大话企业级 Android 开发·第四部分

本教程说明及版权声明

关于国士工作室

我们(国士工作室)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/

1 用户界面 User Interface

小安：博士，前面您在介绍项目目录时多次提到布局文件，我现在还是不大了解，您能不能给我详细的介绍一下呢？
大致：好，对于一个应用而言用户界面是非常重要的一部分，是应用的“脸”，用户对应用的第一印象来自于界面，因此如果没有完美的用户界面，就已经失败了一半，很难留住用户。好的用户界面会极大提高用户的使用欲望并维护客户忠诚度，就好比一个有内涵的女孩长着一张美丽的脸庞，这才会让众多君子念念不忘的吟唱着“窈窕淑女、君子好逑”。下面我就给你讲一讲如何为应用打造一张完美的“脸庞”。另外，我还会教你如何将这个“脸庞”打造成国际明星，让全世界人民都满意，这就是程序如何进行国际化。

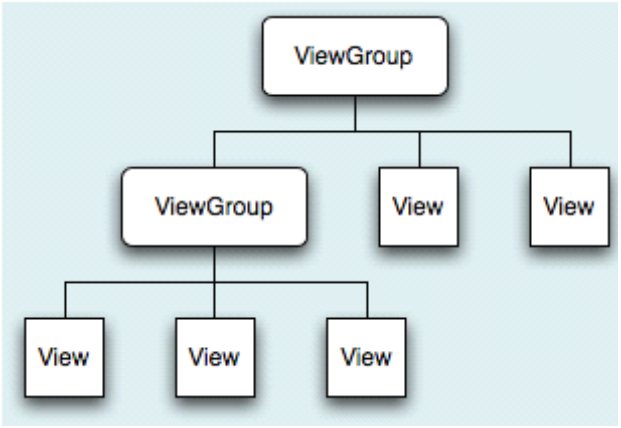
1.1 UI 概述

在 Android 应用中，用户界面是非常重要的，它是人与手机之间传递、交换信息的媒介和对话接口，是 Android 系统的重要组成部分。它实现信息的内部形式与用户可以接受形式之间的转换。iPhone 之所以被人们所推崇，除了其功能强大之外，最重要的是完美的 UI（用户界面）设计，在 Android 系统中，我们也可以开发出与 iPhone 同样绚丽多彩的 UI。

一个 Android 应用的用户界面是由 View 和 ViewGroup 对象构建的。它们有很多的种类，并且都是 View 类的子类，View 类是 Android 系统平台上用户界面表示的基本单元。View 类的一些子类被统称为“widgets（工具）”，它们提供了诸如文本输入框和按钮之类的 UI 对象的完整实现。ViewGroup 是 View 的一个扩展，它可以容纳多个子 View。通过扩展 ViewGroup 类，你可以创建由相互联系的子 View 组成的复合控件。ViewGroup 类同样可以被扩展用作 layout（布局）管理器，如 LinearLayout（线性布局）、TableLayout(表格布局)以及 RelativeLayout(相对布局)等布局架构。并且用户可以通过用户界面与程序进行交互。

- **视图层次**

Android 的官方帮助文档上用如下一幅图来描述 View 与 ViewGroup 之间的组织结构。



图的意思是说多个视图组件（View）可以存放在一个视图容器（ViewGroup）中，该容器可以与其他视图组件共同存放于另一个容器中，但是一个界面文件中必须有且只有一个容器作为根节点。就好比一个箱子里可以装好多水果，这个箱子又可以跟其他水果一块再放入另一个箱子中一样，但是必须有一个大箱子把所有的东西都装进去。

- **布局 Layout**

定义并展现你的视图层次的最常用的方法是使用 XML 布局文件。如同 HTML 一样，XML 为布局提供了一种可读的结构。XML 中的每个元素都是 View 或 ViewGroup 对象（或者是它们的子类）。View 对象是树的叶节点，而 ViewGroup 对象是树的分支（参阅上面的视图层次图）。

XML 元素的名称与它体现的 Java 类相对应。所以一个<TextView>元素将在 UI 中生成一个 TextView 对象，而 <LinearLayout>则创建一个 LinearLayout 视图组。当载入一个布局资源时，Android 系统会根据你布局中的元素初始化这些运行时对象。

您也可以用 Java 代码来绘制 View 和 ViewGroup 对象，并用 addView(View)方法动态的插入新的 View 和 ViewGroup 对象。

您有相当多的方法来对视图进行布局。使用大量不同种类的视图组，您可以有近乎无穷的方式来构建子视图和视图组。Android 提供了一些预定义的视图组，其中包括 LinearLayout、 RelativeLayout、AbsoluteLayout、TableLayout、GridLayout 以及其它的一些。每个都为定义子视图和布局结构提供了一套独特的布局参数。

- **部件 Widgets**

部件是为用户交互界面提供服务的视图对象。Android 提供了一套完整的部件实现，包括按钮、复选框、文本输入框等，以助于快速的构建 UI。Android 还提供了一些更高级的部件，比如日期选择、时钟以及缩放控制。但并没有被局限于 Android 平台提供的这些部件上。如果想创建一些自己的定制动作元素，可以这么做，只要定义自己的视图对象或者扩展或合并现有的部件就行。

- **用户与用户界面的交互**

如果界面仅仅只作为显示，而不能与用户进行交互，则会使应用的实用性和趣味性减去大半。Android 中的每一个 View 都可以触发一系列的事件，如被单击、被触摸等等。当你在用户界面中加入了一些视图和工具之后，你可能想要知道如何让它们与用户交互，进而实现你的动作。如欲获得用户界面事件通知，你需要做以下两件事情之一：

1. 定义一个事件侦听器并将其注册至视图。通常情况下，这是你侦听事件的主要方式。View 类包含了一大堆命名类似 OnXxxListener 的接口，每个都带有一个叫做 OnXxx()的回调方法。OnXxxListener 在其绑定的组件的 Xxx 事件发生时被触发。回调方法 OnXxx()将会在 OnXxxListener 侦听到该事件发生时被调用。比如：View.OnClickListener （用以处理视图中的点击），View.OnTouchListener（用以处理视图中的触屏事件），以及 View.OnKeyListener（用以处理视图中的设备按键事件）。所以，如果你希望你的视图在它被“点击”（比如选择了一个按钮）的时候获得通知，你就要实现 OnClickListener，定义它的 onClick()回调方法（在其中进行相应处理），并将它用 setOnClickListener()方法注册到视图上。切记一点，监听器必须注册到相关的视图上，否则无法发挥作用。
2. 为视图重写一个现有的回调方法。这种方法主要用于你自己实现了一个 View 类，

并想侦听其上发生的特定事件。比如说当屏幕被触摸（onTouchEvent()），当轨迹球发生了移动(onTrackballEvent())或者是设备上的按键被按下(onKeyDown())。这种方式允许你为自己定制的视图中发生的每个事件定义默认的行为，并决定是否需要将事件传递给其它的子视图。这些是 View 类相关的回调方法，所以你只能在你构建自定义组件时定义它们。

- **菜单 Menus**

应用程序菜单是应用程序用户界面中另外一个重要的组成部分。菜单为展现应用程序功能和设置提供了一个可靠的界面。按下设备上的 MENU 键会调出最普通的应用程序菜单。然而，你也可以加入当用户长按一个项目时调出的上下文菜单。

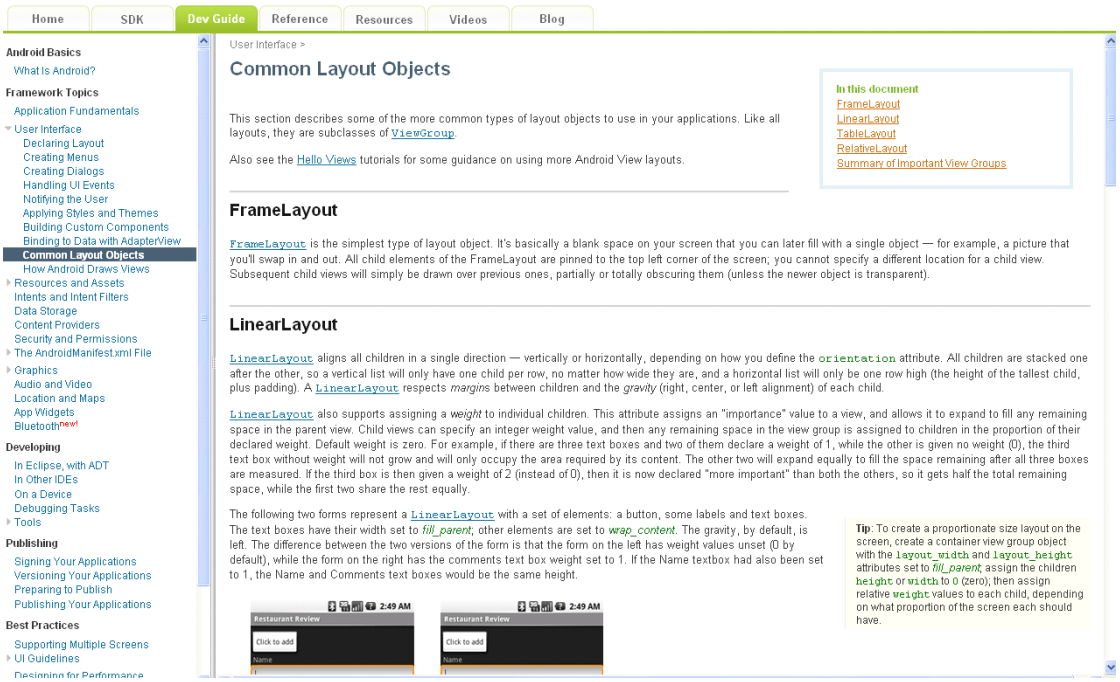
菜单也是用视图层次进行构架的，但你不必自己定义这个架构。你只要为你的 Activity 定义 onCreateOptionsMenu() 和 onCreateContextMenu() 回调方法，并声明你想要包含在菜单中处理的事务或者动作的代码即可。Android 将为你的菜单自动创建视图层次，并在其中绘入你的菜单项。

菜单会自行处理它们的事件，所以你不必为你菜单中的项目注册事件侦听器。当你菜单中的某一项被选定时，框架将自动调用 onOptionsItemSelected() 或 onContextItemSelected()方法。

如同应用程序布局一样。你也可以在一个 XML 文件中定义你菜单中的项目。

1.2布局 Layout

前面我们已经知道LinearLayout（线性布局），Android中还有哪些布局方式呢？我们可以通过查看文档来了解所有的布局信息。参照上一节的内容，进入帮助文档的【Dev Guide】目录下的【User Interface】目录下，选择【Common Layout Objects】超链接，进入如下界面。



在主窗口列出的布局类型有四种，分别是FrameLayout（帧布局）、LinearLayout（线性布局）、TableLayout（表格布局） 、RelativeLayout（相对布局） 。

1.2.1LinearLayout（线性布局）

“LinearLayout”翻译成中文是“线性布局”，所谓线性布局就是在该标签下的所有子元素会根据其orientation属性的值来决定是按行或者是按列逐个显示。

示例main.xml布局文件如下：

```
<?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="@string/name_text"
        />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
```

```
  <Button
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@string/cancle_button"
      />
   <Button
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:text="@string/ok_button"/>
   </LinearLayout>
```

其对应strings.xml内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, UIActivity!</string>
    <string name="app_name">用户界面</string>
    <string name="name_text">请输入用户名</string>
    <string name="ok_button">确定</string>
    <string name="cancle_button">取消</string>
</resources>
```

效果图如下：

图 垂直排列元素示意

其属性“xmlns:android”指定命名空间，顶级元素必须指定命名空间。而在该命名空间中的控件的属性如layout_width，要在属性前加上“android:”做前缀。

其属性“layout_width”指定该元素的宽度，可选值有三种，“fill_parent”、“wrap_content”、具体数字（单位为px）。其中“fill_parent”代表填满其父元素，对于顶级元素来说，其父元素就是整个手机屏幕。“wrap_content”代表该元素的大小仅包裹其自身内容，而数字则代表其占相应的px。

其属性“layout_height”指定该元素的高度，可选参数值与“layout_width”的参数意

义相同。

其属性“orientation”指定子元素排列方式，其中指定为“*vertical*”则是子元素垂直排列，每个子元素会占独立的一行，如上图，而另一个可选值为“*horizontal*”代表子元素水平排列，即每个子元素会占独立的一列。示例main.xml布局文件如下。其对应的strings.xml内容不变。

main.xml

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:text="@string/name_text"
        />
    <EditText
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

   <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cancle_button"
       />
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/ok_button"/>
</LinearLayout>
```

效果图如下：

1.2.2RelativeLayout（相对布局）

相对布局中的视图组件是按相互之间的相对位置来确定的，并不是线性布局中的必须按行或按列单个显示。示例布局文件如下：

main.xml

```
<?xml version="1.0" encoding="utf-8"?>
 <RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/name_text"
        android:id="@+id/text"/>
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/text"
        android:id="@+id/edit"/>
   <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cancle_button"
        android:layout_alignParentRight="true"
        android:layout_below="@id/edit"
        android:id="@+id/cancle"/>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/cancle"
        android:layout_alignTop="@id/cancle"
        android:text="@string/ok_button"/>
    </RelativeLayout>
```

说明：

android:layout_below="@id/text"：将该元素放到id为text的元素的下面

android:layout_toLeftOf="@id/ok" ：放到id为ok的元素左边

android:layout_alignTop="@id/ok" ：对齐id为ok的元素的顶部

界面效果如图：

1.2.3线性布局与相对布局嵌套使用

布局之间可以相互嵌套使用，以完成更为复杂的布局效果。举例来说，下面是一个线性布局当中包含了相对布局的界面：

```
<?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="@string/name_text"
        />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content" />
    <RelativeLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        >
   <Button
        android:layout_width=" wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/cancle_button"
        android:layout_alignParentRight="true"
        android:id="@+id/cancle"/>
    <Button
        android:layout_width=" wrap_content "
        android:layout_height="wrap_content"
        android:layout_toLeftOf="@id/cancle"
```

```
        android:layout_alignTop="@id/cancle"
        android:text="@string/ok_button"/>
    </RelativeLayout>
</LinearLayout>
```

其所对应的 strings.xml 文件内容同前，运行之后效果图如前面相对布局的效果。

1.2.4TableLayout（表格布局）

表格布局的风格跟HTML中的表格比较接近，只是所采用的标签不同。

- <TableLayout> 是顶级元素，说明采用的是表格布局
- <TableRow>定义一个行
- <TextView>定义一个单元格的内容

示例布局文件内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:stretchColumns="0,1,2,3"
   >
    <TableRow>
        <TextView
            android:text="@string/name"
            android:gravity="center"
            android:padding="3dip" />
        <TextView
            android:text="@string/gender"
            android:gravity="center"
            android:padding="3dip" />
        <TextView
            android:text="@string/age"
            android:gravity="center"
            android:padding="3dip" />
        <TextView
            android:text="@string/phonenum"
            android:gravity="center"
            android:padding="3dip" />
    </TableRow>

    <TableRow>
```

```
        <TextView
            android:text="@string/name1"
            android:gravity="center"
            android:padding="3dip" />
        <TextView
            android:text="@string/gender1"
            android:gravity="center"
            android:padding="3dip" />
            <TextView
            android:text="@string/age1"
            android:gravity="center"
            android:padding="3dip" />
        <TextView
            android:text="@string/phonenum1"
            android:gravity="center"
            android:padding="3dip" />
    </TableRow>
    <TableRow>
        <TextView
            android:text="@string/name2"
            android:gravity="center"
            android:padding="3dip" />
        <TextView
            android:text="@string/gender1"
            android:gravity="center"
            android:padding="3dip" />
            <TextView
            android:text="@string/age2"
            android:gravity="center"
            android:padding="3dip" />
        <TextView
            android:text="@string/phonenum2"
            android:gravity="center"
            android:padding="3dip" />
    </TableRow>
</TableLayout>
```

- android:stretchColumns="0,1,2,3"
  该属性指定每行都由第“0、1、2、3”列占满空白空间。
- gravity指定文字对齐方式，本例都设为居中对齐。
- padding指定视图与视图内容间的空隙，单位为像素。

对应的strings.xml文件内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="name">姓名</string>
    <string name="gender">性别</string>
    <string name="age">年龄</string>
    <string name="phonenum">电话</string>
    <string name="gender1">男</string>
    <string name="gender2">女</string>

    <string name="name1">张三</string>
    <string name="age1">25</string>
    <string name="phonenum1">1234567</string>

    <string name="name2">李四</string>
    <string name="age2">24</string>
    <string name="phonenum2">7654321</string>
</resources>
```

界面效果如下：

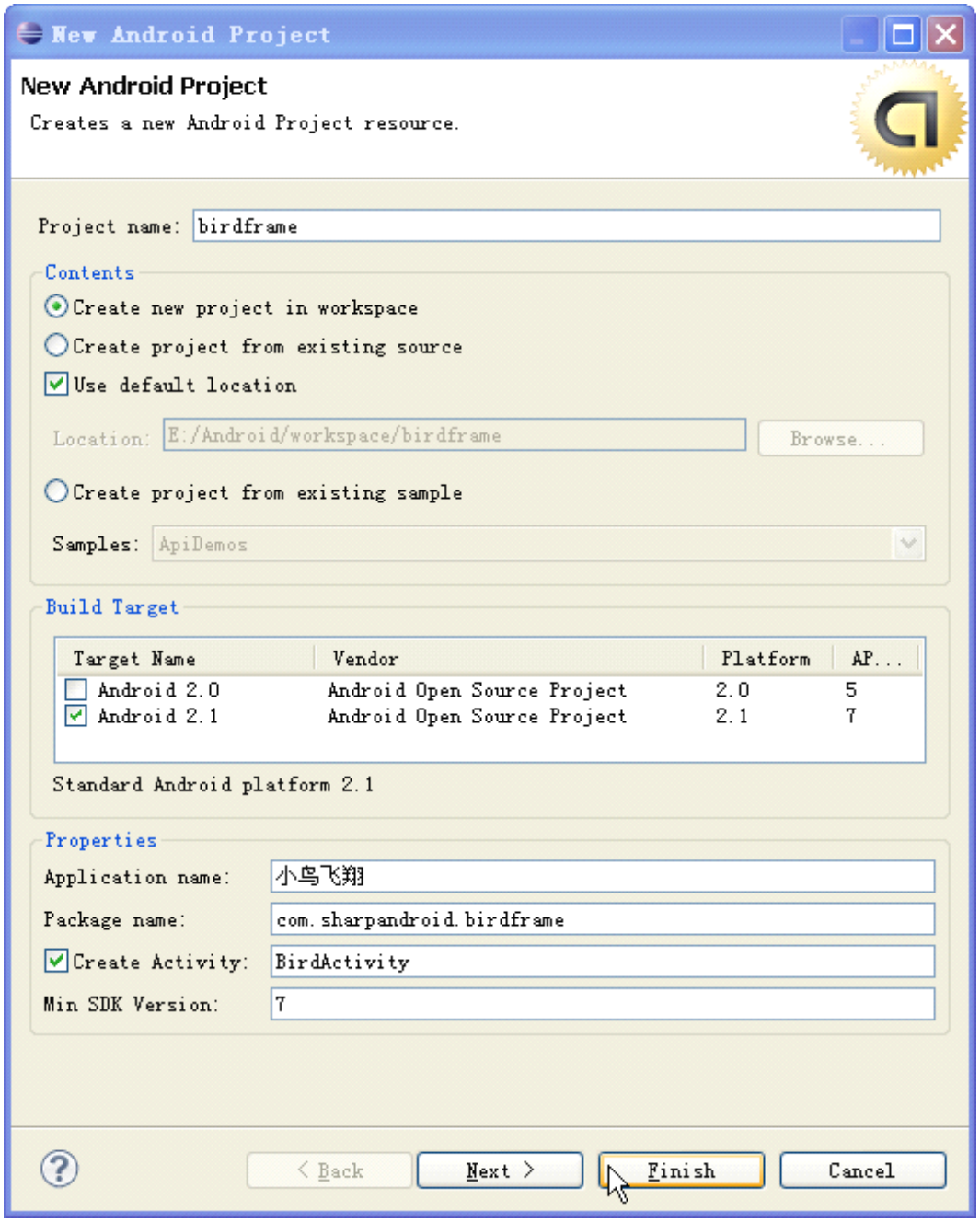
1.2.5FrameLayout（帧布局）

帧布局中的每一个组件都代表一个画面，默认以屏幕左上角作为（0,0）坐标，按组件定义的先后顺序依次逐屏显示，后面出现的会覆盖前面的画面。用该布局可以实现动画效果。

接下来，我们用三幅图片实现一只小鸟飞翔的动画效果。三张图片如下：

1. **创建工程**

首先创建一个Android工程。名称为：“birdframe”。



编写main.xml文件

其内容如下：

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

```
</FrameLayout>
```

在该布局文件中定义一个id为frame的帧布局文件。

编写BirdActivity.java类

内容如下：

```
public class BirdActivity extends Activity {
    FrameLayout frame = null;
    private boolean flag = true;
    //由该类两个方法间的循环调用，实现界面不断更新。
    class MyHandler  extends  Handler{
        int i = 0;
        public void handleMessage(Message msg) {
            i++;
            //总共三幅图，依次显示
            show(i % 3);
            //再次调用sleep方法
            sleep(10);
        }

        public void sleep(long delayMillis) {
            //判断是否继续飞翔
            if(flag) {
                //实质上是调用了一次handleMessage
                sendMessageDelayed(obtainMessage(0), delayMillis);
            }
        }
    }
    //该方法是被调用以更新帧布局的前景图片
    void show(int j) {
        //获取三张图片
        Drawable a = getResources().getDrawable(R.drawable.a);
        Drawable b = getResources().getDrawable(R.drawable.b);
        Drawable c = getResources().getDrawable(R.drawable.c);
        //不同的情况，设置不同的前景
        switch (j) {
        case 0:
            frame.setForeground(a);
            break;
        case 1:
            frame.setForeground(b);
            break;
        case 2:
            frame.setForeground(c);
```

```
            break;
        }
    }
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        frame = (FrameLayout) findViewById(R.id.frame);
        //创建一个Handler子类对象，要调用其方法
        final MyHandler myHandler = new MyHandler();
        myHandler.sleep(10);
        //为fram设置点击事件，当其被点击时，在飞翔与暂停飞翔间切换。
        frame.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                flag = !flag;
                myHandler.sleep(10);
            }
        });
    }
}
```

说明：

由于FrameLayout中后出现的UI控件会覆盖前面出现的UI控件，每次只能显示一个UI控件，因此，我们可以通过在Activity中对每次显示的图片内容进行切换以实现动画效果。或许你会想到开启一条线程来控制切换，但在非主线程中不能更新UI界面，所以，我们使用了Android提供的消息通讯类Handler。该类可以实现非主线程和负责UI的主线程之间的通信，进而间接实现非主线程更新UI界面。由于sleep方法中的sendMessageDelayed(obtainMessage(0), delayMillis);本身会延迟发送一个消息，该消息会被框架传递给handleMessage事件。我们在handleMessage()方法中再次调用sleep()方法，形成一个循环调用。在我们对界面进行点击之前，两个方法会一直循环调用。前景图片也会不断的切换，进而实现动画的效果。

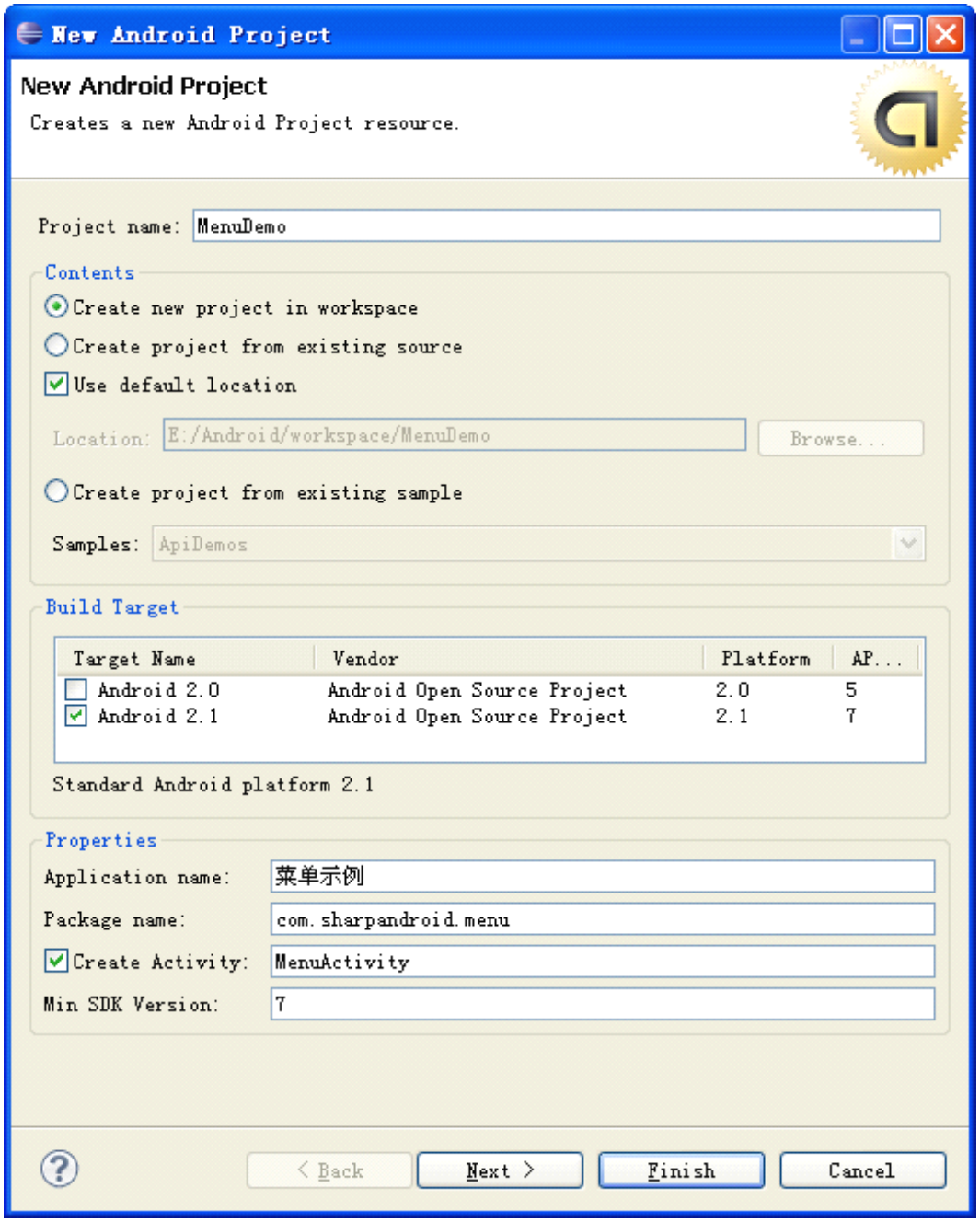
运行程序，效果如下图。

点击图片之后，小鸟停止飞翔。

1.3菜单(Menu)

关于菜单的情况前面已经介绍过了。下面是一个示例程序，演示如何通过 Java 代码以及使用 XML 两种方式为应用程序添加菜单，菜单分为两级，一级菜单项目有“开始”、“取消”、“关于……”，“关于……”菜单下又有两个子菜单“帮助……”、“联系我们……”。其中一级菜单为 Java 代码生成，子菜单为通过 XML 定义。通过 XML 定义，需要在 res/menu 目录下创建一个名为 menu.xml 的文件，定义菜单项。具体过程如下：

创建工程。工程名为“MenuDemo”，Application name 为“菜单示例”。见图：



在 res/menu/目录下创建一个名为 menu.xml 的文件，内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/help"
        android:title="帮助" />
    <item
        android:id="@+id/our"
```

```
        android:title="联系我们" />
</menu>
```

说明:
该文件定义了一组共两个菜单项。

编写MenuActivity.java

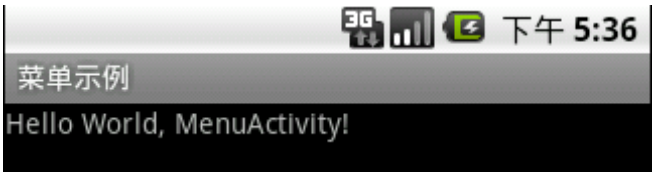
```
package com.sharpandroid.menu;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;

public class MenuActivity extends Activity {

    private static final int OK = 1;
    private static final int CANCLE = 2;
    private static final int ABOUT = 3;

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

    @Override
    //当点击Menu键时会打开菜单，当菜单第一次被打开始，框架回调该方法
    public boolean onCreateOptionsMenu(Menu menu) {
        //为菜单添加一个id为1，标题为”开始“的元素
        menu.add(0, OK, 0, "开始");
        menu.add(0, CANCLE, 0, "取消");
        //为菜单添加一个子菜单，id为3，标题为”关于“，并返回该子菜单对象为file
        Menu file = menu.addSubMenu(0, ABOUT, 0, " 关于");
        //得到一个MenuInflater对象
        MenuInflater inflater = getMenuInflater();
        //调用inflater的inflate方法，获取资源文件中定义的元素，
        //并将这些元素添加进指定的Menu——file
        inflater.inflate(R.menu.submenubyxml, file);
        return true;
    }
```

```
    @Override
    //当一个菜单项被选中时，框架回调该方法，并将被点击的Item传入。
    public boolean onOptionsItemSelected(MenuItem item) {
        //根据被选中的Item进行不同的处理
        switch (item.getItemId()) {
        case OK:
            this.setTitle("开始");
            return true;
        case CANCLE:
            this.setTitle("取消");
            return true;
        case ABOUT:
            this.setTitle("关于");
            return true;
        case R.id.help:
            this.setTitle("帮助信息");
            return true;
        case R.id.our:
            this.setTitle("联系我们");
            return true;
        }
        return false;
    }
}
```

说明：

本例中演示了如何通过 Java 代码创建菜单，并为其添加子菜单，以及如何通过 XML 文件定义菜单项，并在 Java 代码中获得信息之后添加到指定的菜单中。

效果如下：



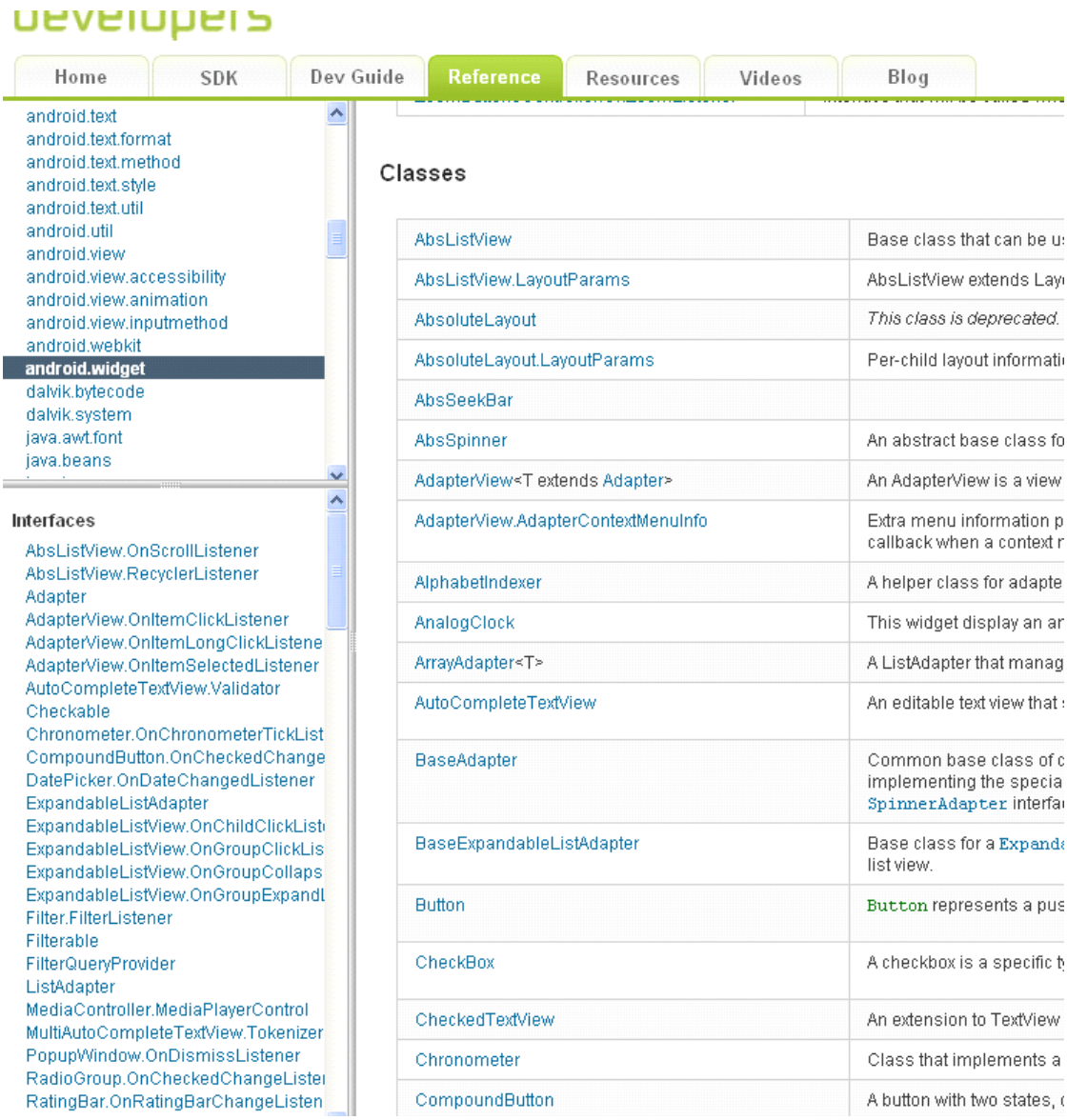
点击【MENU】键，屏幕下方弹出如下菜单。

分别点击【开始】、【取消】返回主界面，并修改标题。点击【关于】，弹出其子菜单，

效果如下图：

1.4常用 UI 控件（Widgets）

Android 当中的 UI 控件的种类有很多种，初学者的学习进度往往会被这么多的控件所阻碍。其实在 Android 开发当中我们经常用的 UI 控件都是比较容易掌握和理解的。其功能可以跟 Html 中的类似元素进行对比，对比学习是我们开发人员必须掌握的能力。在此一些比较简单的 UI 控件不做过多介绍，相信用到之时必定可以轻松掌握。而一些稍微复杂的 UI 控件，如通知、拖动条、进度条等，虽然我们暂时还用不上，但为了统一介绍，方便查阅，特意将这些组件集中起来进行介绍，可以简单了解，另外这些控件的内容比较多，往往看一遍也不可能记住，因此只需知道有某个控件可以完成什么功能，具体细节可以等用到时再来查看即可。当然也可以跳过该部分内容，直接阅读后续部分，待后面的知识中遇到这部分内容时再回过头看相关内容。限于篇幅，一些不常用的控件不再介绍，如果感兴趣可以查阅文档。位置在文档【Reference】目录下【android.widget】。



右侧部分的“Classes”下的内容为 Android 中的控件列表，可以点击相应的超链接查看详细信息。

1.4.1 单选框(RadioButton)

平时上网的过程中，我们经常见到各种单选框，Android 平台为我们提供了单选框的实现方式，利用 RadioGroup 进行分组，在 RadioGroup 内定义若该 RadioButton 选项。

要完成单选框显示，我们需要使用到 RadioGroup 和 RadioButton(单选框)，RadioGroup 用于对单选框进行分组，相同组内的单选框只有一个单选框能被选中。常用方法如下：

- RadioGroup.check(int id);将指定的 RadioButton 设置成选中状态。
- (RadioButton) findViewById(radioGroup.getCheckedRadioButtonId());获取被选中的单选框。
- RadioButton.getText();获取单选框的值
- 调用 setOnCheckedChangeListener()方法，处理单选框被选择事件，把

RadioGroup.OnCheckedChangeListener 实例作为参数传入。

具体应用见后面综合示例。

在布局文件中的应用示例如下：

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">
    <RadioGroup
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:checkedButton="@+id/woman"
        android:id="@+id/sex">
        <RadioButton
            android:text="@string/man"
            android:id="@+id/man"
            />
        <RadioButton
            android:text="@string/woman"
            android:id="@id/woman" />
    </RadioGroup>
</LinearLayout>
```

说明：

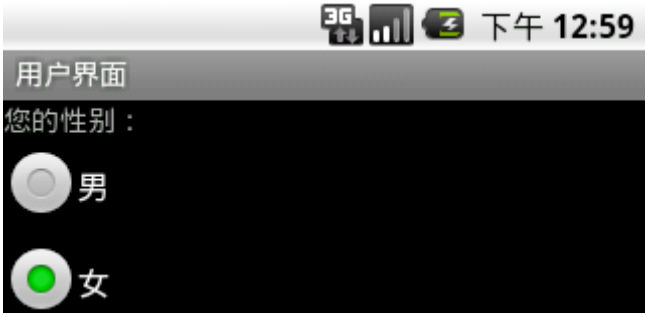
android:checkedButton="@+id/woman"指定 id 为 woman 的 RadioButton 为默认选定项。注意“@”后有“+”;

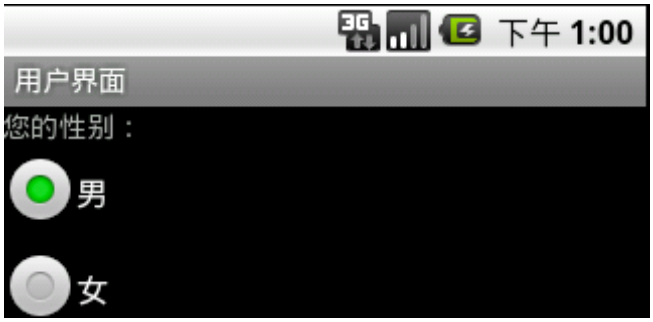
android:id="@id/woman" />

被指定为默认选定的 RadioButton 的 id 不用“+”，直接引用前面定义过的标识即可。

运行效果如图：

“男”和“女”两个选项只能选其一。





1.4.2多选框(CheckBox)

Android 平台为我们提供了多选框的实现方法，每个多选框都是独立的，可以通过迭代所有多选框，然后根据其状态是否被选中再获取其值。CheckBox 类常用方法：

- CheckBox.setChecked(true);将 CheckBox 设置成选中状态。
- CheckBox.getText();获取多选框的值
- CheckBox.isChecked();判断该选项是否被选中
- 调用 setOnCheckedChangeListener()方法，处理多选框被选择事件，把 CompoundButton.OnCheckedChangeListener 实例作为参数传入

应用实例见后面的用户注册实例。

编写 main.xml：

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TextView
        android:layout_width="@dimen/TextViewWidth"
        android:layout_height="wrap_content"
        android:text="@string/favoriteString"
        android:textSize="@dimen/fontSize"
        android:id="@+id/favorite"/>
    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/favorite"
        android:text="@string/pingpang"
        android:id="@+id/checkboxpingpang"/>
    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/checkboxpingpang"
        android:text="@string/football"
        android:id="@+id/checkboxfootball"/>
    <CheckBox
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/favorite"
        android:layout_below="@id/checkboxfootball"
        android:text="@string/basketball"
        android:id="@+id/checkboxbasketball"/>
    <CheckBox
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/checkboxbasketball"
        android:layout_alignTop="@id/checkboxbasketball"
        android:text="@string/tennis"
        android:id="@+id/checkboxtennis"/>
</RelativeLayout>
```

运行结果如图，可以选中多个选项。

图 1-10

1.4.3列表显示（ListView）

ListView 类为 AdapterView 的间接子类，可以列表的形式显示数据，至于显示什么数据以及如何显示数据则需要 Adapter 类及其子类的配合了。若以“家用电视+DVD 机”的放映流程来比喻，则可以将 ListView 理解为电视屏幕，提供一个数据显示的场所；而 Adapter 可以理解为影碟播放机，对播放进行控制；需要在 ListView 中显示的数据理解为碟片，为要显示的内容；数据在 ListView 中显示的格式，可以理解为播放时指定的屏幕制式，如宽屏、全屏等。

播放影片可以使用 DVD 机但也可以使用 VCD 机进行播放，因此，我们可以采用不同的 Adapter，也即其子类，如 SimpleAdapter、ArrayAdapter、CursorAdapter 等等。当然播放的光盘制式可以是 DVD 盘或 VCD 盘，也即数据来源可以有多种渠道，因此我们的数据可以来自自定义的数组、List、数据库、内容提供者（关于从数据库、内容提供者的知识后面会详细介绍）。至于指定数据在 ListView 中的显示方式，我们可以先定义一组元素的预期布局文件，之后，所有的数据项将按此格式显示。

下面我们来学习 ListView 类的常用方法。

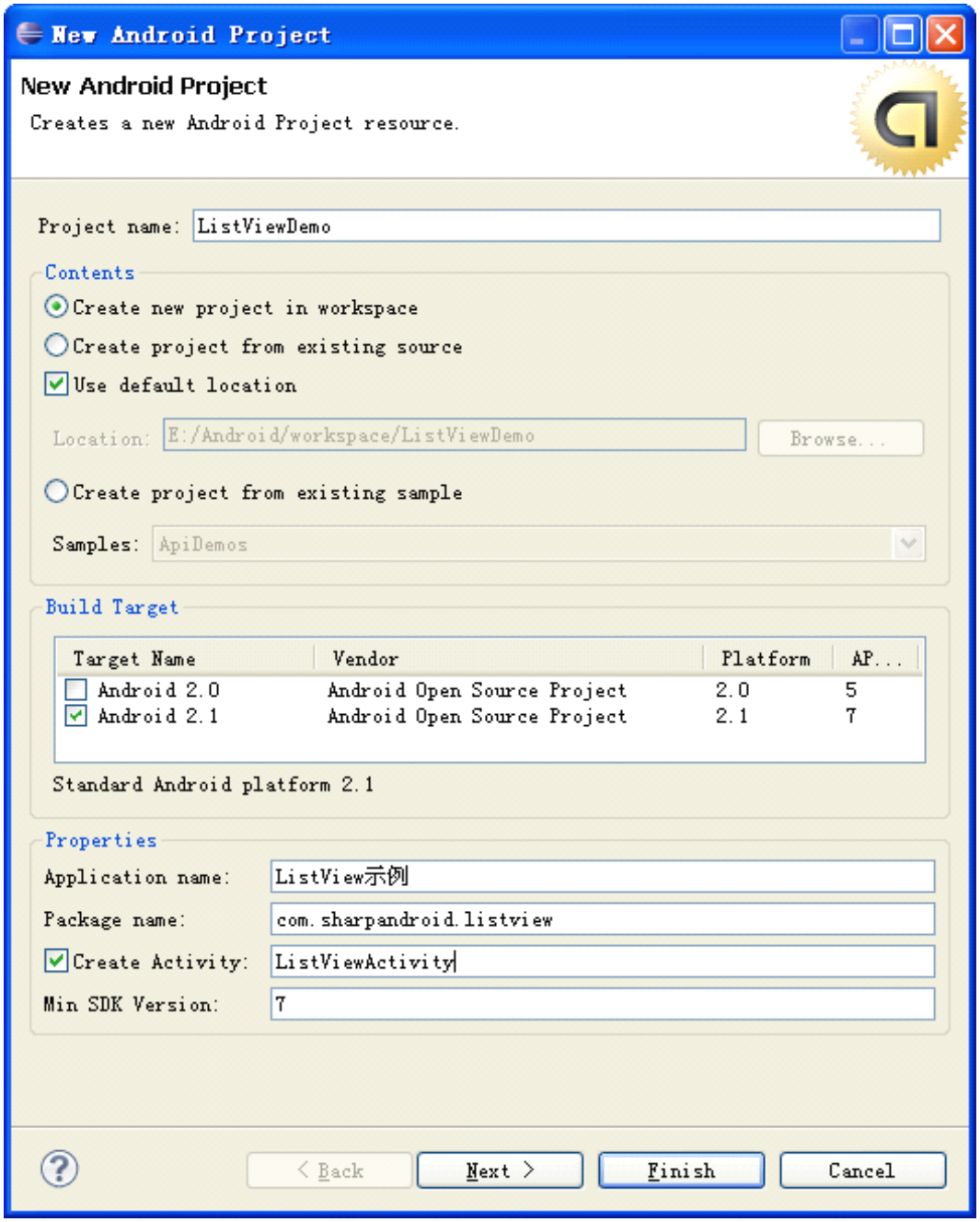
- setAdapter(ListAdapter adapter)
  为 ListView 绑定一个 Adapter
- setChoiceMode(int choiceMode)

为 ListView 指定一个显示的模式，可选值有三个 CHOICE_MODE_NONE（默认值，没有单选或多选效果）、CHOICE_MODE_SINGLE（单选框效果）、 CHOICE_MODE_MULTIPLE（多选框效果）；

- setOnItemClickListener (AdapterView.OnItemClickListener listener)
  为其注册一个元素被点击事件的监听器，当其中某一项被点击时调用其参数 listener 中的 onItemClick（）方法。

本节以 ArrayAdapter 为例，演示如何使用 ListView 显示数据。

创建名为“ListViewDemo”的工程，Application name 为“ListView 示例”。如下图：



编写 string.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, ListViewActivity!</string>
    <string name="app_name">ListView示例</string>
    <string name="name">姓名</string>
</resources>
```

编写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="@string/name"
    />
    <ListView
     android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:id="@+id/listview"
    ></ListView>
</LinearLayout>
```

编写ListViewActivity.java：

```
package com.sharpandroid.listview;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class ListViewActivity extends Activity {
    private ListView listView;
     private String[] name = {"张三","李四","王五","刘六"};
    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        listView = (ListView) findViewById(R.id.listview);
        //创建一个ArrayAdapter
        ArrayAdapter adapter =
            new ArrayAdapter(this, android.R.layout.simple_list_item_1,name);
        listView.setAdapter(adapter);
        //listView注册一个元素点击事件监听器
        listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            //当某个元素被点击时调用该方法
            public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,long
arg3) {
                Toast.makeText(ListViewActivity.this,name[arg2] ,
Toast.LENGTH_LONG).show();
            }
        });
    }
}
```

说明：

```
 ArrayAdapter adapter =     new ArrayAdapter(Context context, int
textViewResourceId, Object[] objects);
```

ArrayAdapter构造方法的参数解释：

context ：当前的Context对象

textViewResourceId：一个包含了TextView元素的布局文件，用来指定ListView中的每一项的显示格式。如前面介绍过的，android.R.layout.simple_list_item_1是Android平台自带的一个布局文件，里面只包含一个TextView标签。其内容如下：

```
        <?xml version="1.0" encoding="utf-8"?>
        <TextView xmlns:android="http://schemas.android.com/apk/res/android"
            android:id="@android:id/text1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceLarge"
            android:gravity="center_vertical"
            android:paddingLeft="6dip"
            android:minHeight="?android:attr/listPreferredItemHeight"
        />
```

objects：要显示的数据，为一个数组

- onItemClick(AdapterView<?> parent, View view, int position, long id)

参数介绍：
parent：被点击的ListView对象
view：被点击的那一项
position：被点击的那一项在ListView中的位置
id　　：被选中的那一行的id

点击其中的“王五”一列，弹出一个Toast，效果如下：

ListActivity 的应用

下面我们将代码修改为通过继承 ListActivity 来实现，现在已经无须 main.xml。

编写 ListViewActivity.java，内容如下：

```
package com.sharpandroid.listview;

import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class ListViewActivity extends ListActivity {
    private String[] name = { "张三", "李四", "王五", "刘六" };

    @Override
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        // 创建一个ArrayAdapter
        ArrayAdapter adapter = new ArrayAdapter(this,
                android.R.layout.simple_list_item_1, name);
        setListAdapter(adapter);
    }

    protected void onListItemClick(ListView l, View v, int position, long id) {
        Toast.makeText(ListViewActivity.this,
name[position],Toast.LENGTH_LONG).show();
    }
}
```

说明：

- setListAdapter(adapter);直接将 adapter 与 ListViewActivity 的 ListView 绑定。并且其直接重写 ListActivity 类的 onListItemClick（）方法就可以对 ListView 中的选项的点击事件进行监听。

执行效果与之前一致。

在之后我们会学习从数据库及内容提供者获取数据，届时再对利用 SimpleAdapter、CursorAdapter 绑定数据进行介绍。

1.4.4 下拉列表框(Spinner)

手机的屏幕较小，因此使用下拉列表来进行选择式输入是一个非常好的方式。Spinner 与 ListView 一样，也是 AdapterView 的一个间接子类，是一个显示数据的窗口。

Spinner 类常用的方法如下：

- Spinner.getItemAtPosition(Spinner.getSelectedItemPosition());获取下拉列表框的值
- 调用 setOnItemSelectedListener()方法，处理下拉列表框被选择事件，把 AdapterView.OnItemSelectedListener 实例作为参数传入

可以在 Java 代码中通过 Adapter 绑定数据，也可以在布局文件中直接引用在资源文件中定义的数组。

编写 arrays.xml，定义 Spinner 中需要显示的数据。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="colors">
        <item>red</item>
        <item>orange</item>
        <item>yellow</item>
        <item>green</item>
        <item>blue</item>
        <item>violet</item>
    </string-array>
```

```
</resources>
```

编写 main.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:padding="10dip"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/color"
    />
    <Spinner android:id="@+id/spinner1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:prompt="@string/color_prompt"
        android:entries="@array/colors"
    />
</LinearLayout>
```

说明：

android:prompt="@string/color_prompt"

设置弹出下拉列表的标题。

android:entries="@array/colors"

指定下拉列表中的数据。本例为在arrays.xm文件中定义的colors数组。

运行效果如图 1-11；

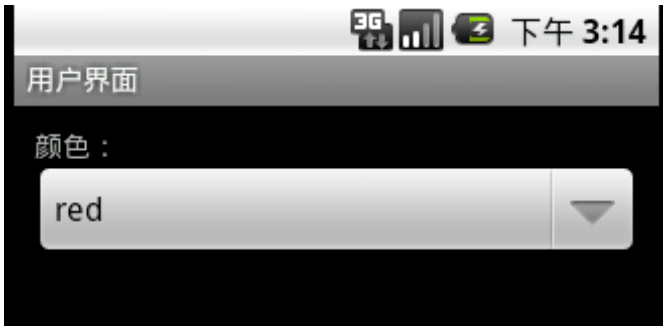


图 1-11

1.4.5进度条（ProgressBar）

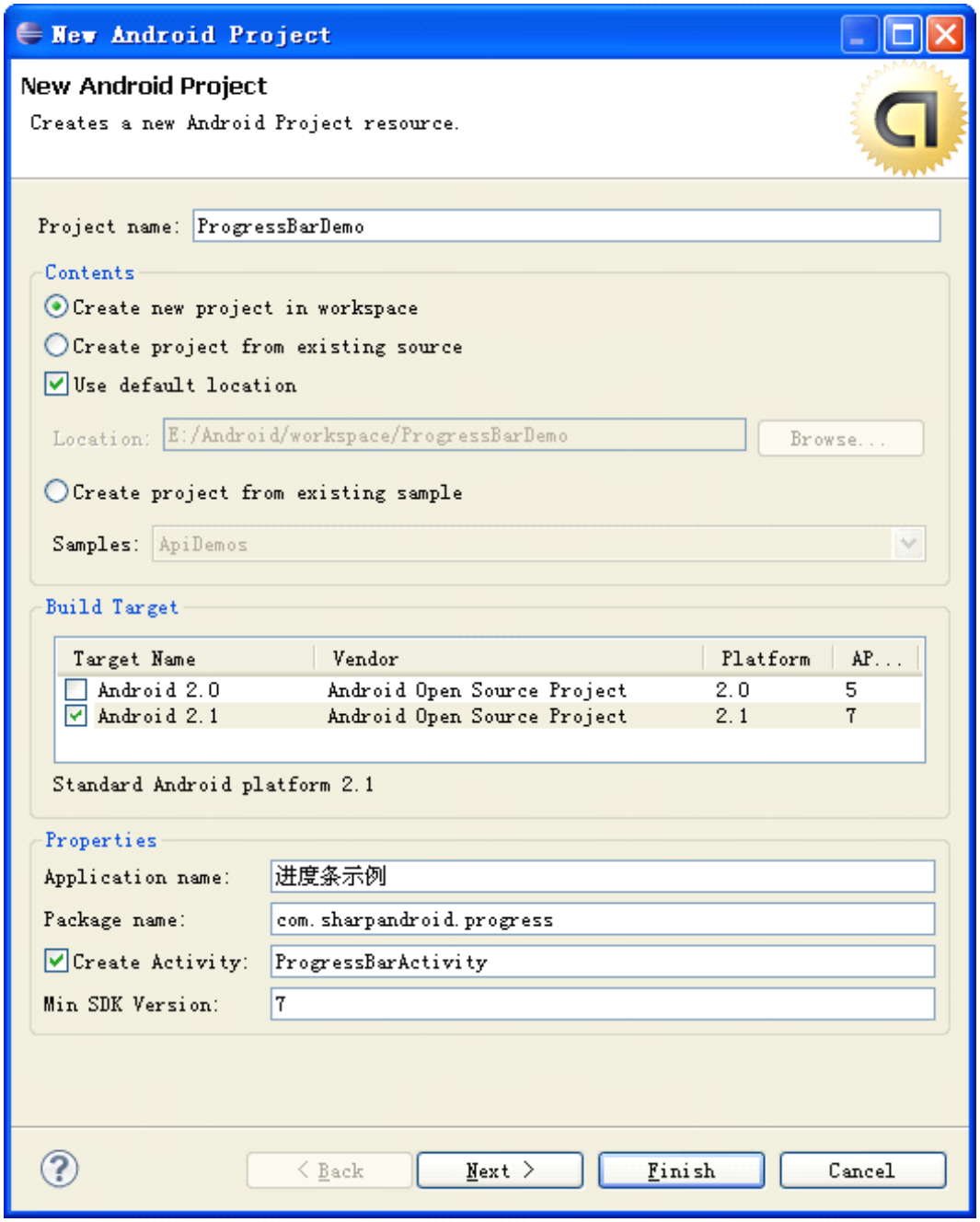
在XP系统中，我们安装软件或者使用千千静听听歌曲时都会遇到进度条的效果，进度条可以带给用户良好的体验，Android系统已经为我们提供了ProgressBar类来完成进度条的效果，我们可以很方便的运用该类。其中最常见的两种进度条是“环形进度条”和“水平进度条”，在布局文件中定义两种进度条的方式比较相似，区别是，定义“水平进度条时”需要加上一项属性“style="?android:attr/progressBarStyleHorizontal"”。 ProgressBar类中常用的方法如下：

ProgressBar.setMax(int max);设置总长度为100

ProgressBar.setProgress(int progress);设置已经开启长度为0,假设设置为50,进度条将进行到一半停止。

下面是一个应用实例。

创建名为“ProgressBarDemo”的工程，其Application name为“进度条示例”。如下图：



编写 strings.xml 文件：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, ProgressBarActivity!</string>
    <string name="app_name">进度条示例</string>
    <string name="progressbar1">环形进度条</string>
    <string name="progressbar">水平进度条</string>
</resources>
```

编写 main.xml 文件：

```
<?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="@string/progressbar1"
        />
    <ProgressBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/progress_bar1"
        ></ProgressBar>
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/progressbar"
        />
    <ProgressBar
        style="?android:attr/progressBarStyleHorizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/progress_bar"
        ></ProgressBar>
</LinearLayout>
```

说明：

```
style="?android:attr/progressBarStyleHorizontal"
```

该行指定该 ProgressBar 为水平样式。

编写 ProgressBarActivity.java 文件：

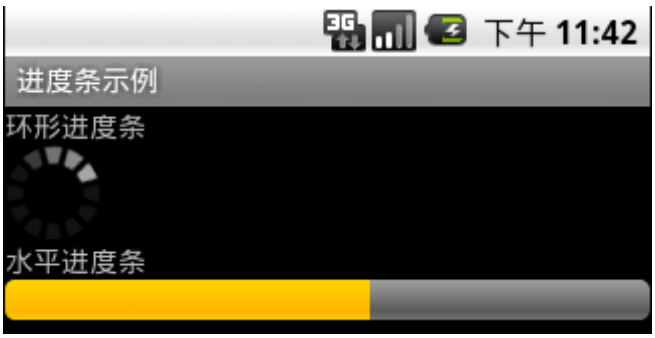
```
package com.sharpandroid.progress;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.widget.ProgressBar;
```

```
public class ProgressBarActivity extends Activity {
    private ProgressBar mProgress;
    private int mProgressStatus = 0;
    //创建一个 Handler 对象
    private Handler mHandler = new Handler();

    protected void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.main);

        mProgress = (ProgressBar) findViewById(R.id.progress_bar);
        //设定进度条的最大值，其将为该进度条显示的基数。
        mProgress.setMax(10000);
        //新开启一个线程
        new Thread(new Runnable() {
            public void run() {
                //循环 10000 次，不停地更新 mProgressStatus 的值
                while (mProgressStatus ++< 10000) {
                    //将一个 Runnable 对象添加到消息队列当中，
                    //并且当执行到该对象时执行 run()方法
                    mHandler.post(new Runnable() {
                        public void run() {
                            //重新设置进度条当前的值
                            mProgress.setProgress(mProgressStatus);
                        }
                    });
                }
            }
        }).start();
    }
}
```

说明：

代码中开启一个新线程，并在新线程中循环 10000 次，每次循环更新进度条的当前值，并且刷新界面。

执行程序，效果如下：

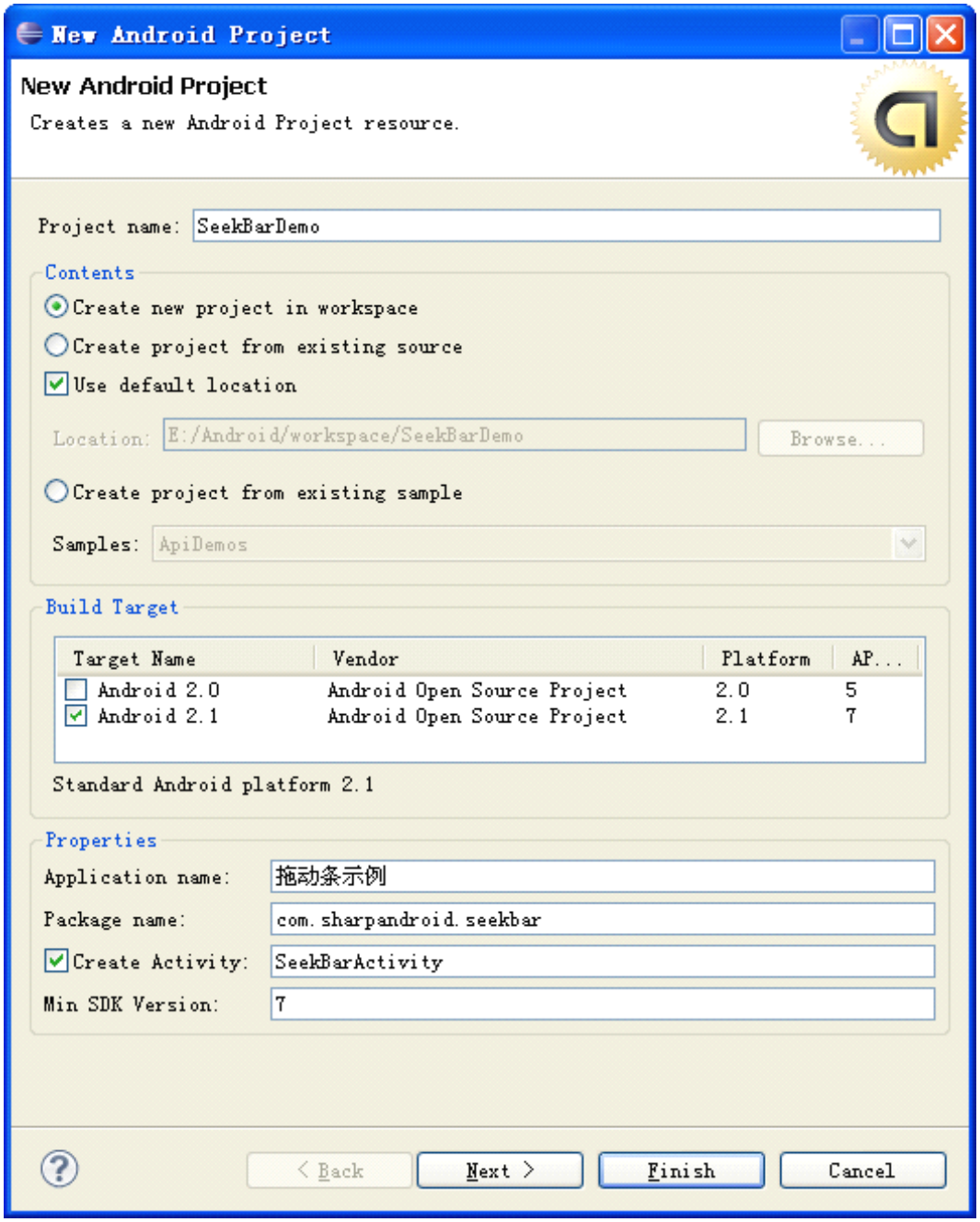


1.4.6拖动条(SeekBar)

进度条只能显示进度，用户不能与程序交互，通过对其操作来改变其值。而拖动条就可以实现此功能。拖动条比较常见，如“千千静听”中的播放进度条就是一个拖动条。Android 平台中的 SeekBar 是 ProgressBar 的间接子类。

- SeekBar.getProgress()：获取拖动条当前值
- 调用 setOnSeekBarChangeListener()方法，处理拖动条值变化事件，把 SeekBar.OnSeekBarChangeListener 实例作为参数传入

下面的实例可以模拟实现该进度（拖动）条，还可以通过拖动游标改变进度值，每次拖动之后仍会自动更新。



编写 strings.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, SeekBarActivity!</string>
    <string name="app_name">拖动条示例</string>
    <string name="speed">播放进度:</string>
</resources>
```

编写 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="@string/speed"
        android:id="@+id/speed"
        />
    <SeekBar
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/seek_bar"
        />
</LinearLayout>
```

编写 SeekBarActivity.java 类：

```
package com.sharpandroid.seekbar;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.widget.SeekBar;
import android.widget.TextView;

public class SeekBarActivity extends Activity {
    private SeekBar seekBar ;
    private TextView textView;
    //标记是否需要刷新
    private boolean flag=true;
    private Handler mHandler = new Handler();

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

        seekBar = (SeekBar) findViewById(R.id.seek_bar);
        textView = (TextView) findViewById(R.id.speed);
```

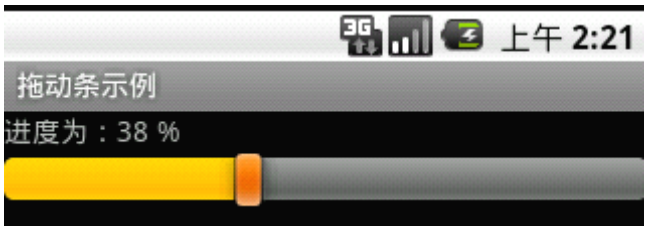
```
        //设定拖动条的最大值，其将为该拖动条显示的基数。
        seekBar.setMax(100);
        //该方法为 seekBar 注册一个监听，当 SeekBar 发生改变时调用参数 1 中的对应方
    法。
        seekBar.setOnSeekBarChangeListener(new SeekBar.OnSeekBarChangeListener()
{
            @Override
            //当游标移动停止时调用该方法
            public void onStopTrackingTouch(SeekBar seekBar) {
                //设置标记为需要刷新
                flag = true;
                //刷新
                refresh();
            }

            @Override
            //当游标开始移动时调用该方法
            public void onStartTrackingTouch(SeekBar seekBar) {
                //停止刷新
                flag = false;
            }

            @Override
            //当进度条游标被改变或者进度更改时调用该方法
            public void onProgressChanged(SeekBar seekBar, int progress,
                    boolean fromUser) {
                //更改 textView 的内容
                textView.setText("进度为：" + progress + "%");
            }
        });
        //创建时就开始自动更新该拖动条
        refresh();
        }

    //该方法自动刷新拖动条的进度值
    private void  refresh(){
          new Thread(new Runnable() {
            public void run() {

                //当进度不到 100，就更新 mProgressStatus 的值
                while (flag && seekBar.getProgress()< 100) {
```

```
                    try {
                        //暂停 1 秒
                        Thread.sleep(1000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }

                    //将一个 Runnable 对象添加到消息队列当中，
                    //并且当执行到该对象时执行 run()方法
                    mHandler.post(new Runnable() {
                        public void run() {
                            //重新设置进度条当前的值，加 1
                            seekBar.setProgress(seekBar.getProgress() + 1);
                        }
                    });
                }
            }
        }).start();
    }
}
```

说明：

当 SeekBarActivity 创建时，为 seekBar 注册监听事件。 并且调用 refresh()方法，自动刷新拖动条的进度值。当拖动条进度值改变时，修改 seekBar 上部 TextView 的内容。

运行结果如图；
初始时自动更新进度：



此时拖动游标，将其往回拖动，显示值为“21%”，之后又自动更新拖动条的进度值。

效果如下图：