

Programación de eventos de componentes de interface de usuario

Les mando estas notas para entender mejor cómo es la programación de eventos de cada componente de usuario en Android. Entonces Recuerden:

- En el archivo layout se declara el identificador de cada componente como:

```
EditText android:id="@+id/TxtInput"
Button android:id="@+id/BtnInputLayout"
RadioGroup android:id="@+id/GrbGrupol"
RadioButton android:id="@+id/RbOpcion1"
CheckBox android:id="@+id/ChkMarcame"
```

y para accesarlo en el programa en Java se a través de la clase R y el id:

```
txtInput = (EditText) findViewById(R.id.TxtInput);
btnComprobar = (Button) findViewById(R.id.BtnInputLayout);
rgOpciones = (RadioGroup) findViewById(R.id.GrbGrupol);
rbOpcion1 = (RadioButton) findViewById(R.id.RbOpcion1);
rbOpcion2 = (RadioButton) findViewById(R.id.RbOpcion2);
cbMarcame = (CheckBox) findViewById(R.id.ChkMarcame);
```

y para acceder a este atributo dentro del mismo archivo layout u otro archivo xml es simplemente con `@id/<identif>`, por ejemplo:

```
<EditText
    android:id="@+id/prenomEdit"
    android:hint="@string/pass" />

<Button
    .....
    android:layout_below="@id/prenomEdit"
    android:text="@string/ok" />
```

y de manera similar con otros elementos que se encuentran en otros folders (Carpetas):

```
android:text="@string/ok" />
<string name="ok">Aceptar</string>
```

- ✚ En general la programación de eventos de componentes de interface de usuario es la siguiente:

- ✚ Declarar en la vista layout el componente de usuario y sus atributos:

- ✚ Edit Text:

```
<EditText android:id="@+id/TxtInput"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
```

```

        android:inputType="text"
        android:layout_below="@id/LblEtiqueta" />

```

Botones

```

<Button android:id="@+id/BtnInputLayout"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/TxtImagenHint"
        android:layout_alignParentRight="true"
        android:text="@string/negrita" />

```

RadioGroup – RadioButton

```

<RadioGroup android:id="@+id/GrbGrupo1"
        android:orientation="vertical"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

    <RadioButton android:id="@+id/RbOpcion1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/opcion_1" />

    <RadioButton android:id="@+id/RbOpcion2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/opcion_2" />

</RadioGroup>


```

CheckBox

```

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

```

-  Declarar en el programa Java (MainActivity) una variable para cada componente:

Edit Text

```

private EditText txtInput;

```

Botones

```
private Button btnNegrita;  
private Button btnComprobar;
```

RadioGroup – RadioButton


En un **RadioGroup** sólo se permite elegir sólo una opción.

```
private RadioGroup rgOpciones;  
private RadioButton rbOpcion1;  
private RadioButton rbOpcion2;
```

CheckBox

En un **CheckBox** se permite elegir varias opciones.

```
private CheckBox cbMarcame;
```

-  Asociar a la variable anterior, el componente de usuario de la vista a través del id y la clase R:

Edit Text

```
txtInput = (EditText) findViewById(R.id.TxtInput);
```

Botones

```
btnComprobar = (Button) findViewById(R.id.BtnInputLayout);
```

RadioGroup – RadioButton

```
rgOpciones = (RadioGroup) findViewById(R.id.GrbGrupo1);
```

```
rbOpcion1 = (RadioButton) findViewById(R.id.RbOpcion1);
```

```
rbOpcion2 = (RadioButton) findViewById(R.id.RbOpcion2);
```

CheckBox

```
cbMarcame = (CheckBox) findViewById(R.id.ChkMarcame);
```

-  Crear el método que se ejecutará cuando ocurra el evento del componente

Edit Text


En este caso, no tiene eventos pero para leer el valor tecleado por el usuario hacer lo siguiente:

```
txtInput.getText().toString();
```

Y para asignarle un valor:


```
txtInput.setText(nuevoTexto);
```

Botones

-  Crear una clase OnClickListener y sobreponer el método onClick que es la acción correspondiente del componente:

```
new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        String num = txtInput.getText().toString();  
  
        if(num.isEmpty() || Integer.parseInt(num)%2 != 0)  
            txtInputLayout.setError("Error: No es un  
                                   número par!");  
  
        else  
            txtInputLayout.setError(null);  
    }  
}
```

RadioGroup – RadioButton

-  Crear una clase OnClickListener y sobreponer el método onClick que es la acción correspondiente del componente:

```
View.OnClickListener list = new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        String opcion = "";  
        switch(view.getId()) {  
            case R.id.RbOpcion1:  
                opcion = "opción 1";  
                break;  
            case R.id.RbOpcion2:  
                opcion = "opción 2";  
                break;  
        }  
  
        lblMensaje.setText("ID opción seleccionada: " +  
opcion);  
    }  
};
```

Es decir, dentro de la definición del método `onClick` se verifica cual `RadioButton` del `RadioGroup` fue elegido mediante un switch a través

de la clase view con `view.getId()` y case `R.id.RbOpcion1`, `R.id.RbOpcion2`, etc.



Otro evento, que existe para el componente `RadioGroup – RadioButton` es cuando sufre un cambio `onCheckedChanged`:

```
rgOpciones.setOnCheckedChangeListener(new  
RadioGroup.OnCheckedChangeListener() {  
    public void onCheckedChanged(RadioGroup group, int  
checkedId) {  
  
        String opcion = "";  
        switch(checkedId) {  
            case R.id.RbOpcion1:  
                opcion = "opción 1";  
                break;  
            case R.id.RbOpcion2:  
                opcion = "opción 2";  
                break;  
        }  
  
        lblMensaje.setText("ID opción seleccionada: " +  
opcion);  
    }  
});
```

CheckBox

Cuando es un `CheckBox` se hace de manera similar a la de un botón:

```
cbMarcame.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View view) {  
        boolean isChecked = ((CheckBox)view).isChecked();
```

```

        if (isChecked) {
            cbMarcame.setText("Checkbox marcado!");
        }
        else {
            cbMarcame.setText("Checkbox desmarcado!");
        }
    }
});

```

verificando ahora que el `CheckBox` fue elegido mediante el método booleano:

```

((CheckBox)view).isChecked()

```

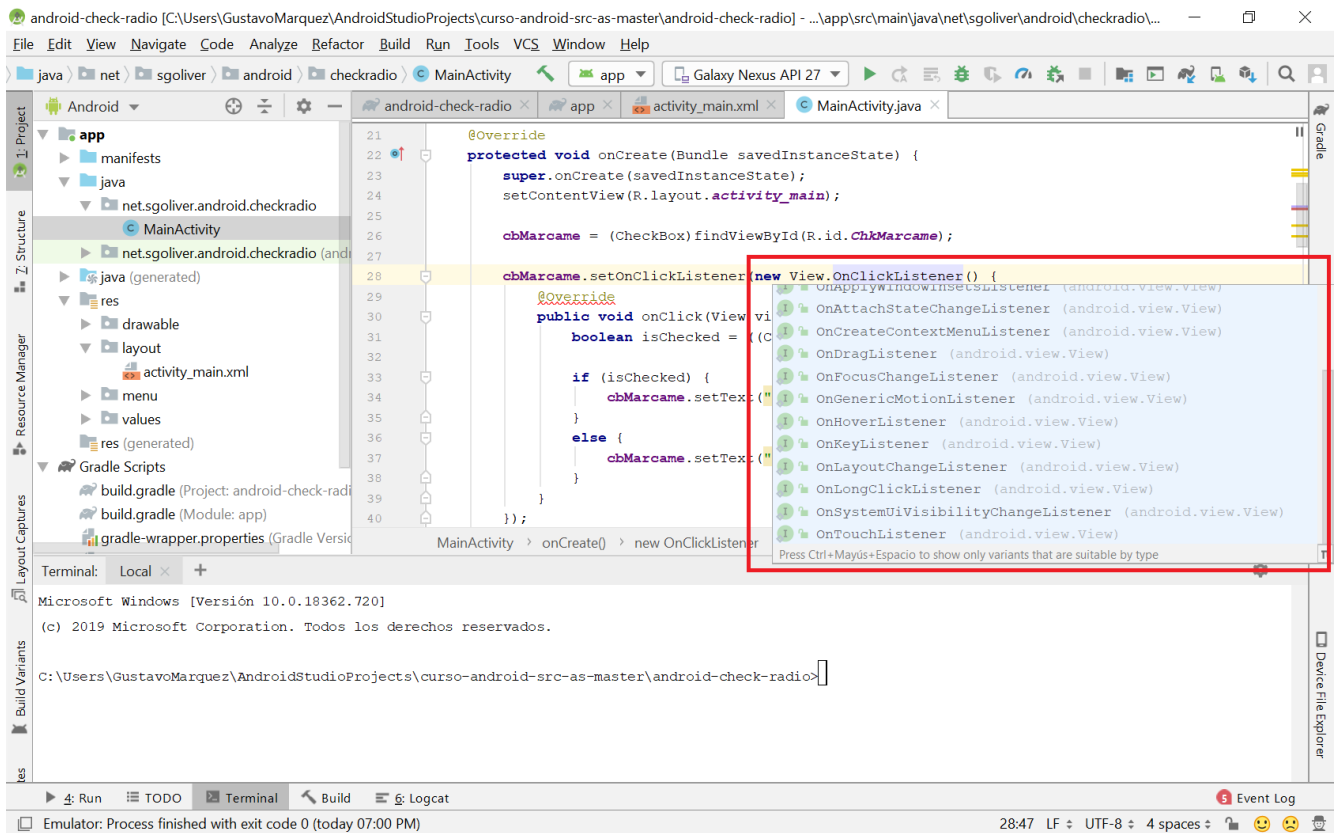
De manera similar al caso anterior, otro evento que existe para el componente checkBox es cuando sufre un cambio `onCheckedChanged`:

```

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

```

En el IDE se pueden ver otro tipo de métodos asociados a otro tipo de eventos:



En estos últimos casos, ver el programa android-check-radio

✚ Como último paso, asignar una instancia de la clase anterior a la variable declarada del componente:

✚ Botones

```

btnComprobar.setOnClickListener( <instancia de la clase
anterior, ver el ejemplo de RadioGroup – RadioButton> );

```

✚ RadioGroup – RadioButton

```

rbOpcion1.setOnClickListener(list);
rbOpcion2.setOnClickListener(list);

```

✚ CheckBox

En este caso se puede hacer al mismo tiempo que se declara el método que se ejecutará cuando ocurra el evento del componente:

```

cbMarcame.setOnClickListener(new View.OnClickListener() {
    @Override

```

```

public void onClick(View view) {
    boolean isChecked = ((CheckBox)view).isChecked();

    if (isChecked) {
        cbMarcame.setText("Checkbox marcado!");
    }
    else {
        cbMarcame.setText("Checkbox desmarcado!");
    }
}
});

```

Esta última técnica también se puede hacer emplear para los eventos de los botones y RadioGroup – RadioButton

La definición de la clase OnClickListener, OnCheckedChangeListener, se puede hacer:

- + En línea, como en los ejemplos anteriores.
 - + En forma separada en la misma clase
 - + En un archivo aparte
- + Algunos atributos adicionales de los componentes gráficos de usuario:

```

+ <TextView
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:textColor="#07F" // Azul cielo
    android:layout_height="48dp"
    android:fontFamily="sans-serif"
    android:gravity="center_vertical"
    android:textSize="25sp" />

+ <ListView android:id="@+id/left_drawer"
    android:layout_width="240dp"
    android:layout_height="match_parent"
    android:layout_gravity="start"
    android:choiceMode="singleChoice"
    android:divider="@android:color/transparent" // Separador
    // android:divider="@android:color/color/colo_holo_red_dark"

    android:orientation="vertical" // Puede ser horizontal
                                   (default)

    android:showDividers="middle"
    android:dividerHeight="1dp" // Ancho del separador
    android:background="#FFF"/>

```



```
android:layout_width="match_parent"  
android:layout_height="wrap_content"  
android:layout_gravity="top"  
android:background="#33b5e5" // Azul Cielo  
android:paddingBottom="4dp"  
android:paddingTop="4dp"  
android:textColor="#fff" /> // Blanco  
android:layout_marginRight="10dp"
```