Andorid应用资源

1.1应用资源概述

Android应用资源可分为两大类。

- 无法通过R资源清单类访问的原生资源，保存在asset目录下。（程序需要通过AssetManager以二进制流的形式来读取文件）
- 可通过R资源清单类访问资源，保存在/res/目录下。

1.1.1资源类型以及存储方式

目录	存放的资源
/res/animator/	存放定义的属性动画xml文件
/res/anim/	存放定义的补间动画的xml文件
/res/drawable/	存放适应不同分辨率的各种位图文件（如*png*、.9.png、*.jpg*、.gif等），此外也可以编译如下各种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；

目录	存放的资源
/res/mipmap/	主要存放适应不同分辨率的应用程序图标，以及其他系统保留的Drawable资源
/res/layout/	存放各种用户界面的布局文件
/res/menu/	存放为应用程序定义各种菜单的资源，包括选项菜单、子菜单、上下文菜单；
/res/raw/	存放任意类型的原生资源（比如音频文件、视频文件等）。在java代码中可通过调用Resources对象的openRawResource（int id）方法来获取该资源的二进制输入流。实际上，如果应用程序需要使用原生资源，也可以把这些原生资源保存到/assets/目录下，然后在应用程序中使用AssetManager来访问这些资源。
/res/values/	存放各种简单值得xml文件。这些简单 值包括字符串值、整数值、颜色值、数组等。 这些资源文件的根元素都是<resources.../>,为该<resource.../>元素添加不同的子元素则代表不同的资源，例如： a.string/integer/bool子元素：代表添加一个字符串值、整数值、或boolean值 b.color子元素：代表添加一个颜色值； c. array子元素或string-array、int-array子元素：代表参加一个数组 d.style子元素：代表一个样式； e.dimen:代表一个尺寸； ... 为了维护方便通常有一下的xml array.xml colors.xml dimens.xml strings.xml style.xml
/res/xml/	存放任意原生XML文件。这些XML文件可以在java代码中用Resource.getXML()方法进行访问

```
<ListView
...
android:entries="@array/books"       <!--引用数组资源-->
/>
```

...

1.2 使用Drawable资源

1.2.1 StateListDrawable资源

介绍

当使用StateListDrawable作为目标组件的背景、前景图片时，StateListDrawable对象所显示的Drawable对象会随着目标组件状态的改变而自动切换。

定义StateListDrawable对象的XML文件的根元素为<selector.../>，该元素可以包含多个<item.../>元素，该元素可指定如下属性。

- android:color或android：drawable：指定颜色或Drawable对象。
- android：state_xxx:指定一个特定状态。

例如：

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:tools="http://schemas.android.com/tools"
    xmlns:android="http://schemas.android.com/apk/res/android"
    tools:ignore="MissingDefaultResource">
    <!--指定特定状态下的颜色-->
    <item android:color="hex_color"
        android:state_pressed="true"/>
</selector>
```

以下是StateListDrawable的<item.../>元素所支持的状态：

属性值	含义
android:state_activte	代表是否处于激活状态
android:state_checkable	代表是否处于可勾选状态
android:state_checked	代表是否处于已勾选状态
android:state_enabled	代表是否处于可用状态
android:state_first	代表是否处于开始状态
android:state_focused	代表是否处于已得到焦点状态
android:state_last	代表是否处于结束状态
android:state_middle	代表是否处于中间状态
android:state_pressed	代表是否处于已被按下状态
android:state_selected	代表是否处于已被选中状态
android:state_window_fcused	代表窗口是否处于已得到焦点状态

1.2 LayerDrawable资源

layerDrawable有如下属性：

- android:drawable:指定作为LayerDrawable元素之一的Drawable对象。
- android：id：为该Drawable对象指定一个标识。
- android：button|top|left|button:他们用于指定一个长度值，用于指定将该Drawable对象绘制到目标组件的指定位置。

1.3 ShapeDrawable资源

ShapeDrawable用于定义一个基本的几何图形（如矩形、圆形、线条等）

- android：shape["rectangle"|"oval"|"line"|"ring"]:指定定义哪种类型的几何图形。

定义ShapeDrawable对象的完整语法格式如下：

```
<shape>
    <!--定义几何图形的四个角的弧度-->
```

```
    <corners
        android:radius="integer"
        android:topLeftRadius="integer"
        android:topRightRadius="integer"
        android:bottomLeftRadius="integer"
        android:bottomRightRadius="integer"
    />
    <!--定义渐变色填充-->
    <gradient
        android:angle="integer"
        android:centerX="integer"
        android:centerY="integer"
        android:centerColor="integer"
        android:endColor="color"
        grandientRadius="integer"
        android:startColor="color"
        android:type=["linear" | "radial" | "sweep"]
        android:usesLevel=["true" | "false" |] />
    <!--定义几何形状的内边距-->
    <padding
        android:left="integer"
        android:top="integer"
        android:right="integer"
        android:bottom="integer"
        />
    <!--定义几何形状的大小-->
    <size
        android:width="integer"
        android:color="color"
        android:dashWidth="integer"
        android:dashGap="integer"/>
    <!--定义使用单种颜色填充-->
    <solid
        android:color="color"/>
    <!--定义为几何形状绘制边框-->
    <stroke
        android:width="integer"
        android:color="color"
        android:dashWidth="integer"
        android:dashGap="integer"/>
</shape>
```

1.3 ClipDrawable资源

ClipDrawable代表从其他位图上截取一个“图片片段”。在xml文件中定义ClipDrawable对象使用<clip.../>元素，该元素的语法为：

```
<clip>
    android:drawable="@drawable/drawable_resource"
    android:clipOrientation=["horizontal"|"vertical"]
    android:gravity=
["top"|"bottom"|"left"|"right"|"center_vertical"|"fill_vertical"|"center_horizon
tal"|"fill_horizontal"|"center"|"fill"|"clip_vertical"|"clip_horizontal"]
</clip>
```

上面的语法格式可以指定如下三个属性。

- android：drawable：指定截取的源Drawable对象。
- android：clipOrientation：指定截取方向，可设置水平截取或垂直截取。
- android：指定截取时的对齐方式。

使用ClipDrawable对象时可调用setLevel（int level）方法来设置截取的区域大小，当level为0时，截取的图片片段为空；当level为10000时，截取整张图片。

1.4 AnimationDrawable资源

AnimationDrawable代表一个动画，android既支持传统的逐帧动画（类似于电影方式，一张图片、一张图片地切换）,也支持通过平移、变换计算出来的补间动画。

以下以补间动画为例来介绍如何定义AnimationDrawable资源。定义补间动画的xml资源文件以<set.../>元素为根元素，该元素可以定义如下4个元素；

- alpha：设置透明度的改变；
- scale：设置图片进行缩放变换；
- translate：设置图片进行位移变换；
- rotate：设置图片进行旋转；

xml存放在/res/animi/路径下，需开发者自己创建；

补间动画定义思路很简单：设置一张图片的开始状态（包括透明度、位置、缩放比、旋转度），并设置该图片的结束状态（包括透明度、位置、缩放比、旋转度），再设置动画的持续时间，Andorid系统会使用动画效果把这张图片从开始状态变换到结束状态。

设置补间动画的语法格式如下：

```
<set
    android:interpolator="@[package:]anim/interpolator_resource"
    android:shareInterpolator=["true"|"false"]
    android:duration="持续时间"
    >
    <alpha
        android:formAlpha="float"
        android:toAlpha="float"
        />
    <scale
        android:fromXScale="float"
        android:toXScale="float"
        android:fromYScale="float"
        android:toYScale="float"
        android:pivotX="float"
        android:pivotY="float"/>
    <translate
        android:fromXDelta="float"
        android:toDelta="float"
        android:fromYDelta="float"
        android:toYDelta="float"
        />
    <rotate
        android:fromDegrees="float"
        android:toDegrees="float"
        android:pivotX="float"
        android:pivotY="float"
    </set>
```

pivotX,pivotY用于（scale和roate）指定变换的“中心点”——比如进行旋转变换时，需要指定“旋轴点”；进行缩放变换时，需要指定“中心点”。

此外上面的、、、、都可以指定一个android：interpolator属性，该属性指定动画的变化速度，可以实现匀速、正加速、负加速、无规则变加速等，Android系统的R.anim类中包含了大量常量，他们定义了不同动画速度，如：

- linear_interpolator:匀速变换；
- accelerate_interpolator：加速变换；
- decelerate_interpolator：减速变换；

  如果想让元素下所有的变换效果使用相同的动画速度，则可以指定 `android:shareInterpolator="true"`

1.5 属性动画资源

Animator代表一个属性动画，但它只是一个抽象类，通常会使用它的子类：AnimatorSet、ValueAnimator、ObjectAnimator、TimeAnimator。

定义属性动画的xml资源文件能以如下三个元素中的任意一个作为根元素。

- <set.../>:他是一个父元素，用于包含<objectAnimator.../>、<animator.../>或<set.../>子元素，该元素定义的资源代表AnimatorSet对象。
- <objectAnimator.../>:用于定义ObjectAnimator动画。
- <animator.../>:用于定义ValueAnimator动画。

定义属性动画的语法格式如下：

```
<set android:ordering=["together"|"sequentially"]>
    <objectAnimator
        android:propertyName="string"
        android:duration="int"
        android:valueFrom="float|int|color"
        android:valueTo="float|int|color"
        android:startOffset="int"
        android:repeatCount="int"
        android:interpolator=""
        android:repeatMode=["repeat"|"reverse"]
        android:valueType=["intType"|"floatType"]/>
    <animator
        android:duration="int"
        android:valueFrom="float|int|color"
        android:valueTo="float|int|color"
        android:startOffset="int"
        android:repeatCount="int"
        android:interpolator=""
        android:repeatMode=["repeat"|"reverse"]
        android:valueType=["intType"|"floatType"]/>

    <set>
    ...
    </set>
</set>
```

1.6使用原始xml资源

在某些时候，Anroid应用有一些初始化的配置信息、应用相关的数据资源需要保存，一般推荐使用xml文件来保存它们，这种资源被称为原始xml资源，以下是如何定义、获取原始xml资源。

1.6.1定义原始xml资源

保存位置：/res/xml/,需自行创建；

在xml中访问语法：`[<package_name>:]xml/file_name`

java中访问语法：`[<package_name>.]R.xml.<file_name>`

可通过Resource如下两个方法获取实际的xml文档；

- XmlResourceParsser getXml(int id):获取XML文档，并使用一个XmlPullParser来解析该XML文档，该方法返回一个解析器对象（XmlResourceParser是XmlPullParser的子类）。
- InputStream openRawResource（int id）：获取xml文档对应的输入流。

大部分情况都可以直接调用getXml(int id)方法来获取XML文档，并对该文档进行解析。Anroid默认使用内置的Pull解析器来解析XML文件。

除使用Pull解析之外，也可以使用DOM或SAX对XML文档进行解析。一般的Java或Kotlin应用会使用JAXP API来解析XML文档；

1.7样式和主题资源

- 存放位置：/res/values/

样式资源文件的根元素是<resources.../>元素，该元素内可包含多个<style.../>子元素 ，每个<style.../>子元素定义一个样式，<style.../>元素指定如下两个属性。

- name:指定样式的名称；
- parent：指定该样式所继承的父样式。当继承某个父样式时，该样式将会获得父样式中的全部格式。当然，当前样式也可以覆盖父样式中指定的格式。

在<style.../>元素内可包含多个<item.../>子元素，每个<item.../>子元素定义一个格式项。

主题与样式的区别主要体现在：

- 主题不能作用于单个View组件，主题应该对整个应用中的所有Activity起作用，或对指定的Activity起作用。
- 主题定义的格式应该是改变窗口外观的格式，例如窗口标题、窗口边框等。

【注意】在Android.manifest可直接配置 还有下直接配置；

1.8 属性资源

存放在/res/values/下，根元素是<resources.../>该元素包含如下两个子元素。

- attr子元素：定义一个属性；

- declare-styleable子元素：定义一个styleable对象，每个styleable对象就是一组attr属性的集合。

1.9 使用原始资源

只要是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 的原始资源可以放在如下两个地方。

- 位于/res/raw/目录下，Android SDK会处理该目录下的原始资源，Android SDK会在R清单类中为该目录下的资源生成一个索引项；
- 位于/assets/目录下，该目录下的资源更是彻底的原始资源。Android应用需要通过AssetManager来管理目录下的原始资源。

AssetManager是一个专门管理/assets/目录下原始资源的管理类，提供了如下两个常用方法；

- InputStream open(String fileName):　根据文件名来获取原始资源对应输入流。
- AssetFileDescriptor openFd(String fileName):根据文件名来获取原始资源对应的AssetFileDescriptor 。AssetFileDescriptor 代表了一项原始资源的描述，应用程序可通过AssetFileDescriptor 来获取原始资源。

1.10 国际化

给/res/values/中的values命名

1.11 自适应不同屏幕资源

Android默认吧drawable目录（存放图片等Drawable资源的目录）分为draw-ldpi、drawable-mdpi、drawable-hdpi、drawable-xhdpi、drawable-xxhdpi这些子目录，他们正是用于不同图片的分辨率屏幕做准备；

通常来说，屏幕资源需要考虑如下两个方面；

- 屏幕尺寸：屏幕尺寸可分为small（小屏幕）、normal（中等屏幕）、large（大屏幕）、xlarge（超大屏幕）4种；
- 屏幕分辨率：屏幕分辨率可分为ldpi(低分辨率)，mdpi（中等分辨率）、hdpi（高分辨率）、xhdpi（超高分辨率）、xxhdpi（超高分辨率）5种；
- 屏幕方向：屏幕方向可分为land（横屏）和port（竖屏）两种。

  当为不同尺寸的屏幕设置用户界面时，每种用户界面总有一个最低的屏幕尺寸要求（低于该屏幕尺寸就没法运行），上面这些通用说法中屏幕尺寸总有一个最低分辨率，该最低分辨率是以dp为单位的，因此我们在定义界面布局时应尽量考虑使用dp为单位；

以下是上面4种屏幕的最低尺寸：

- xlarge屏幕尺寸最小要960dp*720dp;
- large屏幕尺寸最小要640dp*480dp;
- normal屏幕尺寸最小要470dp*320dp;
- small屏幕尺寸最小要426dp*320dp;

通过上面的介绍不难发现，为了提供适应不同屏幕的资源，最简单的做法就是为不同屏幕尺寸、不同屏幕分辨率提供相应的布局资源、Drawable资源。

- 屏幕分辨率：可以为drawable目录增加后缀ldpi（低分辨率）、mdpi（中等分辨率）、hdpi（高分辨率）、xhdpi（超高分辨率）、xxhdpi(超超高分辨率)，分别为不同分辨率的屏幕提供资源；

- 屏幕尺寸：可以为layout、values等目录增加后缀small、normal、large和xlarge，分别为不同尺寸的屏幕提供相应资源；

小结

本文主要介绍了Anroid应用资源的相关内容。Android应用资源是一种非常优秀的、高解耦设计，通过使用资源文件，Android应用可以把各种字符串、图片、颜色、界面布局等交给XML文件配置管理，这样就避免在java或Kotlin代码中以硬编码的方式直接定义这些内容。