

# Aplicaciones Telemáticas

## Tema 2.3. Control de la interfaz de usuario (UI) en Android

J. E. López Patiño, F. J. Martínez Zaldívar

ETSIT-UPV

- 1 Introducción
- 2 Controles de texto y botón
  - Implementando interfaz `View.OnClickListener`
  - Otros controles
- 3 Guardando y restaurando el estado de la UI
- 4 Lanzando actividades
  - Actividad lanzando otra actividad
- 5 Menús de aplicación
- 6 Listas y Selección
  - Listas con `ListView`
  - Listas con `RecyclerView`
  - Selección con menús desplegables con `Spinner`

# Índice

## 1 Introducción

# Introducción

## Control del interfaz:

- Controles básicos:
  - TextView
  - Button
  - ToggleButton
  - CheckBox
  - RadioButton
- Arranque de actividades
- Menús de aplicación
- Controles de listas
  - Adaptadores
  - ListView
  - GridView
  - Spinner

Referencia: <http://www.sgoliver.net/blog/>

# Índice

## 2 Controles de texto y botón

- Implementando interfaz `View.OnClickListener`
- Otros controles

# Índice

- 2 Controles de texto y botón
  - Implementando interfaz `View.OnClickListener`
  - Otros controles

# Registro de función de *callback*

- ¿Qué significa `button.setOnClickListener( objeto )`?  
`setOnClickListener()`: método público de la clase `View`

`java.lang.Object`  $\Rightarrow$  `android.view.View`  $\Rightarrow$  `android.widget.TextView`  $\Rightarrow$  `android.widget.Button`

```
void setOnClickListener(View.OnClickListener l)
```

Register a callback to be invoked when this view is clicked.

El argumento de `setOnClickListener` debe implementar la interfaz `View.OnClickListener`; en este caso: `objeto`

# Help...

```
import android.view.View;
... implements View.OnClickListener {
```

The screenshot shows the Android Developers website in a Chromium browser. The page title is "View.OnClickListener | Android Developers - Chromium". The URL is "developer.android.com/reference/android/view/View.OnClickListener.html". The page has a navigation bar with "Developers", "Design", "Develop", and "Distribute" tabs. Below the navigation bar, there are tabs for "Training", "API Guides", "Reference", "Tools", "Google Services", and "Samples". The "Reference" tab is selected, and the "API Guides" sub-tab is active. The main content area shows the documentation for "View.OnClickListener". It includes a "Summary" section with "Public Methods" and "Class Overview". The "Public Methods" section lists the "onClick" method, which is an abstract void method that is called when a view has been clicked. The "Class Overview" section provides a brief description of the interface. The left sidebar contains a list of API levels and a list of classes, including "View.OnClickListener".

View.OnClickListener | Android Developers - Chromium

developer.android.com/reference/android/view/View.OnClickListener.html

Developers Design Develop Distribute

Training API Guides Reference Tools Google Services Samples

Android APIs API level: 19

android.telephony.gsm  
android.test  
android.test.mock  
android.test.suitebuilder  
android.text  
android.text.format  
android.text.method  
android.text.style  
android.text.util  
android.transition  
android.util  
**android.view**  
android.view.accessibility  
android.view.animation

Menu  
MenuItem  
MenuItem.OnActionExpandLi  
MenuItem.OnMenuItemClickLi  
ScaleGestureDetector.OnScale  
SubMenu  
SurfaceHolder  
SurfaceHolder.Callback  
SurfaceHolder.Callback2  
TextureView.SurfaceTextureLi  
View.OnAttachStateChangeLi  
**View.OnClickListener**  
View.OnCreateContextMenuLi  
View.OnDragListener  
View.OnFocusChangeListener  
View.OnGenericMotionListene  
View.OnHoverListener  
View.OnKeyListener  
View.OnLayoutChangeListener  
View.OnLongClickListener  
View.OnSystemUiVisibilityCh

Use Tree Navigation

public static interface  
**View.OnClickListener**  
Summary: Methods | [Expand All]  
Added in API level 1

android.view.View.OnClickListener

Known Indirect Subclasses  
CharacterPickerDialog, KeyboardView, QuickContactBadge

**Class Overview**

Interface definition for a callback to be invoked when a view is clicked.

**Summary**

Public Methods

abstract void **onClick (View v)**  
Called when a view has been clicked.

**Public Methods**

public abstract void **onClick (View v)**  
Added in API level 1  
Called when a view has been clicked.

**Parameters**

v The view that was clicked.

Except as noted, this content is licensed under [Apache 2.0](#). For details and restrictions, see the [Content License](#).

Android 4.4 (l) — 16 Jan 2014 2:24  
[About Android](#) | [Legal](#) | [Support](#)



# Algunas posibilidades

- Repositorio:

<https://github.com/AATT-ETSIT/U2T3-Ej1-OnClickListener.git>

- Referencias a etiquetas:

- v1.0: argumento de `.setOnClickListener( · )` es un objeto de clase que implementa interfaz `View.OnClickListener`
- v1.1: tanto el objeto anterior como la clase a la que pertenece son anónimos
- v2.0: ahora es `MainActivity` quien implementa la interfaz `View.OnClickListener`
- v3.0: alternativa indicando código a ejecutar en fichero XML
- v4.0: utilización de `vista.getId()` para obtener referencia `R.id.identificador...`

# Índice

## 2 Controles de texto y botón

- Implementando interfaz `View.OnClickListener`
- Otros controles

# Button, CheckBox, RadioButton, ToggleButton, EditText...

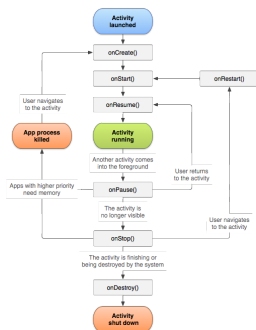
- Acceso a información y eventos de estos objetos de la clase View
- Ejemplo, repositorio:  
<https://github.com/AATT-ETSIT/U2T3-Ej2-TextoYBotones.git>

# Índice

## 3 Guardando y restaurando el estado de la UI

# Estados de una actividad

- Estados:



- Cambio de orientación o trabajo en multi-ventana:

- Algunos elementos de UI no se conservan: contenidos de `EditText`, `TextView` modificados, ...

- Solución:

- Guardar información para restaurar posteriormente dicho estado
- Esta información puede ser tipos primitivos u objetos (`Serializable/Parcelable`)

# Guardando estado UI

- Invocado antes/después de `onStop()`: depende de versión de Android

```
static final String PUNTOS = "PUNTOS";
static final String NIVEL = "NIVEL";
// ...

@Override
public void onSaveInstanceState(Bundle savedInstanceState) {
    // Save the user's current game state
    savedInstanceState.putInt(PUNTOS, puntuacion);
    savedInstanceState.putInt(NIVEL, nivel_juego);

    // Always call the superclass so it can save the view hierarchy state
    super.onSaveInstanceState(savedInstanceState);
}
```

# Restaurando estado UI

- Posibilidades:

- Desde onCreate

```
int puntuacion;
int nivel_juego;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState); // Always call the superclass first

    // Check whether we're recreating a previously destroyed instance
    if (savedInstanceState != null) {
        // Restore value of members from saved state
        puntuacion = savedInstanceState.getInt(PUNTOS);
        nivel_juego = savedInstanceState.getInt(NIVEL);
    } else {
        // Probably initialize members with default values for a new instance
        puntuacion = 0;
        nivel_juego = 1;
    }
    // ...
}
```

- Desde onRestoreInstanceState: no es necesario comprobar si savedInstanceState!=null. Invocado tras onStart()

```
public void onRestoreInstanceState(Bundle savedInstanceState) {
    // Always call the superclass so it can restore the view hierarchy
    super.onRestoreInstanceState(savedInstanceState);

    // Restore state members from saved instance
    puntos = savedInstanceState.getInt(PUNTOS);
    nivel_juego = savedInstanceState.getInt(NIVEL);
}
```

- URL: <https://github.com/AATT-ETSIT/U2T3-Ej-EstadoUI.git>

# Índice

- 4 Lanzando actividades
  - Actividad lanzando otra actividad



# Índice

- 4 Lanzando actividades
  - Actividad lanzando otra actividad

# Actividad $\Rightarrow$ Actividad: esquema general

- Actividad lanzadora:

MainActivity.java

```
...

public class MainActivity extends ...

...

Intent intento = new Intent( MainActivity.this, ActividadLanzada.class );

Bundle b = new Bundle();
b.putString("NOMBRE_DEL_STRING", "Un string" );
b.putInt("IDENTIFICADOR_DEL_INT", 7 );
...
intento.putExtras( b );

startActivity( intento );
...
```

- Actividad lanzada

ActividadLanzada.java

```
...

public class ActividadLanzada extends ...

...
Bundle bundle = getIntent().getExtras();
String unString = bundle.getString("NOMBRE_DEL_STRING");
int x = bundle.getInt("IDENTIFICADOR_DEL_INT");
...
```

# Consideraciones en AndroidManifest.xml

- Es necesario declarar la actividad en AndroidManifest.xml (puede hacerlo el IDE automáticamente)

AndroidManifest.xml

```
...  
<activity android:name=".ActividadLanzada" />  
...
```

- Si deseamos boton UP, entonces tenemos que indicar hacia dónde queremos volver:

AndroidManifest.xml

```
...  
<activity android:name=".ActividadLanzada"  
    android:parentActivityName=".MainActivity" >  
    <!-- The meta-data tag is required if you support API level 15 and lower -->  
    <meta-data  
        android:name="android.support.PARENT_ACTIVITY"  
        android:value=".MainActivity" />  
</activity>  
...
```

# Ejemplos con actividades

- URL: <https://github.com/AATT-ETSIT/U2T3-Ej3-A2A.git>
- v1.0: actividad lanzando otra actividad enviando parámetros en un bundle.
- v1.1. Indicación en `AndroidManifest.xml` de actividad padre: Boton UP. Obsérvese diferencia con boton BACK (creación de nueva actividad/recuperación sin destrucción)
- v1.2. Similar con tres actividades y distintas actividades padres (`AndroidManifest.xml`)
- v1.3. Similar, pero haciendo que UP se comporte como BACK.

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home: // UP pulsado
            onBackPressed(); // Acción BACK
            return true;
    }
    return(super.onOptionsItemSelected(item));
}
```

# Actividad llamando a actividad y esperando resultados

- Actividad lanzadora:

```
...
int CODIGO_PETICION = 5;
Intent intento = new Intent( MainActivity.this, ActividadLanzada.class );

Bundle b = new Bundle();
b.putString("NOMBRE_DEL_STRING", "Un string" );
b.putInt("IDENTIFICADOR_DEL_INT", 7 );
...
intento.putExtras( b );

startActivityForResult( intento, CODIGO_PETICION );
}

@Override
protected void onActivityResult( int codigoPeticion, int codigoResultado, Intent intento ) {
    if ( codigoPeticion == CODIGO_PETICION ) {
        if ( codigoResultado == Activity.RESULT_OK ) {
            int valor_por_defecto = 0;
            String string_por_defecto = "";
            int valor = intento.getIntExtra("RESULTADO.1", valor_por_defecto);
            String cadena = intento.getStringExtra("CADENA", string_por_defecto );
        } else if ( codigoResultado == Activity.RESULT_CANCELED ) {
            ...
        }
    }
}
```

# Consdieraciones (cont.)

- Actividad lanzada:

```
...  
  
Bundle bundle = getIntent().getExtras();  
  
String nombreStr = bundle.getString("NOMBRE_DEL_STRING");  
int x = bundle.getInt("IDENTIFICADOR_DEL_INT");  
  
...  
}  
  
// Código para retornar  
{  
    ...  
    Intent intentoRetorno = new Intent();  
    // Alternativa al empleo de bundle  
    intentoRetorno.putExtra( "ANYOS", anyos );  
    setResult( Activity.RESULT_OK, intentoRetorno );  
    finish();  
}
```

# Ejemplos con actividades, esperando resultados

- URL: <https://github.com/AATT-ETSIT/U2T3-Ej4-A2A-results>
- v1.0: actividad llamando a actividad y esperando resultados
- v1.1: adición de botón UP (sin efecto, salvo vuelta atrás)
- v1.2: con efecto equivalente a validación.
- v1.3: con cancelación explícita
- v1.4: con distintos códigos (distintas actividades llamadas)

# Índice

## 5 Menús de aplicación



# Consideraciones

- Sugerencia de plantilla: *Basic Activity* (no *Empty Activity*)
- URL: <https://github.com/AATT-ETSIT/U2T3-Ej5-Menus.git>
- En la plantilla del proyecto aparecen varios *layout* incluyendo uno al otro
- Aparece también el fichero `menu_main.xml` en directorio `menu`
- Etiqueta v1.0: situación inicial con plantilla
- Etiqueta v1.1: adición de algunas variaciones de menú.
- Probar con plantillas *“Navigation Drawer Activity”* y *“Bottom Navigation Activity”*

## 6 Listas y Selección

- Listas con `ListView`
- Listas con `RecyclerView`
- Selección con menús desplegables con `Spinner`

# Índice

## 6 Listas y Selección

- Listas con ListView
- Listas con RecyclerView
- Selección con menús desplegables con Spinner

# Ejemplo: ListView con Strings

- Consideraciones para ListView:

- Ubicación del elemento ListView en el layout correspondiente
- Generación de Strings literalmente o bien en XML
- Elección del *adaptador*
- URL:

<https://github.com/AATT-ETSIT/U2T3-Ej6-ListView.git>

strings.xml

```
...  
<!-- Alternativa a String [] de Java -->  
<string-array name="dias_laborables">  
  <item>Lunes</item>  
  <item>Martes</item>  
  <item>Miércoles</item>  
  <item>Jueves</item>  
  <item>Viernes</item>  
</string-array>  
...
```



# Índice

## 6 Listas y Selección

- Listas con ListView
- **Listas con RecyclerView**
- Selección con menús desplegables con Spinner

# Consdieraciones para RecyclerView

- Ubicación del elemento `RecyclerView` en el layout correspondiente
- Creación de un *layout* correspondiente a un único elemento de la lista (este será replicado)
- Opcionalmente, crear una clase que se corresponda con un elemento a visualizar
- Creación del adaptador extendiendo `RecyclerView.Adapter< · >`
- Integración de los datos, el adaptador y el `RecyclerView`.
- URL:  
<https://github.com/AATT-ETSIT/U2T3-Ej7-RecyclerView.git>

# Índice

## 6 Listas y Selección

- Listas con `ListView`
- Listas con `RecyclerView`
- Selección con menús desplegables con `Spinner`

# Consdieraciones para Spinner simple

- Ubicación del elemento Spinner en el layout correspondiente
- Peculiaridades del adaptador y de proceso de selección
- URL: <https://github.com/AATT-ETSIT/U2T3-Ej8-Spinner.git>

