FRAGMENTS

Programación multimedia
y dispositivos móviles
2º DAM

ÍNDICE DE CONTENIDOS

1. COMUNICACIÓN ENTRE FRAGMENTS

- Existen múltiples métodos para comunicarse con un fragment
  - Parámetros de inicialización
  - Callbacks a la actividad padre
  - Variables o métodos públicos → DESACONSEJADO

2. PARÁMETROS DE INICIALIZACIÓN

Se pueden establecer una serie de parámetros que enviar al fragment para que pueda realizar su funcionalidad.

Existen 2 modos de hacerlo:

- Constructores
- Argumentos

2.1. CONSTRUCTOR

La primera opción, es enviar datos a través del constructor del Fragmento.

```
public class TextFragment extends Fragment {

    private String text;

    public TextFragment() {
        this.text = "";
    }

    public TextFragment (String text) {
        this.text = text;
    }
```

El constructor, como cualquier otra clase de Java, puede recibir parámetros para inicializar los atributos del fragmento.

2.1. CONSTRUCTOR

De esta forma, al crear el Fragment se le envían los argumentos necesarios.

```
fragment = new TextFragment(item.getText());
```

Y, por lo tanto, podemos utilizar el valor en cualquier método del Fragment.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Carga el layout del fragmento
    View view = inflater.inflate(R.layout.fragment_text, container,  attachToRoot: false);

    // Carga el texto en el TextView
    TextView textView = view.findViewById(R.id.textView);
    textView.setText(this.text);

    return view;
}
```

2.2. ARGUMENTOS

La segunda forma, y más apropiada, es enviar los datos mediante argumentos en un objeto Bundle.

```
fragment = new ImageFragment();
Bundle args = new Bundle();
args.putInt(getResources().getString(R.string.image), item.getImageId());
fragment.setArguments(args);
```

La clase Fragment tiene definido un atributo de tipo Bundle (Básicamente un Array de pares clave-valor) que se emplea para almacenar argumentos que puedan ser utilizados por el fragment durante su construcción (método onCreate o onCreateView). Este atributo se llama Arguments.

2.2. ARGUMENTOS

En el onCreate o en el onCreateView, leemos el contenido de los argumentos recibidos mediante el método getArguments(), indicando el tipo de dato que vamos a leer (getInt(), getString()...) y el nombre de la clave que se ha establecido a la hora de enviar los datos.

```
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    // Carga el layout del fragmento
    View view = inflater.inflate(R.layout.fragment_image, container,  attachToRoot: false);

    // Si ha recibido argumentos
    if (getArguments() != null) {
        // Obtiene el id de la imagen
        int imageId = getArguments().getInt(getResources().getString(R.string.image));
        // Obtiene la vista de imagen
        ImageView imageView = view.findViewById(R.id.imageView);
        // Carga la imagen en la vista
        imageView.setImageDrawable(getContext().getDrawable(imageId));
    }

    return view;
}
```

3. CALLBACKS A LA ACTIVIDAD PADRE

Para que el fragment se pueda comunicar con su actividad padre, deberá definir un interfaz (Listener) para que la actividad padre lo implemente.

Crearemos la interfaz en nuestro proyecto, con el método al que se invocará:

```
public interface OnFragmentEventListener {
    void onFragmentEvent(Item item);
}
```

El fragment que lanzará el evento (ListFragment) tendrá un atributo de esta interfaz:

```
public class ListFragment extends Fragment implements AdapterView.OnItemClickListener {

    private OnFragmentEventListener listener;
```

3. CALLBACKS A LA ACTIVIDAD PADRE

En los métodos onAttach y onDetach, se iniciará y liberará respectivamente esta interfaz:

```
@Override
public void onAttach(Context context) {
    super.onAttach(context);
    // Guarda la referencia al listener de la actividad
    if (context instanceof OnFragmentEventListener) {
        listener = (OnFragmentEventListener) context;
    }
}

@Override
public void onDetach() {
    super.onDetach();
    // Libera la referencia al listener
    listener = null;
}
```

3. CALLBACKS A LA ACTIVIDAD PADRE

A la hora de realizar un evento, y que se necesite que la Actividad sea quién realice la acción, se llamará al método de la interfaz sobre el objeto que hemos creado (puede llevar parámetros si se necesita).

```
@Override
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
    // Evento de selección del ListView

    // Si tiene listener de actividad
    if (listener != null) {
        // Obtiene el Item
        Item item = adapter.getItem(position);
        // Se lo pasa a la actividad
        listener.onFragmentEvent(item);
    }
}
```

3. CALLBACKS A LA ACTIVIDAD PADRE

Por lo tanto, se ejecutará el código del evento en la actividad (MainActivity)

```
public class MainActivity extends AppCompatActivity implements OnFragmentEventListener {
```

```
@Override
public void onFragmentEvent(Item item) {
    // Comprueba la orientación del dispositivo
    int orientation = getResources().getConfiguration().orientation;
    if (orientation == Configuration.ORIENTATION_LANDSCAPE) {
        // Orientación LANDSCAPE, reemplaza el fragmento de detalle

        // Construye el fragmento dependiendo del tipo de Item (imagen o texto)
        Fragment fragment;
```

4. DUDAS Y PREGUNTAS