Tabla de Contenidos

Android es una plataforma que es bastante mas peculiar de lo que parece al principio y permite hacer lo mismo de muchas formas distintas . Esto muchas veces la convierte en una plataforma algo caótica ya que cuando buscas algo por internet salen 5 soluciones diferentes y no te queda claro cual es la correcta. Vamos a ver esto con Android y la gestión de eventos. Para ello vamos a desarrollar una aplicación muy sencilla que tiene tres botones. Lo único que queremos es que cuando pulsemos un botón salga su nombre por pantalla. No es la aplicación “holamundo” pero se acerca.

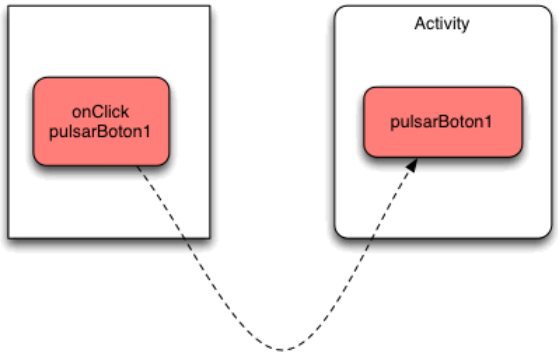
Android Eventos (Sencillo)

Esto en principio es algo facil de conseguir simplemente vamos al layout que tenemos (activity_main.xml) y añadimos un evento de click a cada boton

```
<LinearLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical" >

<Button
 android:id="@+id/boton1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="@string/boton_1"
 android:onClick="pulsarBoton1" />
 <Button
 android:id="@+id/boton2"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="@string/boton_2"
 android:onClick="pulsarBoton2"
 />

 <Button
 android:id="@+id/boton3"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="@string/boton_3"
 android:onClick="pulsarBoton3"

 />

</LinearLayout>
```

El código simplemente añade un evento “onClick” a cada uno de los botones y asocia este evento a una función para cada boton que estará ubicada en la actividad,



Así pues si mostramos el código de la actividad tendremos que tener tres funciones cada una ligada a un botón.

```
package com.arquitectura.blog001;

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

public class MainActivity extends Activity {

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

@Override
```

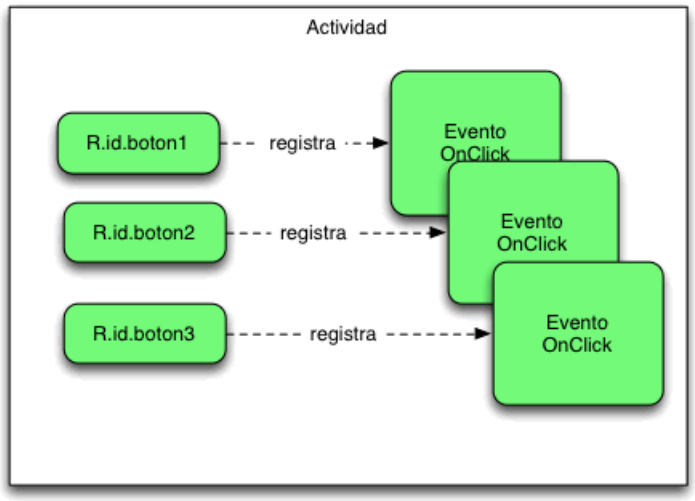
```
 public boolean onCreateOptionsMenu(Menu menu) {
 // Inflate the menu; this adds items to the action bar if it is
present.
 getMenuInflater().inflate(R.menu.main, menu);
 return true;
 }

public void pulsarBoton1(View vista) {

Toast.makeText(this, "boton1", Toast.LENGTH_SHORT).show();

}

public void pulsarBoton2(View vista) {

Toast.makeText(this, "boton2", Toast.LENGTH_SHORT).show();

}

public void pulsarBoton3(View vista) {

Toast.makeText(this, "boton3", Toast.LENGTH_SHORT).show();

}
}
```

Con este código es suficiente para que cada botón imprima su propio mensaje. Ahora bien ¿es la mejor opción? . La realidad es que es una opción sencilla pero problemática ya que si definimos los eventos en el layout pero no las las funciones en la Actividad el código compilará sin ningún problema y se desplegará y únicamente fallara en tiempo de ejecución cuando se percate de que las funciones no existen. Esto no es lo que queremos aunque la

solución es sencilla. No es la mas adecuada ya que generará problemas a los desarrolladores.

Android Eventos (Compilación)

Si queremos tener la seguridad de que los eventos están correctamente ligados a controles deberemos realizar la gestión a traves de la clase R localizando cada control y registrando los eventos.



Vamos a ver el código fuente :

```
package com.arquitectura.blog001;

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 MainActivity extends Activity {
 Button boton1,boton2,boton3;

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

 boton1= (Button) findViewById(R.id.boton1);
 boton1.setOnClickListener(new OnClickListener() {

 @Override
 public void onClick(View v) {

 Toast.makeText(getApplicationContext(), "boton1",
Toast.LENGTH_SHORT).show();
 }
 });

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

 @Override
 public void onClick(View v) {

 Toast.makeText(getApplicationContext(), "boton2",
Toast.LENGTH_SHORT).show();
 }
```

```
 });

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

 @Override
 public void onClick(View v) {

 Toast.makeText(getApplicationContext(), "boton3",
Toast.LENGTH_SHORT).show();
 }
 });

 }
}
```

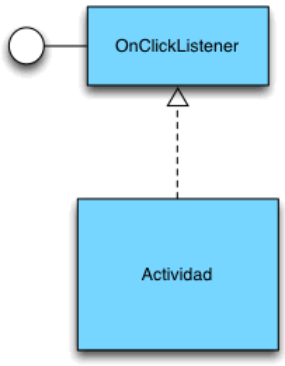
Hemos usado el método setOnClickListener y registrado un evento para cada uno de los botones con lo que en Java se denomina una clase Anónima. Esto permitirá eliminar los atributos onClick del layout.

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

<Button
 android:id="@+id/boton1"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
```

```
 android:text="@string/boton_1"
 />

 <Button
 android:id="@+id/boton2"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="@string/boton_2"

 />

 <Button
 android:id="@+id/boton3"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:text="@string/boton_3"

 />

</LinearLayout>
<pre>
```

Android Eventos (Practicidad)

La opción que acabamos de mostrar da menos problemas ya que chequea todo en tiempo de compilación pero hay que reconocer que es bastante engorrosa ya que genera mucho código (uso de clases anónimas). ¿Como podemos solventar los problemas que tenemos? . La mejor opción en este caso es obligar a que nuestra propia actividad implemente el interface OnClickListener directamente . De esta forma no dependeremos de las clases Anónimas y registraremos como listener para todos los eventos nuestra propia Actividad.

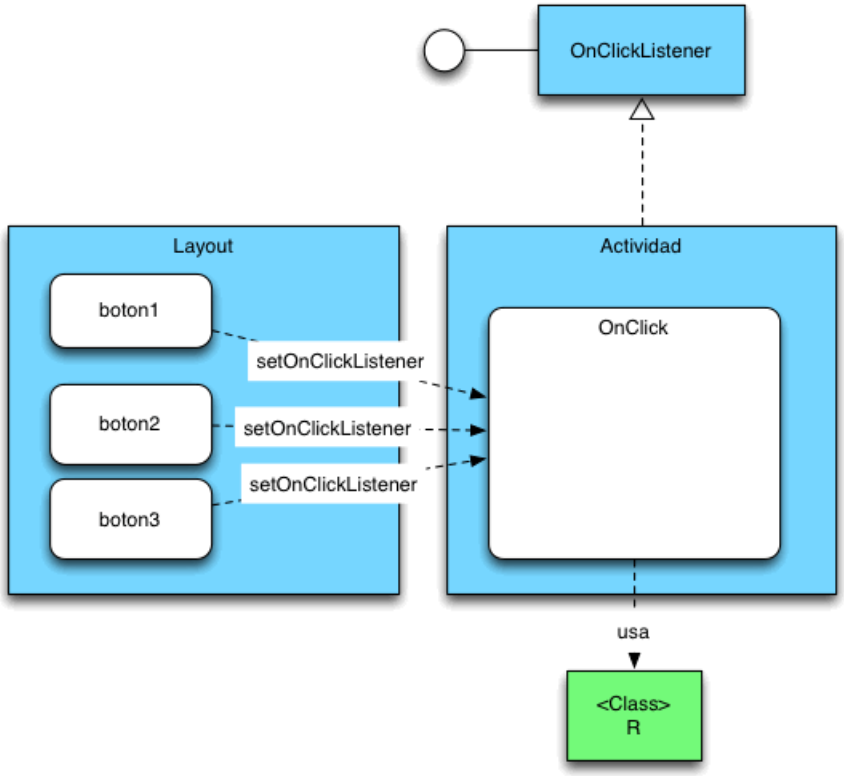


Vamos a verlo en código

```
package com.arquitectura.blog001;

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 MainActivity extends Activity implements OnClickListener{
 Button boton1,boton2,boton3;

 @Override
 protected void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 setContentView(R.layout.activity_main);
 boton1=(Button)findViewById(R.id.boton1);
```

```
 boton1.setOnClickListener(this);
 boton2=(Button)findViewById(R.id.boton2);
 boton2.setOnClickListener(this);
 boton3=(Button)findViewById(R.id.boton3);
 boton3.setOnClickListener(this);
 }

@Override
 public void onClick(View v) {

 if (v.getId()==R.id.boton1) {

 Toast.makeText(getApplicationContext(), "boton1",
Toast.LENGTH_SHORT).show();
 }else if (v.getId()==R.id.boton2) {

 Toast.makeText(getApplicationContext(), "boton2",
Toast.LENGTH_SHORT).show();

}else {

 Toast.makeText(getApplicationContext(), "boton3",
Toast.LENGTH_SHORT).show();
 }

 }
}
```

Como podemos ver usamos la clase R (Recursos) para dilucidar que boton se ha pulsado concretamente a traves del objeto view que contiene el id del control pulsado.El siguiente diagrama intenta terminar por aclarar la relación entre todos los elementos.



Estas son las tres opciones principales que Android soporta para la gestión de eventos con sus puntos fuertes y débiles yo en la mayoría de los casos apuesto por la última opción.