Introducción a Android - Ejercicios

Índice

1. Nuestra primera aplicación Android

En este primer ejercicio practicaremos los pasos vistos en clase para la creación de una nueva aplicación en Android, prestando especial atención al tema de los recursos, el cual puede ser descuidado si no se le presta la suficiente atención desde el principio.

- Creamos en primer lugar un proyecto Android cuyo nombre será **HolaMundo**. Escogeremos la versión 2.3.1 de la plataforma, le daremos a la aplicación el nombre *Hola Mundo*, y crearemos una actividad principal de nombre `HolaMundoActivity.java`, que se encontrará dentro del paquete `es.ua.jtech.android.holamundo`.
- Ejecuta la aplicación en el emulador y comprueba que funciona correctamente. Recuerda que el AVD creado debe tener instalada al menos la versión 2.3.1 de la plataforma para que nuestra aplicación pueda ser ejecutada.
- La interfaz de nuestra aplicación contiene una vista de tipo `TextView`, que mostrará el mensaje *Hello World, HolaMundoActivity!*. Dicha interfaz está definida en los recursos de la aplicación, dentro del archivo `main.xml` de la carpeta `/res/layout/`. Abre el fichero y mira la definición de la vista `TextView`. El elemento XML que la representa contiene un atributo `android:text` que indica el texto a mostrar en pantalla. El valor de ese atributo hace referencia a una cadena definida en el archivo `strings.xml` de la carpeta `/res/values/`. Cambia la cadena en dicho archivo para que el texto mostrado por pantalla sea *Hola Mundo*.
- Vamos a cambiar también el icono de nuestra aplicación. Cualquier proyecto nuevo en Android incluirá un icono por defecto definido en tres diferentes resoluciones, que podrá encontrarse en las carpetas `drawable-hdpi`, `drawable-mdpi` y `drawable-ldpi` dentro de la carpeta de recursos. En cada una de estas carpetas encontraremos un fichero llamado *icon.png* con la resolución correspondiente (alta, media y baja, respectivamente). El sistema escogerá qué icono mostrar en función de la resolución de la pantalla. En las plantillas de los ejercicios se han incluido tres archivos .png conteniendo un icono alternativo para nuestra aplicación. El icono se encuentra almacenado en tres carpetas a distintas resoluciones. El nombre de los archivos es *icono_alternativo.png*. Guarda los ficheros en las carpetas correspondientes dentro de los recursos de la aplicación, y modifica el archivo Manifest del proyecto para que se utilice ese nuevo icono.
- Comprueba con varios AVDs que hagan uso de diferentes resoluciones que el icono se muestra correctamente en el menú de aplicaciones.
- Ahora haremos los cambios necesarios para permitir a nuestra aplicación mostrar el texto en varios idiomas. Para ello creamos una nueva carpeta dentro de la carpeta de recursos a la que llamaremos *values-EN*. Dentro de dicha carpeta crearemos un nuevo fichero *strings.xml*. Este fichero será una copia de *values/strings.xml*, con la única diferencia de que el texto *Hola Mundo* será sustituido por *Hello World*. Al hacer esto, nuestra aplicación mostrará el texto en inglés cuando el dispositivo esté configurado para mostrar los textos en ese idioma, y en español en el resto de casos.

- Prueba a cambiar la configuración de idioma de tu AVD y observa los resultados.

Una vez hecho todo lo anterior vamos a ver cómo podemos modificar el texto de la vista `TextView` desde el código fuente. Esto es algo que se verá en detalle en posteriores sesiones. Debemos seguir los siguientes pasos:

- En primer lugar añadimos una nueva función de nombre `devuelveEntero()` que no recibirá ningún parámetro y devolverá el entero 0.
- Para poder acceder al `TextView`, éste debe tener asignado un identificador. Añadimos el siguiente atributo al elemento `TextView` dentro del archivo `main.xml`:

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

- A partir de este momento podremos acceder al `TextView` en nuestro código por medio del identificador `R.id.texto`.
- Al final del método `onCreate()` creamos una variable para almacenar el objeto que representará el `TextView` en nuestro código y le añadimos el valor devuelto por la función `devuelveEntero()` utilizando el siguiente código:

```
TextView texto = (TextView)findViewById(R.id.texto);
texto.append(" " + new Integer(devuelveEntero()).toString());
```

Nota:

Los elementos de la interfaz gráfica de una actividad son accedidos en nuestro código mediante un identificador de recurso.

2. Depuración en Android por medio de LogCat

Entre las plantillas de la sesión encontrarás el proyecto Fibonacci. Este proyecto consta de una única actividad cuya interfaz gráfica está compuesta por un elemento de tipo `TextView` y dos botones. El `TextView` muestra los dos primeros elementos de la serie de Fibonacci. La serie de Fibonacci se define recursivamente de la siguiente forma:

- El elemento en la posición **n** se calcula como la suma de los elementos en las posiciones **n-1** y **n-2**
- Los dos primeros elementos de la serie valen ambos 1

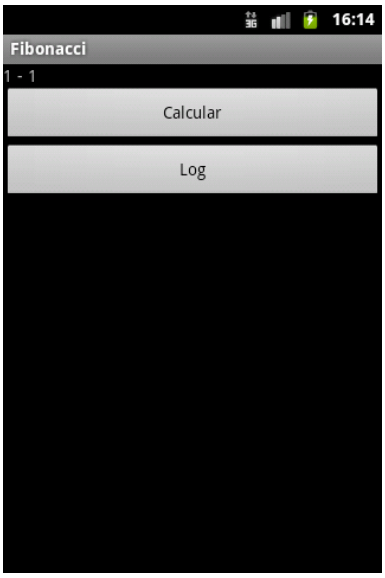

Diagrama de la pila de actividades

En primer lugar haremos los cambios necesarios para que cada vez que se pulse el botón superior se vayan añadiendo elementos de la serie de Fibonacci al `TextView`:

- Crea dos atributos de la clase `FibonacciActivity` para almacenar los dos términos anteriores de la serie de Fibonacci. Estos valores serán usados para calcular un nuevo término cuando se pulse el botón. Estos atributos serán de tipo entero, se llamarán `termino1` y `termino2`, y en ambos casos su valor inicial será 1.
- Para poder añadir un manejador para el botón superior primero hemos de asignarle un identificador en el archivo `main.xml` de la carpeta `layout`. Edita el archivo y añade el siguiente atributo al primero de los botones:

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

- Ahora ya podemos acceder al botón desde el código Java. Añade el siguiente código al método `onCreate()`:

```
Button botonCalcular = (Button)findViewById(R.id.botonCalcular);
botonCalcular.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
                // Manejador para el click del botón

        }
});
```

- Dentro del manejador del botón haz los cambios necesarios para añadir al texto mostrado por pantalla el siguiente término de la serie de Fibonacci. Cada vez que se pulse el botón se deberá añadir al `TextView` el siguiente elemento de la serie y actualizar los atributos `termino1` y `termino2` como corresponda.

A continuación tienes una captura de pantalla que muestra el estado de la aplicación tras haber pulsado unas cuantas veces el botón superior:

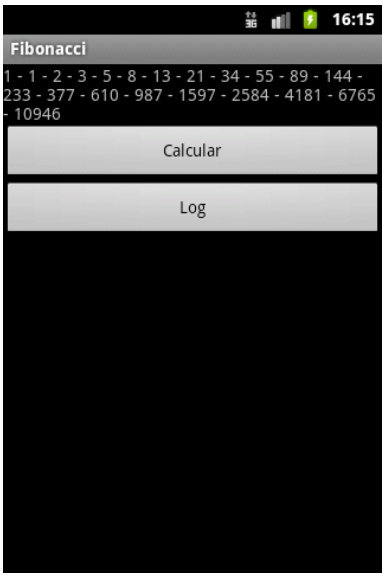

Diagrama de la pila de actividades

El siguiente paso será practicar el uso del sistema de logging de Android (*LogCat*) para depurar nuestras aplicaciones. *LogCat* consiste en un registro a través del cual el SDK va mostrando el estado de las aplicaciones ejecutadas en el dispositivo, excepciones, y otros mensajes. Dentro de Eclipse es posible visualizar el contenido de *LogCat* a partir de la pestaña situada en la parte inferior de la ventana. Si no aparece allí podremos hacer que se muestre de nuevo seleccionando la opción *Window->Show View->Other*, y dentro de la ventana que aparecerá a continuación, la opción LogCat dentro de la carpeta Android.

Para mostrar textos en *LogCat* deberemos hacer uso de alguno de los métodos estáticos proporcionados por la clase Log a tal efecto. Los más utilizados habitualmente son el método `e` y el método `i`. Ambos métodos reciben dos cadenas como parámetro. La primera de ellas se mostrará en el LogCat como un identificador de la aplicación que está lanzando el mensaje en el *LogCat*, mientras que la segunda cadena será el mensaje en sí mismo. La diferencia entre ambos métodos es que el primero se utiliza para lanzar mensajes de error, mientras que el segundo se utiliza para lanzar mensajes de información. Si se examina el *LogCat* desde la interfaz de Eclipse ambos tipos de mensaje se verán de diferente color.

Haremos los siguientes cambios a nuestra aplicación:

- Añadir un manejador para el botón inferior, siguiendo el ejemplo que se ha proporcionado previamente para el botón superior.
- Hacer que al pulsar dicho botón se muestre el valor de las variables `termino1` y `termino2` en el LogCat. Se puede usar cualquiera de los dos métodos vistos (`e` o `i`).
- Observa el funcionamiento de tu aplicación. Pulsa el botón inferior tras haber pulsado varias veces el botón superior y comprueba qué mensajes aparecen en el *LogCat*.

3. Estados de ejecución

Ya hemos visto que una actividad en Android puede pasar por diferentes estados de ejecución dependiendo de, por ejemplo, si se encuentra o no visible en ese momento. También hemos visto el concepto de pila de actividades.

En las plantillas de la sesión tienes el proyecto *EstadosEjecucion*, el cual incluye dos actividades: `Actividad1` y `Actividad2`. La actividad principal es `Actividad1`. Esta actividad contiene un botón. Vamos a hacer los cambios necesarios para que al pulsar dicho botón la actividad `Actividad2` pase a estar en primer plano. Para ello vamos a hacer uso de algunos conceptos que veremos en la siguiente sesión, así que de momento no es necesario tener un conocimiento exhaustivo de lo que estamos haciendo. Sigue los siguientes pasos:

- Crea un manejador para el botón, siguiendo los pasos vistos en ejercicios anteriores.
- Dentro del manejador del botón crea un objeto de la clase `Intent` de la siguiente forma:

```
Intent intent = new Intent(Actividad1.this, Actividad2.class);
```

- Los `Intent` son un elemento clave en las aplicaciones Android. Aprenderemos más sobre ellos en la siguiente sesión. Mientras tanto lo único que debéis saber es que necesitáis un `Intent` para llamar a una nueva actividad.
- Haz una llamada a `startActivity` dentro del manejador del botón, pasando como parámetro el `Intent` creado:

```
Actividad1.this.startActivity(intent);
```

- La llamada a `startActivity` hará que `Actividad2` pase a estar activa en primer plano. La actividad `Actividad1` pasará entonces a la pila de actividades, a la espera de que vuelva a estar visible. Si pulsamos le botón para volver atrás en nuestro dispositivo, la actividad `Actividad2` será destruida, mientras que la actividad `Actividad1` saldrá de la pila de actividades para volver a estar visible y activa.
- Ejecuta y prueba tu aplicación. Comprobarás que ésta no funciona correctamente. Haz los cambios necesarios en el archivo Manifest para que el problema desaparezca.

A continuación haremos uso de los `Toast`, otro mecanismo de comunicación entre la aplicación y el usuario. Mediante un `Toast` podremos mostrar un mensaje flotante sobre la aplicación, para avisar al usuario de que se ha producido algún evento. Para mostrar un Toast, deberemos hacer una llamada de la siguiente manera:

```
Context context = getApplicationContext();
CharSequence text = "Ejemplo de Toast";
int duration = Toast.LENGTH_SHORT; // También podría ser Toast.LENGTH_LONG

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

Nosotros usaremos `Toast` para comprobar el funcionamiento de los manejadores de

evento para el cambio de estado de las actividades. En concreto, tanto en `Actividad1` como en `Actividad2`, haremos que los siguientes manejadores muestren un `Toast`: `onCreate`, `onStart`, `onResume`, `onPause`, `onStop` y `onDestroy`. El `Toast` mostrará el nombre del método y de la actividad. Prueba la aplicación y navega entre las dos actividades, comprobando qué métodos entran en funcionamiento.

Aviso:

No olvides llamar al método correspondiente de la superclase en cada uno de los manejadores anteriores (por ejemplo, no olvides incluir dentro del método `onStart()` una llamada a `super.onStart()`.

4. Esperando el resultado de otra actividad (*)

En el ejercicio anterior hemos visto cómo lanzar una actividad desde cualquier otra mediante el método `startActivity`. Otra forma de lanzar actividades en Android es hacer uso del método `startActivityForResult`. La actividad que hace una llamada a `startActivityForResult` quedará a la espera de que la actividad llamada termine y devuelva un determinado resultado. Vamos a probarlo con una aplicación sencilla basada en el ejemplo del contador visto en clase.

En las plantillas de la sesión se incluye el proyecto `ActividadesResult`. Este proyecto consta de dos actividades, `ActividadPrincipal` y `ContarActivity`. La primera de ellas contiene una vista de tipo `TextView` que inicialmente muestra el valor cero, y un botón para lanzar la segunda actividad. La segunda actividad contiene el ejemplo visto en clase con un botón que muestra el número de veces que éste se ha pulsado. El objetivo del ejercicio es conseguir que al volver de la segunda actividad a la pimera mediante el uso del botón de volver atrás de nuestro dispositivo, se muestre en el `TextView` el número mostrado sobre el botón en la segunda actividad.

Los cambios a realizar serán los siguientes:

- Añade un manejador para el botón de la actividad `ActividadPrincipal`. Desde este manejador haremos uso de `startActivityForResult` para mostrar la actividad `ContarActivity`:

```
Intent intent = new Intent(ActividadPrincipal.this, ContarActivity.class);
// El segundo parámetro de la siguiente función
// es un identificador de petición
// Podría ser definido como una constante
startActivityForResult(intent, 1);
```

- Añade el siguiente código para que cambie el texto del `TextView` de la actividad principal al volver desde la segunda actividad:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent
data) {
        super.onActivityResult(requestCode, resultCode, data);
```

```
        if(resultCode==RESULT_OK && requestCode==1){
                String msg = data.getStringExtra("contador");
                TextView texto = (TextView)findViewById(R.id.texto);
                texto.setText(msg);
        }
}
```

- El método `onActivityResult` es llamado cuando una actividad que fue comenzada con un `Intent` termina. Debemos comprobar antes de hacer nada que el evento se ha producido debido a la terminación de la actividad que se lanzó con un indicador de petición igual a 1 (que fue el valor que se pasó como segundo parámetro en nuestro `startActivityForResult`).
- Por último, en la actividad `ContarActivity`, añadimos el siguiente código dentro del manejador del evento del botón, tras la instrucción que cambia la etiqueta del botón. El código crea un Intent para que sea enviado a `ActividadPrincipal` en el caso en el que se vuelva a ella. Este código podría ser escrito en otra parte de la clase, pero lo dejaremos aquí de momento. Trataremos este tema en más detalle en la siguiente sesión.

```
Intent intent = new Intent();
intent.putExtra("contador", boton.getText());
setResult(RESULT_OK, intent);
```

- Prueba la aplicación. Si no has hecho ningún cambio más ésta no funcionará. Haz los cambios necesarios en el Manifest para que se solucione el problema.