


Open · Opened 3 days ago by  **Rubén Montero**

Actividad que muestra vídeos

Resumen

- Añadiremos una nueva actividad `VideoActivity`. De momento, sólo recibirá dos datos: `id` y `url` del vídeo
- Los mostrará en un `Toast`

Descripción

Ya que podemos hacer *click* en cada celda y mostrar información asociada, pasaremos dicha información a una nueva actividad.

Hasta ahora no hemos *pasado datos de una actividad a otra*. ¡Pero se puede hacer!

Basta con usar `putExtra` cuando lanzamos la actividad y `getXXExtra` en la nueva actividad.

Un detalle: Tipo de dato recuperado en la actividad *destino*

Mientras `putExtra` es el *mismo* método independientemente del tipo de dato (`int`, `String`, `boolean`, ...) que queramos pasar, `getXXExtra` tiene nombres distintos.

Debemos saber *qué* tipo de dato queremos recuperar y emplear el método apropiado¹:

- `getIntExtra`
- `getLongExtra`
- `getFloatExtra`
- `getDoubleExtra`
- `getStringExtra`
- `getStringArrayExtra`
- `getStringArrayListExtra`
- ...

Otro detalle: Valores primitivos por defecto

Cuando usamos algún `getXXExtra`, por ejemplo, `getIntExtra`, ponemos un *segundo* parámetro que representa el *valor por defecto*. En este ejemplo es `-1`:

```
// Si no se ha hecho `putExtra` de "NUMERO" desde la
// actividad origen, entonces aquí, en la actividad
// destino, el valor por defecto es -1
int elNumeroQueHanPasado = getIntExtra("NUMERO", -1);
```

Otro detalle más: Claves

Para identificar los valores que están siendo *transmitidos* de una actividad a otra se usan *claves* de tipo `String`, como `"NUMERO"` en el ejemplo anterior.

Podemos elegir cualquier `String` arbitrariamente. Siempre que *coincidan* en `.putExtra` y `.getXXExtra`.

Si no coinciden... ¡La liamos! Por ello, es una buena práctica definir estas claves en un sólo sitio. Por ejemplo, como variables estáticas en la actividad². Más adelante lo pondremos en práctica.

La tarea

Crea una nueva actividad con estas características:

- El archivo de interfaz se llamará `activity_video.xml` y sólo contendrá un `<ConstraintLayout/>` por defecto
- La clase Java será `VideoActivity.java`. Recuerda que debe heredar de `AppCompatActivity`
- ¡No olvides declarar la actividad en el *manifiesto*!

Luego, al final del `onClick` de `ClipViewHolder.java` **añade** estas líneas para empezar la nueva actividad:

```

@Override
public void onClick(View view) {
    int clipId = clip.getId();
    Context context = view.getContext();

    // Mostramos un Toast para completar la tarea
    // ...

+    // Y ahora... ¡Iniciamos la VideoActivity!
+    Intent intent = new Intent(context, VideoActivity.class);
+    context.startActivity(intent);
}

```

A continuación, **añade** estas dos líneas entre medias de las dos anteriores. Sirven para indicar que queremos *transmitir* dos datos a la nueva actividad. **Sustituye** coherentemente el comentario de la URL del vídeo por un valor correcto:

```

// Y ahora... ¡Iniciamos la VideoActivity!
Intent intent = new Intent(context, VideoActivity.class);
intent.putExtra("CLIP_ID", clipId);
intent.putExtra("CLIP_URL", /* Aquí, la URL del vídeo. ¿De qué atributo de la clase la podemos sacar? */);
context.startActivity(intent);

```

Luego, ve a la `VideoActivity.java` que creaste en un principio.

Añade a `onCreate` las líneas que se muestran a continuación:

```

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

    Intent intent = getIntent(); // Con esto accedemos al Intent que se usó para iniciar la actividad
    int clipId = intent.getIntExtra("CLIP_ID", -1);
    String clipUrl = intent.getStringExtra("CLIP_URL");
    Toast.makeText(this, "videoID: " + clipId, Toast.LENGTH_LONG).show();
}

```

MainActivity
.putExtra 'CLIP_ID'
.putExtra 'CLIP_URL' → inicia y pasa datos a → VideoActivity
.getIntExtra 'CLIP_ID'
.getStringExtra 'CLIP_URL'

(Recuerda que, originalmente, hemos obtenido estos datos del API REST).

¡Enhorabuena! Has conseguido pasar a la `VideoActivity` dos datos que provienen de la celda `ClipViewHolder`, y, además, puedes verificar que el `id` se muestra en un `Toast`.

Para terminar, vamos a refactorizar el código para aplicar una buena práctica aconsejada al principio.

Añade estos atributos estáticos a `VideoActivity.java`:

```

public class VideoActivity extends AppCompatActivity {
    public static final String INTENT_CLIP_ID = "CLIP_ID";
    public static final String INTENT_CLIP_URL = "CLIP_URL";
    // ...
}

```

Luego, **sustitúyelos** en `onCreate` de `VideoActivity.java`:

```

-    int clipId = intent.getIntExtra("CLIP_ID", -1);
-    String clipUrl = intent.getStringExtra("CLIP_URL");
+    int clipId = intent.getIntExtra(VideoActivity.INTENT_CLIP_ID, -1);
+    String clipUrl = intent.getStringExtra(VideoActivity.INTENT_CLIP_URL);

```

...y **sustitúyelos** también en `ClipViewHolder.java`:

```

-    intent.putExtra("CLIP_ID", clipId);
-    intent.putExtra("CLIP_URL", /* ... */);
+    intent.putExtra(VideoActivity.INTENT_CLIP_ID, clipId);
+    intent.putExtra(VideoActivity.INTENT_CLIP_URL, /* ... */);



```

Así, las claves de los valores transmitidos están declaradas en un sólo sitio, y es más difícil que haya *bugs* por errores al teclear.

¡Tarea terminada!

Por último

Sube tus cambios al repositorio en un nuevo *commit*.

-
1. Si queremos *transmitir* una instancia de una *clase* nuestra, deberá implementar la interfaz [Serializable](#) . 
 2. El modificador `final` en Java establece que una variable no puede *reescribirse*. Es por tanto, constante 
-



[Rubén Montero @ruben.montero](#) changed milestone to [%Sprint 2](#) 3 days ago