



Programación multimedia y dispositivos móviles

UT-5.1 - Fragments

<https://developer.android.com/guide/>
<https://www.sgoliver.net/blog/curso-de-programacion-android/indice-de-contenidos/>



¿Qué es un fragmento?

La necesidad de usar fragmentos nace con la versión 3.0 (API 11) de Android debido a los múltiples tamaños de pantalla que estaban apareciendo en el mercado y a la capacidad de orientación de la interfaz (Landscape y Portrait). Estas características necesitaban dotar a las aplicaciones Android de la capacidad para adaptarse y responder a la interfaz de usuario sin importar el dispositivo.

Un fragmento es una sección «modular» de interfaz de usuario embebida dentro de una actividad anfitriona, el cual permite versatilidad y optimización de diseño. Se trata de miniactividades contenidas dentro de una actividad anfitriona, manejando su propio diseño (un recurso layout propio) y ciclo de vida.

Estas nuevas entidades permiten reutilizar código y ahorrar tiempo de diseño a la hora de desarrollar una aplicación. Los fragmentos facilitan el despliegue de tus aplicaciones en cualquier tipo de tamaño de pantalla y orientación.

¿Qué es un fragmento?

Otra ventaja de usarlos es que permiten crear diseños de interfaces de usuario de múltiples vistas. ¿Qué quiere decir eso?, que los fragmentos son imprescindibles para generar actividades con diseños dinámicos, como por ejemplo el uso de pestañas de navegación, expand and collapse, stacking, etc.

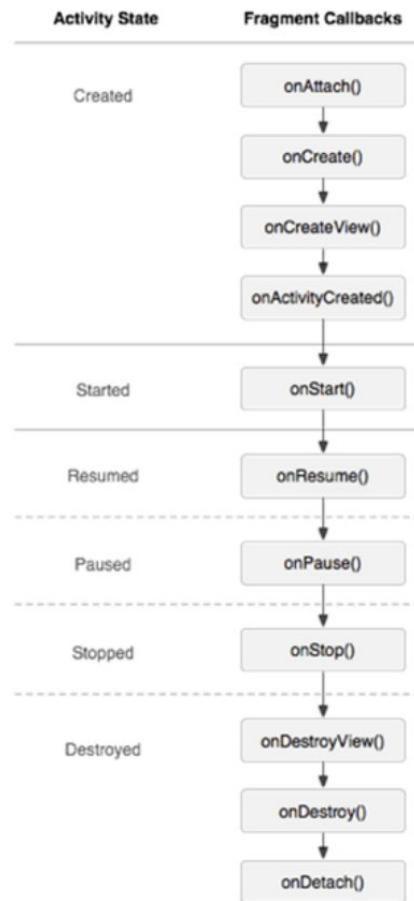


Ciclo de vida de un Fragment

Los fragmentos tienen un ciclo de vida que los asemejan a las actividades. La diferencia está en que el ciclo de vida de un fragmento depende del ciclo de vida de una actividad y además un fragmento posee algunas características extras, ya que es muy común alterarlos en tiempo de ejecución.

Al igual que con las Activities, sobreescribimos los métodos de la clase Fragment para ejecutar nuestro código en determinados puntos del ciclo de vida.

Los puntos más importantes a tener en cuenta dentro del ciclo de vida de un Fragment:





Ciclo de vida de un Fragment

- ❏ **onCreateView**: Llamado al momento de dibujar la interfaz gráfica del Fragment por primera vez. Siempre debemos implementarlo. Debemos devolver como resultado un View que representa la interfaz gráfica ya armada.
- ❏ **onPause**: Llamado cuando un Fragment está siendo abandonado por el usuario. Aquí debemos almacenar el estado de cualquier información que necesitemos conservar, porque posiblemente el Fragment no se vuelva a mostrar (y sea descartado por el sistema).



Creando un fragmento - Paso 1: Creación de fragmento

Los fragmentos son representados en el API de Android por la clase `Fragment`, por lo que cada vez que vayamos a crear un fragmento personalizado debemos crear una nueva clase que herede sus propiedades y comportamientos. Cabe aclarar que también podemos usar subclases de fragmentos creadas previamente en el API de Android como por ejemplo el `ListFragment`, `DialogFragment`, `PreferenceFragment`, etc.

Creamos un proyecto nuevo.

Con el botón derecho elegimos crear un archivo nuevo. Ahora elegimos «Fragment» y luego la opción «Fragment(Blank)», es decir, un fragmento vacío o en blanco. Le llamo “FirstFragment”.

Cada fragmento será como una `miniActividad` y tendrá su XML para pintar los controles gráficos y su clase donde pondremos el código del fragmento.



Creando un fragmento - Paso 1: Creación de fragmento

Cuando creemos un Fragment usando `Fragment(Blank)` tendremos código autogenerado en el archivo kotlin del fragment. Este código es parte de un método para crear instancias de nuestro Fragmento usando el patrón Factory. Usamos este método si necesitamos enviar algún parámetro a nuestro fragment desde el activity.

No es difícil de entender y de hecho si no enviaremos ningún dato del activity al fragment, como es nuestro caso, podríamos limpiar el código (aunque no es estrictamente necesario) y lo dejaremos como sigue:

OJO, NO LO VAMOS A HACER AHORA



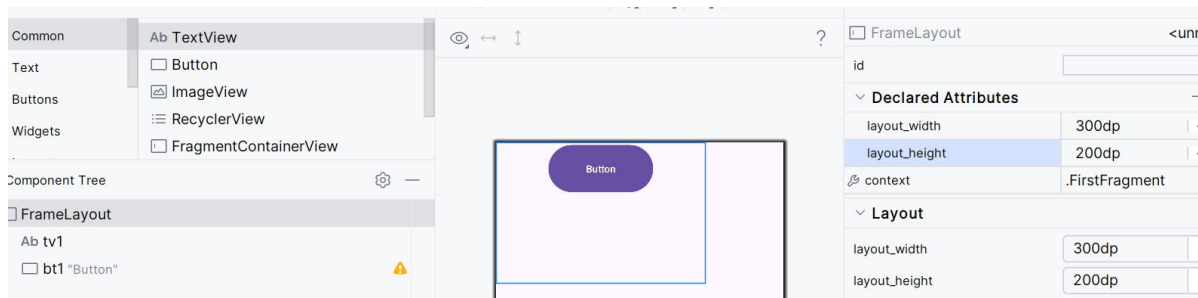
Creando un fragmento - Paso 1: Creación de fragmento

```
class FirstFragment : Fragment() {  
  
    override fun onCreateView(  
        inflater: LayoutInflater, container: ViewGroup?,  
        savedInstanceState: Bundle?  
    ): View? {  
        // Inflate the layout for this fragment  
        return inflater.inflate(R.layout.fragment_first, container, attachToRoot: false)  
    }  
}
```


Creando un fragmento -Paso 1: Creación de fragmento

El siguiente paso es “dibujar” el fragment. Nos vamos al layout y haremos lo siguiente:

- ❑ Al FrameLayout le daré medidas de 300 (ancho) x 200 (alto)
- ❑ Al cuadro de texto le llamaré tv1, le quitaré el texto que viene, le doy medidas de match_parent (ancho) x 100 (alto) y lo coloco abajo (Layout_gravity = bottom)
- ❑ Meto un boton (bt1) que será de 150x75 y con gravity centrado horizontal





Creando un fragmento - Paso 1: Creación de fragmento

Nos vamos a la clase del fragmento y codificamos que cuando se pulse el botón, en el cuadro de texto escribiremos “Mi primer fragmento”.

Esto se hace de forma similar a como lo haríamos en una actividad con la particularidad de que el método `findViewById()` no funciona en el contexto del fragmento. Esto se debe a que el único que puede referenciar a todos los Views de un fragmento es el View padre del fragmento. Dicho view es obtenido cuando se infla el layout.

Para tener una referencia a este simplemente crearemos una instancia de tipo View y asignaremos el resultado al inflar el código XML. Luego invocamos el método `findViewById()` desde la nueva instancia.

Además yo no he conseguido que funcione construyendo un método al que llamar desde el `onClick` del botón del fragmento. El código me ha quedado de la siguiente manera:

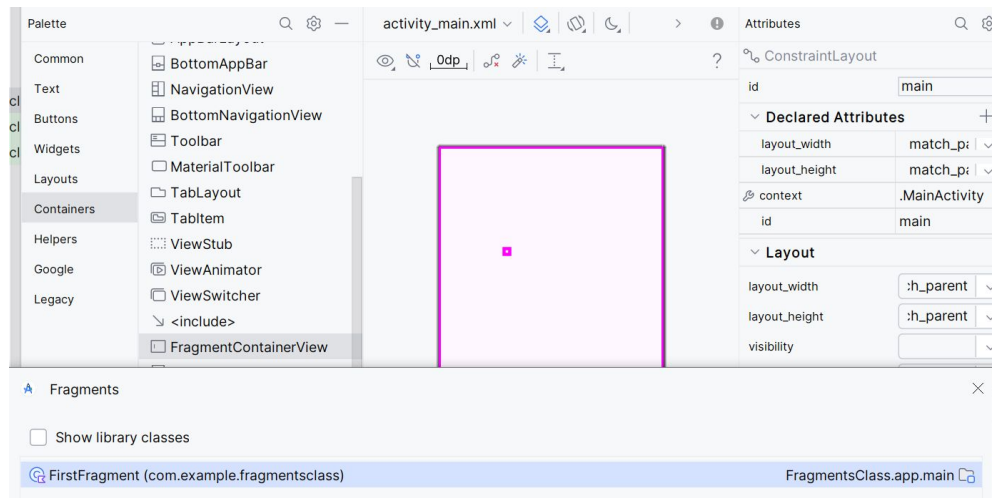


Creando un fragmento - Paso 1: Creación de fragmento

```
override fun onCreateView(  
    inflater: LayoutInflater, container: ViewGroup?,  
    savedInstanceState: Bundle?  
): View? {  
    // Inflate the layout for this fragment  
    val myView = inflater.inflate(R.layout.fragment_first, container, attachToRoot: false)  
    val myBoton = myView.findViewById<Button>(R.id.bt1)  
    val myTextto = myView.findViewById<TextView>(R.id.tv1)  
  
    myBoton.setOnClickListener{  
        myTextto.text = "Mi primer fragmento"  
    }  
  
    return myView  
}
```

Creando un fragmento - Paso 3: Insertar el fragmento en el layout de la actividad

Por último nos vamos a nuestra actividad y añadimos el fragmento (FragmentManager), y al ejecutar ya estaría





Creando un fragmento dinámico

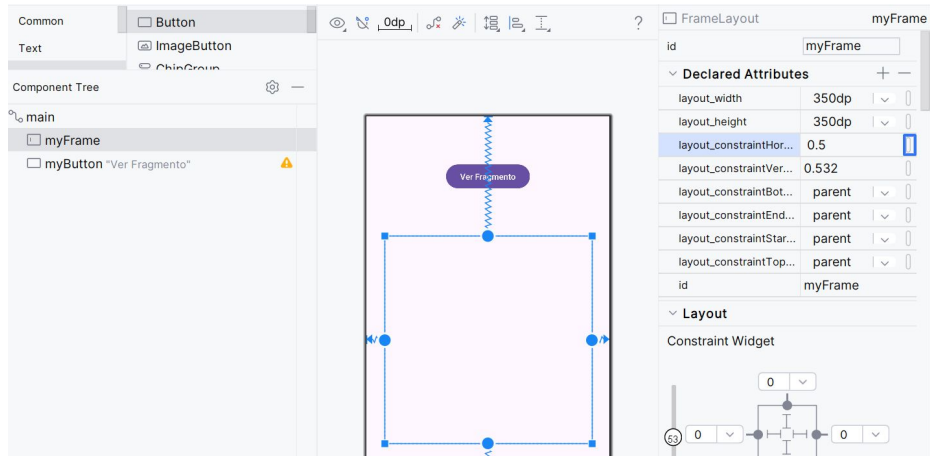
Existen casos de fragmentos que no pueden agregarse al layout de una actividad hasta que la aplicación este corriendo, bien sea por que necesitan la información de la base de datos para sus views o porque todavía el usuario no ha solicitado ver esa parte de la interfaz. En estos casos no es posible agregar los fragmentos como nodos `<fragment>` en el layout de la actividad.

Para crear un fragmento dinámicamente en tiempo de ejecución, haremos casi exactamente lo mismo que hicimos anteriormente: declaramos la clase del fragmento y generamos su layout.

Pero en vez de agregarlo al layout de la actividad, instanciaremos un elemento del tipo fragmento directamente en el código Java de nuestra actividad. Lo que quiere decir que usaremos el diseño programático en vez del declarativo.

Creando un fragmento dinámico

Sobre el proyecto en que estamos trabajando, vamos a la actividad principal y borramos el `FragmentManagerView` que hemos añadido antes. Añadimos un `FrameLayout` en el lugar dónde queremos ver el fragment



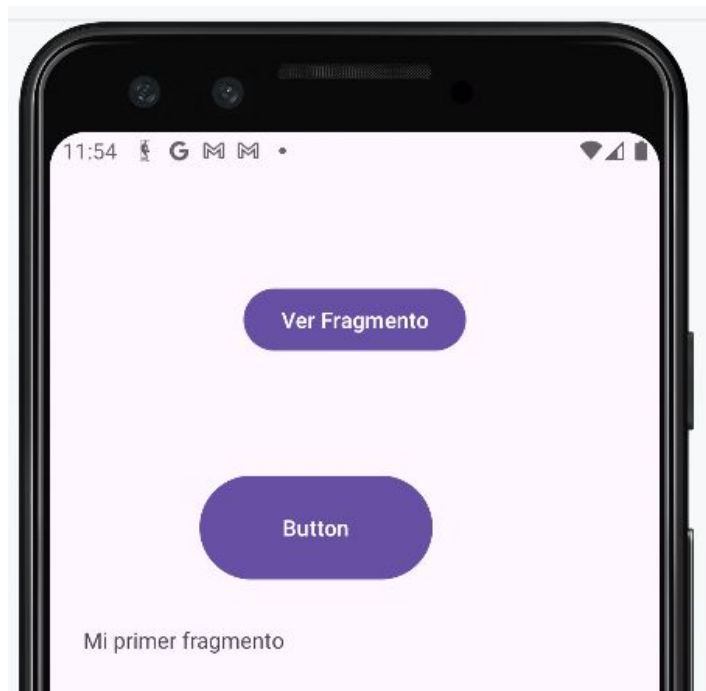


Creando un fragmento dinámico

Codificamos en el botón que nos muestre el fragmento. Son una serie de pasos siempre iguales:

```
myButon.setOnClickListener{  
    //Paso 1: Obtener la instancia del administrador de fragmentos  
    var my_fragmentManager: FragmentManager = getSupportFragmentManager();  
    //Paso 2: Crear una nueva transacción  
    var my_transaction: FragmentTransaction = my_fragmentManager.beginTransaction();  
    //Paso 3: Crear un nuevo fragmento y añadirlo  
    var mifragment : FirstFragment = FirstFragment()  
    my_transaction.add(R.id.myFrame, mifragment);  
    ///Paso 4: Confirmar el cambio  
    my_transaction.commit();  
}
```

Creando un fragmento dinámico





Llamando a un fragmento con parámetros

Hasta ahora hemos visto cómo trabajar con fragmentos que son “auto contenidos”. Pero podemos tener la necesidad de que un fragmento reciba un parámetro a la hora de crearlo.

Como hemos visto, a la hora de crear los fragmentos Android nos deja preparada la estructura necesaria para que este fragmento reciba parámetros a la hora de crearse.

En el mismo proyecto que tenemos, vamos a añadir un edittext en la actividad principal. Lo que pongamos aquí, lo pasaremos al fragmento y será lo que se muestre al pulsar el botón



Llamando a un fragmento con parámetros

Si nos fijamos, en el método onCreate del fragmento es dónde se recogen los parámetros recibidos y se asignan a variables de la clase.

En caso de tener más variables (o ser de de distinto tipo) que las 2 que genera por defecto Android, además de modificar el método newInstance, habría que venir a este método a asignar a cada variable el valor correcto (previamente debemos definir las variables que queramos).

En mi caso por “claridad”, y para entender lo que estamos haciendo, he asignado el parámetro recibido a una variable que indica lo que es y he comentado todo lo referente al 2º parámetro:



Llamando a un fragmento con parámetros

```
// TODO: Rename and change types of parameters
private var mensajeActividad: String? = null
//private var param2: String? = null

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    arguments?.let {
        mensajeActividad = it.getString(ARG_PARAM1)
        //param2 = it.getString(ARG_PARAM2)
    }
}
```



Llamando a un fragmento con parámetros

```
// TODO: Rename and change types and number of parameters  
@JvmStatic  
fun newInstance(param1: String) =  
    FirstFragment().apply {  
        arguments = Bundle().apply {  
            putString(ARG_PARAM1, param1)  
            //putString(ARG_PARAM2, param2)  
        }  
    }  
}
```



Llamando a un fragmento con parámetros

En el método onCreateView al pulsar el botón, asignamos al TextView el nombre que hemos recibido por parámetro:

```
myBoton.setOnClickListener{  
    myTexto.text = mensajeActividad  
}
```

Llamando a un fragmento con parámetros


Para acabar, al asignar el fragmento debemos primero recuperar lo que esté escrito en el editText y se lo pasaremos al fragmento (mifragment) utilizando el constructor newInstance que hemos visto antes

```
myButon.setOnClickListener{  
    //Paso 1: Obtener la instancia del administrador de fragmentos  
    var my_fragmentManager: FragmentManager = getSupportFragmentManager();  
    //Paso 2: Crear una nueva transacción  
    var my_transaction: FragmentTransaction = my_fragmentManager.beginTransaction();  
    //Paso 3: Crear un nuevo fragmento y añadirlo  
    var mifragment : FirstFragment = FirstFragment.newInstance(myEdt.text.toString())  
    my_transaction.add(R.id.myFrame, mifragment);  
    ///Paso 4: Confirmar el cambio  
    my_transaction.commit();  
}
```

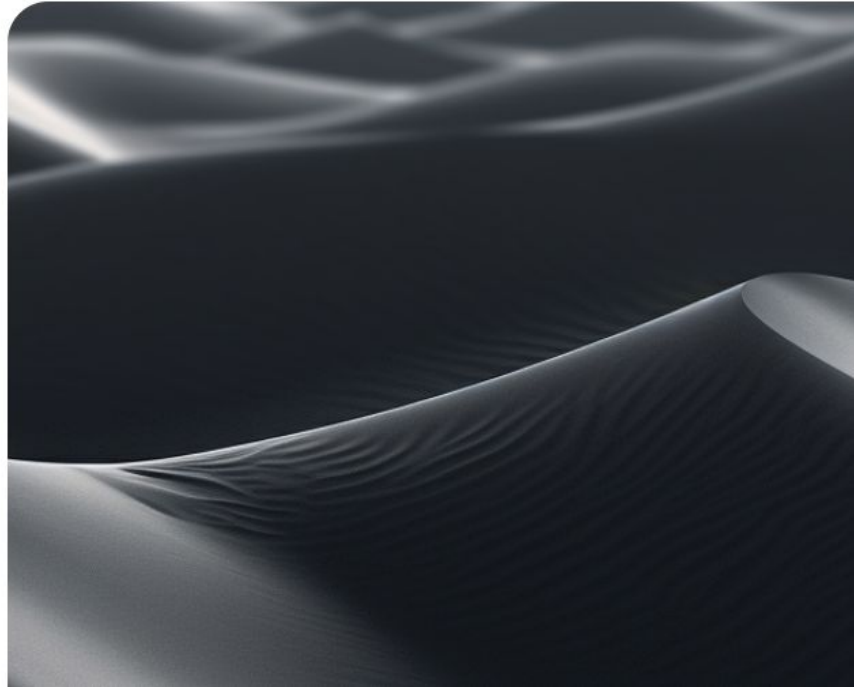


Consideraciones al trabajar con fragmentos

- **ACTIVITY:** Al trabajar con fragmentos no tienes la actividad de forma tan directa como cuando estás en una actividad (this). Para recuperar la actividad a la que está asociada un fragmento (por ejemplo para mostrar un Toast o para poner un Observador) tenemos el método getActivity() o incluso [mejor requireActivity\(\)](#)
- **CONTEXT:** Tampoco tenemos el contexto (que también es this en una actividad o applicationContext), pero el contexto siempre lo podemos sacar de una vista (myView.context)
- **VIEWMODEL:** En mis pruebas no he podido instanciar clases ViewModel desde el fragmento directamente... Para solucionarlo, lo que he hecho es hacerlo desde la actividad y hacerle llegar la clase como argumento. Sospecho que tiene que ver con la relación del ViewModel y la actividad en cuanto a ciclo de vida.
- **BINDING:** Según la [documentación de Android](#) (y [diversas páginas](#) que he consultado) la vinculación de vistas debería funcionar y de forma sencilla con fragmentos... pues a mi no me ha funcionado. Ni idea de porqué.



¿Single activity
o varias
activities en
Android? Pros y
contras







Documentación

<https://www.develou.com/android-aplicaciones-fragmento/>

<https://codigoonclick.com/manejo-de-fragmentos-en-android/>

<https://andygeek.com/posts/Fundamentos%20de%20Android/posts/Comunicando-fragments-con-java/>

<https://academiaandroid.com/activity-y-fragments/>

<https://www.develou.com/tutorial-layouts-en-android/>

<https://www.flipandroid.com/uso-del-contexto-en-un-fragmento.html>

<https://gastack.mx/programming/17436298/how-to-pass-a-variable-from-activity-to-fragment-and-pass-it-back>

<https://gpmess.com/blog/2014/04/16/buenas-practicas-usando-fragments-en-android/>