



大话企业级 Android 开发 · 第四部分

本教程说明及版权声明

- 《大话企业级 Android 开发》是国士工作室为了方便中国 Android 开发者，推动 Android 企业级应用开发，特投入大量心血撰写的书籍，并在网络上免费发布，希望为移动互联网和智能手机时代贡献绵薄之力！所有相关文档版权均属国士工作室所有。
- 本教程是由国士工作室参考官方文档，综合市面相关书籍，经过充分的吸收消化，结合开发实践的一部原创作品，为了本教程及早与广大读者同仁见面、分享，特采用定稿一部分就发布一部分的连载方式发布。读者可以在本博客获取最新内容。
- 未经国士工作室授权，禁止将此文档及其衍生作品以标准（纸质）书籍形式发行。
- 本文档受有关法律的版权保护，对本文档内容的任何未经同意的复制和抄袭行为，将导致相应的法律责任。未经国士工作室同意，任何团体及个人不能用此教程牟利，违者必究。但是：在不收取其他人费用的前提下，您可以自由传播此文档，但必须保证版权信息、文档及其自带标示的完整性。
- 如果对该文档有任何疑问或者建议，请进入官方微博客
<http://www.cnblogs.com/guoshiandroid/>留言或者直接与国士工作室联系（后附联系方式），我们会慎重参考您的建议并根据需要对本文档进行修改，以造福更多开发者！
- 《大话企业级 Android 开发》的最新及完整内容会在国士工作室官方微博客定期更新，请访问国士工作室博客
<http://www.cnblogs.com/guoshiandroid/>获取更多更新内容。





关于国土工作室

我们(国土工作室)是一支专注于 Android 平台企业级应用开发的技术团队，对娱乐多媒体应用有着深刻的理解及研发能力，致力服务于企业用户。为音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络应用向移动互联网发展提供解决方案和技术支持，为企业提供 Android 培训服务等多种业务。

我们尤其擅长于提供从 Android 客户端到服务端的一站式解决方案和技术支持，服务端可以采用 Java EE，也可以采用轻量级流行的 LAMP 技术体系。目前，研发出了比 KU6、优酷更加强大和完善的 Android 视频网站娱乐多媒体客户端软件，并在持续升级中。

目前，我们正在务实而卓有成效的与音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络服务商合作，发展迅速，渴望有志之士的加入，和我们一起为成为世界最好的 Android 软件开发和咨询、培训公司而奋斗，为移动互联网和智能手机时代贡献力量！

联系我们

电话:15711060468

Email:guoshiandroid@gmail.com

博客: <http://www.cnblogs.com/guoshiandroid/>





1 Android 项目的目录结构

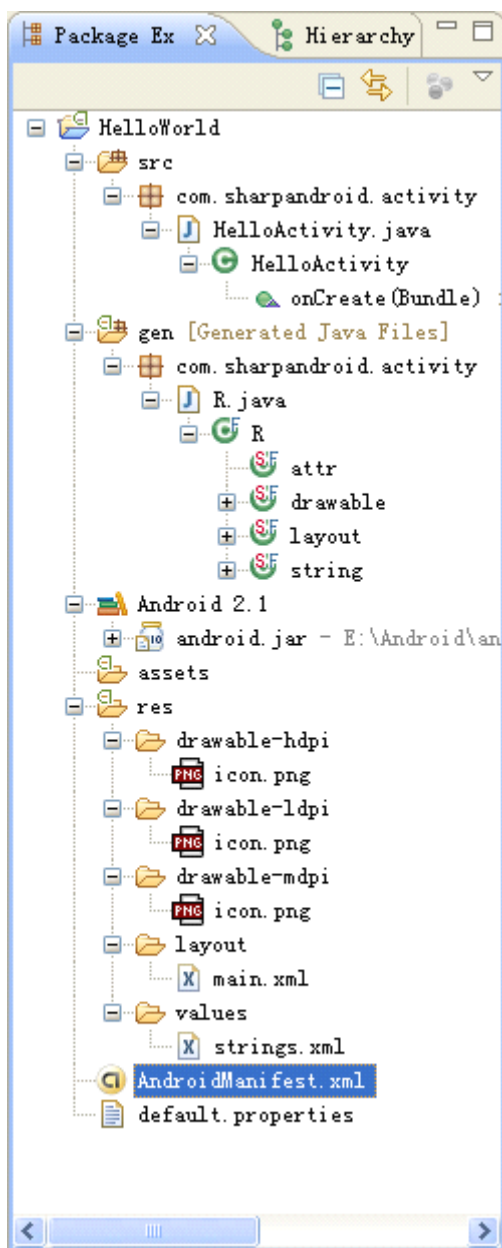
小安：博士，我昨天在网上看到别人开发的 Android 应用太炫了，要是我真的能做出一个这样的项目那该有多好啊？

大致：年轻人，不要急，只要你一步一步来，踏踏实实，跟着本博士好好学，很快就能开发出非常好的应用。为了你早日能够独立开发，更要了解和掌握 Android 的基础知识，刚才你也看到项目在 Eclipse 中的目录结构，接下来我给你介绍下这些目录以及里面文件的作用吧。入乡随俗，作为 Android 应用开发人员，我们非常有必要了解 Android 应用的目录组织结构，这样我们才能对更好的开发 Android 应用。下面主要结合 HelloWorld 项目进行介绍。

1.1 目录结构概述

下图是 HelloWorld 应用在 Eclipse 中的目录层次结构。





下面我们对每个目录及其文件进行介绍：
项目的根下有六个文件（夹）：

其中，资源是 Android 应用程序不可或缺的部分。资源是你想包含和引入到应用程序里面的一些外部元素，比如图片、音频、视频、文本字符串、布局、主题等。每个 Android





应用程序包含一个资源目录（res/）和资产目录（assets/），但资产不经常被使用，因为它们的应用很少。仅在需要读取原始字节流时才需要保存数据在 assets/ 目录。Res/ 和 assets/ 目录均在 Android 项目树的顶端，和源代码目录（src/）处在同一级上。资源和资产从表面上看没多大区别，不过总体上，在存储外部内容时资源用得更多。真正的区别在于任何放置在资源目录里的内容可以通过您的应用程序的 R 类访问，这是被 Android 编译过的。而任何存放在资产目录里的内容会保持它的原始文件格式，为了读取它，你必须使用 AssetManager 来以字节流的方式读取文件。所以保持文件和数据在资源中（res/）中会更方便访问。

1.2 Resource 目录及其下文件详解

res/ 目录下可以有以下几个子目录，部分目录（下表中加*的目录）开发工具并没有自动创建，根据我们需要可以自行创建，介绍如下表。

src/	专门存放我们编写的 java 源代码的包。
android 2.1/	存放 Android 自身的 jar 包。
gen/	该目录不用我们开发人员维护，但又非常重要的目录。该目录用来存放由 Android 开发工具所生成的目录。该目录下的所有文件都不是我们创建的，而是由 ADT 自动生成的。该目录下的 R.java 文件非常重要，后面会详细的介绍。
assets/	该目录用来存放应用中用到的类似于视频文件、MP3 一些媒体文件。
res/	res 是 resource 的缩写。我们称该目录为资源目录。该目录可以存放一些图标、界面文件、应用中用到的文字信息。
AndroidManifest.xml	该文件是功能清单文件，该文件列出了应用中所使用的所有组件，如“activity”，以及后面要学习的广播接收者、服务等组件。后面会详细介绍。
default.properties	该文件一般也不需要手工去更改。该文件存放了项目对应的一些环境配置，如应用要求运行的最低 Android 版本。

资源被编译到最终的 APK 文件里。Android 创建了一个被称为 R 的类，这样你在 Java 代码中可以通过它关联到对应的资源文件。R 类包含的子类的命名由资源在 res/ 目录下的文件夹名称所决定。

关于 res/ 子目录的一些补充说明：

1. 【res/drawable】

res/ 目录下有三个 drawable 文件夹——【drawable-*dpi】，区别只是将图标按分辨率高低来放入不同的目录，【drawable-hdpi】用来存放高分辨率的图标，【drawable-mdpi】用来存放中等分辨率的图标，【drawable-ldpi】用来存放低分辨率的图标。程序运行时可以根据手机分辨率的高低选取相应目录下的图标。不过，如果不想准备过多图片，那么也可以只准备一张图标将其放入三个目录的任何一个中去。

2. 【res/values/】文件夹下常放的文件如下：

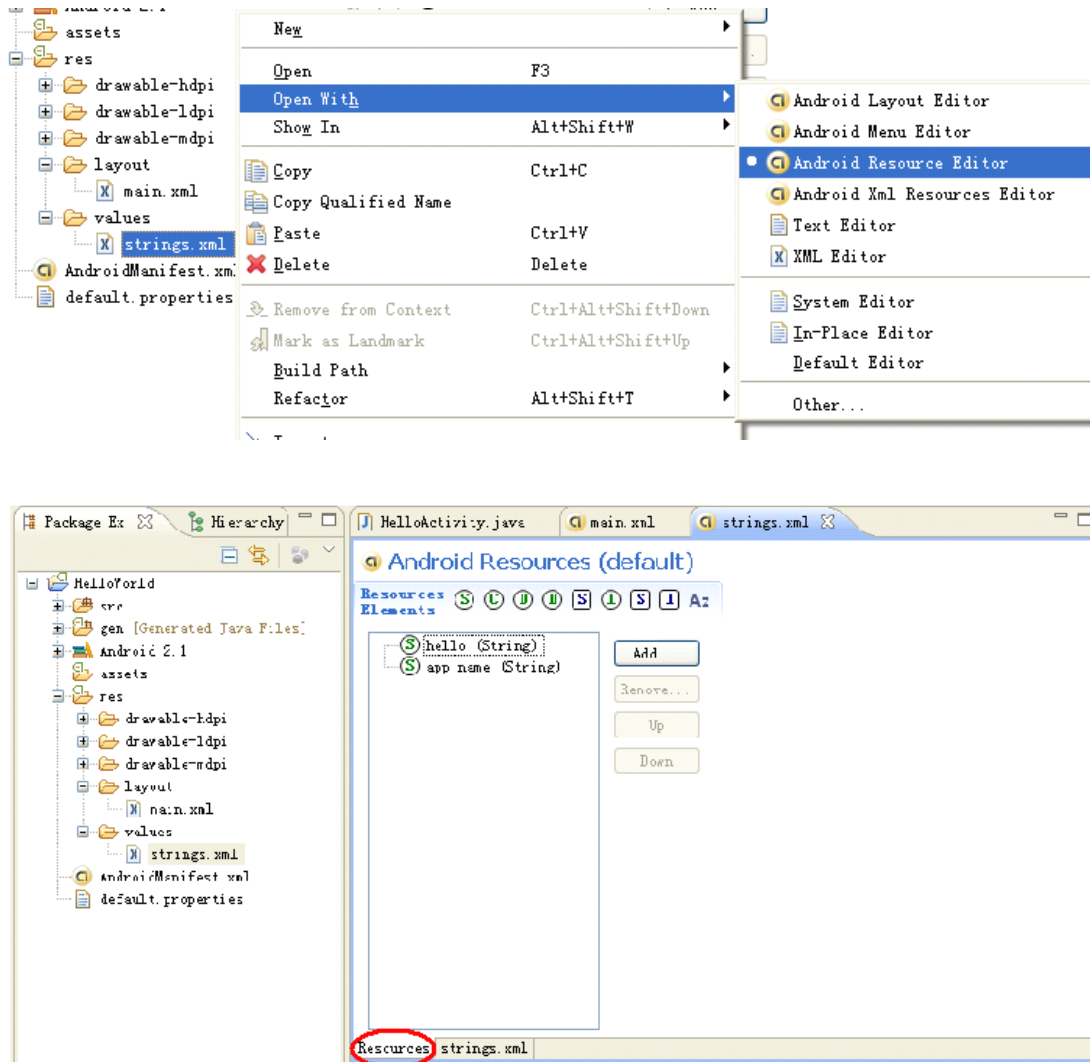


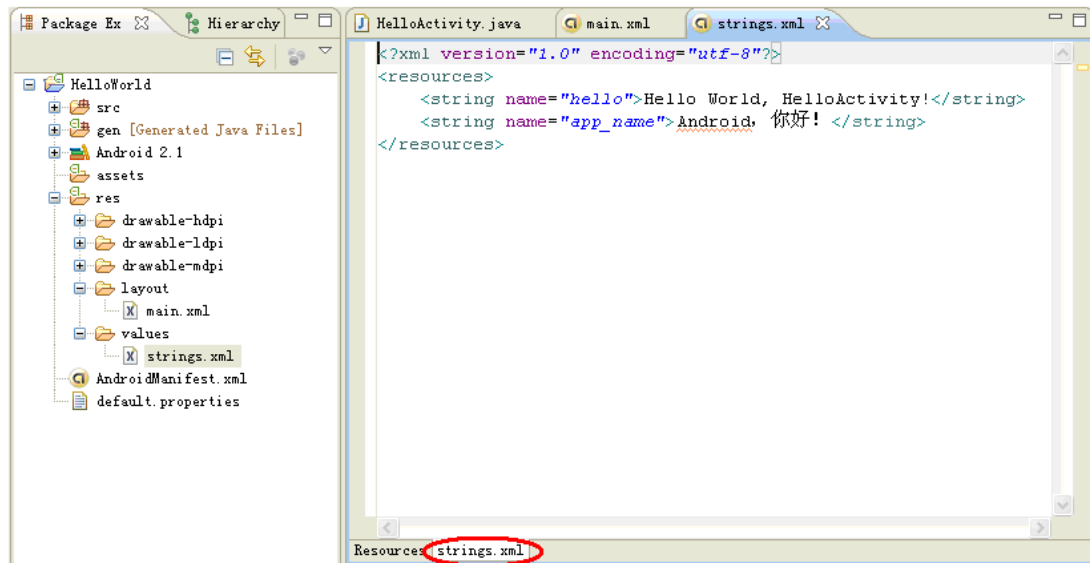


1) strings.xml

用来定义字符串和数值，在 Activity 中使用 `getResources().getString(resourceId)` 或 `getResources().getText(resourceId)` 取得资源。

在 Eclipse 中对 `res/values` 下的文件左键双击或右键单击后选择【Open With】→【Android Resource Editor】编辑器打开。默认是【Resources】显示，点击旁边的【strings.xml】，显示代码内容。如图。





HelloWorld 项目的 strings.xml 文件内容如下：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloActivity!</string>
    <string name="app_name">Android, 你好! </string>
</resources>
```

每个 string 标签声明了一个字符串，name 属性指定其引用名。

为什么需要把应用中出现的文字单独存放在 strings.xml 文中呢？

原因有二，一是为了国际化，Android 建议将在屏幕上显示的文字定义在 strings.xml 中，如果今后需要进行国际化，比如我们开发的应用本来是面向国内用户的，当然要在屏幕上使用中文，而如今我们要让应用走向世界，打入日本市场，当然需要在手机屏幕上显示日语，如果没有把文字信息定义在 strings.xml 中，就需要修改程序内容了。但当我们把所有屏幕上出现的文字信息都集中存放在 strings.xml 文件之后，只需要再提供一个 strings.xml 文件，把里面的汉字信息都修改为日语，再运行程序时，Android 操作系统会根据用户手机的语言环境和国家来自动选择相应的 strings.xml 文件，这时手机界面就会显示出日语。这样做国际化非常的方便。二是为了减少应用的体积，降低数据冗余。假设在应用中要使用“我们一直在努力”这段文字 10000 次，如果我们不将“我们一直在努力”定义在 strings.xml 文件中，而是在每次使用时直接写上这几个字，这样下来程序中将有 70000 个字，这 70000 个字占 136KB 的空间。而由于手机的资源有限，其 CPU 的处理能力及内存是非常有限的，136KB 对手机程序来说是个不小的空间，我们在做手机应用是一定要记住“能省内存就省内存”。而如果将这几个字定义在 strings.xml 中，在每次使用到的地方通过 Resources 类来引用该文字，只占用了 14B，因此对降低应用体积效果是非常有效地。当然我们可能在开发时可能并不会用到这么多的文字信息，但是“不以善小而不为，不以恶小而为之”，作为手机应用开发人员，我们一定要养成良好的编程习惯。





2) arrays.xml

用来定义数组, 在 Activity 中使用 `getResources().getStringArray(resourceId)` 获取一个 String 数组。

示例代码如下:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="sports">
        <item>足球</item>
        <item>篮球</item>
        <item>太极</item>
        <item>乒乓球</item>
    </string-array>
</resources>
```

3) colors.xml

用来定义颜色和颜色字符串数值, 你可以在 Activity 中使用 `getResources().getDrawable(resourceId)` 以及 `getResources().getColor(resourceId)` 取得这些资源。示例如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
    <color name="contents_text">#ff000000</color>
</resources>
```

4) dimens.xml

用来定义尺寸数据, 在 Activity 中使用 `getResources().getDimension(resourceId)` 取得这些资源

```
<?xml version="1.0" encoding="UTF-8"?>
<resources>
    <dimen name="height">80dip</dimen>
</resources>
```

5) styles.xml

定义样式。

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="sharpText">
        <item name="android:textSize">18px</item>
        <item name="android:textColor">#008</item>
    </style>
</resources>
```

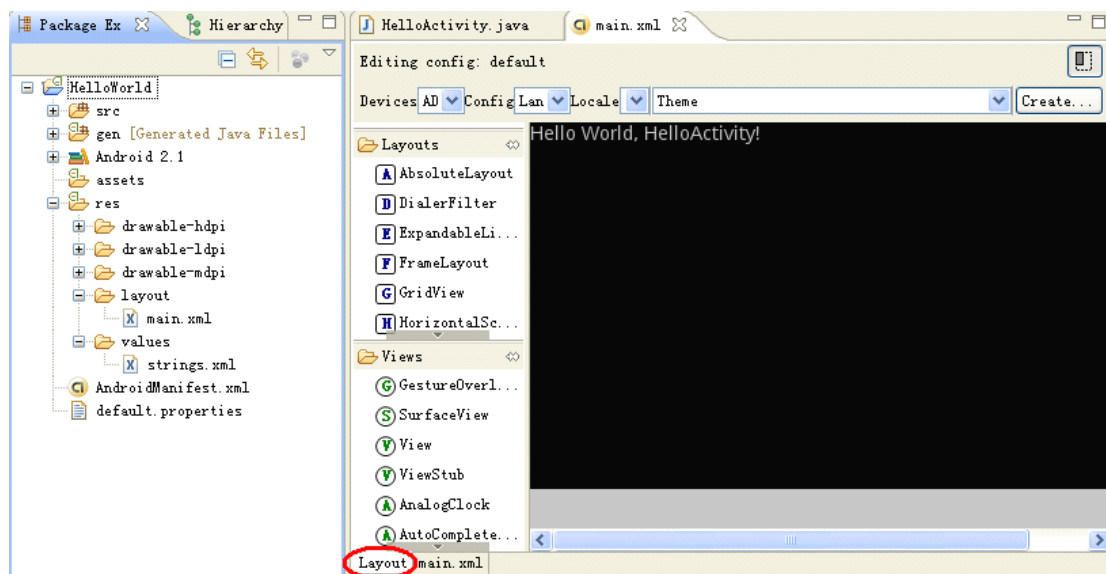
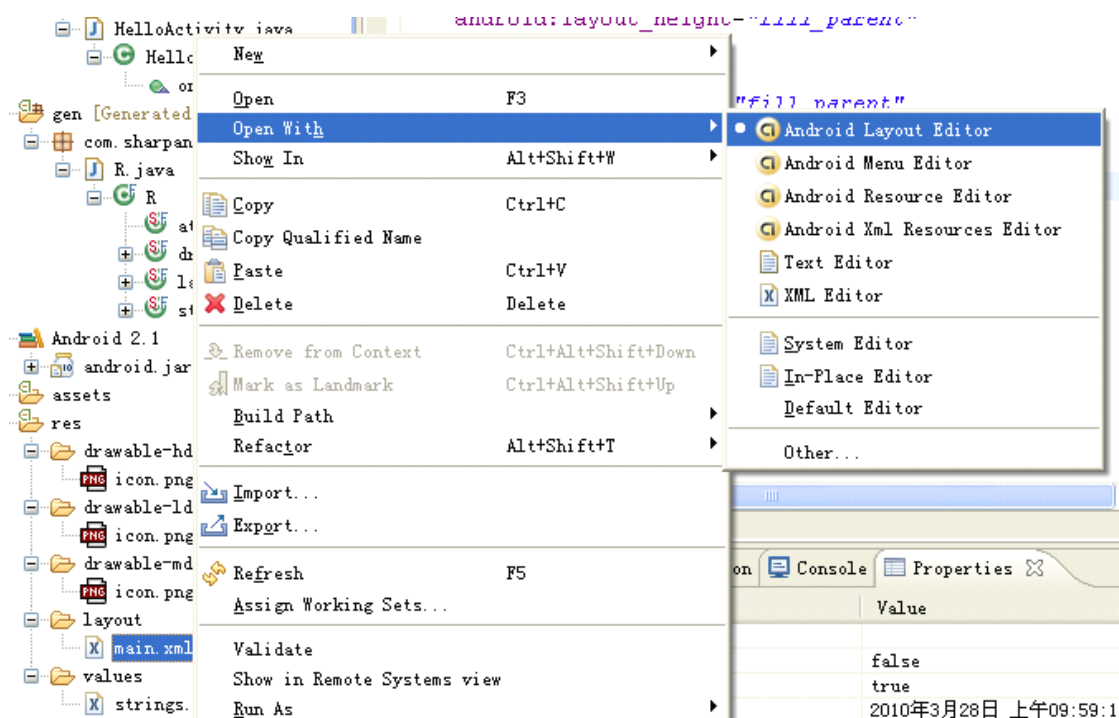
需要注意的是, Android 中的资源文件不要以数字作为文件名, 这样会导致错误。

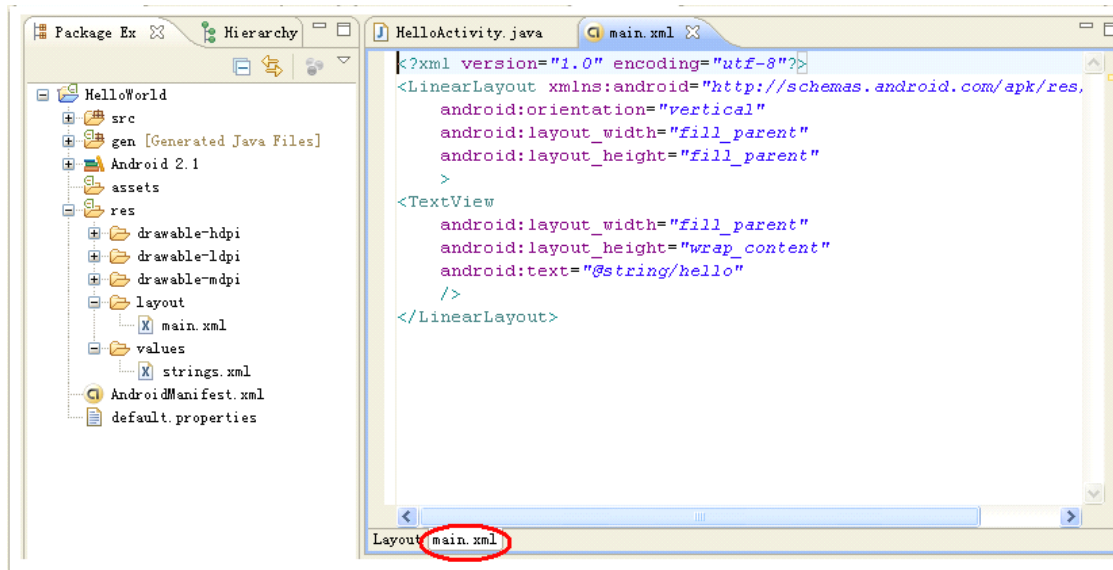




3. res/layout/目录下的布局文件简介

本例中的布局文件是 ADT 默认自动创建的“main.xml”文件。在 Eclipse 中，双击“main.xml”文件，或者右击以选择相应的编辑器，选用“Android Layout Editor”。在编辑区出现如下界面，默认显示的是“Layout 编辑器”的预览效果。可以点击 Layout 选项卡旁边的 main.xml，切换到代码模式以编辑。





与在网页中布局中使用 HTML 文件一样，Android 在 XML 文件中使用 XML 元素来设定屏幕布局。每个文件包含整个屏幕或部分屏幕，被编译进一个视图资源，可以被传递给 `Activity.setContentView` 或被其他布局文件引用。文件保存在工程的 `res/layout/` 目录下，它被 Android 资源编辑器编译。

下面介绍 HelloWorld 项目的 main.xml 文件的内容。

main.xml

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

我们逐元素进行分析说明：

- <LinearLayout>元素

“LinearLayout”翻译成中文是“线性布局”，所谓线性布局就是在该元素下的所有子元素会根据其“**orientation**”属性的值来决定是按行或者是按列逐个显示。后面章节有详细的介绍。

- <TextView>元素

该元素与 HTML 中的 <label> 元素比较相似。也是一种显示控件。

其属性 text 指定在该元素上面显示的文字内容。建议将该文字内容在 strings.xml 文件



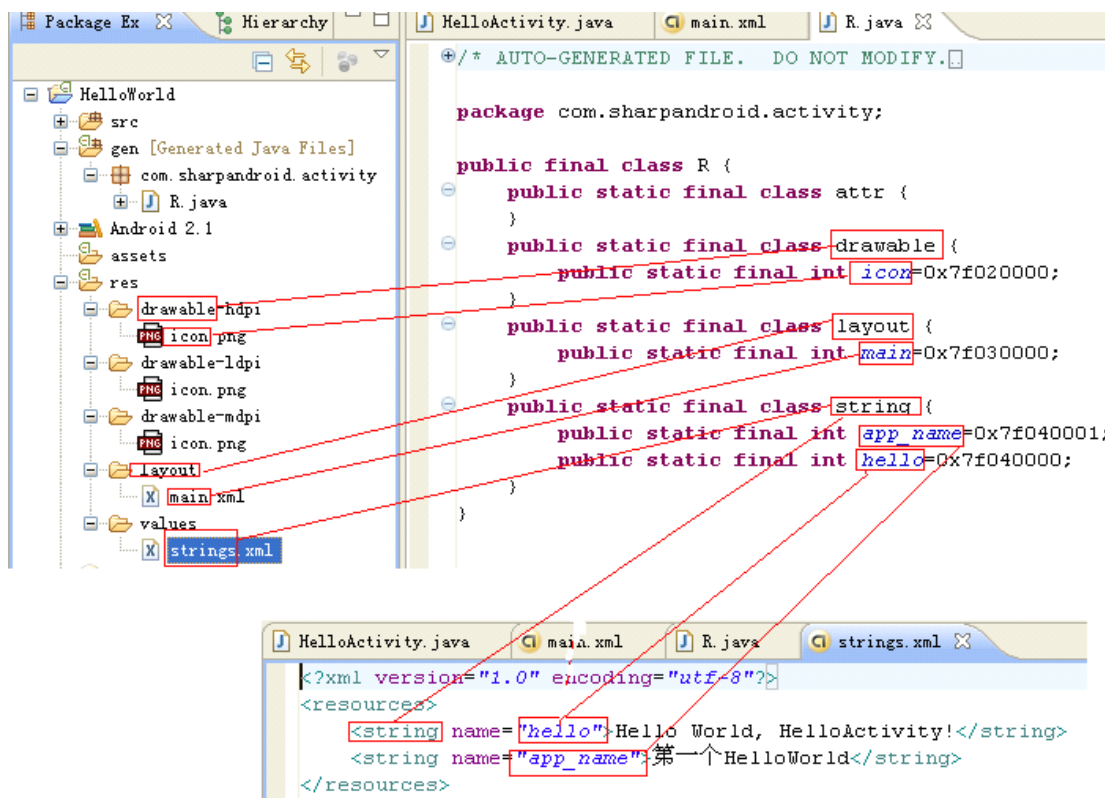


中进行定义，之后再在main.xml文件中通过“@string/stringName”的方式进行引用。

1.2.1 gen/目录下的 R.java 文件详解

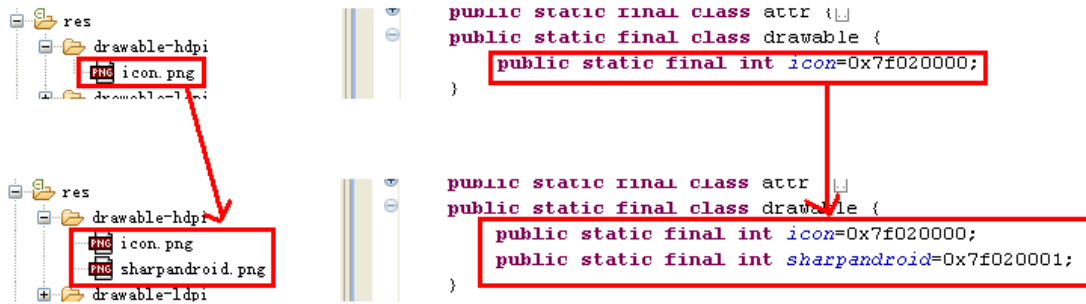
接下来我们再来详细学习 R.java 文件。

R.java 文件中默认有 attr、drawable、layout、string 等四个静态内部类，每个静态内部类分别对应一种资源，如 layout 静态内部类对应 layout 中的界面文件，其中每个静态内部类中的静态常量分别定义一条资源标识符，如“`public static final int main=0x7f030000;`”对应的是 layout 目录下的 main.xml 文件。具体的对应关系，如下图。



由于目前【drawable-*dpi】目录下都只有 icon.png 一个图片文件，因此此时不同像素的同名的 icon.png 文件在 drawable 内部类中只有一个 icon 属性。如果我们在【drawable-*dpi】目录下再添加一幅图片，那么会有什么效果出现呢？请看下图。





添加一幅图片前后变化的对比



sharppandroid.png

现在已经理解了 R.java 文件中内容的来源，也即是当开发者在 res/ 目录中任何一个子目录中添加相应类型的文件之后，ADT 会在 R.java 文件中相应的匿名内部类当中自动生成一条静态 int 类型的常量，对添加的文件进行索引。如果在 layout 目录下添加一个新的界面，那么在 public static final class layout 中也会添加相应的静态 int 常量。相反的，当我们再 res 目录下删除任何一个文件，其在 R.java 中对应的记录会被 ADT 自动删除。再比如说我们在【strings.xml】添加一条记录，在 R.java 的 string 内部类中也会自动增加一条记录。

R.java 文件会给我们开发程序带来很大的方便，比如在程序中我们使用“`public static final int icon=0x7f020000;`”就可以找到其对应的 icon.png 这幅图片。

R.java 文件除了有自动标识资源的“索引”功能之外，还有另一个主要的功能，当 res 目录中的某个资源在应用中没有被使用到，在该应用被编译的时候系统就不会把对应的资源编译到该应用的 APK 包中，这样可以节省 Android 手机的资源。

1.2.2 组件标识符

通过对 R.java 文件的介绍，我们已经了解了 R 文件的索引作用，它可以检索到我们应用中需要使用的资源。下面介绍如何通过 R.java 文件来引用到所需要的资源。

1. 在 Java 程序当中，我们可以按照 Java 的语法来引用。

1) R.resource_type.resource_name

需要注意的是，resource_name 不需要文件的后缀名。

比如说上面的 icon.png 文件的资源标识符可以通过如下方式获取。

R.drawable.icon

2) android.R.resource_type.resource_name

Android 系统本身自带了很多的资源，我们也可以进行引用，只是需要在前面加上





“android.” 以声明该资源来自 Android 系统。

2. 在 XML 文件中引用资源的语法如下：

1) @[package:]type/name

使用我们自己包下的资源可以省略 package。

在 xml 文件中，如 main.xml 以及 AndroidManifest.xml 文件中通过 “@drawable/icon” 的方式获取。其中 “@” 代表的是 R.java 类，“drawable” 代表 R.java 中的静态内部类 “drawable”，“/icon” 代表静态内部类 “drawable” 中的静态属性 “icon”。而该属性可以指向 res 目录下的 “drawable-*dpi” 中的 icon.png 图标。

其他类型的文件也比较类似。凡是在 R 文件中定义的资源都可以通过

“@Static_inner_classes_name/resource_name” 的方式获取。如 “@id/button”，“@string/app_name”。

2) 如果访问的是 Android 系统中带的文件，则要添上包名 “android:”。

如 android:textColor="@android:color/red”。

3. “@+id/string_name” 表达式

顺便说一下，在布局文件当中我们需要为一些组件添加 Id 属性作为标示，可以使用如下的表达式 “@+id/string_name” 其中 “+” 表示在 R.java 的名为 id 的内部类中添加一条记录。如 “@+id/button” 的含义是在 R.java 文件中的 id 这个静态内部类添加一条常量名为 button，该常量就是该资源的标识符。如果 id 这个静态内部类不存在，则会先生成它。通过该方式生成的资源标识符，仍然可以以 “@id/string_name” 的方式引用。示例代码片段如下。

```
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/cancel_button"
    android:layout_alignParentRight="true"
    android:id="@+id/cancel"/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_toLeftOf="@id/cancel"
    android:layout_alignTop="@id/cancel"
    android:text="@string/ok_button"/>

</RelativeLayout>
```

其中，android:id="@+id/cancel" 将其所在的 Button 标识为 cancel，在第二个 Button 中通过 "@id/cancel" 对第一个 Button 进行引用。





1.2.3 AndroidManifest.xml 介绍

如果要查看 AndroidManifest.xml 的详细信息，可以到帮助文档中去了解。路径如下：**【Dec Guide】→【The AndroidManifest.xml File】**超链接。

每个应用程序都有一个功能清单文件 AndroidManifest.xml（一定是这个名字）在它的根目录里。这个清单文件给 Android 系统提供了关于这个应用程序的基本信息，系统在运行任何程序代码之前必须知道这些信息。今后我们开发 Activity、Broadcast、Service 之后都要在 AndroidManifest.xml 中进行定义。另外如果我们使用到系统自带的服务如拨号服务、应用安装服务、GPRS 服务等都必须在 AndroidManifest.xml 中声明权限。

AndroidManifest.xml 主要包含以下功能：

- 命名应用程序的 Java 应用包，这个包名用来唯一标识应用程序；
- 描述应用程序的组件——活动、服务、广播接收者、内容提供者；对实现每个组件和公布其功能（比如，能处理哪些意图消息）的类进行命名。这些声明使得 Android 系统了解这些组件以及它们在什么条件下可以被启动；
- 决定应用程序组件运行在哪个进程里面；
- 声明应用程序所必须具备的权限，用以访问受保护的部分 API，以及和其它应用程序交互；
- 声明应用程序其他的必备权限，用以组件之间的交互；
- 列举测试设备 Instrumentation 类，用来提供应用程序运行时所需的环境配置及其他信息，这些声明只在程序开发和测试阶段存在，发布前将被删除；
- 声明应用程序所要求的 Android API 的最低版本级别；
- 列举 application 所需要链接的库；

程序中使用的所有组件都会在功能清单文件中列出来，所以我们先分析功能清单文件。只有下列元素才是功能清单文件中的合法元素，应用开发者不能添加自己的元素或属性。

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

```
<manifest>
    <uses-permission />
    <permission />
    <permission-tree />
    <permission-group />
    <instrumentation />
    <uses-sdk />
    <uses-configuration />
    <uses-feature />
    <supports-screens />
    <application>
        <activity>
            <intent-filter>
                <action />
                <category />
```





```
        <data />
    </intent-filter>
    <meta-data />
</activity>
<activity-alias>
    <intent-filter> . . . </intent-filter>
    <meta-data />
</activity-alias>
<service>
    <intent-filter> . . . </intent-filter>
    <meta-data/>
</service>
<receiver>
    <intent-filter> . . . </intent-filter>
    <meta-data />
</receiver>
<provider>
    <grant-uri-permission />
    <path-permission />
    <meta-data />
</provider>
<uses-library />
</application>
</manifest>
```

下面以 HelloWorld 项目的功能清单文件为例进行讲解。

AndroidManifest.xml

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





```
<uses-sdk android:minSdkVersion="7" />
</manifest>
```

以下详细介绍各个标签：

1. <manifest>元素

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.sharpandroid.activity"
    android:versionCode="1"
    android:versionName="1.0">
```

该元素是 AndroidManifest.xml 文件的根元素，该元素为必选。其中根据 xml 文件的语法，“xmlns:android”指定该文件的命名空间。功能清单文件会使用“<http://schemas.android.com/apk/res/android>”所指向的一个文件。“package”属性是指定 Android 应用所在的包，以后会经常说到“应用的包”，“应用的包”就是指该属性的内容。

“android:versionCode”指定应用的版本号。如果应用需要不断升级，在升级的时候应该修改该值。“android:versionName”是版本名称，名称的取定可根据爱好而定。

2. <application>元素

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

<application>是非常重要的一个元素，今后我们开发的许多组件都会在该元素下定义的。该元素为必选元素。

<application>的“icon”属性是用来设定应用的图标。

该表达式指向的是 icon.png 图片。在 Eclipse 中双击 icon.png 图片，如下图。对照关系如下图。



icon.png



`"@drawable/icon"`

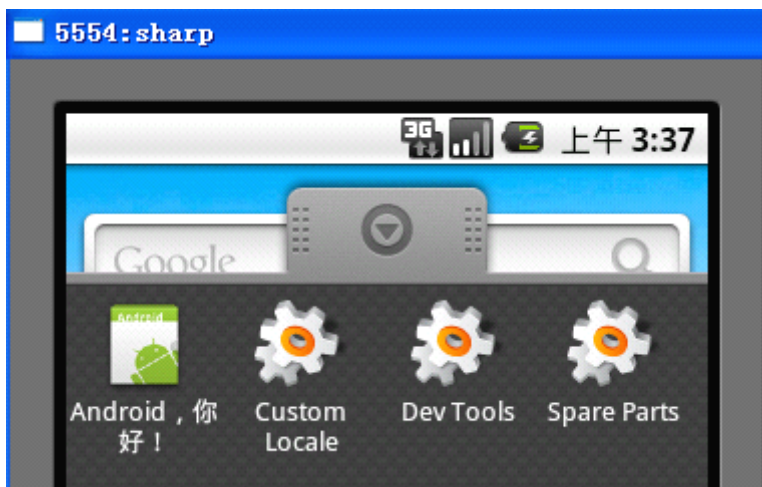
```
public final class R {  
    public static final class attr {  
    }  
    public static final class drawable {  
        public static final int icon=0x7f020000;  
        public static final int sharpandroid=0x7f020001;  
    }  
}
```

Diagram annotations:

- A red box highlights the `R` class name, with a red line pointing to the `@` symbol in the resource reference.
- A red box highlights the `drawable` class name, with a red line pointing to the `drawable` part of the resource reference.
- A red box highlights the `icon` variable name, with a red line pointing to the `/icon` part of the resource reference.

<application>的“label”属性用来设定应用的名称。指定其属性值所用的表达式“@string/app_name”含义与上面的表达式“@drawable/icon”一样，同样是指向 R.java 文件中的 string 静态内部类中的 app_name 属性所指向的资源。在这里它指向的是“strings.xml”文件中的一条记录“app_name”，其值为“Android，你好！”，因此，这种表达方式等价于 `android:label="Android，你好！"`。

两者结合起来，当程序发布到模拟器上之后，会在“抽屉”中显示效果，如下。



3. <activity>元素

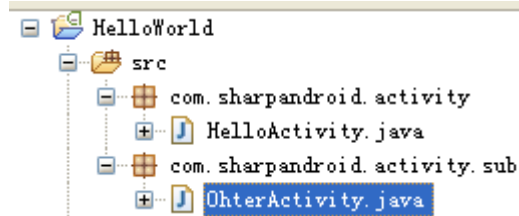
<activity>元素的作用是注册一个 Activity 信息，当我们在创建“HelloWorld”这个项目时，指定了【Create Activity】属性为“HelloActivity”，之后 ADT 在生成项目时帮我们自动创建了一个 Activity 名称就是“HelloActivity.java”，Activity 在 Android 中属于组件，它需要在功能清单文件中进行配置。

<activity>元素的“name”属性指定的是该 Activity 的类名。我们可以看到这个属性值“.HelloActivity”中的“.”，“.”代表的是在上面<manifest>元素的“package”属性中指定的当前包。因而“.HelloActivity”的含义等价于

“com.sharpandroid.activity.HelloActivity.java”。



如果 Activity 在应用的包中，则这个“.”可以去掉，但如果 Activity 不在应用的包中，而在应用的包的子包当中，如下图中的 OtherActivity 这个 Activity 配置时必须这么写“`.sub.OtherActivity`”，“.”不可省略。因此建议在书写时不论何种情况都加上“.”，以免出错。



Activity 只能放在“应用的包”或者其子包里面，而不能在“应用的包”以外的包中。这一点必须牢记。

<activity>元素的“`label`”属性表示 Activity 所代表的屏幕的标题，其属性值的表达式在上面已经介绍过了，不再赘述。

该属性值在 AVD 运行程序到该 Activity 所代表的界面时，会在标题上显示该值。如下图。



4. <intent-filter>元素

翻译成中文是“意图过滤器”。首先简单介绍什么是意图（Intent）。应用程序的核心组件（活动、服务和广播接收器）通过意图被激活，意图代表的是你要做的一件事情，代表你的目的，Android 寻找一个合适的组件来响应这个意图，如果需要会启动这个组件一个新的实例，并传递给这个意图对象。后面会有详细的介绍，在此只需有大致印象即可。

组件通过意图过滤器（intent filters）通告它们所具备的功能——能响应的意图类型。由于 Android 系统在启动一个组件前必须知道该组件能够处理哪些意图，那么意图过滤器需要在 manifest 中以<intent-filter>元素指定。一个组件可以拥有多个过滤器，每一个描述该组件所具有的不同能力。一个指定目标组件的显式意图将会激活那个指定的组件，意图过滤器不起作用。但是一个没有指定目标的隐式意图只在它能够通过组件过滤器任一过滤器时才能激活该组件。

第一个过滤器——

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

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

是最常见的。它表明这个 activity 将在应用程序加载器中显示，就是用户在设备上看到的可供加载的应用程序列表。换句话说，这个 activity 是应用程序的入口，是用户选择运行





这个应用程序后所见到的第一个 activity。

5. 权限 Permissions

HelloWorld 项目的功能清单文件中并没有出现<Permissions>元素，但是 Permission 也是一个非常重要的节点，我们在后面的学习中会经常的用到，在此进行介绍。Permission 是代码对设备上数据的访问限制，这个限制被引入来保护可能会被误用而曲解或破坏用户体验的关键数据和代码。如拨号服务、短信服务等。每个许可被一个唯一的标签所标识。这个标签常常指出了受限的动作。例如，下面是一些 Android 定义的权限：

permission name 值	作用
android.permission.CALL_PHONE	电话服务权限
android.permission.SEND_SMS	发送短信服务权限
android.permission.RECEIVE_SMS	接受短信服务权限
android.permission.READ_PHONE_STATE	监听电话权限
android.permission.MOUNT_UNMOUNT_FILESYSTEMS	在 SDCard 中创建与删除文件权限
android.permission.WRITE_EXTERNAL_STORAGE	往 SDCard 写入数据权限
android.permission.RECEIVE_BOOT_COMPLETED	开机启动, 电池电量变化, 时间改变等 Intent 权限
android.permission.RECORD_AUDIO	音频刻录权限
android.permission.CAMERA	照相机权限
android.permission.INSTALL_PACKAGES	安装程序权限

如申请发送短信服务的权限需要在功能清单文件中添加如下语句：

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

一个功能（feature）最多只能被一个权限许可保护。如果一个应用程序需要访问一个需要特定权限的功能，它必须在 manifest 元素内使用<uses-permission>元素来声明这一点。这样，当应用程序安装到设备上之后，安装器可以通过检查签署应用程序认证的机构来决定是否授予请求的权限，在某些情况下，会询问用户。如果权限已被授予，那应用程序就能够访问受保护的功能特性。如果没有，访问将失败，但不会给用户任何通知。因此我们在使用一些系统服务，如拨号、短信、访问互联网、访问 SDCard 时一定要记得添加相应的权限，否则会出现一些难以预料的错误。

应用程序还可以通过权限许可来保护它自己的组件（活动、服务、广播接收器、内容提供者）。它可以利用 Android 已经定义（列在 android.Manifest.permission 里）或其他应用程序已声明的权限许可，或者定义自己的许可。一个新的许可通过<permission>元素声明。比如，一个 Activity 可以用下面的方式保护：

```
<manifest . . . >
    <permission android:name="com.example.project.DEBIT_ACCT" . . . />
    . . .
    <application . . . >
        <activity android:name="com.example.project.FreneticActivity" . . . >
            android:permission="com.example.project.DEBIT_ACCT"
```



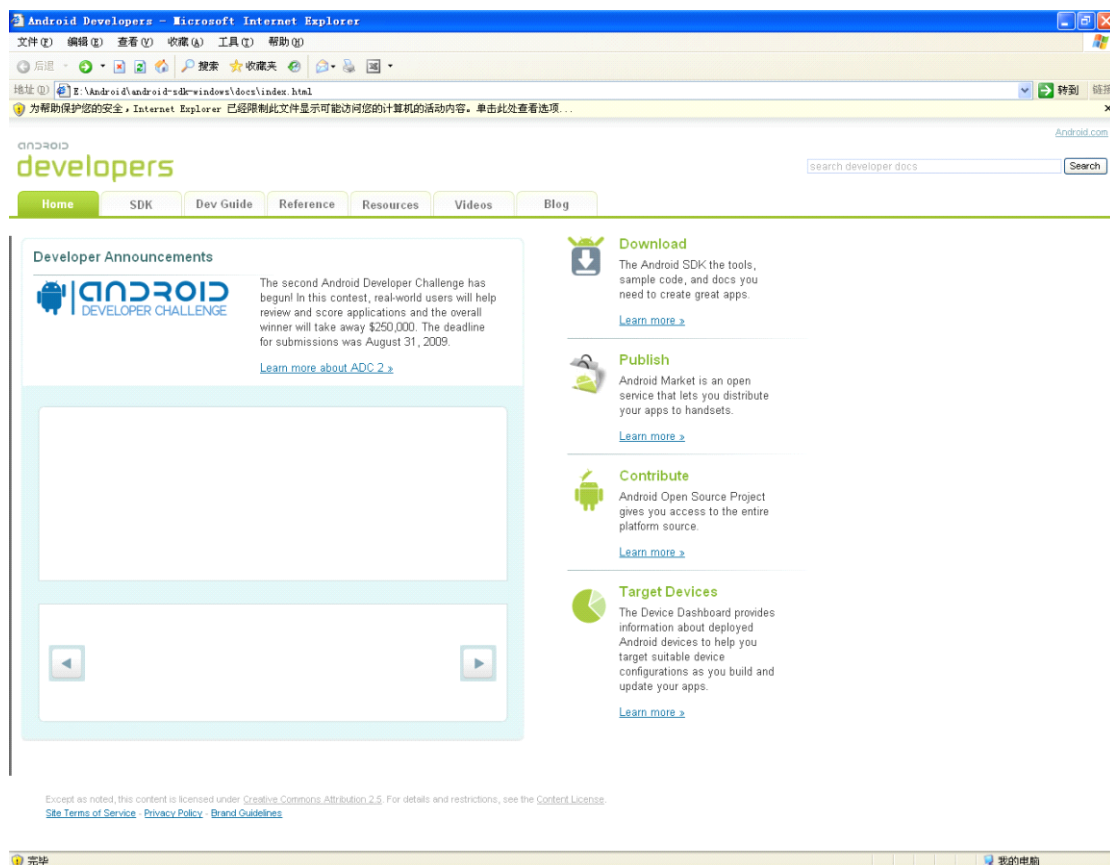


```
        . . . . >
        . . . .
    </activity>
</application>
. . . .
<uses-permission android:name="com.example.project.DEBIT_ACCT" />
. . . .
</manifest>
```

注意，在这个例子里，这个 DEBIT_ACCT 许可并非仅仅在<permission>元素中声明，如果该应用程序的其他组件要使用到该组件，那么它同样声明在<uses-permission>元素里。

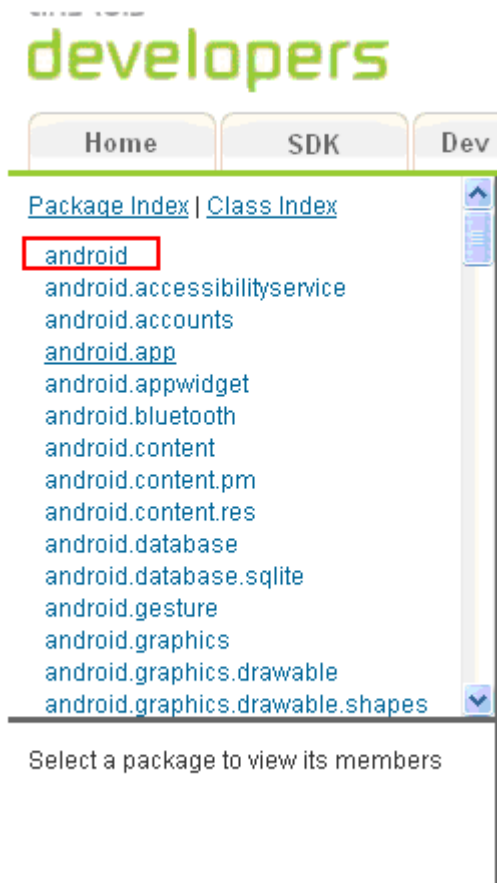
1.2.4 如何在文档中查找权限信息

在前面我们介绍了几种常用的权限，但是我们如何了解 Android 平台中的所有权限信息呢？如查找 CALL_PHONE 这个权限信息，可以按如下步骤操作。到 sdk 解压目录下找到 docs 文档的子目录，本机的目录是“E:\Android\android-sdk-windows\docs”，运行其中的 index.html 文件，页面如下图：



点击【Reference】，其左侧列表如下图：





点击方框中的 android 超链接，进入如下界面。





HomeSDKDev GuideReferenceResources

[Package Index](#) | [Class Index](#)

android

[android.accessibilityservice](#)
[android.accounts](#)
[android.app](#)
[android.appwidget](#)
[android.bluetooth](#)
[android.content](#)
[android.content.pm](#)
[android.content.res](#)
[android.database](#)
[android.database.sqlite](#)
[android.gesture](#)
[android.graphics](#)
[android.graphics.drawable](#)
[android.graphics.drawable.shapes](#)

Classes

[Manifest](#)
[Manifest.permission](#)
[Manifest.permission_group](#)

package

android

Classes | [Description](#)

Contains the resource classes used by
[more...](#)

Classes

Manifest	
Manifest.permission	
Manifest.permission_group	
R	
R.anim	

点击方框中的 Manifest.permission 超链接，进入 Manifest.permission 类页面。如下图。





Home

SDK

Dev Guide

Reference

Resources

Videos

Blog

Package Index | Class Index

android

android.accessibilityservice

android.accounts

android.app

android.appwidget

android.bluetooth

android.content

android.content.pm

android.content.res

android.database

android.database.sqlite

android.gesture

android.graphics

android.graphics.drawable

android.graphics.drawable.shapes

Classes

Manifest

Manifest.permission

Manifest.permission_group

R

R.anim

R.array

R.attr

R.bool

R.color

R.dimen

R.drawable

R.id

R.integer

R.layout

R.plurals

R.raw

R.string

R.style

R.styleable

R.xml

public static final class

Manifest.permission

extends [Object](#)

[java.lang.Object](#)

↳ android.Manifest.permission

Summary

Constants		
String	ACCESS_CHECKIN_PROPERTIES	Allows read/wri
String	ACCESS_COARSE_LOCATION	Allows an appli
String	ACCESS_FINE_LOCATION	Allows an appli
String	ACCESS_LOCATION_EXTRA_COMMANDS	Allows an appli
String	ACCESS_MOCK_LOCATION	Allows an appli
String	ACCESS_NETWORK_STATE	Allows applicati
String	ACCESS_SURFACE_FLINGER	Allows an appli
String	ACCESS_WIFI_STATE	Allows applicati
String	ACCOUNT_MANAGER	Allows applicati
String	AUTHENTICATE_ACCOUNTS	Allows an appli
String	BATTERY_STATS	Allows an appli
String	BIND_APPWIDGET	Allows an appli
String	BIND_INPUT_METHOD	Must be require
String	BLUETOOTH	Allows applicati
String	BLUETOOTH_ADMIN	Allows applicati

向下寻找，则会找到 CALL_PHONE 这个权限信息。如下图。

String	BROADCAST_PACKAGE_REMOVED	Allows an application to broadcast a notification that an application pac
String	BROADCAST_SMS	Allows an application to broadcast an SMS receipt notification
String	BROADCAST_STICKY	Allows an application to broadcast sticky intents.
String	BROADCAST_WAP_PUSH	Allows an application to broadcast a WAP PUSH receipt notification
String	CALL_PHONE	Allows an application to initiate a phone call without going through the I confirm the call being placed.
String	CALL_PRIVILEGED	Allows an application to call any phone number, including emergency r Dialer user interface for the user to confirm the call being placed.

点击 CALL_PHONE 超链接，进入下一个界面，如图。





```
public static final String CALL_PHONE
```

Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call being placed.

Constant Value: "android.permission.CALL_PHONE"

从中得知申请该权限的值为“android.permission.CALL_PHONE”。

以后如果需要申请其他任何权限，都可以通过该方式在 Manifest.permission 类中找到相关的信息。

6. 库 Libraries

每个应用程序都链接到缺省的 Android 库，这个库包含了基础应用程序开发包（实现了基础类如活动、服务、意图、视图、按钮、应用程序、内容提供者等等）。

然而，一些包处于它们自己的库中。如果你的应用程序使用了其他开发包中的代码，它必须显式的请求链接到它们。这个 manifest 必须包含一个单独的<uses-library>元素来命名每一个库。如在进行单元测试的时候需要引入其所需要的库。

代码片段如下：

```
<application android:icon="@drawable/icon"
    android:label="@string/app_name">
    <uses-library android:name="android.test.runner" />
</application>
```

1.3 Android 程序的执行流程分析

经过前面对 Android 项目目录结构的介绍，以及相关文件的讲解，我们对许多细节已经有所了解，只是 Android 程序是如何执行的呢？下面做一个总结。

发布程序到手机上之后，当双击“抽屉”里该应用的图标时，系统会将这个点击事件包装成一个 Intent，该 Intent 包含两个参数

```
{action : "android.intent.action.MAIN",
    category : "android.intent.category.LAUNCHER" },
```

这个意图被传递给 HelloWorld 这个应用之后在应用的功能清单文件中寻找与该意图匹配的意图过滤器，如果匹配成功，找到相匹配的意图过滤器所在的 Activity 元素，再根据<activity>元素的“name”属性来寻找其对应的 Activity 类。接着 Android 操作系统创建该 Activity 类的实例对象，对象创建完成之后，会执行到该类的 onCreate 方法，此 onCreate 方法是重写其父类 Activity 的 onCreate 方法而实现。onCreate 方法用来初始化 Activity 实例对象。如下是 HelloWorld.java 类中 onCreate 方法的代码。

@Override

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

其中 super.onCreate(savedInstanceState) 的作用是调用其父类 Activity 的 onCreate 方法来实现对界面的画图绘制工作。在实现自己定义的 Activity 子类的 onCreate 方法时一定要



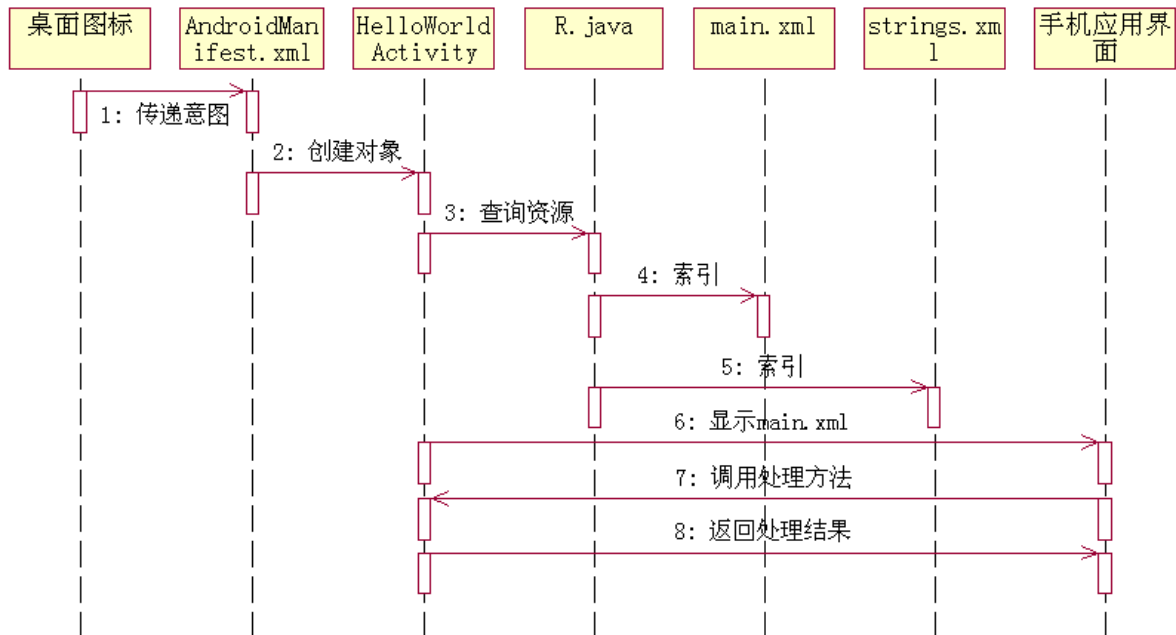


要记得调用该方法，以确保能够绘制界面。

`setContentView(R.layout.main)`的作用是加载一个界面。该方法中传入的参数是“`R.layout.main`”，其含义为`R.java`类中静态内部类`layout`的静态常量`main`的值，而该值是一个指向`res`目录下的`layout`子目录下`main.xml`文件的标识符。因此代表着显示`main.xml`所定义的画面。

关于`Activity`类的执行流程及其生命周期会在后面的部分详细讲解。

Android程序执行的整个序列图如下所示。




Android 应用执行序列图


1.4 修改 HelloWorld 项目的图标

通过上面的介绍，我们对 HelloWorld 项目的组织结构已经有所了解。我们可以对项目内容进行简单的修改，实现图标改变，以及显示内容的改变。

我们该如何修改应用的图标为我们前面加入工程的“`sharpandroid.png`”图标呢？方法如下：修改功能清单文件中`<application>`的“`icon`”属性为“`sharpandroid`”在`R.java`中的引用路径，按照上面的引用语法，写法为“`@drawable/sharpandroid`”，修改后的功能清单中的`<application>`元素内容如下。

```
<application android:icon="@drawable/sharpandroid"
    android:label="@string/app_name">
```

现在我们再次发布程序到 AVD 上，观察图标是否改变。运行结果如下。点击  返回主

界面，点击  拉开“抽屉”，看到“抽屉”中“Android，你好！”的图标变成如下样式。

如下图：



