Recursos (1)

Los recursos son archivos o datos externos utilizados por nuestra aplicación Android: imágenes, vistas, estilos, *strings*, colores…

Los recursos se encuentran en la carpeta del proyecto "*src/main/res*", agrupados en carpetas por tipo de recurso.

Para acceder a un recurso desde nuestra aplicación se utiliza la clase R seguida del tipo y el identificador del recurso concreto:

```
R.layout.activity_main
```

53

Recursos (2)

54

Recursos (3)

La externalización de los recursos de Android ofrece dos ventajas importantes:

- Permite mantener y actualizar dichos recursos de forma independiente al código de la aplicación.
- Permite suministrar recursos alternativos que admiten configuraciones específicas de los dispositivos como idiomas o tamaños de pantalla.

55

Recursos (4)

Pueden proporcionarse recursos alternativos en función de varias características (idioma, tamaño de pantalla, hardware, etc).

Para ello las carpetas que los contienen incluirán cualificadores separados por guiones:

- Código del país del dispositivo y operador (mccAAA-mncAAA): mcc214 para España
- Idioma y región: es-rES, en-rGB, en-rUS…
- Orientación pantalla: horizontal (*land*) o vertical (*port*).

56

Recursos (5)

- Tamaño pantalla:
  small, *medium*, *large* o *xlarge*.
- Alto disponible (h<dimension>dp):
  h720dp, h1024dp …
- Ancho disponible (w<dimension>dp):
  w720dp, w1024dp …
- Ancho mínimo (sw<dimension>dp):
  sw320dp, sw600dp, sw720dp …
- Versión Android (nivel API):
  v9, v14, v21 …

57

Recursos (6)

Se pueden utilizar varios cualificadores en el tipo del recurso, pero tienen que establecerse en un orden determinado, de lo contrario se ignorarán:

https://developer.android.com/guide/topics/resources/providing-resources.html#table2

Ejemplo:

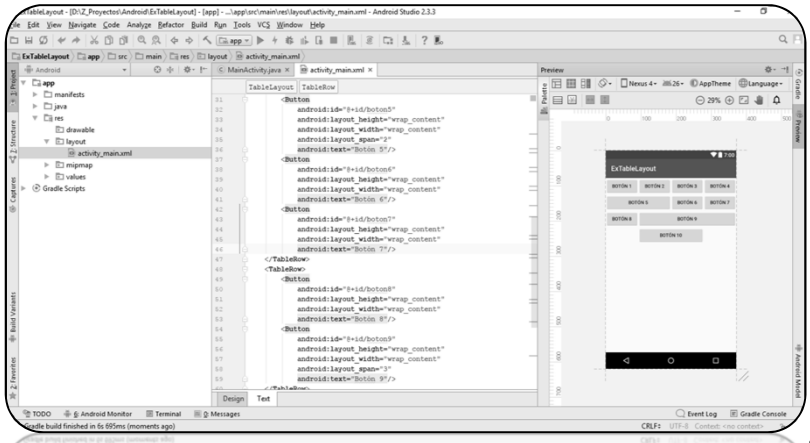
drawable-en-rUS-land

Recurso gráfico, ingles (USA), horizontal

58

Recursos (7)

- ***layout***: archivos XML que definen el diseño de una interfaz de usuario.



59

Recursos (8)

- ***drawable***: recursos gráficos para mostrar en pantalla, generalmente imagen.

  Carpetas: drawable-xxx
  - ldpi: resolución baja (120 dpi)
  - mdpi: resolución media (160 dpi)
  - hdpi: resolución alta (240 dpi)
  - xhdpi: resolución extra-alta (320 dpi)
  - xxhdpi: resolución extra-extra-alta (480 dpi)
  - xxxhdpi: resolución extra-extra-extra-alta (640 dpi)
  - nodpi: recursos no dependientes de la resolución
  - tvdpi: resoluciones asociadas a *Smart TVs*

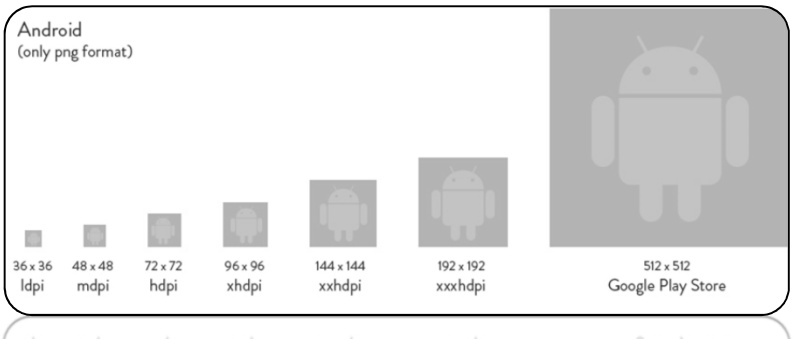
60

Recursos (9)

- ***mipmap***: icono de la aplicación.

Los cualificadores en función de la densidad de pantalla coinciden con los de *drawable*.

Carpetas: mipmap-xxx



61

Recursos (10)

- ***values***: archivos XML que contienen diferentes valores (constantes) que se utilizarán en la aplicación. Suelen agruparse en los siguientes archivos:

- strings.xml: cadenas.
- arrays.xml: listas de elementos.
- colors.xml: colores (hexadecimal).
- dimens.xml: dimensiones de márgenes, *padding*, tamaños, etc.
- styles.xml: estilos de interfaz.

62

Recursos (11)

Cadenas de caracteres

La creación de constantes de tipo *string* para utilizarlas dentro de las aplicaciones es muy común en Android.

Aunque se puede utilizar cualquier nombre de archivo (siempre dentro de "*values*"), habitualmente se almacenan en el archivo "*strings.xml*".

63

Recursos (12)

Cadenas de caracteres

- Crear una cadena de caracteres

```
<resources>
    <string name="app_name">HolaAndroid</string>
</resources>
```

- Acceder a una cadena desde Java
  R.string.nombre_string
- Acceder a una cadena desde XML
  @string/nombre_string

64

Recursos (13)

Cadenas de caracteres

- Obtener el contenido

En primer lugar obtendremos los recursos del sistema mediante *getResources()*.

A continuación, sobre los recursos obtenidos llamaremos al método *getString()* con el identificador de la cadena como parámetro.

```
Resources res = getResources();
String titulo = res.getString(R.string.title);
```

65

Recursos (14)

Cadenas de caracteres

Podemos incluir caracteres especiales como apóstrofos (') utilizando comillas para toda la cadena:

```
<string name="app_name">"Android'2020"</string>
```

O colocar el carácter de escape (\) delante:

```
<string name="app_name">Android\'2020</string>
```

66

Recursos (15)

Cadenas de caracteres

También se pueden incluir argumentos dentro de las cadenas:

```
<string name="mensajes">
 %1$s ha recibido %2$d mensajes nuevos.</string>
```

Y desde la aplicación establecer su valor mediante la función *format*:

```
Resources res = getResources();
String mensaje =
  String.format(res.getString(R.string.mensajes),
  nombre, numMensajes);
```

67

Recursos (16)

Internacionalización

Una de las principales características de Android es que al publicar nuestras aplicaciones en Google Play pueden ser descargadas desde cualquier lugar del mundo, por lo que resulta muy interesante facilitar su uso a usuarios de otros países añadiendo traducciones de los diferentes elementos (menús, textos, etc) mostrados en pantalla.

68

Recursos (17)

Internacionalización

Cuando se ejecuta una aplicación se utiliza el idioma del teléfono, y si éste no está disponible el idioma por defecto (carpeta *values*), habitualmente inglés.

Si queremos que nuestra aplicación esté disponible en otros idiomas basta con crear traducciones de todas las cadenas ubicadas en carpetas *values* para cada idioma: "*values-es*", "*values-fr"*, etc.

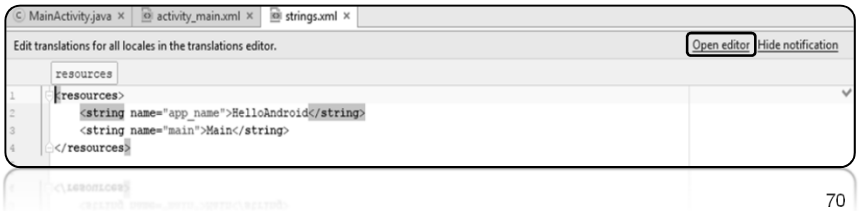
69

Recursos (18)

Internacionalización

Android Studio incorpora una herramienta para facilitar el proceso de traducción.

Para acceder a ella abrimos el archivo que queremos traducir y en la parte superior derecha seleccionamos "*Open editor*".

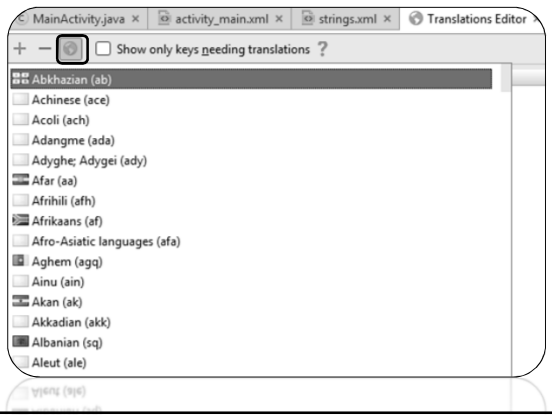


70

Recursos (19)

Internacionalización

En primer lugar seleccionamos los idiomas que queremos añadir:

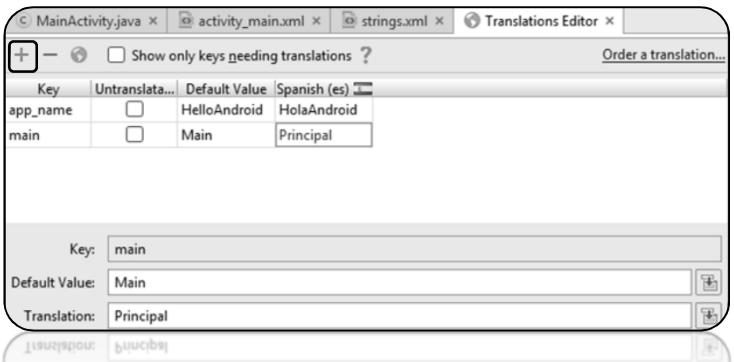


71

Recursos (20)

Internacionalización

Y luego simplemente añadimos los *string* que vayamos a utilizar y su traducción a cada idioma seleccionado.

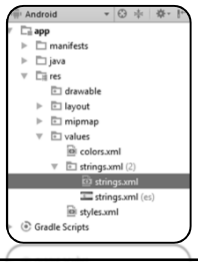


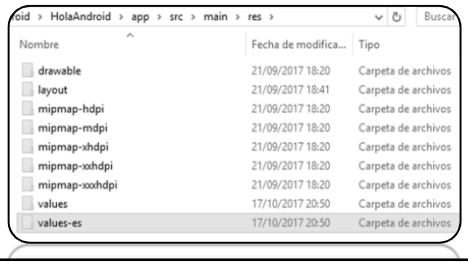
72

Recursos (21)

Internacionalización

Veremos que en nuestro proyecto aparece un nuevo archivo *strings* por cada idioma seleccionado y en la carpeta "res" de nuestro proyecto aparecerá "values-xx" correspondiente al idioma seleccionado:





73

Recursos (22)

Listas

Mediante "*string-array*" podemos almacenar listas de elementos para utilizarlas en menús, listas, desplegables, etc.

Habitualmente se almacenan en el archivo "*arrays.xml*".

```
<string-array name="workdays">
    <item>Monday</item>
    <item>Tuesday</item>
    <item>Wednesday</item>
    <item>Thursday</item>
    <item>Friday</item>
</string-array>
```

Recursos (23)

Listas

Para recuperar los valores procederemos de forma similar a los string pero utilizando el método *getStringArray()*.

```
Resources res = getResources();
String[] days = res.getStringArray(R.array.workdays);
```

75

Recursos (24)

Colores

Mediante "*color*" podemos definir los colores que se utilizan en la aplicación para facilitar su utilización y modificación.

Suelen almacenarse en "*colors.xml*".

La definición de colores puede realizarse en varios formatos: #RGB, #RRGGBB, #ARGB y #AARRGGBB.

A (transparente), R (rojo), G (verde) y B (azul).

76

Recursos (25)

Colores

```
<resources>
    <color name="red">#F00</color>
</resources>
```

77

Recursos (26)

Dimensiones

Mediante "*dimen*" podemos definir valores para los tamaños o espacios de diferentes elementos.

Suelen almacenarse en "*dimens.xml*".

```
<resources>
    <dimen name="ancho_boton">60dp</dimen>
    <dimen name="alto_boton">30dp</dimen>
</resources>
```

78

Recursos (27)

Dimensiones

Estos valores se suelen utilizar en la definición de la interfaz.

```
<Button
    android:layout_width="@dimen/ancho_boton"
    android:layout_height="@dimen/alto_boton"
    android:text="@string/btnOk" />
```

Podemos utilizar dos tipos de unidades:

- dp: tamaño elementos, margen, padding…
- sp: tamaño de fuentes

79

Recursos (28)

Dimensiones

Podemos establecer tamaños diferentes en función del tamaño de pantalla creando distintas carpetas.

Por ejemplo mediante un "*dimens.xml*" dentro de la carpeta "*values-large*" para pantallas grandes en el que definamos un tamaño de letra mayor para las tabletas.

80

Recursos (29)

Identificador de recurso

Cada recurso utilizado en la aplicación tiene un identificador que permite referenciarlo.

Podemos ver el identificador de cada recurso si accedemos a la clase "R":

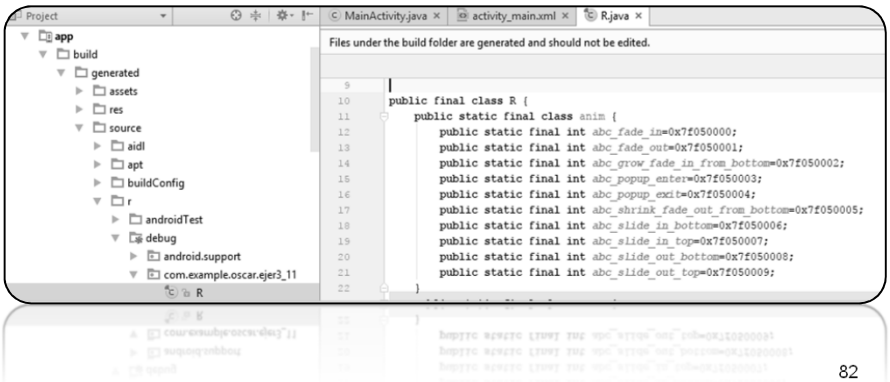
app → build → generated → source → r → debug → Nombre_paquete → R

81

Recursos (30)

Identificador de recurso

Como podemos apreciar cada recurso es identificado mediante un número entero.



82

Recursos (31)

Identificador de recurso

En ocasiones deberemos obtener este identificador para referirnos a un recurso determinado.

Para ello hemos de obtener el contexto de la aplicación mediante *getApplicationContext()* que devolverá un objeto de tipo *Context,* y a partir de él con el método *getIdentifier()* podremos obtener el identificador de cualquier recurso.

83

Recursos (32)

Identificador de recurso

El método *getIdentifier()* recibe como parámetros 3 strings:

- Nombre del recurso (sin extensión).
- Tipo de recurso: *"drawable"*, *"layout"*, etc.
- Paquete.

También se puede incluir el tipo de recurso y/o el paquete en el primer parámetro y establecer a *null* dichos parámetros.

84

Recursos (33)

Identificador de recurso

```
String imageName = "paisaje";
Context c = getApplicationContext();
int id = c.getResources().getIdentifier
    (imageName, "drawable" , c.getPackageName());
```

```
String imageName = "drawable/paisaje";
Context c = getApplicationContext();
int id = c.getResources().getIdentifier
    (imageName, null, c.getPackageName());
```

```
String imageName = "com.acme.miapp:drawable/paisaje";
Context c = getApplicationContext();
int id = c.getResources().getIdentifier
    (imageName, null, null);
```