第3章

用户人机界面

3-1 更改与显示文字标签 — TextView 标签的使用

范例说明

前一章写了 Hello World 之后，一直觉得没有写半行代码对不起自己，所以在本章人机界面一开始，则延续 Hello Wolrd 的气势，进行与 TextView 文字标签的第一次接触。在此范例中，将会在 Layout 中创建 TextView 对象，并学会定义 res/values/strings.xml 里的字符串常数，最后通过 TextView 的 setText 方法，在预加载程序之初，更改 TextView 文字。

运行结果

▲图 3-1 认识 TextView.setText 更改默认 Layout 里定义的文本字符串

范例程序

src/irdc.ex03_01/EX03_01.java

主程序示范以 setText 方法，输出 String 类型的字符串变量。

```
package irdc.ex03_01;

import android.app.Activity;
import android.os.Bundle;

/*必须引用 widget.TextView 才能在程序里声明 TextView 对象*/
import android.widget.TextView;

public class EX03_01 extends Activity
{
```

```
  /*必须引用 widget.TextView 才能在程序里声明 TextView 对象*/
  private TextView mTextView01;

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);

    /* 载入 main.xml Layout，此时 myTextView01:text 为 str_1 */
    setContentView(R.layout.main);

    /* 使用 findViewBtId 函数，利用 ID 找到该 TextView 对象 */
    mTextView01 = (TextView) findViewById(R.id.myTextView01);

    String str_2 = "欢迎来到 Android 的 TextView 世界...";
    mTextView01.setText(str_2);
  }
}
```

res/layout/main.xml

以 android:id 命名 TextView 的 ID 为 mTextView01；在较旧的版本写法与 1.0 的不同，请特别留意。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
android:id="@+id/widget35"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
android:id="@+id/myTextView01"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/str_1"
android:layout_x="61px"
android:layout_y="69px"
>
</TextView>
</AbsoluteLayout>
```

扩展学习

TextView 里的 setText 方法支持以下多态构造方法：

```
public final void setText(CharSequence text)
public final void setText(int resid)
public void setText(CharSequence text, TextView.BufferType type)
public final void setText(int resid, TextView.BufferType type)
public final void setText(char[] text, int start, int len)
```

在此，以最后 setText(char[] text, int start, int len) 为例，第一个参数为 char 数组作为输出依据，第二个参数为从哪一个元素索引开始选取，第三个参数则为要取出多少个元素，请看以下的例子：

```
char char_1[] = new char[5];
char_1[0] = 'D';
char_1[1] = 'a';
char_1[2] = 'v';
char_1[3] = 'i';
char_1[4] = 'd';
mTextView01.setText(char_1,1,3);
```

如上述程序所示，输出的结果是“avi”，因为从第 1 个元素索引开始，共取 3 个元素；最后则要提醒你，TextView.setTextView 不支持 HTML TAG 的输出，所以即便写成这样：

```
mTextView01.setText("<a href=\"http://shop.teac.idv.tw/MyBlog/\">戴维的博
客</a>");
```

实际输出时，也就是纯文本而已，并不会作 HTML TAG 的转换。但若撇开 HTML TAG 之外（如“<”开头的标记），在 TextView 里加上了 android:autoLink="all"，那么正文中若有网址（http://），是可以被显示的，以下这个范例就交给你自己实现看看。

```
<TextView  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:autoLink="all"
  android:text="请访问戴维的博客：http://shop.teac.idv.tw/MyBlog/"
/>
```

3-2 更改手机窗口画面底色
— drawable 定义颜色常数的方法

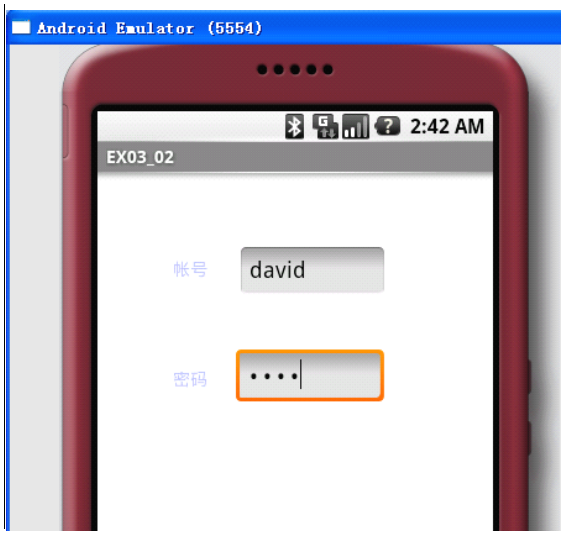
范例说明

在之前的范例运行结果，窗口的底色一律是“深黑色”，这是 SDK 默认的颜色，要更改 Activity 里的窗口底色有许多方法，最简单的方法就是将颜色色码事先定义在 drawable 当中，当程序 onCreate 创建的同时，加载预先定义的画面颜色。

此范例程序的设计方式是在 drawable 里指定 Layout 的背景色（BackGround）为白色，但这里的“白色”（颜色色码为#FFFFFFFF）预先定义在 drawable 当中，当程序运行时，背景就会变为白色。

这是指定 Activity Layout 背景颜色最简单的方法，在范例最末，则将示范如何创建色彩板（color table），让 Android 手机程序可以像使用“常数”般直接取用，并反应在应用程序的运行阶段。

运行结果



▲图 3-2　使用 drawable 设置颜色常数，应用于程序运行时的结果

范例程序

src/irdc.ex03_02/EX03_02.java

程序继承自 Activity 类，并在重写 onCreate 创建之初，直接显示 R.layout.main（main.xml）这个页面安排的布局配置。

```
package irdc.ex03_02;

import android.app.Activity;
import android.os.Bundle;

public class EX03_02 extends Activity
{
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  }
}
```

res/layout/main.xml

在页面布局上，使用了 2 个 TextView 对象，以及 2 个 EditText 对象，关键在于 android:background="@drawable/white" 让程序背景变成了白色，而 android:textColor="@drawable/darkgray" 将 TextView 里的文字颜色（textColor）设为灰色，当中“@drawable/white”及“@drawable/darkgray”的写法则是参考事先于 drawable 里定义好的颜色常数，将在 res/values/color.xml 里看见颜色的定义描述。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
android:id="@+id/widget35"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="@drawable/white"
xmlns:android="http://schemas.android.com/apk/res/android"
>
<TextView
android:id="@+id/widget28"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/str_id"
android:textColor="@drawable/darkgray"
android:layout_x="61px"
android:layout_y="69px"
```

```
>
</TextView>
<TextView
android:id="@+id/widget29"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/str_pwd"
android:textColor="@drawable/darkgray"
android:layout_x="61px"
android:layout_y="158px"
>
</TextView>
<EditText
android:id="@+id/widget31"
android:layout_width="120dip"
android:layout_height="wrap_content"
android:textSize="18sp"
android:layout_x="114px"
android:layout_y="57px"
>
</EditText>
<EditText
android:id="@+id/widget30"
android:layout_width="120dip"
android:layout_height="wrap_content"
android:textSize="18sp"
android:password="true"
android:layout_x="112px"
android:layout_y="142px"
>
</EditText>
</AbsoluteLayout>
```

扩展学习

事先将定义好的颜色代码（color code）以 drawable 的名称（name）存放于 resources 当中，这是开发 Android 程序的好习惯，正如同字符串常数一样，颜色也是可以事先定义好的。在本范例中学会了使用 drawable 的 resource 的定义方法：

```
<drawable name=color_name>color_value</drawable>
```

定义好的 drawable name 常数，必须存放于 res/values 下面，作为资源取用，但定义好的背景颜色并非只能当作是“默认”颜色声明使用，在程序的事件里，是可以通过程序来更改的，如以下程序所示：

```
Resources resources = getBaseContext().getResources();
Drawable HippoDrawable = resources.getDrawable(R.drawable.white);
```

```
TextView tv = (TextView)findViewByID(R.id.text);
tv.setBackground(HippoDrawable);
```

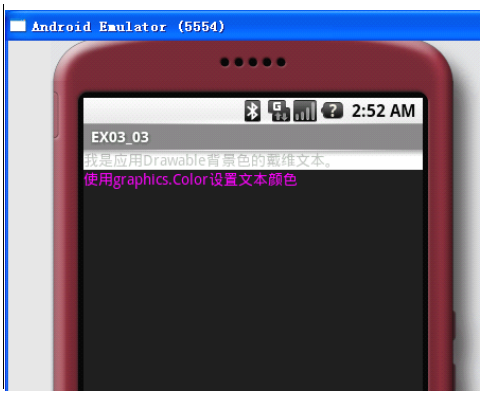
3-3 更改 TextView 文字颜色
— 引用 Drawable 颜色常数及背景色

范例说明

上一个范例通过 Drawable 来定义颜色常数，但实际设计中最常用的方法，则是使用程序控制 TextView 或其它对象的背景色（setBackgroundDrawable 方法），如判断对象被点击时的背景色亮起、当失去焦点时，又恢复成原来的背景色等等。

以下的范例，将扩展前一个范例的实现，预先在 Layout 当中设计好两个 TextView，并在 onCreate 同时，通过两种程序描述方法，实时更改原来 Layout 里 TextView 的背景色以及文字颜色，最后学会使用 Android 默认的颜色常数（graphics.Color）来更改文字的前景色。

运行结果



▲图 3-3 通过 setBackgroundDrawable 方法更改 TextView 的背景色以及 graphics.Color 更改前景色

范例程序

src/lrdc.ex03_03/EX03_03.java

程序里新建两个类成员变量：mTextView01 与 mTextView02，这两个变量在 onCreate 之初，以 findViewById 方法使之初始化为 layout（main.xml）里的 TextView 对象。在当中使用了 Resource 类以及 Drawable 类，分别创建了 resources 对象以及 HippoDrawable 对象，并将前一个范例中所创建的 R.drawable.white 以 getDrawable 方法加载，最后则调用了 setBackgroundDrawable 来更改 mTextView01 的文字底纹。更改 TextView 里的文字，则使用了 setText 方法。

在 mTextView02 当中，使用了 graphics.Color 里的颜色常数，接着，再利用 setTextColor 更改文字的前景色。

```
package irdc.ex03_03;

import android.app.Activity;
import android.content.res.Resources;
import android.graphics.Color;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.widget.TextView;

public class EX03_03 extends Activity
{
  private TextView mTextView01;
  private TextView mTextView02;

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mTextView01 = (TextView) findViewById(R.id.myTextView01);
    mTextView01.setText("我是应用 Drawable 背景色的戴维文本。");

    Resources resources = getBaseContext().getResources();
    Drawable HippoDrawable = resources.getDrawable(R.drawable.white);
    mTextView01.setBackgroundDrawable(HippoDrawable);

    mTextView02 = (TextView) findViewById(R.id.myTextView02);

    /*下使用 Color.MAGENTA 指定文本的颜色为紫红色*/
    mTextView02.setTextColor(Color.MAGENTA);
  }
}
```

扩展学习

```
Resources resources = getBaseContext().getResources();
Drawable HippoDrawable = resources.getDrawable(R.drawable.white);
```

上述这 2 行代码，在前一版本中的写法是这样的：

```
Resources.resources = getDrawable(R.drawable.solid_red);
Drawable HippoDrawable = resources.getDrawable(R.drawable.white);
```

但是在 1.0 之后的版本，Resources 不再支持直接使用 .getDrawable 方法直接取用 drawable，而必须先取得基类的 Context 才行。

此外，在程序里使用了 Color.MAGENTA 让 TextView 里的文字变成了粉红色，事实上，在 Android 里还有以下 12 种常见的颜色：

```
Color.BLACK
Color.BLUE
Color.CYAN
Color.DKGRAY
Color.GRAY
Color.GREEN
Color.LTGRAY
Color.MAGENTA
Color.RED
Color.TRANSPARENT
Color.WHITE
Color.YELLOW
```

这些颜色常数是定义在 android.graphics.Color 里的：

类型	常数	值	色码
int	BLACK	-16777216	0xff000000
int	BLUE	-16776961	0xff0000ff
int	CYAN	-16711681	0xff00ffff
int	DKGRAY	-12303292	0xff444444
int	GRAY	-7829368	0xff888888
int	GREEN	-16711936	0xff00ff00
int	LTGRAY	-3355444	0xffcccccc
int	MAGENTA	-65281	0xffff00ff
int	RED	-65536	0xffff0000
int	TRANSPARENT	0	0x00000000
int	WHITE	-1	0xffffffff
int	YELLOW	-256	0xffffff00

3-4 置换 TextView 文字
— CharSequence 数据类型与 Resource ID 应用

范例说明

从一开始自 Layout 里通过 Resource 初始化 TextView 的文字，到程序中动态更改 TextView 文字，但要如何在代码里取得 Resource 的字符串呢？在 Android 里，确实是有些方法可以直接以 R.string.*直接转换 ID 为 String，不过，这样的数据类型转换是非常规甚至是不妥的，正确的方法是利用 Context.getString 方法来取得存放在 global 里的 Resource ID。以下这个范例将示范如何在程序运行时（runtime），通过 CharSequence 依据 Resource ID 取出字符串，并正确更改 TextView 的文字。

运行结果

▲图 3-4 通过 java.lang.CharSequence 这个 Interface 来取得存放在 global 里的 Resource ID

范例程序

src/irdc.ex03_04/EX03_04.java

主程序的差异主要是在更改 mTextView02 的文字时（setText 方法），合并了 str_3 与 str_2 这两个不同对象，由于 setText 方法同时支持 CharSequence 与 String 类型的参数，故在此示范不同数据类型的字符串进行同步输出。

```
package irdc.ex03_04;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
```

```
public class EX03_04 extends Activity
{
  private TextView mTextView02;

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mTextView02 = (TextView) findViewById(R.id.myTextView02);
    CharSequence str_2 = getString(R.string.str_2);

    String str_3 = "我是程序里调用 Resource 的";
    mTextView02.setText(str_3 + str_2);
  }
}
```

res/layout/main.xml

为了作为对比，在 main.xml 里创建了两个 TextView，并采 LinearLayout 的方式配置，一上一下，在运行结果中 id 为 myTextView01 的 TextView 并没有任何文字的更改，维持一开始的 str_1（参考字符串常数里的文字），但在程序运行后，id 为 myTextView02 的 TextView 则作了文字的实时更改。

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:background="@drawable/white"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  >
  <TextView
  android:id="@+id/myTextView01"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/str_1"
  android:layout_x="30px"
  android:layout_y="50px"
  >
  </TextView>
  <TextView
  android:id="@+id/myTextView02"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:text="@string/str_2"
```

```
    android:layout_x="30px"
    android:layout_y="70px"
    >
    </TextView>
</LinearLayout>
```

扩展学习

虽然在 values/strings.xml 里定义了默认的字符串常数，需留意若遭遇如“?”、“'”、“\”等符号时，必须使用转义字符（\），如下：

```
\?
\'
\\
```

3-5 取得手机屏幕大小
— DisplayMetrics 取得画面宽高的方法

范例说明

在开发手机应用程序时，除了底层对 API 的掌握度之外，最重要的仍是对屏幕分辨率的概念，因各家手机厂商所采用的屏幕尺寸不同，user UI 接口呈现及布局自然也各异。

尽管 Android 可设置为随着窗口大小调整缩放比例，但即便如此，手机程序设计人员还是必须知道手机屏幕的边界，以避免缩放造成的布局（Layout）变形问题。这个范例非常的简短，只需几行程序即可取得手机的分辨率，当中的关键则是 DisplayMetrics 类的应用。

运行结果

▲图 3-5　取得 Android 手机的实际屏幕分辨率

范例程序

src/irdc.ex03_05/EX03_05.java

在 android.util 底下的 DisplayMetrics 对象，记录了一些常用的信息，包含了显示信息、大小、维度、字体等等；在使用时，请记得引用 android.util.DisplayMetrics。值得一提的是 DisplayMetrics 对象里的 widthPixels 及 heightPixels 字段为整数类型，在以下的程序当中，并没有对其作字符串类型的转换，因为字符串连接运算符的缘故，所以输出 strOpt 为字符串。

```
package irdc.ex03_05;

import android.app.Activity;
import android.os.Bundle;
import android.util.DisplayMetrics;
import android.widget.TextView;

public class EX03_05 extends Activity
{
  private TextView mTextView01;

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    /* 必须引用 android.util.DisplayMetrics */
    DisplayMetrics dm = new DisplayMetrics();
```

```
        getWindowManager().getDefaultDisplay().getMetrics(dm);

        String strOpt = "手机屏幕分辨率为: " +
                dm.widthPixels + " × " + dm.heightPixels;

        mTextView01 = (TextView) findViewById(R.id.myTextView01);
        mTextView01.setText(strOpt);
    }
}
```

扩展学习

程序一开始所创建的DisplayMetrics 对象（程序中的dm），不需要传递任何参数（构造时），但在调用 getWindowManager()之后，会取得现有的 Activity 的 Handler，此时，调用 getDefaultDisplay 方法将取得的宽高维度存放于 DisplayMetrics 对象 dm 中，而取得的宽高维度是以像素为单位（Pixel），“像素”所指的是“绝对像素”而非“相对像素”。

3-6 样式化的定型对象 — Style 样式的定义

范例说明

老是要一个个指定文字的大小、颜色也不是办法，有没有类似 CSS 样式的方法可用来指定颜色、大小呢？事实上是有的，在Android 程序开发过程中，也可以通过样式（Style）的方式，初始化 TextView 的文本颜色、大小；当然这个范例只是抛砖引玉，在 Layout 当中的任何对象（以 XML 定义）都可以用样式化的方式来更改其外观。

在以下的范例中，将创建两个 TextView 作为对比，使其呈现两种不同的样式差异作为练习，而 Style 的写法与先前介绍到的颜色常数（color.xml）相同，同样是定义在 res/values 下面，但其 XML 定义的方式不同来看看以下这个范例练习。

运行结果

▲图 3-6　利用 Style 来初始化 TextView

范例程序

src/irdc.ex03_06/EX03_06.java

主程序看起来非常干净，只有加载 R.layout.main 定义布局内容而已，但由于定义在 main.xml 里的语句不同，自然也有不同的样貌呈现。

```
package irdc.ex03_06;

import android.app.Activity;
import android.os.Bundle;

public class EX03_06 extends Activity
{
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
  }
}
```

res/layout/main.xml

诚如先前所述，初始化 TextView 时，指定 Style 属性，使其应用 style.xml 里事先定义好的样式。

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

```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:background="@drawable/white"
  android:orientation="vertical"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  >
  <!-- 应用样式 1 的 TextView -->
  <TextView
  style="@style/DavidStyleText1"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:gravity="center_vertical|center_horizontal"
  android:text="@string/str_text_view1"
  />
  <!-- 应用样式 2 的 TextView -->
  <TextView
  style="@style/DavidStyleText2"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:gravity="center_vertical|center_horizontal"
  android:text="@string/str_text_view2"
  />
</LinearLayout>
```

res/values/style.xml

在此的 style.xml 就是这个范例的关键之处了，当中定义了两个样式名称，分别为 DavidStyleText1 与 DavidStyleText2；留意于<style>TAG 里以<item>描述的属性方式，与先前介绍 Drawable name 的描述类似。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="DavidStyleText1">
    <item name="android:textSize">18sp</item>
    <item name="android:textColor">#EC9237</item>
  </style>
  <style name="DavidStyleText2">
    <item name="android:textSize">14sp</item>
    <item name="android:textColor">#FF7F7C</item>
    <item name="android:fromAlpha">0.0</item>
    <item name="android:toAlpha">0.0</item>
  </style>
</resources>
```

扩展学习

style 与 color 的 XML 语法相类似，皆需要先声明 xml 的版本以及 encoding 为 UTF-8，但其内的 resources 则需要以 stylename 作为样式名称，在最内层才是以 item 定义样式的范围，其语法如下：

```
<style name=string [parent=string] >
  <item name=string>Hex value | string value | reference</item>+
</style>
```

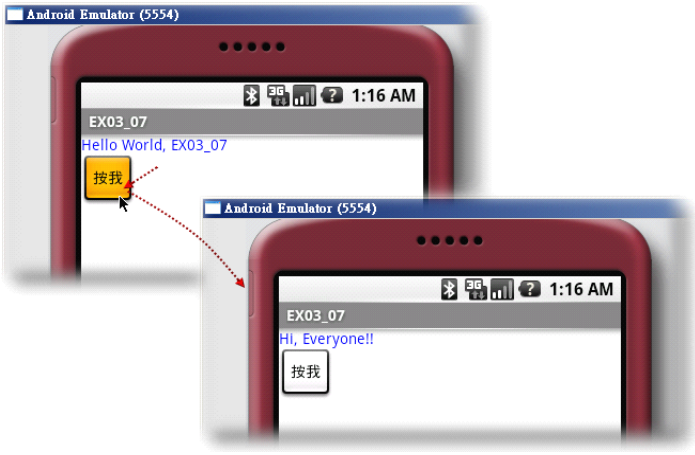
3-7 简易的按钮事件
— Button 事件处理

范例说明

按钮在许多 Windows 窗口应用程序中，是最常见到的控件（Controls），此控件也常在网页设计里出现，诸如网页注册窗体、应用程序里的“确定”等等。

而按钮所触发的事件处理，我们称为 Event Handler，只不过在 Android 当中，按钮事件是由系统的 Button.OnClickListener 所控制，熟悉 Java 程序设计的读者对 OnXxxListener 应该不陌生。以下的范例将示范如何在 Activity 里布局一个按钮（Button），并设计这个按钮的事件处理函数，当点击按钮的同时，更改 TextView 里的文字。

运行结果



▲图 3-7 Android 手机的实际画面运行结果

范例程序

src/irdc.ex03_07/Ex03_07.java

一开始，必须先在 Layout 当中布局一个 Button 及一个 TextView 对象，找不到这两个组件的话，系统会无法运行下去，在开发阶段会造成编译错误。

其次在主程序中，请留意 onCreate 里创建的 Button.OnClickListener 事件，这也是触发按钮时会运行的程序段落；但由于 Eclipse 无法自动加载默认的传递参数（new Button.OnClickListener()），所以，在编写程序描述时，必须自行键入新创建的按钮所需的 OnClickListener() 事件，如下所示：

```
package irdc.ex03_07;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class EX03_07 extends Activity
{
  private Button mButton1;
  private TextView mTextView1;

  /** Called when the activity is first created. */
```

```
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mButton1 =(Button) findViewById(R.id.myButton1);
    mTextView1 = (TextView) findViewById(R.id.myTextView1);

    mButton1.setOnClickListener(new Button.OnClickListener()
    {
      @Override
      public void onClick(View v)
      {
        // TODO Auto-generated method stub
        mTextView1.setText("Hi, Everyone!!");
      }
    });
  }
}
```

扩展学习

本范例中只有一个按钮，但在 Activity 里，其实可以布局数个按钮，只需要在 Layout 里多配置一个按钮对象，以下的程序将创建两个按钮事件作为示范：

```
package irdc.ex03_07;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class EX03_07 extends Activity
{
  private Button mButton1;
  private Button mButton2;
  private TextView mTextView1;

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    mButton1 =(Button) findViewById(R.id.myButton1);
```

```
        mButton2 =(Button) findViewById(R.id.myButton2);
        mTextView1 = (TextView) findViewById(R.id.myTextView1);

        mButton1.setOnClickListener(new Button.OnClickListener()
        {
          @Override
          public void onClick(View v)
          {
            // TODO Auto-generated method stub
            mTextView1.setText("Hi, Everyone!!");
          }
        });

        mButton2.setOnClickListener(new Button.OnClickListener()
        {
          @Override
          public void onClick(View v)
          {
            // TODO Auto-generated method stub
            mTextView1.setText("Hi, David!!");
          }
        });
      }
    }
```

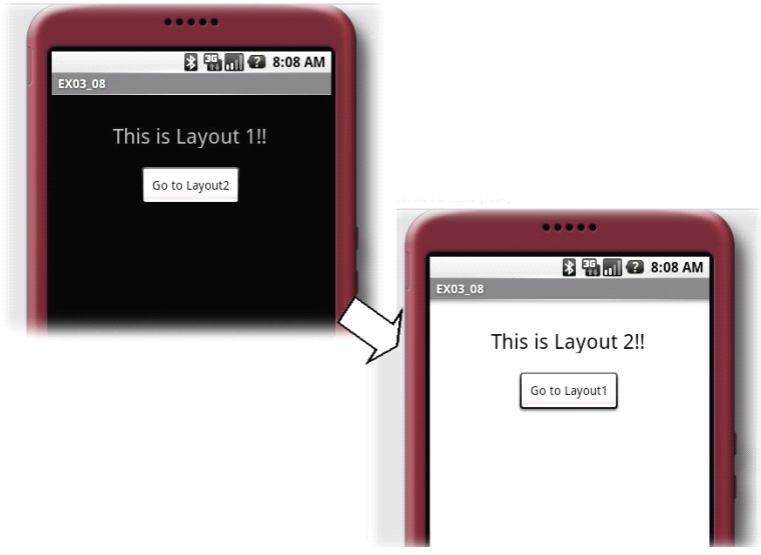
最后来谈谈按钮事件里被重写的 onClick(View v) 函数，此函数唯一的参数是 View 类型的变量 v，这个变量所指的是来自父层（parent）的 ContentView，亦即通过“v.*”可以更改其父层的 View 状态或属性。

3-8 手机页面的转换
— setContentView 的应用

范例说明

在网页的世界里，想要在两个网页间做转换，只要利用超链接（HyperLink）就可以实现，但在手机的世界里，要如何实现手机页面之间的转换呢？最简单的方式就是改变 Activity 的 Layout！在这个范例里头，将布局两个 Layout，分别为 Layout1（main.xml）与 Layout2（mylayout.xml），默认载入的 Layout 为 main.xml，且在 Layout1 当中创建一个按钮，当点击按钮时，显示第二个 Layout（mylayout.xml）；同样地，在 Layout2 里也设计一个按钮，当点击第二个 Layout 的按钮之后，则显示回原来的 Layout1，现在就来示范如何在两个页面之间互相切换。

运行结果



▲图 3-8 手机页面 Layout 间的切换

范例程序

src/irdc.ex03_08/EX03_08.java

主程序中，预加载的 Layout 是 main.xml，屏幕上显示的是黑色背景的“This is Layout 1!!”，在第一个 Layout 上的按钮被点击的同时，改变 Activity 的 Layout 为 mylayout.xml，屏幕上显示变为白色背景的“This is Layout 2!!”，并利用 Button 点击时，调用方法的不同做两个 Layout 间的切换。

```
package irdc.ex03_08;

/* import 相关 class */
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

public class EX03_08 extends Activity
{
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
```

```
    super.onCreate(savedInstanceState);
    /* 载入 main.xml Layout */
    setContentView(R.layout.main);

    /* 以 findViewById()取得 Button 对象，并添加 onClickListener */
    Button b1 = (Button) findViewById(R.id.button1);
    b1.setOnClickListener(new Button.OnClickListener()
    {
      public void onClick(View v)
      {
        jumpToLayout2();
      }
    });
  }

  /* method jumpToLayout2: 将 layout 由 main.xml 切换成 mylayout.xml */
  public void jumpToLayout2()
  {
    /* 将 layout 改成 mylayout.xml */
    setContentView(R.layout.mylayout);

    /* 以 findViewById()取得 Button 对象，并添加 onClickListener */
    Button b2 = (Button) findViewById(R.id.button2);
    b2.setOnClickListener(new Button.OnClickListener()
    {
      public void onClick(View v)
      {
        jumpToLayout1();
      }
    });
  }

  /* method jumpToLayout1: 将 layout 由 mylayout.xml 切换成 main.xml */
  public void jumpToLayout1()
  {
    /* 将 layout 改成 main.xml */
    setContentView(R.layout.main);

    /* 以 findViewById()取得 Button 对象，并添加 onClickListener */
    Button b1 = (Button) findViewById(R.id.button1);
    b1.setOnClickListener(new Button.OnClickListener()
    {
      public void onClick(View v)
    {
        jumpToLayout2();
      }
    });
  }
}
```

res/layout/main.xml

为了凸显 Layout 间切换的效果，特别改变两个 Layout 的背景色及输出文字。在 main.xml 中定义其背景色为黑色，输出文字为“This is Layout 1!!”。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:background="@drawable/black"
  xmlns:android="http://schemas.android.com/apk/res/android"
>
  <TextView
    android:id="@+id/text1"
    android:textSize="24sp"
    android:layout_width="186px"
    android:layout_height="29px"
    android:layout_x="70px"
    android:layout_y="32px"
    android:text="@string/layout1"
  >
  </TextView>
  <Button
    android:id="@+id/button1"
    android:layout_width="118px"
    android:layout_height="wrap_content"
    android:layout_x="100px"
    android:layout_y="82px"
    android:text="Go to Layout2"
  >
  </Button>
</AbsoluteLayout>
```

res/layout/mylayout.xml

在 mylayout.xml 中定义其背景色为白色，输出文字为“This is Layout 2!!”。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:background="@drawable/white"
  xmlns:android="http://schemas.android.com/apk/res/android"
>
  <TextView
    android:id="@+id/text2"
    android:textSize="24sp"
    android:layout_width="186px"
```

```
    android:layout_height="29px"
    android:layout_x="70px"
    android:layout_y="32px"
    android:textColor="@drawable/black"
    android:text="@string/layout2"
    >
  </TextView>
  <Button
    android:id="@+id/button2"
    android:layout_width="118px"
    android:layout_height="wrap_content"
    android:layout_x="100px"
    android:layout_y="82px"
    android:text="Go to Layout1"
  >
  </Button>
</AbsoluteLayout>
```

扩展学习

运用改变 Activity Layout 这个技巧，就可做出手机页面转换的效果，当然亦可搭配之前介绍过的 Style 与 Theme 的设置，进行更加灵活的布局配置运用，例如，让用户自行决定要使用的系统样式、背景及文字颜色等，接着直接应用来改变布局。

再者，利用 setContentView 来置换页面，还有一个特别的优点，即所有程序里的变量皆存在相同的状态，无论是类成员变量、类函数等等，皆可以在一个 Activity 的状态中直接取得，并没有参数传递的问题。打个比喻：Layout1 收集了用户输入的信用卡卡号等付款信息，在“下一步”显示 Layout2 使之显示订单信息，让用户进行确认，并在点击按钮后，调用 Layout3 进行刷卡 Gateway 的授权操作，这当中皆没有需要传递的变量，其手法是将所需要的字段数据，以类成员变量作如下声明：

```
public class EX03_08 extends Activity
{
  public String colVar1;
  public String colVar2;
  public String colVar3;

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    /* 下面的程序略 */
```

3-9 调用另一个 Activity
— Intent 对象的使用

范例说明

前一个范例介绍了如何运用切换 Layout 的方式，进行手机页面间的转换。如果要转换的页面并不单只是背景、颜色或文字内容的不同，而是Activity 的置换，那就不是单单改变Layout 就能完成的，尤其是需要传递的变量不像网页可以通过 Cookie 或 Session，在程序里要移交主控权到另外一个 Activity，光靠先前的 Layout 技巧是办不到的。

那要如何解决 Activity 控制权的移交呢？在 Android 的程序设计中，可在主程序里使用 startActivity() 这个方法来调用另一个 Activity（主程序本身即是一个 Activity），但当中的关键并不在 startActivity 这个方法，而是 Intent 这个特有的对象，Intent 就如同其英文字义，是“想要”或“意图”之意，在主 Activity 当中，告诉程序自己是什么，并想要前往哪里，这就是 Intent 对象所处理的事了。本范例并没有特别的 Layout 布局，而是直接在主 Activity（Activity1）当中部署一个按钮，当点击按钮的同时，告诉主 Activity 前往 Activity2，并在 Activity2 里创建一个回到 Activity1 的按钮，本范例将利用此简易的程序描述，示范如何在一个 Activity 中调用另一个 Activity 的手法。

运行结果

▲图 3-9　在两个 Activity 间做切换

范例程序

src/irdc.ex03_09/EX03_09.java

主程序中加载的 Layout 为 main.xml，屏幕上显示的是黑色背景的“This is Activity 1!!”，在 Button 被点击时调用另一个 Activity（EX03_09_1，参考 AndroidManifest.xml 里的说明），并将主 Activity 关闭 finish()，接着将主控权交给下一个 Activity，即 Activity2。

```
package irdc.ex03_09;

/* import 相关 class */
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.content.Intent;

public class EX03_09 extends Activity
{
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    /* 载入 main.xml Layout */
    setContentView(R.layout.main);

    /* 以 findViewById()取得 Button 对象，并添加 onClickListener */
    Button b1 = (Button) findViewById(R.id.button1);
    b1.setOnClickListener(new Button.OnClickListener()
    {
      public void onClick(View v)
      {
        /* new 一个 Intent 对象，并指定要启动的 class */
        Intent intent = new Intent();
        intent.setClass(EX03_09.this, EX03_09_1.class);

        /* 调用一个新的 Activity */
        startActivity(intent);
        /* 关闭原本的 Activity */
        EX03_09.this.finish();
      }
    });
  }
}
```

src/irdc.ex03_09/EX03_09_1.java

EX03_09_1.java 程序是第二个 Activity 的主程序，其加载的 Layout 为 mylayout.xml，屏幕上所显示的是白色背景的“This is Activity 2!!”，当主 Activity（Activity1）调用这个 Activity（Activity2）后，同样为 Button 添加 onClickListener()，使 Button 被点击时，重新调用 Activity1（EX03_09），并将 Activity2（EX03_09_1）关闭（finish()）。

```
package irdc.ex03_09;

/* import 相关 class */
import android.app.Activity;
import android.os.Bundle;
import android.content.Intent;
import android.view.View;
import android.widget.Button;

public class EX03_09_1 extends Activity
{
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    /* 载入 mylayout.xml Layout */
    setContentView(R.layout.mylayout);

    /* 以 findViewById()取得 Button 对象，并添加 onClickListener */
    Button b2 = (Button) findViewById(R.id.button2);
    b2.setOnClickListener(new Button.OnClickListener()
    {
      public void onClick(View v)
      {
        /* new一个 Intent 对象，并指定要启动的 class */
        Intent intent = new Intent();
        intent.setClass(EX03_09_1.this, EX03_09.class);

        /* 调用一个新的 Activity */
        startActivity(intent);
        /* 关闭原本的 Activity */
        EX03_09_1.this.finish();
      }
    });

  }
}
```

res/layout/main.xml

为了凸显Activity间切换的效果，特别将两个Layout的背景及输出文字有所区别。在main.xml中定义其背景色为黑色，输出文字为“This is Activity 1!!”。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:background="@drawable/black"
  xmlns:android="http://schemas.android.com/apk/res/android"
>
  <TextView
    android:id="@+id/text1"
    android:textSize="24sp"
    android:layout_width="186px"
    android:layout_height="29px"
    android:layout_x="70px"
    android:layout_y="32px"
    android:text="@string/act1"
  >
  </TextView>
  <Button
    android:id="@+id/button1"
    android:layout_width="118px"
    android:layout_height="wrap_content"
    android:layout_x="100px"
    android:layout_y="82px"
    android:text="Go to Activity2"
  >
  </Button>
</AbsoluteLayout>
```

res/layout/mylayout.xml

在 mylayout.xml 中定义其背景色为白色，输出文字为“This is Activity 2!!”。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:background="@drawable/white"
  xmlns:android="http://schemas.android.com/apk/res/android"
>
  <TextView
    android:id="@+id/text2"
    android:textSize="24sp"
    android:layout_width="186px"
    android:layout_height="29px"
```

```
    android:layout_x="70px"
    android:layout_y="32px"
    android:textColor="@drawable/black"
    android:text="@string/act2"
  >
  </TextView>
  <Button
    android:id="@+id/button2"
    android:layout_width="118px"
    android:layout_height="wrap_content"
    android:layout_x="100px"
    android:layout_y="82px"
    android:text="Go to Activity1"
  >
  </Button>
</AbsoluteLayout>
```

AndroidManifest.xml

由于本范例中添加了一个 Activity，所以必须在 AndroidManifest.xml 中定义一个新的 activity，并给予名称 name，否则程序将无法编译运行。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="irdc.ex03_09"
  android:versionCode="1"
  android:versionName="1.0.0">
  <application
    android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity
      android:name=".EX03_09"
      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="EX03_09_1"></activity>
  </application>
</manifest>
```

扩展学习

当系统中新添加 Activity 时，必须在 AndroidManifest.xml 里定义一个新的 activity：

```
<activity android:name="EX03_09_1"></activity>
```

否则系统将会因为找不到 Activity 而发生编译错误。

另外，当程序中出现两个以上的 Activity 时，系统要如何决定主程序是哪一支（entry point）呢？以本范例来说，AndroidManifest.xml 中 Activity EX03_09 的定义如下：

```
<activity android:name=".EX03_09"
  android:label="@string/app_name">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
</activity>
```

其中有一行为 <category android:name="android.intent.category.LAUNCHER" />，这就代表程序启动时，会先运行 EX03_09 这个 Activity，而非 EX03_09_1。需注意的是，这个参数必须要被定义，如果 xml 中没有一支 Activity 有设置这个参数，则程序将不会被运行。

此外，在两支 Java 程序中的最后一行都调用了 finish() 这个方法，它代表这个 Activity 已运作完毕，当系统接收到这个命令时，即会关闭此 Activity，所以此时点击模拟器的返回（Back）键，并不会回到上一个 Activity 的画面，如果要让模拟器的返回键有回上一页的效果，可以将此行程序注释掉。同理，当两个 Activity 在切换时，并非真的只有两个 Activity 在切换，而是在点击按钮时，再重新调用起一个新的 Activity。

3-10 不同 Activity 之间的数据传递 — Bundle 对象的实现

范例说明

在上一个范例里，介绍了如何在 Activity 中调用另一个 Activity，但若需要在调用另外一个 Activity 的同时传递数据，那么就需要利用 android.os.Bundle 对象封装数据的能力，将欲传递的数据或参数，通过 Bundle 来传递不同 Intent 之间的数据。

本范例的设计为一个简易表单的范例，在 Activity1 中收集 User 输入的数据，在离开 Activity1 的同时，将 User 选择的结果传递至下一个 Activity2，以一个简单 BMI “标准体重计算器”示范如何传递数据到下一个 Activity 里。

运行结果

▲图 3-10　在两个 Activity 间做数据的传递

范例程序

src/irdc.ex03_10/EX03_10.java

在第一个 Activity1 主程序中，定义了“性别”选项的 RadioGroup 以及输入身高的“EditText”，并运用 Intent 及 Bundle 对象，在调用 Activity2（EX03_10_1）时，同时将数据传入。关于 EditText 对象的使用在此仅供参考，详细的应用以及属性方法，将会在未来讨论控件时，再详细解说。

```
package irdc.ex03_10;

/* import 相关 class */
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;

public class EX03_10 extends Activity
{
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    /* 载入 main.xml Layout */
    setContentView(R.layout.main);
```

```
    /* 以 findViewById()取得 Button 对象，并添加 onClickListener */
    Button b1 = (Button) findViewById(R.id.button1);
    b1.setOnClickListener(new Button.OnClickListener()
    {
      public void onClick(View v)
      {
        /*取得输入的身高*/
        EditText et = (EditText) findViewById(R.id.height);
        double height=Double.parseDouble(et.getText().toString());
        /*取得选择的性别*/
        String sex="";
        RadioButton rb1 = (RadioButton) findViewById(R.id.sex1);
        if(rb1.isChecked())
        {
          sex="M";
        }
        else
        {
          sex="F";
        }
        /*new一个 Intent 对象，并指定 class*/
        Intent intent = new Intent();
        intent.setClass(EX03_10.this,EX03_10_1.class);

        /*new一个 Bundle对象，并将要传递的数据传入*/
        Bundle bundle = new Bundle();
        bundle.putDouble("height",height);
        bundle.putString("sex",sex);

        /*将 Bundle 对象 assign 给 Intent*/
        intent.putExtras(bundle);

        /*调用 Activity EX03_10_1*/
        startActivity(intent);
      }
    });
  }
}
```

src/irdc.ex03_10/EX03_10_1.java

那么，在 Activity2（EX03_10_1）要如何接收来自 Activity1（EX03_10）传递来的数据呢？试想，在 Activity1 是以 Bundle 封装对象，自然在 Activity2 亦是以 Bundle 的方式解开封装的数据；程序中以 getIntent().getExtras() 方法取得随着 Bundle 对象传递过来的性别与身高，经过计算之后，显示在屏幕上。

```
package irdc.ex03_10;
```

```
/* import 相关 class */
import java.text.DecimalFormat;
import java.text.NumberFormat;
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class EX03_10_1 extends Activity
{
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    /* 加载 main.xml Layout */
    setContentView(R.layout.myalyout);

    /* 取得 Intent 中的 Bundle 对象 */
    Bundle bunde = this.getIntent().getExtras();

    /* 取得 Bundle 对象中的数据 */
    String sex = bunde.getString("sex");
    double height = bunde.getDouble("height");

    /* 判断性别 */
    String sexText="";
    if(sex.equals("M"))
    {
      sexText="男性";
    }
    else
    {
      sexText="女性";
    }

    /* 取得标准体重 */
    String weight=this.getWeight(sex, height);

    /* 设置输出文字 */
    TextView tv1=(TextView) findViewById(R.id.text1);
    tv1.setText("你是一位"+sexText+"\n 你的身高是"
                +height+"厘米\n 你的标准体重是"+weight+"公斤");
  }

  /* 四舍五入的 method */
  private String format(double num)
  {
    NumberFormat formatter = new DecimalFormat("0.00");
```

```
    String s=formatter.format(num);
    return s;
  }

  /* 以 findViewById()取得 Button 对象，并添加 onClickListener */
  private String getWeight(String sex,double height)
  {
    String weight="";
    if(sex.equals("M"))
    {
      weight=format((height-80)*0.7);
    }
    else
    {
      weight=format((height-70)*0.6);
    }
    return weight;
  }
}
```

res/layout/mylayout.xml

mylayout.xml 为（EX03_10_1）的Layout，定义了显示计算结果的 TextView。

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  xmlns:android="http://schemas.android.com/apk/res/android"
>
  <TextView
    android:id="@+id/text1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_x="50px"
    android:layout_y="72px"
  >
  </TextView>
</AbsoluteLayout>
```

AndroidManifest.xml

由于本范例中有两个Activity，所以文件中必须有两个 activity 的声明，否则系统将无法运行，请看以下的描述。

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

```
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="irdc.ex03_10"
  android:versionCode="1"
  android:versionName="1.0.0">
  <application
    android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity
      android:name=".EX03_10"
      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="EX03_10_1"></activity>
  </application>
</manifest>
```

扩展学习

Bundle 对象针对了不同的数据类型提供了许多的方法，例如，此范例中传递 String 类型的数据，使用的方法为 Bundle.putString(stringName,stringValue)：

```
bundle.putDouble("sex",sex);
```

而要传递 Double 类型的数据，使用的方法为 Bundle.putDouble(doubleName,doubleValue)，如下：

```
bundle.putString("height",height);
```

反之，若要由 Bundle 对象中取出数据，则使用 Bundle.getString(stringName)、Bundle.getDouble(doubleName) 等相对应的方法即可。

除了上述简单的传递类型之外，尚有 String[] 与 ArrayList<String> 等封装的方式可供使用参考。

3-11 返回数据到前一个 Activity — startActivityForResult 方法

范例说明

上一个范例中，好不容易将数据从 Activity1 传递至 Activity2，如果要再回到 Activity1，数据该不会要再封装一次吧？而且前一个 Activity1 早就被程序 destroy 了，倘若在 Activity1 最后以 finish() 结束程序，再通过 Activity2 将数据采用 Bundle 的方式通过新打开 Activity1 传递参数，这样的做法虽然也可以恢复 User 输入的数据，但是并不符合我们的期待，尤其是User曾经输入过的数据，如果不小心点击回到上一页，数据就消失不见，这就不妙了。

有鉴于科技始终来自于人性，如果要在次页面加上一个“回上页”的按钮，而非通过模拟器的回复键，且回上页后又能保留之前输入的相关信息，那么就必须使用 startActivityForResult()来唤起一个 Activity。利用这个方法，前一个 Activity1 便会有一个等待次 Activity2 的返回，而返回的数据就可以达到我们想要的结果。

运行结果

图 3-11 将数据返回到前一个 Activity

范例程序

src/irdc.ex03_11/EX03_11.java

在 Activity1 主程序中调用 Activity 的方法更改成 startActivityForResult(intent,0)，其中 0 为下一个 Activity 要返回值的依据，可指定为自行定义的参考标识符（Identifier）。程序覆盖了 onActivityResult() 这个方法，令程序在收到 result 后，再重新加载写回原本输入的值。

```
package irdc.ex03_11;

/* import 相关 class */
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.RadioButton;

public class EX03_11 extends Activity
{
  private EditText et;
  private RadioButton rb1;
  private RadioButton rb2;

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    /* 载入 main.xml Layout */
    setContentView(R.layout.main);

    /* 以 findViewById()取得 Button 对象，并添加 onClickListener */
    Button b1 = (Button) findViewById(R.id.button1);
    b1.setOnClickListener(new Button.OnClickListener()
    {
      public void onClick(View v)
      {
        /*取得输入的身高*/
        et = (EditText) findViewById(R.id.height);
        double height=Double.parseDouble(et.getText().toString());
        /*取得选择的性别*/
        String sex="";
        rb1 = (RadioButton) findViewById(R.id.sex1);
        rb2 = (RadioButton) findViewById(R.id.sex2);
        if(rb1.isChecked())
        {
          sex="M";
        }
        else
        {
          sex="F";
        }

        /*new 一个 Intent 对象，并指定 class*/
        Intent intent = new Intent();
```

```
        intent.setClass(EX03_11.this,EX03_11_1.class);

        /*new 一个 Bundle对象，并将要传递的数 据传入 */
        Bundle bundle = new Bundle();
        bundle.putDouble("height",height);
        bundle.putString("sex",sex);

        /*将 Bundle 对象 assign 给 Intent*/
        intent.putExtras(bundle);

        /*调用 Activity EX03_11_1*/
        startActivityForResult(intent,0);
      }
    });
  }

  /* 覆盖 onActivityResult()*/
  @Override
  protected void onActivityResult(int requestCode, int resultCode,
                                  Intent data)
  {
    switch (resultCode)
    {
      case RESULT_OK:
        /* 取得来自 Activity2 的数据，并显示于画面上 */
        Bundle bunde = data.getExtras();
        String sex = bunde.getString("sex");
        double height = bunde.getDouble("height");

        et.setText(""+height);
        if(sex.equals("M"))
        {
          rb1.setChecked(true);
        }
        else
        {
          rb2.setChecked(true);
        }
        break;
      default:
        break;
    }
  }
}
```

src/irdc.ex03_11/EX03_11_1.java

在 Activity2 的主程序中，设计当 Button 被点击时，将 Bundle 对象与结果返回给前一个 Activity1。

```
package irdc.ex03_11;

/* import 相关 class */
import java.text.DecimalFormat;
import java.text.NumberFormat;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class EX03_11_1 extends Activity
{
  Bundle bunde;
  Intent intent;
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState)
  {
    super.onCreate(savedInstanceState);
    /* 载入 mylayout.xml Layout */
    setContentView(R.layout.myalyout);

    /* 取得 Intent 中的 Bundle 对象 */
    intent=this.getIntent();
    bunde = intent.getExtras();

    /* 取得 Bundle 对象中的数据 */
    String sex = bunde.getString("sex");
    double height = bunde.getDouble("height");

    /* 判断性别 */
    String sexText="";
    if(sex.equals("M"))
    {
      sexText="男性";
    }
    else
    {
      sexText="女性";
    }

    /* 取得标准体重 */
    String weight=this.getWeight(sex, height);

    /* 设置输出文字 */
    TextView tv1=(TextView) findViewById(R.id.text1);
    tv1.setText("你是一位 "+sexText+"\n 你的身高是 "+height+
```

```
                    "厘米\n 你的标准体重是"+weight+"公斤");

    /* 以 findViewById()取得 Button 对象，并添加 onClickListener */
    Button b1 = (Button) findViewById(R.id.button1);
    b1.setOnClickListener(new Button.OnClickListener()
    {
      public void onClick(View v)
      {
        /* 返回 result 回上一个 activity */
        EX03_11_1.this.setResult(RESULT_OK, intent);

        /* 结束这个 activity */
        EX03_11_1.this.finish();
      }
    });
  }

  /* 四舍五入的 method */
  private String format(double num)
  {
    NumberFormat formatter = new DecimalFormat("0.00");
    String s=formatter.format(num);
    return s;
  }

  /* 以 findViewById()取得 Button 对象，并添加 onClickListener */
  private String getWeight(String sex,double height)
  {
    String weight="";
    if(sex.equals("M"))
    {
      weight=format((height-80)*0.7);
    }
    else
    {
      weight=format((height-70)*0.6);
    }
    return weight;
  }
}
```

res/layout/mylayout.xml

mylayout.xml 为 Activity2（EX03_11_1）的 Layout，其中定义了显示计算结果的 TextView 与回上一页的 Button 按钮。

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

```
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  xmlns:android="http://schemas.android.com/apk/res/android"
>
  <TextView
    android:id="@+id/text1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="20sp"
    android:layout_x="50px"
    android:layout_y="72px"
  >
  </TextView>
  <Button
    android:id="@+id/button1"
    android:layout_width="100px"
    android:layout_height="48px"
    android:text="回上一页"
    android:layout_x="110px"
    android:layout_y="180px"
  >
  </Button>
</AbsoluteLayout>
```

AndroidManifest.xml

范例中有两个 Activity，所以 AndroidManifest.xml 里必须有这两个 activity 的声明，否则系统将无法运行。

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
  xmlns:android="http://schemas.android.com/apk/res/android"
  package="irdc.ex03_11"
  android:versionCode="1"
  android:versionName="1.0.0">
  <application
    android:icon="@drawable/icon"
    android:label="@string/app_name">
    <activity
      android:name=".EX03_11"
      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="EX03_11_1"></activity>
  </application>
</manifest>
```

扩展学习

范例中为了在回到上一页时，能够显示之前所输入的数据，故将原本传递次 Activity 的 Intent（里面包含了有数据的 Bundle 对象）再重新返回给主 Activity1。如果要在次 Activity2 中返回其它的数据，例如，经过计算后的结果、数据，此时只需将要返回的数据再放入 Bundle 对象中即可达成。

此外，以本范例而言，其实使用 startActivity()也可达成同样的结果，仅需在主 Activity 被 create 时去判断 Intent 内有没有数据，有的话，就将数据带入；没有的话，就带入空值（null）。但程序还需要再做有无值的比较，较为繁琐，既然 Android API 中有提供更好用的方法，何来不用的道理？更何况如果系统不是只有几行代码，而是几十行、几百行代码，那时头可就大了！

（本章未完结）