Control CheckBox

December 1, 2020

Introducción

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.

1 Diseño del elemento

Para crear cada opción de casilla de verificación, cree en el diseño un **CheckBoxen**. Como el usuario puede seleccionar varios elementos del conjunto de opciones de casillas, cada casilla de verificación se debe de administra por separado.

Para definir un control de este tipo en nuestro layout podemos utilizar el código siguiente

```
<?xml version="1.0" encoding="utf-8"?>
....
<CheckBox
        android:id="@+id/verificar1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:text="@string/texto_mostrar1"
        android:onClick="controlcheckbox"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />

    <CheckBox
        android:id="@+id/verificar2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="16dp"
        android:text="@string/texto_mostrar2"
        android:onClick="controlcheckbox"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
```

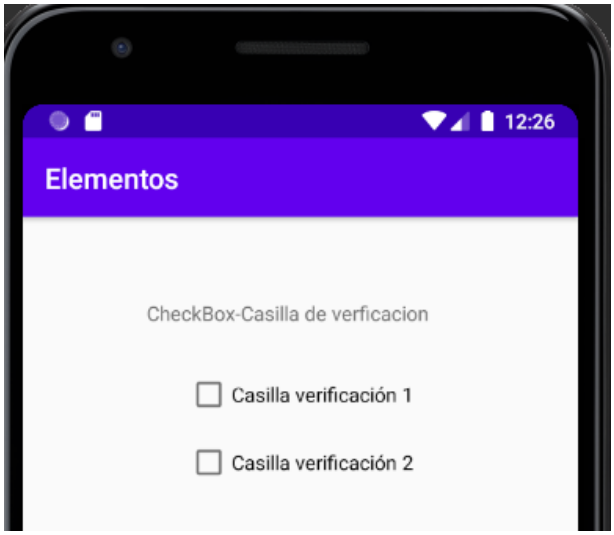
```
23          app:layout_constraintTop_toBottomOf="@+id/verificar1" />
```

Se ha añadido los valores para los string "texto_mostrar1",...

strings.xml

```
1 <resources>
2     <string name="app_name">Elementos</string>
3
4     <string name="texto_mostrar1">Casilla verificación 1</string>
5     <string name="texto_mostrar2">Casilla verificación 2</string>
6 </resources>
```

El resultado al ejecutar el proyecto es:



2 Atributos interesantes

El atributo a destacar es

- El atributo **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.
- Otra de los atributos que tenemos que utilizar es **android:onClick**, a este atributo le vamos a asociar un método java para programar lo que tiene que realizar cuando se selecciona este item.

3 Lógica del CheckBox

En este apartado vamos a explicar el funcionamiento del elemento, para ello vamos a trabajar con el fichero *ActivityMain*.

Esto puedes implementarlo de muchas maneras distintas pero la base es la siguiente. Primero podemos comprobar si el checkbox está checkeado/marcado, y luego, indicar qué ocurre cuando la casilla de verificación esta marcada.

3.1 Obtener el valor del checkbox

La obtención del estado de una casilla de verificación se realiza con el método **isChecked()** de la superclase **CompoundButton**. El retorno es booleano, donde true representa el estado *On* y false el estado *Off*.

A continuación se pone el código del MainActivity para mostrar el mensaje que tiene el string de la opción seleccionada (solo muestra la opción seleccionada, solo una).

```
1
2  public class MainActivity extends AppCompatActivity {
3
4      public Toast toast1;
5
6      @Override
7      protected void onCreate(Bundle savedInstanceState) {
8          super.onCreate(savedInstanceState);
9          setContentView(R.layout.activity_main);
10     }
11
12  // método para evaluar el checkbox onClick
13
14     public void controlcheckbox(View view) {
15         // £esta marcado?
16         boolean checked = ((CheckBox) view).isChecked();
17
18         // indicar que realiza cuando esta marcado o desmarcado
19         switch(view.getId()) {
20             case R.id.verificar1:
21                 if (checked) {
22                     displayToast(getString(R.string.texto_mostrar1));
23                 }
24                 break;
25             case R.id.verificar2:
26                 if (checked) {
27                     displayToast(getString(R.string.texto_mostrar2));
28                 }
29                 break;
30         }
31     }
32
33 //método para mostrar el mensaje
34
35     public void displayToast(String mensaje) {
```

```
        Toast toast1= Toast.makeText(getApplicationContext(), mensaje, Toast.LENGTH_SHORT);
        toast1.show();
    }
}
```

Vamos a modificar el ActivityMain para que muestre el mensaje de las casillas de verificación seleccionadas cuando se selecciona mas de una.

```
public class MainActivity extends AppCompatActivity {

    public Toast toast1;
    public String mensaje1;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    // metodo para evaluar el checkbox onClick
    public void controlcheckbox(View view) {
        // recoger los datos
        CheckBox nverificar1 = (CheckBox)findViewById(R.id.verificar1);
        CheckBox nverificar2 = (CheckBox)findViewById(R.id.verificar2);
        mensaje1="";
        // evaluar cada casilla

           if (nverificar1.isChecked())
                {
                    mensaje1= mensaje1 + getString(R.string.texto_mostrar1);
                }

           if (nverificar2.isChecked()) {
               mensaje1 = mensaje1 + getString(R.string.texto_mostrar2);
                }
       displayToast(mensaje1);
    }

\\método para mostrar el mensaje
    public void displayToast(String mensaje) {
        Toast toast1= Toast.makeText(getApplicationContext(), mensaje, Toast.LENGTH_SHORT);
        toast1.show();
    }
}
```

4 Métodos y clases utilizadas

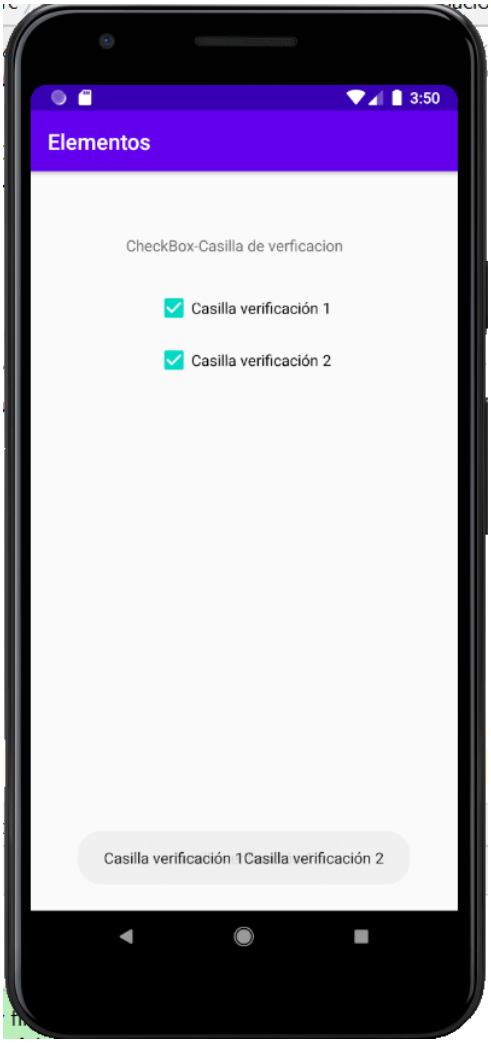
La clase que estamos utilizando es **CheckBox** que hereda de **CompoundButton**.

Api: `https://developer.android.com/reference/android/widget/CheckBox.html`

- **isChecked()** se va ha utilizar para conocer el estado del control y/o asignarle un estado concreto.
- **getId()** de la clase View permite saber el identificador asignado.
- **Concat()** o su equivalente "+" para unir String. Se puede utilizar n para realizar un "Enter".

5 Resultado

El resultado final es el siguiente:

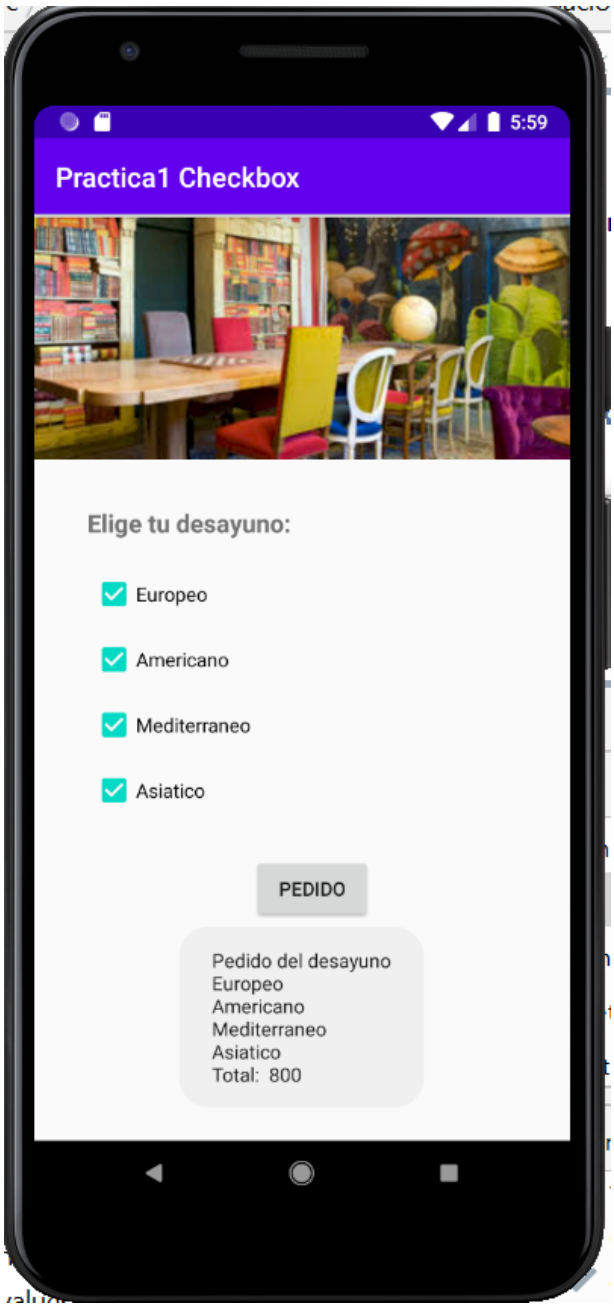


6 Propuestas de proyectos

6.1 Restaurante

Realizar un proyecto con imagen, texto, cuatro casillas de verificación y un botón. En este proyecto se seleccionaría un desayuno de los que nos ofrece la cafetería y se visualizaría los desayunos elegidos mas el importe.

El resultado final debe ser el siguiente.



En este proyecto se puede utilizar la clase **StringBuffer** con el método **append()**. Api: `https://developer.android.com/reference/java/lang/StringBuffer`

El funcionamiento del StringBuffer:

```
//crear un buffer de String
  StringBuffer result = new StringBuffer();
// Añadimos el primer texto al buffer
  result.append("Pedido del desayuno");
// añadir un segundo elemento (Enter)
 result.append("\n");
```