Android 样式系统 | 主题背景和样式

原创 Android 谷歌开发者 6月30日

来自专辑

Android 开发技术

Android 提供了功能强大的样式系统 (Android styling system) 来实现应用的视觉设计，但它也容易被误用。正确地使用样式系统会让您在开发应用的时候更容易维护主题与样式，在开发新功能的时候少一些抓狂，而且还可以支持深色模式。

本系列文章将由 Android 开发者关系团队的工程师 Nick Butcher 和 Chris Banes 共同撰写，与各位开发者们共同揭开 Android 样式系统的神秘面纱，帮助您高效编写时尚的应用界面。

在本系列的第一篇文章中，我会介绍样式系统的基础部件: 主题背景与样式。

- Android styling system
  https://developer.android.google.cn/guide/topics/ui/look-and-feel/themes

主题背景 != 样式

主题背景与样式都使用相同的 <style> 语法，但是它们所服务的目的截然不同，您可以把它们理解为使用键值对 (Key-Value) 来存储数据，其中键 (Key) 代表属性，值 (Values) 代表资源，我们分别来看一下。

样式 (Style) 里有什么?

样式是 View 属性 (View Attributes) 值的集合，您可以把它们理解为 Map<view attribute, resource> 的结构。其中，一组键 (Key) 代表了所有的 View 属性，这里的 View 属性指的是可以在布局文件使用的 Widget 定义的属性。一个样式对应一种类型的 Widget，这是因为不同的部件支持不同的属性集合:

样式是 View 属性 (View Attributes) 值的集合；一个样式对应一种类型的 Widget

```
<!-- Copyright 2019 Google LLC.
   SPDX-License-Identifier: Apache-2.0 -->
<style name="Widget.Plaid.Button.InlineAction" parent="…">
  <item name="android:gravity">center_horizontal</item>
  <item name="android:textAppearance">@style/TextAppearance.CommentAuthor</ite
  <item name="android:drawablePadding">@dimen/spacing_micro</item>
</style>
```

正如您所见，样式中的每一个键 (Key) 其实就是您可以在布局中设置的内容:

```
<!-- Copyright 2019 Google LLC.
   SPDX-License-Identifier: Apache-2.0 -->
<Button …
  android:gravity="center_horizontal"
  android:textAppearance="@style/TextAppearance.CommentAuthor"
  android:drawablePadding="@dimen/spacing_micro"/
```

把这些提炼成样式，可以让您方便地在多个 View 中复用同一个样式，而且还容易维护。

使用方法

布局文件中的每一个独立的 View 都可以使用样式:

```
<!-- Copyright 2019 Google LLC.
   SPDX-License-Identifier: Apache-2.0 -->
<Button …
  style="@style/Widget.Plaid.Button.InlineAction"/>
```

一个 View 只能使用一个样式，可以将其与 Web 技术中使用到的 CSS 样式系统相比较，CSS 样式系统可以允许一个组件使用多个 CSS 类。

范围

样式只有在使用它的 View 上才起作用，如果该 View 包含子 View，那么在这些子 View 上样式是无效的。举个例子，如果您的 ViewGroup 有三个按钮，设置 InlineAction 样式到此 ViewGroup 时，只针对这个 ViewGroup 有效，而对它的三个按钮来说是无效的。样式中定义的值与布局文件中设置的值会融合在一起 (解决方法见这篇文章: 使用样式优先级顺序)。

- 使用样式优先级顺序
  https://medium.com/androiddevelopers/whats-your-text-s-appearance-f3a1729192d

什么是主题背景?

主题背景是一组命名的资源的集合，这些资源可以被样式或者布局文件等引用。它们提供了一种对 Android 资源的语义名称 (Sematic name)，能够让您在其他地方引用这些资源。例如 colorPrimary 就是对一个给定颜色的语义名称。

```
<!-- Copyright 2019 Google LLC.
   SPDX-License-Identifier: Apache-2.0 -->
<style name="Theme.Plaid" parent="…">
  <item name="colorPrimary">@color/teal_500</item>
  <item name="colorSecondary">@color/pink_200</item>
  <item name="android:windowBackground">@color/white</item>
</style>
```

主题背景是由 Map<theme attribute, resource> 结构组成，这些标有名字的资源被称为主题背景属性。主题背景属性跟 View 属性不一样，这是因为它们不是特定 view 类型的属性而是对一个

值的命名，其在应用中有更广泛的用途。主题背景属性为这些标有名字的资源提供了具体的值，在上面的例子中 colorPrimary 属性为这个主题背景设置了具体的值，也就是青绿色 (teal)。通过把主题背景中的资源抽象化，我们可以为不同的主题背景提供不同的值，比如: colorPrimary=orange。

主题背景是一个命名的资源集合，在应用中有更广泛的用途

主题背景类似于接口 (Interface)，在接口的编程中它允许您为公共接口提供不同的实现方法。主题扮演了一个类似的角色，针对主题属性编写布局和样式，我们可以在不同的主题下使用它们，从而提供不同的具体资源。

简化的伪代码如下:

```
/* Copyright 2019 Google LLC.
   SPDX-License-Identifier: Apache-2.0 */
interface ColorPalette {
  @ColorInt val colorPrimary
  @ColorInt val colorSecondary
}

class MyView(colors: ColorPalette) {
  fab.backgroundTint = colors.colorPrimary
}
```

这会让您使用同一套代码可渲染出不同的 MyView 效果，而无需新建构建变体。

```
/* Copyright 2019 Google LLC.
   SPDX-License-Identifier: Apache-2.0 */
val lightPalette = object : ColorPalette { … }
val darkPalette = object : ColorPalette { … }
val view = MyView(if (isDarkTheme) darkPalette else lightPalette)
```

使用方法

您可以把一个主题背景设置给一个组件，这个组件可以包含 Context 或者它本身就是 Context，比如: Activity 或者是 View/ViewGroups。

```
<!-- Copyright 2019 Google LLC.
   SPDX-License-Identifier: Apache-2.0 -->

<!-- AndroidManifest.xml -->
<application …
  android:theme="@style/Theme.Plaid">
<activity …
  android:theme="@style/Theme.Plaid.About"/>

<!-- layout/foo.xml -->
<ConstraintLayout …
  android:theme="@style/Theme.Plaid.Foo">
```

您还可以使用 ContextThemeWrapper 类把一个主题背景设置到已经存在的 Context 上，这时候您可使用 inflate 方法创建布局。

- ContextThemeWrapper
  https://developer.android.google.cn/reference/android/view/ContextThemeWrapper.html#ContextThemeWrapper(android.content.Context,%20int)
- inflate
  https://developer.android.google.cn/reference/android/view/LayoutInflater.html#from(android.content.Context)

主题背景的使用效果取决于您的使用方式，您可以通过引用主题背景属性来创建灵活的 Widget。不同的主题背景可以在未来再提供具体的值，比如为 View 层级结构中的某个部分设置背景颜色。

```
<!-- Copyright 2019 Google LLC.
   SPDX-License-Identifier: Apache-2.0 -->
<ViewGroup …
  android:background="?attr/colorSurface">
```

除了用常量值设置一个颜色 (#ffffff 或者 @color 资源)，我们还可以通过 ?attr/themeAttributeName 语法委托给主题背景来完成。

这个语法表示通过指定的属性名称，从主题背景中获取相应的值。这种级别的解耦方式可以让我们提供不同的程序行为 (比如: 在深色模式与浅色模式下提供不同的背景颜色)，而不用创建多个相似但仅有一小部分不一样的布局或者样式，它将主题中的可变元素分离了出来。

通过使用 ?attr/themeAttributeName 语法获得此主题背景中的语义属性代表的值

范围

任何一个带有 Context (如 Activity, View or ViewGroup) 的对象 (Object) 都可以通过访问 Context 的属性来访问主题背景。这些对象以树的形式组织而成，比如 Activity 包含 ViewGroup，而 ViewGroup 又包含 View。把主题背景设置到一个树状结构的任意一层，此层及下一层都会受到影响。比如把主题背景设置给一个 ViewGroup，此 ViewGroup 包含的所有子 View 都会受到这个主题背景的影响。(而样式恰好相反，它只对被设置的 View 起作用)

```
<!-- Copyright 2019 Google LLC.
   SPDX-License-Identifier: Apache-2.0 -->
<ViewGroup …
  android:theme="@style/Theme.App.SomeTheme">
  <! - SomeTheme also applies to all child views. -->
</ViewGroup>
```

- Context
  https://developer.android.google.cn/reference/android/content/Context
- 主题背景
  https://developer.android.google.cn/reference/android/content/res/Resources.Theme.html

如果您想在浅色屏幕中获取一个由深色主题背景构成的区域，那这个功能会非常有用。更多内容请参见本系列下一篇文章，我们会在之后更新。

请注意，这种功能仅在初始化布局的时候生效。在初始化布局之前需要调用 Context 提供的 setTheme 方法或者是主题背景提供的 applyStyle 方法。布局初始化完毕之后再调用 setTheme

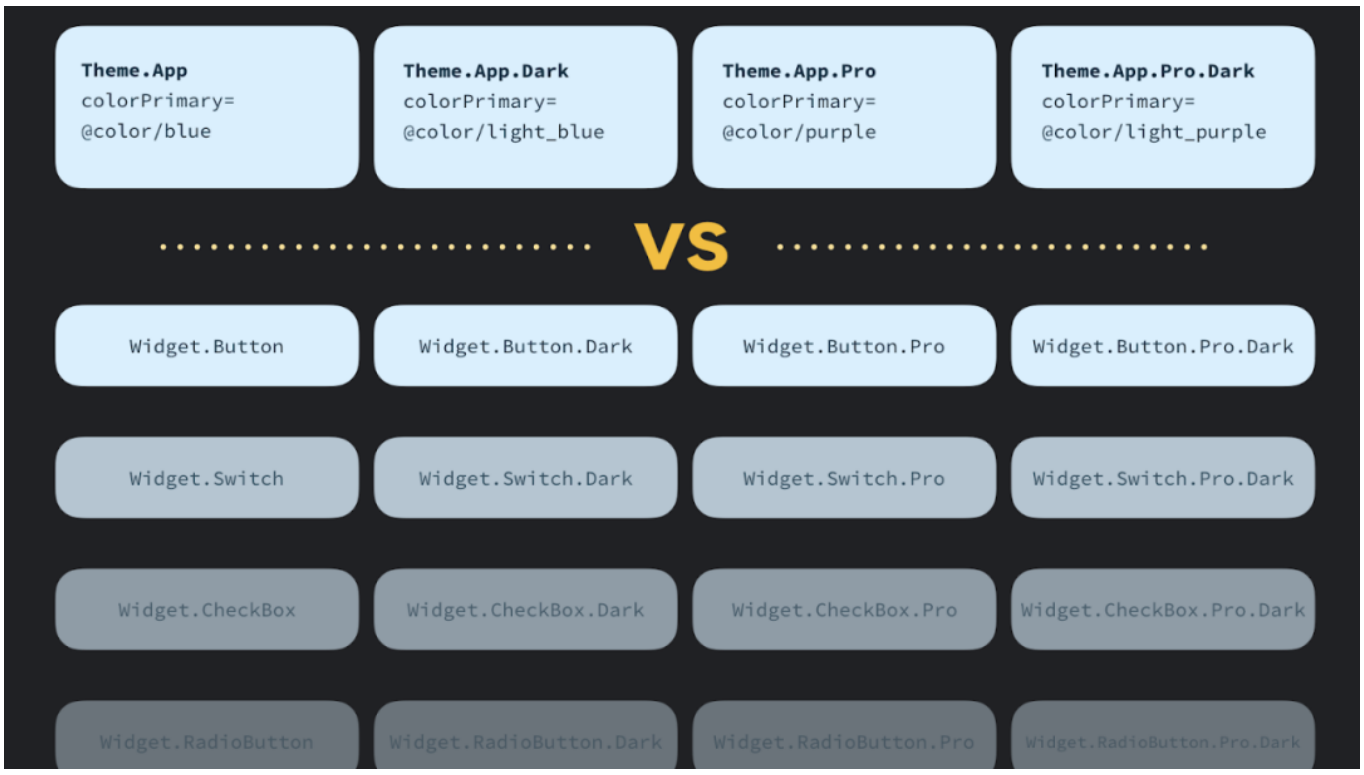
或者 applyStyle 方法，此时对已有的 View 不会造成任何改变。

- setTheme
  https://developer.android.google.cn/reference/android/content/Context#setTheme(int)
- applyStyle
  https://developer.android.google.cn/reference/android/content/res/Resources.Theme?hl=en#applyStyle(int,%20boolean)

不同的关注点

了解主题背景与样式的不同目的与使用方法，会让您更方便地管理样式资源。

举个例子，假设您的应用有一个蓝色主题背景，但某些 Pro 界面需要有花俏的紫色，而且您还想要提供一个调色过的深色主题。如果您只使用样式来实现这个效果，需要分别为 Pro/non-Pro 和 light/dark 创建四个不同的样式。由于样式是特定于一个视图类型 (按钮、开关等)，因此您需要为应用中的每一种 View 类型创建这四个样式。



△ 不含主题的 widgets 或样式的扩展组合

如果改为使用样式和主题背景，则可以将因主题背景变化而发生改变的部分封装为主题背景属性，因此我们仅需要为每种 View 类型定义一个样式。对于上面的示例，我们可以定义 4 个主题背景，为其中的 colorPrimary 主题背景属性提供不同的值，之后当样式引用这些主题的属性时会自动得到正确的值。

混合使用主题背景与样式的方法可能看起来相比之前更复杂了，但是它的好处是把每个主题变化的部分封装了起来。

因此，当您需要把程序的界面从蓝色改为橙色时，只需要修改一个地方就够了，而不需要修改多个样式。它还有助于您避免发生样式泛滥。

理想情况下，针对一个视图类型，您应该只有少数几种样式。如果不使用主题背景，您为几个长得类似的样式创建不同的扩展版本时，就会使得 styles.xml 文件很大，维护起来会非常头疼。

下一篇文章，我们将会跟大家共同探索主题背景的公共属性以及如何创建您自己的主题背景，敬请关注。

android