Android UI 设计

无意中看到的几篇文章，想翻译出来分享给大家。不过声明，翻译后的意思不一定能完全表达作者的意图，如果想看原文，请参考：http://mobiforge.com/designing/story/understanding-user-interface-android-part-1-layouts

到目前为止，我之前的几篇关于 Android 的文章都集中于向你展示如何解决 Android 中的问题，而没有花太多的时间来讨论 Android 应用程序开发的视觉元素——UI 设计。在这篇和接下来的文章，我将带你穿越构建 Android 应用程序的 UI 元素。文章的开始部分，我将讨论 Android 中的一些布局（Layouts）和一些在屏幕上摆放的构件（Widget）。

Android 屏幕 UI 组件

到这个时点，你已经看到 Android 应用程序最基本的单元式 Activity。Activity 用于显示应用程序的 UI，它可能包含许多构件，如 buttons, labels, text boxes 等。一般，你会使用一个 XML 文件（例如，位于 res/layout 文件夹下的 main.xml 文件）来定义你的 UI，它看起来像这样：

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

在运行时，你在 Activity 的 onCreate 事件处理函数里加载 XML UI，使用 Activity 类的 setContentView 方法：

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

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

在编译期间，XML 文件中的元素会编译成相应地 Android GUI 类，并设定了指定的特性。当加载时，Android 系统会创建 Activity 的 UI。

使用 XML 文件来构建 UI 往往是比较容易的，然后，也存在一些时候需要你在运行时动态地构建 UI（例如，当编写游戏）。因此，也有可能完全通过代码来创建的你的 UI。

Views 和 ViewGroups

一个 Activity 包含 View 和 ViewGroup。一个 View 是一个构件，它在屏幕上有一个外观。构件包括 buttons, labels, text boxes 等。一个 View 继承自 android.view.View 基类。

一个或多个 View 可以组合起来放入一个 ViewGroup。一个 ViewGroup（它是特殊类型的 View）提供一个布局，在其上你可以安排 View 的显示和次序。ViewGroup 包括 LinearLayout, FrameLayout 等。一个 ViewGroup 继承自 android.view.ViewGroup 基类。

Android 支持以下的 ViewGroup：

- LinearLayout
- AbsoluteLayout
- TableLayout
- RelativeLayout
- FrameLayout
- ScrollView

接下来的章节将讨论每个 ViewGroup 的细节。注意，在练习中，通常会嵌套不同类型的布局来创建想要的 UI。

创建一个简单工程

创建一个新的 Android 工程，如图 1 所示命名它。

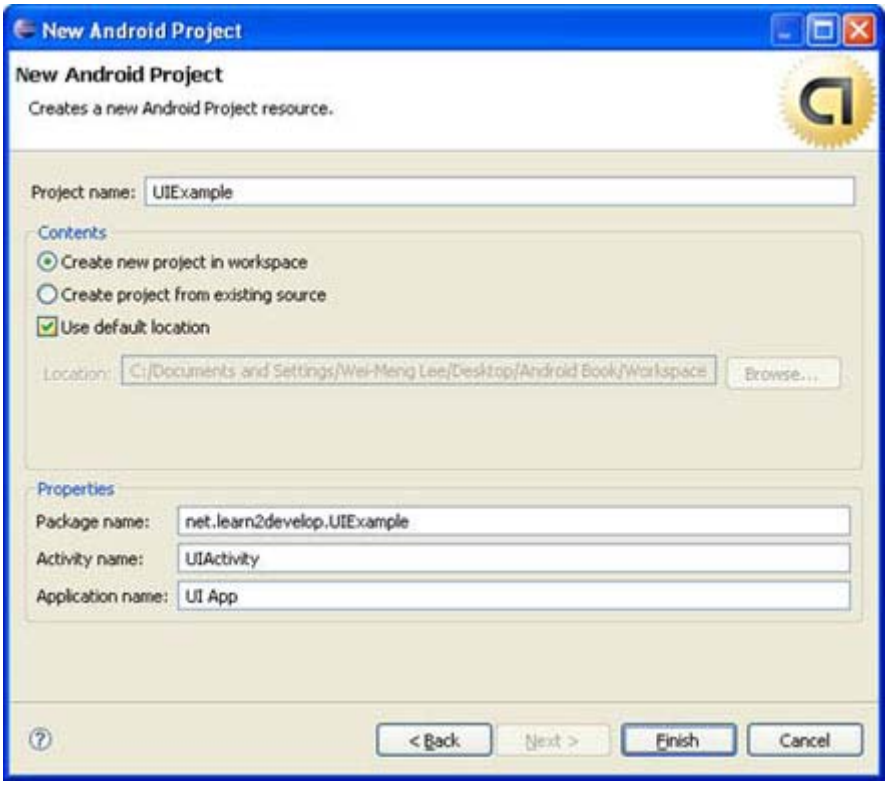

图 1　使用 Eclipse 来创建一个新的 Android 工程

Eclipse 支持很少的 Android UI 设计，因此，你不能在设计面板上进行构件的拖拽。作为替代的，你可以使用免费的 DroidDraw 工具（从 http://www.droiddraw.org/ 获取）。图 2 显示了动作中的 DroidDraw。你可以在不同的布局上拖拽构件，然后使用它来生成等价的 XML 代码。然后 DroidDraw 不是很完美，对于你刚开始 Android 的 UI 设计来说还是非常有用的，并且它是学习 Android 中众多的 View 和 ViewGroup 的得力工具。

图 2 DroidDraw 网页应用程序来设计你的 AndroidUI

你还可以下载独立的 DroidDraw 版本（Windows, Mac OS X, Linux）。

LinearLayout

LinearLayout 以单一的行或列来布局 View。子 View 既可以垂直摆放也可以水平摆放。查看 LinearLayout 如何工作，让我们修改一下工程的 main.xml 文件：

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

```
        android:text="@string/hello"
        />
</LinearLayout>
```

在 main.xml 文件中，可以看到根元素是，并且它包含一个元素。元素控制着其内元素的顺序和显示。

每个 View 和 ViewGroup 都有一堆通用的特性，其中一部分显示在表 1 中。

特性	描述
layout_width	指定 View 或 ViewGroup 的宽度
layout_height	指定 View 或 ViewGroup 的高度
layout_marginTop	指定 View 或 ViewGroup 上方的空间
layout_marginBottom	指定 View 或 ViewGroup 下方的空间
layout_marginLeft	指定 View 或 ViewGroup 左边的空间
layout_marginRight	指定 View 或 ViewGroup 右边的空间
layout_gravity	指定子 View 如何摆放
layout_weight	指定 Layout 的剩余空间的多少分配给 View
layout_x	指定 View 或 ViewGroup 的 x 坐标
layout_y	指定 View 或 ViewGroup 的 y 坐标

表 1 View 和 ViewGroup 的通用特性

注意，这些特性中的部分只在当一个 View 位于某个特定的 ViewGroup 里才能应用。例如，layout_weight 和 layout_gravity 特性只能当 View 位于 LinearLayout 或者 TableLayout 里才能应用。

举个例子，上面的<TextView>元素使用 fill_parent 常量填充了父元素的全部宽度（在这里是屏幕）。它的高度使用 wrap_content 常量，表示它的高度依据内容的高度（在这里，是其内的文本）。如果确实不想让<TextView>View 占据整个行，那么，你也可以设定 layout_width 特性为 wrap_content，像这样：

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

这样设置意味著 View 的宽度与其内包含的文本的宽度一样，你还可以设置宽度为一个绝对的值，像这样：

```
<TextView
    android:layout_width="105px"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
```

在这里，宽度设置为 105 像素宽。让我们修改 main.xml 文件，添加一个<Button>View 吧，如下所示：

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    >
    <TextView
        android:layout_width="105px"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <Button
        android:layout_width="100px"
        android:layout_height="wrap_content"
        android:text="Button"
        />
</LinearLayout>
```

图 3 显示了 View 从左往右摆放。

图 3 LinearLayout 中 View 的摆放

LinearLayout 默认的方向为水平的。如果你想修改它的方向为垂直的，设置 orientation 特性为 vertical，像这样：

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    xmlns:android="http://schemas.android.com/apk/res/android"
    >
```

图 4 显示了修改方向为垂直方向的效果。

图 4 修改方向为垂直方向

在 LinearLayout 中，你可以应用其内 View 的 layout_weight 和 layout_gravity 特性，如下面对 main.xml 的修改所示：

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

```
oid"
    android:orientation="vertical"
    >
    <TextView
        android:layout_width="105px"
        android:layout_height="wrap_content"
        android:text="@string/hello"
        />
    <Button
        android:layout_width="100px"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_gravity="right"
        android:layout_weight="0.2"
        />
    <EditText
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:layout_weight="0.8"
        />
</LinearLayout>
```

图 5 显示了 Button 通过使用 layout_gravity 特性将其摆放在父元素（LinearLayout）的右侧。如此同时，使用 layout_weight 特性，Button 和 EditText 以一定的比率占据屏幕剩余的空间。layout_weight 特性值的总和必须等于 1。

图 5　应用 layout_weight 和 layout_gravity 特性

AbsoluteLayout

AbsoluteLayout 允许你指定孩子的精确位置。假设 main.xml 中定义了下面的 UI：

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    >
    <Button
        android:layout_width="188px"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_x="126px"
        android:layout_y="361px"
        />
    <Button
        android:layout_width="113px"
        android:layout_height="wrap_content"
        android:text="Button"
        android:layout_x="12px"
        android:layout_y="361px"
        />
</AbsoluteLayout>
```

图 6 显示了两个 Button，使用 android:layout_x 和 android:layout_y 特性来定位它们指定的位置。

图 6　在 AbsoluteLayout 中摆放 View

作者的提醒：当屏幕的旋转发生改变，你需要重新摆放你的 View 时，你最好使用 AbsoluteLayout。

TableLayout

TableLayout 以行列的方式来组合 View。你可以使用<TableRow>元素来在表中生成一行。每一行可以包含一个或多个 View。你在行中放置的每个 View 会形成一个单元。每列的宽度由这一列中单元的最大宽度来决定。

以下面的元素来填充 mail.xml，观察 UI 如图 7 所示：

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/and
roid"
    android:layout_height="fill_parent"
    android:layout_width="fill_parent"
    android:background="#000044">
    <TableRow>
        <TextView
            android:text="User Name:"
            android:width ="120px"
            />
```

```
        <EditText
            android:id="@+id/txtUserName"
            android:width="200px" />
    </TableRow>
    <TableRow>
        <TextView
            android:text="Password:"
            />
        <EditText
            android:id="@+id/txtPassword"
            android:password="true"
            />
    </TableRow>
    <TableRow>
        <TextView />
        <CheckBox android:id="@+id/chkRememberPassword"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Remember Password"
            />
    </TableRow>
    <TableRow>
        <Button
            android:id="@+id/buttonSignIn"
            android:text="Log In" />
    </TableRow>
</TableLayout>
```

图 7　使用 TableLayout

注意，在上面的例子中，TableLayout 中有两列四行。Password 正下方的 Text View 填入了一个空的元素。如果你不这么做，记住密码的 Checkbox 将会出现在 Password TextView 的下方，如图 8 所示。

图 8　注意记住密码 Checbox 位置的改变

RelativeLayout

RelativeLayout 允许你指定子 View 的位置相对于其它的 View。假设 main.xml 文件如下面所示：

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/RLayout"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    >
    <TextView
        android:id="@+id/lblComments"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Comments"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true"
        />
    <EditText
        android:id="@+id/txtComments"
        android:layout_width="fill_parent"
        android:layout_height="170px"
        android:textSize="18sp"
        android:layout_alignLeft="@+id/lblComments"
        android:layout_below="@+id/lblComments"
        android:layout_centerHorizontal="true"
        />
    <Button
        android:id="@+id/btnSave"
        android:layout_width="125px"
        android:layout_height="wrap_content"
        android:text="Save"
        android:layout_below="@+id/txtComments"
        android:layout_alignRight="@+id/txtComments"
        />
    <Button
        android:id="@+id/btnCancel"
        android:layout_width="124px"
        android:layout_height="wrap_content"
        android:text="Cancel"
        android:layout_below="@+id/txtComments"
        android:layout_alignLeft="@+id/txtComments"
        />
</RelativeLayout>
```

注意，放入 RelativeLayout 中的 View 都有特性来设置它们与其它 View 对齐。这些特性是：

layout_alignParentTop

layout_alignParentLeft

layout_alignLeft

layout_alignRight

layout_below

layout_centerHorizontal

这些特性的值是你已经为 View 定义的 ID。上面的 XML UI 创建的屏幕如图 9 所示。

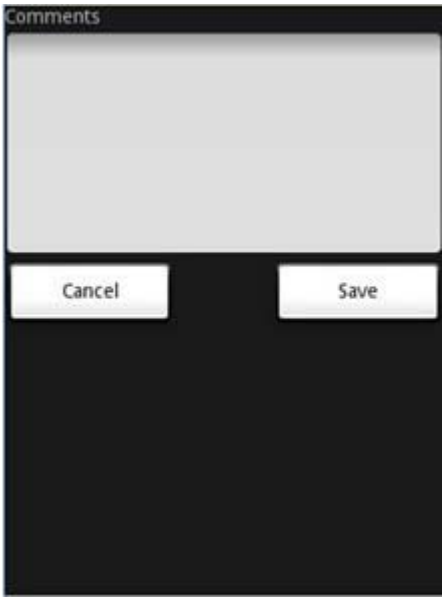

图 9 使用 AbsoluteLayout 来摆放 View

FrameLayout

FrameLayout 是屏幕上的一个占位符，你可以用它来显示单个 View。添加到 FrameLayout 上的 View 总是位于布局的左上角。假设 main.xml 中有以下内容：

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget68"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    >
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="40px"
        android:layout_y="35px"
        >
        <ImageView
            android:src = "@drawable/androidlogo"
```

```
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            />
    </FrameLayout>
</AbsoluteLayout>
</p>

<p>Here, you have a <code>FrameLayout
```

在 AbsoluteLayout 中，放置了一个 FrameLayout，其中嵌入了一个 ImageView。UI 如图 10 所示。

注意：这个例子假设 res/drawable 文件夹下有一个名叫 androidlogo.png 图片。

图 10　使用 FrameLayout

如果你添加另一个 View（例如一个 Button）到 FrameLayout 上，新的 View 将遮盖前一个 View（如图 11 所示）：

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    android:id="@+id/widget68"
```

```
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    >
    <FrameLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="40px"
        android:layout_y="35px"
        >
        <ImageView
            android:src = "@drawable/androidlogo"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            />
        <Button
            android:layout_width="124px"
            android:layout_height="wrap_content"
            android:text="Print Picture"
            />
    </FrameLayout>
</AbsoluteLayout>
```

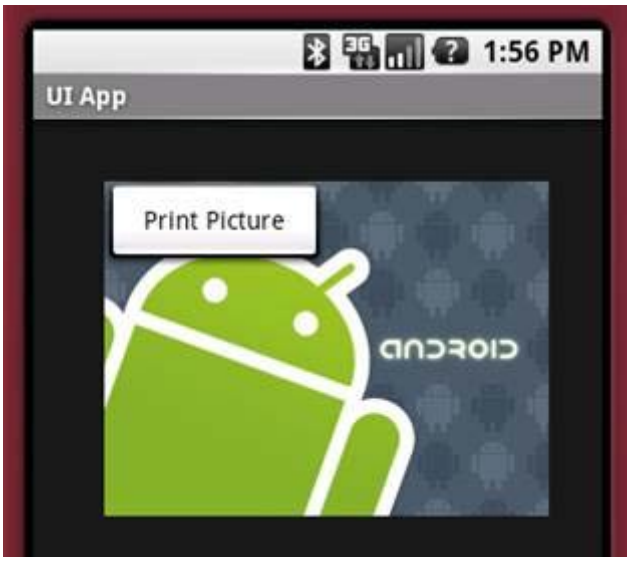

图 11　重叠 View

你可以添加多个 View 到一个 FrameLayout 上，但每一个都将停靠在前一个的上方。

ScrollView

ScrollView 是一种特殊类型的 FrameLayout，它允许用户滚动查看 View 列表，而这些 View 将占据比实际显示更多的空间。ScrollView 仅能容纳一个子 View 或 ViewGroup，一般都是 LinearLayout。

注意：不要将 ListView 和 ScrollView 一起使用。ListView 设计用来显示有联系的信息列表，并且为大列表的处理作了优化。

接下来的 main.xml 的内容包含一个 ScrollView，其中包含一个 LinearLayout，LinearLayout 中依次包含一些 Button 和 EditText：

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    android:id="@+id/widget54"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/andr
oid"
    >
    <LinearLayout
        android:layout_width="310px"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        >
        <Button
            android:id="@+id/button1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 1"
            />
        <Button
            android:id="@+id/button2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 2"
            />
        <Button
            android:id="@+id/button3"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
```

```
            android:text="Button 3"
            />
        <EditText
            android:id="@+id/txt"
            android:layout_width="fill_parent"
            android:layout_height="300px"
            />
        <Button
            android:id="@+id/button4"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 4"
            />
        <Button
            android:id="@+id/button5"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Button 5"
            />
    </LinearLayout>
</ScrollView>
```

图 12 中 ScrollView 在屏幕的右侧显示了一个滚动条。用户可以往上拖拽屏幕来显示位于屏幕下方的 View。

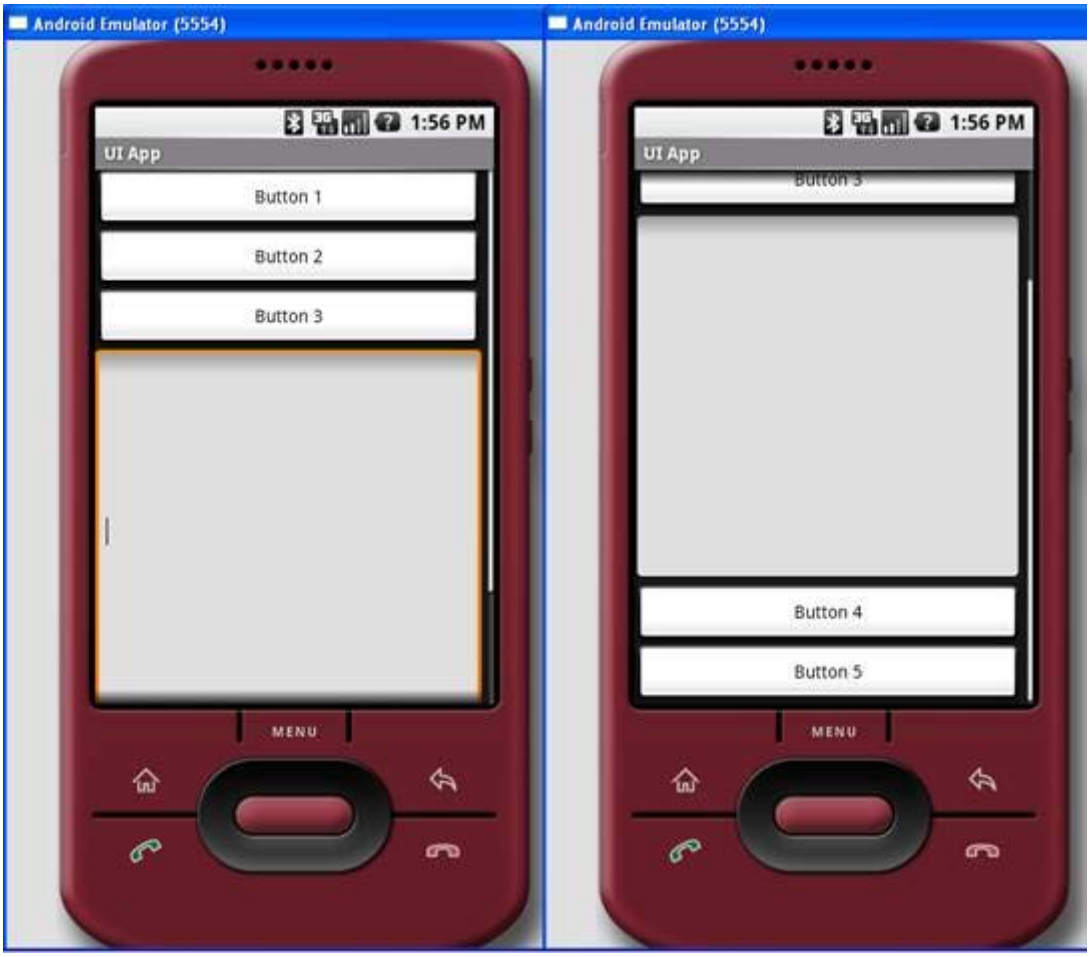

图 12 使用 ScrollView

小结

在这一篇文章里，你已经看到了 Android 中用于创建 UI 的一些布局。下一篇文章中，我将向你展示一些 View（Widget），你可以使用它来创建杀手级的应用程序。至此，祝愉快！

前一篇关于 Android UI 的文章中，你已经看到了组成 Android 应用程序 UI 的组件。Android UI 最基本的单元是 View。一个 View 代表一个构件（Widget），它在屏幕上拥有外观。在这篇文章（以及接下来的两篇）里，你将学习到你在 Android 开发旅程中可能用到的一些常用的 View。特别的，我将 View 进行了分类，如下面的组所示：

- Basic Views – 常用的 View，例如 TextView, EditText, 和 Button
- Picker Views – 允许用户进行选择的 View，例如 TimePicker 和 DatePicker
- List Views – 显示大量项目的 View，例如 ListView 和 Spinner
- Display Views – 显示图片的 View，例如 Gallery 和 ImageSwitcher
- Menus – 显示额外的和上下文菜单项目的 View
- Additional Views – 感兴趣的 View，例如 AnalogClock 和 DigitalClock

在这篇文章里，我将囊括第一组 - Basic Views。接下来的文章将囊括 Picker Views 和 List Views。最后，第三篇文章将囊括 Menus 和 Additional Views。

这篇文章中的所有例子，你需要使用 Eclipse 创建一个新的工程。命名工程如图 1 所示。

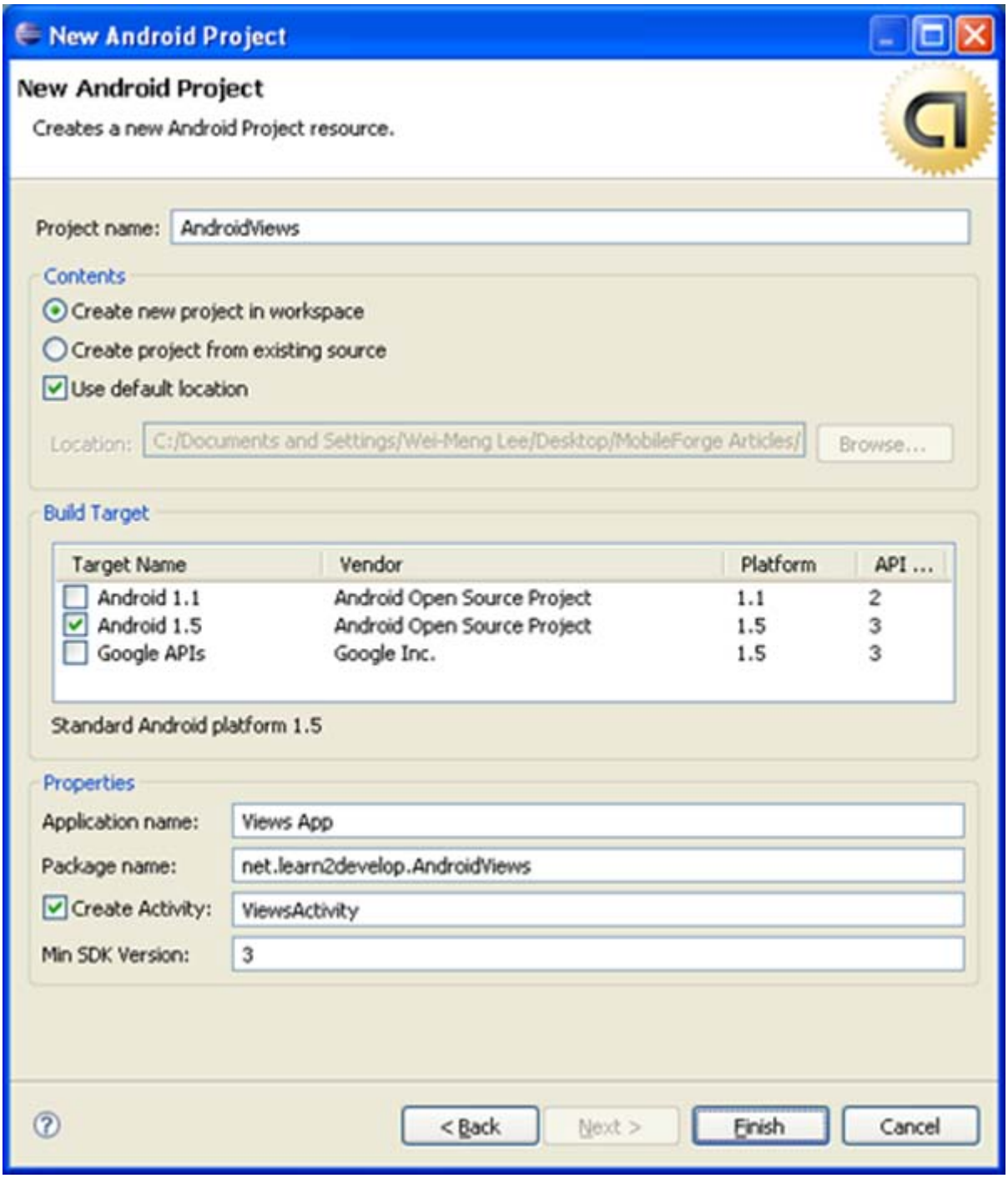

图 1 命名工程

Basic Views

在这一章节，你将学习 Android 中最基本的 View，允许你显示文本信息，也能执行一些基本的选择。特别的，你将学习以下 Views：

- TextView

- EditText
- Button
- ImageButton
- CheckBox
- ToggleButton
- RadioButton
- RadioGroup

TextView View

当你创建一个新的 Android 工程时，Eclipse 总是创建 main.xml 文件（位于 res/layout 文件夹下），其中就包含了一个元素：

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

TextView 用于向用户显示文本。这是最基本的 View，当你开发 Android 应用程序时你一定遇到过。如果你需要允许用户来编辑显示的文本，你需要使用 TextView 的子类- EditText，它将在下一个章节中讨论。

Button，ImageButton，EditText，CheckBox，ToggleButton，RadioButton 和 RadioGroup，包括 TextView，这些都是你经常遇到的。还有一些其它的控件，在以后的使用过程中你会发现的。

一开始，在 res/layout 文件夹下添加一个新的文件，取名 basicviews.xml。将下面的元素填入其中：

```
<?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/btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Save"
        />
    <Button android:id="@+id/btnOpen"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Open"
        />
    <ImageButton android:id="@+id/btnImg1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:src="@drawable/icon"
        />
    <EditText android:id="@+id/txtName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />
    <CheckBox android:id="@+id/chkAutosave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Autosave"
        />
    <CheckBox android:id="@+id/star"
        style="?android:attr/starStyle"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

    <RadioGroup android:id="@+id/rdbGp1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        >
        <RadioButton android:id="@+id/rdb1"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
```

```
            android:text="Option 1"
            />
        <RadioButton android:id="@+id/rdb2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Option 2"
            />
    </RadioGroup>

    <ToggleButton android:id="@+id/toggle1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

注意，使用 id 特性来标识每个 View 元素。每个 View 的 id 必须以“@+id/”标示符开始，后面紧接 View 的名字。

上面的 XML 文件包含以下 View：

- Button – 表示一个可按的按钮构件
- ImageButton – 与 Button 相似，但它可以显示一个图片
- EditText - TextView 的子类，它允许用户编辑文本内容
- CheckBox – 特殊类型的 Button，它拥有两个状态——checked 或 unchecked
- RadioGroup 和 RadioButton – RadioButton 有两个状态——checked 或 unchecked。一旦 RadioButton 是 checked，它就不能是 unchecked。RadioGroup 用于组合一个或多个 RadioButton，因此，允许 RadioGroup 中仅含一个 RadioButton 来选择。
- ToggleButton – 使用亮条来显示 checked/unchecked 状态

图 2 显示了不同状态下的 View。特别的，右侧的图片显示了 CheckBox，RadioButton 和 ToggleButton 处于 checked 状态。

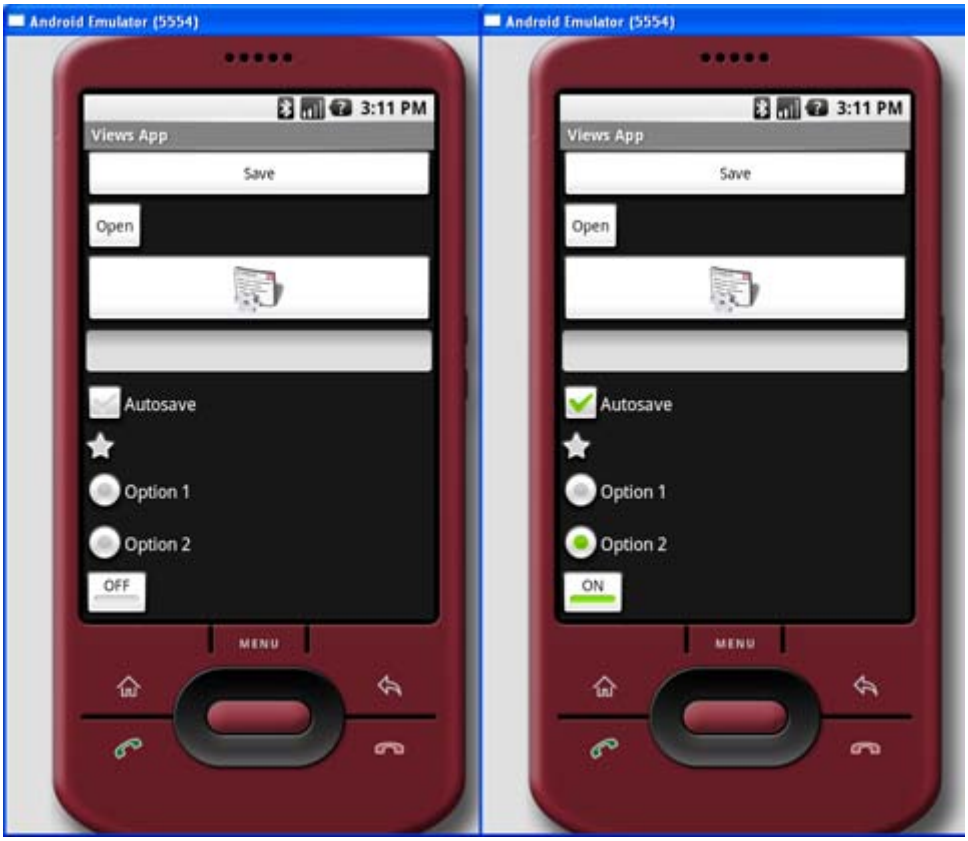

图 2　View 的几种状态

为了处理 View 的通用事件，在 src/net.learn2develop.AndroidViews 文件夹下添加一个新的类，取名 BasicViewsExample.java。在文件中添加以下代码：

```
package net.learn2develop.AndroidViews;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.RadioGroup;
import android.widget.Toast;
import android.widget.ToggleButton;
import android.widget.RadioGroup.OnCheckedChangeListener;

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

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

        //---Button view---
        Button btnOpen = (Button) findViewById(R.id.btnOpe
n);
        btnOpen.setOnClickListener(new View.OnClickListene
r() {
            public void onClick(View v) {
                Toast.makeText(getBaseContext(),
                        "You have clicked the Open button",
                        Toast.LENGTH_SHORT).show();
            }
        });

        Button btnSave = (Button) findViewById(R.id.btnSav
e);
        btnSave.setOnClickListener(new View.OnClickListene
r()
        {
            public void onClick(View v) {
                DisplayToast("You have clicked the Save butt
on");
            }
        });

        //---CheckBox---
        CheckBox checkBox = (CheckBox) findViewById(R.id.ch
kAutosave);
        checkBox.setOnClickListener(new View.OnClickListen
er()
        {
            public void onClick(View v) {
                if (((CheckBox)v).isChecked())
                    DisplayToast("CheckBox is checked");
                else
                    DisplayToast("CheckBox is unchecked");

            }
        });

        //---RadioButton---
        RadioGroup radioGroup = (RadioGroup) findViewById
(R.id.rdbGp1);
        radioGroup.setOnCheckedChangeListener(new OnChecke
```

```
dChangeListener()
        {
            public void onCheckedChanged(RadioGroup group, i
nt checkedId) {
                //---displays the ID of the RadioButton that
is checked---
                DisplayToast(Integer.toString(checkedId));
            }
        });

        //---ToggleButton---
        ToggleButton toggleButton = (ToggleButton) findView
ById(R.id.toggle1);
        toggleButton.setOnClickListener(new View.OnClickLi
stener()
        {
            public void onClick(View v) {
              if (((ToggleButton)v).isChecked())
                  DisplayToast("Toggle button is On");
              else
                 DisplayToast("Toggle button is Off");
            }
        });
    }

    private void DisplayToast(String msg)
    {
         Toast.makeText(getBaseContext(), msg,
                Toast.LENGTH_SHORT).show();
    }
}
```

上面的代码在控件点击的时候作了特殊的处理，显示了一个消息（使用 Toast 类）。

在 AndroidManifest.xml 文件中添加下面粗体的几行来注册新的 BasicViewsExample Activity：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/r
es/android"
      package="net.learn2develop.AndroidViews"
      android:versionCode="1"
      android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:lab
```

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

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

    </application>
</manifest>
```

为了显示 BasicViewsExample Activity，在 ViewsActivity.java 文件中添加以下粗体的语句：

```
package net.learn2develop.AndroidViews;

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

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

        //---load the BasicViewsExample activity---
        startActivity(new Intent(this, BasicViewsExample.cl
ass));

    }
}
```

为了测试应用程序，按下 F11，在 Android 模拟器中调试应用程序。图 3 显示了当 ToggleButton 点击时显示的消息。

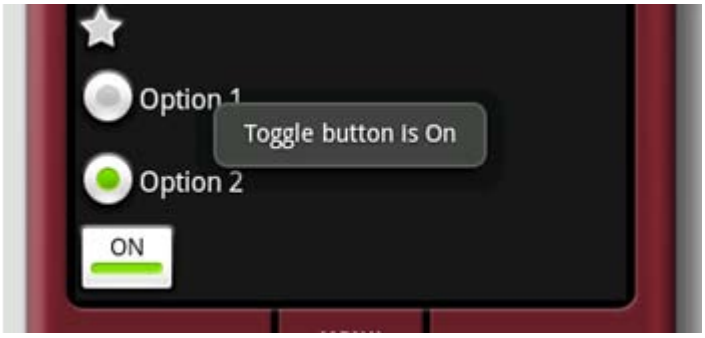

图 3　当 ToggleButton 点击时显示消息

对于 EditText View，你还可以设置它来接受密码，以“.”来替代每个字符。这是通过 password 特性来完成的：

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

图 4 显示了当你在 View 中输入字符串时，你输入的字符一开始总是显示，当你输入下一个字符（或者你什么也不做，一秒后）时前一个才会变成“.”。

图 4　将 password 特性设置为 true

ProgressBar View

ProgressBar 提供了后台工作的视觉反馈。举个例子，你可能从网页上下载一些数据，需要向用户更新下载的状态。在这种情况下，ProgressBar 是这项工作最好的选择。

使用上一章节中创建的 Activity，在 basicviews.xml 文件中插入下面的元素：

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

    <Button android:id="@+id/btnSave"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Save"
        />
    <!-
        Other views omitted
    -->
</LinearLayout>
```

ProgressBar 的默认模式是不确定的——也就是说，它显示成一个循环的动画。这个模式对于那些不清楚何时完成的任务来说非常有用。举个例子，你正在向一个网页服务器发送一些数据，等待服务器的响应。

下面的代码演示了如何用一个线程来模拟一个耗时的任务。当任务完成时，通过设定 ProgressBar 的 Visibility 属性为 GONE（值为 8）来隐藏它。

```
import android.widget.ProgressBar;
import android.os.Handler;

public class BasicViewsExample extends Activity
{
    private static int progress = 0;
    private ProgressBar progressBar;
    private int progressStatus = 0;
    private Handler handler = new Handler();

    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
```

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

        progressBar = (ProgressBar) findViewById(R.id.progr
essbar);

        //---do some work in background thread---
        new Thread(new Runnable()
        {
            public void run()
            {
                //---do some work here---
                while (progressStatus < 10)
                {
                    progressStatus = doSomeWork();
                }

                //---hides the progress bar---
                handler.post(new Runnable()
                {
                    public void run()
                    {
                        //---0 - VISIBLE; 4 - INVISIBLE; 8 - G
ONE---
                        progressBar.setVisibility(8);
                    }
                });
            }

            //---do some long lasting work here---
            private int doSomeWork()
            {
                try {
                    //---simulate doing some work---
                    Thread.sleep(500);
                } catch (InterruptedException e)
                {
                    e.printStackTrace();
                }
                return ++progress;
            }

        }).start();

        //...
```

```
        //...
    }

    //...
}
```

INVISIBLE 和 GONE 常量的区别是，INVISIBLE 常量只是简单的隐藏 ProgressBar。而 GONE 常量表示从 Activity 中移除 ProgressBar，并且不再占据 Activity 的任何空间。

图 5 的左边显示了动作中的 ProgressBar。后台任务完成后，当你设定 Visibility 属性为 GONE 时，ProgressBar 会放弃它占据的空间。

图 5 动作中的 ProgressBar（左），当后台任务完成时消失（右）

如果你不想 ProgressBar 以不确定的模式显示，那么，你可以修改它显示成一个水平条。下面的 style 特性指定了 ProgressBar 显示成一个水平条：

```
<ProgressBar android:id="@+id/progressbar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        style="?android:attr/progressBarStyleHorizontal"
        />
```

下面的代码使得 ProgressBar 从 1 增加到 100，然后从屏幕消失（见图 6）：

```
public class BasicViewsExample extends Activity
{
    private static int progress = 0;
    private ProgressBar progressBar;
    private int progressStatus = 0;
    private Handler handler = new Handler();
```

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

        progressBar = (ProgressBar) findViewById(R.id.progr
essbar);

        //---do some work in background thread---
        new Thread(new Runnable()
        {
            public void run()
            {
                while (progressStatus < 100)
                {
                    progressStatus = doSomeWork();

                    //---Update the progress bar---
                    handler.post(new Runnable()
                    {
                        public void run() {
                            progressBar.setProgress(progressSt
atus);
                        }
                    });

                }
                //---hides the progress bar---
                handler.post(new Runnable()
                {
                    public void run() {
                        //---0 - VISIBLE; 4 - INVISIBLE; 8 - G
ONE---
                        progressBar.setVisibility(8);
                    }
                });
            }

            private int doSomeWork()
            {
                try
                {
                    //---simulate doing some work---
```

```
                Thread.sleep(500);
            } catch (InterruptedException e)
            {
                e.printStackTrace();
            }
            return ++progress;
        }

    }).start();

    //...
    //...
  }
}
```

图 6 动作中的水平 ProgressBar（左），后台任务完成时消失（右）

AutoCompleteTextView View

AutoCompleteTextView 是一个类似于 EditText 的 View（实际上它是 EditText 的子类），不同的是当用户输入字符时，会自动显示出字符补全建议的列表。

在 res/layout 文件夹下添加一个新的文件，取名 autocomplete.xml 并且填入以下元素：

```
<?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 android:id="@+id/txtCountries"
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

在 src/net.learn2develop.AndroidViews 文件夹下添加新的文件，取名 AutoCompleteExample.java。填入以下内容：

```
package net.learn2develop.AndroidViews;

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

public class AutoCompleteExample extends Activity
{
    String[] presidents =
    {
            "Dwight D. Eisenhower",
            "John F. Kennedy",
            "Lyndon B. Johnson",
            "Richard Nixon",
            "Gerald Ford",
            "Jimmy Carter",
            "Ronald Reagan",
            "George H. W. Bush",
            "Bill Clinton",
            "George W. Bush",
            "Barack Obama"
    };

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

        ArrayAdapter<String> adapter = new ArrayAdapter<Str
ing>(this,
            android.R.layout.simple_dropdown_item_1line, pr
esidents);
```

```
        AutoCompleteTextView textView = (AutoCompleteTextVi
ew)
                findViewById(R.id.txtCountries);
        textView.setThreshold(3);
        textView.setAdapter(adapter);
    }
}
```

注意，建议的列表从 ArrayAdapter 对象中获取。setThreshold 方法设定了在建议列表显示成下拉菜单之前用户必须输入的最少字符数。

在 AndroidManifest.xml 文件中添加以下粗体行来注册新的 AutoCompleteExample Activity：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/r
es/android"
     package="net.learn2develop.AndroidViews"
     android:versionCode="1"
     android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:lab
el="@string/app_name">
        <activity android:name=".ViewsActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.
MAIN" />
                <category android:name="android.intent.categ
ory.LAUNCHER" />
            </intent-filter>
        </activity>

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

    </application>
</manifest>
```

修改 ViewsActivity.java 文件来启动 AutoCompleteExample Activity：

```
package net.learn2develop.AndroidViews;

import android.app.Activity;
```

```
import android.content.Intent;
import android.os.Bundle;

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

        startActivity(new Intent(this, AutoCompleteExampl
e.class));

    }
}
```

图 7 显示了输入一些文本时动作中的 AutoCompleteTextView。

图 7 动作中的 AutoCompleteTextView

小结

在这篇文章里，你已经看到一些在 Android 中很常用的 View。在接下来的两篇文章里，我将深入讲解更多的 View 来帮助你构建杀手级的 Android 应用程序。请继续关注下一篇文章，在那里，我将讨论你可能使用到的 Picker 和 List Views。

原文链接：http://www.cnblogs.com/xirihanlin/archive/2009/10/09/1579924.html

即使你的应用程序是快速且响应灵敏的，但一些设计仍然会给用户造成问题——与其它应用程序或对话框未事先计划的交互，意外的数据丢失，意料之外的阻塞等等。避免这些问题，有助于理解应用程序运行的上下文和系统的交互过程，而这些又正影响着你的应用程序。简而言之，你应该竭尽全力去开发一个与系统和其它应用程序流畅交互的应用程序。

一个常见的流畅问题是，一个应用程序的后台处理——例如，一个 Service 或者 BroadcastReceiver——弹出一个对话框来响应一些事件。这可能看起来没啥大碍，尤其是你在模拟器上单独地构建和测试你的应用程序的时候。然而，当你的应用程序运行在真机上时，有可能你的应用程序在没有获得用户焦点时后台处理显示了一个对话框。因此，可能会出现在活跃的应用程序后方显示了你的应用程序的对话框，或者从当前应用程序夺取焦点显示了一个对话框，而不管当前用户正在做什么（例如，正在打电话）。那种行为，对应用程序或用户来说，就不应该出现。

为了避免这些问题，你的应用程序应该使用合适的系统资源来通知用户——Notification 类。使用 Notification，你的应用程序可以在状态栏显示一个 icon 来通知用户已经发生的事情，而不是夺取焦点和打断用户。

另一个流畅问题的例子是未能正确实现 Activity 的 onPause()和其它生命周期方法而造成意外丢失了状态或用户数据。又或者，如果你的应用程序想暴露数据给其它应用程序使用，你应该通过 ContentProvider 来暴露，而不是（举例）通过一个可读的原始文件或数据库来实现。

这些例子的共同点是它们都应该与系统和其它应用程序协作好。Android 系统设计时，就把应用程序看作是一堆松散耦合的组件，而不是一堆黑盒代码。作为开发者来说，允许我们把整个系统看作是更大的组件集合。这有益于我们可以与其它应用程序进行清晰无缝的集成，因此，作为回报，我们应该更好的设计我们的代码。

这篇文章将讨论常见的流畅问题以及如何避免它们。它将囊括这些主题：

1) 别丢弃数据

2) 不要暴露原始数据

3) 不要打断用户

4) 有太多事情要做？在线程里做

5) 不要让一个 Activity 超负荷

6) 扩展系统主题

7) 设计你的 UI 可以应付多屏幕分辨率

8) 假设网络很慢

9) 不要假定触摸屏或键盘

10) 节省设备电池

1）别丢弃数据

一定要记住 Android 是一个移动平台。可以显而易见地说，其它 Activity（例如，“Incoming Phone Call”应用程序）可能会在任何时候弹出来遮盖你的 Activity，记住这个事实很重要。因为这个过程将触发 onSaveInstanceState()和 onPause()方法，并可能导致你的应用程序被杀死。

如果用户在你的应用程序中正在编辑数据时，其它 Activity 出现了，这时，你的应用程序被杀死时可能丢失那些数据。当然了，除非你事先保存了正在进行的工作。“Android 方式”是这样做的：能接收和编辑用户输入的 Android 应用程序应该重写 onSaveInstanceState()方法，并以恰当的方式保存它们的状态。当用户重新访问应用程序时，她能得到她的数据。

进行这种处理方式最经典的例子是 mail 应用程序。如果用户正在输入 email，这时其它 Activity 启动了，mail 应用程序应该把正在编辑的 email 以草稿的方式保存起来。

2）不要暴露原始数据

如果你不想穿着内衣在大街上溜达的话，你的数据也不应该这样。尽管可能存在暴露应用程序的某种形式给其它应用程序，但这通常不是最好的主意。暴露原始数据，要求其它应用程序能够理解你的数据的格式；如果你变更了格式，那么，你将破坏那些没有进行同步更新的应用程序。

“Android 方式”是创建一个 ContentProvider，以一种清晰的、深思熟虑的和可维护的 API 方式暴露你的数据给其它应用程序。使用 ContentProvider，就好像是插入 Java 接口来分离和组装两片高耦合的代码。这意味着你可以修改数据的内部格式，而不用修改由 ContentProvider 暴露的接口，这样，也不会影响其它应用程序。

3）不要打断用户

如果用户正在运行一个应用程序（例如，Phone 程序），断定对用户操作的目的才是安全的。这也就是为什么必须避免创建 Activity，而是直接在当前的 Activity 中响应用户的输入。

那就是说，不要在 BroadcastReceiver 或在后台运行的 Service 中调用 callActivity()。这么做会中断当前运行的应用程序，并导致用户恼怒。也许更糟糕的是，你的 Activity 可能成为“按键强盗”，窃取了用户要提供给前一个 Activity 的输入。视乎你的应用程序所做的事情，这可能是个坏消息。

不选择在后台直接创建 Activity UI，取而代之的是，应该使用 NotificationManager 来设置 Notification。它们会出现在状态栏，并且用户可以在他空闲的时候点击它们，来查看你的应用程序向他显示了什么。

（注意，如果你的 Activity 已经在前台了，以上将不适用：这时，对于用户的输入，用户期望的是看到下一个 Activity 来响应。）

4）有太多事情要做？在线程里做

如果你的应用程序需要执行一些昂贵或耗时的计算的话，你应该尽可能地将它挪到线程里。这将阻止向用户显示可怕的“Application Not Responding”对话框，如果不这样做，最终的结果会导致你的应用程序完全终止。

一般情况下，Activity 中的所有代码，包括它的 View，都运行在相同的线程里。在这个线程里，还需要处理 UI 事件。例如，当用户按下一个按键，一个 key-down 事件就会添加到 Activity 的主线程队列里。事件处理系统需要很快让这个事件出列并得到处理；如果没有，系统数秒后会认为应用程序已经挂起并为用户提供杀死应用程序的机会。

如果有耗时的代码，内联在 Activity 上运行也就是运行在事件处理线程里，这在很大程度上阻塞了事件处理。这会延迟输入处理，并导致 ANR 对话框。为了避免这个，把你的计算移到线程里。在响应灵敏性设计的文章里已经讨论了如何做。

5）不要让一个Activity超负荷

任何值得使用的应用程序都可能有几个不同的屏幕。当设计 UI 屏幕时，请一定要使用多个 Activity 对象实例。

依赖于你的开发背景，你可能理解 Activity 类似于 Java Applet，它是你应用程序的入口点。然而，那并不精确：Applet 子类是一个 Java Applet 的单一入口点，而一个 Activity 应该看作是你的应用程序多个潜在入口点之一。你的“main”Activity 和其它之间的唯一不同点是“main”Activity 正巧是在 AndroidManifest.xml 文件中唯一对“android.intent.action.MAIN”动作感兴趣的 Activity。

因此，当设计你的应用程序的时候，把你的应用程序看作是 Activity 对象的集合。从长远来看，这会使得你的代码更加方便维护。

6）扩展系统主题

当谈到 UI 观感时，巧妙地交融非常重要。用户在使用与自己期望相反的 UI 的应用程序时，会产生不愉快的感觉。当设计你的 UI 时，你应该尽量避免太多自己的主题。相反的，使用同一个主题。你可以重写或扩展你需要的主题部分，但至少在与其它应用程序相同的 UI 基础上开始。详细请参照“应用风格和主题”部分。

7）设计你的UI可以应对多屏幕分辨率

不同的 Android 设备可能支持不同的屏幕分辨率。甚至一些可以自己变更分辨率，例如，切换到风景模式。确保你的布局和图片能足够灵活地在不同的设备屏幕上正常显示。

幸运的是，这很容易做到。简而言之，你需要做的是为主要分辨率提供不同版本的作品，然后为不同的尺寸设计你的布局。（例如，避免使用硬编码位置而使用相对布局。）如果那样做的话，系统会处理剩下的部分，而且你的应用程序在任何设备上都看起来很棒。

8）假设网络很慢

Android 设备会有多种网络连接选项。所有的都提供数据访问，但之间肯定有更快的。其中，速度最慢的是 GPRS，GSM 网络的非 3G 数据服务。即使具备 3G 能力的设备在非 3G 的网络上也会花费很多的时间，所以，网络很慢仍然是一个长期存在的事实。

这就是为什么你应该按照最小化的网络访问和带宽来编写你的代码。你不能假设网络是快速的，所以，你应该总是计划它是慢的。如果你的用户碰巧在一个快速的网络上，那很好——他们的用户体验会提升。你要避免相反的情形：在不同的地点和不同时间，应用程序有时可用，有时慢得令人抓狂，这样的程序可能不会受欢迎。

还有一个潜在的地方是，如果你正在使用模拟器，那么你很容易受它迷糊，因为模拟器使用电脑的网络连接。这比手机网络快很多，所以，你需要修改模拟器设定来模拟较低的网络速度。你可以在 Eclipse 中做到这点，在启动选项的模拟器设置页里设置或者在启动模拟器时通过命令行选项设置。

9）不要假定触摸屏或键盘

Android 可以支持多种外观形状。也就是说，一些 Android 设备拥有全“QWERTY”键盘，而其它可能会有 40 键、12 键或其它键盘设置。同样的，一些设备可能有触摸屏，但一些也会没有。

当创建你的应用程序的时候，记住这一点。不要假定特定的键盘布局——除非你真的想限定你的应用程序只运行在某些设备上。

10）节省设备电池

如果移动设备经常插在墙上，那么，它也就不是很“移动”。移动设备是电池供电的，如果我们能让每次充电的电池使用得更持久一些，那么每个人都会更加开心——尤其是用户。其中两大耗电硬件是处理器和无线；这也就是我们为什么要写尽可能少做工作、尽可能少去使用网络的应用程序的重要原因。

如何让你的应用程序最小化的占用处理器，归根结底还是要写高效代码。为了减少无线的电量消耗，确保对错误条件进行正确的处理，并只获取你要的东西。例如，如果某一个网络操作失败了，不要不断地进行重试。如果失败了一次，有可能是用户不受欢迎，因此，如果你再以正确的方式操作，有可能还会失败；所有你做的都是在浪费电池。

用户是相当聪明的：如果你的程序高耗电，他们是一定会发现的。到那个时点，你唯一可以确定的是，你的程序将很快被卸载掉。

第四部分

在 Android Views 系列文章的最后一篇里，我们将继续介绍其他种类的 View——Menu 和一些额外 cool 的 View。讨论的 View 包含：

- Context Menu
- Options Menu
- AnalogClock
- DigitalClock
- WebView

注意：这篇文章中的所有例子，将延续前篇文章中创建的工程。

Menus

Menus 对于不能在应用程序主 UI 上直接显示额外选项来说非常有用。在 Android 中，主要有两种 Menu：

- Context Menu – 显示一个 Activity 中特定 View 的信息。在 Android 中，通过按下并 Hold 一段时间来激活上下文菜单。
- Options Menu – 显示当前 Activity 的信息。在 Android 中，通过按下 MENU 键来激活选项菜单。

图 1 显示了在浏览器应用程序中的选项菜单。无论用户何时按下 MENU 按钮，选项菜单都会显示。显示的菜单项取决于当前运行的 Activity。

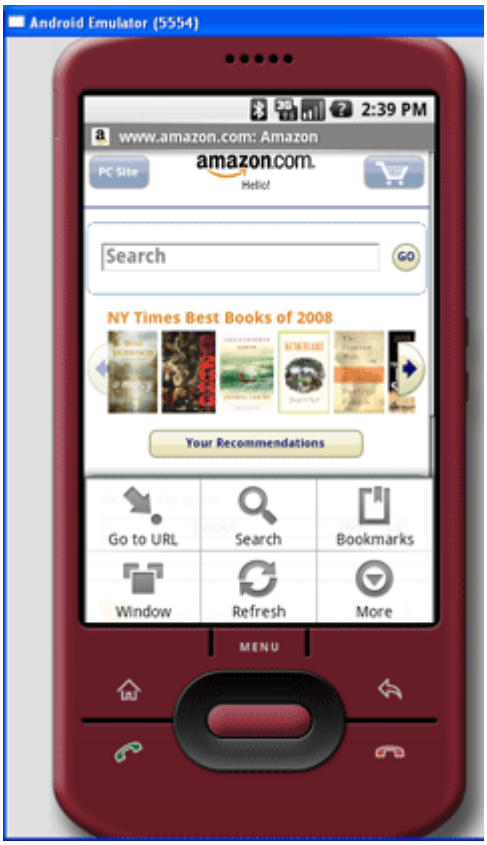

图 1 浏览器程序中的选项菜单

图 2 展示了当用户按下并 Hold 当前页中显示的超链接时，会显示出上下文菜单。显示的菜单项取决于组件或当前选择的 View。为了激活上下文菜单，用户可以通过选择屏幕上的项目、按下并 Hold 一段时间，或者通过简单的按下方向键盘中的中键方式。

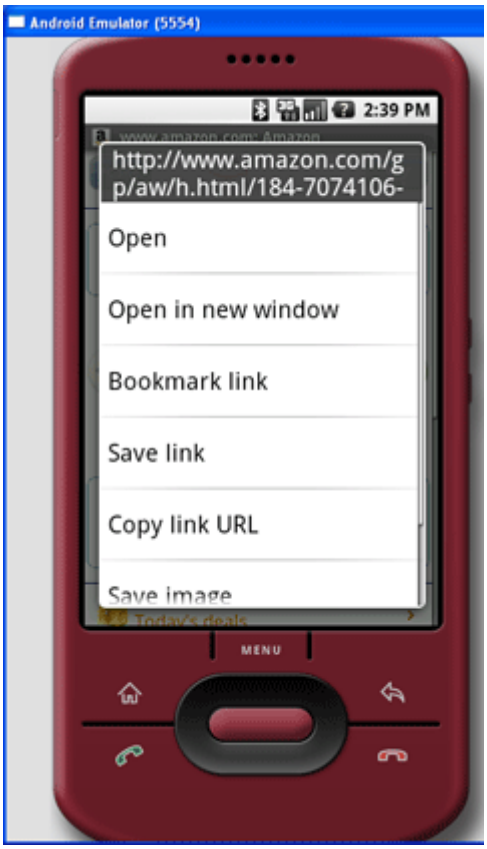

图 2 浏览器程序中的上下文菜单

为了在 Android 中实现 Menu，通过在 res/layout 文件夹下添加一个新的文件并取名 menu.xml。用以下元素进行填充：

```
<?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="fill_parent"
        android:layout_height="wrap_content"
        android:text = "Hello, world!"
        />
 </LinearLayout>
```

然后，添加一个新的类到 src/net.learn2develop.AndroidViews 文件夹中，并取名 MenuExample.java。现在先不管这个文件。修改 AndroidManifest.xml 文件，注册新的 Activity：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
     package="net.learn2develop.AndroidViews"
     android:versionCode="1"
     android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".ViewsActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

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

    </application>
</manifest>
```

创建辅助方法

对于本节中的例子来说，你将在 MenuExample.java 文件中创建两个辅助方法，它们是 CreateMenu() 和 MenuChoice()。定义这两个辅助方法和 onCreate()方法，如下所示：

```
package net.learn2develop.AndroidViews;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.widget.Button;
import android.widget.Toast;

public class MenuExample extends Activity
```

```
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.menu);
        Button btn = (Button) findViewById(R.id.btn1);

        btn.setOnCreateContextMenuListener(this);
    }

    private void CreateMenu(Menu menu)
    {
        menu.setQwertyMode(true);
        MenuItem mnu1 = menu.add(0, 0, 0, "Item 1");
        {
            mnu1.setAlphabeticShortcut('a');
            mnu1.setIcon(R.drawable.alert_dialog_icon);

        }
        MenuItem mnu2 = menu.add(0, 1, 1, "Item 2");
        {
            mnu2.setAlphabeticShortcut('b');
            mnu2.setIcon(R.drawable.ic_popup_reminder);

        }
        MenuItem mnu3 = menu.add(0, 2, 2, "Item 3");
        {
            mnu3.setAlphabeticShortcut('c');
            mnu3.setIcon(R.drawable.icon);
        }
        MenuItem mnu4 = menu.add(0, 3, 3, "Item 4");
        {
            mnu4.setAlphabeticShortcut('d');

        }
        menu.add(0, 3, 3, "Item 5");
        menu.add(0, 3, 3, "Item 6");
        menu.add(0, 3, 3, "Item 7");
    }

    private boolean MenuChoice(MenuItem item)
    {
```

```
        switch (item.getItemId()) {
        case 0:
            Toast.makeText(this, "You clicked on Item 1",
                Toast.LENGTH_LONG).show();
            return true;
        case 1:
            Toast.makeText(this, "You clicked on Item 2",
                Toast.LENGTH_LONG).show();
            return true;
        case 2:
            Toast.makeText(this, "You clicked on Item 3",
                Toast.LENGTH_LONG).show();
            return true;
        case 3:
            Toast.makeText(this, "You clicked on Item 4",
                Toast.LENGTH_LONG).show();
            return true;
        case 4:
            Toast.makeText(this, "You clicked on Item 5",
                Toast.LENGTH_LONG).show();
            return true;
        case 5:
            Toast.makeText(this, "You clicked on Item 6",
                Toast.LENGTH_LONG).show();
            return true;
        case 6:
            Toast.makeText(this, "You clicked on Item 7",
                Toast.LENGTH_LONG).show();
            return true;
        }
        return false;
    }
}
```

CreateMenu()方法创建一堆菜单项，并修改了每个菜单项的外观。add 方法中的参数包括：groupid，itemid，order 和 title。setAlphabeticShortcut()方法为菜单项设置了快捷键，这样，当用户按下特定的键时，菜单项就被选择。setIcon()方法为菜单项设置了图标。

MenuChoice()方法使用 Toast 类来显示选择的菜单项。接下来，拷贝两个图片（如图 3 所示）到 res/drawable 文件夹中。

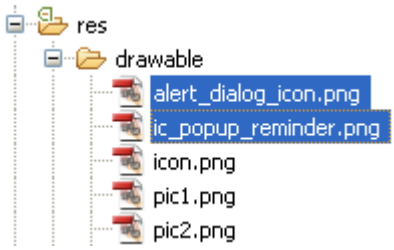

图 3 添加文件到 res/drawable

Options Menu

为了在 Activity 中显示选项菜单，你需要重写两个方法——onCreateOptionsMenu()和 onOptionsItemSelected()。onCreateOptionsMenu()方法在 MENU 按钮被按下时调用。在这个事件中，你需要调用 CreateMenu（）辅助方法来显示选项菜单。当一个菜单项被选中时，onOptionsItemSelected()方法会被调用。在这个情况下，调用 MenuChoice()方法来显示选中的菜单项（和你想要做的任何事情）：

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

        setContentView(R.layout.menu);
    }

    private void CreateMenu(Menu menu)
    {
        //...
    }

    private boolean MenuChoice(MenuItem item)
    {
        //...
    }

    //---only created once---
    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        super.onCreateOptionsMenu(menu);
        CreateMenu(menu);
        return true;
    }
```

```
    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
        return MenuChoice(item);
    }

}
```

修改 ViewsActivity.java 文件来启动 MenuExample Activity：

```
package net.learn2develop.AndroidViews;

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

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

        startActivity(new Intent(this, MenuExample.class));
    }
}
```

按下 F11 来调试应用程序。当你按下 MENU 键时，你将看到如图 4 的选项菜单。

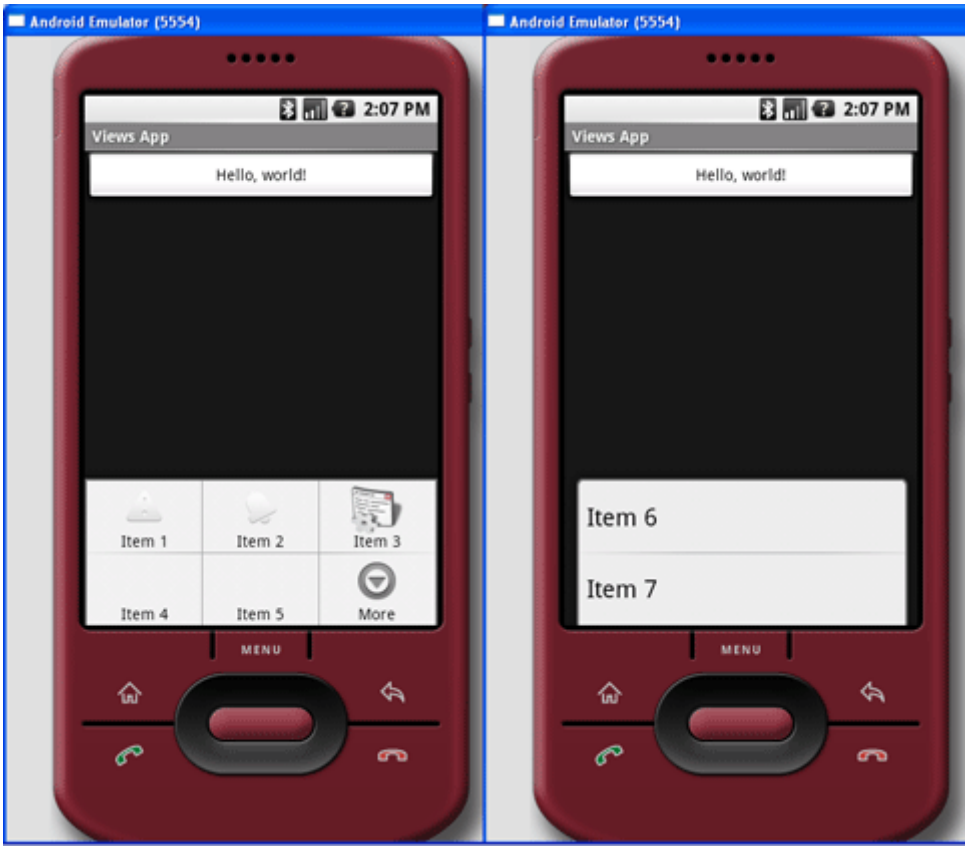

图 4 显示选项菜单

注意菜单项 1、2、3 上显示的图标。此外，如果选项菜单包含 6 个以上菜单项时，这里将出现一个 More 菜单项来表示其它的菜单项。图 4 的右图显示了当按下 More 菜单项时以列表的形式显示的其它菜单项。

Context Menu

如果你想给 Activity 中的 View 关联一个上下文菜单的话，你需要调用特定 View 的 setOnCreateContextMenuListener() 方法。举个例子，接下来的代码将展示如何给 Button 关联一个上下文菜单：

```
package net.learn2develop.AndroidViews;

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

```
import android.widget.Toast;

public class MenuExample extends Activity
{
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.menu);

        Button btn = (Button) findViewById(R.id.btn1);

        btn.setOnCreateContextMenuListener(this);
    }

    private void CreateMenu(Menu menu)
    {
        //...
    }

    private boolean MenuChoice(MenuItem item)
    {
        //...
    }

    //---only created once---
    @Override
    public boolean onCreateOptionsMenu(Menu menu)
    {
        super.onCreateOptionsMenu(menu);
        CreateMenu(menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item)
    {
         return MenuChoice(item);
    }

    @Override
    public void onCreateContextMenu(ContextMenu menu, View
 view,ContextMenuInfo menuInfo)
    {
```

```
        super.onCreateContextMenu(menu, view, menuInfo);
        CreateMenu(menu);
    }

    @Override
    public boolean onContextItemSelected(MenuItem item)
    {
        return MenuChoice(item);
    }

}
```

和选项菜单一样，你需要重写 onCreateContextMenu()和 onContextItemSelected()方法。按下 F11 来调试应用程序。按下 Button 并 Hold 几秒钟，你将看到上下文菜单（如图 5 所示）。注意菜单项 1-4 的菜单名下方的快捷键。

图 5 显示上下文菜单

其它 Views

除了你已经看到的标准 Views，Android SDK 还提供了一些有意思的 View，使得你的应用程序更加有趣味。在接下来的章节，你将了解以下 Views：

- AnalogClock
- DigitalClock
- WebView

AnalogClock 和 DigitalClock

AnalogClock 显示一个模拟的时钟。在 main.xml 文件中填入以下元素：

```
<?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 android:id="@+id/clock1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        />

</LinearLayout>
```

在 ViewsActivity.java 文件中，确保 main.xml 文件加载为 Activity 的 UI：

```
package net.learn2develop.AndroidViews;

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

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

```
    }
}
```

按下 F11，你将看到如图 6 所示的模拟时钟。

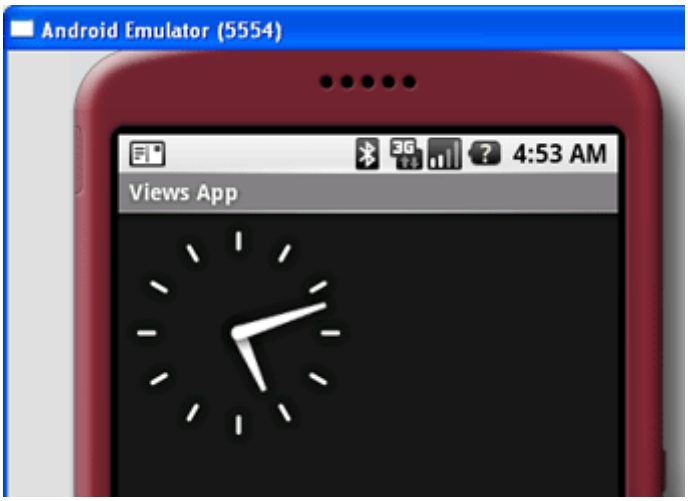

图 6 动作中的模拟时钟

使用 DigitalClock 来取代模拟时钟，这样能看到一个数字时钟。

在 main.xml 文件中添加 DigitalClock 元素，如下所示：

```
<?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 android:id="@+id/clock1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

    <DigitalClock android:id="@+id/clock2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
 </LinearLayout>
```

按下 F11，图 7 同时显示了模拟时钟和数字时钟。

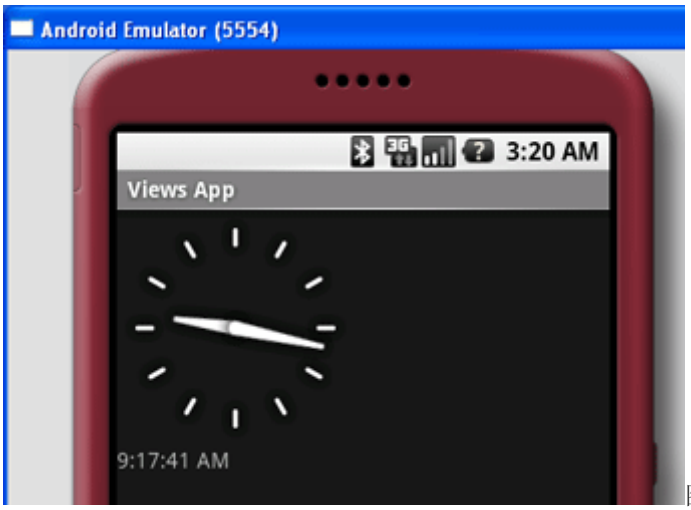

图 7 动作中的数字时钟

WebView

WebView 允许你在 Android 应用程序中嵌入一个网页浏览器。添加一个新的文件到 res/layout 文件夹中，取名 webview.xml。并用以下元素进行填充：

```
<?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 android:id="@+id/webview1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text = "Hello, world!"
        />

</LinearLayout>
```

在 src/net.learn2develop.AndroidViews 文件夹下添加一个新的类，取名 WebExample.java。并用以下内容填充：

```
package net.learn2develop.AndroidViews;

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

```
import android.webkit.WebView;

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

        WebView wv = (WebView) findViewById(R.id.webview1);

        wv.loadUrl("http://www.mobiforge.com");
    }
}
```

WebView 的 loadUrl()方法用于加载一个给定 URL 的页面。修改 AndroidManifest.xml 文件来注册新的 Activity，并添加 INTERNET 权限：

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="net.learn2develop.AndroidViews"
      android:versionCode="1"
      android:versionName="1.0.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".ViewsActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

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

    </application>
```

```
    <uses-permission android:name="android.permission.INTE
RNET">
    </uses-permission>

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

</manifest>
```

为了让 WebView 工作，你需要在 AndroidManifest.xml 文件中添加 INTERNET 权限。修改 ViewsAct ivity.java 文件，来启动 WebViewExample Activity：

```
package net.learn2develop.AndroidViews;

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

public class ViewsActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        startActivity(new Intent(this, WebViewExample.clas
s));
    }
}
```

按下 F11 来启动 Android 模拟器。图 8 显示了动作中的 WebView。

图 8 WebView 正在显示一个网页

你还可以动态创建一个 HTML 字符串并在 WebView 中加载，使用 loadDataWithBaseURL()方法：

```
WebView wv = (WebView) findViewById(R.id.webview1);

        final String mimeType = "text/html";
        final String encoding = "UTF-8";
        String html = "<H1>A simple HTML page</H1><body>" +
            "<p>The quick brown fox jumps over the lazy dog<
/p>";

        wv.loadDataWithBaseURL("", html, mimeType, encodin
g, "");
```

图 9 显示了加载 HTML 字符串的 WebView。

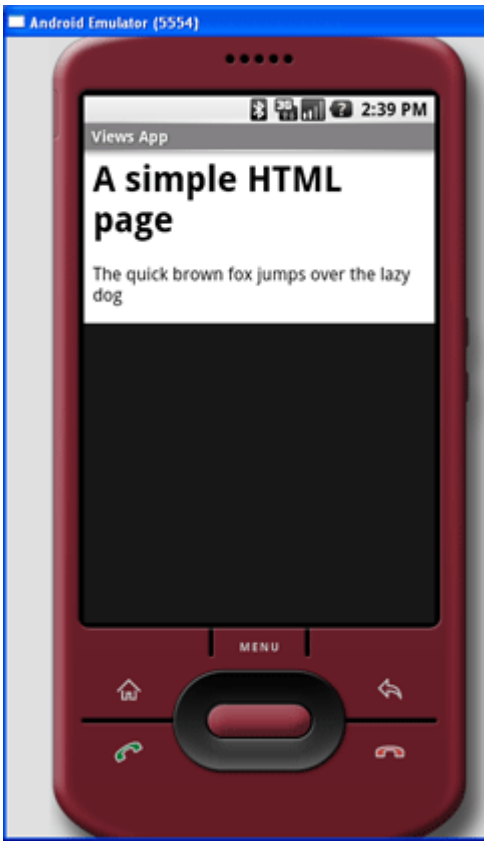

图 9 显示 HTML 字符串的内容

如果你有静态页面，你想通过工程直接加载，你可以把 HTML 页面放在 package 的 assets 文件夹中。图 10 显示了我在 assets 文件夹中添加了一个名为 Index.html 文件。

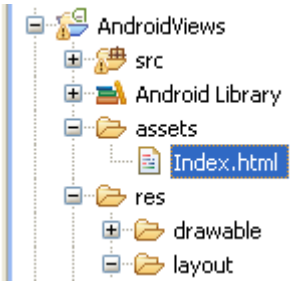

图 10 在 assets 文件夹下添加 HTML 文件

Index.html 的内容如下:

```
<h1>A simple HTML page</h1>
<body>
   <p>The quick brown fox jumps over the lazy dog</p>
```

```
   <img src='http://www.google.com/logos/Logo_60wht.gif'/>
</body>
```

为了将 Index.html 文件中储存的内容加载到 WebView 中，使用 loadUrl()方法如下：

```
WebView wv = (WebView) findViewById(R.id.webview1);

wv.loadUrl("file:///android_asset/Index.html");
```

图 11 中，WebView 加载了内容。

图 11 使用 WebView 加载 HTML 页面

小结

在这篇文章中，你已经看到在 Android 应用程序中如何实现上下文菜单和选项菜单。另外，你还看到如何使用一些比较 cool 的 View，如 AnalogClock 和 DigitalClock。至此，Android Views 系列文章告一段落。我希望你现在已经了解了 Android 中 View 如何工作，并拥有愉快的体验。