**Usando styles nos layouts da sua aplicação Android**

Postado dia 22/03/2012 por André Silva em [Inovação](http://blog.caelum.com.br/category/inovacao/), [Mobile](http://blog.caelum.com.br/category/mobile/) [9](http://blog.caelum.com.br/usando-styles-nos-layouts-da-sua-aplicacao-android/#comments)

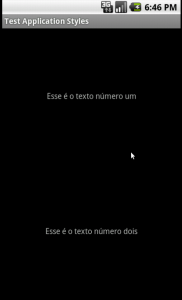
Construir layouts no Android não é uma tarefa fácil e exige bastante esforço. Não só pela fragmentação, mas também pela orientação e outros detalhes. Digamos que estamos construindo uma aplicação que terá dois layouts diferentes. Um para quando o dispositivo estiver com a tela em*portrait* e outro para quando o dispositivo estiver com a tela em *landscape*. Podemos resolver esse problema usando o [Application Resources](http://blog.caelum.com.br/explorando-o-application-resources-do-android/). Basta criarmos um main.xml na pasta layout e outromain.xml na pasta layout-land. Vamos começar com o portrait:

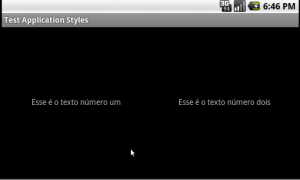
|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical" >        <TextView android:layout\_height="0dp" android:layout\_weight="1"           android:gravity="center" android:layout\_width="match\_parent"           android:text="Esse é o texto número um" />        <TextView android:layout\_height="0dp" android:layout\_weight="1"           android:gravity="center" android:layout\_width="match\_parent"           android:text="Esse é o texto número Dois" />  </LinearLayout> |

Já em landscape (dentro de res/layout-land), teremos android:orientation="horizontal" além da diferença de pesos entre largura e altura:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="horizontal" >     <TextView android:layout\_height="match\_parent" android:layout\_weight="1"         android:gravity="center" android:layout\_width="0dp"         android:text="Esse é o texto número um" />     <TextView android:layout\_height="match\_parent" android:layout\_weight="1"         android:gravity="center" android:layout\_width="0dp"         android:text="Esse é o texto número Dois" />  </LinearLayout> |

Os resultados serão:

[](http://caelum.wpengine.netdna-cdn.com/wp-content/uploads/2012/03/styles_portrait.png)

[](http://caelum.wpengine.netdna-cdn.com/wp-content/uploads/2012/03/styles_landscape.png)

Se não tivessemos criado um main.xml dentro de res/layout-land, em landscape nossa tela apresentaria as duas mensagens uma em cima da outra, aproveitando mal o espaço nesse caso. Ainda assim, temos alguns problemas técnicos, em especial ter de possuir dois main.xml diferentes, com muito código igual em seus interiores. Se tivermos de mudar o layout, teremos de alterar os dois arquivos.

Poderíamos agrupar o que for comum entre os layout, nos poupando de retrabalhos futuros. Para realizarmos o agrupamento de informações do layout, usamos o resource [Style](http://developer.android.com/guide/topics/resources/style-resource.html). Vamos criar, dentro da pasta res/values, um arquivo styles.xml. Dentro dele podemos ter os chamados “estilos” de cada tipo de componente, neste caso para o dispositivo em portrait:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <resources>       <style name="Orientation">            <item name="android:orientation">vertical</item>       </style>       <style name="TextStyle">            <item name="android:layout\_weight">1</item>            <item name="android:gravity">center</item>      </style>      <style name="TextWidth" parent="TextStyle">           <item name="android:layout\_width">match\_parent</item>           <item name="android:layout\_height">@dimen/zero</item>      </style>  </resources> |

Aqui definimos o estilo para o LinearLayout, que no caso de portrait será vertical. Para detalhes de apresentação de texto, definimos o peso e o gravity em um estilo chamado TextStyle (esse nome é arbitrário) e o outro, nomeado TextWidth, com o tamanho do campo. Repare no TextWidth que usamos a tag parent. Fazemos isso para poder reaproveitar estilos em outros estilos, e nesse caso quem utilizar TextWidth terá também os atributos declarados em TextStyle.

Para usarmos esses styles que acabamos de definir, basta aplicar a tag style dentro de nosso layout, como por exemplo style=@style/TextWidth. Veja como ficou nosso main.xml:

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"      style="@style/Orientation"      android:layout\_width="fill\_parent"      android:layout\_height="fill\_parent" >      <TextView style="@style/TextWidth" android:text="@string/textOne" />      <TextView style="@style/TextWidth" android:text="@string/textTwo" />  </LinearLayout> |

Repare que além de colocarmos os styles, nós também colocamos os textos e dimensões em arquivos strings.xml e dimens.xml. Estamos seguindo a convenção definida pelo Android de que dimenões e textos devem estar externalizados em seus respectivos arquivos. A [ferramenta lint](http://tools.android.com/tips/lint) pode te ajudar bastante a achar o que ainda não foi externalizado na sua aplicação.

Ainda temos dois arquivos main.xml, um está na pasta res/layout e outro na res/layout-land. Vamos apagar toda a pasta layout-land. Teremos apenas o arquivo main.xml na pasta res/layout. Agora vamos criar uma pasta chamada res/values-land. Dentro dessa pasta vamos criar um arquivo styles.xml.

Dentro do arquivo styles.xml vamos alterar os estilos que precisamos mudar, no nosso caso a orientação do linear e o tamanho dos TextVieẁs. Para isso, criamos o arquivo style.xml da pastares/values-land.

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <resources>      <style name="Orientation">          <item name="android:orientation">horizontal</item>      </style>      <style name="TextWidth" parent="TextStyle">          <item name="android:layout\_width">@dimen/zero</item>          <item name="android:layout\_height">match\_parent</item>      </style>  </resources> |

Você deve está se perguntando o porquê de não declararmos TextStyle nesse arquivo. Não precisamos escrever o TextStyle por que ele já foi declarado na pasta values default. Quando ele não encontra o style de TextStyle no res/values-land, ele vai procurar no arquivo style.xmldeclarado na pasta default, isso é, res/values. Agora sim, temos apenas um arquivo main.xml. Customizações do nosso layout ficam então definidos em arquivos de styles.