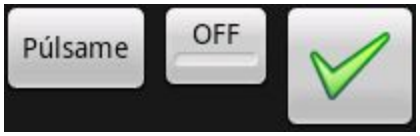
Controles básicos (I)

En este documento nos vamos a centrar en los diferentes tipos de botones y cómo podemos personalizarlos. El SDK de Android nos proporciona tres tipos de botones: el clásico (**Button**), el de tipo on/off (**ToggleButton**), y el que puede contener una imagen (**ImageButton**). En la imágen siguiente vemos el aspecto por default de estos tres controles.



Control Button

Un control de tipo **Button** es el botón más básico que podemos utilizar. En el ejemplo siguiente definimos un botón con el texto "Púlsame" asignando su propiedad `android:text`. Además de esta propiedad podríamos utilizar muchas otras como el color de fondo (`android:background`), estilo de fuente (`android:typeface`), color de fuente (`android:textcolor`), tamaño de fuente (`android:textSize`), etc.

```
<Button android:id="@+id/BtnBoton1"
    android:text="@string/pulsame"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Control ToggleButton

Un control de tipo **ToggleButton** es un tipo de botón que puede permanecer en dos posibles estados, pulsado / no pulsado. En este caso, en vez de definir un sólo texto para el control definiremos dos, dependiendo de su estado.
Así, podremos asignar las propiedades `android:textOn` y `android:textOff` para definir ambos textos.

```
<ToggleButton android:id="@+id/BtnBoton2"
    android:textOn="@string/on"
    android:textOff="@string/off"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Control ImageButton

En un control de tipo **ImageButton** podremos definir una imagen a mostrar en vez de un texto, para lo que deberemos asignar la propiedad `android:src`. Normalmente asignaremos esta propiedad con el descriptor de algún recurso que hayamos incluido en la carpeta /res/drawable. Así, por ejemplo, en nuestro caso hemos incluido una imagen llamada "*ok.png*" por lo que haremos referencia al recurso ***"@drawable/ok"*.**
Adicionalmente, al tratarse de un control de tipo imagen también deberíamos acostumbrarnos a asignar la propiedad `android:contentDescription` con una descripción textual de la imagen, de forma que nuestra aplicación sea lo más accesible posible.

```
<ImageButton android:id="@+id/BtnBoton3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:contentDescription="@string/icono_ok"
    android:src="@drawable/ok" />
```

Cabe decir además, que aunque existe este tipo específico de botón para imágenes, también es posible añadir una imagen a un botón normal de tipo **Button**, a modo de elemento suplementario al texto. Por ejemplo, si quisiéramos añadir un icono a la izquierda del texto de un botón utilizaríamos la propiedad `android:drawableLeft` indicando como valor el descriptor (ID) de la imagen que queremos mostrar:

```
<Button android:id="@+id/BtnBoton5"
    android:text="@string/pulsame"
    android:drawableLeft="@drawable/ok"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content" />
```

Eventos de un botón

Como podemos imaginar, aunque estos controles pueden lanzar muchos otros eventos, el más común de todos ellos y el que querremos capturar en la mayoría de las ocasiones es el evento **onClick**, que se lanza cada vez que el usuario pulsa el botón. Para definir la lógica de este evento tendremos que implementarla definiendo un nuevo objeto `View.OnClickListener()` y asociándolo al botón mediante el método **`setOnClickListener`**`()`.
La forma más habitual de hacer esto es la siguiente:

```
Button btnBoton1 = (Button)findViewById(R.id.BtnBoton1);

btnBoton1.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        lblMensaje.setText("Botón 1 pulsado!");
    }
});
```

En el caso de un botón de tipo **ToggleButton** suele ser de utilidad conocer en qué estado ha quedado el botón tras ser pulsado, para lo que podemos utilizar su método **`isChecked`**`()`. En el siguiente ejemplo se comprueba el estado del botón tras ser pulsado y se realizan acciones distintas según el resultado.

```
ToggleButton btnBoton2 = (ToggleButton)findViewById(R.id.BtnBoton2);

btnBoton2.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        if(btnBoton2.isChecked()){
            lblMensaje.setText("Botón 2: ON");
        }else{
            lblMensaje.setText("Botón 2: OFF");
        }
    }
});
```

Personalizar el aspecto un botón [y otros controles]

En la imagen mostrada al principio del artículo vimos el aspecto que presentan por default los tres tipos de botones disponibles. Pero, ¿y si quisiéramos personalizar su aspecto más allá de cambiar un poco el tipo o el color de la letra o el fondo?
Para cambiar la forma de un botón podríamos simplemente asignar una imagen a la propiedad `android:background`, pero esta solución no nos serviría de mucho porque siempre se mostraría la misma imagen incluso con el botón pulsado, dando poca sensación de elemento "*clickable*".
La solución perfecta pasaría por tanto por definir diferentes imágenes de fondo dependiendo del estado del botón. Pues bien, Android nos da total libertad para hacer esto mediante el uso de ***selectores***. Un *selector* se define mediante un archivo XML localizado en la carpeta /res/drawable, y en él se pueden establecer los diferentes valores de una propiedad determinada de un control dependiendo de su estado.
Por ejemplo, si quisiéramos dar un aspecto plano a un botón ToggleButton, podríamos diseñar las imágenes necesarias para los estados "pulsado" (en el ejemplo `toggle_on.png`) y "no pulsado" (en el ejemplo `toggle_off.png`) y crear un selector como el siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_checked="false" android:drawable="@drawable/toggle_off" />
    <item android:state_checked="true" android:drawable="@drawable/toggle_on" />
</selector>
```

En el código anterior vemos cómo se asigna a cada posible estado del botón una imagen (un elemento *drawable*) determinada. Así, por ejemplo, para el estado "pulsado" (`android:state_checked="true"`) se asigna la imagen `toggle_on`.
Este selector lo guardamos por ejemplo en un archivo llamado toggle_style.xml y lo colocamos como un recurso más en nuestra carpeta de recursos /res/drawable. Hecho esto, tan sólo bastaría hacer referencia a este nuevo recurso que hemos creado en la propiedad `android:background` del botón:

```
<ToggleButton android:id="@+id/BtnBoton4"
    android:textOn="@string/on"
    android:textOff="@string/off"
    android:padding="10dip"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:background="@drawable/toggle_style"/>
```

En la siguiente imagen vemos el aspecto por defecto de un **ToggleButton** (columna izquierda) y cómo ha quedado nuestro **ToggleButton** personalizado (columna derecha).

En las imágenes siguientes se muestra la aplicación de ejemplo desarrollada, donde se puede comprobar el aspecto de cada uno de los tipos de botón comentados para las versiones de Android 2.x y 4.x