

# Android 学习笔记总结

## 第一步：

### Android(1) - 在 Windows 下搭建 Android 开发环境，以及 Hello World 程序

搭建 Android 的开发环境，以及写一个简单的示例程序

- 在 Windows 下搭建 Android 开发环境
- Android 项目的目录结构说明
- 写一个简单的 Hello World 程序

#### 一、在 Windows 下搭建 Android 开发环境

##### 1、安装 JDK (Java Development Kit)

<http://download.java.net/jdk6/>

##### 2、安装 Android SDK

<http://developer.android.com/sdk>

##### 3、安装 Eclipse

<http://www.eclipse.org/>

##### 4、打开 Eclipse，并安装其 Android 插件 (ADT)

打开菜单 "Help" -> "Install New Software", 在 "Availabe Software" 中加入地址

<http://dl-ssl.google.com/android/eclipse/>，然后安装 ADT(Android Development Tools)

##### 5、新建 Android 项目

"New" -> Android Project, Project Name - 项目名称; Build Target - 编译项目的 SDK 版本; Application name - 程序名称; Package name - 包名; Min SDK Version - 程序所支持的最低 SDK 版本代号 (2 对应 1.1, 3 对应 1.5, 4 对应 1.6)

##### 6、运行 Android 项目

打开菜单 "Run" -> "Run Configurations" -> New launch configuration, 设置启动项目名称, 在 Android 选项卡中选择启动项目, 在 Target 选项卡中设置模拟器

##### 7、创建/使用模拟 SD 卡

创建 SD 卡, 运行类似如下命令: `mksdcard -l sdcard 512M d:\android\sdcard.img`

模拟器中使用 SD 卡, 在项目配置的 Target 选项卡的 "Additional Emulator Command Line Options" 框中输入类似如下参数: `-sdcard d:\android\sdcard.img`

##### 8、配置模拟器

运行类似如下命令：android create avd --name android15 --target 2。或者直接在菜单 "Window" -> "Android AVD Manager" 中配置模拟器

#### 9、浏览模拟 SD 卡中的内容

调试程序，在 DDMS 中选择 "File Explorer"，在其中的 sdcard 目录下就是模拟 SD 卡中的内容

#### 10、查看日志 LogCat

Window -> Show View -> Other -> Android -> LogCat

#### 11、在模拟器中安装/卸载 apk

安装 apk 运行类似如下命令：adb install name.apk；卸载 apk 运行类似如下命令：adb uninstall packagename（注：这里的参数是需要卸载的包名）

#### 12、反编译 Android 程序

解压 apk 文件，取出其中的 classes.dex 文件，运行类似如下命令：dexdump.exe -d classes.dex > dump.txt（其意思是将 classes.dex dump 出来，并将反编译后的代码保存到指定的文本文件中）

#### 13、人品不好是出现的某些错误的解决办法

如果出现类似如下的错误等

no classfiles specified

Conversion to Dalvik format failed with error 1

解决办法：Project -> Clean

出现 Android SDK Content Loader 60%（一直卡在 60%）

解决办法：Project -> 去掉 Build Automatically 前面的勾

#### 14、查看 SDK 源代码

先想办法搞到源代码，如这个地址 <http://www.digginmobile.com/android.asp>，然后将其解压到 SDK 根路径下的 sources 文件夹内即可

## 二、Android 项目的目录结构

### 1、src - 用于放置源程序

2、gen - 自动生成 R.java 文件，用于引用资源文件（即 res 目录下的数据）

3、assets - 用于放置原始文件，Android 不会对此目录下的文件做任何处理，这是其与 res 目录不同的地方

4、res/drawable - 用于放置图片之类的资源；res/layout - 用于放置布局用的 xml 文件；res/values - 用于放置一些常量数据

5、AndroidManifest.xml - Android 程序的清单文件，相当于配置文件，配置应用程序名称、图标、Activity、Service、Receiver 等

## 三、Hello World 程序

### 1、res/layout/main.xml

代码

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

设置 ID 的方式：ID 前加前缀，@+id/

引用资源文件内字符串资源的方式：指定的资源名称前加前缀，@string/

-->

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/layout"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"
    />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:id="@+id/txt"
    />
</LinearLayout>
```

## 2、res/values/strings.xml

代码

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">layout 直接调用 values 中的字符串</string>
    <string name="hello2">编程方式调用 values 中的字符串</string>
    <string name="app_name">webabcd_hello</string>
</resources>
```

## 3、res/drawable 目录下放置一个名为 icon.png 的图片文件

## 4、AndroidManifest.xml

代码

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.webabcd.hello"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Main"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

```
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
</application>
<uses-sdk android:minSdkVersion="3" />
</manifest>
```

#### 5、Main.java

代码

```
package com.webabcd.hello;

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

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

        // 将指定的布局文件作为 Activity 所显示的内容
        setContentView(R.layout.main);

        // 动态地在指定的容器控件上添加新的控件
        TextView txt = new TextView(this);
        txt.setText("动态添加控件");
        // setContentView(txt);
        ((LinearLayout)findViewById(R.id.layout)).addView(txt);

        // 引用资源文件内的内容作为输出内容
        TextView txt1 = (TextView)findViewById(R.id.txt);
        txt1.setText(this.getString(R.string.hello2));
    }
}
```

## Android(2) - 布局(Layout)和菜单(Menu)

介绍

在 Android 中各种布局的应用，以及菜单效果的实现

- 各种布局方式的应用，FrameLayout, LinearLayout, TableLayout, AbsoluteLayout, RelativeLayout
- 为指定元素配置上下文菜单，为应用程序配置选项菜单，以及多级菜单的实现

## 1、各种布局方式的演示

res/layout/main.xml

代码

```
<?xml version="1.0" encoding="utf-8"?>
<!--
layout_width - 宽。fill_parent: 宽度跟着父元素走；wrap_content: 宽度跟着本身的内容走；直接指定一个 px 值
来设置宽
layout_height - 高。fill_parent: 高度跟着父元素走；wrap_content: 高度跟着本身的内容走；直接指定一个 px 值
来设置高
-->

<!--
LinearLayout - 线形布局。
    orientation - 容器内元素的排列方式。vertical: 子元素们垂直排列；horizontal: 子元素们水平排列
    gravity - 内容的排列形式。常用的有 top, bottom, left, right, center 等，详见文档
-->
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:gravity="right"
    android:layout_width="fill_parent" android:layout_height="fill_parent">

    <!--
    FrameLayout - 层叠式布局。以左上角为起点，将 FrameLayout 内的元素一层覆盖一层地显示
    -->
    <FrameLayout android:layout_height="wrap_content"
        android:layout_width="fill_parent">
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text="FrameLayout">
        </TextView>
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text="Frame Layout">
        </TextView>
    </FrameLayout>

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="@string/hello" />

    <!--
    TableLayout - 表格式布局。
-->
```

TableRow - 表格内的行，行内每一个元素算作一列  
collapseColumns - 设置 TableLayout 内的 TableRow 中需要隐藏的列的列索引，多个用 “,” 隔开  
stretchColumns - 设置 TableLayout 内的 TableRow 中需要拉伸（该列会拉伸到所有可用空间）的列的列索引，多个用 “,” 隔开  
shrinkColumns - 设置 TableLayout 内的 TableRow 中需要收缩（为了使其他列不会被挤到屏幕外，此列会自动收缩）的列的列索引，多个用 “,” 隔开

-->

```
<TableLayout android:id="@+id/TableLayout01"
    android:layout_width="fill_parent" android:layout_height="wrap_content"
    android:collapseColumns="1">
    <TableRow android:id="@+id/TableRow01" android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView android:layout_width="wrap_content"
            android:layout_weight="1" android:layout_height="wrap_content"
            android:text="行 1 列 1" />
        <TextView android:layout_width="wrap_content"
            android:layout_weight="1" android:layout_height="wrap_content"
            android:text="行 1 列 2" />
        <TextView android:layout_width="wrap_content"
            android:layout_weight="1" android:layout_height="wrap_content"
            android:text="行 1 列 3" />
    </TableRow>
    <TableRow android:id="@+id/TableRow01" android:layout_width="wrap_content"
        android:layout_height="wrap_content">
        <TextView android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:text="行 2 列 1" />
    </TableRow>
</TableLayout>
```

<!--

AbsoluteLayout - 绝对定位布局。

layout\_x - x 坐标。以左上角为顶点

layout\_y - y 坐标。以左上角为顶点

-->

```
<AbsoluteLayout android:layout_height="wrap_content"
    android:layout_width="fill_parent">
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="AbsoluteLayout"
        android:layout_x="100px"
        android:layout_y="100px" />
</AbsoluteLayout>
```

<!--

RelativeLayout - 相对定位布局。

```

    layout_centerInParent - 将当前元素放置到其容器内的水平方向和垂直方向的中央位置（类似的属性有：layout_centerHorizontal, layout_alignParentLeft 等）
    layout_marginLeft - 设置当前元素相对于其容器的左侧边缘的距离
    layout_below - 放置当前元素到指定的元素的下面
    layout_alignRight - 当前元素与指定的元素右对齐
-->
<RelativeLayout android:id="@+id/RelativeLayout01"
    android:layout_width="fill_parent" android:layout_height="fill_parent">
    <TextView android:layout_width="wrap_content" android:id="@+id/abc"
        android:layout_height="wrap_content" android:text="centerInParent=true"
        android:layout_centerInParent="true" />
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="marginLeft=20px"
        android:layout_marginLeft="20px" />
    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="xxx"
        android:layout_below="@id/abc" android:layout_alignRight="@id/abc" />
</RelativeLayout>

</LinearLayout>

```

res/values/strings.xml

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello Layout</string>
    <string name="app_name">webabcd_layout</string>
</resources>

```

Main.java

代码

```

package com.webabcd.layout;


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

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

```

## 2、上下文菜单，选项菜单，子菜单

res/layout/main.xml

 代码

```
<?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:id="@+id/txt1" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="@string/hello_contextMenu" />

    <TextView android:id="@+id/txt2" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="@string/hello_subMenu" />


</LinearLayout>
```

res/values/strings.xml

 代码

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello_contextMenu">Hello Context Menu</string>
    <string name="hello_subMenu">Hello Context Sub Menu</string>
    <string name="app_name">webabcd_menu</string>
</resources>
```

Main.java

 代码

```
package com.webabcd.menu;

import android.app.Activity;
import android.os.Bundle;
import android.view.ContextMenu;
import android.view.Menu;
import android.view.MenuItem;
import android.view.SubMenu;
import android.view.View;
import android.view.ContextMenu.ContextMenuInfo;
import android.widget.TextView;
import android.widget.Toast;
```



// 演示两种菜单的实现方式：上下文菜单（通过在某元素上长按，来呼出菜单）和选项菜单（通过按手机上的菜单按钮，来呼出菜单）

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

        // 为 R.id.txt1 注册一个上下文菜单（在此 TextView 上长按，则会呼出上下文菜单）
        // 具体呼出的菜单内容需要重写 onCreateContextMenu 来创建
        TextView txt1 = (TextView) this.findViewById(R.id.txt1);
        this.registerForContextMenu(txt1);

        // 为 R.id.txt2 注册一个上下文菜单
        TextView txt2 = (TextView) this.findViewById(R.id.txt2);
        this.registerForContextMenu(txt2);
    }

    // 重写 onCreateContextMenu 用以创建上下文菜单
    // 重写 onContextItemSelected 用以响应上下文菜单
    @Override
    public void onCreateContextMenu(ContextMenu menu, View v,
        ContextMenuInfo menuInfo) {
        super.onCreateContextMenu(menu, v, menuInfo);

        // 创建 R.id.txt1 的上下文菜单
        if (v == (TextView) this.findViewById(R.id.txt1)) {

            // ContextMenu.setIcon() - 设置菜单的图标
            // ContextMenu.setHeaderTitle() - 设置菜单的标题
            menu.setHeaderIcon(R.drawable.icon01);
            menu.setHeaderTitle("我是菜单");

            // 用 ContextMenu.add() 来增加菜单项，返回值为 MenuItem
            // 第一个参数：组 ID
            // 第二个参数：菜单项 ID
            // 第三个参数：顺序号
            // 第四个参数：菜单项上显示的内容
            menu.add(1, 0, 0, "菜单 1");

            // MenuItem - 新增菜单项后的返回类型，针对菜单项的其他设置在此对象上操作
            menu.add(1, 1, 1, "菜单 2").setCheckable(true);
        }
    }
}
```

```
}  
// 创建 R.id.txt2 的上下文菜单 ( 多级上下文菜单 )  
else if (v == (TextView) this.findViewById(R.id.txt2)) {  
  
    // ContextMenu.addSubMenu("菜单名称") - 用来添加子菜单。子菜单其实就是一个特殊的菜单  
    SubMenu sub = menu.addSubMenu("父菜单 1");  
    sub.setIcon(R.drawable.icon01);  
    sub.add(0, 0, 0, "菜单 1");  
    sub.add(0, 1, 1, "菜单 2");  
    sub.setGroupCheckable(1, true, true);  
  
    SubMenu sub2 = menu.addSubMenu("父菜单 2");  
    sub2.setIcon(R.drawable.icon01);  
    sub2.add(1, 0, 0, "菜单 3");  
    sub2.add(1, 1, 1, "菜单 4");  
    sub2.setGroupCheckable(1, true, false);  
  
}  
}  
  
// 重写 onCreateOptionsMenu 用以创建选项菜单  
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
  
    MenuItem menuItem = menu.add(0, 0, 0, "菜单 11111111111111111111");  
  
    // MenuItem.setIcon() - 设置菜单项的图标  
    // MenuItem.setTitleCondensed() - 菜单的简标题，如果指定了简标题的话，菜单项上的标题将会以此简标题为准  
    // MenuItem.setAlphabeticShortcut() - 设置选中此菜单项的快捷键  
    // 注：菜单项超过 6 个的话，第 6 个菜单将会变为 More 菜单，多余的菜单会在单击 More 菜单之后显示出来  
    menuItem.setIcon(R.drawable.icon01);  
    menuItem.setTitleCondensed("菜单 1");  
    menuItem.setAlphabeticShortcut('a');  
  
    menu.add(0, 1, 1, "菜单 2").setIcon(R.drawable.icon02);  
    menu.add(0, 2, 2, "菜单 3").setIcon(R.drawable.icon03);  
    menu.add(0, 3, 3, "菜单 4");  
    menu.add(0, 4, 4, "菜单 5");  
    menu.add(0, 5, 5, "菜单 6");  
    menu.add(0, 6, 6, "菜单 7").setIcon(R.drawable.icon04);  
}
```

```
menu.add(0, 7, 7, "菜单 8").setIcon(R.drawable.icon05);

return true;
}

// 重写 onOptionsItemSelected 用以响应选项菜单
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    super.onOptionsItemSelected(item);

    Toast.makeText(Main.this, "被单击的菜单项为：" + String.valueOf(item.getItemId()), Toast.LENGTH_SHORT).show();

    return false;
}
}
```

## Android(3) - 对话框(Dialog)和通知(Notification)

介绍

在 Android 中种对话框及各种通知效果的应用

- 常用对话框的使用，弹出式对话框、日期选择对话框、时间选择对话框、进度条对话框
- 通知（出现在通知列表）和提示性通知（Toast）的演示

### 1、常用对话框的演示

res/layout/main.xml

代码

```
<?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:id="@+id/txtMsg" android:layout_width="wrap_content"
        android:layout_height="wrap_content"> </TextView>

    <Button android:id="@+id/btn1" android:layout_width="wrap_content"
        android:layout_height="wrap_content"> </Button>
    <Button android:id="@+id/btn2" android:layout_width="wrap_content"
        android:layout_height="wrap_content"> </Button>
```

```
<Button android:id="@+id/btn3" android:layout_width="wrap_content"
    android:layout_height="wrap_content"></Button>
<Button android:id="@+id/btn4" android:layout_width="wrap_content"
    android:layout_height="wrap_content"></Button>
<Button android:id="@+id/btn5" android:layout_width="wrap_content"
    android:layout_height="wrap_content"></Button>
<Button android:id="@+id/btn6" android:layout_width="wrap_content"
    android:layout_height="wrap_content"></Button>

</LinearLayout>
```

res/layout/view.xml

 代码

```
<?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:text="我是一个 View"
        android:layout_width="wrap_content" android:layout_height="wrap_content">
    </TextView>

</LinearLayout>
```

res/values/array.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!--
        定义一个名为 ary 的 string 类型的数组
    -->
    <string-array name="ary">
        <item>项目 1</item>
        <item>项目 2</item>
    </string-array>
</resources>
```

Main.java

 代码

```
package com.webabcd.dialog;

import java.util.Calendar;
```

```
import android.app.Activity;
import android.app.AlertDialog;
import android.app.DatePickerDialog;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.app.TimePickerDialog;
import android.app.DatePickerDialog.OnDateSetListener;
import android.app.TimePickerDialog.OnTimeSetListener;
import android.content.DialogInterface;
import android.content.DialogInterface.OnClickListener;
import android.os.Bundle;
import android.view.View;
import android.widget.DatePicker;
import android.widget.TextView;
import android.widget.TimePicker;
import android.widget.Button;

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

        // 弹出式对话框的 Demo。先调用 Builder()，在 Create()，需要显示对话框的是后再调用 show()
        AlertDialog dialog = new AlertDialog.Builder(this).setTitle("弹出对话框").create();
        dialog.show();

        // 以下是各种对话框的 Demo
        MyButtonClickListener listener = new MyButtonClickListener();
        Button btn1 = (Button) this.findViewById(R.id.btn1);
        btn1.setText("简单的对话框的 Demo");
        btn1.setOnClickListener(listener);

        Button btn2 = (Button) this.findViewById(R.id.btn2);
        btn2.setText("包括常用设置的对话框(数据来自 xml)");
        btn2.setOnClickListener(listener);

        Button btn3 = (Button) this.findViewById(R.id.btn3);
        btn3.setText("弹出的对话框的内容是一个 View");
        btn3.setOnClickListener(listener);

        Button btn4 = (Button) this.findViewById(R.id.btn4);
        btn4.setText("日期选择对话框");
```

```
btn4.setOnClickListener(listener);

Button btn5 = (Button) this.findViewById(R.id.btn5);
btn5.setText("时间选择对话框");
btn5.setOnClickListener(listener);

Button btn6 = (Button) this.findViewById(R.id.btn6);
btn6.setText("进度条对话框");
btn6.setOnClickListener(listener);
}

class MyButtonClickListener implements View.OnClickListener {
    @Override
    public void onClick(View v) {

        // 具体的对话框的实现可以通过重写 onCreateDialog 完成

        switch (v.getId()) {
            case R.id.btn1:
                Main.this.showDialog(0);
                break;
            case R.id.btn2:
                Main.this.showDialog(1);
                break;
            case R.id.btn3:
                Main.this.showDialog(2);
                break;
            case R.id.btn4:
                Main.this.showDialog(3);
                break;
            case R.id.btn5:
                Main.this.showDialog(4);
                break;
            case R.id.btn6:
                Main.this.showDialog(5);
                break;
        }
    }
}

@Override
public Dialog onCreateDialog(int id) {
    switch (id) {
```

```
case 0:
    // 一个简单的弹出对话框
    return new AlertDialog.Builder(this).setTitle("这是一个简单的弹出对话框的 Demo")
        .create();

case 1:
    // 一个相对复杂的弹出对话框
    return new AlertDialog.Builder(this)
        .setTitle("标题") // 设置标题
        // .setCustomTitle(View) // 以一个 View 作为标题
        .setIcon(R.drawable.icon01) // 设置标题图片
        // .setMessage("信息") // 需要显示的弹出内容
        .setPositiveButton("确定", new OnClickListener() { // 设置弹框的确认按钮所显示的文本,以及单击
// 按钮后的响应行为
            @Override
            public void onClick(DialogInterface a0, int a1) {
                TextView txtMsg = (TextView) Main.this.findViewById(R.id.txtMsg);
                txtMsg.append("单击了对话框上的“确定”按钮\n");
            }
        })
        .setItems(R.array.ary, new DialogInterface.OnClickListener() { // 弹框所显示的内容来自一个数组。
// 数组中的数据会一行一行地依次排列
            public void onClick(DialogInterface dialog, int which) {
            }
        })
        // 其他常用方法如下
        // .setMultiChoiceItems(arg0, arg1, arg2)
        // .setSingleChoiceItems(arg0, arg1, arg2)
        // .setNeutralButton(arg0, arg1)
        // .setNegativeButton(arg0, arg1)
        .create();

case 2:
    // 弹出对话框为指定的 View 的 Demo
    return new AlertDialog.Builder(this).setTitle("此对话框的内容是一个 View")
        .setView(this.findViewById(R.layout.view)).create();

case 3:
    // 弹出日期选择对话框
    Calendar c = Calendar.getInstance();
    return new DatePickerDialog(this, new OnDateSetListener() {
        @Override
        public void onDateSet(DatePicker view, int year, int monthOfYear, int dayOfMonth) {
            TextView txtMsg = (TextView) Main.this.findViewById(R.id.txtMsg);
```

```
txtMsg.append("新设置的日期为：" + String.valueOf(year) + "-"
    + String.valueOf(monthOfYear) + "-"
    + String.valueOf(dayOfMonth) + "\n");
}
}, c.get(Calendar.YEAR), c.get(Calendar.MONTH), c.get(Calendar.DATE));

case 4:
    // 弹出时间选择对话框
    Calendar c2 = Calendar.getInstance();
    return new TimePickerDialog(this, new OnTimeSetListener() {
        @Override
        public void onTimeSet(TimePicker view, int hourOfDay, int minute) {
            TextView txtMsg = (TextView) Main.this.findViewById(R.id.txtMsg);
            txtMsg.append("新设置的时间为："
                + String.valueOf(hourOfDay) + ":"
                + String.valueOf(minute) + "\n");
        }
    }, c2.get(Calendar.HOUR), c2.get(Calendar.MINUTE), true);

case 5:
    // 弹出进度条对话框
    ProgressDialog progress = new ProgressDialog(this);
    progress.setMessage("loading...");
    return progress;

default:
    return null;
}
}
```

## 2、各种提示效果的演示

res/layout/main.xml

代码

```
<?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">


    <Button android:id="@+id/btn1" android:layout_width="wrap_content"
        android:layout_height="wrap_content"></Button>
```



```
<Button android:id="@+id/btn2" android:layout_width="wrap_content"
    android:layout_height="wrap_content"></Button>
<Button android:id="@+id/btn3" android:layout_width="wrap_content"
    android:layout_height="wrap_content"></Button>
<Button android:id="@+id/btn4" android:layout_width="wrap_content"
    android:layout_height="wrap_content"></Button>

</LinearLayout>
```

res/layout/view.xml

 代码

```
<?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:id="@+id/txtMsg" android:layout_width="wrap_content"
        android:layout_height="wrap_content">
    </TextView>

</LinearLayout>
```

Main.java

 代码

```
package com.webabcd.notification;

import android.app.Activity;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class Main extends Activity {
```

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

    // 通过 Toast.makeText().show() 来实现提示性的通知效果
    // 短时间的提示性通知的 Demo
    Button btn1 = (Button) this.findViewById(R.id.btn1);
    btn1.setText("短时间提示");
    btn1.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            Toast.makeText(Main.this, "我是短时间提示", Toast.LENGTH_SHORT).show();
        }
    });

    // 长时间的提示性通知的 Demo
    Button btn2 = (Button) this.findViewById(R.id.btn2);
    btn2.setText("长时间提示");
    btn2.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            Toast.makeText(Main.this, "我是长时间提示", Toast.LENGTH_LONG).show();
        }
    });

    // 以一个 View 作为提示性通知的 Demo
    Button btn3 = (Button) this.findViewById(R.id.btn3);
    btn3.setText("以一个 View 做提示");
    btn3.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            View view = inflateView(R.layout.view);
            TextView txtMsg = (TextView) view.findViewById(R.id.txtMsg);
            txtMsg.setText("提示内容");

            Toast toast = new Toast(Main.this);
            toast.setView(view);
            toast.setDuration(Toast.LENGTH_LONG);
            toast.show();
        }
    });

    Button btn4 = (Button) this.findViewById(R.id.btn4);
    btn4.setText("发出一个通知 ( Notification ) ");
}
```

```
btn4.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        // 实例化通知管理器
        NotificationManager nm = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

        // 指定单击通知后所打开的详细的通知页面（单击通知后打开 NotificationView）
        PendingIntent contentIntent = PendingIntent.getActivity(
            Main.this, 0, new Intent(Main.this, NotificationView.class), 0);

        // 实例化一个通知，并指定其图标和标题（在提示栏上显示）
        Notification n = new Notification(R.drawable.icon01, "我是滚动的通知信息我是滚动的通知信息我是滚动的通知信息", System.currentTimeMillis());

        // 设置通知的发送人和通知的详细内容（打开提示栏后在通知列表中显示）
        n.setLatestEventInfo(Main.this, "通知发送人", "我是详细的通知信息我是详细的通知信息我是详细的通知信息", contentIntent);

        // 100 毫秒延迟后，震动 250 毫秒，暂停 100 毫秒后，再震动 500 毫秒
        n.vibrate = new long[] { 100, 250, 100, 500 };

        // 发出通知（其中第一个参数为通知标识符）
        nm.notify(0, n);
    }
});

// 将指定的 xml 资源转换为一个 View
private View inflateView(int resource) {
    LayoutInflater vi = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    return vi.inflate(resource, null);
}

// 打开详细通知页后此 Activity 会被 Pause，从详细通知页返回后此 Activity 会被 Resume
@Override
protected void onPause() {
    // TODO Auto-generated method stub
    super.onPause();

    Log.d("MyDebug", "onPause");
}

@Override
protected void onResume() {
    // TODO Auto-generated method stub
```

```
super.onResume();

Log.d("MyDebug", "onResume");
}
}
```

NotificationView.java

代码

```
package com.webabcd.notification;

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

// 单击通知列表的某个通知后，所打开的详细的通知页
public class NotificationView extends Activity {
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.view);

        TextView txtMsg = (TextView) this.findViewById(R.id.txtMsg);
        txtMsg.setText("点通知之后所链接到的 Activity");

        NotificationManager nm = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        // 取消显示在通知列表中的指定通知（参数为通知标识符）
        nm.cancel(0);

        // 需要关闭此 Activity 的话就 finish 它既可
        // this.finish();
    }
}
```

## Android(4) - 活动(Activity), 服务(Service), 广播(Broadcast), 广播接收器(BroadcastReceiver)

### 介绍

在 Android 中使用 Activity, Service, Broadcast, BroadcastReceiver

活动(Activity) - 用于表现功能

服务(Service) - 相当于后台运行的 Activity

广播(Broadcast) - 用于发送广播

广播接收器(BroadcastReceiver) - 用于接收广播

Intent - 用于连接以上各个组件，并在其间传递消息

1、演示 Activity 的基本用法，一个 Activity 启动另一个 Activity，启动另一个 Activity 时为其传递参数，被启动的 Activity 返回参数给启动者的 Activity

Main.java

代码

```
package com.webabcd.activity;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
```

```
public class Main extends Activity {

    TextView txt;

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

        super.onCreate(savedInstanceState);

        this setContentView(R.layout.main);

        txt = (TextView) this.findViewById(R.id.txt);

        txt.setText("Activity 1");

        Button btn = (Button) this.findViewById(R.id.btn);

        btn.setText("启动另一个 Activity");

        btn.setOnClickListener(new Button.OnClickListener() {

            @Override
            public void onClick(View v) {

                // 实例化 Intent , 指定需要启动的 Activity

                Intent intent = new Intent();
```

```
intent.setClass(Main.this, MyActivity.class);
```

```
// 实例化 Bundle , 设置需要传递的参数
```

```
Bundle bundle = new Bundle();
```

```
bundle.putString("name", "webabcd");
```

```
bundle.putDouble("salary", 100.13);
```

```
// 将需要传递的参数赋值给 Intent 对象
```

```
intent.putExtras(bundle);
```

```
// startActivity(intent); // 启动指定的 Intent ( 不等待返回结果 )
```

```
// Main.this.finish();
```

```
// 启动指定的 Intent , 并等待返回结果
```

```
// 其中第二个参数如果大于等于零 , 则返回结果时会回调
```

onActivityResult() 方法

```
startActivityForResult(intent, 0);
```

```
}
```

```
});
```

```
Log.d("MyDebug", "onCreate");
```

```
}
```

```
// 被启动的 Activity 返回结果时的回调函数

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent
data) {

    if (resultCode == Activity.RESULT_OK){

        Bundle bundle = data.getExtras();

        String name = bundle.getString("name");

        double salary = bundle.getDouble("salary");

        txt.setText("Activity 1" + "\n 名字：" + name + "\n 薪水：" +
String.valueOf(salary));

    }

}

@Override

protected void onStart() {

    // TODO Auto-generated method stub

    super.onStart();

    Log.d("MyDebug", "onStart");
```



```
}
```

```
@Override
```

```
protected void onStop() {
```

```
    // TODO Auto-generated method stub
```

```
    super.onStop();
```

```
    Log.d("MyDebug", "onStop");
```

```
}
```

```
@Override
```

```
protected void onRestart() {
```

```
    // TODO Auto-generated method stub
```

```
    super.onRestart();
```

```
    Log.d("MyDebug", "onRestart");
```

```
}
```

```
@Override
```

```
protected void onPause() {
```

```
    // TODO Auto-generated method stub
```

```
    super.onPause();
```

```
        Log.d("MyDebug", "onPause");
    }

    @Override
    protected void onResume() {
        // TODO Auto-generated method stub
        super.onResume();

        Log.d("MyDebug", "onResume");
    }

    @Override
    protected void onDestroy() {
        // TODO Auto-generated method stub
        super.onDestroy();

        Log.d("MyDebug", "onDestroy");
    }
}
```

MyActivity.java

代码

```
package com.webabcd.activity;

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

// 被另一个 Activity 所启动的 Activity
public class MyActivity extends Activity {

    Intent intent;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        this setContentView(R.layout.main2);
    }
}
```

```
// 获取启动者传递过来的参数

intent = this getIntent();

Bundle bundle = intent.getExtras();

String name = bundle.getString("name");

double salary = bundle.getDouble("salary");


TextView txt = (TextView) this.findViewById(R.id.txt);

txt.setText("Activity 2" + "\n 名字：" + name + "\n 薪水：" +
String.valueOf(salary));


Button btn = (Button) this.findViewById(R.id.btn);

btn.setText("返回前一个 Activity");

btn.setOnClickListener(new Button.OnClickListener() {

    public void onClick(View v) {

        // 返回参数给启动者

        MyActivity.this.setResult(Activity.RESULT_OK, intent);

        MyActivity.this.finish();

    }

});

}

}
```

## AndroidManifest.xml

### 代码

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

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

    package="com.webabcd.activity" android:versionCode="1"

    android:versionName="1.0">

    <application

                                android:icon="@drawable/icon"

        android:label="@string/app_name">

        <activity android:name=".Main" android:label="@string/app_name">

            <intent-filter>

                <action android:name="android.intent.action.MAIN" />

                <category

        android:name="android.intent.category.LAUNCHER" />

            </intent-filter>

        </activity>

        <!--

            如果有需要用到的 Activity ，则都要在这里做相应的配置

        -->

        <activity android:name=".MyActivity" android:label="Activity 2" />

    </application>
```

```
<uses-sdk android:minSdkVersion="3" />

</manifest>
```

## 2、Service, Broadcast, BroadcastReceiver 的演示

Main.java

代码

```
package com.webabcd.service;

import android.app.Activity;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.ServiceConnection;
import android.os.Bundle;
import android.os.IBinder;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.TextView;
```

```
/*  
 * startService() 和 bindService() 的区别  
 * startService() - 正常理解就好  
 * bindService() - 使当前上下文对象（本例中就是 Activity）通过一个  
ServiceConnection 对象绑定到指定的 Service。这样，如果上下文对象销毁了的话，  
那么其对应的 Service 也会被销毁  
 */  
  
public class Main extends Activity implements OnClickListener {  
  
    private TextView txtMsg;  
  
    @Override  
  
    public void onCreate(Bundle savedInstanceState) {  
  
        super.onCreate(savedInstanceState);  
  
        setContentView(R.layout.main);  
  
        setTitle("android 之 service");  
  
        this.findViewById(R.id.btnStart).setOnClickListener(this);  
        this.findViewById(R.id.btnStop).setOnClickListener(this);  
        this.findViewById(R.id.btnBind).setOnClickListener(this);  
    }  
}
```

```
this.findViewById(R.id.btnUnbind).setOnClickListener(this);
```

```
txtMsg = (TextView)findViewById(R.id.txtMsg);
```

```
// 实例化自定义的 BroadcastReceiver
```

```
receiver = new UpdateReceiver();
```

```
IntentFilter filter = new IntentFilter();
```

```
// 为 BroadcastReceiver 指定 action，使之用于接收同 action 的广播
```

```
filter.addAction("com.webabcd.service.msg");
```

```
// 以编程方式注册 BroadcastReceiver。配置方式注册
```

BroadcastReceiver 的例子见 AndroidManifest.xml 文件

```
// 一般在 OnStart 时注册，在 OnStop 时取消注册
```

```
this.registerReceiver(receiver, filter);
```

```
// this.unregisterReceiver(receiver);
```

```
}
```

```
@Override
```

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

```
    Intent intent = new Intent(Main.this, MyService.class);
```

```
    switch (v.getId()) {
```



```
        case R.id.btnStart:

            this.startService(intent);

            break;

        case R.id.btnStop:

            this.stopService(intent);

            break;

        case R.id.btnBind:

            this.bindService(intent, conn, Context.BIND_AUTO_CREATE);

            break;

        case R.id.btnUnbind:

            this.unbindService(conn);

            break;

    }

}
```

// bindService() 所需的 ServiceConnection 对象

```
private ServiceConnection conn = new ServiceConnection() {

    @Override

    public void onServiceConnected(ComponentName className, IBinder

service) {

    }

}
```

```
@Override

    public void onServiceDisconnected(ComponentName className) {

        }

};

private String msg="";

private UpdateReceiver receiver;

// 实现一个 BroadcastReceiver , 用于接收指定的 Broadcast

public class UpdateReceiver extends BroadcastReceiver{

    @Override

    public void onReceive(Context context, Intent intent) {

        msg = intent.getStringExtra("msg");

        txtMsg.append(msg + "\n");

    }

}

}
```

MyService.java

代码

```
package com.webabcd.service;
```

```
import android.app.Service;
```

```
import android.content.Intent;
```

```
import android.os.IBinder;
```

```
import android.util.Log;
```

```
// 演示 Service 的生命周期。具体信息运行程序后在 LogCat 中查看
```

```
public class MyService extends Service {
```

```
    @Override
```

```
    public IBinder onBind(Intent intent) {
```

```
        Log.d("MyDebug", "onBind");
```

```
        sendMsg("onBind");
```

```
        // TODO Auto-generated method stub
```

```
        return null;
```

```
    }
```

@Override

```
public void onCreate() {  
  
    // TODO Auto-generated method stub  
  
    super.onCreate();  
  
    Log.d("MyDebug", "onCreate");  
  
    sendMsg("onCreate");  
}
```

@Override

```
public void onDestroy() {  
  
    // TODO Auto-generated method stub  
  
    super.onDestroy();  
  
    Log.d("MyDebug", "onDestroy");  
  
    sendMsg("onDestroy");  
}
```

@Override

```
public void onRebind(Intent intent) {  
  
    // TODO Auto-generated method stub  
  
    super.onRebind(intent);  
}
```

```
        Log.d("MyDebug", "onRebind");

        sendMsg("onRebind");
    }

    @Override

    public void onStart(Intent intent, int startId) {

        super.onStart(intent, startId);

        Log.d("MyDebug", "onStart");

        sendMsg("onStart");
    }

    @Override

    public boolean onUnbind(Intent intent) {

        Log.d("MyDebug", "onUnbind");

        sendMsg("onUnbind");

        // TODO Auto-generated method stub

        return super.onUnbind(intent);
    }
```

```
// 发送广播信息

private void sendMsg(String msg){

    // 指定广播目标的 action （注：指定了此 action 的 receiver 会接收此广播）

    Intent intent = new Intent("com.webabcd.service.msg");

    // 需要传递的参数

    intent.putExtra("msg", msg);

    // 发送广播

    this.sendBroadcast(intent);

}

}
```

MyBootReceiver.java

代码

```
package com.webabcd.service;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
```

```
import android.util.Log;
```

```
public class MyBootReceiver extends BroadcastReceiver {
```

```
    // 用于接收满足条件的 Broadcast ( 相应的 Broadcast 的注册信息详见  
    AndroidManifest.xml , 当系统启动完毕后会调用这个广播接收器 )
```

```
    @Override
```

```
    public void onReceive(Context arg0, Intent arg1) {
```

```
        Log.d("MyDebug", "onReceive");
```

```
        // 启动服务
```

```
        Intent service = new Intent(arg0, MyService.class);
```

```
        arg0.startService(service);
```

```
    }
```

```
}
```

AndroidManifest.xml

代码

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

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.webabcd.service" android:versionCode="1"
    android:versionName="1.0">
    <application
        android:icon="@drawable/icon"
        android:label="@string/app_name">
        <activity android:name=".Main" android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category
                    android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <!--
            如果有需要用到的 service ，则都要在这里做相应的配置
        -->
        <service android:name=".MyService"></service>

        <!--
            注册一个 BroadcastReceiver
            其 intent-filter 为 android.intent.action.BOOT_COMPLETED ( 用于
            接收系统启动完毕的 Broadcast )
```



```
-->

<receiver android:name=".MyBootReceiver">

    <intent-filter>

        <action

android:name="android.intent.action.BOOT_COMPLETED" />

    </intent-filter>

</receiver>

</application>

<!--

    接受系统启动完毕的 Broadcast 的权限

-->

<uses-permission

android:name="android.permission.RECEIVE_BOOT_COMPLETED" />

    <uses-sdk android:minSdkVersion="3" />

</manifest>
```

Android(5) - 控件(View)之 TextView, Button, ImageButton,  
ImageView, CheckBox, RadioButton, AnalogClock, DigitalClock

## 介绍

在 Android 中使用各种控件(View)

- TextView - 文本显示控件
- Button - 按钮控件
- ImageButton - 图片按钮控件
- ImageView - 图片显示控件
- CheckBox - 复选框控件
- RadioButton - 单选框控件
- AnalogClock - 钟表（带表盘的那种）控件
- DigitalClock - 电子表控件

## 1、TextView 的 Demo

textview.xml

代码

```
<?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 - 文本显示控件

-->

```
<TextView android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content" android:id="@+id/textView" />
```

```
</LinearLayout>
```

\_TextView.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.widget.TextView;
```

```
public class _TextView extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        // TODO Auto-generated method stub
```

```
        super.onCreate(savedInstanceState);
```

```
        this setContentView(R.layout.textview);
```

```
// 设置 Activity 的标题  
  
setTitle("TextView");  
  
TextView txt = (TextView) this.findViewById(R.id.textView);  
  
// 设置文本显示控件的文本内容，需要换行的话就用 “\n”  
  
txt.setText("我是 TextView\n 显示文字用的");  
  
}  
}
```

## 2、Button 的 Demo

button.xml

代码

```
<?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:id="@+id/textView" />
```

```
<!--
```

```
    Button - 按钮控件
```

```
-->
```

```
<Button android:id="@+id/button"
        android:layout_width="wrap_content" android:layout_height="wrap_content">
    </Button>
```

```
</LinearLayout>
```

\_Button.java

代码

```
package com.webabcd.view;

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

public class _Button extends Activity {
```

@Override

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    // TODO Auto-generated method stub
```

```
    super.onCreate(savedInstanceState);
```

```
    this setContentView(R.layout.button);
```

```
    setTitle("Button");
```

```
    Button btn = (Button) this.findViewById(R.id.button);
```

```
    btn.setText("click me");
```

```
    // setOnClickListener() - 响应按钮的鼠标单击事件
```

```
    btn.setOnClickListener(new Button.OnClickListener(){
```

```
        @Override
```

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

```
            TextView txt = (TextView) _Button.this.findViewById(R.id.textView);
```

```
            txt.setText("按钮被单击了");
```

```
        }
```

```
    });
```

```
}
```

```
}
```

### 3、ImageButton 的 Demo

imagebutton.xml

代码

```
<?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:id="@+id/textView" />

    <!--
        ImageButton - 图片按钮控件
    -->

    <ImageButton android:id="@+id/imageButton"
        android:layout_width="wrap_content" android:layout_height="wrap_content">

    </ImageButton>

</LinearLayout>
```

\_ImageButton.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.widget.Button;
```

```
import android.widget.ImageButton;
```

```
import android.widget.TextView;
```

```
public class _ImageButton extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        // TODO Auto-generated method stub
```

```
        super.onCreate(savedInstanceState);
```

```
        this setContentView(R.layout.imagebutton);
```

```
        setTitle("ImageButton");
```



```
ImageButton imgButton = (ImageButton) this.findViewById(R.id.imageBut
ton);

// 设置图片按钮的背景

imgButton.setBackgroundResource(R.drawable.icon01);

// setOnClickListener() - 响应图片按钮的鼠标单击事件

imgButton.setOnClickListener(new Button.OnClickListener(){

    @Override

    public void onClick(View v) {

        TextView txt = (TextView) _ImageButton.this.findViewById(R.id.textVi
ew);

        txt.setText("图片按钮被单击了");

    }

});

}
```

#### 4、ImageView 的 Demo

imageview.xml

代码

```
<?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">

    <!--
        ImageView - 图片显示控件
    -->

    <ImageView android:id="@+id/imageView" android:layout_width="wrap_c
ontent"
        android:layout_height="wrap_content"> </ImageView>

</LinearLayout>
```

\_ImageView.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.widget.ImageView;
```

```
public class _ImageView extends Activity {  
  
    @Override  
  
    protected void onCreate(Bundle savedInstanceState) {  
  
        // TODO Auto-generated method stub  
  
        super.onCreate(savedInstanceState);  
  
        this setContentView(R.layout.imageview);  
  
        setTitle("ImageView");  
  
        ImageView imgView = (ImageView) this.findViewById(R.id.imageView);  
  
        // 指定需要显示的图片  
  
        imgView.setBackgroundResource(R.drawable.icon01);  
  
    }  
}
```

## 5、CheckBox 的 Demo

checkbox.xml

代码

```
<?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:id="@+id/textView" />
```

```
<!--
```

```
    CheckBox - 复选框控件
```

```
-->
```

```
<CheckBox android:text="CheckBox01" android:id="@+id/chk1"
```

```
    android:layout_width="wrap_content" android:layout_height="wrap_content"> </CheckBox>
```

```
<CheckBox android:text="CheckBox02" android:id="@+id/chk2"
```

```
    android:layout_width="wrap_content" android:layout_height="wrap_content"> </CheckBox>
```

```
<CheckBox android:text="CheckBox03" android:id="@+id/chk3"
```

```
    android:layout_width="wrap_content" android:layout_height="wrap_content"> </CheckBox>
```

```
</LinearLayout>
```

```
_CheckBox.java
```

代码

```
package com.webabcd.view;

import android.app.Activity;
import android.os.Bundle;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;

public class _CheckBox extends Activity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        // TODO Auto-generated method stub

        super.onCreate(savedInstanceState);

        this setContentView(R.layout.checkbox);

        setTitle("CheckBox");

        CheckBox chk = (CheckBox) this.findViewById(R.id.chk1);

        // setOnCheckedChangeListener() - 响应复选框的选中状态改变事件

        chk.setOnCheckedChangeListener(new CompoundButton.OnCheckedCha
```

```
ngelListener() {  
  
    @Override  
  
    public void onCheckedChanged(CompoundButton buttonView, boolean  
isChecked) {  
  
        TextView txt = (TextView) _CheckBox.this.findViewById(R.id.textView);  
  
        txt.setText("CheckBox01 的选中状态 :  
" + String.valueOf(isChecked));  
  
    }  
  
    });  
  
}  
  
}
```

## 6、RadioButton 的 Demo

radiobutton.xml

代码

```
<?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:id="@+id/textView" />
```

```
<!--
```

RadioButton - 单选框控件

RadioGroup - 对其内的单选框控件做分组

checkedButton - 指定组内被选中的单选框的 ID

```
-->
```

```
<RadioGroup android:id="@+id/radioGroup"
```

```
    android:layout_width="fill_parent" android:layout_height="fill_parent"
```

```
    android:checkedButton="@+id/rad3" android:orientation="horizontal"
```

```
    android:gravity="center_vertical|center_horizontal">
```

```
    <RadioButton android:text="rad1" android:id="@+id/rad1"
```

```
        android:layout_width="wrap_content" android:layout_height="wrap_co  
ntent"> </RadioButton>
```

```
    <RadioButton android:text="rad2" android:id="@+id/rad2"
```

```
        android:layout_width="wrap_content" android:layout_height="wrap_co  
ntent"> </RadioButton>
```

```
    <RadioButton android:text="rad3" android:id="@+id/rad3"
```

```
        android:layout_width="wrap_content" android:layout_height="wrap_co  
ntent"> </RadioButton>
```

```
</RadioGroup>
```

```
</LinearLayout>
```

\_RadioButton.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.widget.RadioButton;
```

```
import android.widget.RadioGroup;
```

```
import android.widget.TextView;
```

```
public class _RadioButton extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        // TODO Auto-generated method stub
```

```
        super.onCreate(savedInstanceState);
```

```
        this setContentView(R.layout.radiobutton);
```

```
        setTitle("RadioButton");
```



```
RadioGroup group = (RadioGroup) this.findViewById(R.id.radioGroup);

// setOnCheckedChangeListener() - 响应单选框组内的选中项发生变化时的事件

group.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {

    @Override

    public void onCheckedChanged(RadioGroup group, int checkedId) {

        TextView txt = (TextView) _RadioButton.this.findViewById(R.id.textVi

ew);

        txt.setText(((RadioButton)findViewById(checkedId)).getText() + " 被选中");

    }

});

}
```

## 7、AnalogClock 的 Demo

analogclock.xml

代码

```
<?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">

    <!--

        AnalogClock - 钟表（带表盘的那种）控件

    -->

    <AnalogClock android:id="@+id/analogClock"

        android:layout_width="wrap_content" android:layout_height="wrap_cont

ent">

    </AnalogClock>

</LinearLayout>
```

\_AnalogClock.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
public class _AnalogClock extends Activity {  
  
    @Override  
  
    protected void onCreate(Bundle savedInstanceState) {  
  
        // TODO Auto-generated method stub  
  
        super.onCreate(savedInstanceState);  
  
        this setContentView(R.layout.analogclock);  
  
        setTitle("AnalogClock");  
    }  
}
```

## 8、DigitalClock 的 Demo

digitalclock.xml

代码

```
<?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">  
  
    <!--
```

DigitalClock - 电子表控件

-->

```
<DigitalClock android:id="@+id/digitalClock"
    android:layout_width="wrap_content" android:layout_height="wrap_cont
ent">

</DigitalClock>

</LinearLayout>
```

\_DigitalClock.java

代码

```
package com.webabcd.view;

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

public class _DigitalClock extends Activity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        // TODO Auto-generated method stub

        super.onCreate(savedInstanceState);
```

```
this.setContentView(R.layout.digitalClcok);  
  
setTitle("DigitalClcok");  
  
}  
  
}
```

## Android(6) - 控件(View)之 DatePicker, TimePicker, ToggleButton, EditText, ProgressBar, SeekBar, AutoCompleteTextView, MultiAutoCompleteTextView

### 介绍

在 Android 中使用各种控件(View)

- DatePicker - 日期选择控件
- TimePicker - 时间选择控件
- ToggleButton - 双状态按钮控件
- EditText - 可编辑文本控件
- ProgressBar - 进度条控件
- SeekBar - 可拖动的进度条控件
- AutoCompleteTextView - 支持自动完成功能的可编辑文本控件

- MultiAutoCompleteTextView - 支持自动完成功能的可编辑文本控件，允许输入多值（多值之间会自动地用指定的分隔符分开）

## 1、DatePicker 的 Demo

datepicker.xml

代码

```
<?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">

    <!--
        DatePicker - 日期选择控件
    -->

    <DatePicker android:id="@+id/datePicker"
        android:layout_width="wrap_content" android:layout_height="wrap_content">

    </DatePicker>

</LinearLayout>
```

\_DatePicker.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
public class _DatePicker extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        // TODO Auto-generated method stub
```

```
        super.onCreate(savedInstanceState);
```

```
        this setContentView(R.layout.datepicker);
```

```
        // 具体的应用可参见对话框中的示例
```

```
        setTitle("DatePicker");
```

```
    }
```

```
}
```

## 2、TimePicker 的 Demo

timepicker.xml

代码

```
<?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">

    <!--

        TimePicker - 时间选择控件

    -->

    <TimePicker android:id="@+id/timePicker"

        android:layout_width="wrap_content" android:layout_height="wrap_cont

ent">

        </TimePicker>

    </LinearLayout>
```

\_TimePicker.java

代码

```
package com.webabcd.view;
```



```
import android.app.Activity;

import android.os.Bundle;

public class _TimePicker extends Activity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        // TODO Auto-generated method stub

        super.onCreate(savedInstanceState);

        this setContentView(R.layout.timepicker);

        // 具体的应用可参见对话框中的示例

        setTitle("TimePicker");

    }

}
```

### 3、ToggleButton 的 Demo

togglebutton.xml

代码

```
<?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:id="@+id/textView" />
```

```
<!--
```

ToggleButton - 双状态按钮控件

textOn - 当按钮状态为 true 时所显示的文本

textOff - 当按钮状态为 false 时所显示的文本

```
-->
```

```
<ToggleButton android:id="@+id/toggleButton"
```

```
    android:layout_width="wrap_content" android:layout_height="wrap_content"
```

```
    android:textOn="关闭" android:textOff="打开" />
```

```
</LinearLayout>
```

\_ToggleButton.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import android.widget.ToggleButton;


public class _ToggleButton extends Activity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        // TODO Auto-generated method stub

        super.onCreate(savedInstanceState);

        this setContentView(R.layout.togglebutton);

        setTitle("ToggleButton");

        final ToggleButton btn = (ToggleButton) this.findViewById(R.id.toggleButt

on);

        // setOnClickListener() - 响应按钮的鼠标单击事件

        btn.setOnClickListener(new Button.OnClickListener(){

            @Override
```

```
public void onClick(View v) {  
    TextView txt = (TextView) _ToggleButton.this.findViewById(R.id.textV  
iew);  
  
    // ToggleButton.isChecked() - 双状态按钮的按钮状态  
    txt.setText("按钮状态：" + String.valueOf(btn.isChecked()));  
}  
});  
}  
}
```

#### 4、EditText 的 Demo

edittext.xml

代码

```
<?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">  
  
    <!--  
        EditText - 可编辑文本控件  
    -->
```

```
<EditText android:id="@+id/editText" android:layout_width="fill_parent"
    android:layout_height="wrap_content">

</EditText>
```

```
</LinearLayout>
```

\_EditText.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.widget.EditText;
```

```
public class _EditText extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        // TODO Auto-generated method stub
```

```
        super.onCreate(savedInstanceState);
```

```
        this setContentView(R.layout.edittext);
```

```
setTitle("EditText");

EditText txt = (EditText) this.findViewById(R.id.editText);

txt.setText("我可编辑");

}

}
```

## 5、ProgressBar 的 Demo

progressbar.xml

代码

```
<?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">

    <!--

        ProgressBar - 进度条控件

    -->

    <!--以下分别为大、中、小的进度条控件（圆圈状）-->

    <ProgressBar android:id="@+android:id/progress_large"
```

```
style="?android:attr/progressBarStyleLarge" android:layout_width="wrap_
_content"

    android:layout_height="wrap_content" />

<ProgressBar android:id="@+android:id/progress"

    android:layout_width="wrap_content" android:layout_height="wrap_cont
ent" />

<ProgressBar android:id="@+android:id/progress_small"

    style="?android:attr/progressBarStyleSmall" android:layout_width="wrap_
content"

    android:layout_height="wrap_content" />
```

```
<!--
```

进度条控件（条状）的演示

style - 进度条的样式，本例使用内置样式

max - 进度的最大值

progress - 第一进度位置

secondaryProgress - 第二进度位置

```
-->
```

```
<ProgressBar android:id="@+id/progress_horizontal"

    style="?android:attr/progressBarStyleHorizontal" android:layout_width="
200px"

    android:layout_height="wrap_content" android:max="100"
```

```
android:progress="50" android:secondaryProgress="75" />
```

```
</LinearLayout>
```

\_ProgressBar.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.view.Window;
```

```
// 另见对话框中的进度条
```

```
public class _ProgressBar extends Activity {
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    // TODO Auto-generated method stub
```

```
    super.onCreate(savedInstanceState);
```

```
    // 设置特性以允许在应用程序的标题栏上显示进度条（条状）
```



```
requestWindowFeature(Window.FEATURE_PROGRESS);

// 设置特性以允许在应用程序的标题栏上显示进度条（圆圈状）

requestWindowFeature(Window.FEATURE_INDETERMINATE_PROGRESS);

this.setContentView(R.layout.progressbar);

setTitle("ProgressBar");

// 在标题栏上显示进度条（条状）

setProgressBarVisibility(true);

// 在标题栏上显示进度条（圆圈状）

setProgressBarIndeterminateVisibility(true);

// 指定进度条的进度

setProgress(50 * 100);

setSecondaryProgress(75 * 100);

}

}
```

## 6、SeekBar 的 Demo

seekbar.xml

代码

```
<?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">

    <!--

        SeekBar - 可拖动的进度条控件

            max - 进度的最大值

            progress - 第一进度位置

            secondaryProgress - 第二进度位置

    -->

    <SeekBar android:id="@+id/seekBar" android:layout_width="fill_parent"

        android:layout_height="wrap_content" android:max="100"

        android:progress="50" android:secondaryProgress="75" />

    <TextView android:id="@+id/progress" android:layout_width="fill_parent"

        android:layout_height="wrap_content" />

    <TextView android:id="@+id/tracking" android:layout_width="fill_parent"

        android:layout_height="wrap_content" />
```

```
</LinearLayout>
```

\_SeekBar.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.widget.SeekBar;
```

```
import android.widget.TextView;
```

```
public class _SeekBar extends Activity implements
```

```
    SeekBar.OnSeekBarChangeListener {
```

```
    SeekBar mSeekBar;
```

```
    TextView mProgressText;
```

```
    TextView mTrackingText;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    // TODO Auto-generated method stub
```

```
super.onCreate(savedInstanceState);

this setContentView(R.layout.seekbar);

setTitle("SeekBar");

mSeekBar = (SeekBar) findViewById(R.id.seekBar);

// setOnSeekBarChangeListener() - 响应拖动进度条事件
mSeekBar.setOnSeekBarChangeListener(this);

mProgressText = (TextView) findViewById(R.id.progress);
mTrackingText = (TextView) findViewById(R.id.tracking);
}

// 拖动进度条后，进度发生改变时的回调事件
public void onProgressChanged(SearchBar seekBar, int progress,
    boolean fromTouch) {
    mProgressText.setText(progress + "%");
}

// 拖动进度条前开始跟踪触摸
public void onStartTrackingTouch(SearchBar seekBar) {
    mTrackingText.setText("开始跟踪触摸");
}
```

```
}
```

```
// 拖动进度条后停止跟踪触摸
```

```
public void onStopTrackingTouch(SeekBar seekBar) {
```

```
    mTrackingText.setText("停止跟踪触摸");
```

```
}
```

```
}
```

## 7、AutoCompleteTextView 的 Demo

autocompleteTextView.xml

代码

```
<?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">
```

```
<!--
```

```
    AutoCompleteTextView - 支持自动完成功能的可编辑文本控件
```

```
-->
```

```
<AutoCompleteTextView android:id="@+id/editText"
```

```
        android:layout_width="fill_parent" android:layout_height="wrap_content"

    />

</LinearLayout>
```

\_AutoCompleteTextView.java

代码

```
package com.webabcd.view;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;

public class _AutoCompleteTextView extends Activity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setContentView(R.layout.autocompletetextview);

        setTitle("AutoCompleteTextView");
```

```
// 实例化适配器，指定显示格式及数据源

ArrayAdapter<String> adapter = new ArrayAdapter<String>(

    this,

    android.R.layout.simple_dropdown_item_1line,

    ary);

AutoCompleteTextView textView = (AutoCompleteTextView) findViewById(R.id.editText);

// 指定自动完成控件的适配器

textView.setAdapter(adapter);

}

// 自动完成控件的所需数据的数据源

private String[] ary = new String[] {

    "abc",

    "abcd",

    "abcde",

    "abcdef",

    "abcdefg",

    "hij",

    "hijk",

    "hijkl",
```

```
"hijklm",  
"hijklmn",  
};  
}
```

## 8、MultiAutoCompleteTextView 的 Demo

multiautocompletetextview.xml

代码

```
<?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">  
  
    <!--  
        MultiAutoCompleteTextView - 支持自动完成功能的可编辑文本控件 ,允许输入  
        多值 ( 多值之间会自动地用指定的分隔符分开 )  
    -->  
  
    <MultiAutoCompleteTextView android:id="@+id/editText"  
        android:layout_width="fill_parent" android:layout_height="wrap_content"  
    />
```



</LinearLayout>

\_MultiAutoCompleteTextView.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.widget.AdapterView;
```

```
import android.widget.MultiAutoCompleteTextView;
```

```
public class _MultiAutoCompleteTextView extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.multiautocompletetextview);
```

```
        setTitle("MultiAutoCompleteTextView");
```

```
        // 实例化适配器，指定显示格式及数据源
```

```
ArrayAdapter<String> adapter = new ArrayAdapter<String>(
    this,
    android.R.layout.simple_dropdown_item_1line,
    ary);

MultiAutoCompleteTextView textView = (MultiAutoCompleteTextView) findViewById(R.id.editText);

textView.setAdapter(adapter);

// 设置多个值之间的分隔符，此处为逗号
textView.setTokenizer(new MultiAutoCompleteTextView.CommaTokenizer());
}

// 自动完成控件的所需数据的数据源
private String[] ary = new String[] {
    "abc",
    "abcd",
    "abcde",
    "abcdef",
    "abcdefg",
    "hij",
    "hijk",
}
```

```
"hijkl",  
"hijklm",  
"hijklmn",  
};  
}
```

## Android(7) - 控件(View)之 ZoomControls, Include, VideoView, WebView, RatingBar, Tab, Spinner, Chronometer, ScrollView

介绍

在 Android 中使用各种控件(View)

- ZoomControls - 放大/缩小按钮控件
- Include - 整合控件
- VideoView - 视频播放控件
- WebView - 浏览器控件
- RatingBar - 评分控件
- Tab - 选项卡控件
- Spinner - 下拉框控件

- Chronometer - 计时器控件
- ScrollView - 滚动条控件

## 1、ZoomControls 的 Demo

zoomcontrols.xml

代码

```
<?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">

    <!--
        放大/缩小按钮控件
    -->

    <ZoomControls android:id="@+id/zoomControls"
        android:layout_width="wrap_content" android:layout_height="wrap_content"> </ZoomControls>

</LinearLayout>
```

\_ZoomControls.java

代码

```
package com.webabcd.view;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Toast;
import android.widget.ZoomControls;

public class _ZoomControls extends Activity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        // TODO Auto-generated method stub

        super.onCreate(savedInstanceState);

        this setContentView(R.layout.zoomcontrols);

        setTitle("ZoomControls");

        ZoomControls zoomControls = (ZoomControls) this.findViewById(R.id.zoomControls);
```

```
// setOnZoomInClickListener() - 响应单击放大按钮的事件

zoomControls.setOnZoomInClickListener(new OnClickListener() {

    public void onClick(View v) {

        Toast.makeText(_ZoomControls.this, "单击了放大按钮

", Toast.LENGTH_SHORT).show();

    }

});

// setOnZoomOutClickListener() - 响应单击缩小按钮的事件

zoomControls.setOnZoomOutClickListener(new OnClickListener() {

    public void onClick(View v) {

        Toast.makeText(_ZoomControls.this, "单击了缩小按钮

", Toast.LENGTH_SHORT).show();

    }

});

}

}
```

## 2、Include 的 Demo

include.xml

代码

```
<?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">

    <!--

        include - 整合控件，将指定的 layout 整合进来

        layout - 指定需要整合 layout

    -->

    <include android:id="@+id/cell1" layout="@layout/include_1" />

    <include android:id="@+id/cell2" android:layout_width="fill_parent" layout

    ="@layout/include_2" />

</LinearLayout>
```

include\_1.xml

代码

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

<TextView xmlns:android="http://schemas.android.com/apk/res/android"

    android:text="TextView01" android:layout_width="wrap_content"
```

```
android:layout_height="wrap_content">
```

```
</TextView>
```

include\_2.xml

代码

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

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:text="TextView02" android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content">
```

```
</TextView>
```

\_Include.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
public class _Include extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```



```
// TODO Auto-generated method stub

super.onCreate(savedInstanceState);

this setContentView(R.layout.include);

setTitle("Include");

}

}
```

### 3、VideoView 的 Demo

videoview.xml

代码

```
<?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">

    <!--

        VideoView - 视频播放控件

    -->

    <VideoView android:id="@+id/videoView" android:layout_width="wrap_co

ntent"
```

```
        android:layout_height="wrap_content">

        </VideoView>

    </LinearLayout>
```

\_VideoView.java

代码

```
package com.webabcd.view;

import android.app.Activity;
import android.net.Uri;
import android.os.Bundle;
import android.widget.MediaController;
import android.widget.VideoView;

public class _VideoView extends Activity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        // TODO Auto-generated method stub

        super.onCreate(savedInstanceState);

        this setContentView(R.layout.videoview);
```

```
setTitle("VideoView");

VideoView videoView = (VideoView) findViewById(R.id.videoView);

// 指定需要播放的视频的地址
videoView.setVideoURI(Uri.parse("android.resource://com.webabcd.view/"
+ R.raw.demo));

// videoView.setVideoPath();

// 设置播放器的控制条
videoView.setMediaController(new MediaController(this));

// 开始播放视频
videoView.start();

}

}
```

#### 4、WebView 的 Demo

webview.xml

代码

```
<?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">
```

```
<!--
```

```
    WebView - 浏览器控件 ( WebKit 内核 )
```

```
-->
```

```
<WebView android:layout_width="fill_parent"
```

```
    android:layout_height="wrap_content" android:id="@+id/webView" />
```

```
</LinearLayout>
```

```
_WebView.java
```

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.webkit.WebSettings;
```

```
import android.webkit.WebView;
```

```
public class _WebView extends Activity {  
  
    @Override  
  
    protected void onCreate(Bundle savedInstanceState) {  
  
        // TODO Auto-generated method stub  
  
        super.onCreate(savedInstanceState);  
  
        this setContentView(R.layout.webview);  
  
  
        setTitle("WebView");  
  
  
        WebView webView = (WebView) findViewById(R.id.webView);  
  
  
        // 配置浏览器，使其可支持 JavaScript  
  
        WebSettings webSettings = webView.getSettings();  
  
        webSettings.setJavaScriptEnabled(true);  
  
  
        // 清除浏览器缓存  
  
        webView.clearCache(true);  
  
        // 指定浏览器需要解析的 url 地址  
  
        webView.loadUrl("http://webabcd.cnblogs.com/");  
  
        // 指定浏览器需要解析的 html 数据  
  
        // webView.loadData("<a href='http://webabcd.cnblogs.com/'>webabcd
```

```
</a>", "text/html", "utf-8");
```

```
}
```

```
}
```

## 5、RatingBar 的 Demo

ratingbar.xml

代码

```
<?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">
```

```
<!--
```

RatingBar - 评分控件

numStars - 评分控件的星星的数量

rating - 当前评分的值

```
-->
```

```
<RatingBar android:id="@+id/ratingBar" android:numStars="5"
```

```
    android:rating="1.5" android:layout_width="wrap_content"
```

```
    android:layout_height="wrap_content">
```

```
</RatingBar>
```

```
<TextView android:id="@+id/textView" android:layout_width="wrap_content"
android:layout_height="wrap_content" />

</LinearLayout>
```

\_RatingBar.java

代码

```
package com.webabcd.view;

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

public class _RatingBar extends Activity implements RatingBar.OnRatingBarChangeListener {

    private RatingBar mRatingBar;

    private TextView mTextView;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    // TODO Auto-generated method stub
```

```
    super.onCreate(savedInstanceState);
```

```
    this setContentView(R.layout.ratingbar);
```

```
    setTitle("RatingBar");
```

```
    mTextView = (TextView) findViewById(R.id.textView);
```

```
    mRatingBar = (RatingBar) findViewById(R.id.ratingBar);
```

```
    // setOnRatingBarChangeListener() - 响应评分值发生改变的事件
```

```
    mRatingBar.setOnRatingBarChangeListener(this);
```

```
}
```

```
@Override
```

```
public void onRatingChanged(RatingBar ratingBar, float rating,
```

```
    boolean fromUser) {
```

```
    mTextView.setText(String.valueOf(rating));
```

```
}
```

```
}
```



## 6、Tab 的 Demo

tab.xml

代码

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

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent">

    <!-- Tab 1 的内容 -->

    <TextView android:id="@+id/view1" android:layout_width="fill_parent"
        android:layout_height="fill_parent" android:text="tab1 content" />

    <!-- Tab 2 的内容 -->

    <TextView android:id="@+id/view2" android:layout_width="fill_parent"
        android:layout_height="fill_parent" android:text="tab2 content" />

</FrameLayout>
```

\_Tab.java

代码

```
package com.webabcd.view;
```

```
import android.app.TabActivity;
```

```
import android.content.Intent;
```

```
import android.os.Bundle;
```

```
import android.view.LayoutInflater;
```

```
import android.widget.TabHost;
```

```
// 实现 Tab 功能的话要继承 TabActivity
```

```
public class _Tab extends TabActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        // TODO Auto-generated method stub
```

```
        super.onCreate(savedInstanceState);
```

```
        TabHost tabHost = getTabHost();
```

```
        LayoutInflater.from(this).inflate(R.layout.tab, tabHost.getTabContentView()  
, true);
```

```
        // Tab 1 的内容
```

```
        tabHost.addTab(tabHost.newTabSpec("tab1")
```

```
.setIndicator("tab1")

.setContent(R.id.view1));

// Tab 2 的内容 ( 设置了 Tab 图片 )

tabHost.addTab(tabHost.newTabSpec("tab2")

.setIndicator("tab2", getResources().getDrawable(R.drawable.icon01))

.setContent(R.id.view2));

// Tab 3 的内容 ( 设置 Tab 的内容为指定的 Activity )

tabHost.addTab(tabHost.newTabSpec("tab3")

.setIndicator("tab3")

.setContent(new Intent(this, _TextView.class)));

}

}
```

## 7、Spinner 的 Demo

spinner.xml

代码

```
<?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:id="@+id/textView" />
```

```
<!--
```

Spinner - 下拉框控件

```
-->
```

```
<Spinner android:id="@+id/spinner" android:layout_width="fill_parent"
```

```
android:layout_height="wrap_content" />
```

```
</LinearLayout>
```

\_Spinner.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.widget.AdapterView;
```

```
import android.widget.AdapterView;

import android.widget.Spinner;

import android.widget.TextView;


public class _Spinner extends Activity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        // TODO Auto-generated method stub

        super.onCreate(savedInstanceState);

        this setContentView(R.layout.spinner);

        setTitle("Spinner");

        Spinner spinner = (Spinner) findViewById(R.id.spinner);

        // 设置下拉框控件的标题文本

        spinner.setPrompt("请选择");

        // 实例化适配器，指定显示格式及数据源

        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResou

rce(

        this, R.array.colors, android.R.layout.simple_spinner_item);
```

```
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
```

```
spinner.setAdapter(adapter);
```

```
// setSelectedListener() - 响应下拉框的选中值发生变化的事件
```

```
spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
```

```
    @Override
```

```
    public void onItemSelected(AdapterView<?> arg0, View arg1,
```

```
        int arg2, long arg3) {
```

```
        TextView textView = (TextView)_Spinner.this.findViewById(R.id.textView);
```

```
        textView.setText(((TextView)arg1).getText());
```

```
    }
```

```
    @Override
```

```
    public void onNothingSelected(AdapterView<?> arg0) {
```

```
    }
```

```
});
```

```
}
```

```
}
```

## 8、Chronometer 的 Demo

chronometer.xml

代码

```
<?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">

    <!--
        Chronometer - 计时器控件
    -->

    <Chronometer android:id="@+id/chronometer"
        android:layout_width="wrap_content" android:layout_height="wrap_content" />

    <Button android:id="@+id/btnStart" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="开始计时">
        <requestFocus />
    </Button>
```

```
<Button android:id="@+id/btnStop" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="停止计时">

</Button>
```

```
<Button android:id="@+id/btnReset" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:text="计时器复位">

</Button>
```

```
</LinearLayout>
```

\_Chronometer.java

代码

```
package com.webabcd.view;

import android.app.Activity;
import android.os.Bundle;
import android.os.SystemClock;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Chronometer;
```



```
public class _Chronometer extends Activity {  
  
    private Chronometer mChronometer;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        // TODO Auto-generated method stub  
        super.onCreate(savedInstanceState);  
        this setContentView(R.layout.chronometer);  
  
        setTitle("Chronometer");  
  
        Button button;  
  
        mChronometer = (Chronometer) findViewById(R.id.chronometer);  
        // 设置计时器所显示的时间格式  
        mChronometer.setFormat("计时 : (%s)");  
  
        button = (Button) findViewById(R.id.btnStart);  
        button.setOnClickListener(mStartListener);  
  
        button = (Button) findViewById(R.id.btnStop);
```

```
button.setOnClickListener(mStopListener);
```

```
button = (Button) findViewById(R.id.btnReset);
```

```
button.setOnClickListener(mResetListener);
```

```
}
```

```
View.OnClickListener mStartListener = new OnClickListener() {
```

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

```
        // 启动计时器
```

```
        mChronometer.start();
```

```
    }
```

```
};
```

```
View.OnClickListener mStopListener = new OnClickListener() {
```

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

```
        // 暂停计时器
```

```
        mChronometer.stop();
```

```
    }
```

```
};
```

```
View.OnClickListener mResetListener = new OnClickListener() {
```

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

```
// 复位计时器，即停止计时器
```

```
mChronometer.setBase(SystemClock.elapsedRealtime());
```

```
}
```

```
};
```

```
}
```

## 9、ScrollView 的 Demo

scrollview.xml

代码

```
<?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">
```

```
<!--
```

ScrollView - 滚动条控件

scrollbarStyle - 滚动条的样式

```
-->
```

```
<ScrollView android:id="@+id/scrollView"
```

```
    android:layout_width="fill_parent" android:layout_height="200px"
```

```
    android:scrollbarStyle="outsideOverlay" android:background="@android:
```

```
drawable/edit_text">
```

```
<TextView android:layout_width="fill_parent"
```

```
android:layout_height="wrap_content" android:id="@+id/textView" />
```

```
</ScrollView>
```

```
</LinearLayout>
```

\_ScrollView.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.widget.TextView;
```

```
public class _ScrollView extends Activity {
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
// TODO Auto-generated method stub
```

```
super.onCreate(savedInstanceState);
```

```
this setContentView(R.layout.scrollview);
```



## 1、TextSwitcher 的 Demo

textswitcher.xml

代码

```
<?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">

    <Button android:id="@+id/btnChange" android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:text="改变文字" />

    <!--
    TextSwitcher - 文字转换器控件 ( 改变文字时增加一些动画效果 )
    -->

    <TextSwitcher android:id="@+id/textSwitcher"
        android:layout_width="fill_parent" android:layout_height="wrap_content"
    />

</LinearLayout>
```

\_TextSwitcher.java

代码

```
package com.webabcd.view;
```

```
import java.util.Random;
```

```
import android.app.Activity;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.view.animation.Animation;
```

```
import android.view.animation.AnimationUtils;
```

```
import android.widget.Button;
```

```
import android.widget.TextSwitcher;
```

```
import android.widget.TextView;
```

```
import android.widget.ViewSwitcher;
```

```
public class _TextSwitcher extends Activity implements ViewSwitcher.ViewFact
```

```
ory {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        // TODO Auto-generated method stub
```

```
        super.onCreate(savedInstanceState);
```

```
this.setContentView(R.layout.textswithcer);

setTitle("TextSwithcer");

final TextSwitcher switcher = (TextSwitcher) findViewById(R.id.textSwitche
r);

// 指定转换器的 ViewSwitcher.ViewFactory
switcher.setFactory(this);

// 设置淡入和淡出的动画效果
Animation in = AnimationUtils.loadAnimation(this, android.R.anim.fade_in)
;
Animation out = AnimationUtils.loadAnimation(this, android.R.anim.fade_
out);

switcher.setInAnimation(in);
switcher.setOutAnimation(out);

// 单击一次按钮改变一次文字
Button btnChange = (Button) this.findViewById(R.id.btnChange);
btnChange.setOnClickListener(new View.OnClickListener() {

    @Override

    public void onClick(View v) {
```



```
switcher.setText(String.valueOf(new Random().nextInt()));  
  
}  
  
});  
  
}
```

// 重写 ViewSwitcher.ViewFactory 的 makeView() , 返回一个 View

@Override

```
public View makeView() {  
  
    TextView textView = new TextView(this);  
  
    textView.setTextSize(36);  
  
    return textView;  
  
}  
  
}
```

## 2、Gallery 的 Demo

gallery.xml

代码

```
<?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">
```

```
<!--
```

Gallery - 缩略图浏览器控件

spacing - 缩略图列表中各个缩略图之间的间距

```
-->
```

```
<Gallery android:id="@+id/gallery" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:spacing="20px" />
```

```
</LinearLayout>
```

\_Gallery.java

代码

```
package com.webabcd.view;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
```

```
import android.widget.ImageView;

import android.widget.Toast;

import android.widget.Gallery.LayoutParams;


public class _Gallery extends Activity {

    @Override

    protected void onCreate(Bundle savedInstanceState) {

        // TODO Auto-generated method stub

        super.onCreate(savedInstanceState);

        this setContentView(R.layout.gallery);


        setTitle("Gallery");


        Gallery gallery = (Gallery) findViewById(R.id.gallery);

        // 为缩略图浏览器指定一个适配器

        gallery.setAdapter(new ImageAdapter(this));

        // 响应 在缩略图列表上选中某个缩略图后的 事件

        gallery.setOnItemClickListener(new AdapterView.OnItemClickListener() {

            @Override

            public void onItemClick(AdapterView<?> parent, View v,
```

```
        int position, long id) {  
  
            Toast.makeText(_Gallery.this, String.valueOf(position), Toast.LENGTH  
_SHORT).show();  
  
        }
```

```
        @Override  
  
        public void onNothingSelected(AdapterView<?> arg0) {  
  
        }  
  
    });  
}
```

// 继承 BaseAdapter 用以实现自定义的图片适配器

```
public class ImageAdapter extends BaseAdapter {
```

```
    private Context mContext;
```

```
    public ImageAdapter(Context context) {  
  
        mContext = context;  
  
    }
```

```
    public int getCount() {
```

```
        return mThumbIds.length;
    }

    public Object getItem(int position) {
        return position;
    }

    public long getItemId(int position) {
        return position;
    }

    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView image = new ImageView(mContext);

        image.setImageResource(mThumbIds[position]);
        image.setAdjustViewBounds(true);
        image.setLayoutParams(new Gallery.LayoutParams(
            LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));

        return image;
    }
}
```

// 需要显示的图片集合

```
private Integer[] mThumbIds = { R.drawable.icon01, R.drawable.icon02,  
    R.drawable.icon03, R.drawable.icon04, R.drawable.icon05 };  
}
```

### 3、ImageSwitcher 的 Demo

imageswitcher.xml

代码

```
<?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">  
  
    <Gallery android:id="@+id/gallery" android:layout_width="fill_parent"  
        android:layout_height="wrap_content" android:spacing="20px" />  
  
    <!--  
        ImageSwitcher - 图片转换器控件 ( 改变图片时增加一些动画效果 )  
    -->  
  
    <ImageSwitcher android:id="@+id/imageSwitcher"
```

```
        android:layout_width="fill_parent" android:layout_height="wrap_content"

    />

</LinearLayout>
```

\_ImageSwitcher.java

代码

```
package com.webabcd.view;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.ViewSwitcher;
import android.widget.Gallery.LayoutParams;
```

// 图片转换器的使用基本同文字转换器

// 以下是一个用 ImageSwitcher + Gallery 实现的经典的图片浏览器的 Demo

```
public class _ImageSwitcher extends Activity implements
```

```
    ViewSwitcher.ViewFactory {
```

```
    private ImageSwitcher mSwitcher;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        // TODO Auto-generated method stub
```

```
        super.onCreate(savedInstanceState);
```

```
        this setContentView(R.layout.imageswithcer);
```

```
        setTitle("ImageSwithcer");
```

```
        mSwitcher = (ImageSwitcher) findViewById(R.id.imageSwitcher);
```

```
        mSwitcher.setFactory(this);
```

```
        mSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_in));
```

```
        mSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
            android.R.anim.fade_out));
```



```
Gallery gallery = (Gallery) findViewById(R.id.gallery);

gallery.setAdapter(new ImageAdapter(this));

gallery.setOnItemClickListener(new AdapterView.OnItemClickListener() {

    @Override

    public void onItemClick(AdapterView<?> parent, View v,

        int position, long id) {

        mSwitcher.setImageResource(mImageIds[position]);

    }

    @Override

    public void onNothingSelected(AdapterView<?> arg0) {

    }

});

}

public class ImageAdapter extends BaseAdapter {

    private Context mContext;
```

```
public ImageAdapter(Context context) {  
    mContext = context;  
}  
  
public int getCount() {  
    return mThumbIds.length;  
}  
  
public Object getItem(int position) {  
    return position;  
}  
  
public long getItemId(int position) {  
    return position;  
}  
  
public View getView(int position, View convertView, ViewGroup parent) {  
    ImageView image = new ImageView(mContext);  
  
    image.setImageResource(mThumbIds[position]);  
    image.setAdjustViewBounds(true);  
    image.setLayoutParams(new Gallery.LayoutParams(  

```

```
LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
```

```
    return image;
}
}
```

```
private Integer[] mThumbIds = { R.drawable.icon01, R.drawable.icon02,
    R.drawable.icon03, R.drawable.icon04, R.drawable.icon05 };
```

```
private Integer[] mImageIds = { R.drawable.icon01, R.drawable.icon02,
    R.drawable.icon03, R.drawable.icon04, R.drawable.icon05 };
```

```
@Override
```

```
public View makeView() {
    ImageView image = new ImageView(this);
    image.setMinimumHeight(200);
    image.setMinimumWidth(200);
    image.setScaleType(ImageView.ScaleType.FIT_CENTER);
    image.setLayoutParams(new ImageSwitcher.LayoutParams(
        LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));
    return image;
}
```

```
}  
  
}
```

#### 4、GridView 的 Demo

gridview.xml

代码

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

```
<!--
```

GridView - 网格控件

numColumns="auto\_fit" - 列数自适应

stretchMode - 缩放模式 ( stretchMode="columnWidth" - 缩放与列宽大小同步 )

```
-->
```

```
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:id="@+id/gridView" android:layout_width="fill_parent"
```

```
    android:layout_height="fill_parent" android:padding="10px"
```

```
    android:verticalSpacing="10px" android:horizontalSpacing="10px"
```

```
    android:numColumns="auto_fit" android:columnWidth="60px"
```

```
    android:stretchMode="columnWidth" android:gravity="center">
```

```
</GridView>
```

\_GridView.java

代码

```
package com.webabcd.view;
```

```
import android.app.Activity;
```

```
import android.content.Context;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.view.ViewGroup;
```

```
import android.widget.BaseAdapter;
```

```
import android.widget.GridView;
```

```
import android.widget.ImageView;
```

```
public class _GridView extends Activity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        // TODO Auto-generated method stub
```

```
        super.onCreate(savedInstanceState);
```

```
        this setContentView(R.layout.gridview);
```

```
setTitle("GridView");

GridView gridView = (GridView) findViewById(R.id.gridView);

// 指定网格控件的适配器为自定义的图片适配器
gridView.setAdapter(new ImageAdapter(this));
}

// 自定义的图片适配器
public class ImageAdapter extends BaseAdapter {

    private Context mContext;

    public ImageAdapter(Context context) {
        mContext = context;
    }

    public int getCount() {
        return mThumbIds.length;
    }

    public Object getItem(int position) {
        return position;
    }
}
```

```
}
```

```
public long getItemId(int position) {
```

```
    return position;
```

```
}
```

```
public View getView(int position, View convertView, ViewGroup parent) {
```

```
    ImageView imageView;
```

```
    if (convertView == null) {
```

```
        imageView = new ImageView(mContext);
```

```
        imageView.setLayoutParams(new GridView.LayoutParams(48, 48));
```

```
        imageView.setAdjustViewBounds(false);
```

```
        imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
```

```
        imageView.setPadding(5, 5, 5, 5);
```

```
    } else {
```

```
        imageView = (ImageView) convertView;
```

```
    }
```

```
    imageView.setImageResource(mThumbIds[position]);
```

```
    return imageView;
```

```
}
```

```
// 网格控件所需图片数据的数据源

private Integer[] mThumbIds = { R.drawable.icon01, R.drawable.icon02,
    R.drawable.icon03, R.drawable.icon04, R.drawable.icon05 };

}

}
```

## 5、ListView 的 Demo

main\_list\_adapter.xml

代码

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

<!--
    自定义列表适配器的 layout
-->

<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:id="@+id/text" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:textSize="16sp">

</TextView>
```



```
</LinearLayout>
```

MainListAdapter.java

代码

```
package com.webabcd.view;
```

```
import java.util.List;
```

```
import android.content.Context;
```

```
import android.view.LayoutInflater;
```

```
import android.view.View;
```

```
import android.view.ViewGroup;
```

```
import android.widget.BaseAdapter;
```

```
import android.widget.TextView;
```

```
// 继承 BaseAdapter 以实现自定义的列表适配器
```

```
public class MainListAdapter extends BaseAdapter {
```

```
    private LayoutInflater mInflater;
```

```
    private List<String> mData;
```

```
public MainListAdapter(Context context, List<String> data) {  
    mInflater = LayoutInflater.from(context);  
    mData = data;  
}  
  
@Override  
public int getCount() {  
    return mData.size();  
}  
  
@Override  
public Object getItem(int position) {  
    return mData.get(position);  
}  
  
@Override  
public long getItemId(int position) {  
    return position;  
}  
  
@Override  
public View getView(int position, View convertView, ViewGroup parent) {
```

```
TextView text;

if (convertView == null) {

    // 指定一个 layout 作为自定义列表适配器的 layout

    convertView = mInflater.inflate(R.layout.main_list_adapter, null);

    text = (TextView) convertView.findViewById(R.id.text);

    convertView.setTag(text);

} else {

    text = (TextView) convertView.getTag();

}

String mItem = mData.get(position);

text.setText(mItem);

return convertView;

}

}
```

Main.java

代码

```
package com.webabcd.view;

import java.util.ArrayList;

import java.util.List;

import android.app.ListActivity;

import android.content.Intent;

import android.os.Bundle;

import android.view.View;

import android.widget.ListView;

// 此处要继承 ListActivity , 用以实现 ListView 的功能

public class Main extends ListActivity {

    private List<String> mData;

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

    @Override

    public void onCreate(Bundle savedInstanceState) {

        super.onCreate(savedInstanceState);

        setTheme(android.R.style.Theme_Light);
```

```
setContentView(R.layout.main);

mData = getData();

// 使用自定义的列表适配器来展现数据

MainListAdapter adapter = new MainListAdapter(this, mData);

// 如需使用系统内置的列表适配器，则可以使用类似如下的方法

// ArrayAdapter<String> adapter = new ArrayAdapter<String>(this, andr
oid.R.layout.simple_expandable_list_item_1, mData);

this.setListAdapter(adapter);
}

// ListView 的数据源

private List<String> getData() {

    List<String> items = new ArrayList<String>();

    items.add("TextView");

    items.add("Button");

    items.add("ImageButton");

    items.add("ImageView");

    items.add("CheckBox");
```

```
items.add("RadioButton");

items.add("AnalogClock");

items.add("DigitalClock");

items.add("DatePicker");

items.add("TimePicker");

items.add("ToggleButton");

items.add("EditText");

items.add("ProgressBar");

items.add("SeekBar");

items.add("AutoCompleteTextView");

items.add("MultiAutoCompleteTextView");

items.add("ZoomControls");

items.add("Include");

items.add("VideoView");

items.add("WebView");

items.add("RatingBar");

items.add("Tab");

items.add("Spinner");

items.add("Chronometer");

items.add("ScrollView");

items.add("TextSwitcher");

items.add("ListView");
```

```
        items.add("Gallery");

        items.add("ImageSwitcher");

        items.add("GridView");

        items.add("ExpandableList");

        return items;
    }

    // ListView 中某项被选中后的逻辑

    @Override

    protected void onItemClick(ListView l, View v, int position, long id) {

        Intent intent = new Intent();

        intent.setClassName(this, "com.webabcd.view._" + mData.get(position));

        startActivityForResult(intent, 0);

    }

}
```

## 6、ExpandableList 的 Demo

\_ExpandableList.java

代码

```
package com.webabcd.view;
```

```
import android.app.ExpandableListActivity;
```

```
import android.os.Bundle;
```

```
import android.view.ContextMenu;
```

```
import android.view.Gravity;
```

```
import android.view.MenuItem;
```

```
import android.view.View;
```

```
import android.view.ViewGroup;
```

```
import android.view.ContextMenu.ContextMenuInfo;
```

```
import android.widget.AbsListView;
```

```
import android.widget.BaseExpandableListAdapter;
```

```
import android.widget.ExpandableListAdapter;
```

```
import android.widget.ExpandableListView;
```

```
import android.widget.TextView;
```

```
import android.widget.Toast;
```

```
import android.widget.ExpandableListView.ExpandableListContextMenuInfo;
```

```
// ExpandableList - 可展开/收缩列表
```

```
// 继承 ExpandableListActivity 以实现列表的可展开/收缩的功能
```

```
public class _ExpandableList extends ExpandableListActivity {
```



```
private ExpandableListAdapter mAdapter;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
// TODO Auto-generated method stub
```

```
super.onCreate(savedInstanceState);
```

```
setTitle("ExpandableList");
```

```
mAdapter = new MyExpandableListAdapter();
```

```
setListAdapter(mAdapter);
```

```
registerForContextMenu(this.getExpandableListView());
```

```
}
```

```
// 为列表的每一项创建上下文菜单（即长按后呼出的菜单）
```

```
@Override
```

```
public void onCreateContextMenu(ContextMenu menu, View v,
```

```
ContextMenuInfo menuInfo) {
```

```
menu.setHeaderTitle("ContextMenu");
```

```
menu.add(0, 0, 0, "ContextMenu");
```

```
}
```

```
// 单击上下文菜单后的逻辑
```

```
@Override
```

```
public boolean onContextItemSelected(MenuItem item) {
```

```
    ExpandableListContextMenuInfo info = (ExpandableListContextMenuInfo)
    item.getMenuInfo();
```

```
    String title = ((TextView) info.targetView).getText().toString();
```

```
    int type = ExpandableListView.getPackedPositionType(info.packedPosition);
```

```
    if (type == ExpandableListView.PACKED_POSITION_TYPE_CHILD) {
```

```
        int groupPos = ExpandableListView.getPackedPositionGroup(info.packedPosition);
```

```
        int childPos = ExpandableListView.getPackedPositionChild(info.packedPosition);
```

```
        Toast.makeText(this, title + " - Group Index: " + groupPos + " Child Index: " + childPos, Toast.LENGTH_SHORT).show();
```

```
        return true;
```

```
    } else if (type == ExpandableListView.PACKED_POSITION_TYPE_GROUP) {
```

```
        int groupPos = ExpandableListView.getPackedPositionGroup(info.packedPosition);
```

```
Toast.makeText(this, title + " - Group Index: " + groupPos, Toast.LENGTH_SHORT).show();
```

```
return true;
```

```
}
```

```
return false;
```

```
}
```

```
public class MyExpandableListAdapter extends BaseExpandableListAdapter {
```

```
// 父列表数据
```

```
private String[] groups =
```

```
{
```

```
"group1",
```

```
"group2",
```

```
"group3",
```

```
"group4"
```

```
};
```

```
// 子列表数据
```

```
private String[][] children =
```

```
{
```

```
{ "child1" },  
  
{ "child1", "child2" },  
  
{ "child1", "child2", "child3" },  
  
{ "child1", "child2", "child3", "child4" }  
  
};
```

```
@Override
```

```
public Object getChild(int groupPosition, int childPosition) {  
  
    return children[groupPosition][childPosition];  
  
}
```

```
@Override
```

```
public long getChildId(int groupPosition, int childPosition) {  
  
    return childPosition;  
  
}
```

```
@Override
```

```
public int getChildrenCount(int groupPosition) {  
  
    return children[groupPosition].length;  
  
}
```

```
// 取子列表中的某一项的 View
```

```
@Override
```

```
public View getChildView(int groupPosition, int childPosition,  
    boolean isLastChild, View convertView, ViewGroup parent) {  
    TextView textView = getGenericView();  
    textView.setText(getChild(groupPosition, childPosition).toString());  
    return textView;  
}
```

```
@Override
```

```
public Object getGroup(int groupPosition) {  
    return groups[groupPosition];  
}
```

```
@Override
```

```
public int getGroupCount() {  
    return groups.length;  
}
```

```
@Override
```

```
public long getGroupId(int groupPosition) {  
    return groupPosition;  
}
```

```
// 取父列表中的某一项的 View
```

```
@Override
```

```
public View getGroupView(int groupPosition, boolean isExpanded,
```

```
View convertView, ViewGroup parent) {
```

```
    TextView textView = getGenericView();
```

```
    textView.setText(getGroup(groupPosition).toString());
```

```
    return textView;
```

```
}
```

```
@Override
```

```
public boolean hasStableIds() {
```

```
    return true;
```

```
}
```

```
@Override
```

```
public boolean isChildSelectable(int groupPosition, int childPosition) {
```

```
    return true;
```

```
}
```

```
// 获取某一项的 View 的逻辑
```

```
private TextView getGenericView() {
```

```
AbsListView.LayoutParams lp = new AbsListView.LayoutParams(  
    ViewGroup.LayoutParams.FILL_PARENT, 48);  
  
TextView textView = new TextView(_ExpandableList.this);  
textView.setLayoutParams(lp);  
textView.setGravity(Gravity.CENTER_VERTICAL | Gravity.LEFT);  
textView.setPadding(32, 0, 0, 0);  
  
return textView;  
}  
}  
}
```

## Android(9) - 数据库支持(SQLite), 内容提供器(ContentProvider)

### 介绍

在 Android 中使用 SQLite, ContentProvider

- 数据库支持(SQLite) - Android 开发平台提供了操作 SQLite 数据库的相关 API
- 内容提供器(ContentProvider) - 当数据需要在应用程序之间共享时，可以在某程序中使用 ContentProvider 定义 URI，以使其它应用程序可以通过此 URI 访问指定的数据

### 1、SQLite 的 Demo

DatabaseHelper.java

代码

```
package com.webabcd.SQLite;
```

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteDatabase.CursorFactory;

// 数据库操作的 Helper 类
public class DatabaseHelper extends SQLiteOpenHelper {

    DatabaseHelper(Context context, String name, CursorFactory cursorFactory, int version) {
        super(context, name, cursorFactory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // TODO 创建数据库后，对数据库的操作
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // TODO 更改数据库版本的操作
    }

    @Override
    public void onOpen(SQLiteDatabase db) {
        super.onOpen(db);

        // TODO 每次成功打开数据库后首先被执行
    }
}
```

Main.java

代码

```
package com.webabcd.SQLite;

import java.util.Random;

import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```



```
import android.widget.TextView;

public class Main extends Activity {

    private DatabaseHelper dbHelper;

    private static final String DATABASE_NAME = "db.db";
    private static final int DATABASE_VERSION = 1;
    private static final String TABLE_NAME = "employee";

    TextView txtMsg;

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

        dbHelper = new DatabaseHelper(this, DATABASE_NAME, null,
            DATABASE_VERSION);

        txtMsg = (TextView) this.findViewById(R.id.txtMsg);

        Button btn1 = (Button) this.findViewById(R.id.btn1);
        btn1.setText("创建表");
        btn1.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                CreateTable();
            }
        });

        Button btn2 = (Button) this.findViewById(R.id.btn2);
        btn2.setText("插入 3 条记录");
        btn2.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                insertItem();
            }
        });

        Button btn3 = (Button) this.findViewById(R.id.btn3);
        btn3.setText("删除全部记录");
        btn3.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                deleteItem();
            }
        });
    }
}
```

```
    }
    });

    Button btn4 = (Button) this.findViewById(R.id.btn4);
    btn4.setText("更新指定数据");
    btn4.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            updateItem();
        }
    });

    Button btn5 = (Button) this.findViewById(R.id.btn5);
    btn5.setText("显示全部数据");
    btn5.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            showItems();
        }
    });

    Button btn6 = (Button) this.findViewById(R.id.btn6);
    btn6.setText("删除表");
    btn6.setOnClickListener(new Button.OnClickListener() {
        public void onClick(View v) {
            dropTable();
        }
    });
}

// 创建数据表
private void CreateTable() {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    String sql = "CREATE TABLE IF NOT EXISTS " + TABLE_NAME
        + " (ID INTEGER PRIMARY KEY, Name VARCHAR, Age INTEGER);";
    try {
        db.execSQL(sql);
        txtMsg.append("数据表成功创建\n");
    } catch (SQLException ex) {
        txtMsg.append("数据表创建错误\n" + ex.toString() + "\n");
    }
}

// 插入数据
private void insertItem() {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
```

```
try {
    Random random = new Random();
    for (int i = 0; i < 3; i++) {
        String sql = "insert into " + TABLE_NAME
            + " (name, age) values ('name" + String.valueOf(i)
            + "', " + random.nextInt() + ")";
        // execSQL() - 执行指定的 sql
        db.execSQL(sql);
    }
    txtMsg.append("成功插入 3 条数据\n");
} catch (SQLException ex) {
    txtMsg.append("插入数据失败\n" + ex.toString() + "\n");
}

// 删除数据
private void deleteItem() {
    try {
        SQLiteDatabase db = dbHelper.getWritableDatabase();
        db.delete(TABLE_NAME, " id < 999999", null);
        txtMsg.append("成功删除数据\n");
    } catch (SQLException e) {
        txtMsg.append("删除数据失败\n");
    }
}

// 更新数据
private void updateItem() {
    SQLiteDatabase db = dbHelper.getWritableDatabase();

    try {
        ContentValues values = new ContentValues();
        values.put("name", "批量更新后的名字");

        db.update(TABLE_NAME, values, "id < ?", new String[] { "3" });
        txtMsg.append("成功更新数据\n");
    } catch (SQLException e) {
        txtMsg.append("更新数据失败\n");
    }
}

// 查询数据
private void showItems() {
```

```
SQLiteDatabase db = dbHelper.getReadableDatabase();

try {
    String[] column = { "id", "name", "age" };
    Cursor cursor = db.query(TABLE_NAME, column, null, null, null,
        null, null);
    Integer num = cursor.getCount();
    txtMsg.append("共 " + Integer.toString(num) + " 条记录\n");
    cursor.moveToFirst();

    while (cursor.getPosition() != cursor.getCount()) {
        txtMsg.append(Integer.toString(cursor.getPosition()) + ", "
            + String.valueOf(cursor.getString(0)) + ", "
            + cursor.getString(1) + ", "
            + String.valueOf(cursor.getString(2)) + "\n");
        cursor.moveToNext();
    }
} catch (SQLException ex) {
    txtMsg.append("读取数据失败\n" + ex.toString() + "\n");
}

// 删除数据表
private void dropTable() {
    SQLiteDatabase db = dbHelper.getWritableDatabase();
    String sql = "DROP TABLE IF EXISTS " + TABLE_NAME;
    try {
        db.execSQL(sql);
        txtMsg.append("数据表删除成功\n");
    } catch (SQLException ex) {
        txtMsg.append("数据表删除错误\n" + ex.toString() + "\n");
    }
}
}
```

## 2、ContentProvider 的 Demo

MyUser.java

代码

```
package com.webabcd.contentprovider;

import android.net.Uri;
import android.provider.BaseColumns;
```

```
// 自定义 ContentProvider 所需的实体类
public class MyUser {

    // 必须要有 _id 字段。本例中 BaseColumn 类中已经包含了 _id 字段
    public static final class User implements BaseColumns {

        // 定义 CONTENT_URI
        public static final Uri CONTENT_URI = Uri.parse("content://com.webabcd.MyContentProvider");

        // 表数据列
        public static final String USER_NAME = "USER_NAME";
    }
}
```

MyContentProvider.java

代码

```
package com.webabcd.contentprovider;

import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;

import org.apache.http.util.EncodingUtils;

import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.MatrixCursor;
import android.net.Uri;

// 继承 ContentProvider 以实现自定义的 ContentProvider ( 基于文件的信息存储 )
public class MyContentProvider extends ContentProvider {

    private File file;
    private FileOutputStream out;
    private FileInputStream in;

    // ContentProvider 的删除数据接口
    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        // TODO Auto-generated method stub
        return 0;
    }
}
```

```
@Override
public String getType(Uri uri) {
    // TODO Auto-generated method stub
    return null;
}

// ContentProvider 的插入数据接口
@Override
public Uri insert(Uri uri, ContentValues values) {

    try {
        out = new FileOutputStream(file);
        out.write(values.getAsString(MyUser.User.USER_NAME).getBytes());
        out.close();

        int rowId = 0;
        Uri rowUri = ContentUris.appendId(
            MyUser.User.CONTENT_URI.buildUpon(), rowId).build();
        getContext().getContentResolver().notifyChange(rowUri, null);

        return rowUri;
    } catch (Exception e) {
        return null;
    }
}

// 创建用于保存信息的文件
@Override
public boolean onCreate() {
    try {
        // 每个包中应用程序的私有目录为 : /data/data/包名/
        // SD 卡目录为 : /sdcard
        file = new File("/data/data/com.webabcd.contentprovider/",
            "demo.txt");
        if (!file.exists())
            file.createNewFile();

        return true;
    } catch (Exception ex) {
        return false;
    }
}
```

```
// ContentProvider 的查询数据接口
@Override
public Cursor query(Uri uri, String[] projection, String selection,
    String[] selectionArgs, String sortOrder) {

    String content;

    try {
        in = new FileInputStream(file);
        int length = (int) file.length();
        byte[] buffer = new byte[length];
        in.read(buffer, 0, length);
        content = EncodingUtils.getString(buffer, "UTF-8");
        in.close();

        String[] columns = new String[] { MyUser.User._ID, MyUser.User.USER_NAME };
        MatrixCursor cur = new MatrixCursor(columns);
        String[] values = new String[] { "0", content };
        cur.moveToFirst();
        cur.addRow(values);

        return cur;
    } catch (Exception e) {
        return null;
    }
}

// ContentProvider 的更新数据接口
@Override
public int update(Uri uri, ContentValues values, String selection,
    String[] selectionArgs) {
    // TODO Auto-generated method stub
    return 0;
}
}
```

Main.java

代码

```
package com.webabcd.contentprovider;

import java.util.Random;

import android.app.Activity;
import android.content.ContentUris;
```

```
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.Contacts;
import android.provider.Contacts.People;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;

/*
 * 几个常用的系统内置的 ContentProvider 如下：
 * content://media/internal/images 这个 URI 将返回设备上存储的所有图片
 * content://contacts/people/ 这个 URI 将返回设备上的所有联系人信息
 * content://contacts/people/45 这个 URI 返回单个结果（联系人信息中 ID 为 45 的联系人记录）
 */
public class Main extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Button btn1 = (Button) this.findViewById(R.id.btn1);
        btn1.setText("新增联系人记录");
        btn1.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                Random random = new Random();
                insertRecords("name" + String.valueOf(random.nextInt()), String
                    .valueOf(random.nextInt()));
            }
        });

        Button btn2 = (Button) this.findViewById(R.id.btn2);
        btn2.setText("查看联系人记录");
        btn2.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                displayRecords();
            }
        });

        Button btn3 = (Button) this.findViewById(R.id.btn3);
        btn3.setText("清除联系人记录");
    }
}
```



```
btn3.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        deleteRecords();
    }
});

Button btn4 = (Button) this.findViewById(R.id.btn4);
btn4.setText("更新联系人记录");
btn4.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        // 此处只是演示，id 来自 People_ID，可参见 displayRecords() 是如何获取 id 的
        int id = 0;
        updateRecord(id, "修改后的 name");
    }
});

Button btn5 = (Button) this.findViewById(R.id.btn5);
btn5.setText("新增记录到 MyContentProvider");
btn5.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        insertRecord2MyContentProvider("webabcd");
    }
});

Button btn6 = (Button) this.findViewById(R.id.btn6);
btn6.setText("获取记录从 MyContentProvider");
btn6.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        displayRecord2MyContentProvider();
    }
});
}

// 调用 ContentProvider 的插入接口
private void insertRecords(String name, String phoneNum) {
    ContentValues values = new ContentValues();
    values.put(People.NAME, name);
    Uri uri = getContentResolver().insert(People.CONTENT_URI, values);
    Log.d("MyDebug", uri.toString());
    Uri numberUri = Uri.withAppendedPath(uri,
        People.Phones.CONTENT_DIRECTORY);
    Log.d("MyDebug", numberUri.toString());

    values.clear();
}
```

```
values.put(Contacts.Phones.TYPE, People.Phones.TYPE_MOBILE);
values.put(People.NUMBER, phoneNum);
getContentResolver().insert(numberUri, values);
}

// 调用 ContentProvider 的查询接口
private void displayRecords() {
    String[] columns = new String[] { People._ID, People.NAME,
        People.NUMBER };
    Uri contacts = People.CONTENT_URI;
    Log.d("MyDebug", contacts.toString());
    Cursor cur = managedQuery(contacts, columns, // 要返回的数据字段
        null, // WHERE 子句
        null, // WHERE 子句的参数
        null // Order-by 子句
    );

    if (cur.moveToFirst()) {
        String id = null;
        String name = null;
        String phoneNo = null;
        while (cur.getPosition() != cur.getCount()) {
            id = cur.getString(cur.getColumnIndex(People._ID));
            name = cur.getString(cur.getColumnIndex(People.NAME));
            phoneNo = cur.getString(cur.getColumnIndex(People.NUMBER));

            Toast.makeText(this, id + " / " + name + " / " + phoneNo,
                Toast.LENGTH_SHORT).show();
            cur.moveToNext();
        }
    }
}

// 调用 ContentProvider 的删除接口
private void deleteRecords() {
    Uri uri = People.CONTENT_URI;
    Log.d("MyDebug", uri.toString());
    getContentResolver().delete(uri, null, null);
    // getContentResolver().delete(uri, "NAME=" + "name", null);
}

// 调用 ContentProvider 的更新接口
private void updateRecord(int recordNo, String name) {
    Uri uri = ContentUris.withAppendedId(People.CONTENT_URI, recordNo);
```

```

Log.d("MyDebug", uri.toString());
ContentValues values = new ContentValues();
values.put(People.NAME, name);
getContentResolver().update(uri, values, null, null);
}

// 调用自定义 ContentProvider 的插入接口
private void insertRecord2MyContentProvider(String name) {
    ContentValues values = new ContentValues();
    values.put(MyUser.User.USER_NAME, name);
    getContentResolver().insert(MyUser.User.CONTENT_URI, values);
}

// 调用自定义 ContentProvider 的查询接口
private void displayRecord2MyContentProvider() {
    String[] columns = new String[] { MyUser.User.USER_NAME };
    Uri uri = MyUser.User.CONTENT_URI;
    Cursor cur = managedQuery(uri, columns, null, null, null);

    while (cur.getPosition() != cur.getCount()) {
        String id = cur.getString(cur.getColumnIndex(People.ID));
        String name = cur.getString(cur.getColumnIndex(MyUser.User.USER_NAME));
        Toast.makeText(this,
            id + " / " + name,
            Toast.LENGTH_SHORT).show();
        cur.moveToNext();
    }
}
}

```

AndroidManifest.xml

代码

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.webabcd.contentprovider"
    android:versionCode="1"
    android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".Main"
            android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

```

```
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>

<!--
    配置一个自定义的 ContentProvider"
-->
<provider android:name="MyContentProvider" android:authorities="com.webabcd.MyContentProvider" />
</application>
<uses-permission android:name="android.permission.WRITE_CONTACTS"></uses-permission>
<uses-permission android:name="android.permission.READ_CONTACTS"></uses-permission>
<uses-sdk android:minSdkVersion="3" />
</manifest>
```

## Android(10) - HTTP 通信, XML 解析, 通过 Handler 实现异步消息处理

介绍

在 Android 中与服务端做 HTTP 通信，解析 XML，通过 Handler 实现异步消息处理

- HTTP 通信 - 与服务端做 HTTP 通信，分别以 GET 方式和 POST 方式做演示
- XML 解析 - 可以用两种方式解析 XML，分别是 DOM 方式和 SAX 方式
- 异步消息处理 - 通过 Handler 实现异步消息处理，以一个自定义的异步下载类来说明 Handler 的用法

### 1、HTTP 通信和 XML 解析的 Demo

MySAXHandler.java

代码

```
package com.webabcd.communication;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

// 继承 DefaultHandler 以实现指定 XML 的 SAX 解析器
// DOM - W3C 标准，需要把 xml 数据全部加载完成后才能对其做解析，可对树做任意遍历
// SAX - 流式解析，通过事件模型解析 xml，只能顺序解析
public class MySAXHandler extends DefaultHandler {

    private boolean mIsTitleTag = false;
```

```
private boolean mIsSalaryTag = false;
private boolean mIsBirthTag = false;
private String mResult = "";

// 打开 xml 文档的回调函数
@Override
public void startDocument() throws SAXException {
    // TODO Auto-generated method stub
    super.startDocument();
}

// 关闭 xml 文档的回调函数
@Override
public void endDocument() throws SAXException {
    // TODO Auto-generated method stub
    super.endDocument();
}

// 一发现元素开始标记就回调此函数
@Override
public void startElement(String uri, String localName, String qName,
    Attributes attributes) throws SAXException {
    if (localName == "title")
        mIsTitleTag = true;
    else if (localName == "salary")
        mIsSalaryTag = true;
    else if (localName == "dateOfBirth")
        mIsBirthTag = true;
    else if (localName == "employee")
        mResult += "\nname:" + attributes.getValue("name");
}

// 一发现元素结束标记就回调此函数
@Override
public void endElement(String uri, String localName, String qName)
    throws SAXException {
    if (localName == "title")
        mIsTitleTag = false;
    else if (localName == "salary")
        mIsSalaryTag = false;
    else if (localName == "dateOfBirth")
        mIsBirthTag = false;
}
```

```
// 一发现元素值或属性值就回调此函数
@Override
public void characters(char[] ch, int start, int length)
    throws SAXException {
    if (mIsTitleTag)
        mResult += new String(ch, start, length);
    else if (mIsSalaryTag)
        mResult += " salary:" + new String(ch, start, length);
    else if (mIsBirthTag)
        mResult += " dateOfBirth:" + new String(ch, start, length);
}

public String getResult(){
    return mResult;
}
}
```

Main.java

代码

```
package com.webabcd.communication;

import java.io.BufferedInputStream;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.net.URLConnection;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
```

```
import org.apache.http.protocol.HTTP;
import org.apache.http.util.ByteArrayBuffer;
import org.apache.http.util.EncodingUtils;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;

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

public class Main extends Activity {

    private TextView textView;

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

        textView = (TextView) this.findViewById(R.id.textView);

        Button btn1 = (Button) this.findViewById(R.id.btn1);
        btn1.setText("http get demo");
        btn1.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                httpGetDemo();
            }
        });

        Button btn2 = (Button) this.findViewById(R.id.btn2);
        btn2.setText("http post demo");
        btn2.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                httpPostDemo();
            }
        });

        Button btn3 = (Button) this.findViewById(R.id.btn3);
```

```
// DOM - Document Object Model
btn3.setText("DOM 解析 XML");
btn3.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        DOMDemo();
    }
});

Button btn4 = (Button) this.findViewById(R.id.btn4);
// SAX - Simple API for XML
btn4.setText("SAX 解析 XML");
btn4.setOnClickListener(new Button.OnClickListener() {
    public void onClick(View v) {
        SAXDemo();
    }
});
}

// Android 调用 http 协议的 get 方法
// 本例：以 http 协议的 get 方法获取远程页面响应的内容
private void httpGetDemo(){
    try {
        // 模拟器测试时，请使用外网地址
        URL url = new URL("http://xxx.xxx.xxx");
        URLConnection con = url.openConnection();

        String result = "http status code: " + ((URLConnection)con).getResponseCode() + "\n";
        // HttpURLConnection.HTTP_OK

        InputStream is = con.getInputStream();
        BufferedInputStream bis = new BufferedInputStream(is);
        ByteBuffer bab = new ByteBuffer(32);
        int current = 0;
        while ( (current = bis.read()) != -1 ){
            bab.append((byte)current);
        }
        result += EncodingUtils.getString(bab.toByteArray(), HTTP.UTF_8);

        bis.close();
        is.close();

        textView.setText(result);
    } catch (Exception e) {
        textView.setText(e.toString());
    }
}
```



```
}  
}  
  
// Android 调用 http 协议的 post 方法  
// 本例：以 http 协议的 post 方法向远程页面传递参数，并获取其响应的内容  
private void httpPostDemo(){  
    try {  
        // 模拟器测试时，请使用外网地址  
        String url = "http://5billion.com.cn/post.php";  
        Map<String, String> data = new HashMap<String, String>();  
        data.put("name", "webabcd");  
        data.put("salary", "100");  
  
        DefaultHttpClient httpClient = new DefaultHttpClient();  
        HttpPost httpPost = new HttpPost(url);  
        ArrayList<BasicNameValuePair> postData = new ArrayList<BasicNameValuePair>();  
        for (Map.Entry<String, String> m : data.entrySet()) {  
            postData.add(new BasicNameValuePair(m.getKey(), m.getValue()));  
        }  
  
        UrlEncodedFormEntity entity = new UrlEncodedFormEntity(postData, HTTP.UTF_8);  
        httpPost.setEntity(entity);  
  
        HttpResponse response = httpClient.execute(httpPost);  
  
        String result = "http status code: " + response.getStatusLine().getStatusCode() + "\n";  
        // HttpURLConnection.HTTP_OK  
  
        HttpEntity httpEntity = response.getEntity();  
  
        InputStream is = httpEntity.getContent();  
        result += convertStreamToString(is);  
  
        textView.setText(result);  
    } catch (Exception e) {  
        textView.setText(e.toString());  
    }  
}  
  
// 以 DOM 方式解析 XML ( xml 数据详见 res/raw/employee.xml )  
private void DOMDemo(){  
    try {  
        DocumentBuilderFactory docFactory = DocumentBuilderFactory.newInstance();  
        DocumentBuilder docBuilder = docFactory.newDocumentBuilder();
```

```
Document doc = docBuilder.parse(this.getResources().openRawResource(R.raw.employee));
Element rootElement = doc.getDocumentElement();
NodeList employeeNodeList = rootElement.getElementsByTagName("employee");

textView.setText("DOMDemo" + "\n");
String title = rootElement.getElementsByTagName("title").item(0).getFirstChild().getNodeValue();
textView.append(title);
for (int i=0; i<employeeNodeList.getLength(); i++){
    Element employeeElement = (Element)employeeNodeList.item(i);
    String name = employeeElement.getAttribute("name");
    String salary = employeeElement.getElementsByTagName("salary").item(0).getFirstChild().getNodeValue();
    String dateOfBirth = employeeElement.getElementsByTagName("dateOfBirth").item(0).getFirstChild().getNodeValue();
    textView.append("\nname: " + name + " salary: " + salary + " dateOfBirth: " + dateOfBirth);
}
} catch (Exception e) {
    textView.setText(e.toString());
}
}

// 以 SAX 方式解析 XML ( xml 数据详见 res/raw/employee.xml )
// SAX 解析器的实现详见 MySAXHandler.java
private void SAXDemo(){
    try {
        SAXParserFactory saxFactory = SAXParserFactory.newInstance();
        SAXParser parser = saxFactory.newSAXParser();
        XMLReader reader = parser.getXMLReader();

        MySAXHandler handler = new MySAXHandler();
        reader.setContentHandler(handler);
        reader.parse(new InputSource(this.getResources().openRawResource(R.raw.employee)));
        String result = handler.getResult();
        textView.setText("SAXDemo" + "\n");
        textView.append(result);
    } catch (Exception e) {
        textView.setText(e.toString());
    }
}

// 辅助方法，用于把流转换为字符串
private String convertStreamToString(InputStream is) {
    BufferedReader reader = new BufferedReader(new InputStreamReader(is));
    StringBuilder sb = new StringBuilder();
```

```
String line = null;
try {
    while ((line = reader.readLine()) != null) {
        sb.append(line + "\n");
    }
} catch (IOException e) {
    e.printStackTrace();
} finally {
    try {
        is.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

return sb.toString();
}
}
```

2、用 Handler 来实现异步消息处理，以一个可以实时汇报下载进度的异步下载类为例  
开发一个 Android 类库，本例中此类库名为 webabcd\_util

New -> Java Project

项目上点右键 -> Build Path -> Add Libraries -> User Library -> User Libraries -> New -> 为类库起个名字 -> 选中这个类库 -> Add JARs 导入 Android 的 jar 包

项目上点右键 -> Build Path -> Add Libraries -> User Library -> 选择 Android 库

DownloadManagerAsync.java

代码

```
package webabcd.util;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileOutputStream;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.URL;
import java.net.URLConnection;

import org.apache.http.protocol.HTTP;

import android.os.Handler;
```

```
import android.os.Message;
import android.util.Log;

// 以一个实例，即异步下载，来演示 Android 的异步消息处理（用 Handler 的方式）
public class DownloadManagerAsync {

    public DownloadManagerAsync() {

    }

    // 实例化自定义的 Handler
    EventHandler mHandler = new EventHandler(this);

    // 按指定 url 地址下载文件到指定路径
    public void download(final String url, final String savePath) {
        new Thread(new Runnable() {
            public void run() {
                try {
                    sendMessage(FILE_DOWNLOAD_CONNECT);
                    URL sourceUrl = new URL(url);
                    URLConnection conn = sourceUrl.openConnection();
                    InputStream inputStream = conn.getInputStream();

                    int fileSize = conn.getContentLength();

                    File savefile = new File(savePath);
                    if (savefile.exists()) {
                        savefile.delete();
                    }
                    savefile.createNewFile();

                    FileOutputStream outputStream = new FileOutputStream(
                        savePath, true);

                    byte[] buffer = new byte[1024];
                    int readCount = 0;
                    int readNum = 0;
                    int prevPercent = 0;
                    while (readCount < fileSize && readNum != -1) {
                        readNum = inputStream.read(buffer);
                        if (readNum > -1) {
                            outputStream.write(buffer);
                            readCount = readCount + readNum;
                        }
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        }).start();
    }
}
```

```
        int percent = (int) (readCount * 100 / fileSize);
        if (percent > prevPercent) {
            // 发送下载进度信息
            sendMessage(FILE_DOWNLOAD_UPDATE, percent,
                readCount);

            prevPercent = percent;
        }
    }
}

outputStream.close();
sendMessage(FILE_DOWNLOAD_COMPLETE, savePath);

} catch (Exception e) {
    sendMessage(FILE_DOWNLOAD_ERROR, e);
    Log.e("MyError", e.toString());
}
}
}).start();
}

// 读取指定 url 地址的响应内容
public void download(final String url) {
    new Thread(new Runnable() {
        public void run() {
            try {
                sendMessage(FILE_DOWNLOAD_CONNECT);
                URL sourceUrl = new URL(url);
                URLConnection conn = sourceUrl.openConnection();
                conn.setConnectTimeout(3000);
                BufferedReader reader = new BufferedReader(
                    new InputStreamReader(conn.getInputStream(),
                        HTTP.UTF_8));

                String line = null;
                StringBuffer content = new StringBuffer();
                while ((line = reader.readLine()) != null) {
                    content.append(line);
                }

                reader.close();

                sendMessage(FILE_DOWNLOAD_COMPLETE, content.toString());
            }
        }
    }).start();
}
```

```
        } catch (Exception e) {
            sendMessage(FILE_DOWNLOAD_ERROR, e);
            Log.e("MyError", e.toString());
        }
    }
}).start();
}

// 向 Handler 发送消息
private void sendMessage(int what, Object obj) {
    // 构造需要向 Handler 发送的消息
    Message msg = mHandler.obtainMessage(what, obj);
    // 发送消息
    mHandler.sendMessage(msg);
}

private void sendMessage(int what) {
    Message msg = mHandler.obtainMessage(what);
    mHandler.sendMessage(msg);
}

private void sendMessage(int what, int arg1, int arg2) {
    Message msg = mHandler.obtainMessage(what, arg1, arg2);
    mHandler.sendMessage(msg);
}

private static final int FILE_DOWNLOAD_CONNECT = 0;
private static final int FILE_DOWNLOAD_UPDATE = 1;
private static final int FILE_DOWNLOAD_COMPLETE = 2;
private static final int FILE_DOWNLOAD_ERROR = -1;

// 自定义的 Handler
private class EventHandler extends Handler {
    private DownloadManagerAsync mManager;

    public EventHandler(DownloadManagerAsync manager) {
        mManager = manager;
    }

    // 处理接收到的消息
    @Override
    public void handleMessage(Message msg) {
```

```
switch (msg.what) {
case FILE_DOWNLOAD_CONNECT:
    if (mOnDownloadConnectListener != null)
        mOnDownloadConnectListener.onDownloadConnect(mManager);
    break;
case FILE_DOWNLOAD_UPDATE:
    if (mOnDownloadUpdateListener != null)
        mOnDownloadUpdateListener.onDownloadUpdate(mManager,
            msg.arg1);
    break;
case FILE_DOWNLOAD_COMPLETE:
    if (mOnDownloadCompleteListener != null)
        mOnDownloadCompleteListener.onDownloadComplete(mManager,
            msg.obj);
    break;
case FILE_DOWNLOAD_ERROR:
    if (mOnDownloadErrorListener != null)
        mOnDownloadErrorListener.onDownloadError(mManager,
            (Exception) msg.obj);
    break;
default:
    break;
}
}
}

// 定义连接事件
private OnDownloadConnectListener mOnDownloadConnectListener;
public interface OnDownloadConnectListener {
    void onDownloadConnect(DownloadManagerAsync manager);
}
public void setOnDownloadConnectListener(OnDownloadConnectListener listener) {
    mOnDownloadConnectListener = listener;
}

// 定义下载进度更新事件
private OnDownloadUpdateListener mOnDownloadUpdateListener;
public interface OnDownloadUpdateListener {
    void onDownloadUpdate(DownloadManagerAsync manager, int percent);
}
public void setOnDownloadUpdateListener(OnDownloadUpdateListener listener) {
    mOnDownloadUpdateListener = listener;
}
```

```
// 定义下载完成事件
private OnDownloadCompleteListener mOnDownloadCompleteListener;
public interface OnDownloadCompleteListener {
    void onDownloadComplete(DownloadManagerAsync manager, Object result);
}
public void setOnDownloadCompleteListener(
    OnDownloadCompleteListener listener) {
    mOnDownloadCompleteListener = listener;
}

// 定义下载异常事件
private OnDownloadErrorListener mOnDownloadErrorListener;
public interface OnDownloadErrorListener {
    void onDownloadError(DownloadManagerAsync manager, Exception e);
}
public void setOnDownloadErrorListener(OnDownloadErrorListener listener) {
    mOnDownloadErrorListener = listener;
}
}
```

调用上面的自定义的 Android 类库

项目上点右键 -> Properties -> Java Build Path -> Projects -> Add 引用上面的类库

Main.java

代码

```
package com.webabcd.handler;

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

import webabcd.util.DownloadManagerAsync;

public class Main extends Activity implements
    DownloadManagerAsync.OnDownloadCompleteListener,
    DownloadManagerAsync.OnDownloadUpdateListener,
    DownloadManagerAsync.OnDownloadErrorListener {

    TextView txt;

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



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

DownloadManagerAsync manager = new DownloadManagerAsync();
manager.setOnDownloadCompleteListener(this);
manager.setOnDownloadUpdateListener(this);
manager.download("http://files.cnblogs.com/webabcd/Android.rar", "/sdcard/Android.rar");

txt = (TextView) this.findViewById(R.id.txt);
txt.setText("开始下载");
}

public void onDownloadComplete(DownloadManagerAsync manager, Object result) {

    txt.setText("下载完成");
}

public void onDownloadUpdate(DownloadManagerAsync manager, int percent) {

    txt.setText("下载进度 : " + String.valueOf(percent) + "%");
}

public void onDownloadError(DownloadManagerAsync manager, Exception e) {

    txt.setText("下载出错");
}
}
```

OK

[\[源码下载\]](#)