

3. LAYOUTS EN ANDROID

Objetivos

- Veremos el funcionamiento de setContentView. MainActivity.java y activity_main.xml son los dos principales ficheros en los que se basa una Activity o ventana de Android. La Api setContentView permite la carga de la vista o layout de la Activity desde el evento onCreate de MainActivity.
- También estudiaremos el uso del fichero strings.xml para definir los textos utilizados en nuestra aplicación y que introduciremos en los layouts para el diseño de los interfaces
- Y por último veremos los diferentes tipos de layouts xml que utiliza Android, de los cuales los mas importantes son el LinearLayout y el ConstraintLayout

3. LAYOUTS EN ANDROID

Paso 1. Si creamos una nueva aplicación tipo Empty View Activity, nos crea los dos componentes principales de una actividad de Android:

- MainActivity.java: Gestor controles en java
- activity_main.xml: Layout para el diseño de la interfaz de la aplicacion

```
MainActivity.java x
1 package com.example.myapplication;
2
3 import ...
6
7 public class MainActivity extends AppCompatActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.activity_main);
13    }
14 }
```

```
activity_main.xml x
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9
10    <TextView
11        android:layout_width="wrap_content"
12        android:layout_height="wrap_content"
13        android:text="Hello World!"
14        app:layout_constraintBottom_toBottomOf="parent"
15        app:layout_constraintEnd_toEndOf="parent"
16        app:layout_constraintStart_toStartOf="parent"
17        app:layout_constraintTop_toTopOf="parent" />
18
19 </androidx.constraintlayout.widget.ConstraintLayout>
```

R.layout.activity_main corresponde a un objeto View, creado en tiempo de ejecución a partir del recurso *activity_main.xml*. Esta forma de trabajar requiere menos memoria.

3. LAYOUTS EN ANDROID

API setContentView

- Permite cargar la vista de la actividad de Android. Sólo se pueden enlazar los controles del layout mediante findViewById una vez que se ha cargado el layout con setContentView. Si se hace antes dará error:

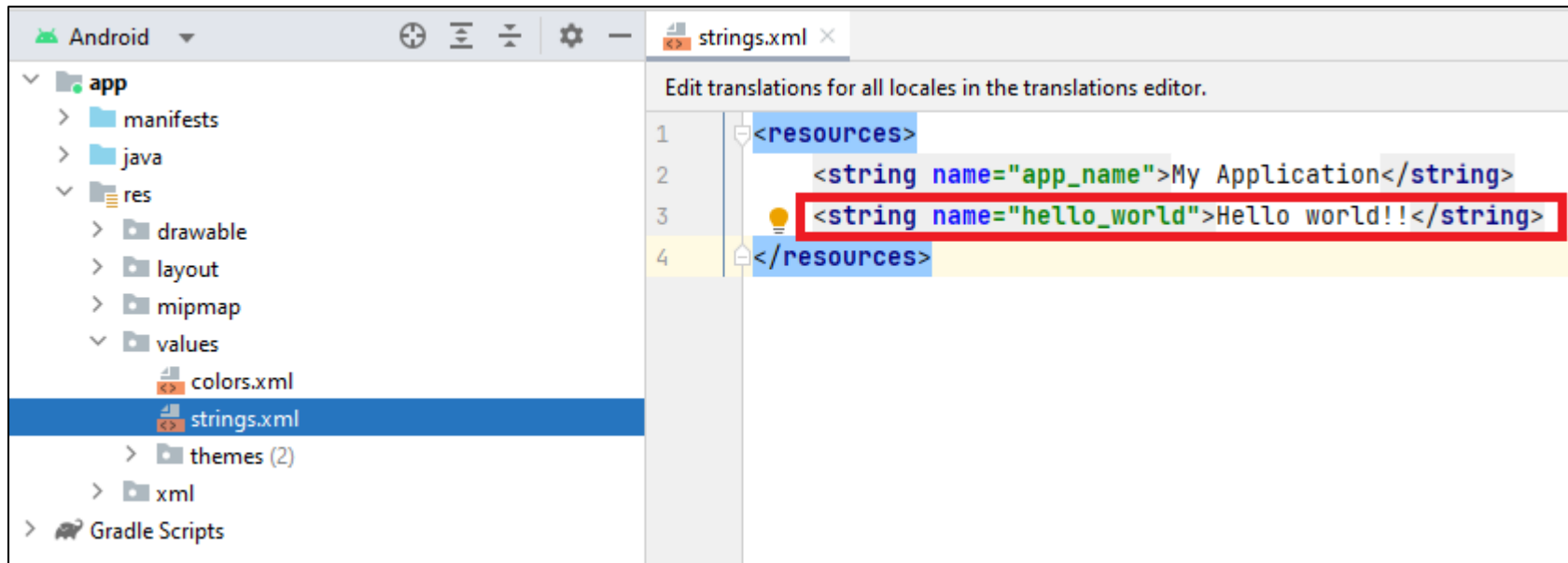
```
public class MainActivity extends AppCompatActivity {  
    2 usages  
    private Button button;  
    4 usages  
    private EditText editText1;  
    2 usages  
    private EditText editText2;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        button = (Button) findViewById(R.id.button1);  
        editText1 = (EditText) findViewById(R.id.editText1);  
        editText2 = (EditText) findViewById(R.id.editText2);  
    }  
}
```

Orden correcto:

- 1) Declaración de las variables java que representan objetos de vista
- 2) Carga de la vista
- 3) Enlazado de las variables java con los controles xml

3. LAYOUTS EN ANDROID

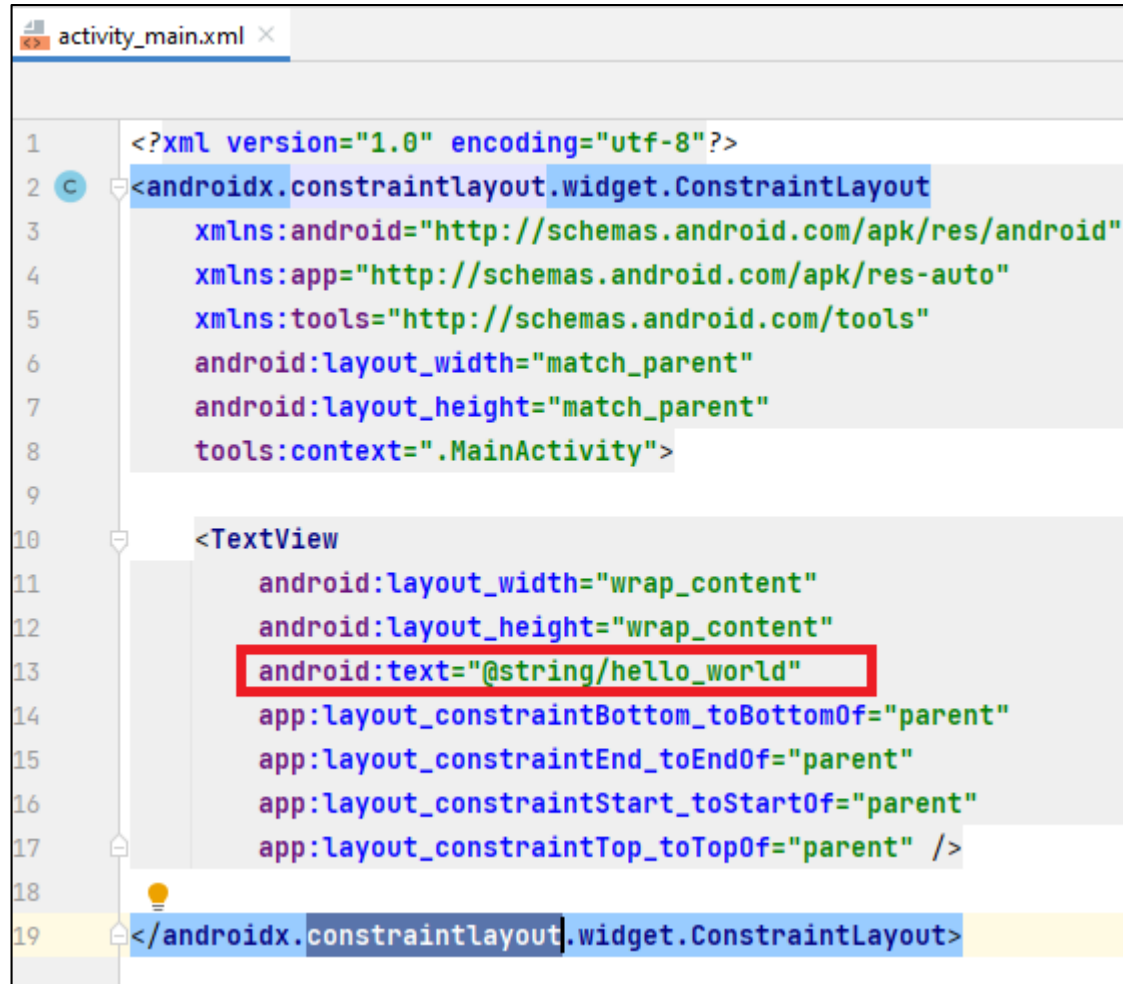
Paso 2. Crea una variable en el fichero res/values/strings.xml de nombre hello_world. Su valor de referencia será @string/hello_world



El fichero strings.xml facilita la organización y localización de los mensajes de texto que contiene nuestra aplicación.

3. LAYOUTS EN ANDROID

Paso 3. En activity_main.xml sustituye el texto “hello world!” por la variable de referencia “@string/hello_world”. Y ejecuta la aplicación.



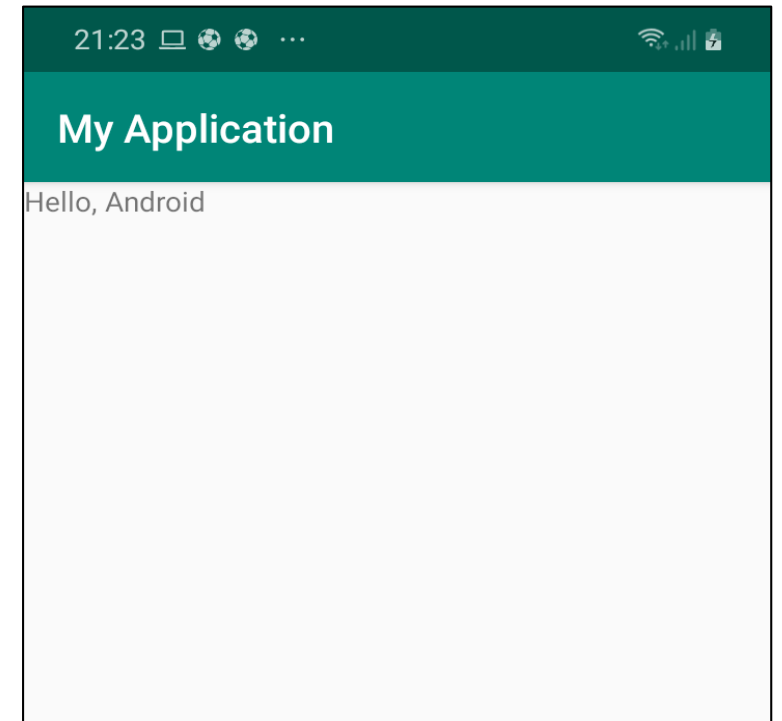
```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      tools:context=".MainActivity">
9
10     <TextView
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:text="@string/hello_world"
14         app:layout_constraintBottom_toBottomOf="parent"
15         app:layout_constraintEnd_toEndOf="parent"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19 </androidx.constraintlayout.widget.ConstraintLayout>
```

Nota: Esta es la práctica recomendada en Android para la inserción de textos en nuestra aplicación. En el layout no pondremos el texto directo, sino variables referenciadas cuyo contenido estará en el fichero strings.xml

3. LAYOUTS EN ANDROID

Paso 4. Comenta la última sentencia del método onCreate() y añade las tres que se muestran a continuación. Ejecuta la aplicación

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        //setContentView(R.layout.activity_main);  
        TextView texto = new TextView(context: this);  
        texto.setText("Hello, Android");  
        setContentView(texto);  
    }  
}
```

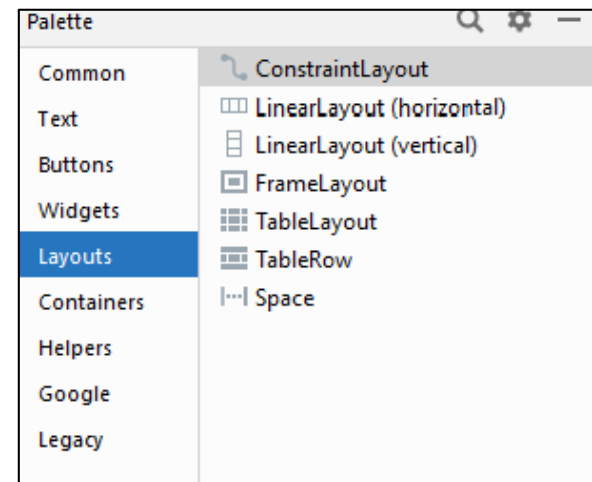


NOTA: setContentView puede cargar vistas simples como un TextView, o button, o layouts en formato xml, que combinan varios elementos de tipo vista

3. LAYOUTS EN ANDROID

Layout: Es un contenedor de una o más vistas y controla su comportamiento y posición. Un *Layout* puede contener a otro *Layout* y es un descendiente de la clase *View*. *Layouts* más utilizados en Android:

- **RelativeLayout:** Dispone elementos en relación a otro o al padre (Deprec)
- **AbsoluteLayout:** Posiciona los elementos de forma absoluta (Deprecated)
- **LinearLayout:** Dispone los elementos en una fila o columna.
- **TableLayout + TableRow:** Distribuye los elementos de forma tabular.
- **FrameLayout:** Permite el cambio dinámico de los elementos que contiene
- **ConstraintLayout:** Versión mejorada de RelativeLayout, que permite una edición visual desde el editor y trabajar con porcentajes



3. LAYOUTS EN ANDROID

LinearLayout Vertical/horizontal

Layout más utilizado y muy simple. Distribuye los elementos uno detrás de otro, bien de forma horizontal o vertical.

```
activity_main.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_height="match_parent"
5     android:layout_width="match_parent"
6     android:orientation="vertical">
7
8     <AnalogClock
9         android:layout_width="wrap_content"
10        android:layout_height="wrap_content"/>
11
12    <CheckBox
13        android:layout_width="wrap_content"
14        android:layout_height="wrap_content"
15        android:text="Un checkBox"/>
16
17    <Button
18        android:layout_width="wrap_content"
19        android:layout_height="wrap_content"
20        android:text="Un botón"/>
21
22    <TextView
23        android:layout_width="wrap_content"
24        android:layout_height="wrap_content"
25        android:text="Un texto cualquiera"/>
26 </LinearLayout>
```



3. LAYOUTS EN ANDROID

TableLayout + TableRow

Distribuye los elementos de forma tabular. Se utiliza la etiqueta *TableRow* cada vez que queremos insertar una nueva línea.

```
activity_main.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <TableLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:layout_height="match_parent"
5      android:layout_width="match_parent">
6      <TableRow>
7          <AnalogClock
8              android:layout_width="wrap_content"
9              android:layout_height="wrap_content"/>
10         <CheckBox
11             android:layout_width="wrap_content"
12             android:layout_height="wrap_content"
13             android:text="Un checkBox"/>
14     </TableRow>
15     <TableRow>
16         <Button
17             android:layout_width="wrap_content"
18             android:layout_height="wrap_content"
19             android:text="Un botón"/>
20         <TextView
21             android:layout_width="wrap_content"
22             android:layout_height="wrap_content"
23             android:text="Un texto cualquiera"/>
24     </TableRow>
25 </TableLayout>
```

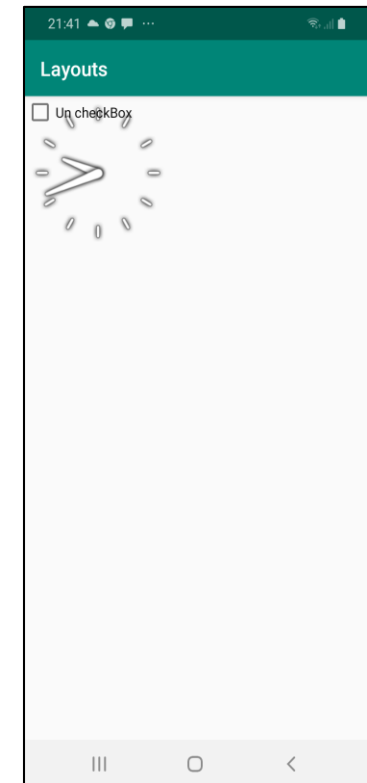


3. LAYOUTS EN ANDROID

FrameLayout

Posiciona todos los elementos usando todo el contenedor, sin distribuirlos espacialmente. Se usa cuando queremos que varias vistas ocupen un mismo lugar. Podemos hacer que solo una sea visible (con la propiedad visibility), o superponerlas.

```
activity_main.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <FrameLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     android:layout_height="match_parent"
5     android:layout_width="match_parent">
6     <AnalogClock
7         android:layout_width="wrap_content"
8         android:layout_height="wrap_content"/>
9     <CheckBox
10         android:layout_width="wrap_content"
11         android:layout_height="wrap_content"
12         android:text="Un checkBox"/>
13     <Button
14         android:layout_width="wrap_content"
15         android:layout_height="wrap_content"
16         android:text="Un botón"
17         android:visibility="invisible"/>
18     <TextView
19         android:layout_width="wrap_content"
20         android:layout_height="wrap_content"
21         android:text="Un texto cualquiera"
22         android:visibility="invisible"/>
23 </FrameLayout>
24
```



3. LAYOUTS EN ANDROID

ConstraintLayout

Versión más flexible y eficiente de RelativeLayout. Layout por defecto Android

```
activity_main.xml x
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     tools:context=".MainActivity">
9     <AnalogClock
10         android:id="@+id/AnalogClock01"
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         app:layout_constraintLeft_toLeftOf="parent"
14         app:layout_constraintTop_toTopOf="parent"/>
15     <CheckBox
16         android:id="@+id/checkbox"
17         android:layout_width="wrap_content"
18         android:layout_height="wrap_content"
19         android:text="CheckBox"
20         tools:ignore="MissingConstraints" />
21     <Button
22         android:id="@+id/button"
23         android:layout_width="wrap_content"
24         android:layout_height="wrap_content"
25         android:text="Button"
26         tools:ignore="MissingConstraints" />
27     <TextView
28         android:id="@+id/textView"
29         android:layout_width="wrap_content"
30         android:layout_height="wrap_content"
31         android:text="TextView"
32         tools:ignore="MissingConstraints" />
33 </androidx.constraintlayout.widget.ConstraintLayout>
```

