Android移动开发

Android Mobile Application Development

第8讲 资源Resources

吴吟凡

计算机学院 一教505

yfwu@hdu.edu.cn

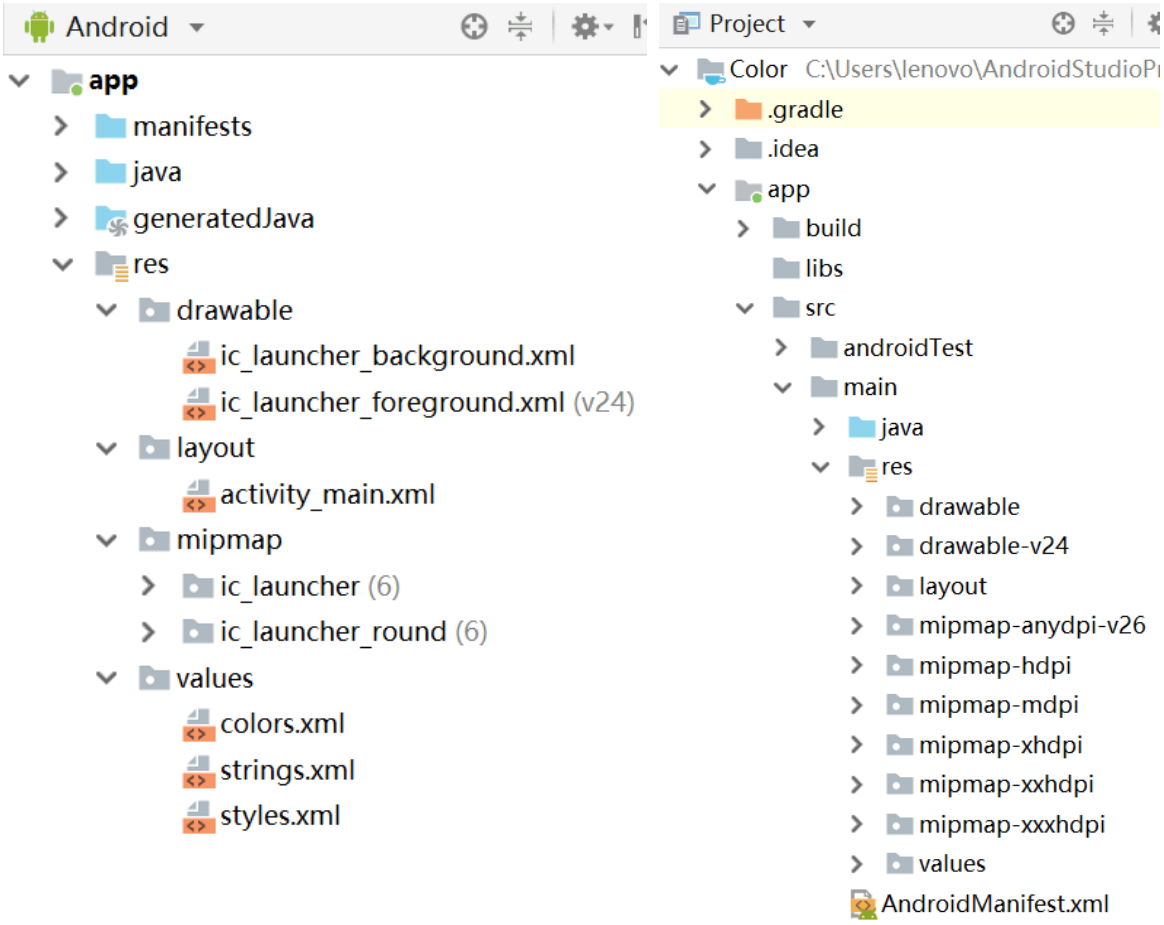
Android资源

Android资源介绍

- Android中的资源是在代码中使用的外部文件，是程序中重要的组成部分
- 在应用程序中经常会使用字符串、菜单、图像、声音、视频等内容，这些统称为资源。
  - 这些文件作为应用程序的一部分，被编译到应用程序的apk包中
  - 如果资源文件过大，也可以将资源作为外部文件使用
- 在代码中使用Context的getResources()方法得到Resources对象，该对象提供了获得各种类型资源的方法

Android资源存储

- Android 资源文件的存储操作对于Android资源也是非常重要的
- 在Android开发中，资源文件是使用频率最高的，如layout、string、drawable都是经常用到的，并且对开发提供了很大的方便。资源文件目录如右图所示



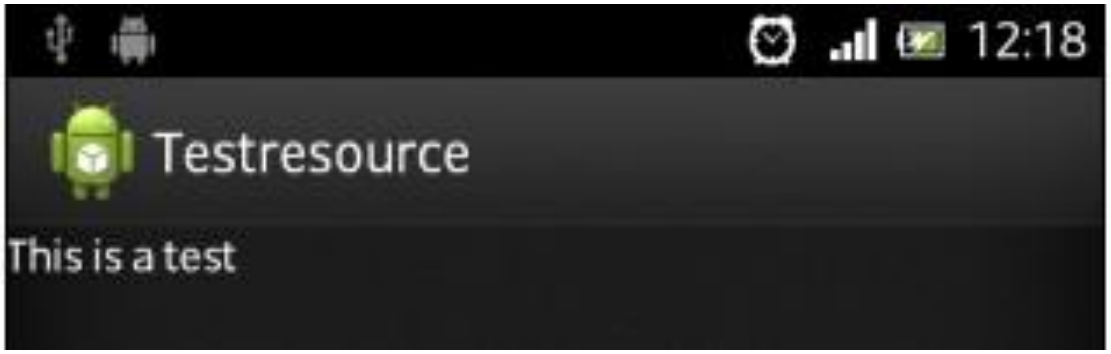
资源创建

- 开发人员可以使用Android提供的大量系统资源，同时还允许开发人员定制自己的资源，这些资源都可以通过R类中相应的ID来引用。
  - 资源文件名的命名要符合Java变量名的命名规则
- 资源的创建非常简单，根据资源类别，将资源文件复制到对应目录下。例：将mp3文件复制到res/raw下：

  - 在Android项目中，该资源文件可通过其ID进行引用：R.raw.test

添加资源示例

- 创建一个工程，在res/values/string.xml添加一个资源。即在该文件中添加如下代码：
  - <string name="test">"This is a test"</string>
- 由ADT自动生成R类（R.java）
  - public static final int test = 0x7f040002;
- 在代码中对此资源进行调用：
  - tv.setText(**getResources().getString(R.string.test)**);



资源类型

- 字符串和字符串数组：res/values/strings.xml、res/values/arrays.xml
- 颜色值和尺寸：res/values/colors.xml、res/values/dimens.xml
- 位图图像：res/drawable
- 动画序列：res/anim
- 菜单文件：res/menu
- XML文件：res/xml
- 原始文件：res/raw
- 布局文件：res/layout
- 样式和主题：res/values
- 任意类型：assets

资源类型

- 不同类型的资源可以用相应的资源类进行管理

int getColor(int id)

Drawable getDrawable(int id)

InputStream openRawResource(int id)

float getDimension(int id)

字符串资源

- 字符串存储在res/values目录下的XML文件中，其XML文件名也可以任意取
- 字符串使用<string name="...">...</string>标签定义，其中name属性标识key，也即R.string类中的变量ID
- 例如：

```
<string name="name">张三</string>
<string name="password">123456</string>
```

  - 在代码中可以用R.string.name和R.string.password引用name和password资源
  - 在布局中可以用@string/name和@string/password引用资源

```
textview1.setText(getResources().getString(R.string.name));
```

```
<TextView ... android:text="@string/name" />
```

数组资源

- 数组可以被当作资源保存在XML中，保存路径为res/values目录下，其XML文件名也可以任意取，它包括两种类型的资源：
  - 字符串数组：使用<string-array>标签定义
  - 整数数组：使用<integer-array>标签定义
- 例：
  - 可以在代码中引用数组：

```
<resources>
  <string-array name="name">
     <item >张三</item>
     <item >李四</item>
   </string-array>
   <integer-array name="age">
     <item>11</item>
     <item>16</item>
   </integer-array>
</resources>
```

```
String[] names = getResources().getStringArray(R.array.name);
int[] ages = getResources().getIntArray(R.array.age);
```

颜色资源

- 颜色资源通常在res/values/colors.xml中定义
- Android中颜色值资源的规范是以“#”开头，有四种表示方式
  - #RGB、#ARGB、#RRGGBB、#AARRGGBB
  - 其中RGB为Red Green Blue三原色，A为Alpha透明度，取值均为0~255；A取0表示完全透明，255为不透明；(0,0,0)为黑色、(255,255,255)为白色
- 例：

```
<resources>
   <color name="color_btn">#F0F</color>
   <color name="color_bk">#4567</color>
</resources>
```

颜色资源

- 使用资源定义的颜色
  - 在代码中可以用R.color.color_btn和R.color.color_bk引用color_btn和color_bk颜色资源

```
int color = this.getColor(R.color.color_btn);
```

  - 在布局中可以用@color/color_btn和@color/color_bk引用资源

- 其他创建颜色的方法
  - 预定义的颜色： android.graphics.Color类的静态常量

```
int color = Color.BLUE;
```

  - 静态工厂方法

```
int color = Color.argb(127, 255, 0, 255);
```

尺寸资源

- 尺寸资源保存在res/values下的任意命名的XML文件中，用<dimen>标签定义
  - 尺寸单位：px, in, mm, pt, dp, sp
- 例：

```
<resources>
   <dimen name="size1">30sp</dimen>
   <dimen name="size2">30px</dimen>
   <dimen name="size3">2.1in</dimen>
   <dimen name="size4">10pt</dimen>
   <dimen name="size5">6dp</dimen>
   <dimen name="size6">10sp</dimen>
</resources>
```

  - 在代码中获得尺寸资源

```
float size1 =getResources().getDimension(R.dimen.size1);
```

图像资源

- 图像资源放置在res/drawable目录下
  - 图像资源类型：png, jpg, jpeg, gif, 9.png
  - 不能存在多个文件名相同的文件（即使它们的扩展名不同）
- 例：在res/drawable目录下有一张background.png图片
  - 在代码中引用

```
Drawable pic = this.getDrawable(R.drawable.background);
```

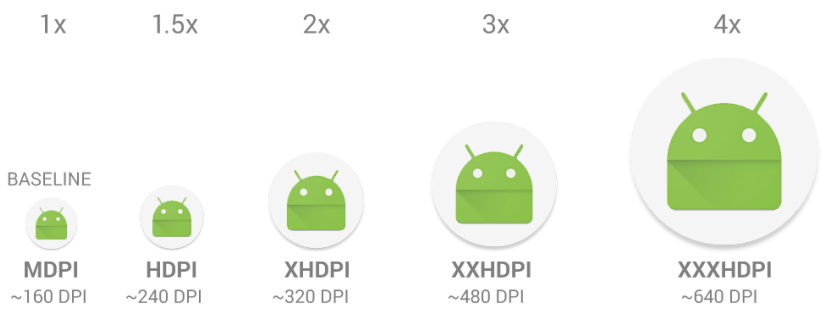
  - 在布局文件中引用

```
android:background = "@drawable/background"
```

图像资源

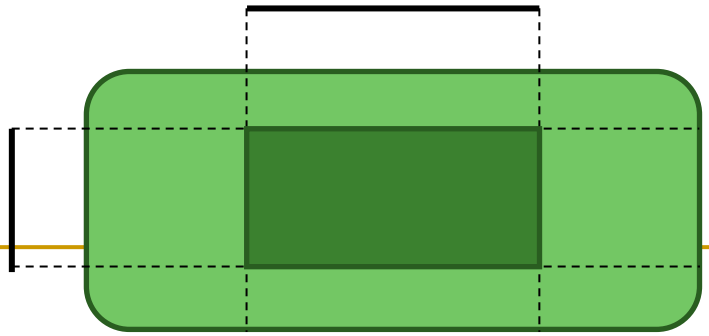
- 不同分辨率屏幕下使用不同图像
  - Android在不同像素密度（ pixel densities）的屏幕下会自动缩放图片，造成模糊
  - 每一张图片提供多个分辨率版本
    - Android根据设备屏幕的像素密度自动选择图片

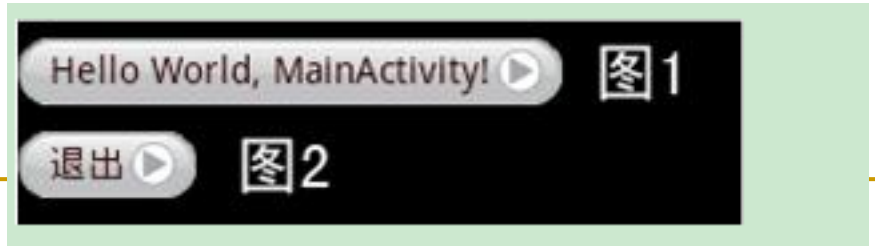
```
res/
  drawable-xxxhdpi/
    awesome-image.png
  drawable-xxhdpi/
    awesome-image.png
  drawable-xhdpi/
    awesome-image.png
  drawable-hdpi/
    awesome-image.png
  drawable-mdpi/
    awesome-image.png
```



图像资源

- 为不同分辨率的手机、平板提供拉伸但不失真的图像：.9.png文件
- Android Studio中右键点击png图片，选择Create 9-Patch file
  - 在原图片四周各添加宽度为1像素的线条
  - 缩放规则
    - 左边黑线代表图片垂直拉伸的区域，上边黑线代表水平拉伸区域
    - 右边黑线代表内容绘制的垂直区域，下边黑线代表内容绘制的水平区域
    - 右边和下边的线是可选的，左边和上边的线不能省略





动画资源

- Android动画的4种类型
  - alpha：渐变透明度动画效果
  - scale：渐变尺寸伸缩动画效果
  - translate：画面移动动画效果
  - rotate：画面转动动画效果
- Android动画有2种模式
  - 帧动画
  - 补间动画

```
<set xmlns:android="http://schemas
      .android.com/apk/res/android">
  <alpha android:fromAlpha="0.1"
    android:toAlpha="1.0"
    android:duration="5000" />
</set>
```

```
Animation myAnimation = AnimationUtils.loadAnimation(this, R.anim.animtest);
```

菜单资源

- 放在res/menu目录下的XML文件
- 使用<menu>设置根节点，<item><group>设置菜单项和分组

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
  <item android:id="@+id/zhonghua"
        android:icon="@drawable/zh" android:title="中华"/>
  <group android:id="@+id/all" >
    <item  android:id="@+id/xiaoxiongmao"
           android:icon="@drawable/xxm" android:title="小熊猫"/>
    <item  android:id="@+id/jinsihou"
           android:icon="@drawable/jsh" android:title="金丝猴"/>
  </group>
</menu>
```

菜单资源

- 可以创建多种菜单
  - 选项菜单：onCreateOptionMenu
  - 上下文菜单：onCreateContextMenu

```
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    MenuInflater inflater = getMenuInflater();
    inflater.inflate(R.menu.menu, menu);
    return true;
}
```

```
menu.getItem(5).getSubMenu().setHeaderIcon(R.drawable.ts);
```

原始文件资源

- 存储在res/raw目录下，可用R类引用

```
String text = “”;
InputStream in =getResources().openRawResource(R.raw.text);
int length = in.available();
byte[] buffer = new byte[length];
in.read(buffer);
text = EncodingUtils.getString(buffer, "UTF-8");
in.close();
```

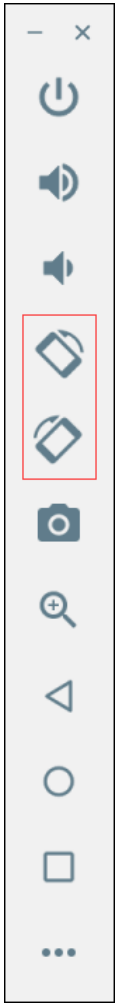
- 存储在assets目录下，用文件名引用

```
String path = "file:///android_asset/文件名";
InputStream in =getResources().getAssets().open(path);
```

```
InputStream path = getClass().getResourceAsStream("/assets/文件名");
```

布局资源

- 布局资源目录
  - res/layout
- 横屏布局
  - 将模拟器切换到横向模式
    - Ctrl+Left / Ctrl+Right
  - 为屏幕横向模式创建一个不同的布局
    - res/layout-land/main.xml



样式资源

- 存放在res/values目录下的XML文件

```
<resources>
   <style name="myTextViewStyle">
      <item name="android:textSize">50px</item>
      <item name="android:textColor">#00FF00</item>
   </style>
</resources>
```

```
<TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/hello_world"
      style="@style/myTextViewStyle"/>
```

主题资源

- 主题：包含一种或多种格式化属性的集合，可用于application或某个Activity

```
<resources>
  <style name= “MyTheme” parent= “Theme.AppCompat.Light">
    <item name="android:windowBackground">@color/theme_color</item>
    <item name="android:colorBackground">@color/theme_color</item>
  </style>
</resources>
```

```
<application ... android:theme="@style/MyTheme" >
```

```
<activity... android:theme="@style/MyTheme" >
```

使用系统资源

- SDK中提供了大量的系统资源，使用这些资源可以方便地实现一些设置，应用形式如下：
  - android.R.resourceType.resourceId
  - resourceType表示资源类型，例如string、color等。resourceId表示资源ID，即R类的内嵌类中定义的int类型的ID。
- 系统资源可以在代码之中使用，也可以在XML布局文件中使用。例如：
  - setTextColor(getColor(android.R.color.black));
  - android:src="@android:drawable/ic_media_play"

资源国际化

- 由于Android的产品手机，平板或者电视等等要出产不同的国家和地区，不同的国家和地区之间的语言是不相同的
- 国际化就是指系统或者程序界面文字，图片等会随着系统语言的切换而切换，以达到最舒服的用户体验
- 通过提供不同语言的字符串实现界面字符串的国际化

实现国际化过程

- 为字符串资源建立国际化目录，然后将相应的资源文件放到这些目录中, 国际化目录规则如下：
  - 资源目录 + 国际化配置选项
- 其中资源目录是指 res 目录中的子目录
  - 例如：values,drawable等
- 国际化配置选择项包含很多部分，中间用 " - " 分隔
  - 例如：表示中文和中国的配置选项是 zh-rCN; 表示英文和美国的配置选项是en-rUS；zh和en表示中文和英文；CN 和 US 表示中国和美国。

实现国际化过程

- 资源目录命名规则
  - 值之间用连字符-连接
  - 值是大小写敏感的
  - 每种限定词只能有一种选择
    - drawable-rEN-rFR不允许
  - 可添加多种限定词
    - drawable-en-rUS-480*320
    - drawable-ldpi, drawable-mdpi, drawable-hdpi, drawable-xhdpi
  - 带限定词的目录不能被嵌套
    - res/drawable/drawable-en不允许

实现国际化实例

- 在res 目录中建立两个文件夹: values-zh-rCN 和 values-en-rUS。 并在这两个目录中各建立一个strings.xml。
  - values-zh-rCN 目录中的 strings.xml 文件

```
<resources>
   <string name="app_name">中文</string>
   <string name="junior">初中</string>
   ......
</resources>
```

  - values-en-rUS 目录中的strings.xml 文件

```
<resources>
   <string name="app_name">English</string>
   <string name="junior">junior</string>
   ......
</resources>
```

实现国际化实例

- 引用资源文件代码如下：

```
<TextView  android:id="@+id/textView1"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/junior" />
 <TextView android:id="@+id/textView2"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/senior" />
<TextView android:id="@+id/textView3"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="@string/university" />
```

实现国际化实例

- 当将系统切换中/英文，应用程序将显示不同效果，如下图所示。