Android 布局技巧

Android 平台提供了大量的 UI 构件，你可以将这些小的视觉块（构件）搭建在一起，呈现给用户复杂且有用的画面。然而，应用程序有时需要一些高级的视觉组件。为了满足这一需求，并且能高效的实现，你可以把多个标准的构件结合起来成为一个单独的、可重用的组件。

例如，你可以创建一个可重用的组件包含一个进度条和一个取消按钮，一个 Panel 包含两个按钮（确定和取消动作），一个 Panel 包含图标、标题和描述等等。简单的，你可以通过书写一个自定义的 View 来创建一个 UI 组件，但更简单的方式是仅使用 XML 来实现。

在 Android XML 布局文件里，一般，每个标签都对应一个真实的类实例（这些类一般都是 View 的子类）。UI 工具包还允许你使用三个特殊的标签，它们不对应具体的 View 实例：、、。这篇文章将描述如何使用来创建一个单纯的 XML 视觉组件。了解更多关于如何使用的资料，请参看《合并布局》文章，尤其是它与结合起来使用体现出来的强大威力。

元素的作用如同它的名字一样；它用于包含其它的 XML 布局。使用这个标签如下面的例子所示：

```
<com.android.launcher.Workspace
    android:id="@+id/workspace"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"

    launcher:defaultScreen="1">
```

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

    <include android:id="@+id/cell2"
layout="@layout/workspace_screen" />

    <include android:id="@+id/cell3"
layout="@layout/workspace_screen" />


</com.android.launcher.Workspace>
```

在中，只需要 layout 特性。这个特性，不带 android 命名空间前缀，它表示你想包含的布局的引用。在这个例子中，相同的布局被包含了三次。这个标签还允许你重写被包含布局的一些特性。上面的例子显示了你可以使用 android:id 来指定被包含布局中根 View 的 id；它还可以覆盖已经定义的布局 id。相似的，你可以重写所有的布局参数。这意味着任何 android:layout_*的特性都可以在中使用。下面是例子：

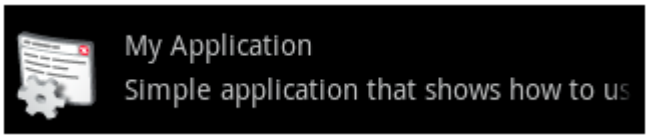
```
 <include android:layout_width="fill_parent"
1 layout="@layout/image_holder" />
2
3<include android:layout_width="256dip"
 layout="@layout/image_holder" />
```

这个标签，在依据设备设置定制 UI 时表现得尤为有用。举个例子，Activity 的主要布局放置在 layout/文件夹下，其它布局放置在 layout-land/和 layout-port/下。这样，在垂直和水平方向时你可以共享大多数的 UI 布局。

Android UI 工具包提供了一些布局管理器，它们使用起来相当容易，而且，大多数的时候，你只需要使用它们最基本的特征来实现 UI。

执着于基本特征的使用对于创建 UI 来说，往往不是最高效的。一个常见的例子就是滥用 LinearLayout，它将会导致 View 树中的 View 数量激增。View——更糟的是，布局管理器——添加到应用程序里都会带来一定的消耗：初始化，布局和绘制变得更加缓慢。嵌套布局的花销尤其“昂贵”，例如，如果你嵌套了一些 LinearLayout，并使用了 weight 参数，这会导致子元素要计算两次。

让我们看一个非常简单且常见的布局例子：一个列表项，左边是一个图标，右边是标题和描述，上方是标题，下方是可选的描述。列表项可能看起来如下图：



为了清楚地认识 View 之间（一个 ImageView 和两个 TextView）的相对位置，下图是使用 HierarchyViewer 抓获的布局剪影：

实现这个布局，直接使用 LinearLayout 就可以了。列表项本身是一个水平的 LinearLayout，里面有一个 ImageView 和一个垂直的 LinearLayout，垂直的 LinearLayout 里包含两个 TextView。以下是这个布局的源代码：

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

android:layout_height="?android:attr/listPreferredItemHei
ght"
    android:padding="6dip">
    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_marginRight="6dip"
        android:src="@drawable/icon" />
    <LinearLayout
        android:orientation="vertical"
        android:layout_width="0dip"
        android:layout_weight="1"
        android:layout_height="fill_parent">
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="0dip"
            android:layout_weight="1"
            android:gravity="center_vertical"
            android:text="My Application" />
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="0dip"
```

```
            android:layout_weight="1"
            android:singleLine="true"
            android:ellipsize="marquee"
            android:text="Simple application that shows how to
use RelativeLayout" />
        </LinearLayout>
</LinearLayout>
```

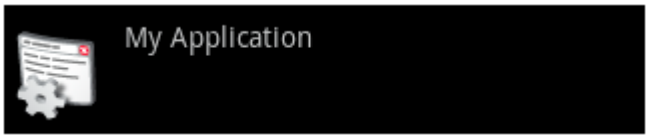
如果你将它作为ListView的item,它能正常工作,但却是相当浪费的。相同的布局可以使用RelativeLayout进行重写,相对于每个列表项来说,可以节省一个View,且View层级上更好,只有一层。使用RelativeLayout也很简单:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android
"
    android:layout_width="fill_parent"

android:layout_height="?android:attr/listPreferredItemHei
ght"
    android:padding="6dip">
    <ImageView
        android:id="@+id/icon"
        android:layout_width="wrap_content"
        android:layout_height="fill_parent"
        android:layout_alignParentTop="true"
        android:layout_alignParentBottom="true"
        android:layout_marginRight="6dip"
        android:src="@drawable/icon" />
<TextView
        android:id="@+id/secondLine"
        android:layout_width="fill_parent"
        android:layout_height="26dip"
        android:layout_toRightOf="@id/icon"
        android:layout_alignParentBottom="true"
        android:layout_alignParentRight="true"
        android:singleLine="true"
        android:ellipsize="marquee"
        android:text="Simple application that shows how to use
RelativeLayout" />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@id/icon"
        android:layout_alignParentRight="true"
```

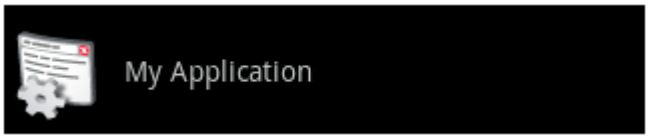
```
        android:layout_alignParentTop="true"
        android:layout_above="@id/secondLine"
        android:layout_alignWithParentIfMissing="true"
        android:gravity="center_vertical"
        android:text="My Application" />
</RelativeLayout>
```

新的实现与老的实现看起来效果完全一致，除了一种情况。每个列表项显示两行文字：标题和可选的描述。当某一个列表项的描述不可获得时，应用程序可能希望将第二个 TextView 的 Visibility 设为 GONE。LinearLayout 实现版表现得很完美，但 RelativeLayout 实现版就有点差强人意了：

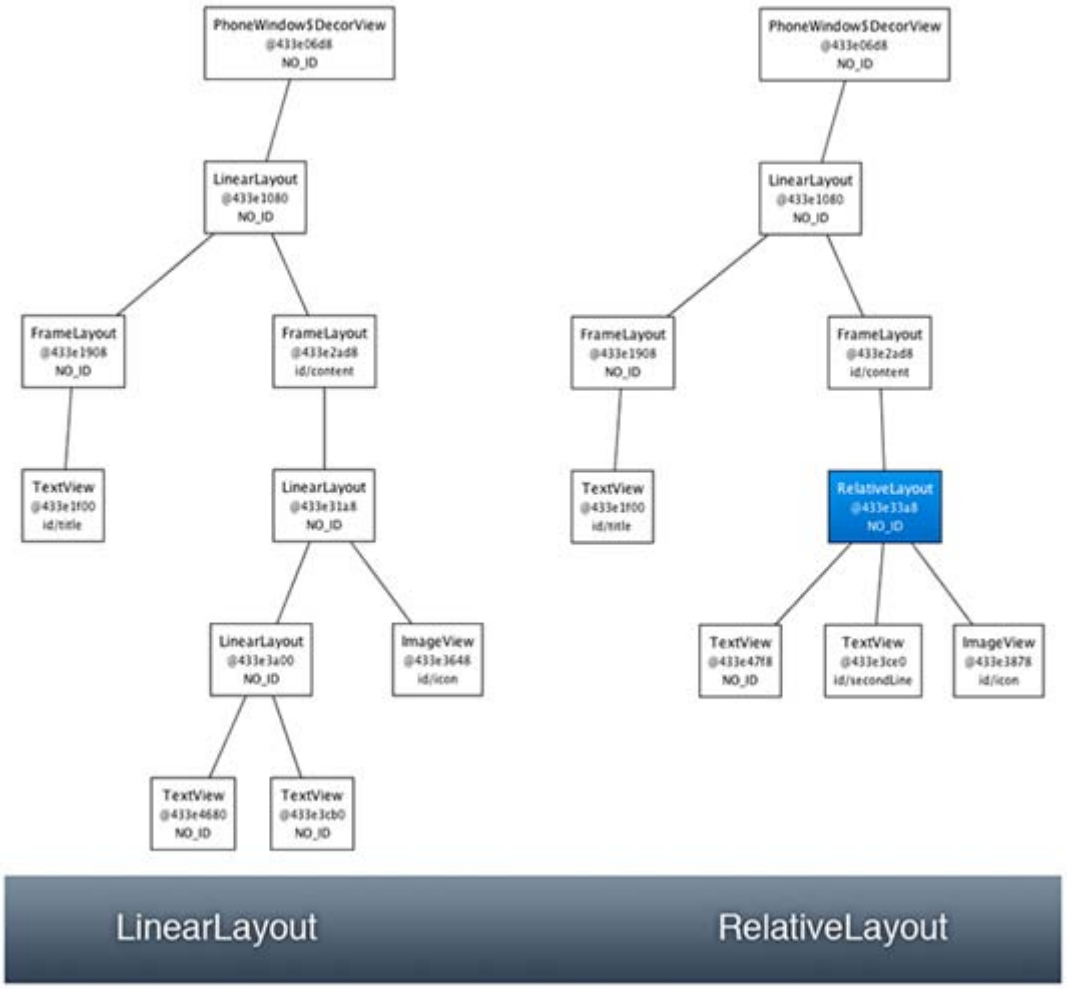


在 RelativeLayout 里，每个 View 都是和父元素 RelativeLayout 对齐或是和其它 View 对齐的。例如，我们声明描述部分是和 RelativeLayout 的底部对齐，标题位于其上并与 RelativeLayout 的顶端对齐。当描述 GONE 时，RelativeLayout 不知道怎么去放置标题的底边缘。为了解决这个问题，你可以使用一个非常简单的布局参数：layout_alignWithParentIfMissing。

这个布尔参数告诉 RelativeLayout：如果目标对象消失时使用自己的边缘作为锚点。例如，如果你放置一个 View 到另一个 Visibiity 属性设为 GONE 的 View 的右边，且设定 alignWithParentIfMissing 为 true，RelativeLayout 就会将其左边缘作为 View 的对齐锚点。在我们的这个场合，使用 alignWithParentIfMissing 的结果是 RelativeLayout 将标题部分的底部与自己的底部对齐。结果如下所示：



现在，我们的布局表现得很完美了，即使描述部分的 Visibility 属性设为 GONE。更好的是，层级更加简单，因为我们不再使用 LinearLayout，而且，更加高效了。当我们使用 HierarchyViewer 来比较两个实现版的时候，事实就更明显了：



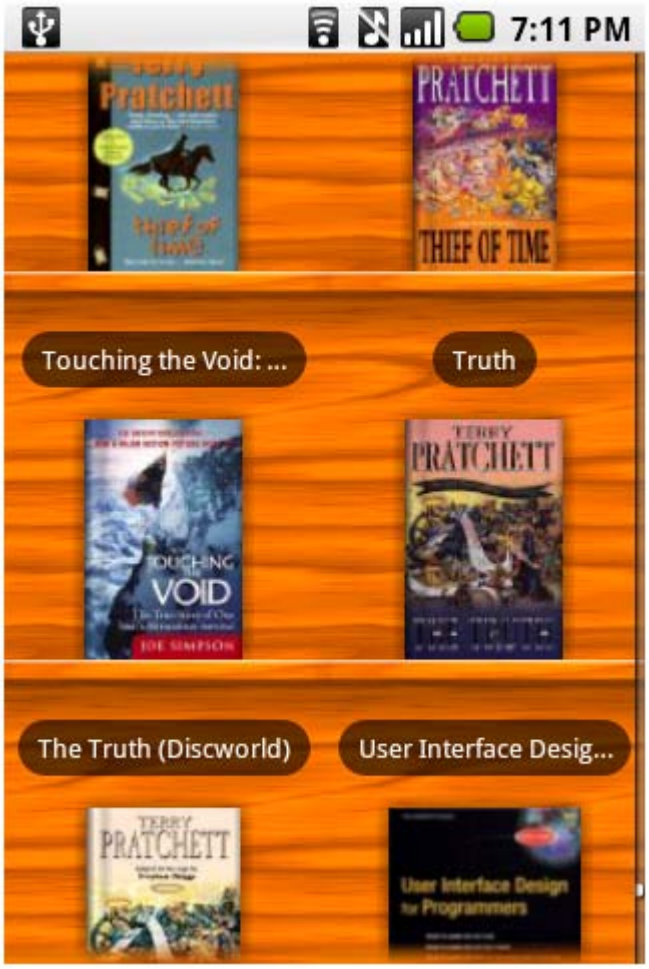
另外，当你使用这么一个布局作为 ListView 的列表项时，这种差异就更为重要了。希望这个简单的例子能让你了解布局，了解如何优化你的 UI。

多亏了标签，在 Android 里，很容易就能做到共享和重用 UI 组件。在 Android 开发中，很容易就能创建出复杂的 UI 结构，结果呢，用了很多的 View，且其中的一些很少使用。针对这种情况，谢天谢地，Android 还为我们提供了一个特别的构件——ViewStub，它可以使你充分享受的好处而不会造成无用 View 的浪费。

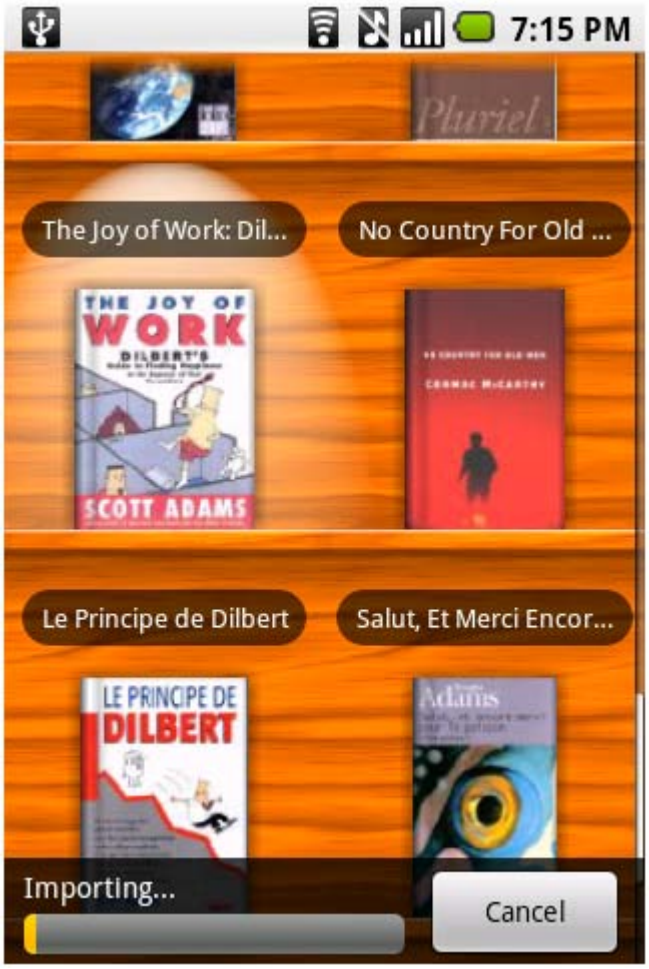
ViewStub

ViewStub 是一个看不见的，轻量级的 View。它没有尺寸，也不会绘制以及以某种形式参与到布局中来。这意味着 ViewStub 去 inflate 以及保留在 View 层次中的代价是很廉价的。ViewStub 最佳的描述称之为“懒惰的 include”。ViewStub 中引用的布局只在你想添加到 UI 上时才会显示。

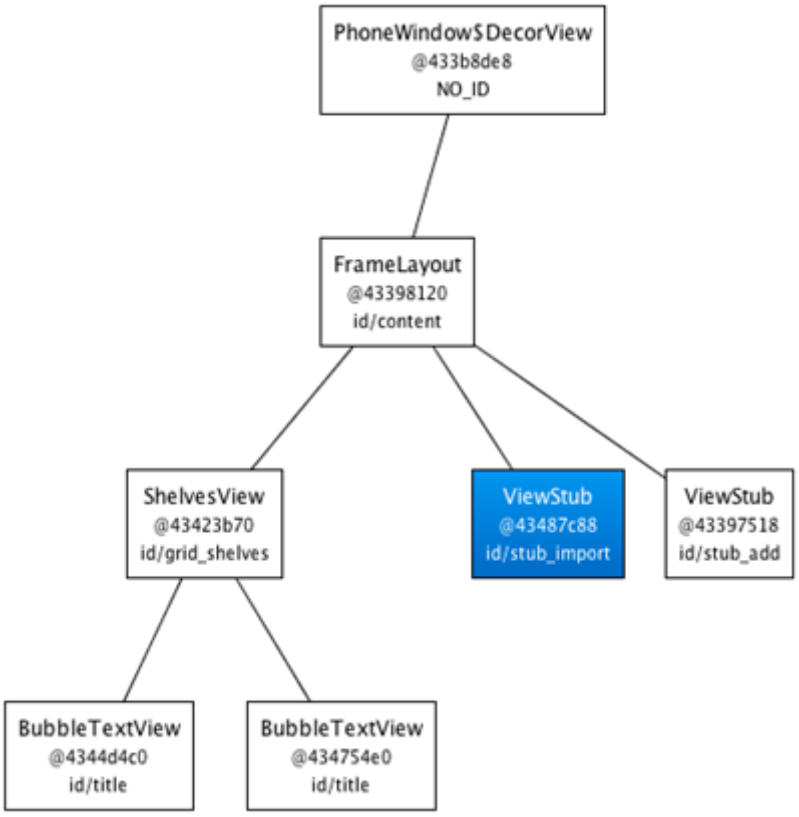
下面的截图来自于 Shelves 应用程序。图中 Activity 显示的内容是给用户呈现可浏览的书籍列表：



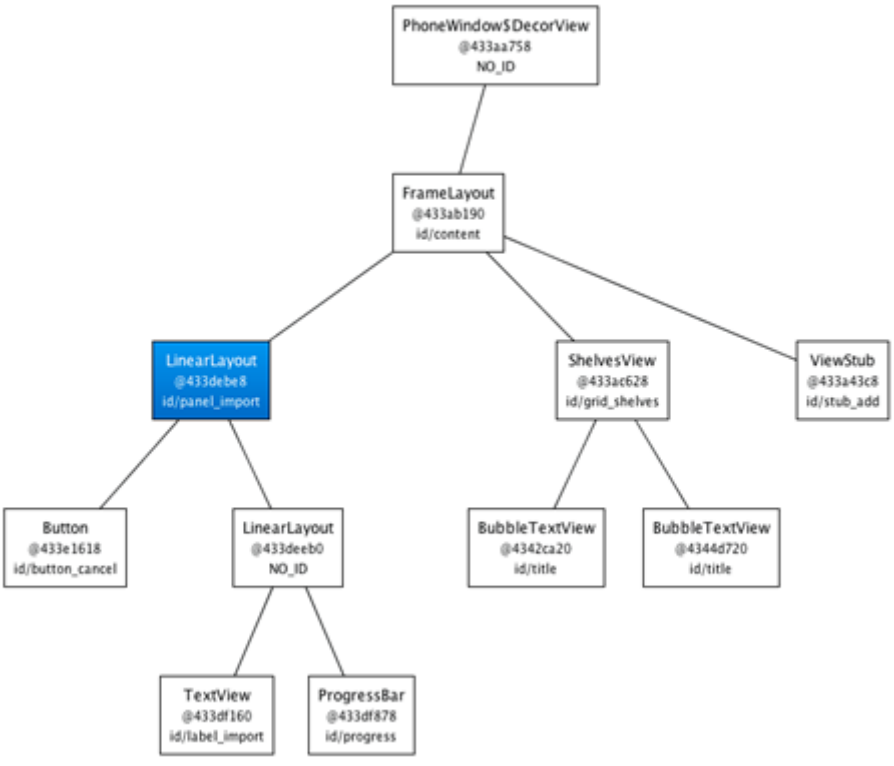
相同的 Activity 也用于用户添加或导入新的书籍。在这个操作中，Shelves 显示了一个额外的 UI。下面的截图显示了在导入期间，会在屏幕的底部显示一个进度表和一个取消按钮：



由于导入书籍不是一个常有的操作，至少相对于浏览书籍列表来说不是，因此，导入 panel 由 ViewStub 来承载：



当用户进行一个导入操作时，ViewStub 被 inflate，此时由它引用的布局文件内容替代显示：



为了使用 ViewStub，你所有需要做的是指定 android:id 特性，便于以后 inflate，指定 android:layout 特性，引用布局文件。ViewStub 还允许你使用第三个特性，android:inflatedId，你可以使用它来重写包含的布局文件中的根元素的 id。最后，在 ViewStub 上设定的 layout_*参数将会应用到包含的布局文件的顶部。这里有个例子：

```
<ViewStub
  android:id="@+id/stub_import"
  android:inflatedId="@+id/panel_import"
  android:layout="@layout/progress_overlay"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:layout_gravity="bottom" />
```

当你准备 inflate ViewStub 时，调用 inflate()方法即可。你还可以设定 ViewStub 的 Visibility 为 VISIBLE 或 INVISIBLE，也会触发 inflate。注意的是，使用 inflate()方法能返回布局文件的根 View：

```
((ViewStub)
findViewById(R.id.stub_import)).setVisibility(View.VISIB
LE);
// or
View importPanel = ((ViewStub)
findViewById(R.id.stub_import)).inflate();
```

有一点需要记住的是：当 ViewStub inflate 后，这个 ViewStub 就从 View 层次中移除了。因此，没有必要保留一个对 ViewStub 的引用（如在类的字段里）。

ViewStub 是快捷编程与高效编程之间的产物。与其手动的 inflate View 并在运行时添加到 View 层次上，不如简单的使用 ViewStub。它相当“廉价”且易于使用。ViewStub 唯一的缺点是现在不支持<merge />标签。

应用程序签名

概述

Android 系统要求，所有的程序经过数字签名后才能安装。Android 系统使用这个证书来识别应用程序的作者，并且建立程序间的信任关系。证书不是用于用户控制哪些程序可以安装。证书不需要授权中心来签名：Android 应用程序上使用自己签名的证书是完全允许且普遍的。

理解 Android 应用程序签名有以下几个重要点：
所有的应用程序都必须签名。系统不会安装任何一个不签名的程序。
· 你可以使用自己的证书来签名。不需要任何授权中心。
· 当你要为最终用户发布你的应用程序的时候，你必须签入一个合适的密钥。你不可以发布程序的时候还使用 SDK 工具签入的 Debug Key。
· 系统只在安装应用程序的时候检测证书的有效期。如果应用程序在安装之后证书失效了，那么，应用程序还是可以正常工作。
· 你可以使用标准工具——Keytool 和 Jarsigner——生成 Key 并签名 apk 文件。
· 一旦你为应用程序签名了，一定要使用 zipalign 工具来优化最终的 APK 包。
Android 系统不会安装和运行没有正确签名的应用程序。这条规则适用于任何运行 Android 系统的地方，不管是真机还是模拟器。正是由于这个原因，你必须在模拟器或真机上运行/调试程序之前对程序进行签名。当你调试应用程序时，Android SDK 工具替你对应用程序进行了签名。Eclipse 的 ADT 插件和 Ant 编译工具都提供了两种签名模式——Debug 模式和 Release 模式。
· 当开发和测试时，你可以使用 Debug 模式 。在 Debug 模式下，编译工具使用内嵌在 JDK 中的 Keytool 工具来创建一个 keystore 和一个 key（包含公认的名字和密码）。在每次编译的时候，使用这个 Debug Key 来为 apk 文件签名。由于密码是公认的，在每次编译的时候，也不需要提示你输入 keystore 和 key 密码。
· 当你的程序准备发布时，你必须在 Release 模式下，使用密钥来为 apk 文件签名。有以下两种方式可以做到：
o 命令行中使用 Keytool 和 Jarsigner。在这个方法中，首先需要编译出一个未签名的 apk。然后使用 Jarsigner（或相似的工具），用你的密钥为 apk 手动签名。如果你没有合适的密钥，你可以运行 Keytool 来手动生成自己的 keystore/key。
o 使用 ADT 导出向导。如果你使用 Eclipse/ADT 插件进行开发，你可以使用导出向导来编译程序，生成密钥（如果需要），并为 apk 签名，所有这些操作都在导出向导中。一旦你的程序签名了，别忘了运行 zipalign 来为 apk 进行额外的优化。
签名策略

应用程序签名的某些方面可能会影响应用程序的开发，特别是你打算一起发布多个应用程序的时候。
一般来说，推荐的策略是在整个应用程序寿命内，所有的程序签上相同的证书。以下有几个应该这么做的原因：
· 应用程序升级——当你对应用程序进行升级时，如果你想用户平稳的升级，那么，你就需要签上相同的证书。当系统安装一个升级应用程序时，如果新版本的证书与老版本的证书有匹配的话，那么，系统才会允许进行升级。如果你没有为版本签上合适的证书，当你安装时，你需要给应用程序指定一个新的包名——在这种情况下，用户安装的新版本，被当作是一个全新的应用程序。
· 应用程序模块化——如果应用程序请求的话，Android 系统允许签有相同证书的应用程序运行在相同的进程里，这样，系统就会把它们看作是一个单一的应用程序。用这种方法配置应用程序，用户可以选择更新每个独立的模块。
· 代码/数据权限共享——Android 系统提供了基于签名的权限检查，因此，如果应用程序间签有特定的证书，那么，它们之间可以共享功能。通过多个程序签有相同的证书并且使用基于签名的权限检查，你的程序可以以一种安全的方式共享代码和数据。

还有一个决定签名策略的重要因素是：如何设定 key 的有效期。
· 如果你计划支持单个应用程序的升级，你需要确保你的 key 拥有一个超过期望的应用程序生命周期的有效期。推荐使用 25 年或更多的有效期。当你的 key 过期了，用户也就不能平稳的更新到新版本了。
· 如果你想给多个无关的应用程序签上相同的 key，那么，你必须确保 key 的有效期超过所有应用程序所有版本的生命周期，包括将来有可能添加到这一阵营的程序。
· 如果你想在 Android Market 上发布你的程序，key 的有效期必须在 2033.10.22 以后。Market 服务器强制这一要求，目前是保证用户可以平稳的更新他们的程序。

当你设计应用程序时，一定要把这些点记在脑子里，并且使用一个合适的证书来为应用程序签名。
签名的基本设定

在你开始之前，你必须保证 Keytool 对 SDK 编译工具来说是可利用的。多数情况下，你可以通过设置 JAVA_HOME 环境变量来告诉 SDK 编译工具如何找到 Keytool。另外，你还可以添加 JDK 中 Keytool 的路径到 PATH 的变量里。
如果你在 Linux 上开发，并且使用 GNU 编译器来编译 Java，那么，请确保系统是使用 JDK 中的 Keytool，而不是 gcj。如果 Keytool 已经在你的 PATH 中，它有可能是对/usr/bin/keytool 的符号链接。在这种情况下，检查符号链接的目标，确保它是指向 JDK 中的 Keytool。
如果你打算对公众释放你的应用程序，你还需要 Jarsigner 工具。Jarsigner 和 Keytool 都包含在 JDK 中。
Debug 模式下签名

Android 编译工具提供了 Debug 签名模式，使得开发和调试应用程序更加容易，而且还满足 Android 系统的签名要求。当使用 Debug 模式编译你的 app 时，SDK 工具会调用 Keytool 工具自动创建一个 Debug 的 keystore 和 key。然后，这个 Debug key 会自动用于 apk 的签名，这样，你不需要使用你自己的 key 来为应用程序包签名。
SDK 工具使用预先定义好的名字/密码来创建 Debug keystore/key：
· Keystore 名字：“debug.keysotre”
· Keystore 密码：“android”
· Key 别名：“androiddebugkey”
· Key 密码：“android”
· CN：“CN=Android Debug,O=Android,C=US”

如果需要的话，你可以改变 Debug keystore/key 的位置和名字，或者提供一个自定义的 Debug keysotre/key。然而，任何自定义的 Debug keystore/key 必须使用和默认 Debug key（上面描述的）相同的名字和密码。（在 Eclipse/ADT 中，操作 Windows>Preferences>Android>Build 实现。）
注意： 你不能将签有 Debug 证书的应用程序发布给公众。

Eclipse 用户

如果你在 Eclipse/ADT 下开发（并且已经按照上面描述的“签名的基本设定”配置了 Keytool），Debug 模式下签名默认是开启的。当你运行或是调试应用程序时，ADT 会使用 Debug 证书进行签名，并运行 zipalign，然后安装到选择的模拟器或是连接上的设备。整个过程不需要你参与，前提是 ADT 能访问 Keytool。
Ant 用户

如果你使用 Ant 来编译你的 apk 文件，需要在 ant 命令中添加 debug 选项来开启 Debug 签名模式（假设你正在使用由 android 工具生成 build.xml 文件）。当你运行 ant debug 来编译你的程序时，编译脚本会生成一个 keystore/key，并为 apk 进行签名。然后脚本会使用 zipalign 工具对 apk 进行对齐处理。整个过程不需要你参与。阅读“其它 IDE 下开发：Debug 模式编译”来了解更多的信息。

Debug 证书过期

Debug 模式下签名用的证书（默认是 Eclipse/ADT 和 Ant 编译）自从它创建之日起，1 年后就会失效。当证书失效时，你会得到一个编译错误，在 Ant 编译上，错误如下：

```
debug:

[echo] Packaging bin/samples-debug.apk, and signing it with
a debug key...

[exec] Debug Certificate expired on 8/4/08 3:43 PM
```

在 Eclipse/ADT 中，Android 控制台上你将会看到一个相似的错误。

为了解决这个问题，只需要删掉 debug.keystore 文件即可。AVD 默认存储的位置在：~/.android/avd（OS X 和 Linux），C:\Documents and Settings\\.android\（Windows XP），C:\Users\\.android\（Windows Vista）。

当下一次编译的时候，编译工具会重新生成一个新的 keystore 和 Debug key。

Release 模式下签名

当你的程序准备好释放给其它用户时，你必须：

1. 获取一个合适的密钥
2. 在 Release 模式下编译程序
3. 使用密钥签名程序
4. 对齐 APK 包

如果你是使用 Eclipse/ADT 插件开发，你可以使用导出向导来完成编译、签名和对齐等操作。在整个过程中，导出向导甚至还可以生成一个新的 keystore 和密钥。因此，如果你使用 Eclipse，你可以直接跳到“使用 Eclipse ADT 编译和签名”。

1．获取一个合适的密钥

为了进行程序的签名，首先，你必须有一个合适的密钥。密钥指：

- 个人持有。
- 代表个人、公司或组织实体的身份。

- 拥有一个有效期。有效期推荐超过 25 年。

如果你在 Android Market 上发布你的程序，需要注意一点的是：程序的有效期需要在 2033.10.22 之后。你不能上传一个应用程序，而它的 key 的有效期是在这个日期之前。

- 不是由 Android SDK 工具生成的 Debug key。

如果你没有一个合适的 key，你一定要使用 Keytool 来生成一个。如“基本设定”中描述的，确保 Keytool 可用。

为了用 Keytool 生成一个 key，使用 keytool 命令并传入一些可选参数，如下表所示。

警告：确保密钥的安全。一定要阅读“安全储存你的密钥”中讨论如何确保你的密钥的安全以及这对你和用户为何如此重要。尤其是，当你生成你的密钥时，一定要为 keystore 和 key 使用强密码。

Keytool 选项	**描述**
-genkey	生成一个键对（公钥和密钥）
-v	日志输出
-keystore .keystore	命名包含密钥的 keystore
-storepass	设定 keystore 的密码。 基于安全考虑，不要在命令行中包含这一选项，除非你在一个安全的计算机上。如果你不提供，Keytool 会提示你输入。这样，你的密码就不会储存在 shell 记录中了。
-alias	设定 key 的别名
-keyalg	指定生成 key 时使用的加密算法。支持 DSA 和 RSA。
-dname	指定一个描述谁创建 key 的名字。该值将填入证书发行人的字段。 注意：你不需要在命令行中指明这些选项。如果不提供，Jarsigner 会提示你输入每个字段（CN，OU 等）。
-validity	Key 的有效期，天数。 注意：推荐使用 10000 或更大的数。
-keypass	Key 的密码。 基于安全考虑，不要在命令行中包含这一选项，除非你在一个安全的计算机上。如果你不提供，Keytool 会提示你输入。这样，你的密码就不会储存在 shell 记录中了。

下面是使用 Keytool 命令生成密钥的例子：

```
$ keytool -genkey -v -keystore my-release-key.keystore

-alias alias_name -keyalg RSA -validity 10000
```

运行上面的例子命令，Keytool 会提示你输入 keystore 和 key 的密码，并且会提示你输入 key 中其它的字段。然后，它会生成一个叫做 my-release-key.keystore 的文件。keystore 和 key 受你输入的密码保护。keystore 中包含一个 key，有效期为 10000 天。别名将在后面用到，在程序签名时指当前这个 keystore。

了解更多关于 Keytool 的信息，请参考 http://java.sun.com/j2se/1.5.0/docs/tooldocs/#security。

2. 在 Release 模式下编译程序

为了给用户释放程序，你必须在 Release 模式下编译程序。在 Release 模式下，编译程序不会进行默认签名，并且你需要使用密钥进行签名。

注意: 你不能释放未签名的程序，或签有 Debug key 的程序。

使用 Eclipse

右击 Package Explorer 中的工程，选择 Android Tools>Export Unsigned Application Package，导出一个未签名的 apk。然后指定未签名 apk 的存储位置。（另外，你也可以在 Eclipse 中打开 AndroidManifest.xml 文件，打开 Manifest Tab，然后点击 Export an unsigned APK）。

注意：你也可以使用导出向导来完成编译和签名步骤，参考“使用 Eclipse/ADT 编译和签名”。

使用 Ant

如果你正在使用 Ant，那么你可以在 ant 命令中加入 release 选项来开启 Release 模式。例如，如果你在包含 build.xml 文件的文件夹上运行 Ant，命令可能看起来是这样：

```
ant release
```

一般，编译脚本在编译 apk 的时候不会进行签名。输出的文件位于工程的 bin/文件夹中，名为 -unsigned.apk。由于 apk 文件还没有签名，所以你必须手动的使用密钥进行签名，然后使用 zipalign 进行对齐。

然而，Ant 编译脚本也可以替你执行签名和对齐操作，前提是你在 build.properties 文件中提供了 keystore 的名字和 key 的别名。如果提供了这些信息，编译脚本在执行 ant release 命令时会提示你输入 keystore 和 key 的密码，对包进行签名并对齐。最后输出的文件位于工程的 bin/文件夹中，名为 -release.apk。如果按照上述自动签名和对齐操作执行，那么，你就可以跳过下面的手动步骤（步骤 3 和 4）。了解如何在 build.properties 文件中指定 keystore 和 alias，请参考“其它 IDE 下开发：Release 模式编译”。

3. 使用密钥签名程序

如果你已经准备好要签名的程序包的话，你可以使用 Jarsigner 工具进行签名。如“基本设定”中描述的，请确保 Jarsigner 工具可用。此外，确保包含密钥的 keystore 可用。

为了签名应用程序，你需要运行 Jarsigner，并引用应用程序的 apk 及包含密钥的 keystore。下表列出了你可能使用的选项。

Jarsigner 选项	**描述**
-keystore .keystore	指定包含密钥的 keystore 的名字
-verbose	日志输出
-storepass	指定 keystore 的密码。 基于安全考虑，不要在命令行中包含这一选项，除非你在一个安全的计算机上。如果你不提供，Jarsigner 会提示你输入。这样，你的密码就不会储存在 shell 记录中了
-keypass	指定密钥的密码。 基于安全考虑，不要在命令行中包含这一选项，除非你在一个安全的计算机上。如果你不提供，Jarsigner 会提示你输入。这样，你的密码就不会储存在 shell 记录中了

下面是一个使用 Jarsigner 对 my_application.apk 进行签名的例子，使用了上面创建的 keystore。

```
1$ jarsigner -verbose -keystore my-release-key.keystore
2
3my_application.apk alias_name
```

运行上面的示例命令，Jarsigner 会提示你输入 keystore 和 key 的密码。然后它会修改 apk 文件，意味着 apk 文件现在已经签上名了。注意：你可以使用不同的 key 对 apk 多次签名。

检验 apk 文件是否签名，可以使用下面的命令：

```
1$ jarsigner -verify my_signed.apk
```

如果 apk 签名正确，Jarsigner 输出“jar verified”。如果你想了解更多的细节，你可以尝试这些命令：

```
1$ jarsigner -verify -verbose my_application.apk
```

或者

```
1$ jarsigner -verify -verbose -certs my_application.apk
```

上面的命令，添加-certs 选项，将会显示“CN=”行，描述谁创建了密钥。

注意: 如果你看到“CN=Android Debug”，这意味 apk 文件使用 Android SDK 生成的 Debug key 签的名。如果你想发布你的程序，你必须使用自己的密钥替换 Debug key 进行签名。
了解更多 Jarsigner 的信息，请参考 http://java.sun.com/j2se/1.5.0/docs/tooldocs/#security。
4. 对齐 APK 包

一旦你对 apk 文件使用密钥进行签名后，一定要运行 zipalign 进行对齐。这个工具能做到让那些未压缩的数据以特定的字节对齐。以 4 字节对齐能优化性能。当对齐后，Android 系统能通过 mmap()阅读文件，即使它们包含二进制数据，而不是从包中拷贝所有的数据。好处是在运行程序时减少了随机读取内存的消耗。
zipalign 由 Android SDK 提供，包含在 tools/文件夹下。想对齐签名后的 apk，执行：

```
zipalign -v 4 your_project_name-unaligned.apk
your_project_name.apk
```

-v 标志表示开始日志输出（可选）。4 表示对齐的字节（不要使用非 4 的数字）。第一个文件参数是你的签名后 apk（输入），第二个文件参数是目的 apk 文件（输出）。如果你想覆盖已经存在的 apk，添加-f 标志。

注意： 在你使用 zipalign 优化包之前，输入的 apk 必须是使用密钥签名后的。如果在 zipalign 操作之后再签名，那么对齐操作就白做了。

了解更多信息，请阅读“zipalign”工具的文档。

使用 Eclipse / ADT 编译和签名

如果你在使用 Eclipse/ADT 插件，你可以使用导出向导导出一个签名的 apk（甚至可以创建一个新的 keystore，如果需要的话）。导出向导替你做了所有与 Keytool 和 Jarsigner 交互的工作，并且使用 GUI 来签名应用程序，替代了上面提到的手动编译、签名和对齐的操作。一旦向导完成了编译和签名，也还会使用 zipalign 执行包的对齐操作。由于导出向导要使用 Keytoo 和 Jarsigner。你应该确保它们是可用的，如“签名的基本设定”中描述的那样。

安全储存你的密钥

维护密钥的安全是极其重要的。如果你让其他人使用了你的 key，或者你把 keystore 和密码放在一个不安全的地方以至于第三方人员找到并使用了的话，那么，你的授权认证和用户的信任都将受到连累。

如果第三方没有经过你的允许拿走了你的 key，那个人就可以签名并发布应用程序，并恶意替换或攻击你的正版程序。这个人还可以签名并发布应用程序，利用你的名义来攻击其它程序或者系统自身，或者破坏、偷取用户数据。

开发者的名声依赖于正确的储存你的密钥，直到它过期。这里有几个安全储存密钥的提示：

- 为 key 和 keystore 设置强密码。
- 当你使用 Keytool 生成密钥的时候，不要再命令行中添加-storepass 和-keypass 选项。如果你这样做了，在 shell 记录中就可以获取你的密码。
- 相似的，当使用 Jarsigner 来签名程序时，也不要在命令行中添加-storepass 和-keypass 选项。
- 不要把密钥给别人或借给别人，不要让未授权的人知道你的 keystore 和 key 的密码。

总而言之，只要你在生成、使用和储存密钥时有安全意识的话，它还是很安全的。

XML 中 定义 menu

和 Android UI layout 一样，我们也可以在 XML 中定义应用程序的菜单。通过在菜单的 onCreateOptionsMenu 方法中膨胀菜单 layout。这样做会使我们的程序代码简单多了，而且尽可能的将更多的界面设计部分放到 XML，便于浏览。

1. 在工程的/res/文件夹下创建 menu 文件夹，用来保存你为应用程序定义的菜单 XML 文件。

```
<?xml version="1.0" encoding="utf-8"?>
<menu
xmlns:android="http://schemas.android.com/apk/res/android
">
<item android:title="Play" android:id="@+id/media_play"
  android:icon="@android:drawable/ic_media_play"/>
<item android:title="Pause" android:id="@+id/media_pause"
  android:icon="@android:drawable/ic_media_pause"/>
<item android:title="File" android:id="@+id/file">
<menu>
<item android:title="Open..."
android:id="@+id/file_open"/>
<item android:title="Save" android:id="@+id/file_save"/>
<item android:title="Save as"
android:id="@+id/file_saveas"/>
<item android:title="Exit" android:id="@+id/file_exit"/>
</menu>
</item>
<item android:title="Edit" android:id="@+id/edit">
<menu>
<group>
<item android:title="Copy" android:id="@+id/edit_copy"/>
<item android:title="Paste" android:id="@+id/edit_paste"/>
<item android:title="Cut" android:id="@+id/edit_cut"/>
<item android:title="Delete"
android:id="@+id/edit_delete"/>
</group>
</menu>
</item>
</menu>
```

在菜单 XML layout 中，有三个有效的元素：menu、group、item。item 和 group 必须是 menu 的子元素，且 item 必须是 group 的子元素。另外的 menu 可以是 item 的子元素（为了创建子菜单）。下面的 XML 片段显示了菜单的层次定义：

2. 重写 Activity 的 onCreateOptionsMenu 方法，通过 MenuInflater.inflate 方法来膨胀菜单 XML。

```
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.menu_option, menu);
```

3. 在 Activity 的 onOptionsItemSelected 方法中处理每个菜单项的点击事件：

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
super.onOptionsItemSelected(item);
switch (item.getItemId()) {
case R.id.media_play:
break;
case R.id.media_pause:
break;
case R.id.file_open:
break;
case R.id.file_save:
…
}
return true;
}
```

在 XML 可以定义菜单项的图标、快捷键、checkbox 等更多特征，了解更多请查阅 SDK 中关于菜单的主题。

演示的效果如图：


9:22 AM
XMLMenu
Hello World, XMLMenu!
Play
Pause
File
Edit
MENU

Android 传感器 随处监控您的环境

对于 Java™ 开发人员来说，Android 平台是通过使用硬件传感器创建创新应用程序的理想平台。我们将学习一些可用于 Android 应用程序的接口连接选项，包括使用传感器子系统和录制音频片段。

利用配备 Android 的设备的硬件功能可以构建哪些应用程序呢？任何需要电子监视和监听的应用程序都可以构建。婴儿监视器、安全系统，甚至地震仪都可以。理论上讲，您不能同时 出现在两个地方，但 Android 可以利用一些可行的方法实现这一点。纵观本文始末，您必须记住，使用的 Android 设备不仅仅局限于“手机”，还可以是部署在固定位置、具有无线网络连接的设备，比如 EDGE 或 WiFi。下 载 本文示例的源文件。

Android 传感器功能

使用 Android 平台有一个很新颖的地方，那就是您可以在设备内部访问一些“好工具”。过去，访问设备底层硬件的能力一度让移动开发人员感到非常棘手。尽管 Android Java 环境的角色仍然是您和设备的桥梁，但 Android 开发团队让许多硬件功能浮出了水面。该平台是一个开源平台，因此您可以自由地编写代码实现您的任务。

如果尚未安装 Android，您可以 下载 Android SDK。您还可以 浏 览 android.hardware 包的内容并参考本文的示例。android.media 包 包含了一些提供有用和新颖功能的类。

Android SDK 中包含的一些面向硬件的功能描述如下。

表 1. Android SDK 中提供的面向硬件的特性

特性	描 述
android.hardware.Camera	允许应用程序与相 机交互的类，可以截取照片、获取预览屏幕的图像修改用来治理相机操作的参数。
android.hardware.SensorManager	允 许访问 Android 平台传感器的类。并非所有配备 Android 的设备都支持 SensorManager 中的所有传感器，虽然这种可能性让人非常兴奋。（可用传感器的简介见下文）
android.hardware.SensorListener	在 传感器值实时更改时，希望接收更新的类要实现的接口。 应用程序实现该接口来监视硬件中一个或多个可用传感器。 例如，本文中的 代 码 包含实现该接口的类，实现后可以监视设备的方向和内置的加速表。
android.media.MediaRecorder	用 于录制媒体样例的类，对于录制特定位置（比如婴儿保育）的音频活动非常有用。 还可以分析音频片段以便在访问控件或安全应用程序时进行身份鉴定。
android.FaceDetector	例如，它可 以帮助您通过声音打开门，以节省时间，不需要从房产经纪人处获取钥匙。 允 许对人脸（以位图形式包含）进行基本识别的类。不可能有两张完全一样的脸。 可以使用该类作为设备锁定方法，无需记密码 — 这是手机的生物特征识别功能。
android.os.*	包 含几个有用类的包，可以与操作环境交互，包括电源管理、文件查看器、处理器和消息类。 和许多可移动设备一样，支持 Android 的电话可能会消耗大量电能。 让设备在正确的时间“醒来”以监视感兴趣的事件是在设计时需要首先关注的方面。

java.util.Date java.util.Timer java.util.TimerTask	当测量实际的事件时，数据和时间往往很重要。 例如，java.util.Date 类允许您在遇到特定的事件或状况时获取时间戳。 您可以使用 java.util.Timer 和 java.util.TimerTask 分别执行周期性任务或时间点任务。

android.hardware.SensorManager 包含几个常量，这表示 Android 传感器系统的不同方面，包括：

传感器类型

方向、加速表、光线、磁场、临近性、温度等。

采样率

最快、游戏、普通、用户界面。当应用程序请求特定的采样率时，其实只是对传感器子系统的一个提示，或者一个建议。不保证特定的采样率可 用。

准确性

高、低、中、不可靠。

SensorListener 接口是传感器应用程序的中心。它包括两个必需方法：

- onSensorChanged(int sensor,float values[]) 方法在传感器值更改时调用。该方法只对受此应用程序监视的传感器调用（更多内容见下文）。该方法的参数包括：一个整数，指示更改的传感器；一个浮点值数 组，表示传感器数据本身。有些传感器只提供一个数据值，另一些则提供三个浮点值。方向和加速表传感器都提供三个数据值。
- 当传感器的准确性更改时，将调用 onAccuracyChanged(int sensor,int accuracy) 方法。参数包括两个整数：一个表示传感器，另一个表示该传感器新的准确值。

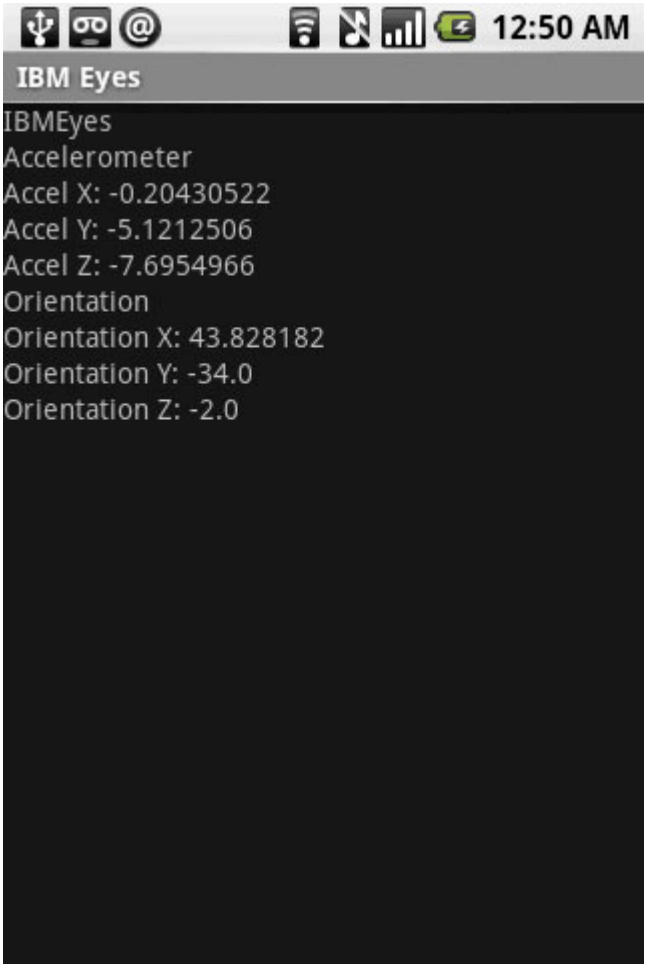
要与传感器交互，应用程序必须注册以侦听与一个或多个传感器相关的活动。注册使用 SensorManager 类的 registerListener 方法完成。本文中的 代 码示例 演示了如何注册和注销 SensorListener。

记住，并非所有支持 Android 的设备都支持 SDK 中定义的所有传感器。如果某个传感器无法在特定的设备上使用，您的应用程序就会适当地降级。

传感器示例

样例应用程序仅监控对方向和加速表传感器的更改（源代码见 下 载）。当收到更改时，传感器值在 TextView 小部件的屏幕上显示。图 1 展示了该应用程序的运行情况。

图 1. 监视加速和方向



使用 Eclipse 环境和 Android Developer Tools 插件创建的应用程序。（关于使用 Eclipse 开发 Android 应用程序的信息，请参见 参 考资料。）清单 1 展示了该应用程序的代码。

清单 1. IBMEyes.java

```
package com.msi.ibm.eyes;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
import android.hardware.SensorManager;
import android.hardware.SensorListener;
public class IBMEyes extends Activity implements
SensorListener {
    final String tag = "IBMEyes";
    SensorManager sm = null;
    TextView xViewA = null;
    TextView yViewA = null;
    TextView zViewA = null;
    TextView xViewO = null;
    TextView yViewO = null;
```

```
    TextView zViewO = null;

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
       // get reference to SensorManager
        sm = (SensorManager)
getSystemService(SENSOR_SERVICE);
        setContentView(R.layout.main);
        xViewA = (TextView) findViewById(R.id.xbox);
        yViewA = (TextView) findViewById(R.id.ybox);
        zViewA = (TextView) findViewById(R.id.zbox);
        xViewO = (TextView) findViewById(R.id.xboxo);
        yViewO = (TextView) findViewById(R.id.yboxo);
        zViewO = (TextView) findViewById(R.id.zboxo);
    }
    public void onSensorChanged(int sensor, float[] values)
{
        synchronized (this) {
            Log.d(tag, "onSensorChanged: " + sensor + ", x: "
+
values[0] + ", y: " + values[1] + ", z: " + values[2]);
            if (sensor == SensorManager.SENSOR_ORIENTATION) {
                xViewO.setText("Orientation X: " + values[0]);
                yViewO.setText("Orientation Y: " + values[1]);
                zViewO.setText("Orientation Z: " + values[2]);
            }
            if (sensor == SensorManager.SENSOR_ACCELEROMETER)
{
                xViewA.setText("Accel X: " + values[0]);
                yViewA.setText("Accel Y: " + values[1]);
                zViewA.setText("Accel Z: " + values[2]);
            }
        }
    }

    public void onAccuracyChanged(int sensor, int accuracy)
{
      Log.d(tag,"onAccuracyChanged: " + sensor + ", accuracy:
" + accuracy);
    }
    @Override
    protected void onResume() {
```

```
        super.onResume();
      // register this class as a listener for the orientation
and accelerometer sensors
        sm.registerListener(this,
                SensorManager.SENSOR_ORIENTATION
|SensorManager.SENSOR_ACCELEROMETER,
                SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onStop() {
        // unregister listener
        sm.unregisterListener(this);
        super.onStop();
    }
}
```

编写应用程序必须基于常见的活动，因为它只是利用从传感器获取的数据更新屏幕。在设备可能在前台执行其他活动的应用程序中，将应用程序构建为服务可 能更加合适。

该活动的 onCreate 方法可以引用 SensorManager，其中包含所 有与传感器有关的函数。onCreate 方法还建立了对 6 个 TextView 小部件的引用，您需要使用传感器数据值更新这些小部件。

onResume() 方法使用对 SensorManager 的引用通过 registerListener 方法注册传感器更新：

- 第一个参数是实现 SensorListener 接口的类的实例。
- 第二个参数是所需传感器的位掩码。在本例中，应用程序从 SENSOR_ORIENTATION 和 SENSOR_ACCELEROMETER 请求数据。
- 第三个参数是一个系统提示，指出应用程序更新传感器值所需的速度。

应用程序(活动)暂停后，需要注销侦听器，这样以后就不会再收到传感器更新。这通过 SensorManager 的 unregisterListener 方法实现。惟一的参数是 SensorListener 的实例。

在 registerListener 和 unregisterListener 方法调用中，应用程序使用关键字 this。注意类定义中的 implements 关键字，其中声明了该类实现 SensorListener 接口。这就是要将它传递到 registerListener 和 unregisterListener 的原因。

SensorListener 必须实现两个方法 onSensorChange 和 onAccuracyChanged。示例应用程序不关心传感器的准确度，但关注传感器当前的 X、Y 和 Z 值。onAccuracyChanged 方法实质上不执行任何操作；它只在每次调用时添加一个日志项。

似乎经常需要调用 onSensorChanged 方法，因为加速表和方向传感器正在快速发送数据。查看第一个参数确定哪个传感器在发送数据。确认了发送数据的传感器之后，将使用方法第二个参数传递的浮点 值

数组中所包含的数据更新相应的 UI 元素。该示例只是显示这些值，但在更加高级的应用程序中，还可以分析这些值，比较原来的值，或者设置某种模式识别算法来确定用户（或外部环境）的行为。

现在您已经了解了传感器子系统，接下来的部分将回顾一个在 Android 手机上录制音频的代码样例。该样例运行在 DEV1 开发设备上。

使用 MediaRecorder

android.media 包包含与媒体子系统交互的类。使用 android.media.MediaRecorder 类进行媒体采样，包括音频和视频。MediaRecorder 作为状态机运行。您需要设置不同的参数，比如源设备和格式。设置后，可执行任何时间长度的录制，直到用户停止。

清单 2 包含的代码在 Android 设备上录制音频。显示的代码不包括应用程序的 UI 元素（完整源代码见 下 载）。

清单 2. 录制音频片段

```
MediaRecorder mrec ;
File audiofile = null;
private static final String TAG="SoundRecordingDemo";
protected void startRecording() throws IOException
{
   mrec.setAudioSource(MediaRecorder.AudioSource.MIC);

mrec.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP
);

mrec.setAudioEncoder(MediaRecorder.AudioEncoder.AMR_NB);
   if (mSampleFile == null)
   {
       File sampleDir =
Environment.getExternalStorageDirectory();
       try
       {
          audiofile = File.createTempFile("ibm", ".3gp",
sampleDir);
       }
       catch (IOException e)
       {
           Log.e(TAG,"sdcard access error");
           return;
       }
   }
   mrec.setOutputFile(audiofile.getAbsolutePath());
```

```
   mrec.prepare();
   mrec.start();
}
protected void stopRecording()
{
   mrec.stop();
   mrec.release();
   processaudiofile(audiofile.getAbsolutePath());
}
protected void processaudiofile()
{
   ContentValues values = new ContentValues(3);
   long current = System.currentTimeMillis();
   values.put(MediaStore.Audio.Media.TITLE, "audio" +
audiofile.getName());
   values.put(MediaStore.Audio.Media.DATE_ADDED, (int)
(current / 1000));
   values.put(MediaStore.Audio.Media.MIME_TYPE,
"audio/3gpp");
   values.put(MediaStore.Audio.Media.DATA,
audiofile.getAbsolutePath());
   ContentResolver contentResolver = getContentResolver();

   Uri base = MediaStore.Audio.Media.EXTERNAL_CONTENT_URI;
   Uri newUri = contentResolver.insert(base, values);

   sendBroadcast(new
Intent(Intent.ACTION_MEDIA_SCANNER_SCAN_FILE, newUri));
}
```

在 startRecording 方法中，实例化并初始化 MediaRecorder 的实例：

- 输入源被设置为麦克风（MIC）。
- 输出格式被设置为 3GPP（*.3gp 文件），这是移动设备专用的媒体格式。
- 编码器被设置为 AMR_NB，这是音频格式，采样率为 8 KHz。NB 表示窄频。SDK 文档 解释了不同的数据格式和可用的编码器。

音频文件存储在存储卡而不是内存中。External.getExternalStorageDirectory() 返回存储卡位置的名称，在该目录中将创建一个临时文件名。然后，通过调用 setOutputFile 方法将文件关联到 MediaRecorder 实例。音频数据将存储到该文件中。

调用 prepare 方法完成 MediaRecorder 的初始化。准备开始录制流程时，将调用 start 方法。在调用 stop 方法之前，将对存储卡上的文件进行录制。release 方法将释放分配给 MediaRecorder 实例的资源。

音频采样完成之后，需要采取以下步骤：

- 向设备的媒体库添加该音频。
- 执行一些模式识别步骤确定声音：
  - 这是婴儿的啼哭声吗？
  - 这是所有人的声音吗？是否要解锁手机？
  - 这是“芝麻开门”吗？是否要打开通往“秘密通道”的大门？
- 自动将音频文件上传到网络位置以便处理。

在该代码样例中，processaudiofile 方法将音频添加到媒体库。使用 Intent 通知设备上的媒体应用程序有新内容可用。

关于该代码片段最后要注意的是：如果您试用，它一开始不会录制音频。您将看到创建的文件，但是没有任何音频。您需要向 AndroidManifest.xml 文件添加权限：

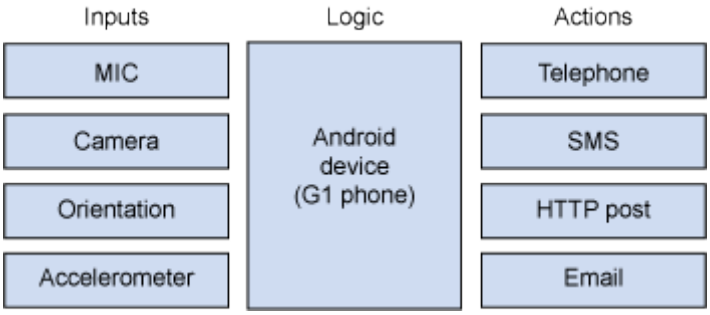
```
<uses-permission
android:name="android.permission.RECORD_AUDIO"></uses-permission>
```

现在，您已经学了一点关于与 Android 传感器和录制音频相关的内容。下一节将更全面的介绍与数据采集和报告系统有关的应用程序架构。

Android 作为传感器平台

Android 平台包含各种用于监视环境的传感器选项。有了输入或模拟选项数组，以及高级计算和互联功能，Android 成为构建实际系统的最佳平台。图 2 显示了输入、应用程序逻辑、通知方法或输出之间的简单视图。

图 2. 以 Android 为中心的传感器系统的方块图



该架构很灵活；应用程序逻辑可以划分为本地 Android 设备和服务器端资源（可以实现更大的数据库和计算功能）。例如，本地 Android 设备上录制的音轨可以 POST 到 Web 服务器，其中将根据音频模式数据库比较数据。很明显，这仅仅是冰山一角。希望您能更深入地研究，让 Android 平台超越移动电话的范畴。

结束语

在本文中，我们介绍了 Android 传感器。样例应用程序度量了方向和加速，以及使用 MediaRecorder 类与录制功能进行交互。对于构建实际系统，Android 是一个灵活、有吸引力的平台。Android 领域发展迅速，并且不断壮大。请务必关注该平台。

Andriod 应用对 XML 的解析

Android 系统中，对于 XML 文件的读取主要采用的是 SAX 方法。SAX 是一个用于处理 XML 事件驱动的“推”模型，虽然它不是 W3C 标准， 但它却是一个得到了广泛认可的 API。SAX 解析器不像 DOM 那样建立一个完整的文档树，而是在读取文档时激活一系列事件，这些事件被推给事件处理器，然 后由事件处理器提供对文档内容的访问。

常见的事件处理器有三种基本类型：

a. 用于访问 XML DTD 内容的 DTDHandler；

b. 用于低级访问解析错误的 ErrorHandler；

c. 用于访问文档内容的 ContentHandler，这也是最普遍使用的事件处理器。

因为 SAX 具有边扫描边读取的特性，无需将整个文件读入内存中，更好的节省了移动设备中的资源。但是这种方法对于操作节点显得略有复杂。

下面以本人在学习中所看过的一个视频教程中的实例，说明 Android 应用程序对于 XML 文档的解析过程。

用于开启解析的 Activity：

```
    public class XMLActitity extends Activity {
        /** Called when the activity is first created. */
        private Button parseButton ;
        @Override
        public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            parseButton =
(Button)findViewById(R.id.parseButton);
            parseButton.setOnClickListener(new
ParseButtonListener());
        }

        //设定监听器，当点击按钮时，开始进行解析。
        class ParseButtonListener implements OnClickListener{

            @Override
            public void onClick(View v) {
                HttpDownloader hd = new HttpDownloader();
    //将网络上的一个 XML 资源下载并解析成为一个字符串，这里的解析在一
```

```
个外部类中进行，这里不再列出。
            String resultStr =
hd.download("http://107:8081/voa1500/test.xml");
            System.out.println(resultStr);
            try{
  //下面是解析XML 的

                // 创建一个SAXParserFactory
                SAXParserFactory factory =
SAXParserFactory.newInstance(); // 获得一个XMLReader
                XMLReader reader =
factory.newSAXParser().getXMLReader();
                // 为XMLReader 设置内容处理器
                reader.setContentHandler(new
MyContentHandler());
                //开始解析文件。注意这个方法的参数，需要是输入流。

还有另外一种不同参数，需要时系统内部资源的URI
                reader.parse(new InputSource(new
StringReader(resultStr)));

            }
            catch(Exception e){
                e.printStackTrace();
            }
        }

    }
}
```

内容处理器:

SAX 时间作为驱动的解析模型，需要相应的事件处理函数。SAX 将不同的事件处理函数按照类型分不同接口中。对于文件内容的处理函数，在内容处 理器接口（ContentHandler）中。

```
public class MyContentHandler extends DefaultHandler {
    String hisname, address, money, sex, status;
    String tagName;
```

```
    //当开始解析文档时，触发这个函数
        public void startDocument() throws SAXException {
            System.out.println("````````begin````````");
        }
    //当结束解析文档时，触发这个函数
        public void endDocument() throws SAXException {
            System.out.println("````````end````````");
        }
    //当扫描到开始标签时,触发这个函数。
        public void startElement(String namespaceURI, String
localName,
                String qName, Attributes attr) throws
SAXException {
            //使用全局变量tagName 作为当前正在解析节点的标记。
    tagName = localName;
            if (localName.equals("worker")) {
                //获取标签的全部属性。标签的所有属性存储在数组中。
                for (int i = 0; i < attr.getLength(); i++) {
    //打印其中某个属性的名称和属性值
                    System.out.println(attr.getLocalName(i) +
"=" + attr.getValue(i));
                }
            }
        }

        public void endElement(String namespaceURI, String
localName, String qName)
                throws SAXException {
            //在workr 标签解析完之后，会打印出所有得到的数据
            tagName = "";
            if (localName.equals("worker")) {
                this.printout();
            }
        }
    //当扫描到标签内容时，触发这个函数。标签的内容存储在字符数组中。
        public void characters(char[] ch, int start, int
```

```
length)
            throws SAXException {
    //首先判断当前扫描到的标签，然后根据当前标签判断标签内容
        if (tagName.equals("name"))
            hisname = new String(ch, start, length);
        else if (tagName.equals("sex"))
            sex = new String(ch, start, length);
        else if (tagName.equals("status"))
            status = new String(ch, start, length);
        else if (tagName.equals("address"))
            address = new String(ch, start, length);
        else if (tagName.equals("money"))
            money = new String(ch, start, length);
    }

    private void printout() {
        System.out.print("name: ");
        System.out.println(hisname);
        System.out.print("sex: ");
        System.out.println(sex);
        System.out.print("status: ");
        System.out.println(status);
        System.out.print("address: ");
        System.out.println(address);
        System.out.print("money: ");
        System.out.println(money);
        System.out.println();
    }

}
```

在上面程序中，需要注意，我们并非实现了 ContentHandler 接口，而是继承了已经实现该接口的 DefaultHandler。我再付类中已经实现了接口中所有方法，并将方法实现为空方法，我们继承父类只需要重写那些我们需要的方法即可。

Android 网络编程之 WebKit 应用

Android 中，提供了 WebKit 引擎用于对网页浏览和操作进行编程。Google 对 WebKit 进行了封装，提供了丰富的 Java 接口，其中最重要的便是 android.webkit.WebView 控件。

1. WebViewer 控件

Android 提 供了 WebView 控件专门 用来浏览网页。其使用方法和其他控件一样，需要在布局文件中进行布局，然后在程序中就可以使用并进行设置了。通过 loadUrl 方法，可以访问网页。代码如下：

```
1    wb=(WebView)findViewById(R.id.wb);
2    wb.loadUrl(url);
```

对于浏览器的设置，可以通过 WebSettings 来设置 WebView 的一些属性、状态等。代码如下：

```
    WebSettings webSettings = mWebView.getSettings();
          webSettings.setJavaScriptEnabled(true);
          //设置可以访问文件
          webSettings.setAllowFileAccess(true);
          //设置支持缩放
          webSettings.setBuiltInZoomControls(true);
```

2. WebViewClient 和 WebChromClient

WebViewClient 和 WebChromClientshi 可以看作是辅助 WebView 管理网页中各种通知、请求等事件以及 JavaScript 时间的两个类。

2.1 WebViewClient

通过 WebView 的 setWebViewClient 方法指定一个 WebViewClient 对象。通过覆盖该类的方法来 辅助 WebView 浏览网 页。代码如下：

```
    mWebView.setWebViewClient(new WebViewClient()
          {
              public boolean
shouldOverrideUrlLoading(WebView view, String url)
              {
                  view.loadUrl(url);
                  return true;
              }
              @Override
              public void onPageFinished(WebView view,
String url)
             {
                 super.onPageFinished(view, url);
             }
             @Override
             public void onPageStarted(WebView view, String
```

```
url, Bitmap favicon)
            {
                super.onPageStarted(view, url, favicon);
            }
        });
```

2.2 WebChromClient

对于网页中使用的 JavaScript 脚本语言，就可以使用该类处理 Js 事件，如对话框加载进度等。例如：

```
    mWebView.setWebChromeClient(new WebChromeClient(){
            @Override
            //处理javascript中的alert
            public boolean onJsAlert(WebView view, String
url, String message,
                    final JsResult result)
            {
                //构建一个Builder来显示网页中的对话框
                Builder builder = new
Builder(Activitythis);
                builder.setTitle("提示对话框");
                builder.setMessage(message);

builder.setPositiveButton(android.R.string.ok,
                        new AlertDialog.OnClickListener() {
                            public void
onClick(DialogInterface dialog, int which) {
                                //点击确定按钮之后,继续执行网页
中的操作
                                result.confirm();
                            }
                        });
                builder.setCancelable(false);
                builder.create();
                builder.show();
                return true;
            };
    }
```

Drawable 资源

Drawable 资源是对图像的一个抽象，你可以通过 getDrawable(int)得到并绘制到屏幕上。这里有几种不同类型的 Drawable：

Bitmap File

一个 Bitmap 图像文件（.png、.jpg 或.gif）。BitmapDrawable。

Nine-Patch File

一个带有伸缩区域的 PNG 文件，可以基于 content 伸缩图片（.9.png）。NinePatchDrawable。

State List

一个 XML 文件，为不同的状态引用不同的 Bitmap 图像（例如，当按钮按下时使用不同的图片）。StateListDrawable。

Color

定义在 XML 中的资源，指定一个矩形（圆角可以有）的颜色。PaintDrawable。

Shape

一个 XML 文件，定义了一个几何形状，包括颜色和渐变。ShapeDrawable。

AnimationDrawable 资源的说明在【Animation 资源】文章中。

Bitmap File

基本的 Bitmap 图像。Android 支持几种不同格式的 Bitmap 文件：.png（最佳）、.jpg（可接受）、.gif（不要）。

注意：Bitmap 文件可能会被 aapt 工具进行无损图像压缩优化。例如，一个真彩色的 PNG（不超过 256 色）可能会被转换成一个带有颜色板的 8 位 PNG。这样做能保证图片质量一样，但减少内存占用。因此，需要了解的是放在这个文件夹下的二进制图像在编译时可能会发生变更。如果你打算以位流方式读取图像来转化成 Bitmap 的话，可以把它们放到 res/raw 文件中，在这里，它们不会被优化。

File Location：

res/drawable/filename.png (.png, .jpg, 或.gif)

文件名会被当作资源 ID 使用。

Complied Resource Datatype：

指向 BitmapDrawable 的资源指针。

Resource Reference：

R.drawable.filename（Java）

@[package:]drawable/filename（XML）

Example：

在 res/drawable/myimage.png 位置保存了一张图片，在 Layout XML 中可以应用这个图片到一个 View 上：

```
<ImageView
      android:layout_height="wrap_content"
      android:layout_width="wrap_content"
      android:src="@drawable/myimage" />
```

下面的代码可以以 Drawable 方式得到图片：

```
Resources res = getResources();
   Drawable drawable =
res.getDrawable(R.drawable.myimage);
```

Nine-Patch File

NinePatch 是一种 PNG 图像，可以定义拉伸区域，当 View 的 content 超出图像边界的话，Android 会拉伸它。典型用法是把这个图像设置为 View 的背景，而这个 View 至少有一个尺寸设置为“wrap_content”，当这个 View 变大来容纳 content 时，Nine-Patch 图像也会拉伸来匹配 View 的大小。

File Location：

res/drawable/filename.9.png

文件名将被当作资源 ID 使用。

Complied Resource Datatype：

指向 NinePatchDrawable 的资源指针。

Resource Reference：

R.drawable.filename（Java）

@[package:]drawable/filename（XML）

Example：

在 res/drawable/myninepatch.9.png 位置保存了一张图片，在 Layout XML 中可以应用这个图片到一个 View 上：

```
<Button
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:background="@drawable/myninepatch" />
```

State List

StateListDrawable 是定义在 XML 中的 Drawable 对象，能根据状态来呈现不同的图像。例如，Button 存在多种不同的状态（pressed、focused 或 other），使用 StateListDrawable，你可以为 Button 的每个状态提供不同的按钮图像。

你可以在 XML 文件中描述状态列表。在
元素里的每个代表每个图像。每个使用不同的特性来描述使用的时机。

当每次状态改变时，StateList 都会从上到下遍历一次，第一个匹配当前状态的 item 将被使用——选择的过程不是基于“最佳匹配”，只是符合 state 的最低标准的第一个 item。

File Location

Res/drawable/filename.xml

文件名将被当作资源 ID 使用。

Complied Resource Datatype：

指向 StateListDrawable 的资源指针。

Resource Reference：

R.drawable.filename（Java）

@[package:]drawable/filename（XML）

Syntax：

```
<?xml version="1.0" encoding="utf-8"?>
<selector
xmlns:android="http://schemas.android.com/apk/res/android
"
    android:constantSize=["true" | "false"]
    android:dither=["true" | "false"]
    android:variablePadding=["true" | "false"] >
    <item

android:drawable="@[package:]drawable/drawable_resource"
        android:state_pressed=["true" | "false"]
        android:state_focused=["true" | "false"]
        android:state_selected=["true" | "false"]
        android:state_active=["true" | "false"]
        android:state_checkable=["true" | "false"]
        android:state_checked=["true" | "false"]
        android:state_enabled=["true" | "false"]
        android:state_window_focused=["true" | "false"] />
</selector>
```

Elements：

必须。必须是根元素。可以包含一个或多个元素。

Attributes：

xmlns:android

String。必须。定义 XML 的命名空间，必须是

“http://schemas.android.com/apk/res/android”。

android:constantSize

Boolean。“true”表示随着状态变化，Drawable 的大小保持不变（所有状态中最大的 size）；“false”表示大小会变化。默认是 false。

android:dither

Boolean。“true”表示当 Bitmap 和屏幕的不是相同的像素设定时支持 Bitmap 抖动（例如，ARGB 8888 的 Bitmap 和 RGB 565 的屏幕）；“false”表示不支持。默认是“true”。

android:variablePadding

Boolean。“true”表示 Drawable 的 Padding 可以变化；“false”表示 Padding 保持相同（所有状态的最大 Padding）。使能这一特征需要在状态变化时处理 Layout，一般都不支持。默认值是 false。

定义特定状态的 Drawable，通过它的特性指定。必须是
的子元素。

Attributes：

android:drawable

Drawable 资源。必须。指向一个 Drawable 资源。

android:state_pressed

Boolean。“true”表示按下状态使用（例如按钮按下）；“false”表示非按下状态使用。

android:state_focused

Boolean。“true”表示聚焦状态使用（例如使用滚动球/D-pad 聚焦 Button）；“false”表示非聚焦状态使用。

android:state_selected

Boolean。“true”表示选中状态使用（例如 Tab 打开）；“false”表示非选中状态使用。

android:state_checkable

Boolean。“true”表示可勾选状态时使用；“false”表示非可勾选状态使用。（只对能切换可勾选—非可勾选的构件有用。）

android:state_checked

Boolean。“true”表示勾选状态使用；“false”表示非勾选状态使用。

android:state_enabled

Boolean。“true”表示可用状态使用（能接收触摸/点击事件）；“false”表示不可用状态使用。

android:window_focused

Boolean。“true”表示应用程序窗口有焦点时使用（应用程序在前台）；“false”表示无焦点时使用（例如 Notification 栏拉下或对话框显示）。

注意：记住一点，StateList 中第一个匹配当前状态的 item 会被使用。因此，如果第一个 item 没有任何状态特性的话，那么它将每次都被使用，这也是为什么默认的值必须总是在最后（如下面的例子所示）。

Example：

XML 文件保存在 res/drawable/button.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<selector
xmlns:android="http://schemas.android.com/apk/res/android
">
    <item android:state_pressed="true"
          android:drawable="@drawable/button_pressed" />
<!-- pressed -->
    <item android:state_focused="true"
          android:drawable="@drawable/button_focused" />
<!-- focused -->
    <item android:drawable="@drawable/button_normal" />
<!-- default -->
</selector>
```

Layout XML 将这个 Drawable 应用到一个 View 上：

```
<ImageView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/button" />
```

Color

定义在 XML 中的 color，可以当作 Drawable 使用，来填充矩形区域（圆角可以有）。这种 Drawable 的行为很像是颜色填充。

注意：Color Drawable 是一种简单的资源，可以使用 name 特性来引用其值（不再是 XML 文件的名）。因此，你可以在一个 XML 文件中的元素下添加多个 Color Drawable。

File Location：

res/drawable/filename.xml

文件名随意。元素的 name 将会当作资源 ID 使用。

Complied Resource Datatype：

指向 PaintDrawable 资源的指针。

Resource Reference：

R.drawable.color_name（Java）

@[package:]drawable/color_name（XML）

Syntax：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <drawable name="color_name"
        >color</drawable>
</resources>
```

Elements：

必须。必须是根节点。

没有特性。

一个 color Drawable。其值可以是任何有效的十六进制颜色值或者 Color 资源。Color 值总是以“#”开头，后面紧跟 Alpha-Red-Green-Blue 信息，格式是：#RGB、#ARGB 或者#AARRGGBB。

Attributes：

name

String。必须。Color 的名字。这个名字将被当作资源 ID 使用。

Example：

XML 文件保存在 res/drawable/color.xml。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <drawable name="solid_red">#f00</drawable>
    <drawable name="solid_blue">#0000ff</drawable>
</resources>
```

Layout XML 将会把这个 Color Drawable 应用到一个 View 上：

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/solid_blue" />
```

代码中获取 Color Drawable 并应用到 View 上：

```
1Resources res =  getResources();
```

```
Drawable redDrawable =
res.getDrawable(R.drawable.solid_red);

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

Shape

定义在 XML 中的几何形状。

File Location：

res/drawable/filename.xml

文件名将被当作资源 ID 使用。

Complied Resource Datatype：

指向 ShapeDrawable 的资源指针。

Resource Reference：

R.drawable.filename（Java）

@[package:]drawable/filename（XML）

Syntax：

```
<?xml version="1.0" encoding="utf-8"?>
<shape
xmlns:android="http://schemas.android.com/apk/res/android
"
    android:shape=["rectangle" | "oval" | "line" | "ring"] >
    <gradient
        android:angle="integer"
        android:centerX="integer"
        android:centerY="integer"
        android:centerColor="integer"
        android:endColor="color"
        android:gradientRadius="integer"
        android:startColor="color"
        android:type=["linear" | "radial" | "sweep"]
        android:usesLevel=["true" | "false"] />
    <solid
        android:color="color" />
```

```
    <stroke
        android:width="integer"
        android:color="color"
        android:dashWidth="integer"
        android:dashGap="integer" />
    <padding
        android:left="integer"
        android:top="integer"
        android:right="integer"
        android:bottom="integer" />
    <corners
        android:radius="integer"
        android:topLeftRadius="integer"
        android:topRightRadius="integer"
        android:bottomLeftRadius="integer"
        android:bottomRightRadius="integer" />
</shape>
```

Elements：

必须。必须是根元素。

Attributes：

android:shape

Keyword。定义 Shape 的类型。有效的值包括：

Value	Desciption
″rectangle″	矩形。默认形状。
″oval″	椭圆。
″line″	水平直线。需要元素定义线的宽度。
″ring″	环形。

接下来的特性只能在 android:shape=”ring”时使用：

android:innerRadius

Dimension。内环的半径。

android:innerRadiusRatio

Float。以环的宽度比率来表示内环的半径。例如，如果 android:innerRadiusRatio=”5”，内环半径等于环的宽度除以 5。这个值可以被 android:innerRadius 覆盖。默认值是 9。

android:thickness

Dimension。环的厚度。

android:thicknessRatio

Float。以环的宽度比率来表示环的厚度。例如，如果 android:thicknessRatio="2"，厚度就等于环的宽度除以 2。这个值可以被 android:thickness 覆盖。默认值是 3。

android:useLevel

Boolean。“true”表示可以当作 LevelListDrawable 使用。一般都为“false”。

为 Shape 指定渐变色。

Attributes：

android:angle

Integer。渐变色的角度值。0 表示从左到右，90 表示从下到上。必须是 45 的倍数，默认是 0。

android:centerX

Float。渐变色中心的 X 相对位置（0-1.0）。当 android:type="linear"时无效。

android:centerY

Float。渐变色中心的 Y 相对位置（0-1.0）。当 android:type="linear"时无效。

android:centerColor

Color。可选的颜色，出现在 start 和 end 颜色之间。

android:endColor

Color。end 颜色。

android:gradientRadius

Float。渐变色的半径。当 android:type="radial"时有效。

android:startColor

Color。start 颜色。

android:type

Keyword。渐变色的样式。有效值为：

Value Description
"linear"线性渐变，默认值。
"radial"环形渐变。start 颜色是处于中间的颜色。
"sweep" sweep 渐变

android:useLevel

Boolean。“true”表示可以当作 LevelListDrawable 使用。

填充 shape 的单一色。

Attributes：

android:color

Color。这个颜色会应用到 shape 上。

shape 的线形。

Attributes：

android:width

Dimension。线的厚度。

android:color

Color。线的颜色。

android:dashGap

Dimension。间断线间的距离。仅在 android:dashWidth 设定时有效。

android:dashWidth

Dimension。间断线的大小。仅在 android:dashGap 设定时有效。

内部 View 元素的边距。

Attributes：

android:left

Dimension。左内边距。

android:top

Dimension。上内边距。

android:right

Dimension。右内边距。

android:bottom

Dimension。下内边距。

为 shape 创建圆角。当 shape 是一个矩形时有效。

Attributes：

android:radius

Dimension。圆角的半径。会被下面的特性覆盖。

android:topLeftRadius

Dimension。左上圆角半径。

android:topRightRadius

Dimension。右上圆角半径。

android:bottomLeftRadius

Dimension。左下圆角半径。

android:bottomRightRadius

Dimension。右下圆角半径。

Examples：

XML 文件保存在 res/drawable/gradient_box.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
xmlns:android="http://schemas.android.com/apk/res/android
"
    android:shape="rectangle">
    <gradient
        android:startColor="#FFFF0000"
        android:endColor="#80FF00FF"
```

```
        android:angle="45"/>
    <padding android:left="7dp"
        android:top="7dp"
        android:right="7dp"
        android:bottom="7dp" />
    <corners android:radius="8dp" />
</shape>
```

Layout XML 将被当作 ShapeDrawable 应用到一个 View 上：

```
<TextView
    android:background="@drawable/gradient_box"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content" />
```

代码中也可以获得 ShapeDrawable，然后应用到 View 上：

```
Resources res = getResources();
Drawable shape = res. getDrawable(R.drawable.gradient_box);

TextView tv = (TextView)findViewByID(R.id.textview);
tv.setBackground(shape);
```