# 1 Xestion de eventos III: Listeners e clases anónimas

## 1.1 Sumario



## 1.2 Introdución

- En Android hai moitas formas de interceptar eventos do usuario.
- A clase View proporciona varias formas de xestionar eventos.
- Por exemplo, cando un usuario toca un botón é chamado o método callback **onTouchEvent()** dese obxecto.
- Pero para interceptar ese evento debemos estender a clase e implementar o método.
- Pero non sería práctico para poder manexar o evento crear unha (sub)clase para cada obxecto Vista.
- Por iso a Clase Vista (View) contén unha colección de interfaces anidadas.
- Estas interfaces chámanse **Event Listeners (Escoitadores de eventos)** e están listas para capturar a iteración do usuario coa UI.

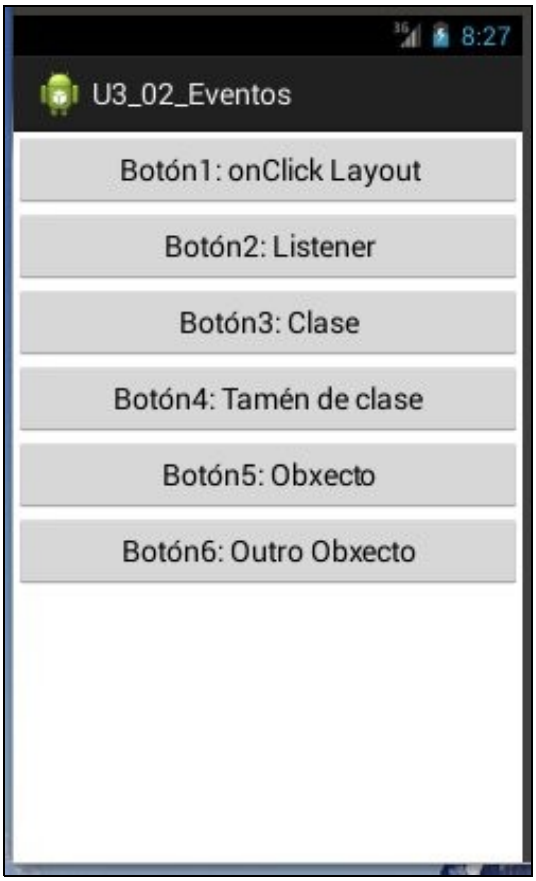
- A continuación imos ver algunhas formas de capturar eventos:

- **Referencias**
  - http://developer.android.com/guide/topics/ui/ui-events.html

## 1.3 Caso práctico

- Comezamos creando o proxecto: **U3_02_Eventos**
- Comezaremos capturado o evento **Click** e logo ao final veremos outros eventos.

- A aplicación coa que se traballará é a seguinte:




### 1.3.1 XML do Layout

- A liña 11 marca unha das formas máis sinxelas de capturar o evento Click sobre un compoñente.

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <Button
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Botón1: onClick Layout"
        android:onClick="onBotonClick"
         />

    <Button
        android:id="@+id/boton2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Botón2: Listener"
        />

<Button
        android:id="@+id/boton3"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Botón3: Clase"
        />

    <Button
        android:id="@+id/boton4"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Botón4: Tamén de clase"
```

```
        />

    <Button
        android:id="@+id/boton5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Botón5: Obxecto"
        />

    <Button
        android:id="@+id/boton6"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="Botón6: Outro Obxecto"
        />
</LinearLayout>
```

### 1.3.2 O Código Java

- A continuación preséntase o código Java, pero este vai ser debullado nos seguintes apartados.

```
package com.example.u3_02_eventos;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class U3_02_Eventos extends Activity {

@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_u3_02__eventos);

Button boton2 = (Button) findViewById(R.id.boton2);
boton2.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
Toast.makeText(getApplicationContext(), "Premeches o Botón2", Toast.LENGTH_SHORT).show();
}
});

Button boton3 = (Button) findViewById(R.id.boton3);
boton3.setOnClickListener(new XestionEventos());

Button boton4 = (Button) findViewById(R.id.boton4);
boton4.setOnClickListener(new XestionEventos());

Button boton5 = (Button) findViewById(R.id.boton5);
boton5.setOnClickListener(_OnClickListener);

Button boton6 = (Button) findViewById(R.id.boton6);
boton6.setOnClickListener(_OnClickListener);

}

private OnClickListener _OnClickListener = new OnClickListener() {

@Override
public void onClick(View v) {
Button btn = (Button) v;
Toast.makeText(v.getContext(), "Premeches" + btn.getText(), Toast.LENGTH_SHORT).show();
```

```
}
};

public void onBotonClick(View v) {
Toast.makeText(this, "Premeches o Botón1", Toast.LENGTH_SHORT).show();
}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
// Inflate the menu; this adds items to the action bar if it is present.
getMenuInflater().inflate(R.menu.u3_02__eventos, menu);
return true;
}

}
```

### 1.3.3 Propiedade android:onClick

- Como xa se viu é unha propiedade XML que teñen varios compoñentes (vistas).
- Hai que definir un método:
    - ♦ Que sexa public
    - ♦ Que non devolva nada: void
    - ♦ Que teña un parámetro da clase View, que será quen provoque o evento.
- No exemplo anterior son as liñas 50-52:

```
public void onBotonClick(View v) {
Toast.makeText(this, "Premeches o Botón1", Toast.LENGTH_SHORT).show();
}
```

## 1.4 Listener

- Un **Event Listener** é unha interface da clase Vista (View) que contén un único método de tipo callback que hai que implementar
- Android invoca estes métodos cando a Vista detecta que o usuario está provocando un tipo concreto de interacción con ese elemento da interface de usuario.

Existen os seguintes métodos callback:

- ♦ **onClick()**: de **View.OnClickListener**. Este método invócase cando o usuario toca un elemento cun dedo (modo contacto), fai click coa bola de navegación (TrackBall) do dispositivo ou preme a tecla "Enter" estando nun compoñente.
- ♦ **onLongClick()**: de **View.OnLongClickListener**. Este método chámase cando o usuario toca e mantén o dedo sobre un elemento (modo de contacto), fai click sen soltar coa bola de navegación (TrackBall) ou preme a tecla "Enter" perante un segundo estando nun elemento.
- ♦ **onFocusChange()**: de **View.OnFocusChangeListener**. Invócase cando o usuario move o cursor cara unha Vista ou se alonxa desta utilizando a bola de navegación ou usando as teclas de navegación.

**onKey()**: de **View.OnKeyListener**. Chámase cando o usuario se centra nun elemento e preme ou libera una tecla do dispositivo.

- ♦ ...
- **onLongClick()** verase nun exemplo posterior.

- Estes métodos son os únicos que se van implementar nas súas respectivas interfaces.
- Estas interfaces teñen o formato: **on...Listener()**
- Unha vez que temos implementada a interface témoslla que pasar como parámetro á vista (view) correspondente a través de **vista.set...Listener()**.

- Lembrar que en java os interfaces son clases abstractas que so definen os atributos e métodos (cos parámetros) que vai ter esa clase pero non os implementa. Por tanto sempre que se implemente unha interface hai que implementar os métodos que define.
- Neste caso a interface View.OnClickListener está definida como segue: http://developer.android.com/reference/android/view/View.OnClickListener.html

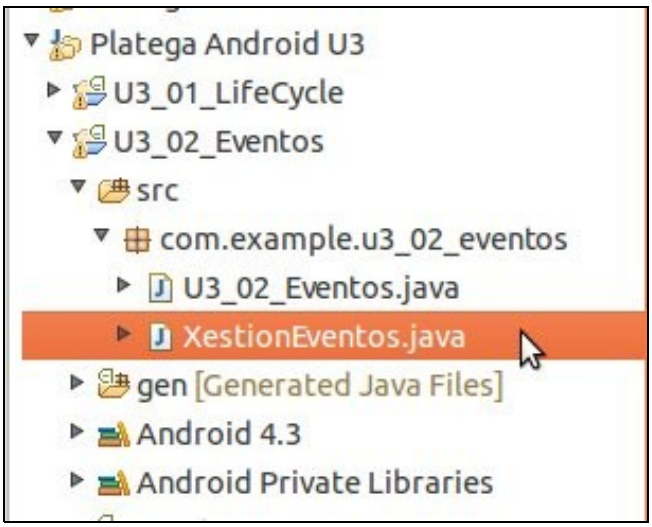
```
public static interface View.OnClickListener {
public abstract void onClick (View v);
```

```
}
```

- Neste exemplo, cando implementemos a interface **View.OnClickListener** debemos implementar o método **onClick**.
- A continuación, a modo de exemplo, imos ver 3 formas de implementar esta interface. O que se faga con esta pódese facer dun xeito semellante con calquera outra.

### 1.4.1 Implementar a interface a través dunha clase

- Este é o método máis sinxelo, para entender. Pero obríganos a crear unha nova clase.
- Creamos unha nova clase que implemente o interface **OnClickListener**.




- Código Java da clase: **XestionEventos**

```
package com.example.u3_02_eventos;

import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class XestionEventos implements OnClickListener {

@Override
public void onClick(View v) {
// TODO Auto-generated method stub

Button btn = (Button) v;
Toast.makeText(v.getContext(), "Premeches "+btn.getText(), Toast.LENGTH_SHORT).show();
}
}
```

- Observar nas **Liñas 28 e 31** (do código principal) como se crea unha nova clase XestionEventos para cada botón.
- Esa clase é a que vai implementar o método onClick(), como se ve enriba.

```
Button boton3 = (Button) findViewById(R.id.boton3);
boton3.setOnClickListener(new XestionEventos());

Button boton4 = (Button) findViewById(R.id.boton4);
boton4.setOnClickListener(new XestionEventos());
```

- Pero esta forma é un *engorro*, pois hai que estar creando clases.

### 1.4.2 Crear un obxecto que implemente a interface

- Imos crear un obxecto que implemente a interface e que poida ser pasado como parámetro as vistas que desexemos cada vez que se fai click nelas.

- **_OnClickLister** é un obxecto de tipo OnClickListener. **Liñas 41-48** de arriba.

```
private OnClickListener _OnClickListener = new OnClickListener() {

@Override
public void onClick(View v) {
Button btn = (Button) v;
Toast.makeText(v.getContext(), "Premeches" + btn.getText(), Toast.LENGTH_SHORT).show();
}
};
```

- Pasamos ese obxecto ás vistas que desexemos cando se faga Click nelas.

```
Button boton5 = (Button) findViewById(R.id.boton5);
boton5.setOnClickListener(_OnClickListener);

Button boton6 = (Button) findViewById(R.id.boton6);
boton6.setOnClickListener(_OnClickListener);
```

### 1.4.3 A través dunha clase anónima

- Exemplo: **Liñas 18-25** do código
- Se no canto de crear un obxecto que implemente a interface e logo pasarlle ese obxecto como parámetro ao escoitador asociado ao botón podemos pasarlle como parámetro ao escoitador o código que implementa o obxecto.
- Deste xeito, non se crea nin unha clase nin un obxecto de modo explícito ao que se lle asocia un nome. Senón que se crea un obxecto de tipo OnClickListener ao que non se lle asocia ningún nome, de aí de **anónimo**.

```
Button boton2 = (Button) findViewById(R.id.boton2);
boton2.setOnClickListener(new OnClickListener() {

@Override
public void onClick(View v) {
Toast.makeText(getApplicationContext(), "Premeches o Botón2", Toast.LENGTH_SHORT).show();
}
});
```