Laboratorio 4.3

Apellidos: Moreno Vera
Nombres: Felipe Adrian
Código: 20120354I
Asignatura: Programación en Dispositivos Móviles (CC481)

2016 - I

Indice

Actividad 1

1. Lo primero será crear nuestra interfaz como se muestra en la siguiente imagen y el xml de la vista.

```
<ProgressBar
android:id="@+id/pbarProgreso"
style="?android:attr/progressBarStyleHorizontal"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:indeterminate="false"
android:layout_marginLeft="5dip"
android:layout_marginRight="5dip"
android:max="100" />
<Button
        android:id="@+id/btnSinHilos"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="btnSinhilos"
        android:text="Ejecución sin hilos" />
```

2. Nos vamos a la vista controlador de nuestra actividad principal.

1. Vemos nuestras propiedades. Realice lo mismo con los Button.

```
private Button btnSinHilos;
private Button btnHilo;
private Button btnAsyncTask;
private Button btnCancelar;
private Button btnAsyncDialog;
private ProgressBar pbarProgreso;
private ProgressDialog pDialog;
private MiTareaAsincrona tarea1;
private MiTareaAsincronaDialog tarea2;
```

2. A continuación cargamos nuestra vista teniendo en cuenta que debemos capturar el PogressBar. ¿Habría que capturar los Button?

```
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btnSinHilos = (Button)findViewById(R.id.btnSinHilos);
        btnHilo = (Button)findViewById(R.id.btnHilo);
        btnAsyncTask = (Button)findViewById(R.id.btnAsyncTask);
        btnCancelar = (Button)findViewById(R.id.btnCancelar);
        btnAsyncDialog = (Button)findViewById(R.id.btnAsyncDialog);
        pbarProgreso = (ProgressBar)findViewById(R.id.pbarProgreso);
}
```

Si para darles funcionalidad, y que no haya solamente pulsasiones erróneas.

3. Creamos un método para prolongar una tarea 1 segundo.

```
private void tareaLarga() {
        try {
                Thread.sleep(1000);
        } catch(InterruptedException e) { }
}
```

4. Añadimos funcionalidad a nuestro Button haciendo una ejecución de 10 segundos. ¡Tenga en cuenta que los eventos deben estar dentro de onCreate!

```
btnSinHilos.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
                pbarProgreso.setMax(100); //valor máximo de ProgressBar
                pbarProgreso.setProgress(0); // Empieza vacía ProgressBar
                for(int i=1; i<=10; i++) {
                        tareaLarga();
                        pbarProgreso.incrementProgressBy(10); //incremento
                }
        Toast.makeText(MainHilos.this, "Tarea finalizada!",
        Toast.LENGTH_SHORT).show();
        }
});
```

5. Ejecute la aplicación y comente lo que sucede.

Cuando apretamos el boton “Sin hilo”, la pantalla queda congelada, y tras 10 segundos vuelve a funcionar, saliendo el Toast de Tarea Finalizada.

6. Realice Click en otro punto del emulador y fíjese en el diálogo que nos aparece. Ese error quiere decir que está realizando una operación de larga duración en el hilo principal y es por eso que no responde hasta que acaba su ejecución.

```
I/Choreographer: Skipped 601 frames!  The application may be doing too much work on its main thread.
I/Choreographer: Skipped 600 frames!  The application may be doing too much work on its main thread.
```

3. Volvamos a nuestro código y vamos a resolver este problema. Para que nuestro proceso se ejecute en un segundo hilo de ejecución tendremos que invocar al objeto Runnable de nuestra clase Threads.

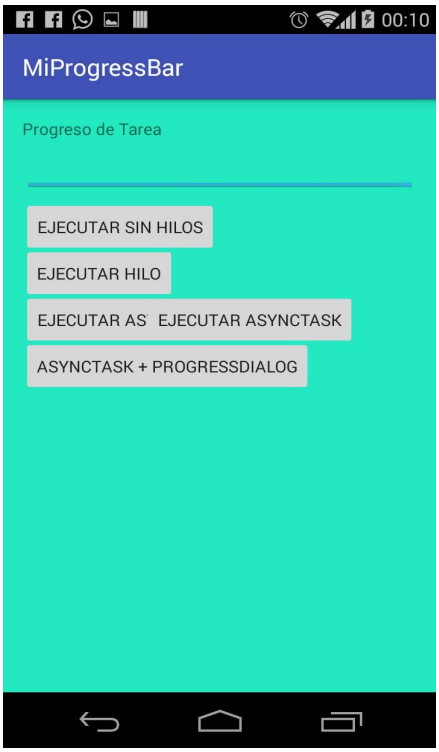
1. Recuerde que para actuar sobre cada control de la clase principal hay que utilizar el método post() o runOnUIThread(). El siguiente código pertenece al evento onClick() del Button “Con Hilos”.

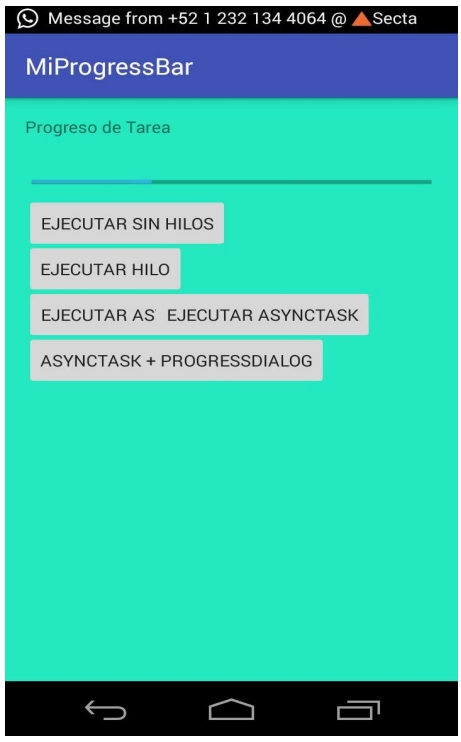
```
new Thread(new Runnable() {
public void run() {
        pbarProgreso.post(new Runnable() {
        public void run() {
                pbarProgreso.setProgress(0);
```

```
        }
});
for(int i=1; i<=10; i++) {
        tareaLarga();
        pbarProgreso.post(new Runnable() {
                public void run() {
                        pbarProgreso.incrementProgressBy(10);
                }
        });
}
runOnUiThread(new Runnable() {
        public void run() {
                Toast.makeText(MainHilos.this, "Tarea finalizada!",
                Toast.LENGTH_SHORT).show();
        }
});
}
}).start();
```

4. Ejecutamos la aplicación y vemos como la creación de un hilo secundario ha librado ese hilo principal pudiendo interactuar con él.

Ahora la aplicación ejecuta normalmente, y se muestra el progreso de la barra conforme pasan los segundos.





Actividad 2

1. Una vez leido y comprendido el texto anterior creamos nuestra clase ASyncTask en nuestro clase principal.

```
private class MiTareaAsincrona extends AsyncTask<Void, Integer, Boolean>{
	@Override
	protected Boolean doInBackground(Void... params) {
		for(int i=1; i<=10; i++) {
			tareaLarga();
			publishProgress(i*10);
			if(isCancelled())
			break;
		}
		return true;
	}
	@Override
	protected void onProgressUpdate(Integer... values) {
		int progreso = values[0].intValue();
		pbarProgreso.setProgress(progreso);
	}
	@Override
	protected void onPreExecute() {
		pbarProgreso.setMax(100);
		pbarProgreso.setProgress(0);
	}
	@Override
	protected void onPostExecute(Boolean result) {
	if(result)
		Toast.makeText(MainHilos.this, "Tarea finalizada!",
		Toast.LENGTH_SHORT).show();
	}
	@Override
	protected void onCancelled() {
		Toast.makeText(MainHilos.this, "Tarea cancelada!",
		Toast.LENGTH_SHORT).show();
	}
}
```

2. Vemos anteriormente como se ha introducido el método onCancelled() que lo que se pretende es que nos salga un objeto tipo Toast en caso de que se pulse el Button Cancelar. Añada esta función.
3. Ejecute la aplicación y verifique su funcionamiento.

Apretamos cancelar =======>

4. Vamos a ver ya el último Button de nuestra aplicación en la que queremos que mientras se esté ejecutando nuestra aplicación el usuario no pueda interactuar con ella apareciendo un objeto tipo ProgressDialog.

1. . Del siguiente código perteneciente al evento de un Button explique cada método utilizado

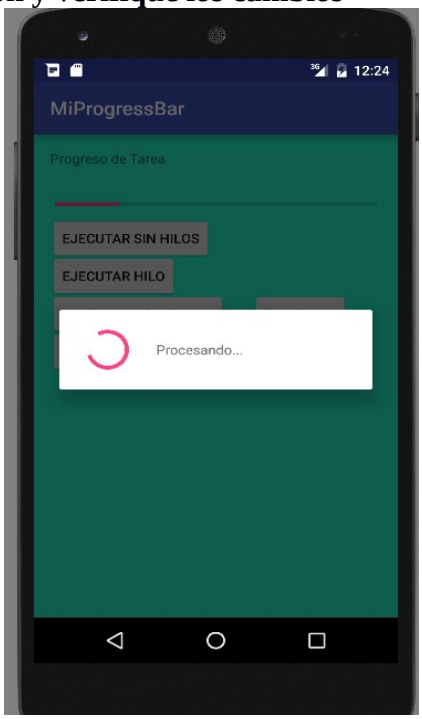
```
btnAsyncDialog.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
        // Instancia un nuevo ProgressDialog
        pDialog = new ProgressDialog(MainHilos.this);
        // setea el estilo del ProgressDialog
        pDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);
        // Setea el mensaje que va a mosrrar mientras se ejecuta
        pDialog.setMessage("Procesando...");
        // Setea la opción de ser cancelable (con true) si fuera falsse, no se cancela.
        pDialog.setCancelable(true);
        // Le da el máximo numero, similar al PrograssBar
        pDialog.setMax(100);
        // Instancia la clase MiTareaAsincronaDialog
        tarea2 = new MiTareaAsincronaDialog();
```

```
            tarea2.execute(); // La ejecuta
        });
```

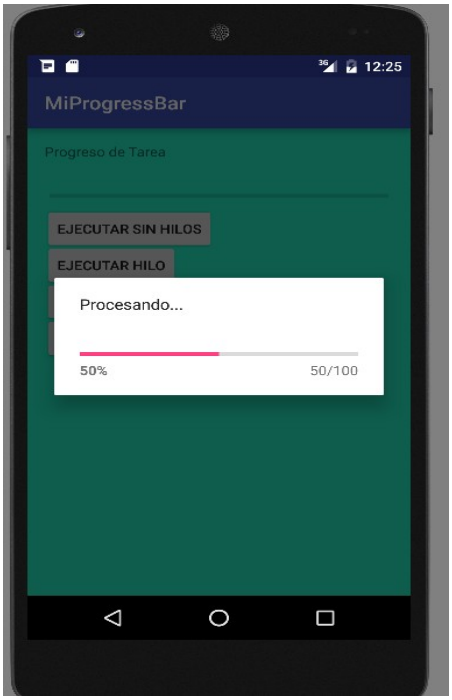
2. Ahora creamos una nueva clase ASyncTask para este nuevo Button. Explique las diferencias con la anterior.

```
private class MiTareaAsincronaDialog extends AsyncTask <Void,Integer, Boolean> {
        @Override
        protected Boolean doInBackground(Void... params) {
                for(int i=1; i<=10; i++) {
                        tareaLarga();
                        publishProgress(i*10);
                        if(isCancelled())
                                break;
                }
                return true;
        }
        @Override
        protected void onProgressUpdate(Integer... values) {
                int progreso = values[0].intValue();
                pDialog.setProgress(progreso);
        }
        @Override
        protected void onPreExecute() {
                pDialog.setOnCancelListener(new OnCancelListener() {
                        @Override
                        public void onCancel(DialogInterface dialog) {
                                MiTareaAsincronaDialog.this.cancel(true);
                        }
                });
                pDialog.setProgress(0);
                pDialog.show();
        }
        @Override
        protected void onPostExecute(Boolean result) {
                if(result) {
                        pDialog.dismiss();
                        Toast.makeText(MainHilos.this, "Tarea finalizada!",
                        Toast.LENGTH_SHORT).show();
                }
        }
}
```

3. Ejecute la aplicación y verifique los cambios



4. Cambie STYLE_SPINNER por STYLE_HORIZONTAL.



Actividad 3

1. Nuestro layout será simplemente un objeto ProgressBar y otro Button.
2. Vamos a crear nuestra clase IntentService. Para ello nos guiamos de los siguiente puntos.

1. Lo primero es crear nuestra clase JAVA llamándola "MyIntentService"
2. Una vez creada una segunda actividad recordad que se debe dejar constancia en el AndroidManifest dentro de la etiqueta application y luego de activity

<service android:name=".MiIntentService"></service>

```
<activity android:name=".MainActivity">
    <service android:name=".MiIntentService"></service>
```

3. Verificamos el constructor de la clase.

```
public MiIntentService() {
        super("MiIntentService");
}
```

4. Los nombres de las acciones deben ser un identificador único por lo que suelen preceder con el paquete Java (nos aseguramos).

public static final String ACTION_PROGRESO ="net.manwest.intent.action.PROGRESO";
public static final String ACTION_FIN ="net.manwest.intent.action.FIN";

```
public static final String ACTION_PROGRESO ="net.manwest.intent.action.PROGRESO";
public static final String ACTION_FIN ="net.manwest.intent.action.FIN";
```

5. Creamos la tarea en segundo plano en el método onHandleIntent()

```
protected void onHandleIntent(Intent intent) {
    int iter = intent.getIntExtra("iteraciones", 0);
    for (int i = 1; i <= iter; i++) {
      tareaLarga();
//Comunicamos el progreso
      Intent bcIntent = new Intent();
      bcIntent.setAction(ACTION_PROGRESO);
      bcIntent.putExtra("progreso", i * 10);
      sendBroadcast(bcIntent);
    }
    Intent bcIntent = new Intent();
    bcIntent.setAction(ACTION_FIN);
    sendBroadcast(bcIntent);
  }
```

6. Por último, y como los ejercicios anteriores, creamos la función que llevará el progreso

```
private void tareaLarga() {
  try {
    Thread.sleep(1000);
  } catch(InterruptedException e) { }
}
```

3. Una vez realizado el servicio tendremos que llamarlo desde la actividad principal para su ejecución.

1. Añadimos las propiedades de los objetos de Layout.

private Button btnEjecutar;
private ProgressBar pbarProgreso;

2. Dentro del método onCreate() añadimos el evento onClick de Button que es la llamada de un Intent tal cual lo hemos visto ya. También añadimos la captura de los objetos en el Layout.

```
btnEjecutar.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    Intent msgIntent = new Intent(MainActivity.this,MiIntentService.class);
    msgIntent.putExtra("iteraciones", 10);
    startService(msgIntent);
  }
});
```

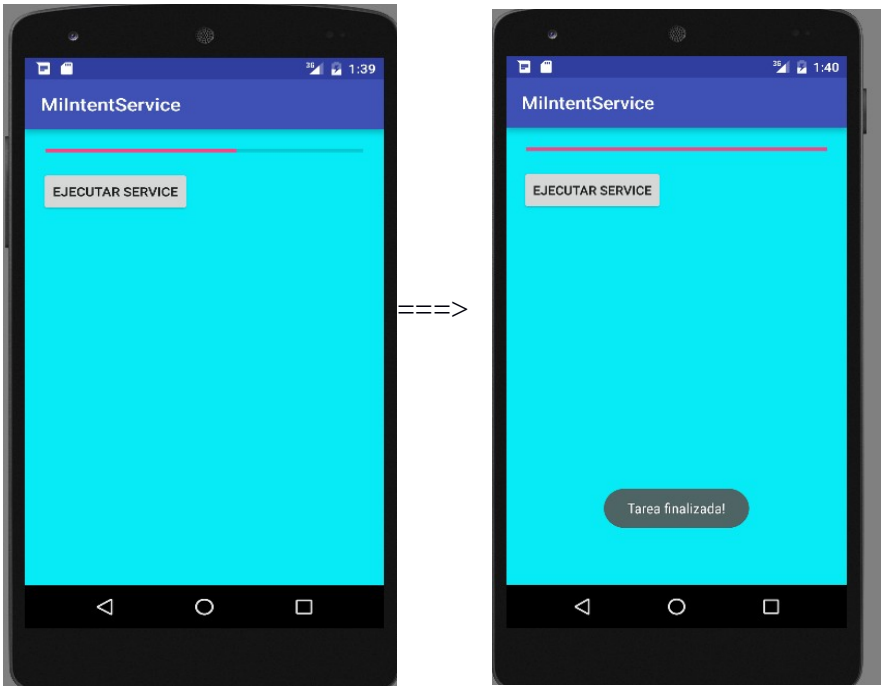
3. Aunque se va a ver en la siguiente práctica, para poder ver el progreso, ya que se ha enviado en formato Broadcast, tendremos que crear la correspondiente clase interna que extienda de BroadcastReceiver. En dicha clase el método onReceive() sera el gestor de mensajes de ACTION_PROGRESO y ACTION_FIN. Resaltar que en caso de recibir primero actualizará la barra y si es el segundo mostrará un objeto Toast.

```
public class ProgressReceiver extends BroadcastReceiver {
  @Override
  public void onReceive(Context context, Intent intent) {
    if ( intent.getAction().equals(
        MiIntentService.ACTION_PROGRESO)) {
      int prog = intent.getIntExtra("progreso", 0);
      pbarProgreso.setProgress(prog);  }
    else if(intent.getAction().equals(MiIntentService.ACTION_FIN)) {
      Toast.makeText(MainActivity.this, "Tarea finalizada!",
          Toast.LENGTH_SHORT).show();    }
  }
}
```

4. Por último, tendremos que crear un IntentFilter que asociaremos mediante addAction() posteriormente los instanciar dos mensajes nuestro a capturar BroadcastReceiver para y registrándolo pasándole la instancia y el filtro de mensajes. Este código va dentro de onCreate().

```
IntentFilter filter = new IntentFilter();
filter.addAction(MiIntentService.ACTION_PROGRESO);
filter.addAction(MiIntentService.ACTION_FIN);
ProgressReceiver rcv = new ProgressReceiver();
registerReceiver(rcv, filter);
```

5. Compila la aplicación ;-)



Practica

1. En este laboratorio hemos vistos dos componentes muy importantes de programación concurrente en Android pero, tal y como hemos descrito existe un tercero también de gran importancia: IntentService. Se pide un ejercicio sencillo en el que se utilice el IntentService con una barra de progreso explicando tanto el concepto teórico, diferencias con Threads y AsyncTask y el código. Enviar también el código trabajado.

La aplicación fue hecha en la actividad 3, asi que solo queda explicar las diferencias entre el uso de Threads y el AsyncTask.

La interface Runnable Thread es el núcleo de Java threading . La interface Runnable debe ser implementado por cualquier clase cuyas instancias están destinados a ser ejecutado por un hilo.

AsyncTask es una clase de conveniencia para hacer algún trabajo en un nuevo hilo y utilizar los resultados en el hilo del que lo llamaron (por lo general el hilo de interfaz de usuario) cuando haya terminado. Es sólo un envoltorio que utiliza un par de runnables pero maneja todos los entresijos de la creación del thread y manejos de mensajería entre los hilos.

Entonces, Si necesita un código simple, se puede usar AsyncTask pero si necesitas rapidez, usar el tradicional java thread.

Link del github con los códigos del laboratorio:

https://github.com/Jenazad/PDM/tree/master/Laboratorio_4

Referencias

http://www.ajpdsoft.com/modules.php?name=News&file=article&sid=603
http://developer.android.com/intl/es/reference/android/os/AsyncTask.html
http://stackoverflow.com/questions/16160064/using-asynctask
http://www.sgoliver.net/blog/tareas-en-segundo-plano-en-android-i-thread-y-asynctask/
http://stackoverflow.com/questions/17474818/difference-between-asynctask-and-thread-runnable