Tipos de Recursos y Recursos del Sistema

Mª Alejandra Tomás Fernández

Recursos de la aplicación (I)

- La **definición de los recursos en Android** es un aspecto muy importante en el diseño de una aplicación.
- Una de sus **principales ventajas** es que facilita a los diseñadores gráficos e introductores de contenido trabajar en paralelo con los programadores.
- **Añadir un recurso** a nuestra aplicación es muy sencillo:
  - Hay que añadir **un fichero dentro de una carpeta determinada de nuestro proyecto.**
  - Para cada uno de los recursos que añadamos el sistema crea, de forma automática, **un id de recurso dentro de la clase R**.

Recursos de la aplicación (II)

- **Tipos de recursos**

  Según la carpeta que utilicemos el recurso creado será de un tipo específico. Pasamos a enumerar las carpetas y tipos posibles:

carpeta `identificador`	descripción
res/drawable/ `R.drawable`	Ficheros en bitmap (.png, .jpg o .gif). Ficheros PNG en formato Nine-patch (.9.png). Ficheros XML con descriptores gráficos (ver clase Drawable)
res/layout/ `R.layout`	Ficheros XML con los Layouts usados en la aplicación.
res/menu/ `R.menu`	Ficheros XML con los descriptores de menús que podemos asignar a nuestras actividades
res/anim/ `R.anim`	Fichero XML que permite definir una animación Tween para una vista.
res/animator `R.animator`	Fichero XML que permite modificar las propiedades de un objeto a lo largo del tiempo. Ver sección: animación de propiedades. Solo desde la versión 3.0.
res/xml/ `R.xml`	Otros ficheros XML, como los ficheros de preferencias
res/raw/ `R.raw`	Ficheros que se encuentran en formato binario. Por ejemplo ficheros de audio o vídeo.
res/mipmap/ `R.mipmap`	Es una carpeta con la misma finalidad que *res/drawable*. La única diferencia es que si ponemos los gráficos en mipmap, estos no son rescalados para adaptarlos a la densidad gráfica del dispositivo donde se ejecuta la aplicación, sino que se buscará en las subcarpetas el gráfico con la densidad más parecida y se utilizará directamente. Es recomendable guardar los iconos de aplicación en esta carpeta
res/color/ `R.color`	Define un recurso de color que es una lista de colores que se aplicarán en función del estado de una vista (si está pulsada, con el foco,...). Se hace con un selector.
res/values/	Ficheros XML que definen un determinado valor para definir un color, estilo, cadena de caracteres… se describen en la siguiente tabla.

Recursos de la aplicación (III)

- Veamos los tipos de recursos que encontramos dentro de la carpeta *values*:

fichero por defecto `identificador`	Descripción
strings.xml `R.string`	Identifica cadenas de caracteres `<string name="saludo">¡Hola Mundo!</string>`

colors.xml
`R.color`
	Un color definido en formato ARGB (alfa, rojo, verde y azul). Los valores se indican en hexadecimal en uno de los siguientes formatos: #RGB, #ARGB, #RRGGBB ó #AARRGGBB
	
	`<color name="verde_opaco">#0f0</color>`
	`<color name="red_translucido">#80ff0000</color>`

dimensions.xml
`R.dimen`
	Un número seguido de una unidad de medida.
	
	px - pixeles, mm - milímetros, in – pulgadas, pt – puntos (=1/72 pulgadas), dp – píxeles independientes de la densidad (=1/160 pulgadas), sp – igual que dp pero cambia según las preferencias de tamaño de fuente.
	
	`<dimen name="alto">2.2mm</dimen>`
	`<dimen name="tamano_fuente">16sp</dimen>`

Recursos de la aplicación (IV)

styles.xml
R.style
	Definen una serie de atributos que pueden ser aplicados a una vista o a una actividad. Si se aplican a una actividad se conocen como temas.
	
	`<style name="TextoGrande"  parent="@style/Text">`
	`  <item name="android:textSize">20pt</item>`
	`  <item name="android:textColor">#000080</item>`
	
	`</style>`

R.integer	Define un valor entero. `<integer name="max_asteroides">5</integer>`
R.bool	Define un valor booleano. `<bool name="misiles_ilimitados">true</bool>`

R.id	Define un recurso de ID único. La forma habitual de asignar ID a los recursos es utilizando el atributo **id="@+id/nombre"**. Aunque en algunos casos puede ser interesante disponer de IDs previamente creados, para que los elementos así nombrados tengan una determinada función. Este tipo de IDs se utilizan en las vistas TabHost y ListView.
	
	`<item type="id" name="button_ok"/>`
	`<item type="id" name="dialog_exit"/>`

Recursos de la aplicación (V)

R.array	Una serie ordenada de elementos. Pueden ser de strings, de enteros o de recursos (TypedArray)
	
	`<string-array name="dias_semana">`
	`<item>lunes</item>`
	`<item>martes</item>`
	`</string-array>`
	
	`<integer-array name="primos">`
	`<item>2</item>`
	`<item>3</item>`
	`<item>5</item>`
	`</integer-array>`
	
	`<array name="asteroides">`
	`<item>@drawable/asteroide1</item>`
	`<item>@drawable/asteroide1</item>`
	`</array>`

Acceso a los Recursos nuestros (I)

- Una vez definido un **recurso** este **puede ser utilizado**:
  - **desde un fichero XML** o,
  - **desde Java.**

- ***Ejemplo de acceso a un recurso desde XML***:

```
<ImageView
        android:layout_height="@dimen/alto"
        android:layout_width="fill_parent"
        android:background="@drawable/asteroide"
        android:text="@string/saludo"
        android:text_color="@color/verde_opaco"/>
```

Acceso a los Recursos nuestros (II)

- ***Ejemplo de acceso a un recurso desde CÓDIGO JAVA***:

```
Resources res =getResources();
```

```
Drawable drawable =res.getDrawable(R.drawable.asteroide);
String saludo = res.getString(R.string.saludo);
int color =res.getColor(R.color.verde_opaco);
float tamanoFuente =res.getDimension(R.dimen.tamano_fuente);
int maxAsteroides =res.getInteger(R.integer.max_asteroides);
boolean ilimitados = res.getBoolean(R.bool.misiles_ilimitados);
```

Acceso a NO ARRAYS

```
String[] diasSemana =res.getStringArray(R.array.dias_semana);
int[] primoss =res.getIntArray(R.array.primos);
TypedArray asteroides =res.obtainTypedArray(R.array.asteroides);
Drawable asteroide1 =asteroides.getDrawable(0);
```

Acceso a ARRAYS

Recursos del Sistema

Mª Alejandra Tomás Fernández

Recursos del sistema Android

- Además de nuestros recursos de aplicación, **Android dispone de un conjunto de recursos:**
  - **almacenados en el sistema** y
  - que **están disponibles para CUALQUIER APLICACIÓN**

- Se puede **acceder a estos recursos**:
  - **desde un fichero XML:**
  - **desde Java:**

VENTAJAS DE USAR Recursos del Sistema

1. **No consumen memoria en nuestra aplicación.**
2. **Los usuarios están familiarizados con ellos:**

   Ejemplo:

   android.R.drawable.ic_menu_edit 

3. **Se adaptan a las diferentes versiones.**
4. **Se adaptan a las configuraciones locales:**

   Ejemplo:

   android.R.string.cancel → Si yo utilizo este recurso que será “Cancelar”, “Cancel”, “取消”,... según el idioma escogido por el usuario..

Recursos del sistema Android

- Se puede **acceder a estos recursos**:

  - **desde un fichero XML:**
    - **Anteponiendo el prefijo "android:"** al recurso.
    - **Ejemplo:**

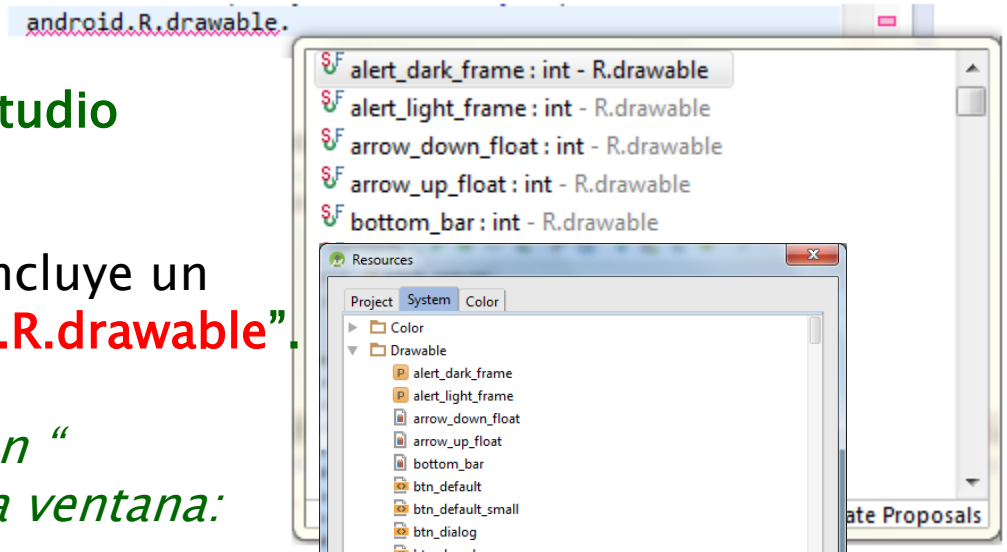
      @android:layout/simple_list_item_1

  - **desde Java:**
    - **Usando la clase**: **android.R**
    - Se usa la **misma estructura de clases anidadas**:
      - android.R.**drawable**
      - android.R.**string**
      - android.R.**layout.** **Ejemplo:** **android.R.layout.simple_list_item_1**

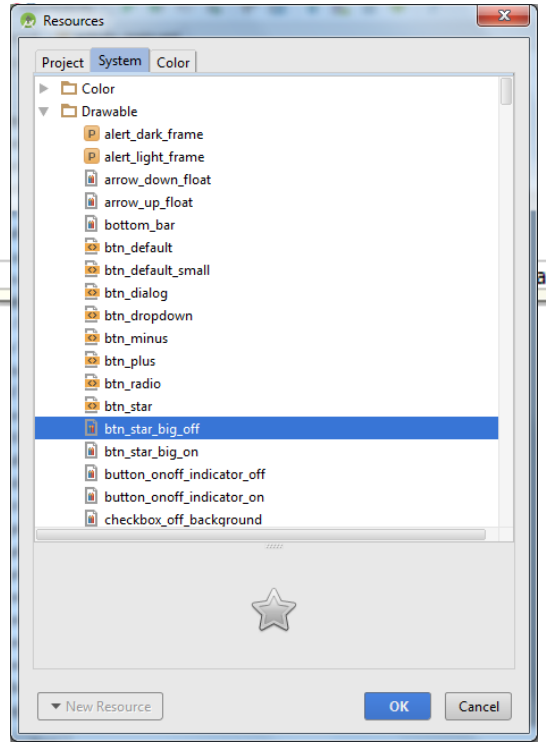
¿Cómo saber qué recursos del sistema tenemos? Explorarlos (I)

¿Qué recursos tenemos disponibles?

- Podemos usar **la opción de autocompletar de Android Studio**



- En el **editor de Layouts** se incluye un **visor de Recursos "android.R.drawable"**. *Al añadir en modo visual un componente gráfico como un "ImageView" se muestra esta ventana:*



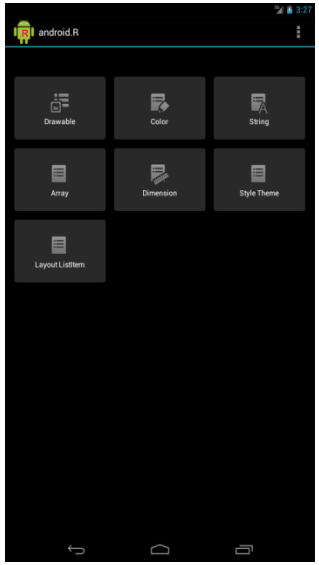
- Para **explorar el resto de recursos** disponibles (*no drawable*), te **recomiendo las siguientes aplicaciones en un terminal Android**:

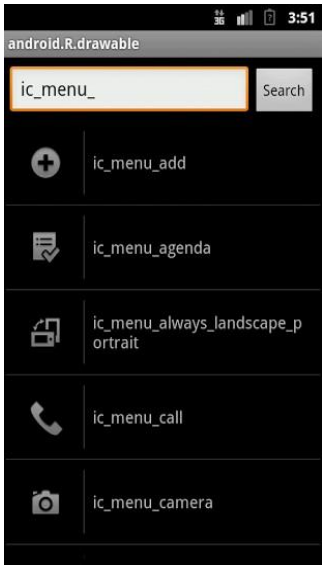
¿Cómo saber qué recursos del sistema tenemos? Explorarlos (II)

Aplicaciones terminal Android que muestrar recursos:

Instala en un terminal la **aplicación android.R** (*está en GooglePlay):*

- *Muestra 7 tipos de recursos agrupados por tipos y en forma visual, además del nombre del recurso.*





Para ver los **recursos de animación R.explorer**

Mª Alejandra Tomás Fernández

Recursos Alternativos

Mª Alejandra Tomás Fernández

Recursos alternativos (I)

- En Android se diferencian dos tipos de recursos:

1. **Recursos por defecto:** se usan cuando no se ha definido un recurso alternativo o no se cumplen las condiciones de aplicación.

   SIEMPRE DEBEN EXISTIR.

   SI SÓLO USAMOS RECURSOS ALTERNATIVOS, LA APP PUEDE ROMPER CUANDO NO COINCIDA LA CONFIGURACIÓN.

2. **Recursos alternativos:** son definidos para una **configuración específica**.

   - Para definir un recurso alternativo hay que alojarlo en una nueva **carpeta con un sufijo** que indique **cuándo hay que utilizarlo**.

   - Estos sufijos pueden hacer referencia a la **orientación del dispositivo,** al **lenguaje**, la **región**, la **densidad de píxeles**, la **resolución**, el **método de entrada**,…

Recursos alternativos (II)

REGLAS PARA LOS SUFIJOS (QUALIFIER):

1. Se pueden **especificar varios sufijos** para un mismo conjunto de recursos, pero separado por "–":

   dravable-en-Rus-land

2. Los sufijos **deben estar en el orden** listado en la table 2. Por ejemplo:

   MAL: drawable-hdpi-port/

   BIEN: drawable-port-hdpi/

3. Los **directorios** de recursos alternativos **no pueden anidarse.**

Por ejemplo, no se puede hacer: res/drawable/drawable-en

Recursos alternativos (III)

4. Sólo se puede usar un valor de cada sufijo en un mismo directorio.

   a. Por ejemplo, si queremos usar el mismo fichero drawable para ESPAÑOL y FRANCÉS. **NO PUEDO HACER**:

      llamar al directorio: **drawable-rES-rFR/**

   b. En ese ejemplo **SÍ DEBO HACER**:

      crear dos directorios de recursos, cada uno con sus ficheros:

      drawable-rES/

      drawable-rFR/

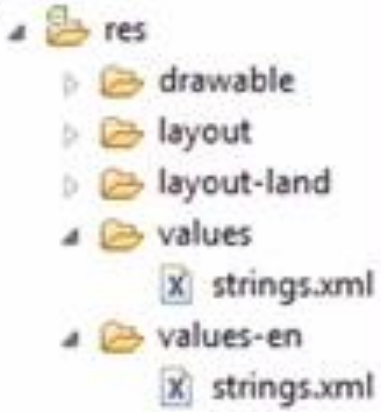
      También podría crear un **alias para recursos**. Ya lo veremos más adelante.

DIFERENTES IDIOMAS

Mª Alejandra Tomás Fernández

Recursos alternativos para soportar diferentes IDIOMAS(IV)

- El fichero *res/values/strings.xml* permite introducir los textos utilizados.
- Para que estos textos dependan del idioma seleccionado en Android hay que crear varias carpetas según el idioma.



Layout usado por defecto (values)

Layout para inglés (values-en)

Recursos alternativos para soportar diferentes IDIOMAS(V)

- Un ejemplo del contenido de esos ficheros strings.xml podría ser:

English (default locale), /values/strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="title">My Application</string>
    <string name="hello_world">Hello World!</string>
</resources>
```

Spanish, /values-es/strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="title">Mi Aplicación</string>
    <string name="hello_world">Hola Mundo!</string>
</resources>
```

French, /values-fr/strings.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="title">Mon Application</string>
    <string name="hello_world">Bonjour le monde !</string>
</resources>
```

DISEÑO PARA MÚLTIPLES PANTALLAS

Mª Alejandra Tomás Fernández

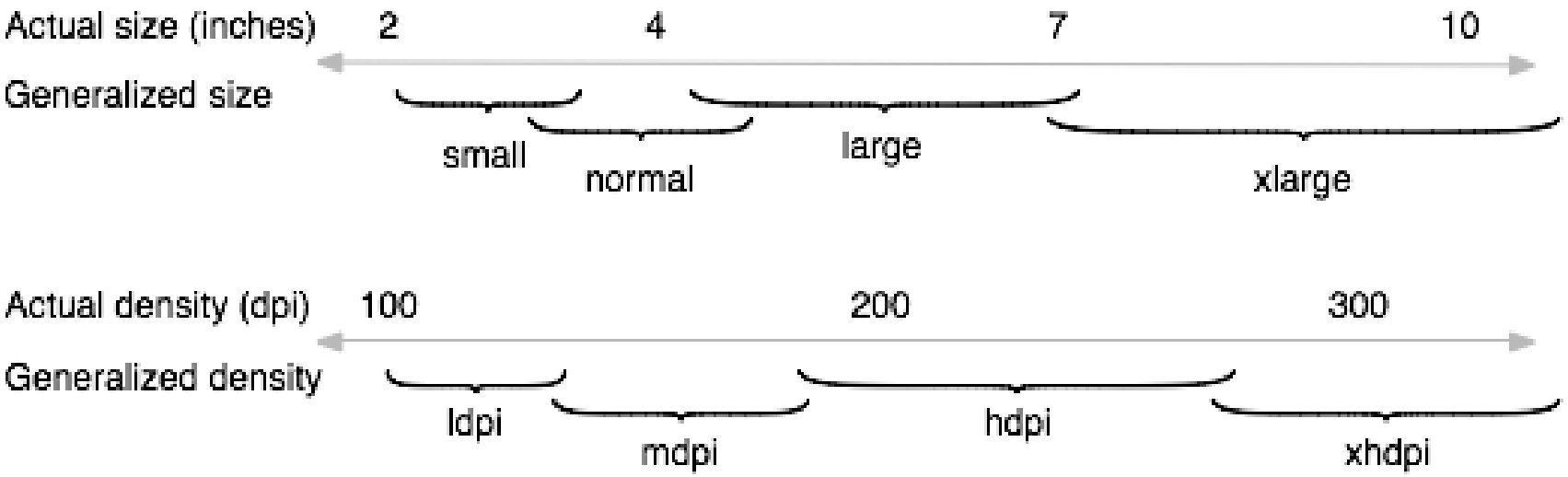
Recursos alternativos soportar DIFERENTES PANTALLAS (VI)

- Android clasifica las pantallas según dos características:
  - TAMAÑO (*SIZE*): *small, normal, large, xlarge*
    - *ORIENTACIÓN (ORIENTATION):* *port , land*

      Se considera una variación del tamaño de la pantalla.
  - DENSIDAD (DENSITY):

    ldpi (low) ~120dpi
    mdpi (medium) ~160dpi
    hdpi (high) ~240dpi
    xhdpi (extra-high) ~320dpi
    xxhdpi (extra-extra-high) ~480dpi
    xxxhdpi (extra-extra-extra-high) ~640dpi

Recursos alternativos
soportar DIFERENTES PANTALLAS (VII)



Recursos alternativos
soportar DIFERENTES PANTALLAS (VIII)

- Para que nuestra aplicación soporte diferentes configuraciones de pantalla (tamaño, densidad, resolución,…), deberíamos:

1. **Declarar explícitamente en el manifest qué tamaños de pantalla soporta** nuestra app. Etiqueta <supports-screens>
   - Evita que dispositivos con pantallas no compatibles descargen la app.
2. **Proveer diferentes layouts para diferentes tamaños** (size) de pantallas.
   - Usando los sufijos: *small, normal, large, and xlarge*. .
   - Por ejemplo, para un layout para una pantalla extragrande, debería ir en la carpeta: layout-xlarge/
   - A partir de Android 3.2 (API level 13), fue "deprecated" lo de **small,…**
     - Ahora se usa sw<N>dp como sufijo.
     - Esto define el ancho mínimo requerido por nuestro layout.
     - Por ejemplo, si nuestro layout requiere **al menos 600 dp de ancho** de pantalla, debería situarlo en: **layout-sw600dp/**

Recursos alternativos
soportar DIFERENTES PANTALLAS (IX)

3. **Proveer diferentes bitmap drawables para diferentes densidades :**
   - Usando los sufijos: *ldpi, mdpi, hdpi,…* .
   - Por ejemplo, bitmaps para pantallas de **alta densidad**, deberían ir en drawable-hdpi/.
   - Si se van a **escalar los bitmaps**: fondos botón,… usar NINE-PATCH BITMAPS

EJEMPLOS VARIOS:

```
res/layout/my_layout.xml             // layout for normal screen size ("default")
res/layout-large/my_layout.xml       // layout for large screen size
res/layout-xlarge/my_layout.xml      // layout for extra-large screen size
res/layout-xlarge-land/my_layout.xml // layout for extra-large in landscape orientation

res/drawable-mdpi/graphic.png        // bitmap for medium-density
res/drawable-hdpi/graphic.png        // bitmap for high-density
res/drawable-xhdpi/graphic.png       // bitmap for extra-high-density
res/drawable-xxhdpi/graphic.png      // bitmap for extra-extra-high-density

res/mipmap-mdpi/my_icon.png        // launcher icon for medium-density
res/mipmap-hdpi/my_icon.png        // launcher icon for high-density
res/mipmap-xhdpi/my_icon.png       // launcher icon for extra-high-density
res/mipmap-xxhdpi/my_icon.png      // launcher icon for extra-extra-high-density
res/mipmap-xxxhdpi/my_icon.png     // launcher icon for extra-extra-extra-high-density
```

Cómo admitir varios tamaños de pantalla

Mª Alejandra Tomás Fernández

Recursos alternativos
TAMAÑOS DE PANTALLA(X)

- **DISEÑO RELATIVO**:
  - Para garantizar que el **diseño es flexible** y que se adapta a varios tamaños de pantalla ***(los componentes se adaptan),*** debes utilizar los valores "**wrap_content**" y "**match_parent**" para la **altura y el ancho** de algunos componentes de la vista. :
- **DISEÑO ADAPTATIVO**:
  - Consiste en **hacer diseños alternativos, totalmente diferentes para diferentes pantallas**. Así los componentes pueden **organizarse de formas totalmente distintas** para aprovechar mejor el tamaño de la pantalla.

Recursos alternativos
TAMAÑOS DE PANTALLA(XI)

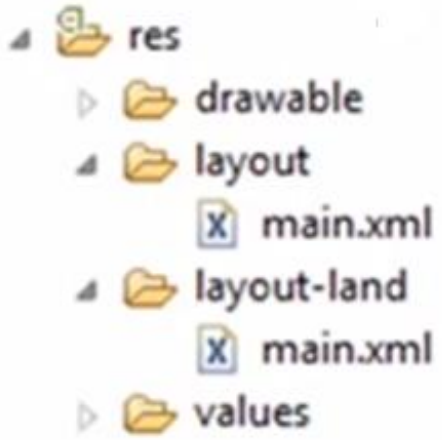
- PINCHA EN ESTE ENLACE PARA VER CÓMO SE HACE
- OTRO ENLACE MÁS COMPLETO
- MUY BUEN EJEMPLO DE USO (a partir del apartado Cómo utilizar calificadores de tamaño)-> VERLO AQUÍ

Mª Alejandra Tomás Fernández

Recursos alternativos

soportar DIFERENTES ORIENTACIONES(XII)

- Android conoce si el terminal está en apaisado o en vertical.
- Puede resultar muy interesante cambiar el diseño del interfaz de usuario en función de la posiciones



Layout usado por defecto (layout)

Layout para apaisado (layout-land)

Recursos alternativos concatenar SUFIJOS (XIII)

- Se puede indicar varios sufijos concatenados. **Pero tienen un orden**:

res/values-en-rUS/strings.xml

res/values-en-rUK/strings.xml

Recursos alternativos
SUFIJOS más usados (XIV)

- Orientación de pantalla:
  port / land
- Tamaño de pantalla:
  small / normal /large/ xlarge
- Idioma:
  en / fr / es / …
- Región:
  rUS / rUK / …
- Densidad de píxeles (dpi):
  ldpi / mdpi/ hdpi / xhdpi / xxhdpi / nodpi /…
- Nivel de API:
  v3 / v4 / v5 /…
- Modo nocturno:
  night / no night