Controles básicos (III)

En este documento vamos a ver cómo utilizar otros dos tipos de controles básicos en muchas aplicaciones, los *checkboxes* y los *radiobuttons*.

Control CheckBox

Un control *checkbox* se suele utilizar para marcar o desmarcar opciones en una aplicación, y en Android está representado por la clase del mismo nombre, **CheckBox**. La forma de definirlo en nuestra interfaz y los métodos disponibles para manipularlos desde nuestro código son análogos a los ya comentados para el control **ToggleButton**.
De esta forma, para definir un control de este tipo en nuestro *layout* podemos utilizar el código siguiente, que define un *checkbox* con el texto "Márcame":

```
<CheckBox android:id="@+id/ChkMarcame"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/marcame"
    android:checked="false" />
```

En cuanto a la personalización del control podemos decir que éste extiende [indirectamente] del control **TextView**, por lo que todas las opciones de formato ya comentadas previamente son válidas también para este control.
Además, podremos utilizar la propiedad `android:checked` para inicializar el estado del control a marcado (*true*) o desmarcado (*false*). Si no establecemos esta propiedad el control aparecerá por defecto en estado desmarcado.
En el código de la aplicación podremos hacer uso de los métodos **`isChecked`**`()` para conocer el estado del control, y **`setChecked`**`(estado)` para establecer un estado concreto para el control.

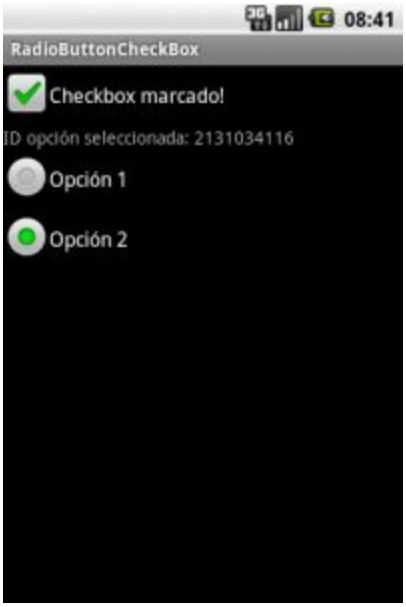
```
if (checkBox.isChecked()) {
    checkBox.setChecked(false);
}
```

En cuanto a los posibles eventos que puede lanzar este control, el más interesante, al margen del siempre válido **onClick**, es sin duda el que informa de que ha cambiado el estado del control, que recibe el nombre de **onCheckedChanged**. Para implementar las acciones de este evento podríamos utilizar la siguiente lógica, donde tras capturar el evento, y dependiendo del nuevo estado del control (variable **isChecked** recibida como parámetro), haremos una acción u otra:

```
private CheckBox cbMarcame;
cbMarcame = (CheckBox)findViewById(R.id.chkMarcame);

cbMarcame.setOnCheckedChangeListener(
    new CheckBox.OnCheckedChangeListener() {
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (isChecked) {
                cbMarcame.setText("Checkbox marcado!");
            } else {
                cbMarcame.setText("Checkbox desmarcado!");
            }
        }
    });
```

Control RadioButton



Al igual que los controles *checkbox*, un *radiobutton* puede estar marcado o desmarcado, pero en este caso suelen utilizarse dentro de un grupo de opciones donde una, y sólo una, de ellas debe estar marcada obligatoriamente, es decir, que si se marca una de las opciones se desmarcará automáticamente la que estuviera activa anteriormente.

En Android, un grupo de botones *radiobutton* se define mediante un elemento **RadioGroup**, que a su vez contendrá todos los elementos **RadioButton** necesarios. Veamos un ejemplo de cómo definir un grupo de dos controles *radiobutton* en nuestra interfaz:

```
<RadioGroup android:id="@+id/gruporb"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <RadioButton android:id="@+id/radio1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/opcion_1" />

    <RadioButton android:id="@+id/radio2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/opcion_2" />
</RadioGroup>
```

En primer lugar vemos cómo podemos definir el grupo de controles indicando su orientación (vertical u horizontal) al igual que ocurría por ejemplo con un *LinearLayout*. Tras esto, se añaden todos los objetos RadioButton necesarios indicando su ID mediante la propiedad `android:id` y su texto mediante `android:text`.

Una vez definida la interfaz podremos manipular el control desde nuestro código java haciendo uso de los diferentes métodos del control **RadioGroup**, los más importantes: `check(id)` para marcar una opción determinada mediante su ID, `clearCheck()` para desmarcar todas las opciones, y `getCheckedRadioButtonId()` que como su nombre indica devolverá el ID de la opción marcada (o el valor -1 si no hay ninguna marcada). Veamos un ejemplo:

```
RadioGroup rg = (RadioGroup)findViewById(R.id.gruporb);
rg.clearCheck();
rg.check(R.id.radio1);
int idSeleccionado = rg.getCheckedRadioButtonId();
```

En cuanto a los eventos lanzados, a parte de **onClick**, al igual que en el caso de los checkboxes, el más importante será el que informa de los cambios en el elemento seleccionado, llamado también en este caso **onCheckedChange**.
Vemos cómo tratar este evento del objeto **RadioGroup**, haciendo por ejemplo que una etiqueta de texto cambie de valor al seleccionar cada opción:

```
private TextView lblMensaje;
private RadioGroup rgOpciones;

lblMensaje = (TextView)findViewById(R.id.LblSeleccion);
rgOpciones = (RadioGroup)findViewById(R.id.gruporb);

rgOpciones.setOnCheckedChangeListener(
    new RadioGroup.OnCheckedChangeListener() {
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            lblMensaje.setText("ID opción seleccionada: " + checkedId);
        }
    });
```