Diseño del Interfaz - III

Trabajando con Contenedores

Hasta ahora, hemos estado trabajando con widgets y contenedores que hemos definido en el **activity_main.xml**. Lo único que teníamos que hacer para trabajar con ellos desde en el **MainActivity.java** era simplemente referenciarlos mediante una variable:

```
private ImageButton imageButton = null;
imageButton = (ImageButton) findViewById( R.id.imageButton );
```

Pero ésta no es la única forma de trabajar con ellos. Es posible recorrer los hijos de un Contendor en busca de uno en concreto; es posible crear un widget en una actividad y asignárselo a un Contenedor; o incluso responder a los eventos de los widget-hijo de un Contenedor. Sí, tu **activity_main.xml** no deja de ser un árbol xml normal y corriente que puede ser recorrido de la misma forma: acceder a una hoja, añadir nuevas hojas, etc.

Recorrer un Contenedor

Si has trabajado anteriormente recorriendo xml mediante Java (con librerías), encontrarás similitudes evidentes en Android. En el siguiente ejemplo tienes cómo se recorren los widgets de un contenedor. Es la tarea más básica que se puede hacer. Se resume a obtener una referencia de cada widget, algo que se puede hacer con un simple bucle **for**. La clase abstracta **GroupView** ofrece una serie de métodos para ello: **getChildAt ()** permite recuperar el hijo en la posición X; **getChildCount ()** retorna el número de elementos del Contenedor; etc.

Por tanto, podríamos hacer:

```
public void getAllChildren (){
    View view;
    GridLayout gridLayout = (GridLayout) findViewById( R.id.myGrid );
    for (int i = 0; i < gridLayout.getChildCount(); i++){
        view = gridLayout.getChildAt( i );
        System.out.println( "Hijo num" + i + gridLayout.toString());
    }
}
```

Hay que tener cuidado porque no todos los widgets de un mismo contenedor son del mismo tipo. Podemos guardarlos en una variable **View**, pero esto no nos servirá cuando queramos asignar un Evento a un **Button** y no a un **ImageButton** (por ejemplo). Para diferenciar los diferentes Widgets podemos usar el método o **getClass ()**, y después hacer un *cast* del objeto a una Clase determinada. Por ejemplo:

```
public void putEventOnlyToButtons (){
    View view;
    GridLayout gridLayout = (GridLayout) findViewById( R.id.myGrid );
    for (int i = 0; i < gridLayout.getChildCount(); i++){
        view = gridLayout.getChildAt( i );
        System.out.println( "Hijo num" + i + gridLayout.toString());

        // If a button...
        Button button = null;
        if (view.getClass().getSimpleName().equals("Button")){
                button = (Button) view; // Cast to button
                button.setOnClickListener(...); // Asign Event
        }
    }
```

Añadir elementos a un Contenedor

De la misma forma, es posible añadir elementos a un Contenedor que ya existe. Lo único que tienes que tener en cuenta es:

1) Cada widget necesita una serie de parámetros definidos SI o SI, los llamados Parámetros de Layout. No es lo mismo añadir un widget a un AbsoluteLayout que a un RelativeLayout. No indicarlos generará errores imprevistos. La forma más fácil de saber cuáles son es usar el método **setLayoutParams ()** del Widget, pues te pedirá que le pases todos los parámetros que necesita.

```
// Layout Params
myButton.setLayoutParams(new ViewGroup.LayoutParams(
        ViewGroup.LayoutParams.WRAP_CONTENT,
        ViewGroup.LayoutParams.WRAP_CONTENT) );
```

2) Todos los widgets necesitan un ID propio. Para evitar problemas, lo mejor es dejar que Android genere ese ID dinámicamente usando **generateViewId ()**.

3) Para añadir un añadir un Widget al Contenedor se usa el método **addView ()**.

Práctica 17

Realiza la práctica propuesta en el documento **06 – Contenedores Programáticos [Java]**. Debería de ser sencilla de realizar con lo que ya sabes.

Práctica 18

Realiza la práctica propuesta en el documento **06 – Contenedores Programáticos [Kotlin]**. Debería de ser sencilla de realizar con lo que ya sabes.

Práctica 19

[Java] Repite la práctica **02 – Calculador [Java]**, pero cada vez que se pulse el botón de resultado todos los botones cambiarán de color.

> **Nivel Experto**: Es posible escribir esta app de forma que, en el peor de los casos, tanto cuando se lanza la App, cuando se pulsa un botón de color, o se pulsa el Reset, sólo se hacen como máximo 18 evaluaciones de condición (Evaluar un if, o un for, o un while, etc.).

Práctica 20

[Kotlin] Repite la práctica **02 – Calculador [Kotlin]**, pero cada vez que se pulse el botón de resultado todos los botones cambiarán de color.