COMUNICACIÓN ENTRE FRAGMENTOS Y/O ACTIVITY

En esta actividad vamos a ver cómo pasar información entre dos fragmentos o entre un fragmento y la actividad que lo contiene.

Paso 1: Creación del proyecto y primeros pasos

Creamos un nuevo proyecto con una Empty Activity. El activity_main.xml cambiamos ConstraintLayout por un FrameLayout y recordad ponerle su id

```
android:id="@+id/contenedor"
```

A continuación creamos un nuevo fragmento, con un Fragment Blank. Este en su layout contendrá dos botones, cada uno de los cuales al ser pulsado van a generar una determinada información que será pasada al activity que contiene a este fragmento para que haga alguna acción con ellos.

De las distintas opciones que puedes marcar al crear un Fragment Blank, deja marcada "Include interface". Esta opción nos va a generar código necesario para permitir la comunicación de información de este fragmento y el activity que lo contiene.

Paso 2: Estudiando el código generado en el fragmento

```
public class ControlesFragment extends Fragment {

    private OnFragmentInteractionListener mListener;
```

Además de estos dos métodos:

```
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    if (context instanceof OnFragmentInteractionListener) {
        mListener = (OnFragmentInteractionListener) context;
    } else {
        throw new RuntimeException(context.toString()
                + " must implement OnFragmentInteractionListener");
    }
}

@Override
public void onDetach() {
    super.onDetach();
    mListener = null;
}
```

- onAttach: se lanza cuando nosotros insertamos un fragmentos dentro de un activity, es decir, cuando en el MainActivity, generamos el fragmento a cargar agregando el código correspondiente

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

    getSupportFragmentManager()
            .beginTransaction()
            .add(R.id.contenedor,new ControlesFragment())
            .commit();
}
```

Al ejecutarse el commit, el método onAttach, como vemos recibe como parámetro una referencia al Activity en el que hemos insertado el fragmento, de manera que el context es directamente el activity. Internamente el código lo que hace es comprobar que el si el activity en la que he cargado el fragmento implementa el elemento OnFragmentInteractionListener, que si bajamos en el código vemos que

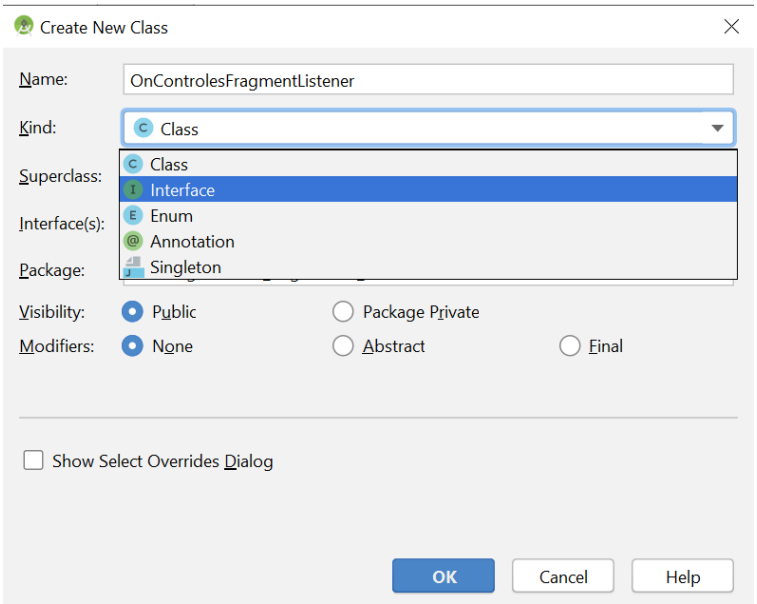
es una interface, me asegura que estará implementando el método que se encuentra definido en dicha interface

- Como esta interface es autogenerada vamos a borrarla para cambiarla por una propia que haga que nuestro programa realice lo que nosotros deseamos.

Paso 3: generar nuestra propia interface

Borramos en el ControlesFragment la interface OnFragmentInteractionListener. Además borrar también el método *void onButtonPressed(Uri uri).*

Creamos una nueva interface: sobre carpeta java: new→java class



En esta interface vamos a definir la implementación de dos métodos:

```
public interface OnControlesFragmentListener {
    public void botonColorClicked(String color);
    public void botonTextoClicked(String texto);
}
```

Este OncontrolesFragmentListener es la interface que voy a usar en el fragmento. Vuelvo al código del fragmento y cambio las referencias que había a la interface inicial por esta nueva que he creado.

En el método onAttach, si el activity implementa mi interface, voy a castear context (que es una referencia al Activity), para que tenga el tipo de mi interface y lo guardo en mListener.

Mi inteface implementa define dos método que representarán la función de dos botones, por tanto vamos a definir los dos botones en el layout de fragment_controles.xml

Una vez generados los dos botones vamos a crear la funcionalidad de dichos botones una vez que se vaya a cargar dicha interface. Para ello vamos a modificar el método onCreateView del controles_fragment.java. Lo que hacemos es que antes de devolver la vista creada con el return, vamos a declarar los dos botones que debe contener y a darles una funcionalidad definiendo se escuchador de eventos para cada uno de ellos. El código quedaría como se muestra a continuación:

```java
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Inflate the layout for this fragment
    View view = inflater.inflate(R.layout.fragment_controles, container, attachToRoot: false);

    Button btnColor = (Button) view.findViewById(R.id.buttonColor);
    Button btnTexto = (Button) view.findViewById(R.id.buttonTexto);

    btnColor.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // Cuando pulse el botón color del Fragment
            mListener.botonColorClicked("Rojo");
        }
    });

    btnTexto.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            // Cuando pulse el botón texto del Fragment
            mListener.botonTextoClicked("Hola usuario");
        }
    });

    return view;
}
```

Al hacer click en alguno de los botones estamos invocando al activity mediante el mListener, y en él estamos llamando al método botonColorClicked (o botonTextoClicked), entonces lo que hacemos es llamar al activity que ha incluido a esste fragmento, a través del getSupportFragmentManager

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

    getSupportFragmentManager()
            .beginTransaction()
            .add(R.id.contenedor,new ControlesFragment())
            .commit();
}
```

Si ejecutamos este código, no debemos olvidar que hay que decirle que implemente la clase que hemos creado y por tanto que implemente los dos métodos definidos en dicha interface

```java
public class MainActivity extends AppCompatActivity implements OnControlesFragmentListener {

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


        getSupportFragmentManager() FragmentManager
                .beginTransaction() FragmentTransaction
                .add(R.id.contenedor,new ControlesFragment()) FragmentTransaction
                .commit();
    }

    @Override
    public void botonColorClicked(String color) {
        Toast.makeText( context: this,  text: "Estoy en el activity, color recibido: "+color, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void botonTextoClicked(String texto) {
        Toast.makeText( context: this,  text: "Estoy en el activity, texto recibido: "+texto, Toast.LENGTH_SHORT).show();
    }
```