CENTRE ESPECÍFIC
D´EDUCACIÓ A DISTÀNCIA DE
LA COMUNITAT VALENCIANA

UD 04

PROGRAMACIÓN MULTIMEDIA Y DISPOSITIVOS MÓVILES 22/23
CFGS DAM

FRAGMENTS Y PREFERENCIAS

DISEÑO E INTRODUCCIÓN A LA ESTRUCTURA *FRAGMENT*

Autor: Carlos Espinosa
c.espinosamoreno@edu.gva.es
Fecha: 2022/2023

Índice

1. Objetivos de la unidad

- Conocer los fragments y sus características básicas
- Aprender el uso de fragments en una aplicación
- Conocer las preferencias, su uso y las alternativas disponibles.
- Utilizar Toast para la personalización de mensajes

2. Introducción a los fragments

Los fragments son una sección de una interfaz de usuario que está incluida en una activity, pero que funciona con cierta independencia. Son muy utilizados en un nivel más avanzado y uno de los principales usos es el de adaptar una aplicación a diferentes tamaños de pantalla. Como sabes, no es lo mismo programar para un reloj de 2 pulgadas que para una tablet de 9.

Una de las principales características de los fragments, como hemos comentado es LA independencia: En caso de que la actividad esté iniciada (en ejecución), el fragment podrá estar iniciado o bien estar en pausa. Sin embargo, si la actividad pasara a onDestroy o fuera de foco, el fragment también asumiría ese estado. La principal es la reutilización, ya que podemos usar el fragment en diferentes actívities. La otra es su independencia, que debemos recordar no es ilimitada. Y la tercera es el dinamismo, ya que, por ejemplo, a la hora de crear menús avanzados que puedan desplegarse, podría ser conveniente el uso de fragments.

Los fragments pueden ser estáticos o dinámicos. En el caso de los estáticos no muestran movimiento, mientras que los dinámicos si muestran movimiento.

Android introduce los fragmentos en Android 3.0 (nivel de API 11), principalmente para admitir diseños de IU más dinámicos y flexibles en pantallas grandes, como las de las tablets. Como la pantalla de una tablet es mucho más grande que la de un teléfono, hay más espacio para combinar e intercambiar componentes de la IU. Los fragmentos admiten esos diseños sin la necesidad de que administres cambios complejos en la jerarquía de vistas. Al dividir el diseño de una actividad en fragmentos, puedes modificar el aspecto de la actividad durante el tiempo de ejecución y conservar esos cambios en una pila de actividades administrada por la actividad.

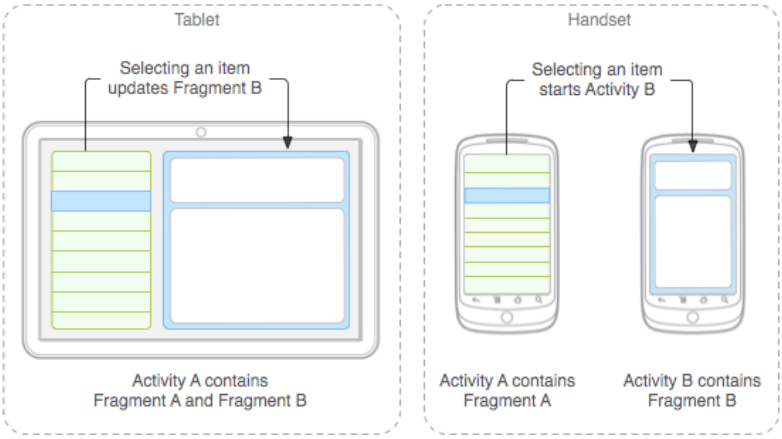


Imagen: ejemplo de la manera en que dos módulos de la IU definidos por fragmentos se pueden combinar en una actividad para un diseño de tablet y se presentan por separado para un diseño de teléfono.

3. Principales características de los fragments

Las principales características de un fragment son:

- Tiene su propio ciclo de vida (aunque con limitaciones)
- Tiene eventos propios de inicio y cierre
- Puede eliminarse de una actividad en cualquier momento
- Puede agregarse a una actividad en cualquier momento
- Siempre debe estar en una actividad (no puede ser independiente)
- Un mismo fragment puede usarse en varias Actividades
- Dispone de entradas y salidas de datos propias
- Se compone de dos ficheros: ○ XML ○ KOTLIN A la hora de implementar, cada fragment ha de implementarse en una clase diferente con una estructura similar a la de las Activity, pero con diferencias.
- El fragment tiene que extenderse
- El ciclo de vida de un fragment es distinto: ○ Va ligado al de la Activity que lo contiene (y por tanto si esta pasa a ser destruida, la misma suerte correrán los fragments que contenga) ○ Esta propiedad no es recíproca, es decir, se puede destruir un fragment sin destruir la Activity que lo contiene. ○ Incluye nuevos estados, estos estados son:

En la etapa CREATED:

- onAttach() es cuando el fragment se asocia a la activity
- onCreate() cuando se crea el fragment
- onCreateView() cuando se diseña el layout
- onActivityCreated() cuando se muestra el método onCreate() de la propia actividad.

En el estado STARTED:

- onStart() cuando se llama al fragment para que sea iniciado

En RESUMED:

- onResume() el fragment vuelve a verse en la Activity

En el estado PAUSED:

- onPause() hace que el fragment pase a un estado en el que deja de estar en activo o primer plano

En STOPPED:

- onStop() que permite que el fragment mantenga su información y su estado

En DESTROYED

- onDestroyView() es el primer paso, el que destruye el layout del fragment.
- onDestroy() que destruirá al propio fragment
- onDetach() que finaliza la relación entre el fragment y la Activity.

4. Creación y diseño de fragments en Kotlin

Ahora que hemos visto los diferentes estados en los que puede estar un fragment, es hora de saber cómo hemos de introducirlo en nuestra aplicación. Como sabes, una Activity está formada por dos ficheros, uno de código en Kotlin y otro en XML. A la hora de introducir un fragment en una Activity, puede hacerse mediante el fichero de código o mediante el de XML, y como veremos a continuación cada una de las formas tendrá sus pros y sus contras.

Una de las cosas más importantes que tenemos que tener en cuenta a la hora de diseñar un fragment es que su gestión debe ser propia y que deberá ser capaz de modificarse sin intervención de la Activity en la que se halle (esto es lo que hace que sea tan versátil), convirtiéndose así en un elemento base en el diseño de layouts. Generalmente, debes implementar al menos los métodos OnCreate, OnCreateView y onPause.

Declarar el fragmento en el archivo de diseño de la actividad.

En este caso, puedes especificar propiedades de diseño para el fragmento como si fueran una vista. Por ejemplo, aquí te mostramos un archivo de diseño para una actividad con dos fragmentos:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="horizontal"
android:layout_width="match_parent"
android:layout_height="match_parent">
<fragment android:name="com.example.news.ArticleListFragment"
    android:id="@+id/list"
    android:layout_weight="1"
    android:layout_width="0dp"
    android:layout_height="match_parent" />
<fragment android:name="com.example.news.ArticleReaderFragment"
    android:id="@+id/viewer"
    android:layout_weight="2"
    android:layout_width="0dp"
    android:layout_height="match_parent" />
</LinearLayout>
```

El atributo android:name de <fragment> especifica la clase Fragment para crear una instancia en el diseño.

Cuando el sistema crea el diseño de esta actividad, crea una instancia para cada fragmento especificado en el diseño, además de llamar al método onCreateView(), con el objetivo de recuperar el diseño de cada fragmento. El sistema inserta el objeto View que muestra el fragmento directamente en lugar del elemento <fragment>.

Cada fragmento requiere un **identificador único** que el sistema puede usar para restaurar el fragmento si se reinicia la actividad (y que tú puedes usar para que el fragmento realice transacciones, como quitarlo). Hay tres formas de proporcionarle un ID a un fragmento:

- Proporciona el atributo android:id con un ID único.
- Proporciona el atributo android:tag con una string única.

4.1 Preferencias

Las preferencias son la información que una aplicación guarda, haciendo que la interrelación con el usuario sea personalizada. Para llevar a cabo la personalización puede hacerse con persistencias (cosa que veremos en la unidad 6 o mediante la clase PreferenceActivity. Las preferencias se almacenan en ficheros XML, en la carpeta shared_preferences, dentro del fichero nombreAplicacion_preferences. En el caso de que uses las preferencias para almacenar otros valores puedes usar otros ficheros. Las alternativas son básicamente:

● getSharedPreferences() es imprescindible indicar el nombre del fichero de preferencias. Es útil cuando necesitas acceder al fichero desde diferentes Activity o varios ficheros de preferencias

● getPreferences() No es necesario indicar el nombre. Utilizar cuando solo necesitas un fichero de preferencias en la Activity. Además, independientemente del modo empleado, hay que incluir el tipo de permiso que queremos otorgar, estos son: ● MODE_PRIVATE para otorgar únicamente acceso ● MODE_WORLD_READABLE permite la lectura ● MODE_WORLD_WRITEABLE permite lectura y escritura. Lo que devolverá un objeto de la clase SharedPreferences. Para escribir las preferencias, siguiendo el ejemplo del triaje hospitalario:

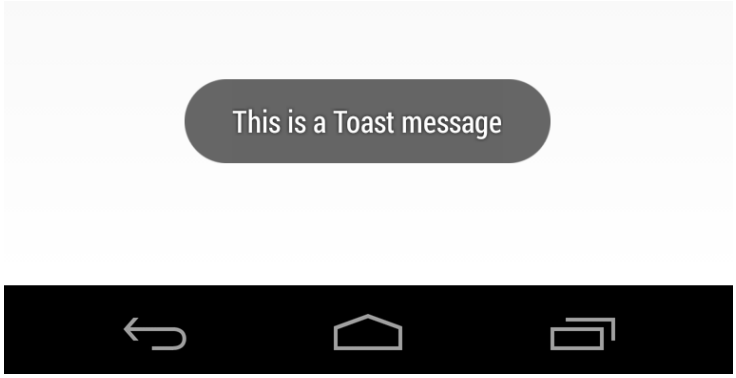
PREFERENCIAS	PREFERENCIAS COMPARTIDAS
No cambian cuando el dispositivo reinicia	Pueden cambiar con el reinicio del dispositivo
Almacenan los datos en dispositivos de respaldo	Almacenan los datos en ficheros XML
Funcionan muy bien con datos sensibles	No son la mejor opción para datos sensibles

5. Diálogos y notificaciones:

Las notificaciones son alertas visuales que aparecen en la barra de estado en Android, son por ejemplo los avisos de una llamada perdida, de un Whatsapp o una actualización de una aplicación. Dentro de las notificaciones encontramos diferentes tipos, en función de su complejidad y grado de interacción con el usuario.

Toast:

Es la notificación más sencilla, se trata de mensajes que aparecen en la pantalla durante algún tiempo (segundos) para informar de algo. No requieren interacción por parte del usuario y pueden ser posicionados en cualquier lugar (mediante setGravity).

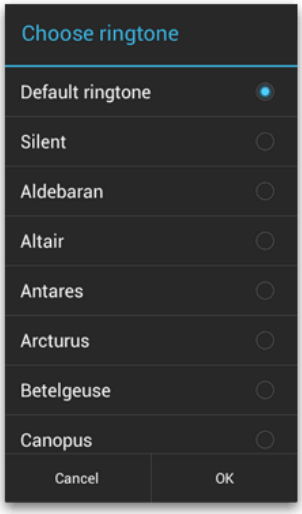


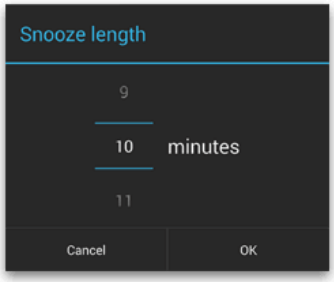
Para su definición lo único que tenemos que especificar es el tiempo que durará y el mensaje que queremos mostrar.

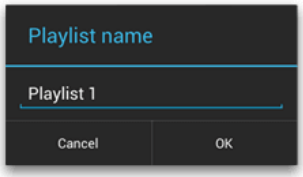
```
Toast.makeText(this, "Notificación corta", Toast.LENGTH_SHORT).show()
```

Diálogos:

Una vez conocemos la forma más sencilla que presentan las notificaciones (es decir, aquellas que no requiere interacción por parte del usuario), vamos a ver los diálogos, entendidos como objetos de Android en los que se requiere una interacción sencilla por parte del usuario como una confirmación de algo (o rechazo), una elección entre alternativas, etc.







Dialogo simple: es el más sencillo en su creación y también en la interacción con el usuario. Para su creación, necesitaremos dos clases:

- AlertDialog: que será la que muestre u oculte el diálogo
- AlertDialog.Builder: ○ para establecer el mensaje: setMessage() ○ el título: setTitle() ○ el estilo: setView()

Los botones: para iniciar la acción: setPositiveButton() , para cancelar la acción setNegativeButton() ○ una lista: setItems() ○ otras características.

Dialogo de confirmación: es el utilizado cuando nosotros mostramos el mensaje y la función del usuario es únicamente aceptar o rechazar lo que estamos planteando

Dialogo de selección: se utiliza cuando queremos darle al usuario una serie de opciones de respuesta. En este caso se hará utilizando setItems() Importante, definir primero los items:

En el caso de querer que el usuario pueda elegir más de una opción, los cambios serán sustituir: SetSingleChoiceClickListener() por setMultipleChoiceItems() y en public void onClick(DialogInterface dialog, int item) añadiremos public void onClick(DialogInterface dialog, int item, boolean isChecked) Recuerda siempre incluir el código para que se muestre

5.1 Otros tipos de fragments

DialogFragment:

Muestra un diálogo flotante. Usar esta clase para crear un diálogo es una buena alternativa al uso de métodos del asistente de diálogos en la clase Activity, ya que puedes incorporar un diálogo del fragmento en la pila de actividades de fragmentos administrados por la actividad, lo que le permite al usuario volver a un fragmento descartado.

ListFragment:

Muestra una lista de elementos administrados por un adaptador (como un SimpleCursorAdapter), al igual que ListActivity. Proporciona varios métodos para administrar una vista de lista, como la devolución de llamada onListItemClick() para manipular eventos de clic. (Ten en cuenta que el método preferido para mostrar una lista es utilizar RecyclerView en lugar de ListView. En este caso, necesitarías crear un fragmento que incluya un RecyclerView en su diseño. Consulta Cómo crear una lista con RecyclerView para ver cómo hacerlo).

PreferenceFragmentCompat:

Muestra una jerarquía de objetos Preference en forma de lista. Este objeto se usa a fin de crear una pantalla de configuración para tu app.

6. Bibliografía

[1] Android. Programación Multimedia y de dispositivos móviles. Garceta.
[2] Programación Multimedia y Dispositivos Móviles. Editorial Síntesis
[3] Android App Development. For Dummies
[4] Beginning Android Programming with Android Studio. Wrox
[5] Java Programming for Android Developers. For Dummies
[6] Android Programming with Android Studio
[7] https://academiaandroid.com/
[8] https://developer.android.com/
[9] https://www.redhat.com
[10] https://abhiandroid.com/
[11] https://desarrolloweb.com