INTERFACES GRÁFICAS DE ANDROID

"Activities"

Jonatan Fuentes García. - 2ºDam-Semi.

Índice

Activity

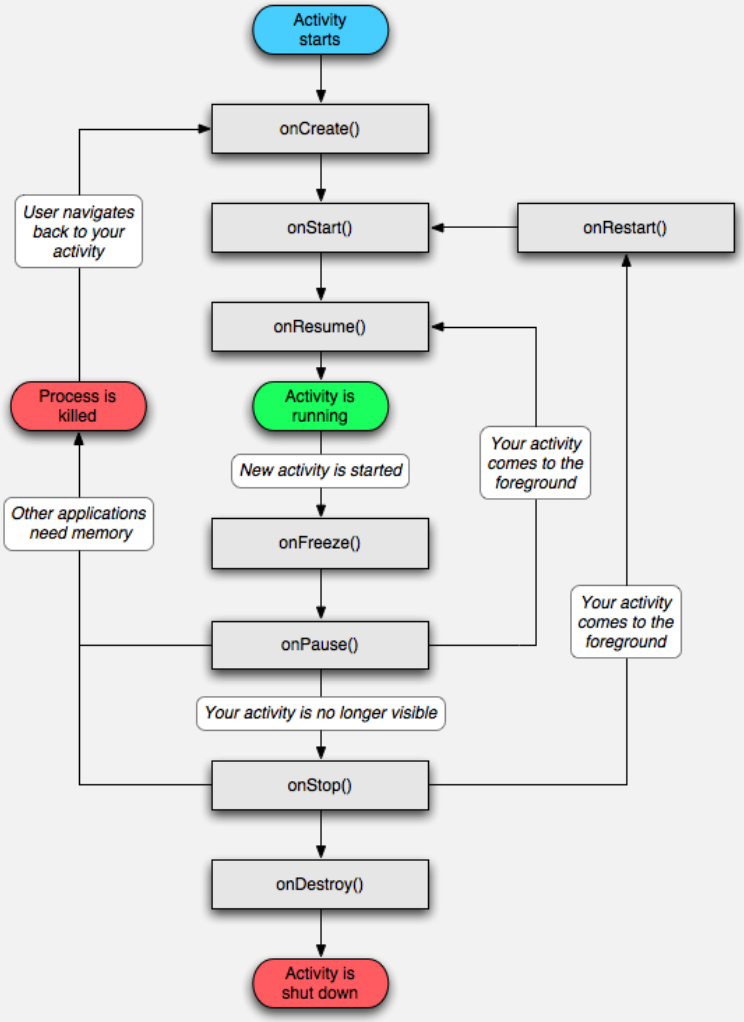
La clase Activity es una clase que está enfocada a la interacción con el usuario. Para crear la ventana del activity deberíamos llamar en primer lugar al método setContentView (View), generalmente se representan en un ventana de pantalla completa pudiendo modificarse y usarse de otras formas, tales como: Ventanas flotantes llamando al método (R.att.windowsIsFloating) o ventanas incorporadas a otra actividad (ActivityGroup).

Hay dos métodos principales que implementan las subclases de Activity:

- **OnCreate(bundle):** Inicializa la actividad para posteriormente llamar a setContentView(int) con un diseño para el IU, se usará findViewById(int) para recuperar a los widgets de esa IU para poder interactuar con ellos mediante programación.

- **onPause():** El usuario deja su actividad y cualquier cambio realizado por el usuario debe registrares y guardarse en el ContentProvider donde se retienen dichos datos.

Todas los Activities deben estar correctamente declaradas en el paquete AndroidManifest.xml.

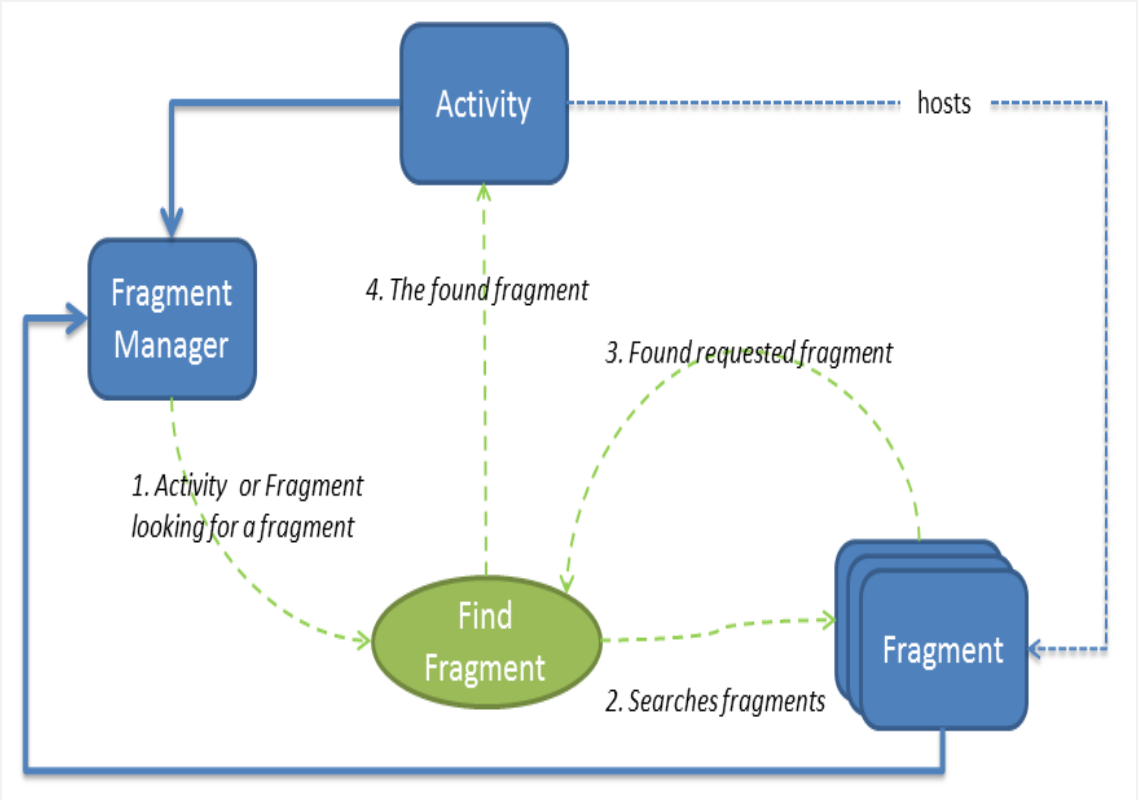


Fragmentos

La clase Fragment junto con su subclase FragmentActivity sirven para crear interfaces de usuario más sofisticadas para pantallas más grandes y ayuda a escalar su aplicación entre pantallas pequeñas y grandes.

Un fragment es una parte de la interfaz del usuario incluida en la clase Activity puedes tanto crear múltiples fragmentos en una actividad, como, usar un fragmento en varias Activities. Siempre estará incluido en el interior de la Activity viéndose afectada la interacción de dicho Fragment por el de la Activity anfitriona, es decir, cuando la Activity está pausada también lo están todo sus fragments y cuando se destruye ocurre lo mismo con todos sus frament s.

Cuando añades un fragment a una activity, este se ubica en un ViewGroup según las vistas de jerarquías de una activity, puedes agregarlo con un elemento -fragment- o -ViewGroup- existente en el código de la aplicación, los fragments también pueden ser utilizados en segundo plano siendo invisible para la activity.



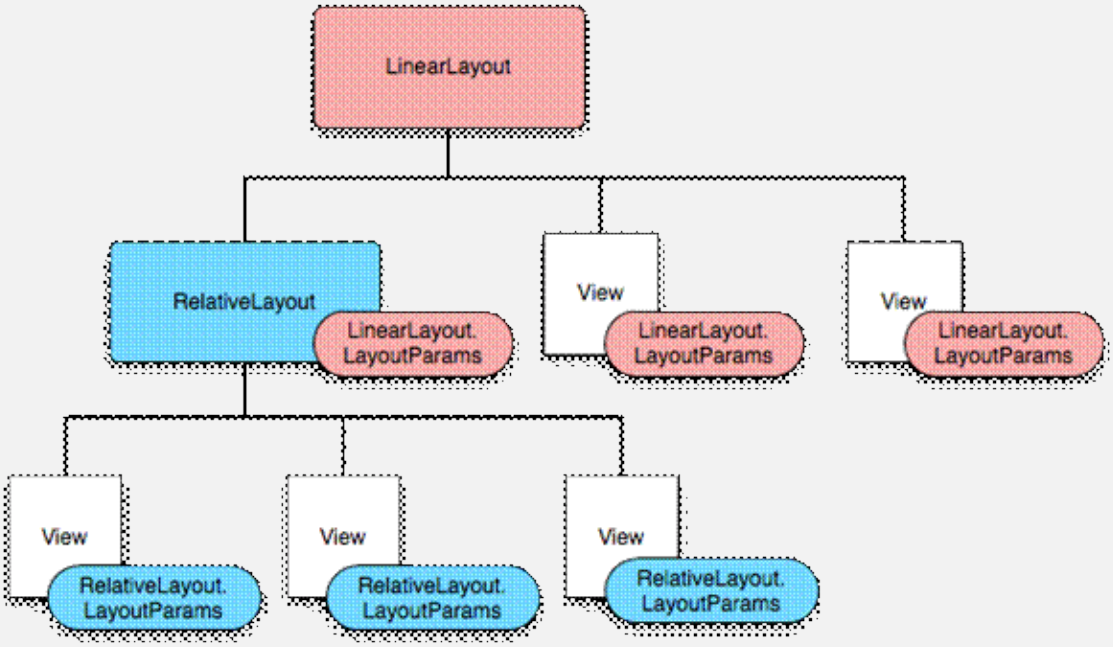
View

Es un objeto de la clase android.view.View, una estructura de datos cuyas propiedades contiene los datos de una capa, la información específica de la pantalla y permite establecer los layouts. Una view tiene layout, drawing, focus change, scrolling, etc..

La clase view es útil como clase base para los widgets, que son unas subclases ya implementadas que dibujan los elementos en la pantalla. Los widgets contienen sus propias medidas, pero puedes usarlas para construir tu interfaz más rápidamente. La lista de widgets que puedes utilizar incluye Text, EditText, InputMethod, MovementMethod, Button, RadioButton, CheckBox, y ScrollView.

También existen los denominados viewgroup que son objetos de la clase android.view.Viewgroup, es un objeto especial de view cuya función es contener y controlar la lista de views y de otros viewgroups. Los viewgroups te permiten añadir estructuras a la interfaz y acumular complejos elementos en la pantalla que son diseccionados por una sola entidad.

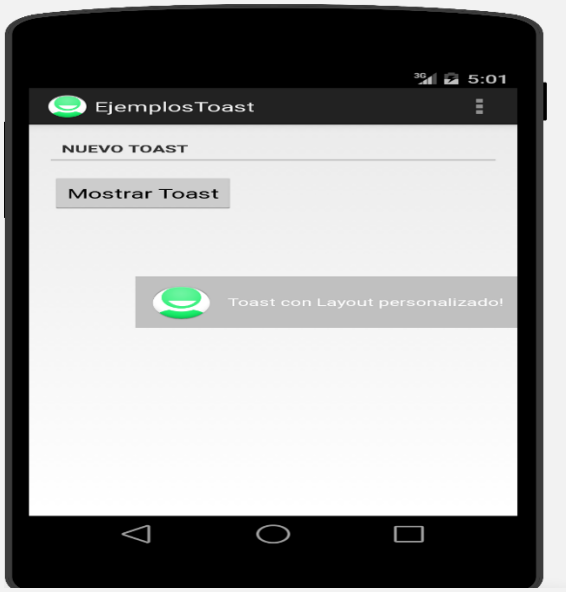
La clase viewgroup es útil como base de la clase layouts, que son subclases implementadas que proveen los tipos más comunes de los layouts. Los layouts proporcionan una manera de construir una estructura para una lista de views.



Los views y viewgroups deben estar contenidos en los layouts, los cuales contienen otros elementos presentes en una vista. Dentro de cada layout podemos poner todos los elementos necesarios, incluidos otros layouts. Así conseguiremos estructurar la pantalla de la manera deseada.

Toast

Un Toast es básicamente un mensaje que aparece y a los pocos segundos desaparece, generalmente se utilizan para informar de algo al usuario, es una de las mejores opciones si queremos mostrar un mensaje o aviso rápido en las aplicaciones.



En primer lugar se crea una instancia de Toast con un maketest() métodos. Pasaremos a tener tres parámetros: la aplicación Context, el mensaje de texto y la duración del brindis. Devuelve el objeto Toast incializado correctamente y para mostrar la notificación utilizaríamos show().

También se podrían encadenar sus métodos, quedando como en el siguiente ejemplo:

“Toast.makeText (contexto, texto, duración). Show()”

Utilizaremos el método setGravity (int,int,int) si lo que queremos es cambiar la posición del toast, introduciendo como parámetros un Gravity constante, eje x, eje y, como por ejemplo;

“Toast.setGravity (Gravity.top p gravity.left, 0 , 0)”

Para crear una vista personalizada del Toast podremos utilizar el método setView(View) y lo haríamos de la siguiente forma; Definiremos un diseño de vista en XML o en el c´´odigo de nuestra aplicación y pasaremos el método setView(View). Por ejemplo:

```
TextView textView = (TextView) toast_layout.findViewById(R.id.toastMessage);
        textView.setText(R.string.mensaje);
    toast.setDuration(Toast.LENGTH_SHORT);
    toast.show();
```

Listener

Denominamos Listener a los eventos de entrada que interceptan dichos eventos desde las interación del usuario con la aplicación, consiste en capturar los eventos desde el objeto especifico con el que interactua el usuario.

Dentro de las clases de objetos View que se usarán, observaremos varios métodos de callback públicos que pueden ser útiles. Estas interfaces llamadas Gestores de Eventos permiten capturarla interacción con el usuario podremos incluir estos métodos de callback;

- **onclick():** Desde View.OnClickListener. Este método se llama cuando el usuario toca el elemento (en el modo táctil), o selecciona el elemento con las teclas de navegación o la bola de seguimiento y presiona la tecla “Entrar” adecuada o la bola de seguimiento.
- **onLongClick():** Desde View.OnLongClickListener. Este método se llama cuando el usuario toca y mantiene presionado el elemento (en el modo táctil), o selecciona el elemento con las teclas de navegación o la bola de seguimiento y mantiene presionada la tecla “Entrar” adecuada o la bola de seguimiento (durante un segundo).
- **onFocusChange():** Desde View.OnFocusChangeListener. Este método se llama cuando el usuario navega hacia el elemento o sale de este utilizando las teclas de navegación o la bola de seguimiento.
- **onKey():** Desde View.OnKeyListener. Este método se llama cuando el usuario se centra en el elemento y presiona o libera una tecla física en el dispositivo.
- **onTouch:** Desde View.OnTouchListener. Este método se llama cuando el usuario realiza una acción calificada como un evento táctil, por ejemplo, presionar, liberar o cualquier gesto de movimiento en la pantalla (dentro de los límites del elemento).
- **onCreateContextMenu():** Desde View.OnCreateContextMenuListener. Este método es llamado cuando se crea un Context Menu (como resultado de un "clic largo" sostenido). Consulta la explicación sobre menús contextuales en la guía para desarrolladores Menus.

  Estos métodos son los únicos habitantes de su respectiva interfaz. Para definir uno de estos métodos y manejar sus eventos, implementa la interfaz anidada en su actividad o defínela como una clase anónima. Luego, pasa una instancia de tu implementación al método View.set...Listener() respectivo. (P. ej., llama a setOnClickListener() y pasa tu implementación deOnClickListener).

```
2
3   import android.os.Bundle;
4   import android.support.v7.app.AppCompatActivity;
5   import android.view.View;
6
7   public class ActividadBotones extends AppCompatActivity implements View.OnClickListener {
8
9       Class 'ActividadBotones' must either be declared abstract or implement abstract method 'onClick(View)' in 'OnClickListener'
10      protected void onCreate(Bundle savedInstanceState) {
11          super.onCreate(savedInstanceState);
12          setContentView(R.layout.actividad_botones);
13      }
14
15  }
16
```