Aplicaciones Telemáticas

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

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

ETSIT-UPV

Índice

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

Índice

Registro de función de *callback*

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

`java.lang.Object` ⇒ `android.view.View` ⇒ `android.widget.TextView` ⇒ `android.widget.Button`

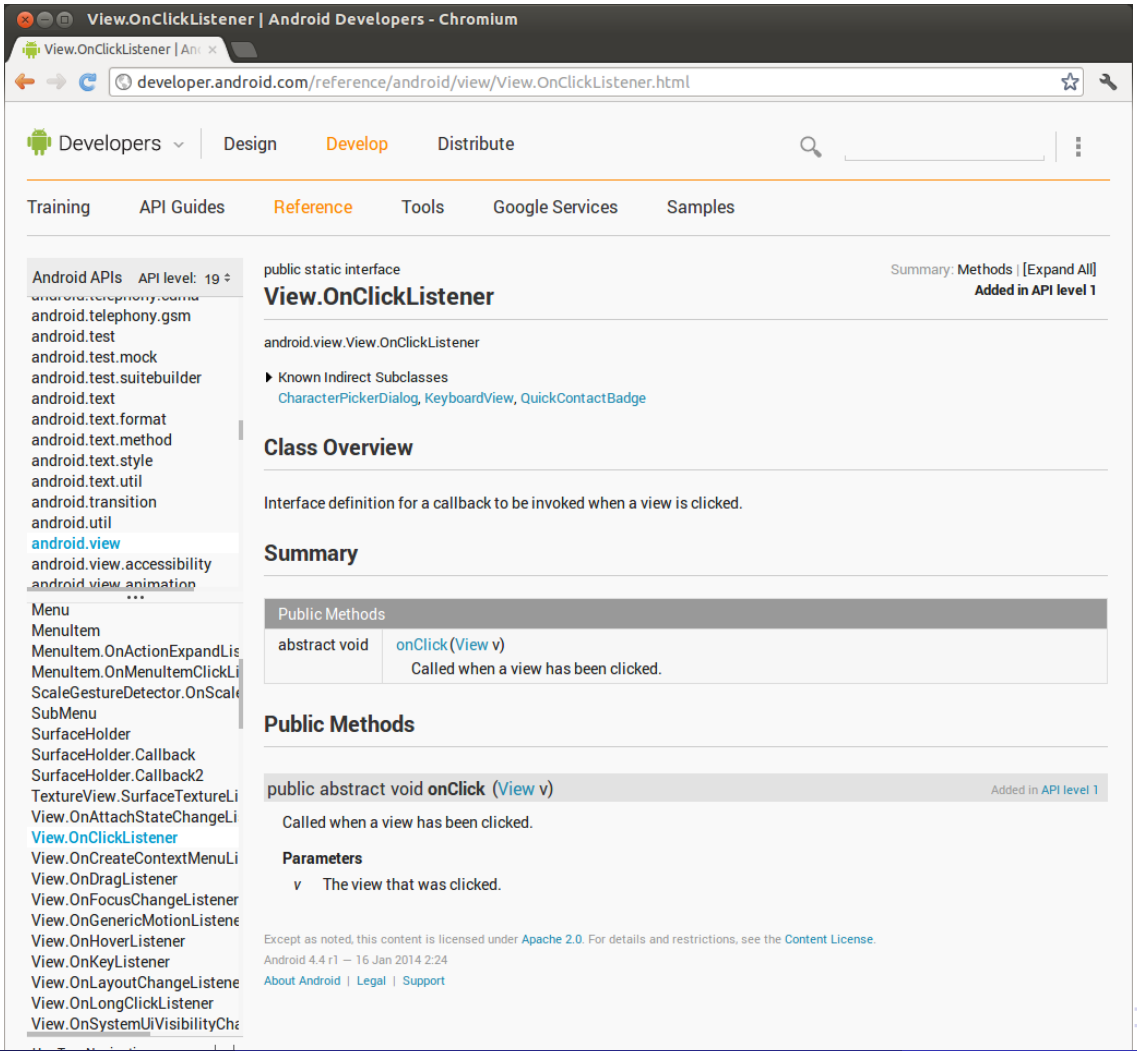
```
void setOnClickListener(View.OnClickListener l)

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

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

Help. . .

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



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

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

Índice

Actividad ⇒ 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 activivdad en `AndroidManifest.xml` (puede hacerlo el IDE automáticametne)

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 activida 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();
}
```

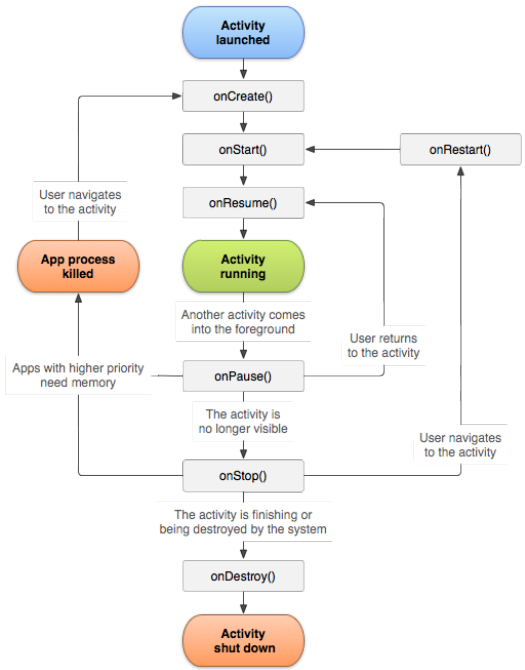
Ejemplos con actividades, esperando resultados

- 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

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/despues 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

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"*

Índice

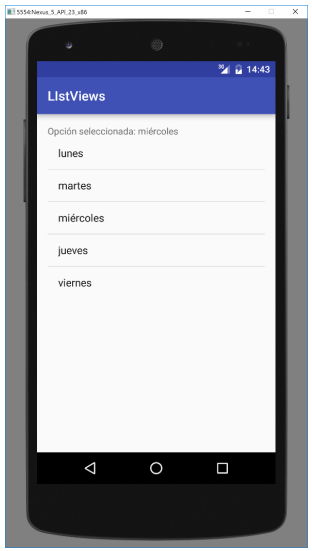
Índice

Ejemplo: ListView con Strings

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>
...
```

- Consdieraciones 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`



Índice

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`$<\cdot>$
- Integración de los datos, el adaptador y el `RecyclerView`.
- URL:
  `https://github.com/AATT-ETSIT/U2T3-Ej7-RecyclerView.git`

Índice

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`