FRAGMENTS

Programación multimedia
y dispositivos móviles
2º DAM

ÍNDICE DE CONTENIDOS

1. INTRODUCCIÓN

- Un Fragment representa una parte de la interfaz de usuario en una Activity.
- Los fragmentos aparecen en Android a partir del API 11, permitiendo interfaces de usuario más flexibles al admitir mayor variedad de tamaños de pantallas sin tener que diseñar vistas distintas para actividades similares.
- Puedes combinar múltiples fragmentos en una sola actividad para crear una IU multipanel y volver a usar un fragmento en múltiples actividades.
- Podemos ver un fragmento como una sección modular de una actividad que tiene su ciclo de vida propio, recibe sus propios eventos de entrada y que puedes agregar o quitar mientras la actividad se esté ejecutando.

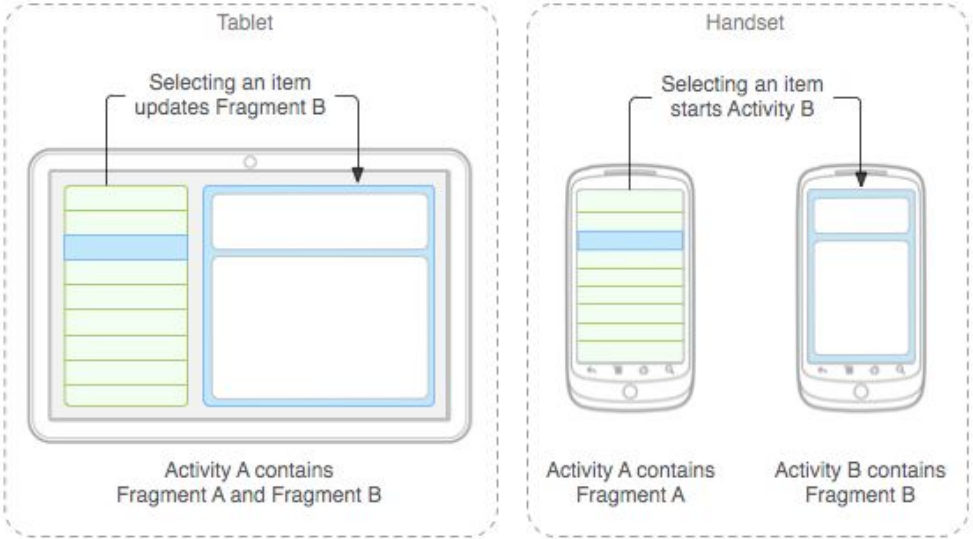
1. INTRODUCCIÓN

- Los fragmentos son secciones o módulos que se incluyen en una actividad principal, donde deben ser integrados, y aunque tienen su propio ciclo de vida, están ligados al ciclo de vida de la actividad que lo contiene.
- Cada fragmento define su propio diseño de vista de forma similar a lo que sucede con una Activity. Por tanto, un fragment estará compuesto:
    - Su diseño dado mediante un fichero xml.
    - Su controlador definido por una clase Java.

1. INTRODUCCIÓN

- El carácter modular de los fragmentos permite usarlos en una o varias actividades y combinarlos según la apariencia que se desea que tenga la aplicación. Igualmente, los fragmentos disponen de sus propios eventos, que pueden ser controlados y programados.
- Debemos diseñar cada fragment como un componente totalmente independiente, es decir, su comportamiento no debe depender de otros fragments o componentes de la aplicación

1. INTRODUCCIÓN

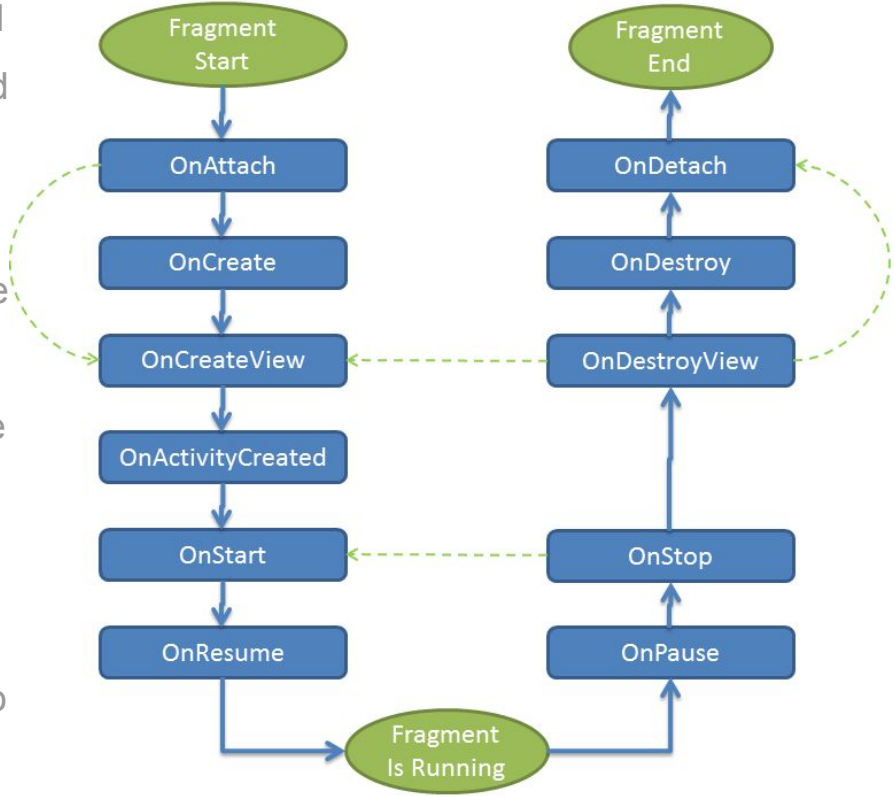


2. CICLO DE VIDA DE LOS FRAGMENTS

- Mientras la Activity se encuentra en ejecución, puede manipularse cada fragmento en ella contenido.
- Si la Activity entra en pausa, entrarán en pausa todos los fragmentos que contiene.
- Los fragmentos serán destruidos cuando la Activity contenedora se destruya.
- Al igual que sucede con las Activitys, puede conservarse el estado del fragmento mediante un Bundle, llamando al método onSaveInstanceState().
- Mientras las Activitys se añaden automáticamente a la pila de actividades (Back Stack) que es gestionada por el sistema, los fragmentos deberán ser añadidos por petición nuestra mediante la invocación de addToBackStack().

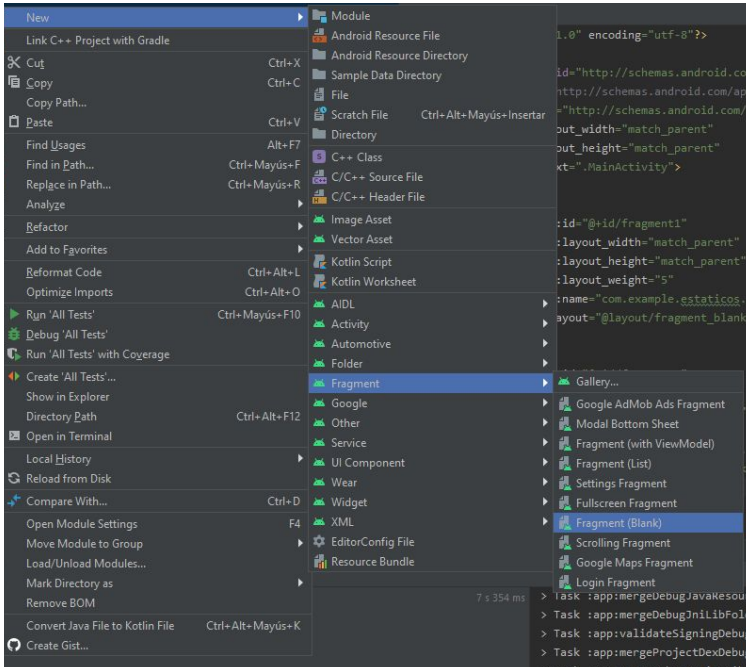
2. CICLO DE VIDA DE LOS FRAGMENTS

- **onAttach** → este método responde cuando el fragmento se asocia a una actividad contenedora.
- **onCreateView** → se llama al mostrarse el fragmento por primera vez, donde debe crearse la vista.
- **onActivityCreated** → se ejecuta al terminar de crearse la actividad principal.
- **onDestroyView** → llamado al destruirse la jerarquía de vistas asociadas al fragmento.
- **onDetach** → llamado al dejar de estar asociado el fragmento a la actividad contenedora.



3. CREAR UN FRAGMENT

Para crear un fragment lo haremos de la misma forma que creamos un Activity en Android Studio. Con esto crearemos los dos componentes del fragmento: su xml y su clase java.



3. CREAR UN FRAGMENT

- Por defecto, los únicos métodos que necesitaremos en un fragment para poder usarlo son:
  - Constructor
  - onCreate
  - onCreateView → En este método es donde enlazaremos la vista xml con el controlador Java

```java
public class BlankFragment extends Fragment {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_blank, container,  attachToRoot: false);
    }


}
```

3. CREAR UN FRAGMENT

Podemos insertar un fragment dentro de nuestra actividad de dos formas:

- Forma estática (Durante el diseño)
- Forma dinámica (En runtime).

4. FRAGMENT ESTÁTICO

- Para agregar un fragmento a una Activity durante el proceso de diseño usaremos la etiqueta <fragment… />
- Dentro de la etiqueta fragment, podemos destacar las siguientes propiedades:
  - name → Hace referencia a la clase Java del fragment
  - layout → Hace referencia al componente XML del fragment
  - weight → (No es exclusiva de fragment) Es el “peso” que tendrá nuestro fragmento dentro de la vista, de esta forma podemos distribuir varios fragmentos dentro de un mismo contenedor.

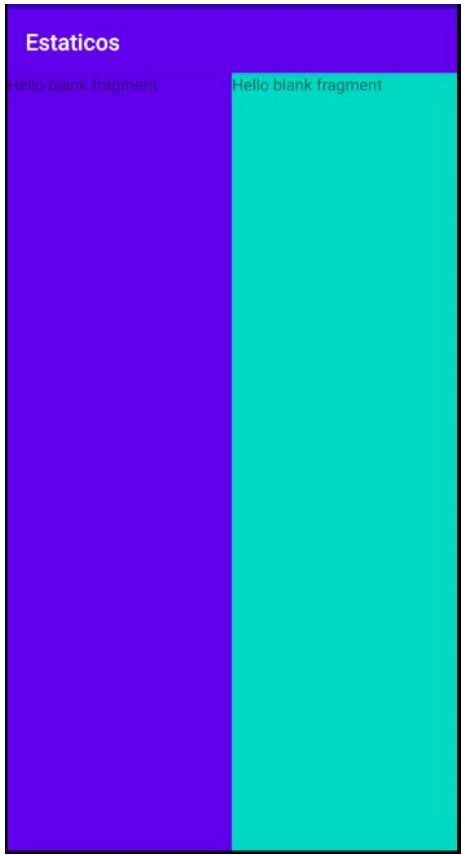
```
<fragment
    android:id="@+id/fragment1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_weight="5"
    android:name="com.example.estaticos.BlankFragment"
    tools:layout="@layout/fragment_blank"/>
```

4. FRAGMENT ESTÁTICO

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <fragment
        android:id="@+id/fragment1"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="5"
        android:name="com.example.estaticos.BlankFragment"
        tools:layout="@layout/fragment_blank"/>

    <fragment
        android:id="@+id/fragment2"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_weight="5"
        android:name="com.example.estaticos.BlankFragment2"
        tools:layout="@layout/fragment_blank2"/>
</LinearLayout>
```



4. FRAGMENT DINÁMICO

- Del mismo modo que podemos cargar nuestros fragments desde el propio diseño XML, es posible cargarlos en runtime a través del controlador de una Activity.
- Esto nos permite crear UI dinámicas cargando diferentes fragmentos según conveniencia.
- Para cargar un fragmento dinámicamente tenemos que seguir 4 sencillos pasos
    - Obtener una instancia de fragmentManager() → El servicio de la API Android encargado de gestionar las transacciones de fragmentos.
    - Crear una transacción
    - Crear una instancia del fragmento que vamos a cargar.
    - Añadir el fragmento a la transacción, indicando el contenedor en el que lo vamos a cargar
    - Ejecutar la transacción

4. FRAGMENT DINÁMICO

- Se indicará en la activity el lugar indicado para un fragment:

```
<FrameLayout
    android:id="@+id/fragment2"
    android:layout_width="516dp"
    android:layout_height="match_parent" />
```

4. FRAGMENT DINÁMICO

- Ejemplo para cargar un fragment:

```
/*Comenzamos el tratamiento del fragmento
 * Obtener una instancia de FragmentManager() -> El servicio de la API Android
 * encargado de gestionar las transacciones de fragmentos*/
FragmentManager fragmentManager = getSupportFragmentManager();
/*Crear una transaccion*/
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
/*Crear una instancia del fragmento a cargar*/
BlankFragment miFragmento = new BlankFragment();
/*Añadir el fragmento a la transacción,
 * indicando el contenedor en el que lo vamos a cargar*/
/*R.id.fragment2 es el id del elemento de la etiqueta FrameLayout*/
fragmentTransaction.replace(R.id.fragment2, miFragmento);
/*Ejecutar la transacción*/
fragmentTransaction.commit();
```

4. FRAGMENT DINÁMICO

- Para acceder a un fragmento ya añadido, se utiliza getFragmentById()

```
BlankFragment fragment = (BlankFragment) fragmentManager.findFragmentById(R.id.fragment2);
```

Si quieres que se queden en la pila los fragmentos, debes indicarlo antes de ejecutar la transacción.

```
FragmentManager fragmentManager = getSupportFragmentManager();
FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
BlankFragment2 fragment2 = new BlankFragment2();
fragmentTransaction.replace(R.id.fragment2, fragment2);
fragmentTransaction.addToBackStack(null);
fragmentTransaction.commit();
```

5. DUDAS Y PREGUNTAS