Estilos y Temas

Mª Alejandra Tomás Fernández

Estilos y Temas

- En el **diseño web** se usan **hojas de estilo en cascada** (**CSS**) para crear un **patrón de diseño y aplicarlo a varias páginas**.
- En **diseño de Layouts** de una aplicación se utilizan unas herramientas similares conocidas como **estilos y temas.**
- Los **estilos y temas** permitirán crear **patrones de estilo que podrán ser utilizados en cualquier parte de la aplicación.**
- Estas herramientas **ahorrarán mucho trabajo** y permitirán conseguir un d**iseño homogéneo en toda una aplicación.**

Los estilos (I)

- Un **estilo** es una **colección de propiedades que definen el formato que tendrá una vista**.
- Podemos especificar cosas como **tamaño**, **márgenes**, **color**, **fuentes**, etc.
- **Un estilo se define en ficheros XML**, **diferente al fichero XML *Layout* que lo utiliza**.
- Veamos un ***ejemplo***. El siguiente código:

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

Es equivalente a escribir:

Los estilos (II)

```
<TextView
    style="@style/MiEstilo"
    android:text="Un texto" />
```

- Habiendo creado en el fichero ***res/values/styles.xml*** con el siguiente código:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <style name="MiEstilo"
     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>
```

- Observa como **un estilo puede heredar la propiedades de un padre** (parámetro parent) y a partir de estas propiedades realizar modificaciones

Heredar de un estilo propio (I)

- Si vas a **heredar de un estilo definido por ti no** es necesario **utilizar el atributo parent**.
- Puedes **utilizar el mismo nombre de un estilo** ya creado **y completar el nombre con un punto más un sufijo.**

- ***Por ejemplo***:

```
<style name="MiEstilo.grande">
     <item name="android:textSize">18pt</item>
</style>
```

Crearía un **nuevo estilo que sería igual a MiEstilo más la nueva propiedad indicada**.

Heredar de un estilo propio (II)

A su vez **puedes definir otro estilo a partir de este**:

```
<style name="MiEstilo.grande.negrita">
     <item name="android:textStyle">bold</item>
</style>
```

Los temas (I)

- Un **tema** es un **estilo aplicado a toda una actividad o aplicación,** *en lugar de a una* ***vista individual***.
- **Cada elemento del estilo solo se aplicará a aquellos elementos donde sea posible**.
- ***Por ejemplo***, **MiEstilo solo afectará al texto.**

- **Para aplicar un tema a toda una aplicación:**
  - edita el **fichero *AndroidManifest.xml*** y
  - añade el **parámetro `android:theme` en la etiqueta `application`.**

Los temas (II)

- **Para aplicar un tema a una Activity:**
  - edita el **fichero *AndroidManifest.xml*** y
  - añade el **parámetro `android:theme` en la etiqueta `activity`:**

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

- Además de crear tus propios temas **vas a poder utilizar algunos disponibles en el sistema.**

Los temas del sistema (III)

- **Android también tiene temas ya predefinidos que podemos usar *(al igual que tiene otros recursos)*:**

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

- Estos temas también los **podemos adaptar a nuestro gusto**:
  - Podemos ***añadir el tema del sistema como tema padre de uno nuestro*** personalizado.
  - ***Por ejemplo***, podemos modificar el típico tema “Theme.Light” para usar nuestro propio color así:

```
<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>
```

- Puedes encontrar una lista de todos los estilos y temas disponibles en Android en: http://developer.android.com/reference/android/R.style.html

Los temas (IV)

- **MUY IMPORTANTE:**

  - Si nuestra Activity extiende de AppCompatActivity

```
public class MainActivity extends AppCompatActivity
```

hemos de aplicar siempre un tema que sea descendiente de **Theme.AppCompat. .**

¿Cómo seleccionar temas diferentes según la API de Android? (V)

- Las versiones nuevas de Android añaden temas nuevos.
- Si queremos poder usarlo cuando nuestra app se ejecuta en terminales con estas APIs, podemos hacer esto:
  - Personalizo el ***tema Light*** (***por ejemplo***) y lo pongo **como tema por defecto** añadiendo el fichero XML en la carpeta *res/values (typically res/values/styles.xml):*

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

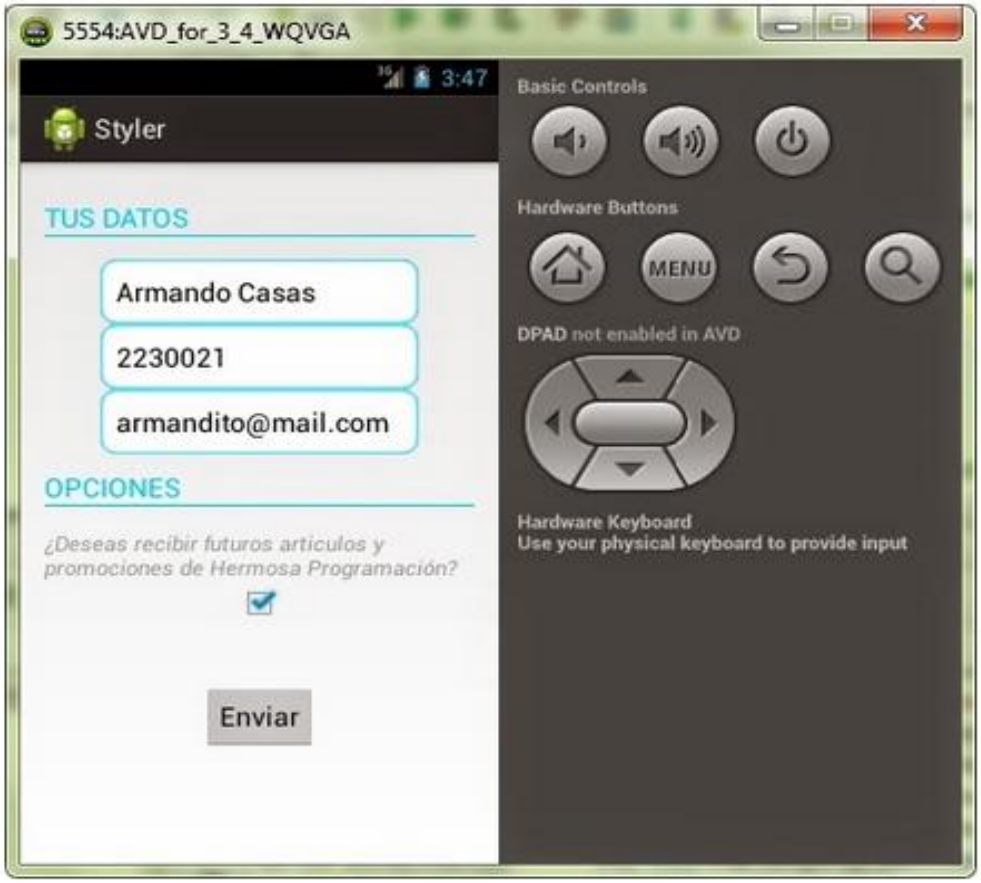
  - Cuando mi app se ejecute **en Android 3.0 (API 11) o superior**, quiero usar ***ese mismo tema, pero su versión HOLOGRÁFICA*** que es exclusiva de esas APIs. Para ello, escribo el siguiente **fichero alternativo "styles.xml" (FIJAROS QUE EL NOMBRE DEL ESTILO DEBE SER EL MISMO, ya que es un recurso alternativo)** y lo guardo en *res/values-v11:*

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

MUY BUEN EJEMPLO (I)

App De Ejemplo

Crearemos una pequeña aplicación que use un tema personalizado. Esta actividad contendrá un formulario hipotético para envío de datos a los suscriptores de **Hermosa Programación**. La idea es crear un tema que abarque los elementos generales de la aplicación y luego crear estilos para views específicos.



MUY BUEN EJEMPLO (II)

```
<resources>
    <!--Tema para el formulario-->
    <style name="AppTheme" parent="android:Theme.Holo.Light.DarkActionBar">
        <!-- Estilos para Edit Texts-->
        <item name="android:editTextStyle">@style/EditTextStyle</item>
        <!--Estilos de ventana-->
        <item name="android:windowFullscreen">true</item>
    </style>

    <!--Estilos personalizados para los componentes del formulario-->
    <style name="Header" parent="@android:style/Widget.Holo.Light.TextView">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textAppearance">
                                    ?android:attr/textAppearanceMedium</item>
        <item name="android:textColor">@android:color/holo_blue_bright</item>
        <item name="android:layout_marginTop">10dp</item>
    </style>
```

MUY BUEN EJEMPLO (III)

```
<style name="Message" parent="@android:style/Widget.Holo.Light.TextView">
        <item name="android:textStyle">italic</item>
        <item name="android:textColor">@android:color/darker_gray</item>
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
    </style>

    <style name="Separator">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">1dp</item>
        <item name="android:background">
                                    @android:color/holo_blue_bright</item>
        <item name="android:layout_marginBottom">
                                    @dimen/activity_vertical_margin</item>
    </style>

    <style name="EditTextStyle"
                          parent="@android:style/Widget.Holo.Light.EditText">
        <item name="android:background">@drawable/rectangle</item>
        <item name="android:padding">10dp</item>
    </style>

</resources>
```

MUY BUEN EJEMPLO (IV)

Hago un paréntesis para señalar el recurso que hemos usado en el background de los edit texts. Se trata de una forma creada manualmente para representar el contenido del fondo.

Para ello se creó un nuevo recurso drawable llamado rectangle.xml con la siguiente descripción:

```
<?xml version="1.0" encoding="utf-8"?>

<layer-list xmlns:android="http://schemas.android.com/apk/res/android" >

    <item android:top="1dp" android:bottom="1dp">
        <shape
            android:shape="rectangle">

            <stroke
                android:width="1dp"
                android:color="@android:color/holo_blue_bright" />
            <solid android:color="#ffffffff" />

            <corners
                android:radius="10dp"/>

        </shape>
    </item>

</layer-list>
```

MUY BUEN EJEMPLO (V)

Ahora diseñaremos el layout de nuestra actividad Main con la siguiente descripción:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"

    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    tools:context=".Main"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        style="@style/Header"
        android:text="@string/dataHeader"
        android:id="@+id/dataHeader"/>

    <View style="@style/Separator" />

    <EditText
        android:inputType="textPersonName"
        android:ems="10"
        android:id="@+id/nameField"
        android:layout_gravity="center_horizontal"
        android:hint="@string/nameField"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
```

MUY BUEN EJEMPLO (VI)

```
<EditText
    android:inputType="phone"
    android:ems="10"
    android:id="@+id/phoneField"
    android:layout_gravity="center_horizontal"
    android:hint="@string/phoneField"
    android:layout_height="wrap_content"
    android:layout_width="wrap_content" />

<EditText
    android:inputType="textEmailAddress"
    android:ems="10"
    android:id="@+id/emailField"
    android:layout_gravity="center_horizontal"
    android:hint="@string/emailField"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />

<TextView
    android:text="@string/optionsHeader"
    android:id="@+id/optionsHeader"
    style="@style/Header" />

<View style="@style/Separator" />

<TextView
    android:text="@string/newsletterText"
    android:id="@+id/newsletterText"
    android:layout_gravity="center_horizontal"
    style="@style/Message" />
```

MUY BUEN EJEMPLO (VII)

```
    <CheckBox
        android:id="@+id/confirmBox"
        android:layout_gravity="center_horizontal"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content" />

    <Button
        android:text="@string/sendButton"
        android:id="@+id/sendButton"
        android:layout_gravity="center_horizontal"
        android:layout_marginTop="41dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</LinearLayout>
```

Los Text Fields no tienen asignado ningún estilo, ya que su forma es heredada del tema de la aplicación.

Atributos de Estilo

Mª Alejandra Tomás Fernández

Atributos de estilo (I)

- Cuando aplicamos estilos a nuestros layout puede interesarnos **acceder a un atributo concreto de un estilo**.
- Para eso tenemos una ***sintaxis*** específica que podemos usar en nuestros XML:

```
?[<nombre_paquete>:][<tipo_recurso>/]<nombre_recurso>
```

- Por ejemplo, si queremos colocar un texto pequeño, podemos aplicar como estilo:

```
?android:attr.textAppearanceSmall
```

Atributos de estilo (II)

For example, here's how you can reference an attribute to set the text color to match the "primary" text color of the system theme:

```
<EditText id="text"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="?android:textColorSecondary"
    android:text="@string/hello_world" />
```

Here, the `android:textColor` attribute specifies the name of a style attribute in the current theme. Android now uses the value applied to the `android:textColorSecondary` style attribute as the value for `android:textColor` in this widget. Because the system resource tool knows that an attribute resource is expected in this context, you do not need to explicitly state the type (which would be `?android:attr/textColorSecondary`)—you can exclude the `attr` type.