Deseño de pantallas: Layouts

Sumario

Introdución

- Un **Layout** é un elemento da Interface de Usuario (UI).
- Nel podemos definir os elementos visuais que compoñen a pantalla.
- Pode ser definido en ficheiros **xml** ou en código **Java** en tempo de execución. Xa vimos no apartado anterior as vantaxes dos ficheiros xml.
- Cada ficheiro xml asociado a unha pantalla/layout debe conter un **elemento raíz** e dentro deste poderanse ir engadindo máis layouts e obxectos fillos até construír unha xerarquía de Vistas (Views) que definirán a pantalla/layout.

- No seguinte exemplo a liña 2 indica o inicio do elemento raíz que se pecha na liña 15.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:orientation="vertical" >
    <TextView android:id="@+id/text"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="I am a TextView" />
    <Button android:id="@+id/button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:onclick="onButtonClick"
            android:text="I am a Button"/>
</LinearLayout>
```

Cargar recurso XML: layout

- Como xa sabemos, cada vez que se compila, cada ficheiro layout XML é compilado a unha obxecto Vista accesible a través de Java por medio da clase R.

- O layout cargarase cando se chame ao método **onCreate()** da Activity chamando ao método **setContentView()** ao que se lle pasa como referencia o recurso do layout a través da Clase R e do nome do layout (o nome do ficheiro xml, neste caso: main_layout.xml)

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main_layout);
}
```

- Deste xeito cargarase en pantalla o layout con todos os elementos visuais que contén.

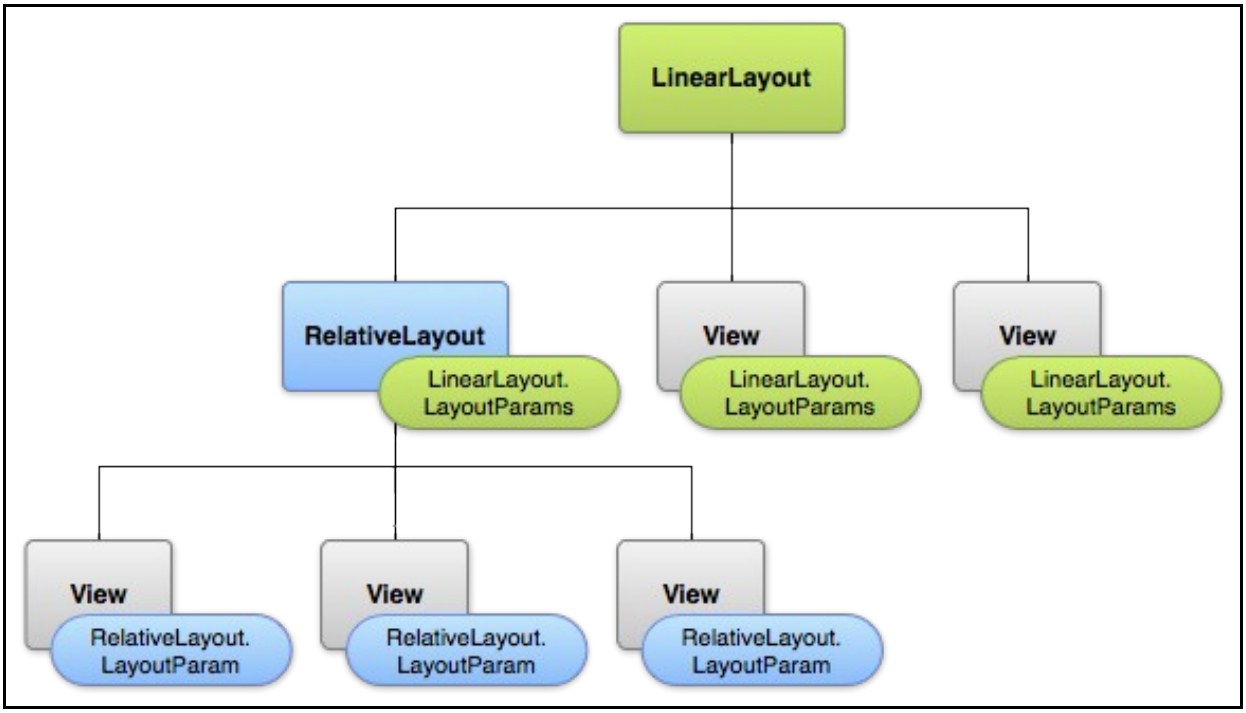
- Un layout ten **atributos** ao igual que calquera View como xa se indicou no apartado anterior.

- Un layout estende a clase ViewGroup como se víu no apartado anterior.

- Referencias:
    - Layouts:http://developer.android.com/guide/topics/ui/declaring-layout.html
    - Parámetros do layout: http://developer.android.com/reference/android/view/ViewGroup.LayoutParams.html

Parámetros dun layout

- Os atributos xml dun layout, que comezan chamándose **layout**_*algo*, definen os atributos que son apropiados para cada ViewGroup onde se atopa a Vista



- A imaxe amosa a como os Views fillos herdan os **LayoutParams** dos contedores, dos pais.
- Cada View fillo debe definir os seus parámetros apropiados en función do seu pai, aínda que pode definir parámetros para os seus fillos.
- Cada fillo contén propiedades tipo que definen o seu tamaño e posición:

Tamaño

- Definir valores para os atributos: **layout_width** e **layout_height**.
    - ♦ Normalmente usarase estes dous valores:
        - ◊ **wrap_content**: axusta o tamaño ao contido.
        - ◊ **match_parent**: aumenta de tamaño até o tamaño do contedor pai.
    - ♦ Outros posible tipos de valores son:
        - ◊ **px, dp, sp, in, mm, pt**.
        - ◊ As unidades px, dp e sp xa se viron nun apartado anterior.
- En Java podemos capturar o tamaño dunha View con:
    - ♦ Para saber o **tamaño desexado** do View temos **getMeasuredWidth()** e **getMeasuredHeight()**.
    - ♦ Para saber o **ancho e alto real** do View usaremos **getWidth()** e **getHeight()**.

Posición

- As views son un rectángulo xeometricamente falando.
- Unha view está definida por:
    - ♦ Unha **localización**: expresada polo par de coordenadas: **esquerda e arriba** (left e top)
    - ♦ Unha **dimensión**: expresada como **ancho e alto** (width e height)
    - ♦ A unidade para medir a localización e a dimensión é o píxel (px).
- En Java para **obter a localización** temos:
    - ♦ getLeft() e getTop()
    - ♦ getRight() e getBottom()
    - ♦ Devolven o X e Y relativa ao seu pai.

Recheo. Padding

- O **Padding** é o recheo/espacio que deixa o contido da vista con respecto aos seus lados.
- O tamaño do **Padding** exprésase en calquera das unidades vistas previamente (px, pt, dp, sp, mm, in) para os lados esquerdo, superior, dereito e inferior (left, top, right, bottom) da vista.
    - Por exemplo un paddind de 2px no lado esquerdo indica que o contido da vista vai comezar 2px á dereita do borde esquerdo.
- O tamaño da vista é todo: contido máis recheo o espaciado.

- O tamaño do padding dunha view pode expresarse en **XML** como:
    - android:padding (Neste caso o mesmo recheo para os catro lados)
    - android:paddingLeft
    - android:paddingTop
    - android:paddingRight
    - android:paddingBottom

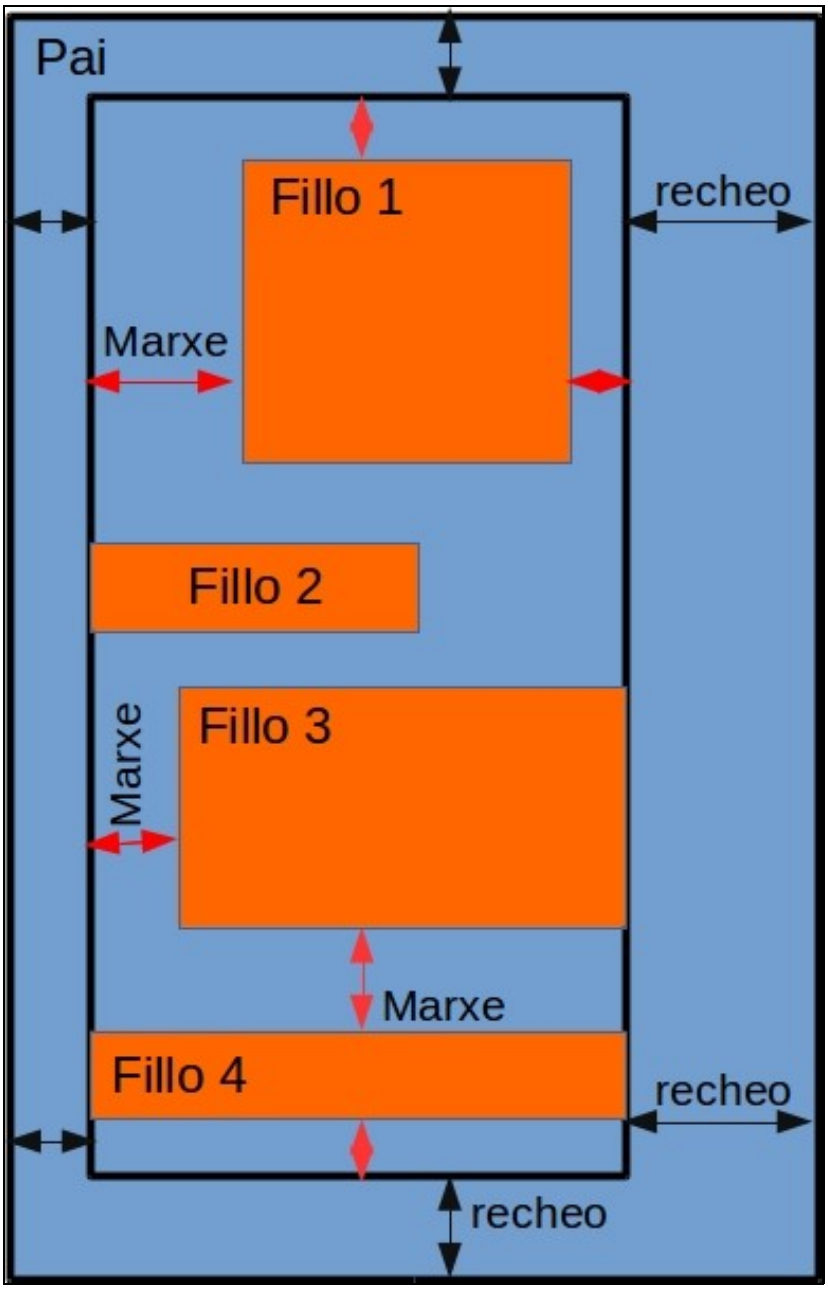
- O tamaño do padding dunha view pode expresarse en **Java** con:
    - setPadding(int, int, int, int)

- O tamaño do padding dunha view pode capturarse en **Java** con:
    - getPaddingLeft(), getPaddingTop(), getPaddingRight() and getPaddingBottom()

Marxes

- android:layout_margin
- android:layout_marginBottom
- android:layout_marginTop
- android:layout_marginLeft
- android:layout_marginRight

Explicación Gráfica


Pai
Fillo 1
recheo
Marxe
Fillo 2
Fillo 3
Marxe
Marxe
Fillo 4
recheo
recheo

- A vista **PAI** ten definida un recheo distinto para cada lado: os elementos que contén o PAI van estar separados do borde do pai pola distancia definida no recheo.
- Cada vista **Fillo** ten definido un marxe (ou varios) con respecto aos límites que lle marca o PAI.
    - **Fillo 1**: Pola esquerda e por arriba, comeza onde remata o recheo do PAI, pero a maiores este elemento ten definidos uns marxes propios para eses lados.
    - **Fillo 2**: Non ten definido ningún marxe esquerdo.
    - ...

Construción dun layout

- Un layout pode ser construído de 2 formas:
    - **Ficheiro xml**
    - **Cun Adaptador**: cando non se coñece o contido do layout ou o seu contido é dinámico podemos usar a clase de Java **AdapterView** que en tempo de execución creará o layout en engadirá nel as vistas que se indiquen.

- Imos ver nos seguintes apartados como se constrúen layouts.
- Entre os máis comúns están FrameLayout, LinearLayout, RelativeLayout,
- Cada un deles dispón os elementos visuais no seu interior de distintas formas.
- Cada un deles pode conter layouts do mesmo ou distinto tipo e así sucesivamente.

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