


## Android 样式系统 | 主题背景覆盖



谷歌开发者   
已认证的官方帐号

已关注

3 人赞同了该文章

在 Android 样式系统系列的前几篇文章中，我们探讨了 样式和主题背景之间的区别，讨论了 使用主题背景和主题背景属性的好处，并重点介绍了一些 常用的主题背景属性。

今天，我们聚焦于主题背景的实际使用，如何将它们应用到我们的应用中，以及如何构建主题背景。

### 范围

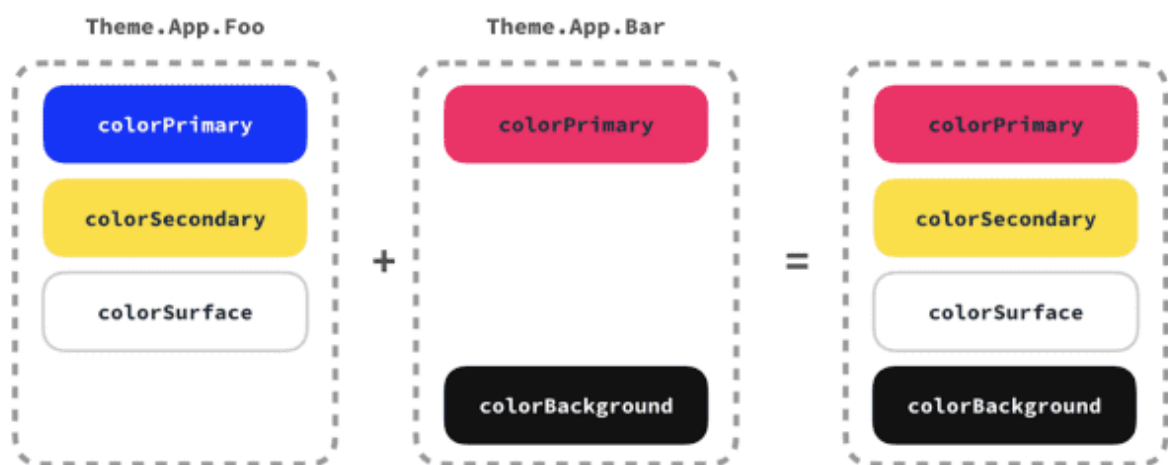
在 上一篇文章 中，我们提到：

任何一个拥有或者自己本身就是 Context (如 Activity, View or ViewGroup) 的对象都可以通过访问 Context 的属性来获取 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。比如在 ViewGroup 上设置一个主题背景，此 ViewGroup 包含的所有子 View 都会受到这个主题背景的影响。(只适用于单个 View 的样式则恰恰相反)

在树结构中的任何层级上设置主题背景，都不会替换当前生效的主题背景，但会将其覆盖 (Overlay)。一起看看下面这个 Button，该 Button 设置了一个主题背景，但是它父结构也指定了一个主题背景：

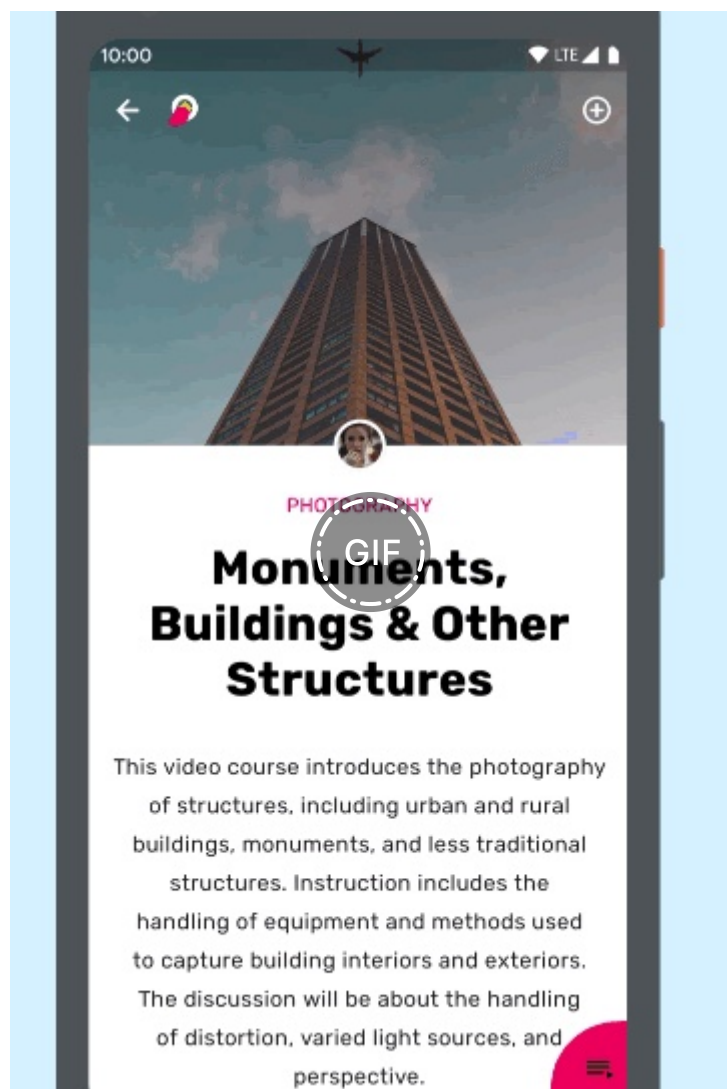
```
<!-- Copyright 2019 Google LLC.
SPDX-License-Identifier: Apache-2.0 -->
<ViewGroup ...
    android:theme="@style/Theme.App.Foo">
    <Button ...
        android:theme="@style/Theme.App.Bar"/>
</ViewGroup>
```

如果在两个主题背景中都指定了同一属性，则最邻近的 (local) 设置会生效，即 Bar 中的设置被应用于该 Button。任何在主题背景 Foo 中有指定，但是在主题背景 Bar 中未指定的属性也被应用于此 Button。



覆盖了各自的主题背景

这或许是一个不太恰当的例子，但样式化应用中不同外观的子区域时，这项技术的价值则被凸显出来。例如，浅色内容上有深色的工具栏，或者该界面 (比如，[Owl 示例应用](#)) 中显示了大面积的粉色主题背景但显示相关内容的底部具有蓝色主题背景：



通过在蓝色分区的根部 (Root) 设置主题背景的方式，可级联到它所有的子视图。

## 过度重叠

由于主题背景会覆盖树结构中更高一级的主题背景，因此请务必留意主题背景所指定的内容，以避免它意外替换您本想要保留的属性。例如，您可能只是想改变视图 (View) 的背景颜色 (通常由 `colorSurface` 控制)，即，您不打算更新该主题背景的其他部分。基于此，您可以试试**主题背景覆盖 (Theme Overlay)** 的技术。

设计这些主题背景的目的在于覆盖其他主题背景。它们的作用范围需要尽可能的狭小，也就是说，它们仅定义 (或继承) 最小化的属性。实际上，主题背景覆盖通常 (但并不总是) 是没有父级的，例如：

```
<!-- Copyright 2019 Google LLC.
      SPDX-License-Identifier: Apache-2.0 -->
<style name="ThemeOverlay.MyApp.DarkSurface" parent="">
  <item name="colorSurface">#121212</item>
</style>
```

*主题背景覆盖是限定范围的主题背景，定义的属性要越少越好，它的作用只是为了覆盖另外一个主题背景*

按照惯例，我们以 "ThemeOverlay" 为前缀给这些主题背景覆盖起名字。[MDC](#) (和 [AppCompat](#)) 提供了许多有用的主题背景覆盖 (Theme Overlay)，您可以使用它们来把应用程序子区域的颜色从浅色转换到深色：

- [ThemeOverlay.MaterialComponents.Dark](#)
- [ThemeOverlay.MaterialComponents.Light](#)

根据定义，主题背景覆盖不会指定很多内容，同时也不应单独使用。例如，作为您 Activity 的主题背景。实际上，您可以认为在应用中可以使用两种 "类型" 主题：

1. **"完整" 主题背景。** 它们定义了一个屏幕所需的一切。它们继承了另一个 "完整" 主题背景 (如，`Theme.MaterialComponents`)，因此可以将其设置为 Activity 主题背景。
2. **主题背景覆盖。** 仅应用于 "完整" 的主题背景。由于其不会指定重要且必要的信息，因此不应该单独使用。

## 永远存在

总会有一个有效的主题背景，即使您未在应用中的任何地方指定一个主题背景，您也会继承 **默认主题**。因此，上面的示例只是一种简化，因此您绝对不应该在 View 中使用一个 "完整" 的主题背

景，而应使用主题背景覆盖：

```
<!-- Copyright 2019 Google LLC.  
      SPDX-License-Identifier: Apache-2.0 -->  
<ViewGroup ...  
-   android:theme="@style/Theme.App.Foo">  
+   android:theme="@style/ThemeOverlay.App.Foo">  
<Button ...  
-   android:theme="@style/Theme.App.Bar"/>  
+   android:theme="@style/ThemeOverlay.App.Bar"/>  
</ViewGroup>
```

这些主题背景覆盖不会孤立地存在，因为它们本身会被外围的 Activity 的主题背景所覆盖。

## 成本效益

使用主题背景需要一些运行时的代价。每次您声明 `android:theme` 时，您都在创建一个新的 **ContextThemeWrapper**，它会分配新的主题背景 (Theme) 和资源 (Resources) 实例。它还需要解决多层级样式化的间接引用问题。

注意不要过度使用主题，您应该监控它们的影响，特别是在重复使用的情况下，例如：RecyclerView 项的布局或者配置文件。

## 在上下文中使用

我们曾说过主题背景与 Context 相关联，这意味着，如果您在代码中使用 Context 来获取资源 (Resource)，请确保您使用的是正确的 Context。例如，您可以在代码中的某个位置获取 Drawable：

```
someView.background = AppCompatResources.getDrawable(requireContext(),  
    R.drawable.foo)
```

如果 Drawable 引用了主题背景属性 (所有的 Drawable 从 API 21+ 开始生效，VectorDrawables 可以通过 Jetpack 从 API 14+ 开始生效)，则应确保使用正确的 Context 来加载 Drawable。如果不清楚 Context 是否正确的话，您可能会遇到在尝试应用背景主题到子层级时不生效的情况，届时您可能会陷入困惑并且搞不清楚究竟发生了什么。例如，如果您使用 Fragment 或 Activity 的 Context 来加载 Drawable，应用在树结构底层的主题背景就会失效。最佳做法是，应使用离资源 (Resource) 最近的 Context：

```
someView.background = AppCompatResources.getDrawable(someView.context,  
    R.drawable.foo)
```

## 误用

我们已经讨论了树结构中存在的主题背景和 Context: Activity > ViewGroup > View 等。将这种思维模型扩展到 Application 级，听起来很吸引人——毕竟您可以在 manifest 中通过 标签指定一个主题背景。**千万不要被愚弄！**

Application Context **不保留**任何主题背景相关信息，您在 manifest 中设置的主题背景仅用作未明确设置主题背景的 Activity 的默认选择。因此，您绝不要在 Application Context 中 **加载资源** (如 Drawable 或者颜色，因为它们可能因主题背景不同而不同) 或者用来解析主题背景属性。

*切勿使用 Application Context 加载可使用的资源*

这也是为什么我们把 "完整" 主题背景应用到 Activity，并从 Application 主题背景维度对这种组织结构进行了扩展。级别的主题背景不会覆盖 级别的主题背景。

## 强调

希望这篇文章已经解释清楚了主题背景覆盖在树结构中的功能，以及在样式化我们 App 的时候如何使用这个功能。使用 android:theme 标签为布局中的分段设置主题背景，并仅在您需要调整属性的地方使用主题背景覆盖。请注意使用正确的主题背景和 Context 来加载资源，并谨慎使用 Application Context！

编辑于 10-26

[Android](#)   [Android 开发](#)