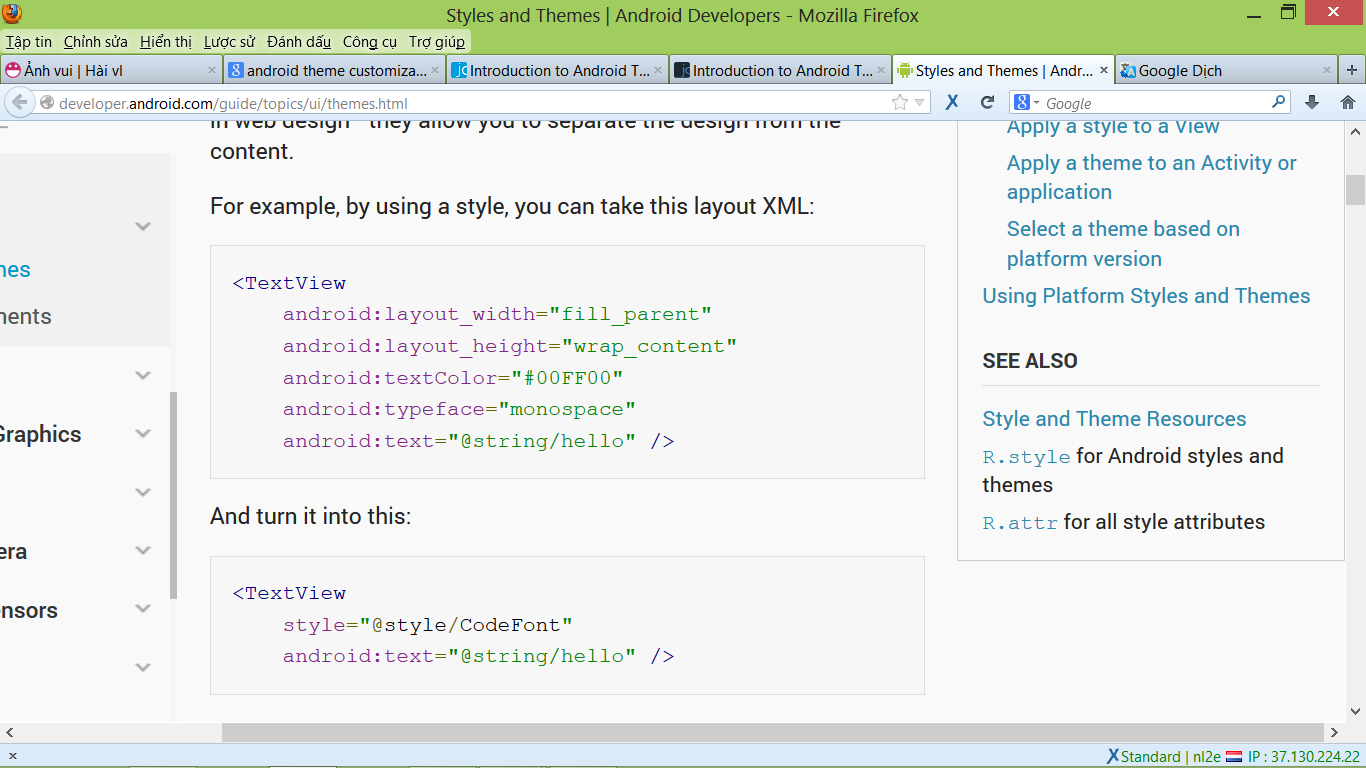
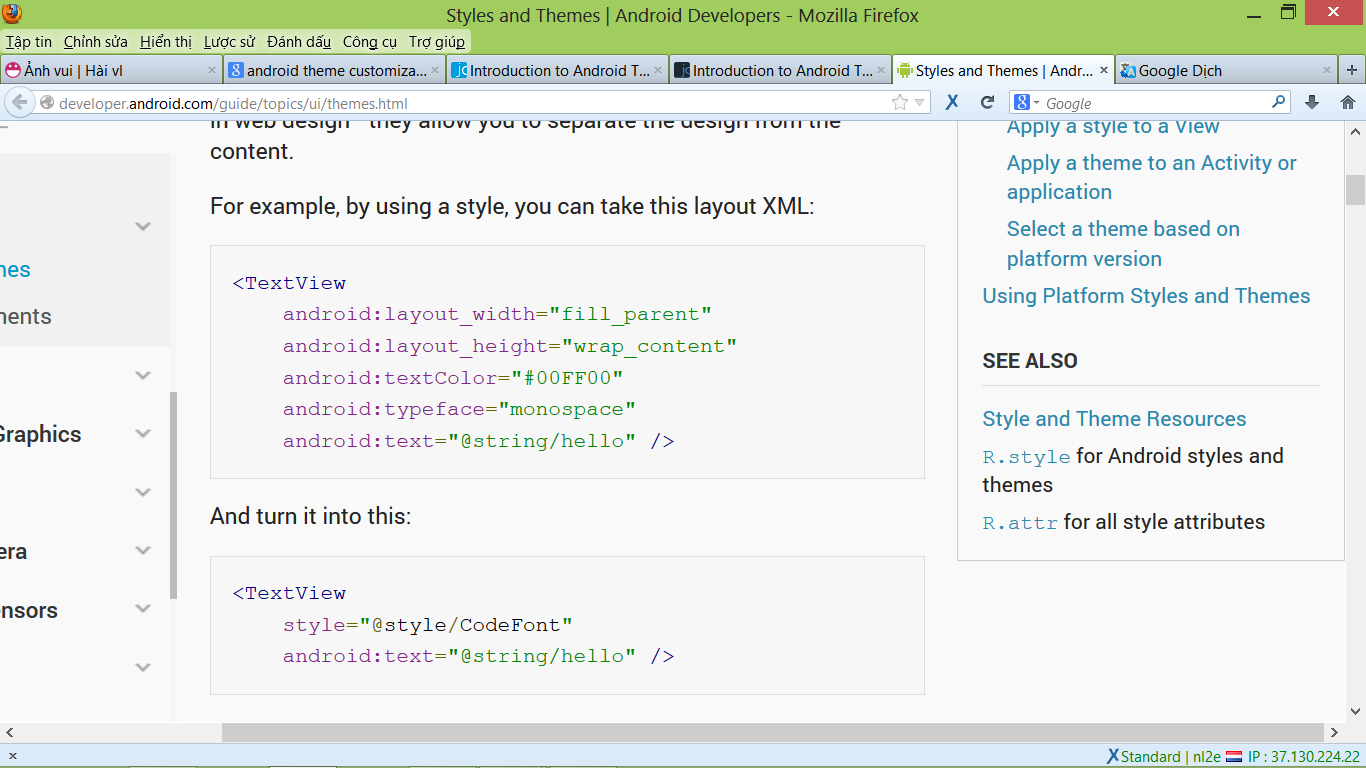
Style và Theme

# Style

Một style là một tập hợp các thuộc tính về định dạng cho một View. Style bao gồm các thuộc tính về chiều cao, lề, font chữ, màu chữ, màu nền…. Một style được định nghĩa trong xml và được gán vào view. Style trong android tương tự như CSS trong thiết kế web. VD ta có một textview như sau:



Và ta tách cách thuộc tính ra bỏ vào một style tên CodeFont sau đó áp style đó trở lại cho textview. Cách này giúp mã của ta tách thành nhiều phần riêng biệt cũng như có thể tái sử dụng.



# Theme

Một Theme chính là một style được áp cho toàn bộ activity hoặc chương trình chứ không áp cho từng view riêng lẻ. Khi một style được xem như theme và áp cho activity thì toàn bộ view trong activity để dính style đó. Vd ta có thể áp style CodeFont như vd trên cho toàn Activity khi đó tất cả textview trong activity sẽ dùng chung style là CodeFont.

# Định nghĩa một style

Để tạo ra các style ta cần tạo ra file xml và chứa trong đường dẫn res/values/ . Tên file xml là tùy ý (tuy nhiên khi ta tạo project thì đã có một file tên styles.xml rồi, ta có thể thêm trong file này).

Nút gốc của file phải là <resources>

Cho mỗi style tạo ra ta thêm một thẻ <style> với thuộc tính name là tên của style. Bên trong ta dùng các thẻ <item> để định nghĩa cho từng thuộc tính.

<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>

Tạo ra một TextView và áp style vào như sau:

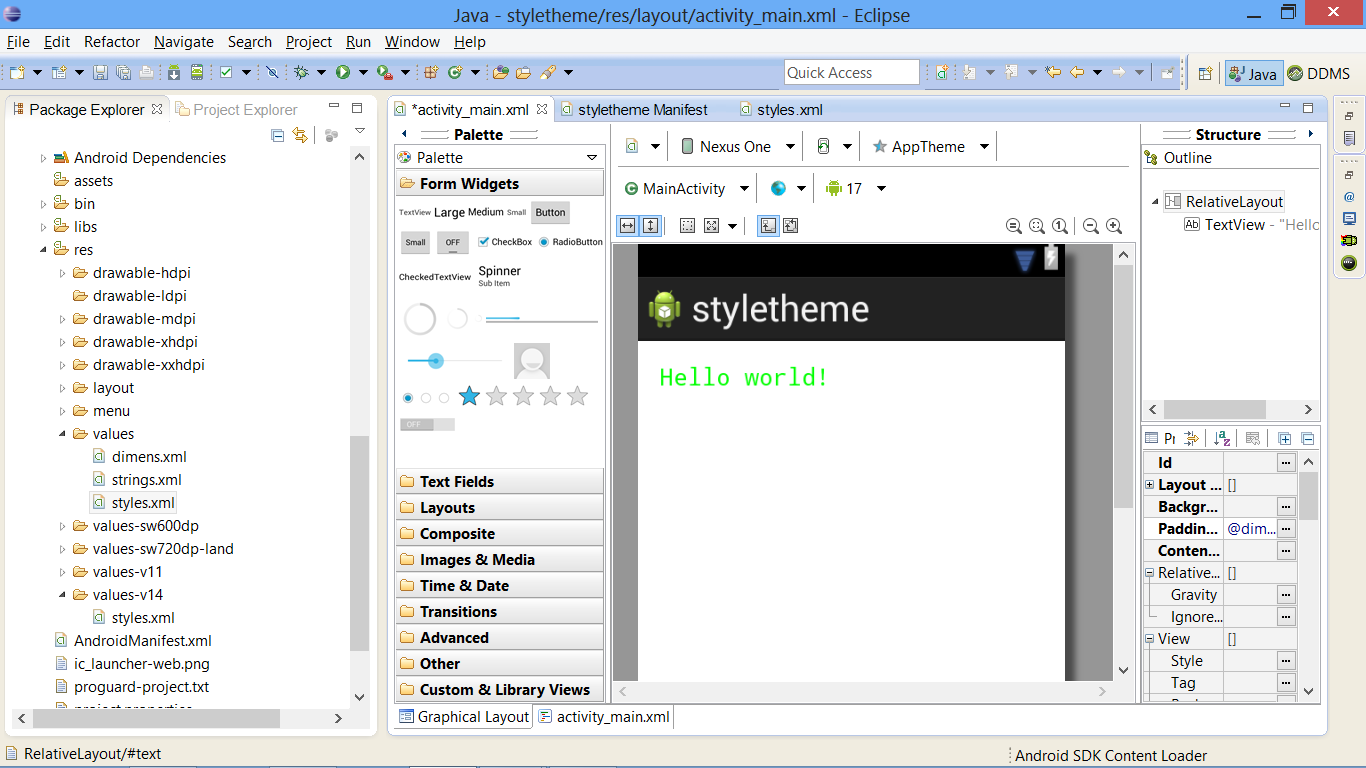
<TextView

android:layout\_width=*"wrap\_content"*

android:layout\_height=*"wrap\_content"*

style=*"@style/CodeFont"*

android:text=*"@string/hello\_world"* />



# Sự kế thừa Inheritance

Thuộc tính parent trong thẻ <style> cho phép ta chỉ ra một style của hệ thống mà ta sẽ kế thừa. Ta nên kế thừa từ một style của hệ thống sau đó viết dậm thêm các thuộc tính mà ta cần thay đổi. xem lại ví dụ trên

<style name=*"GreenText"* parent=*"@android:style/TextAppearance"*>

<item name=*"android:textColor"*>#ff0000</item>

</style>

Nếu muốn kế thừa từ một style của ta đã có trước đó ta không dùng thuộc tính parent mà dùng tên của style trước và thêm dấu chấm vào. Vd:

<style name=*"CodeFont.Red"* >

<item name=*"android:textColor"*>#ff0000</item>

</style>

Chú ý rằng ta không dùng thuộc tính parent nữa nhưng do style của ta kế thừa từ CodeFont mà CodeFont kế thừa từ một style hệ thống do đó style của ta vẫn dính style của hệ thống.

Và cứ thế ta có thể dùng tiếp dấu “.” để kế thừa tiếp.

<style name=*"CodeFont.Red.Big"*>

<item name=*"android:textSize"*>30sp</item>

</style>

# Các thuộc tính style

Các style có thể tra cứu trong R.attr theo địa chỉ <http://developer.android.com/reference/android/R.attr.html>

Thuộc tính style có nhiều nhưng không phải tất cả các loại view đều có chung các style, một view sẽ hỗ trợ một số style này và một view khác sẽ hỗ trợ một số loại style khác. Tuy nhiên khi ta áp các style vào một loại view mà nó không hỗ trợ thì đơn giản nó sẽ bỏ qua.

Có một số style không có view nào hỗ trợ nó, nó chỉ dùng cho Theme chứ không dùng cho view. Các style này áp cho toàn bộ cửa sổ activity ví dụ những style để ẩn application title, status bar, hay thay đổi nền của window nên nó không áp cho view. Những style dạng này sẽ bắt đầu bằng từ khóa window vd: windowNoTitle, windowBackground.

# Áp Style và Theme vào UI

Có 2 cách để áp một style:

* Áp style cho từng view, bằng cách thêm thuộc tính style cho view trong file layout XML.
* Áp style cho toàn bộ Activity hoặc application bằng cách thêm thuộc tính android:theme cho <activity> hoặc <application> trong manifest.

Khi áp 1 style cho View thì chỉ có view đó dính style. Khi áp style cho ViewGroup, thì các view con trong ViewGroup đó không kế thừa thuộc tính style (nó không được áp style). Khi đó muốn áp style cho tất cả các view con ta dùng đến theme.

# Áp theme cho Activity hoặc Application

Để gán theme cho tất cả các activity trong chương trình (application). Ta mở AndroidManifest.xml và thêm trong thẻ <application> thuộc tính android:theme. VD:

<application android:theme=*"@style/AppTheme"* >

Nếu muốn áp cho activity thì cũng tương tự nhưng viết trong thẻ Activity

<activity android:theme=*"@style/GreenText"* >

Có nhiều bộ theme đã được xây dựng sẵn để ta có thể sử dụng. Khi ta dùng các theme của hệ thống thì ta dùng theo @android:style/tenthemhethong .Ví dụ ta có thể sử dụng theme của Dialog để làm Activity xuất hiện giống Dialog.

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

Hoặc muốn nền trong suốt ta có thể dùng theme Translucent

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

Nếu thích một theme nhưng muốn sửa nó ta tạo một style mới và cho style kế thừa từ theme với thuộc tính parrent rồi sửa các thuộc tính mình thích.

<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>

(chú ý windowBackground chỉ hỗ trợ mà từ một tài nguyên)

# Áp theme dựa theo phiên bản

Các version mới có các theme mới, ta muốn dùng nó nhưng vẫn có thể tương thích với version cũ (tức là chạy trên version mới ra theme mới chạy trên version cũ ra theme cũ). Ví dụ khi đó ta tạo ra file style theo thư mục res/values/

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

...

</style>

Sau đó để có thể dùng theme Holo chỉ hỗ trợ từ 3.0 (api11) thì ta sẽ tạo ra một thư mục value mới trong res như sau res/values-v11 và trong đó ta có style áp theo kiểm Holo

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

...

</style>

Sau đó ta dùng theme này thì nó sẽ tự động áp theme tùy theo version.

Danh sách các thuộc tính có thể dùng trong theme có thể tìm thấy tại R.styleable.Theme: <http://developer.android.com/guide/topics/ui/themes.html>