直接创建 OnClickListener 的匿名内部类传入按钮的 setOnClickListener()方法和参数中，具体的代码如下：

```
public class MainActivity extends Activity {
    //Activity1 创建时调用的方法
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.i("Activity1","onCreate()");
        Button button = (Button)findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
              Intentintent = new Intent(MainActivity.this,SecondActivity.class);
              startActivity(intent);
           }
        });
    }
```

以上就是单击事件的处理过程,后两种匿名内部类与主类中实现 OnClickListener 接口的方式在平常的 Android 开发中使用较为普遍,所以需要熟练掌握。

3.2 Intent 的使用

3.2.1 Intent 浅析

在 Android 系统中,组件之间的通信需要使用到 Intent。Intent 中文翻译为“意图”,Intent 最常用的是绑定应用程序组件,并在应用程序之间进行通信。它一般用于启动 Activity、服务、发送广播等,承担了 Android 应用程序三大核心组件之间的通信功能。

使用 Intent 开启 Activity 时,对应的方法为 startActivity(Intent intent)和 startActivityForResult(Intent intent,int requestCode);开启 Service 时,常用的有 ComponentName startService(Intent intent)和 boolean bindService(Intent service,ServiceConnection conn,int flags);开启 BroadcastReceiver 方法有多种,就不一一列举了。

Android 中使用 Intent 的方式有两种,分别为显式 Intent 和隐式 Intent,接下来将在 3.2.2 节和 3.2.3 节详细介绍这两种方式。

3.2.2 显式 Intent

显式 Intent 就是在通过 Intent 启动 Activity 时,需要明确指定激活组件的名称,例如通过一个 Activity 启动另外一个 Activity 时,就可以通过这种方式,具体的代码如下:

```
//创建 Intent 对象,指定启动的类名 SecondActivity
Intent intent = new Intent(MainActivity.this,SecondActivity.class);
//启动 Activity
startActivity(intent);
```

通过上述代码可以看出,使用显式 Intent 时,首先需要通过 Intent 的构造方法来创建 Intent 对象。构造方法有两个参数,分别为启动 Activity 的上下文和需要启动的 Activity 类名。除了通过指定类名的方式启动组件外,显式 Intent 还可以根据目标组件的包名、全路径来指定开启的组件。具体的代码如下:

```
//setClassName("包名","类的全路径名称");
intent.setClassName("com.jxust.cn","com.jxust.cn.chapter_shengtime");
//启动 Activity
startActivity(intent);
```

Activity 类提供了 startActivity(Intent intent)方法,该方法专门用于启动 Activity,它接收一个 Intent 参数,然后通过将构建好的 Intent 参数传入方法里来启动 Activity。

使用这两种方式启动 Activity,能够在程序中很清晰地看到,其“意图”很明显,因此称为显式 Intent。

3.2.3 隐式 Intent


视频讲解

在程序中没有明确指定需要启动的 Activity，Android 系统会根据在 Androidmanifest.xml 文件中设置的动作(action)、类别(category)、数据(Uri 和数据类型)来启动合适的组件。具体代码如下：

```
<activity android:name = ".MainActivity">
  <intent-filter>
    <!—设置 action 属性,根据 name 设置的值来指定启动的组件 -->
    <action android:name = "android.intent.action.MAIN" />
    <category android:name = "android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

在上述代码中，<action>标签指定了当前 Activity 可以响应的动作为 android.intent.action.MAIN，而<category>标签则包含了一些类别信息，只有当这两者中的内容同时匹配时，Activity 才会启动。使用隐式 Intent 启动 Activity 的具体代码如下：

```
Intent intent = new Intent();
intent.setAction("android.intent.action.MAIN");
startActivity(intent);
```

通过以上的学习，已经初步了解了显式 Intent 和隐式 Intent 的使用。显式 Intent 启动组件时必须要指定组件的名称，一般只在本应用程序切换组件时使用。而隐式 Intent 使用的范围更广，不仅可以启动本应用程序内的组件，还可以开启其他应用的组件，如打开系统的照相机、图库等。

3.3 Activity 中的数据传递方式

在 Android 开发中，经常需要在 Activity 中进行数据传递，这里就需要使用 3.2 节讲到的 Intent 来实现 Activity 之间数据的传递。

使用 Intent 进行数据传递时只需要调用 putExtra()方法把数据存储进去即可。这个方法有两个参数，是一种“键值对”的形式，第一个参数为 key，第二个参数为 value。实现传递参数的具体代码如下：

```
//定义字符串变量存储一个值
String str = "android";
Intent intent = new Intent(this,SecondActivity.class);
//传递参数
intent.putExtra("receive_str",str);
startActivity(intent);
```

上述代码中将一个字符串变量 str 传递到 SecondActivity 中,然后需要在 SecondActivity 中接收这个参数,具体的代码如下:

```
Intent intent = this.getIntent();
String receive_str = intent.getStringExtra("receive_str");
```

上面就是通过 Intent 传递和接收参数的一种简单方式,如果需要传递的参数比较多,就需要使用 putExtras()方法传递数据,该方法传递的是 Bundle 对象,具体的代码如下:

```
Intent intent = new Intent(this,SecondActivity.class);
Bundle bundle = new Bundle();
bundle.putString("phone","123456");
bundle.putString("sex","男");
bundle.putString("age","18");
intent.putExtras(bundle);
startActivity(intent);
```

上述代码使用 Bundle 对象传递参数,在 SecondActivity 中取出这些参数的具体代码如下:

```
Intent intent = this.getIntent();
Bundle bundle = intent.getExtras();
String phone = bundle.getString("phone");
```

在上述代码中,在接收 Bundle 对象封装的数据时,需要先接收对应的 Bundle 对象,然后再根据 key 取出 value。接下来将在 3.4 节讲解如何使用在布局文件中定义的各个控件以及如何进行数据的传递并显示。


视频讲解

3.4 用户注册案例讲解

本节的用户注册布局与第 2 章的用户注册案例布局是一样的,本节主要讲的是如何使用布局中定义的控件以及数据传递与接收。

(1) Activity1 的布局代码与第 2 章的布局代码一样。

(2) Activity1 代码如下:

```
public class Activity1 extends Activity implements View.OnClickListener,
RadioGroup.OnCheckedChangeListener{
    //定义字符串用来保存各个信息
    private String phone_str = "";
    private String paswd_str = "";
    //默认为"男性"被选中
    private String sex_str = "男性";
    private String hobby_str = "1";
    private String city_str = "";
```

```
//组件定义
EditText phone_edit,paswd_edit;
RadioGroup sex_group;
RadioButton nan_but,nv_but;
CheckBox play,read,music;
Button register;
Spinner spinner;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //组件初始化
    phone_edit = (EditText)findViewById(R.id.phone);
    paswd_edit = (EditText)findViewById(R.id.paswd);
    sex_group = (RadioGroup) findViewById(R.id.sex);
    //添加监听事件
    sex_group.setOnCheckedChangeListener(this);
    nan_but = (RadioButton)findViewById(R.id.nan);
    read = (CheckBox)findViewById(R.id.read_book);
    play = (CheckBox)findViewById(R.id.play_ball);
    music = (CheckBox)findViewById(R.id.music);
    register = (Button)findViewById(R.id.register);
    //添加监听事件
    register.setOnClickListener(this);
    spinner = (Spinner)findViewById(R.id.spinner);
    final String[] city = new String[]{"北京","上海","武汉","南京","南昌","信阳"};
    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_1,city);
    spinner.setAdapter(adapter);
    //城市下拉列表添加监听事件
    spinner.setOnItemSelectedListener(new Spinner.OnItemSelectedListener() {
        @Override
    public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
            city_str = city[i];
        }
        @Override
        public void onNothingSelected(AdapterView<?> adapterView) {
        }
    });
}
@Override
public void onClick(View view) {
    switch (view.getId()){
        case R.id.register:
            //获取手机号和密码
            phone_str = phone_edit.getText().toString();
            paswd_str = paswd_edit.getText().toString();
            //获取兴趣爱好即复选框的值
            hobby_str = "";        //清除上一次已经选中的选项
            if(read.isChecked()){
```

```
                    hobby_str += read.getText().toString();
                }if(play.isChecked()){
                    hobby_str += play.getText().toString();
            }if(music.isChecked()){
                    hobby_str += music.getText().toString();
            }
                    Intent intent = new Intent(this,SecondActivity.class);
                    Bundle bundle = new Bundle();
                    bundle.putString("phone",phone_str);
                    bundle.putString("paswd",paswd_str);
                    bundle.putString("sex",sex_str);
                    bundle.putString("hobby",hobby_str);
                    bundle.putString("city",city_str);
                    intent.putExtras(bundle);
                    startActivity(intent);
                    break;
            }
        }
        @Override
        public void onCheckedChanged(RadioGroup radioGroup, @IdRes int i) {
            //根据用户选择来改变 sex_str 的值
            sex_str = i == R.id.nan?"男性":"女性";
        }
    }
```

上述代码主要是对 Activity1 的布局文件 activity_main.xml 中的控件进行初始化以及添加监听事件,然后对选择的数据进行传递。

(3) SecondActivity 负责接收数据并显示出来。布局文件 second_layout.xml 中定义了一个 TextView 负责显示数据,代码如下:

```
<?xml version = "1.0" encoding = "utf - 8"?>
<LinearLayout xmlns:android = "http://schemas.android.com/apk/res/android"
    android:layout_width = "match_parent"
    android:layout_height = "match_parent">
    <TextView
        android:id = "@ + id/show_content"
        android:layout_width = "wrap_content"
        android:layout_height = "wrap_content"
        android: layout_gravity = "center"
        android:text = "TextView" />
</LinearLayout>
```

(4) SecondActivity 代码如下:

```
public class SecondActivity extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.second_layout);
        Intent intent = this.getIntent();
        Bundle bundle = intent.getExtras();
        String phone = bundle.getString("phone");
        String paswd = bundle.getString("paswd");
        String sex = bundle.getString("sex");
        String hobby = bundle.getString("hobby");
        String city = bundle.getString("city");
        TextView show_text = (TextView)findViewById(R.id.show_content);
show_text.setText("手机号为：" + phone + "\n" + "密码为：" + paswd + "\n" + "性别是：" + sex +
"\n" + "爱好是：" + hobby + "\n" + "城市是：" + city);
    }
}
```

（5）在 AndroidManifest.xml 文件中注册 SecondActivity，代码如下所示：

```
<activity android:name = ".SecondActivity"></activity>
```

上述代码编写完成以后，接下来输入手机号、密码，选择性别、爱好、城市，然后单击“注册”按钮，跳转到接收数据的界面。注册界面和接收数据界面分别如图 3-10 和图 3-11 所示。

图 3-10　注册界面

图 3-11　接收数据界面

以上就是用户注册的详细介绍，其中包含了组件的使用、数据的传递和接收、按钮的单击事件等，这些都是进行 Android 开发的基本知识，需要熟练掌握和应用。

本章小结

本章首先讲解了 Activity 的基本知识，包含从 Activity 的概念、生命周期，到后面的从一个 Activity 跳转到另外一个 Activity 生命周期中方法的执行过程；然后讲解了 Intent 的使用以及数据的传递和接收；最后结合一个用户注册的实例讲解了控件的使用以及监听事

件的处理,这些都需要开发者熟练掌握。

习题

1. 简述一个 Activity 跳转到另一个 Activity 时,两个 Activity 生命周期方法的执行过程。

2. 编写一个程序,要求在第一个界面中输入两个数字,在第二个界面显示第一个界面两个数字的和。