TextView. Definición de recursos XML

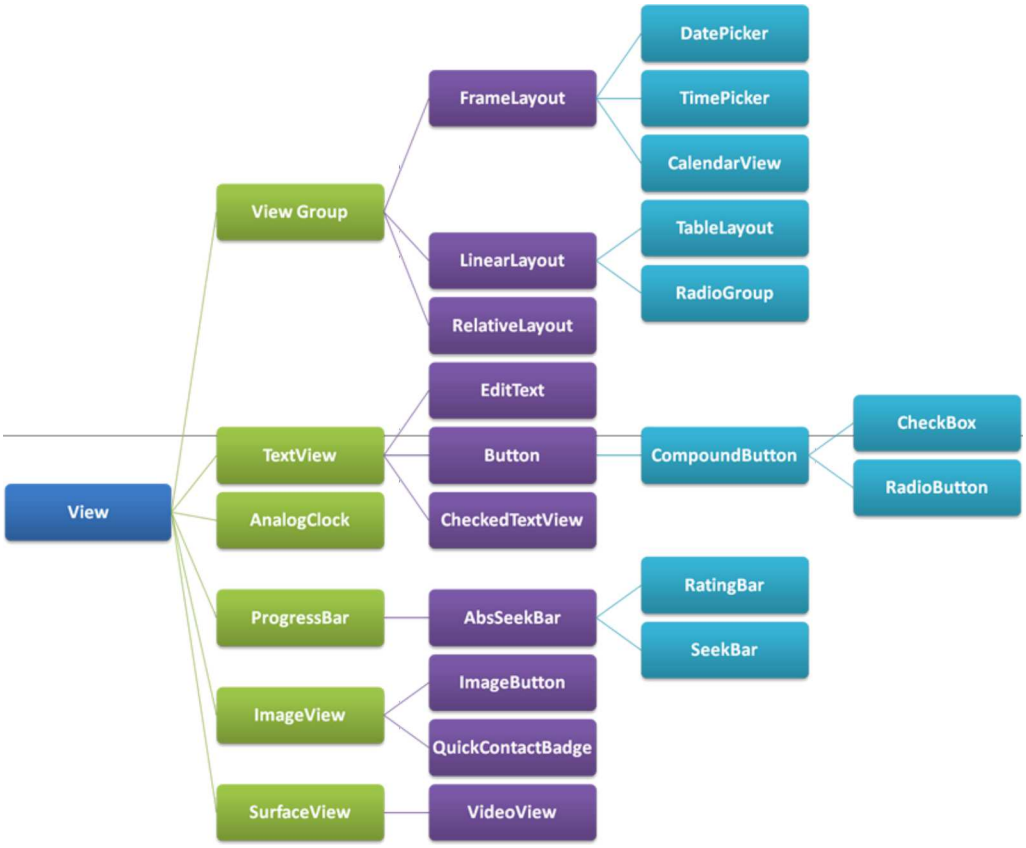
Sumario

Introdución

- Un **TextView** é unha etiqueta de texto que serve para amosar texto na aplicación.
- Este control é unha subclase da clase **View**.
- Opcionalmente pódese permitir editar o contido.
- A clase que implementa o control vén configurada para non permitir a edición.

- Aproveitando que este é o primeiro control que imos ver, tamén imos explicar como se definen **constantes** para usar nos recursos XML.

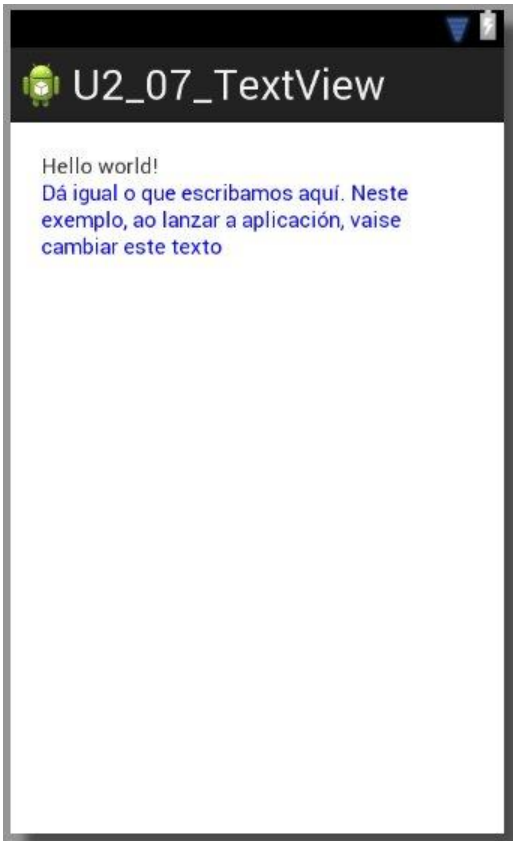
- **Referencias**:
  - A clase **View**: http://developer.android.com/reference/android/view/View.html
    - Antes de continuar é importante familiarizarse (non fai falla aprender nada) cos métodos, subclases e atributos que ten esta clase de que van colgar todos os demais controis. Co cal, aparte de pararse na seguinte imaxe, é aconsellable premer no enlace anterior.



- Imaxe obtida de: http://www.itcsolutions.eu/2011/08/27/android-tutorial-4-procedural-vs-declarative-design-of-user-interfaces
- Control TextView: http://developer.android.com/reference/android/widget/TextView.html
  - Observar no enlace anterior o valor do atributo **editable**.
- Recomendacións de tipografía: http://developer.android.com/design/style/typography.html
  - Observar no enlace anterior o tamaño recomendado en px do lanzador dunha aplicación en Google Play.
- Cores: http://developer.android.com/guide/topics/resources/more-resources.html#Color
  - Observar en que formatos se pode describir unha cor.

Casos prácticos

- Comezamos creando un novo proxecto: **U2_07_TextView**.
- Imos comezar creando un layout con 2 TextViews, onde o segundo TextView se modifica en tempo de execución (ao iniciarse a aplicación).



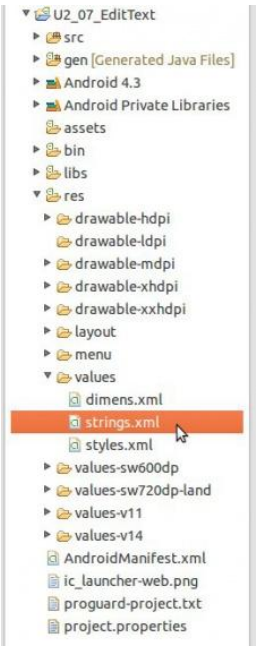
- A imaxe amosa en tempo de deseño os 2 TextViews.
- Le o contido da etiqueta azul.

- O XML do layout asociado a esa imaxe é:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="20sp"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/tv_orixinal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/hello_world" />

    <TextView
        android:id="@+id/tv_java"
```

```
16          android:layout_width="wrap_content"
            android:layout_height="wrap_content"
17 18          android:textColor="#00F"
19          android:text="Dá igual o que escribamos aquí. Neste exemplo, ao lanzar a aplicación, vaise cambiar este texto" />
20
21 </LinearLayout>
```

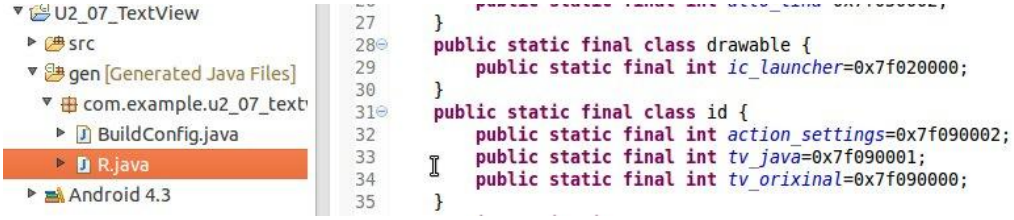
- **NOTA ECLIPSE**: os campos dos controis pódense poñer na orde que se desexe. Se se preme **SHIFT+CTRL+F** Eclipse vai organizar o código e ordenar os campos. Esa combinación de teclas pode usarse tanto en código XML como Java.
- Nas liñas 9 e 15 asóciase un ID a cada control, TextView, que logo usaremos en Java.
- Observar as diferenzas entre as liñas 12 e 19. A primeira amosará o texto que contén a constante definida en XML (noutro ficheiro XML), a segunda amosa o texto directamente.
- Como xa se sabe, neste caso, a definición da constante **@string/hello_world** está no ficheiro de recursos: **/res/values/strings.xml**.




```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3
4     <string name="app_name">U2_07_TextView</string>
5     <string name="action_settings">Settings</string>
6     <string name="hello_world">Hello world!</string>
7
8 </resources>
```

- Como xa se dixo é recomendable usar o primeiro caso, pero no material vaise *abusar* do segundo caso para que se entendan os exemplos máis facilmente.
- No segundo caso, como xa se indicou, o IDE Eclipse dará unha advertencia na liña 19 porque recomenda que esa propiedade se declare a través dunha constante.

- As liñas 9 e 15 están creando o ID para os controis para que poidan ser accedidos dende Java a través da Clase R.
- Observar na imaxe a declaración deses IDs.

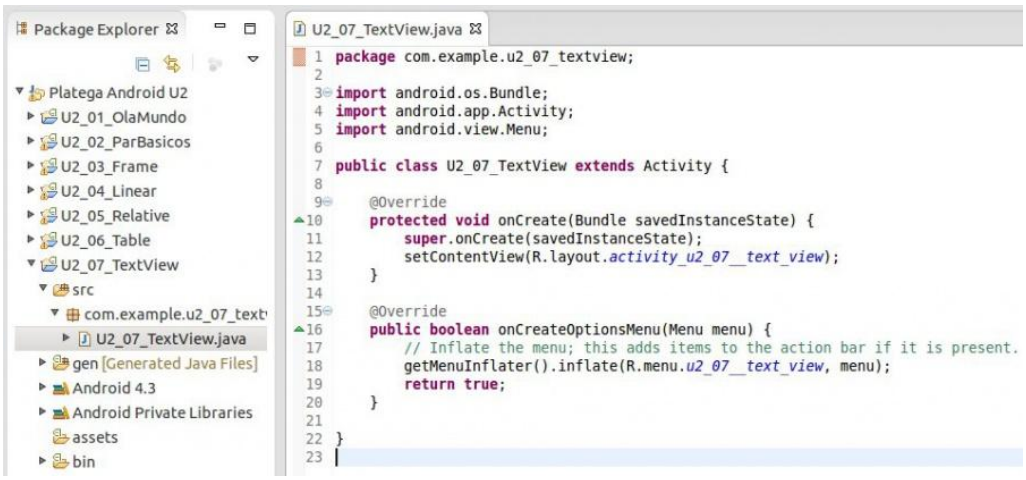


- Isto xerase automaticamente cando se ten **gardado o documento XML** onde se declaran os controis.

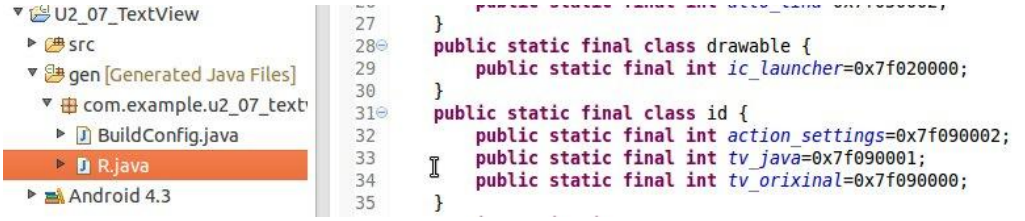
- Finalmente mirar como se define a cor azul (RGB), liña 18. Esta definición tamén podería estar usando unha constante declarada noutro ficheiro XML, como se verá ao final deste apartado.

Acceder e manipular o control dende Java

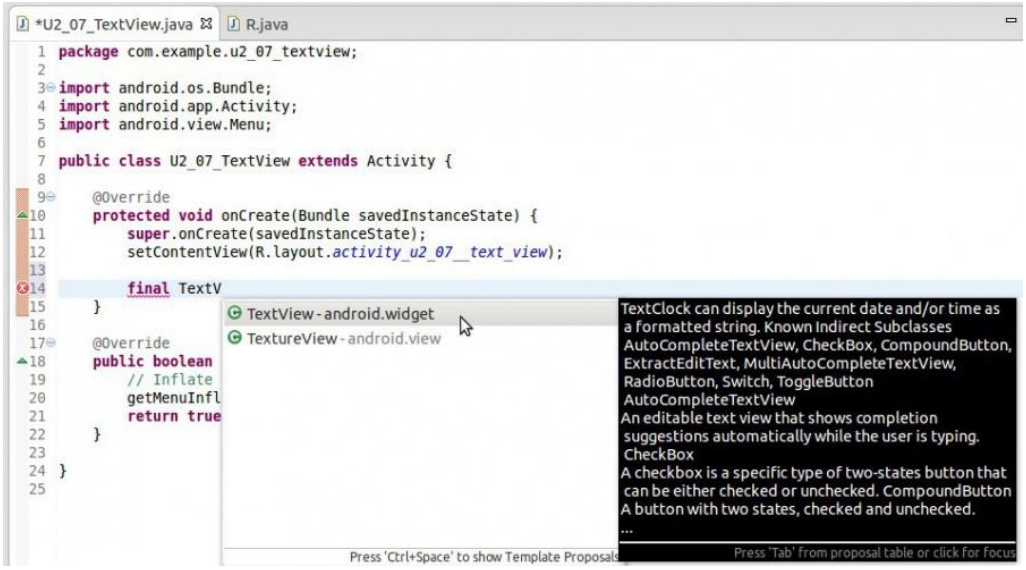
- A continuación imos acceder ao control EditText declarado en XML dende Java e imos realizar accións sobre o control.
- Para iso imos á clase Java que lanza a aplicación. (Por agora só hai unha clase Java).



- Como xa se indicou, podemos acceder aos elementos visuais declarados en XML a través do seu ID usando de intermediaria a clase R.



- Dentro do método **onCreate()** que é o encargado de lanzar a Activity, como se verá máis adiante.



- Comezamos a declarar un obxecto de tipo TextView.
- A medida que imos escribindo o tipo se prememos **CTRL+ Barra espaciadora** xa sabemos que nos vai a autocompletar, pero ademais vai **importar o paquete** que define onde está declarada a clase.

*U2_07_TextView.java | R.java

```
1  package com.example.u2_07_textview;
2
3  import android.os.Bundle;
4  import android.app.Activity;
5  import android.view.Menu;
6  import android.widget.TextView;
7
8  public class U2_07_TextView extends Activity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_u2_07__text_view);
14
15         final TextView
16     }
17
```

- Declaramos un obxecto de tipo TextView e queremos asocialo ao primeiro TextView do Layaout (ID:tv_orixinal).

```
8  public class U2_07_TextView extends Activity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_u2_07__text_view);
14
15         final TextView tvOrixinal = (TextView) findViewById(R.id.)
16     }
17
18     @Override
19     public boolean onCreateOptionsMenu(Menu menu) {
20         // Inflate the menu; this adds items to the action bar if
21         getMenuInflater().inflate(R.menu.u2_07__text_view, menu);
22         return true;
23     }
```

action_settings : int - R.id
tv_java : int - R.id
tv_orixinal : int - R.id
class : Class<com.example.u2_07_textview.R.id>
this

- Para iso precisamos o método **findViewById()**:
  - **Recibe como parámetro unha constante** da Clase R: neste caso o id asociado a un elemento visual. (CTRL+Barra espaciadora, para localizar a constante).
  - **Devolve un obxecto** de tipo **View**.
- Todo control visual é unha **subclase** de **View**, por tanto como obtemos un obxecto View é preciso facer un **casting (TextView)** para que faga unha conversión de tipos ao tipo desexado, neste caso dun **obxecto tipo View** a un **obxecto de tipo TextView**.

- A continuación amósase o código Java que vai manipular o TextView

```
1  package com.example.u2_07_textview;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5  import android.view.Menu;
6  import android.widget.TextView;
7
8  public class U2_07_TextView extends Activity {
9
10     @Override
11     protected void onCreate(Bundle savedInstanceState) {
12         super.onCreate(savedInstanceState);
13         setContentView(R.layout.activity_u2_07__text_view);
14
15         final TextView tvOrixinal = (TextView) findViewById(R.id.tv_orixinal);
16         final TextView tvJava = (TextView) findViewById(R.id.tv_java);
17
18         tvJava.setText("Etiqueta modificada en java: "+tvOrixinal.getText());
19     }
20
21     @Override
22     public boolean onCreateOptionsMenu(Menu menu) {
23         // Inflate the menu; this adds items to the action bar if it is present.
24         getMenuInflater().inflate(R.menu.u2_07__text_view, menu);
25         return true;
```

```
26      }
27  }
```

- Liña 15: temos un obxecto que apunta a TextView orixinal, ao primeiro TextView do Layout.
- Liña 16: temos un obxecto que apunta ao novo TextView, ao segundo do Layout.
- Liña 18: modificamos o texto do segundo TextView. O contido é unha cadea de texto concatenada (+) co texto que ten o primeiro TextView.
  - Observar a función dos métodos: **setText()** e **getText()**. No seguinte enlace pódense ver estes métodos públicos: http://developer.android.com/reference/android/widget/TextView.html#pubmethods.
- Cando se lance a aplicación vaise executar ese código, co cal o texto do segundo TextView non vai ser o que se indicou no Layout en tempo de deseño senón o que se indica en tempo de execución.
- A imaxe amosa a execución da aplicación:

- Observar o contido da segunda liña, non é o que se asignou no Layout.

Manipulación html dunha etiqueta de texto

- As Etiquetas de texto non son texto (String) senón que son código semellante ao html.
- Entón na liña 18 anterior parece que hai unha contradición: concatenar unha cadea con *algo* html.
- Teríamos que ter usado o método **toString()**: **tvOrixinal.getText().toString()**
- Pero en Java non fai falla poñelo porque ese método chámase automaticamente sempre que o obxecto se concatene con outro String (Neste caso a cadea de texto).

- Para obter a seguinte imaxe non se vai tocar o XML e vaise facer todo en Java.

- Na segunda etiqueta cambiouse: cor, tamaño da fonte e púxose en **negriña** unha palabra.

- Código JAVA ...

```java
package com.example.u2_07_textview;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.text.Html;
import android.view.Menu;
import android.widget.TextView;

public class U2_07_TextView extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u2_07__text_view);

        final TextView tvOrixinal = (TextView) findViewById(R.id.tv_orixinal);
        final TextView tvJava = (TextView) findViewById(R.id.tv_java);

        tvJava.setText("Etiqueta modificada en java: "+tvOrixinal.getText());
        tvJava.append(". Máis texto. ");
        tvJava.setTextColor(Color.RED);
        tvJava.setTextSize(20);
        tvJava.append(Html.fromHtml("<p><br>Isto está en <b>negriña</b> e noutra liña.</p>"));
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.u2_07__text_view, menu);
        return true;
    }
}
```

- Para importar os paquetes premer (SHIFT + CTRL + O)

- Liña 21: engade contido á etiqueta polo final da mesma e amosa o contido final.
- Liña 22: cambiar a cor usando unha constante estática.
- Liña 23: cambiar o tamaño da fonte
- Liña 24: devolve a cadea en formato HTML no que en Android se podería chamar *Texto con estilo mostrable*

- A continuación vaise modificar a aplicación para que poidamos recuperar o contido exacto dun *TextView*, non o que amosa por pantalla.
- Tamén se vai usar o método **toString()** para ver o seu resultado.

- A imaxe amosa o deseño do Layout

- Observar a liña de separación e despois desta, dúas etiquetas de texto.

- O Ficheiro XML asociado

```
1 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:orientation="vertical"
6     android:padding="20sp" >
7
8     <TextView
9         android:id="@+id/tv_orixinal"
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content"
12        android:text="@string/hello_world" />
13
14    <TextView
15        android:id="@+id/tv_java"
16        android:layout_width="wrap_content"
17        android:layout_height="wrap_content"
18        android:text="Dá igual o que escribamos aquí. Neste exemplo, ao lanzar a aplicación, vaise cambiar este texto"
19        android:textColor="#00F" />
20
21    <View
22        android:layout_width="match_parent"
23        android:layout_height="2sp"
24        android:background="#000" />
25
26    <TextView
27        android:id="@+id/tv_html"
28        android:layout_width="wrap_content"
29        android:layout_height="wrap_content"
```

```
        android:text="Cadena html do tv_java. Realizarase en java" />

    <TextView
        android:id="@+id/tv_string"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="String do tv_java. Realizarase en java"
        android:textColor="#00F"
        android:textSize="16sp" />

</LinearLayout>
```

- Observar as liñas marcadas
- A imaxe amosa a aplicación en execución:

- Observar como a primeira etiqueta despois da liña ten o contido do que almacena o EditText vermello.
- A segunda etiqueta despois da liña amosa a etiqueta vermella pasada polo método **toString**. Observar como xa non hai negriña.

- O código que fai iso posible é o seguinte:

```
package com.example.u2_07_textview;

import android.R.string;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.text.Html;
import android.text.Spanned;
import android.view.Menu;
import android.widget.TextView;

public class U2_07_TextView extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u2_07__text_view);

        final TextView tvOrixinal = (TextView) findViewById(R.id.tv_orixinal);
        final TextView tvJava = (TextView) findViewById(R.id.tv_java);
        final TextView tvHtml = (TextView) findViewById(R.id.tv_html);
```

```
22          final TextView tvString = (TextView) findViewById(R.id.tv_string);
23
24          tvJava.setText("Etiqueta modificada en java: "+tvOrixinal.getText());
25
26          //tvJava.setText(tvJava.getText()+". Máis texto. ");
27          tvJava.append(". Máis texto. ");
28          tvJava.setTextColor(Color.RED);
29          tvJava.setTextSize(20);
30          tvJava.append(Html.fromHtml("<p><br>Isto está en <b>negriña</b> e noutra liña.</p>"));
31
32
33          tvHtml.setText(Html.toHtml((Spanned)(tvJava.getText())));
34
35          //tvString.setText(""+tvJava.getText());
36          tvString.setText(tvJava.getText().toString());
37
38      }
39
40      @Override
41      public boolean onCreateOptionsMenu(Menu menu) {
42          // Inflate the menu; this adds items to the action bar if it is present.
43          getMenuInflater().inflate(R.menu.u2_07__text_view, menu);
44          return true;
45      }
46  }
```

- Liña 33: colle o valor da etiqueta vermella e a pasa a formato Html.
- Liña 36: pasa a cadea vermella a String. Sería o equivalente á liña 35 (Concatenar cunha cadea).

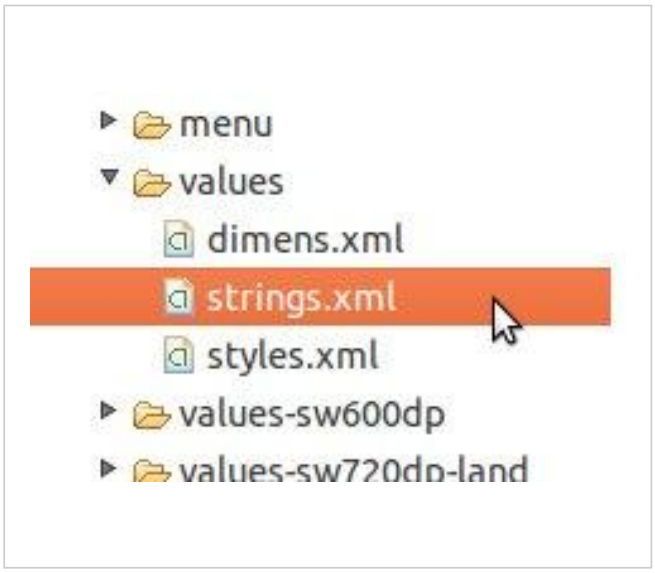
Definición de constantes/recursos xml

- Como xa vimos na definición XML do Layout temos valores postos á *ferro*.
- Sería bo definir esas propiedades noutros ficheiros XML, deste xeito permítese a reutilización e a internacionalización.

Recursos string

Advertencia no XML indicando que ...

usemos un recurso de tipo @string.



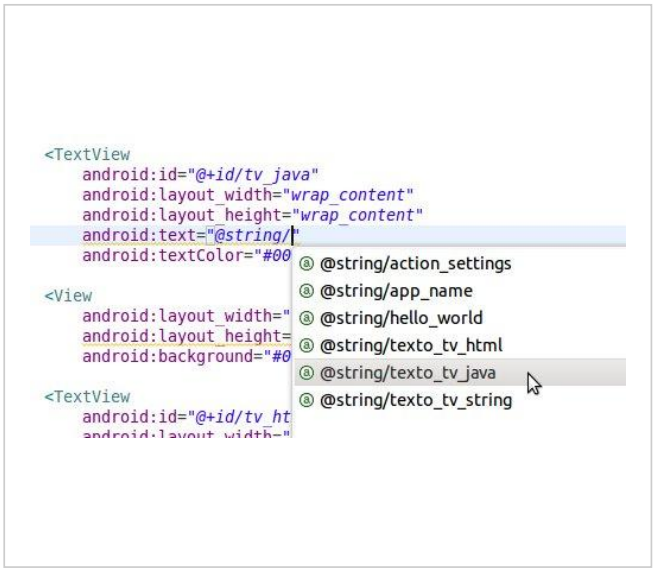
Editamos o ficheiro **res/values/strings.xml** ou creamos un novo en **/res/values**

- Engadimos os novos recurso de tipo string. Observar as liñas marcadas.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3
4     <string name="app_name">U2_07_TextView</string>
5     <string name="action_settings">Settings</string>
6     <string name="hello_world">Hello world!</string>
7
8     <string name="texto_tv_java">Dá igual o que escribamos aquí. Neste exemplo, ao lanzar a aplicación, vaise cambiar este texto</string>
9     <string name="texto_tv_html">Cadena html do tv_java. Realizarase en java</string>
10    <string name="texto_tv_string">String do tv_java. Realizarase en java</string>
11
12 </resources>
```

- Gardar o ficheiro

Uso recursos string



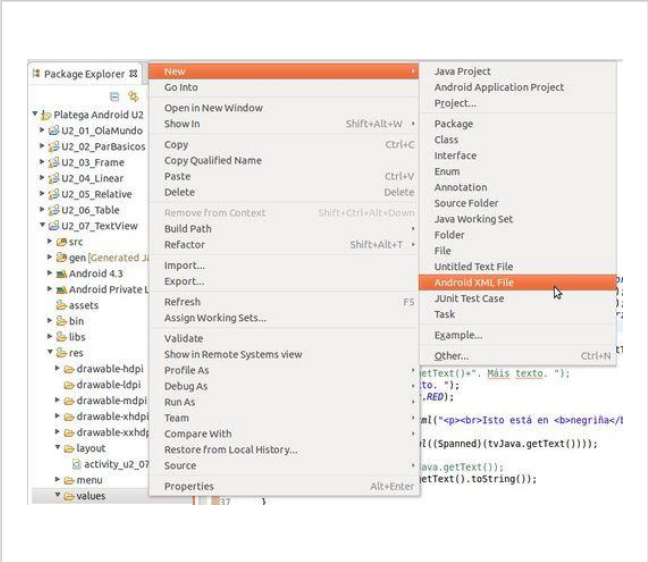
No layout facemos uso dos recursos anteriores: (CTRL + Barra espaciadora). Lembrar gardar antes o ficheiro de recursos anterior.

Xa quedan menos advertencias: A da dimensión podería declararse no ficheiro **/res/values/dimens.xml** ou crear un novo.

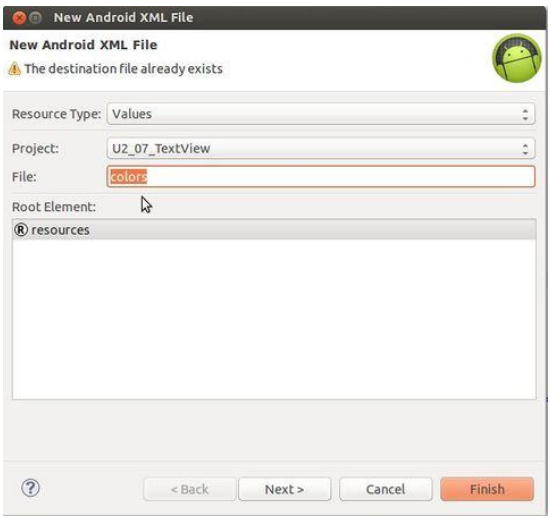
```
<resources>

    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
    <dimen name="alto_liña">2sp</dimen>

</resources>
```

- A continuación vaise crear un ficheiro de recurso para as cores.

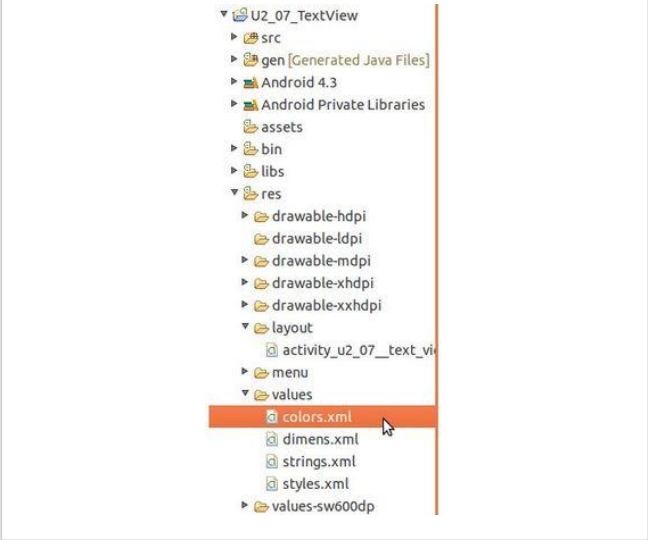
Crear ficheiro de recursos



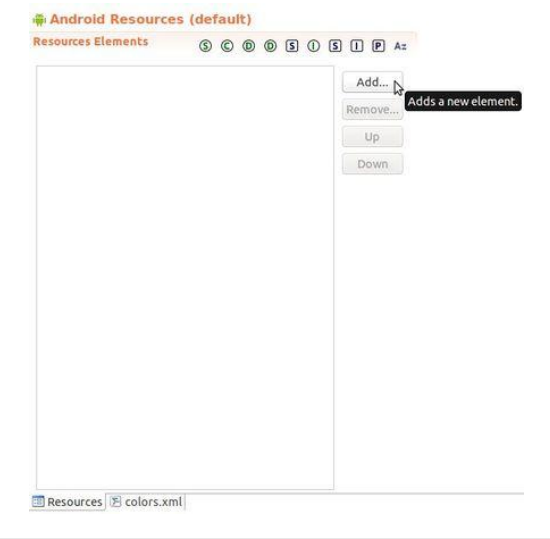
En **/res/values** creamos un novo ficheiro **Android XML File**



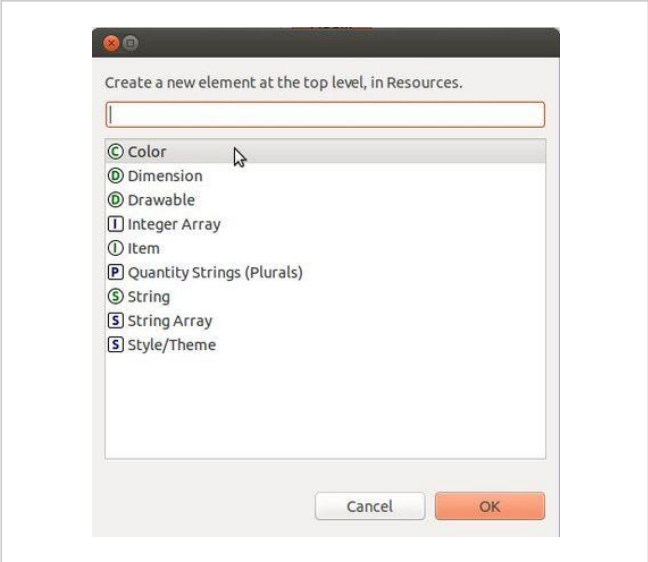
Indicamos un nome significativo, vale calquera. Neste caso *colors* e gardamos.



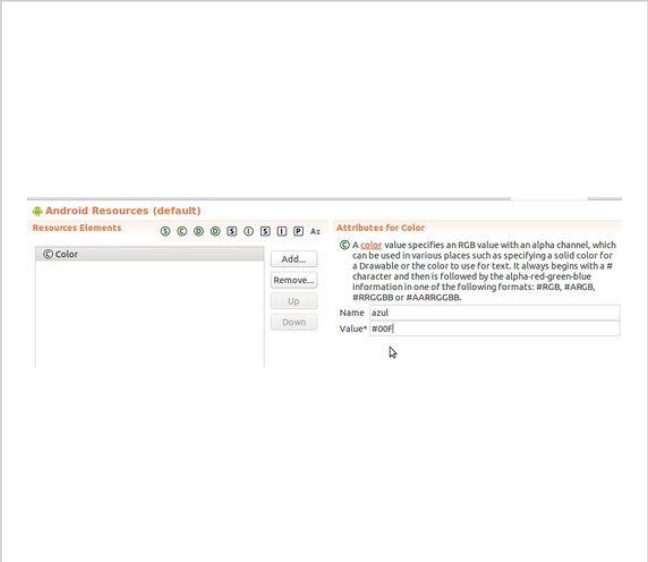
Imos ao ficheiro



Engadimos un elemento



Seleccionar **Color** e aceptar.



Engadimos a definición dunha cor graficamente ...

... en modo texto.

- Ficheiro de recurso de cores:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="azul">#00F</color>
    <color name="negro">#000</color>

</resources>
```

- Uso dos recursos no layout. Como sempre, observar as liñas marcadas

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="20sp" >

    <TextView
        android:id="@+id/tv_orixinal"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
12          android:text="@string/hello_world" />
13
14      <TextView
15          android:id="@+id/tv_java"
16          android:layout_width="wrap_content"
17          android:layout_height="wrap_content"
18          android:text="@string/texto_tv_java"
19          android:textColor="@color/azul" />
20
21      <View
22          android:layout_width="match_parent"
23          android:layout_height="@dimen/alto_liña"
24          android:background="@color/negro" />
25
26      <TextView
27          android:id="@+id/tv_html"
28          android:layout_width="wrap_content"
29          android:layout_height="wrap_content"
30          android:text="@string/texto_tv_html" />
31
32      <TextView
33          android:id="@+id/tv_string"
34          android:layout_width="wrap_content"
35          android:layout_height="wrap_content"
36          android:textColor="@color/azul"
37          android:textSize="16sp"
38          android:text="@string/texto_tv_string"/>
39
40  </LinearLayout>
```

- Anda!!! Quedounos na liña 37 un valor candidato a ser definido nunha constante tipo *dimen*. Seguro que o participante no curso é quen de facelo.

-- Ángel D. Fernández González e Carlos Carrión Álvarez -- (2015).
