样式和主题

本文内容

另请参阅

- 样式和主题资源
- 适用于 Android 样式和主题的 `R.style`
- 适用于所有样式属性的 `R.attr`

样式是指为 `View` 或窗口指定外观和格式的属性集合。样式可以指定高度、填充、字体颜色、字号、背景色等许多属性。 样式是在与指定布局的 XML 不同的 XML 资源中进行定义。

Android 中的样式与网页设计中层叠样式表的原理类似 — 您可以通过它将设计与内容分离。

例如，通过使用样式，您可以将以下布局 XML：

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="#00FF00"
    android:typeface="monospace"
    android:text="@string/hello" />
```

简化成这个样子：

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />
```

布局 XML 中所有与样式有关的属性都已移除，并置于一个名为 `CodeFont` 的样式定义内，然后通过 `style` 属性加以应用。 您会在下文中看到对该样式的定义。

主题是指对整个 `Activity` 或应用而不是对单个 `View`（如上例所示）应用的样式。 以主题形式应用样式时，Activity 或应用中的每个视图都将应用其支持的每个样式属性。 例如，您可以 Activity 主题形式应用同一 `CodeFont` 样式，之后该 Activity 内的所有文本都将具有绿色固定宽度字体。

定义样式

要创建一组样式，请在您的项目的 `res/values/` 目录中保存一个 XML 文件。 可任意指定该 XML 文件的名称，但它必须使用 `.xml` 扩展名，并且必须保存在 `res/values/` 文件夹内。

该 XML 文件的根节点必须是 `<resources>`。

对于您想创建的每个样式，向该文件添加一个 `<style>` 元素，该元素带有对样式进行唯一标识的 `name` 属性（该属性为必需属性）。然后为该样式的每个属性添加一个 `<item>` 元素，该元素带有声明样式属性以及属性值的 `name`（该属性为必需属性）。 根据样式属性，`<item>` 的值可以是关键字字符串、十六进制颜色值、对另一资源类型的引用或其他值。以下是一个包含单个样式的示例文件：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <style name="CodeFont" parent="@android:style/TextAppearance.Medium">
        <item name="android:layout_width">fill_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

`<resources>` 元素的每个子项都会在编译时转换成一个应用资源对象，该对象可由 `<style>` 元素的 `name` 属性中的值引用。 可从 XML 布局以 `@style/CodeFont` 形式引用该示例样式（如上文引言中所示）。

`<style>` 元素中的 `parent` 属性是可选属性，它指定应作为此样式所继承属性来源的另一样式的资源 ID。 如果愿意，您可在随后替换这些继承的样式属性。

切记，在 XML 中定义您想用作 Activity 或应用主题的样式与定义视图样式的方法完全相同。 诸如上文所定义的样式可作为单个视图的样式加以应用，也可作为整个 Activity 或应用的主题加以应用。 后文将阐述如何为单个视图应用样式或如何以应用主题形式应用样式。

继承

您可以通过 `<style>` 元素中的 `parent` 属性指定应作为您的样式所继承属性来源的样式。您可以利用它来继承现有样式的属性，然后只定义您想要更改或添加的属性。 您可以从自行创建的样式或平台内建的样式继承属性。 （如需了解有关从 Android 平台定义的样式继承属性的信息，请参阅下文的使用平台样式和主题。） 例如，您可以继承 Android 平台的默认文本外观，然后对其进行修改：

```
    <style name="GreenText" parent="@android:style/TextAppearance">
        <item name="android:textColor">#00FF00</item>
    </style>
```

如果您想从自行定义的样式继承属性，则 *不必* 使用 `parent` 属性， 而是只需将您想继承的样式的名称以前缀形式添加到新样式的名称之中，并以句点进行分隔。 例如，要创建一个继承上文定义的 `CodeFont` 样式的新样式，但将颜色设置为红色，您可以按如下方式创建这个新样式：

```
    <style name="CodeFont.Red">
        <item name="android:textColor">#FF0000</item>
    </style>
```

请注意，`<style>` 标记中没有 `parent` 属性，但由于 `name` 属性以 `CodeFont` 样式名称（这是您创建的一个样式）开头，因此这个样式会继承该样式的所有样式属性。 这个样式随后会替换 `android:textColor` 属性，将文本设置为红色。 您可以 `@style/CodeFont.Red` 形式引用这个新样式。

您可以通过使用句点链接名称继续进行这样的继承，次数不限。 例如，您可以通过以下代码将 `CodeFont.Red` 扩大：

```
    <style name="CodeFont.Red.Big">
        <item name="android:textSize">30sp</item>
    </style>
```

这段代码同时从 `CodeFont` 和 `CodeFont.Red` 样式继承，然后添加 `android:textSize` 属性。

> **注：**这种通过将名称链接起来的继承方法只适用于由您自己的资源定义的样式。 您无法通过这种方法继承 Android 内建样式。 要引用内建样式（例如 `TextAppearance`），您必须使用 `parent` 属性。

样式属性

既然您已了解了样式是如何定义的，就需要了解什么类型的样式属性（由 `<item>` 元素定义）可以使用。您多半已经熟悉了其中的一些，例如 `layout_width` 和 `textColor`。 当然，还有许多其他样式属性可供您使用。

相应的类引用最便于查找适用于特定 `View` 的属性，其中列出了所有支持的 XML 属性。 例如，TextView XML 属性表中所列的所有属性都可在

TextView 元素（或其其中一个子类）的样式定义中使用。 该引用中列出的其中一个属性是 android:inputType，因此，如果您正常情况下会在 <EditText> 元素中放置 android:inputType 属性，如下所示：

```
<EditText
    android:inputType="number"
    ... />
```

您就可以改为给包括该属性的 EditText 元素创建一个样式：

```
<style name="Numbers">
  <item name="android:inputType">number</item>
  ...
</style>
```

这样您的布局 XML 现在便可实现这个样式：

```
<EditText
    style="@style/Numbers"
    ... />
```

这个简单示例可能显得工作量更大，但如果您添加更多样式属性并将能够在各种地方重复使用样式这一因素考虑在内，就会发现回报可能很丰厚。

如需查看所有可用样式属性的参考资料，请参阅 R.attr 参考资料。 切记，所有 View 对象仍然不接受样式属性，因此正常情况下您应该引用所支持样式属性的具体 View 类。 不过，如果您应用样式的 View 不支持所有样式属性，该 View 将只应用那些受支持的属性，并直接忽略其他属性。

不过，某些样式属性任何 View 元素都不提供支持，只能以主题形式应用。 这些样式属性应用于整个窗口而非任何类型的 View。例如，主题的样式属性可以隐藏应用标题、隐藏状态栏或更改窗口的背景。 这些类型的样式属性不属于任何 View 对象。要发现这些仅主题样式属性，请在 R.attr 参考资料中查看有关以 window 开头的属性的内容。 例如，windowNoTitle 和 windowBackground 是只有在样式以主题形式应用于 Activity 或应用时才起作用的样式属性。 请参阅下文有关以主题形式应用样式的信息。

> **注：**别忘了使用 android: 命名空间为每个 <item> 元素中的属性名称添加前缀。 例如：<item name="android:inputType">。

对 UI 应用样式和主题

设置样式的方法有两种：

- 如果是对单个视图应用样式，请为布局 XML 中的 View 元素添加 style 属性。
- 或者，如果是对整个 Activity 或应用来应用样式，请为 Android 清单中的 <activity> 或 <application> 元素添加 android:theme 属性。

当您对布局中的单个 View 应用样式时，该样式定义的属性只应用于该 View。 如果对 ViewGroup 应用样式，子 View 元素将**不会**继承样式属性 — 只有被您直接应用样式的元素才会应用其属性。 不过，您*可以*通过以主题形式应用样式，使所应用的样式作用于所有 View 元素。

要以主题形式应用样式定义，您必须在 Android 清单中将样式应用于 Activity 或应用。 如果您这样做，Activity 或应用内的每个 View 都将应用其支持的每个属性。 例如，如果您对某个 Activity 应用前面示例中的 CodeFont 样式，则所有支持这些文本样式属性的 View 元素也会应用这些属性。 任何不支持这些属性的 View 都会忽略这些属性。 如果某个 View 仅支持部分属性，将只应用这些属性。

对视图应用样式

为 XML 布局中的视图设置样式的方法如下：

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />
```

现在该 TextView 将按照名为 CodeFont 的样式的定义设置样式（请参阅上文定义样式中的示例）。

> **注：**`style` 属性不使用 `android:` 命名空间前缀。

对 Activity 或应用应用主题

要为您的应用的所有 Activity 设置主题，请打开 `AndroidManifest.xml` 文件并编辑 `<application>` 标记，在其中加入带样式名称的 `android:theme` 属性。 例如：

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

如果您只想对应用中的一个 Activity 应用主题，则改为给 `<activity>` 标记添加 `android:theme` 属性。

正如 Android 提供了其他内建资源一样，有许多预定义主题可供您使用，可免于自行编写。 例如，您可以使用 `Dialog` 主题，为您的 Activity 赋予类似对话框的外观：

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

或者，如果您希望背景是透明的，则可使用 Translucent 主题：

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

如果您喜欢某个主题，但想做些调整，只需将该主题添加为您的自定义主题的 `parent`。 例如，您可以像下面这样对传统明亮主题进行修改，使用您自己的颜色：

```
<color name="custom_theme_color">#b0b0ff</color>
<style name="CustomTheme" parent="android:Theme.Light">
    <item name="android:windowBackground">@color/custom_theme_color</item>
    <item name="android:colorBackground">@color/custom_theme_color</item>
</style>
```

（请注意，此处颜色需要以单独资源形式提供，因为 `android:windowBackground` 属性仅支持对另一资源的引用；不同于 `android:colorBackground`，无法为其提供颜色字面量。）

现在，在 Android 清单内使用 `CustomTheme` 替代 `Theme.Light`：

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

根据平台版本选择主题

新版本的 Android 可为应用提供更多主题，您可能希望在这些平台上运行时可以使用这些新增主题，同时仍可兼容旧版本。 您可以通过自定义主题来实现这一目的，该主题根据平台版本利用资源选择在不同父主题之间切换。

例如，以下这个声明所对应的自定义主题就是标准的平台默认明亮主题。 它位于 `res/values` 之下的一个 XML 文件（通常是 `res/values/styles.xml`）中：

```
<style name="LightThemeSelector" parent="android:Theme.Light">
    ...
</style>
```

为了让该主题在应用运行在 Android 3.0（API 级别 11）或更高版本系统上时使用更新的全息主题，您可以在 `res/values-v11` 下的 XML 文件中加入一个替代主题声明，但将父主题设置为全息主题：

```
<style name="LightThemeSelector" parent="android:Theme.Holo.Light">
    ...
</style>
```

现在像您使用任何其他主题那样使用该主题，您的应用将在其运行于 Android 3.0 或更高版本的系统上时自动切换到全息主题。

`R.styleable.Theme` 提供了可在主题中使用的标准属性的列表。

如需了解有关根据平台版本或其他设备配置提供备用资源（例如主题和布局）的详细信息，请参阅提供资源文档。

使用平台样式和主题

Android 平台提供了庞大的样式和主题集合，供您在应用中使用。 您可以在 `R.style` 类中找到所有可用样式的参考资料。 要使用此处所列样式，请将样式名称中的所有下划线替换为句点。 例如，您可以使用 `"@android:style/Theme.NoTitleBar"` 应用 `Theme_NoTitleBar` 主题。

不过，`R.style` 参考资料并不完备，未对样式做全面说明，因此查看这些样式和主题的实际源代码可让您更清楚地了解每个样式提供的样式属性。如需查看更详实的 Android 样式和主题参考资料，请参阅以下源代码：

- Android 样式 (styles.xml)
- Android 主题 (themes.xml)

这些文件有助于您通过示例进行学习。例如，在 Android 主题源代码中，您可以找到 `<style name="Theme.Dialog">` 的声明。 在该定义中，您可以看到用来为 Android 框架使用的对话框设置样式的所有属性。

如需了解有关 XML 中样式和主题语法的详细信息，请参阅样式资源文档。

如需查看您可用来定义样式或主题的可用样式属性（例如“windowBackground”或“textAppearance”）的参考资料，请参阅 `R.attr` 或您创建的样式所对应的 View 类。