res/values目录

arrays.xml

存放字符串数组
引用方式： @array/books

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="books">
        <item>疯狂Java讲义</item>
        <item>疯狂Ajax讲义</item>
        <item>疯狂XML讲义</item>
        <item>疯狂Android讲义</item>
    </string-array>
</resources>
```

strings.xml

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

    <string name="app_name">SimpleListViewTest</string>
    <string name="hello_world">Hello world!</string>
    <string name="action_settings">Settings</string>

</resources>
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="c1">#F00</color>
    <color name="c2">#0F0</color>
</resources>
```

dimens.xml

```
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>

    <dimen name="title_font_size">18sp</dimen>
</resources>
```

styles.xml、themes.xml、attrs.xml

android系统中，根据不同的主题，对于styles.xml和themes.xml，另外还存在styles_hole.xml，
styles_material.xml； themes_hole.xml， themes_material.xml，如果特定主题下对应的风格找不到，
再到styles.xml和themes.xml中找。

举例

attrs.xml

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

```
<resources>
    <declare-styleable name="Theme">
        ...
        <attr name="timePickerStyle" format="reference" />
        ...
    </declare-styleable>

    ...

    <declare-styleable name="TimePicker">
        ...
        <attr name="internalLayout" format="reference"/>

        <attr name="timePickerMode">
            <enum name="spinner" value="1" />
            <enum name="clock" value="2" />
        </attr>
        ...
    </declare-styleable>
</resources>
```

themes_material.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    ...
    <style name="Theme.Material.Light" parent="Theme.Light">
        ...
        <item name="timePickerStyle">@style/Widget.Material.Light.TimePicker</item>
        ...
    </style>
    ...
</resources>
```

style_material.xml

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

    <style name="Widget.Material.TimePicker">
        ...
        <item name="internalLayout">@layout/time_picker_material</item>
        ...
    </style>

    <style name="Widget.Material.Light.TimePicker"
                    parent="Widget.Material.TimePicker">
       ...
    </style>
</resources>
```

代码如下：

```
public TimePicker(Context context, AttributeSet attrs) {
    this(context, attrs, R.attr.timePickerStyle, 0);
}

public TimePicker(Context context, AttributeSet attrs,
        int defStyleAttr, int defStyleRes) {

    final TypedArray a = context.obtainStyledAttributes(
            attrs, R.styleable.TimePicker, defStyleAttr, defStyleRes);

    final int requestedMode = a.getInt(
                R.styleable.TimePicker_timePickerMode, MODE_SPINNER);
```

```
    final int layoutResourceId = a.getResourceId(
            R.styleable.TimePicker_internalLayout, R.layout.time_picker_material);

    a.recycle();
}
```

attrs.xml

attrs.xml用于定义一个范围内的属性名和属性类型，`<declare-styleable>`标签用于规定属性所在的范围
这个范围可以是一个特定的主题范围，也可以是一个特定的控件范围，
因为不同范围内容的属性可以同名，所以当我们需要获取该属性时，要指定范围去获取。如获取TimePicker下的internalLayout属性，可以这样获取：

```
final TypedArray a = context.obtainStyledAttributes(
            attrs, R.styleable.TimePicker, defStyleAttr, defStyleRes);

final int layoutResourceId = a.getResourceId(
            R.styleable.TimePicker_internalLayout, R.layout.time_picker_material);
```

TypeArray提供不同的方法用于获取不同类型的属性，常用的属性类型有：

```
color、dimension、integer、reference、string、枚举类型、boolean、float、
```

还可以是多个类型的组合，如：

```
color|reference， string|reference
```

另外`<attr>`标签也可以不设置format属性，此时attr标签表示的属性的属性类型在代码中解析再另外判断。

style.xml

style.xml用于定义某种风格，并在此风格下，给attrs.xml中声明的属性设置特定的值，
style中定义的风格，可以通过如下方式应用：

```
代码中 R.style.TimePicker  （如果风格名字中有"."，则替换成"_"）

xml文件中 @style/TimePicker
```

子标签item的name属性的属性值就是在attrs.xml中定义的属性，这里不需要指定属性是哪个范围的，因为在obtainStyledAttributes方法中会指定。

.. 两个item标签中的内容，就是属性的属性值

对于一个reference类型的属性，还可以通过@style引用另一个风格，

style标签的parent属性，可以为当前style指定一个父style，并且父style中设置的属性和属性值，会继承给
当前style。

```
<style name="MyPopupMenu" parent="@android:style/Widget.Material.ListPopupWindow">
    <item name="android:popupBackground">@drawable/popup_background_material</item>
    <item name="android:dropDownHorizontalOffset">50dp</item>
    <item name="android:dropDownVerticalOffset">0dp</item>
</style>
```

themes.xml

themes.xml跟style.xml的类似，只不过themes.xml一般用来设置主题的风格，对该主题下的多个控件进行配置。
而style.xml一般用来设置某一个控件的风格，只针对某一个控件进行配置。

res/anim 目录

放置View动画的xml文件，

引用方式 R.anim.slide_in_right 或 @anim/slide_in_right

举例如下：

slide_in_right.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 设置从右边拖进来的动画
    android:duration指定动画持续时间  -->
    <translate
        android:fromXDelta="100%p"
        android:toXDelta="0"
        android:duration="@android:integer/config_mediumAnimTime" />
</set>
```

slide_out_left.xml：

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android">
    <!-- 设置从左边拖出去的动画
    android:duration指定动画持续时间 -->
    <translate
        android:fromXDelta="0"
        android:toXDelta="-100%p"
        android:duration="@android:integer/config_mediumAnimTime" />
</set>
```

res/raw

存放一些资源文件，如.mp3音频文件(music.mp3)
引用方式 R.raw.music

res/menu

此目录用于存放创建菜单的文件

mymenu.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context="com.demo.zengk.MainActivity">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        app:showAsAction="never" />
</menu>
```

通过ManuInflater.inflate(R.menu.mymen, Menu)方法引用，将菜单文件中定义的菜单项加载到Menu对象中

获取系统资源文件

对于com.android.internal.R引用的drawable，style，string，color等资源，可通过如下方式获取：

```
Resources.getSystem().getIdentifier("sym_call_missed", "drawable", "android")
或者android.R.drawable.sym_call_missed
```

xml文件的命名空间

xmlns:app

在项目需求中，我们往往使用系统自带的属性以及控件是不够的，我们可能需要导入自定义控件的一些属性，或者support支持包之类的。
为了引入自定义的属性，我们可以`xmlns:前缀=http://schemas.android.com/apk/res/应用程序包路径`，
将其导入。
但现在的普遍做法是使用`xmlns:app="http://schemas.android.com/apk/res-auto"`，
因为res-auto可以引用所有的自定义包名，包括第三方Android库项目的包名下的自定义属性

xmlns:tools

xmlns:tools="http://schemas.android.com/tools"

tools命名空间可以告诉Android Studio，哪些属性在运行的时候是被忽略的，只在设计布局的时候有效。

tools可以覆盖android的所有标准属性，将android:换成tools:即可；同时在运行的时候就连tools:本身都是被忽略的，不会被带进apk中。

tools的预览效果

```
tools：text="xxxxx"
```

实际项目中，字符串应该是通过@string来引用的，但是开发过程中为了避免麻烦，经常会直接给text属性设置字符串来预览效果，如android：text="xxxxx"，这样做的话，打包成apk运行时，也会这样显示，但这不是我们想看到的，所以可以通过tools：text="xxxxx"来实现同样的效果，并且tools命名空间指定的text属性在运行时会被忽略

AndroidManifest.xml 配置文件

uses-sdk标签

通过uses-sdk标签可配置最小Android版本，目标Android版本

```
<uses-sdk android:minSdkVersion="21"
    android:targetSdkVersion="26"/>
```

Android 应用资源分类

Android 应用资源可以分为两大类：

```
无法通过R资源清单类访问的原生资源，保存在assets目录下（AndroidStudio中app/src/main/assets；
Eclipse中res/assets）

可通过R资源清单类访问的资源，保存在res目录下，AndroidStudio中app/src/main/res
```

对于res/目录下的应用资源，Android SDK在编译应用时会在R类中为它们创建对应的索引项。

res目录下应用资源的具体目录分类

Android要求在res/目下下用不同的子目录保存不同的应用资源：

```
res/animator/  存放定义属性动画的XML文件

res/anim/  存放定义补间动画的XML文件

res/color/  存放定义不同状态下颜色列表的XML文件

res/drawable/  存放各种位图文件(.png|.9.png|.jpg|.gif等)，
                此外，也可以存放编译成如下各种Drawable对象的XML文件：
                    BitmapDrawable对象
                    NinePatchDrawable对象
```

```
                    StateListDrawable对象
                    ShapeDrawable对象
                    AnimationDrawable对象
                    Drawable的其他各种子类对象

res/layout/  存放各种用户界面的布局文件

res/menu/  定义应用程序中各类菜单的资源，可以是选项菜单，上下文菜单，子菜单

res/raw/  存放任意类型的原生资源（如音频文件，视频文件等）。
            在Java代码中调用Resources.openRawResource(int id)方法来获取该资源文件的IO流。
            实际上，也可以将此目录下的原生资源文件保存到assets目录下，使用AssetManager访问。

res/values/  存放各种简单资源的XML文件，包括字符串资源、数组资源、颜色资源、尺寸资源、数值资源、
                样式资源、id资源、样式属性资源等。

            这些资源都可以放在根标签为<resources>的XML文件中，<resources>标签下添加不同的
            子标签则代表不同的资源。如<string>标签代表字符串、<integer>标签代表整数值、
            <bool>标签代表boolean值、<color>标签代表颜色值、<array>标签或<string-array>标签
            代表数组、<style>标签代表一个样式、<dimen>标签代表尺寸，等等。

            以上资源标签其实都可以放在res/values目录下的同一个根标签为<resources>的XML文件中，
            只不过这样做不好维护，所以Android建议使用不同的XML文件来存放不同类型的资源，如：
                arrays.xml：  存放<array>，<string-array>标签表示的数组资源

                colors.xml：  存放<color>标签表示的颜色资源

                dimens.xml：  存放<dimen>标签表示的尺寸值资源

                strings.xml：  存放<string>标签表示字符串资源

                styles.xml：  存放<style>标签表示的样式资源

                attrs.xml：  存放<declare-styleable>标签表示的样式属性资源

                常见的还有bools.xml，integers.xml，ids.xml，config.xml等资源文件

res/xml/  放置我们自定义的一些原始xml文件资源，如一些配置信息。
```

不同屏幕分辨率对应的 res/drawable/ 目录

Android除了提供res/drawable目录下，还为不同屏幕分辨率的手机提供了如下目录：

```
res/drawable-ldpi  低分辨率

res/drawable-mdpi  中等分辨率

res/drawable-hdpi  高分辨率

res/drawable-xhdpi 超高分辨率
```

通常我们只把会生成Drawable对象的XML文件放在res/drawable/。
而对于各种图片格式的位图文件，一般会为了适配屏幕分辨率制作出多个文件名相同的，但尺寸大小不同的图片文件，
分别放在对应分辨率的drawable目录下。这样Android系统就可以根据屏幕分辨率选择不同drawable目录下的图片。

数组资源

res/values/目录下的arrays.xml文件专门用来定义数组资源，指定数组资源的标签有

```
<array> 定义普通类型的数组，如Drawable数组，color数组
<string-array>  定义字符串数组
<integer-array> 定义整数数组
```

使用`<array>`标签定义普通数组资源举例：

```
<array name="preloaded_freeform_multi_window_drawables">
  <item>@drawable/decor_maximize_button_dark</item>
  <item>@drawable/decor_maximize_button_light</item>
</array>

<array name="sim_colors">
    <item>@color/Teal_700</item>
    <item>@color/Blue_700</item>
    <item>@color/Indigo_700</item>
    <item>@color/Purple_700</item>
    <item>@color/Pink_700</item>
    <item>@color/Red_700</item>
</array>
```

在Java程序中通过R.array.name来获取资源ID，Resources提供如下方法根据资源ID获取实际数组：

```
String[] getStringArray(int id)  获取字符串数组

int[] getIntArray(@ArrayRes int id)  获取整型数组

TypedArray obtainTypedArray(int id) 获取普通数组
```

调用完obtainTypedArray方法获取到普通数组之后，必须要调用TypedArray.recycle()方法回收资源

TypedArray表示一个通用类型的数组，提供getXxx(int index)方法来获取指定索引处的数组元素

res/xml目录下的原始资源

res目录下默认是没有xml文件夹的，需要手动创建。

引用res/xml下的原始xml文件的方式：

```
    @[<package_name>:]xml/file_name

    [<package_name>.]R.xml.<file_name>
```

Java代码中，可以通过Resources类提供的如下方法获取实际的xml文件数据：

```
XmlReousrceParser getXml(int id)  使用XmlPullParser解析器来解析xml文件

InputStream openRawResource(int id)  获取xml文件对应的输入流
```

Android系统内置了Pull解析器，所以推荐使用getXml方法来解析xml文件；

如果想使用其他解析器，如DOM、SAX、JAXP、dom4j、JDOM来解析xml文件，需要先调用openRawResource方法将xml文件转成IO流的形式传递给其他解析器解析。

XmlReousrceParser是XmlPullParser的子类

main/res/raw和main/assets目录下原始资源

类似于声音文件及其他类型的文件，只要Android没有为之提供专门的支持，这种资源都被称为原始资源。

Android中的原始资源可以放在如下两个地方

```
main/res/raw/   在此目录下原始资源，Android SDK会在R清单类中为其生成一个资源ID

main/assets/    此目录下的原始资源，Android不会为其生成ID索引，只能通过AssetManager来访问。
```

注意assets目录在AndroidStudio项目中放在main目录下，在Eclipse中是放在res目录下的

res/raw下原始资源的访问方式如下：

```
@[<package_name>:]raw/file_namme

[<package_name>.]R.raw.<file_name>

举例：

//作为MediaPlayer的音频资源
MediaPlayer.create(context, R.raw.bomb);

//转成流的形式
InputStream is = context.getResources().openRawResource(R.raw.bomb)

//转成AssetFileDescriptor对象
AssetFileDescriptor afd =  context.getResources().openRawResourceFd(R.raw.bomb)
```

main/assets/目录下的原始资源使用AssetManger提供的方法访问：

```
public final InputStream open(String fileName)

public final AssetFileDescriptor openFd(String fileName)
```

参数fileName直接写assets目录下的文件名即可，不需要指出路径。

国际化支持（Internationalization，I18N）

国际化就是指同一个版本app能够自适应不同地区的市场。

一个国际化支持很好的app，会随着不同区域使用而呈现出本地语言的提示。
这个过程也被称为本地化（Localization， L10N）

Java的国际化

将Java中使用到的字符串保存在一个资源文件中，资源文件的内容就是key-value字符串，文件命名形式如下：

```
baseName_language_country.properties

baseName_language.properties

baseName.properties
```

baseName是自定义的文件名；language和country是Java支持的语言和国家。

Java支持的语言和国家可以通过java.util.Locale类提供的
`public static Locale[] getAvailableLocales()`方法获取，返回的Locale数组中包含了Java所支持的
国家和语言：

```
public final String getDisplayLanguage()
public String getLanguage()

public final String getDisplayCountry()
public String getCountry()
```

如果*.properties资源文件中包含了非西欧字符（如中文），需要将其转成Unicode编码格式的字符才能被Java
正确引用。通过%JAVA_HOME%/bin目录下的native2ascii工具进行转换：

```
native2ascii 原资源文件  目标资源文件
```

执行上述指令后，再将目标资源文件重命名成原资源文件，并替换掉原资源文件即可。

Java代码中实现国际化需要用到如下几个类：

```
java.util.ResourceBundle  用于加载一个国家、语言相关的资源包

java.util.Locale  封装一个特定的国家/区域、语言环境

java.util.MessageFormat  格式化带占位符的字符串

举例

    Locale myLocale = Locale.getDefault(Locale.Category.FORMAT);

    ResourceBundle bundle = ResourceBundle.getBundle("baseName", myLocale);

    String str = bundle.getString("key"); //通过baseName*.properties中的key获取对应value
```

Android中的国际化支持

因为在规范的项目中，Android所引用的字符串都应该在res/values/目录下的资源文件中的配置。
所以Android通过为不同国家、语言提供不同的values定制目录来支持国际化：

```
values-语言代码-r国家代码
```

此时，简体中文环境下的字符串放在res/value-zh-rCN/，美式英语环境下的字符串放在res/value-en-rUS/。
如果不存在特定环境下对应的values目录或者对应的values目录中没有找到匹配的字符串，那么Android系统还会在默认的res/values/目录下查找

另外如果应用程序中引用的图片也要随国家、语言环境的改变而改变，那么还需要对drawable目录做定制：

```
drawable-语言代码-r国家代码
```

此时，如果drawable目录还需要支持不同的屏幕分辨率，那么drawable目录的定制如下：

```
drawable-zh-rCN-mdpi、drawable-zh-rCN-hdpi、...
```

测试时，将手机设置项中的Language&input下的language选项切换成不同的语言环境即可看到效果。