# Tareas en segundo plano en Android (II): IntentService | sgoliver.net

*sgoliver*

En el artículo anterior del Curso de Programación Android vimos cómo ejecutar tareas en segundo plano haciendo uso de hilos (Thread) y tareas asíncronas (AsyncTask). En este nuevo artículo nos vamos a centrar en una alternativa menos conocida, aunque tanto o más interesante, para conseguir el mismo objetivo: ejecutar una determinada tarea en un hilo independiente al hilo principal de la aplicación. Esta opción se llama IntentService, y no es más que un tipo particular de servicio Android que se preocupará por nosotros de la creación y gestión del nuevo hilo de ejecución y de detenerse a sí mismo una vez concluida su tarea asociada.

Como en el caso de las AsyncTask, la utilización de un IntentService va a ser tan sencilla como extender una nueva clase de IntentService e implementar su método onHandleIntent(). Este método recibe como parámetro un Intent, que podremos utilizar para pasar al servicio los datos de entrada necesarios.

A diferencia de las AsyncTask, un IntentService no proporciona métodos que se ejecuten en el hilo principal de la aplicación y que podamos aprovechar para "comunicarnos" con nuestra interfaz durante la ejecución. Éste es el motivo principal de que los IntentService sean una opción menos utilizada a la hora de ejecutar tareas que requieran cierta vinculación con la interfaz de la aplicación. Sin embargo tampoco es imposible su uso en este tipo de escenarios ya que podremos utilizar por ejemplo mensajes *broadcast* (y por supuesto su BroadcastReceiver asociado capaz de procesar los mensajes) para comunicar eventos al hilo principal, como por ejemplo la necesidad de actualizar controles de la interfaz o simplemente para comunicar la finalización de la tarea ejecutada. En este artículo veremos cómo implementar este método para conseguir una aplicación de ejemplo similar a la que construimos en el artículo anterior utilizando AsyncTask.

Empezaremos extendiendo una nueva clase derivada de IntentService, que para ser originales llamaremos MiIntentService. Lo primero que tendremos que hacer será implementar un constructor por defecto. Este constructor lo único que hará será llamar al constructor de la clase padre pasándole el nombre de nuestra nueva clase.

A continuación implementaremos el método onHandleIntent(). Como ya hemos indicado, este método será el que contenga el código de la tarea a ejecutar en segundo plano. Para simular una tarea de larga duración utilizaremos el mismo bucle que ya vimos en el artículo anterior con la novedad de que esta vez el número de iteraciones será variable, de forma que podamos experimentar con cómo pasar datos de entrada a través del Intent recibido como parámetro en onHandleIntent(). En nuestro caso de ejemplo pasaremos un sólo dato de entrada que indique el número de iteraciones. Por tanto, lo primero que haremos será obtener este dato a partir del Intent mediante el método getIntExtra(). Una vez conocemos el número de iteraciones, tan sólo nos queda ejecutar el bucle y comunicar el progreso tras cada iteración.

Como ya hemos comentado, para comunicar este progreso vamos a hacer uso de mensajes *broadcast*. Para enviar este tipo de mensajes necesitamos construir un Intent, asociarle un nombre de acción determinada que lo identifique mediante setAction(), e incluir los datos que necesitemos comunicar añadiendo tantos *extras* como sean necesarios mediante el método putExtra(). Los nombres de las acciones se suelen preceder con el paquete java de nuestra aplicación de forma que nos aseguremos que es un identificador único. En nuestro caso lo llamaremos "net.sgoliver.intent.action.PROGRESO" y lo definiremos como atributo estático de la clase para mayor comodidad, llamado ACTION_PROGRESO. Por su parte, los datos a comunicar en nuestro ejemplo será solo el nivel de progreso, por lo que sólo añadiremos un extra a nuestro intent con dicho dato. Por último enviaremos el mensaje llamando al método sendBroadcast() pasándole como parámetro el intent recién creado. Veamos cómo quedaría el código completo.

```
1
   public class MiIntentService extends IntentService {
2
       public static final String ACTION_PROGRESO =
3
           "net.sgoliver.intent.action.PROGRESO";
4
       public static final String ACTION_FIN =
5
           "net.sgoliver.intent.action.FIN";
6
       public MiIntentService() {
7
               super("MiIntentService");
8
           }
9
       @Override
10
       protected void onHandleIntent(Intent intent)
11
       {
12
           int iter = intent.getIntExtra("iteraciones", 0);
13
           for(int i=1; i<=iter; i++) {
14
               tareaLarga();
15
               //Comunicamos el progreso
16
               Intent bcIntent = new Intent();
17
               bcIntent.setAction(ACTION_PROGRESO);
18
               bcIntent.putExtra("progreso", i*10);
19
               sendBroadcast(bcIntent);
20
           }
21
           Intent bcIntent = new Intent();
22
           bcIntent.setAction(ACTION_FIN);
23
           sendBroadcast(bcIntent);
       }
24
       private void tareaLarga()
```

```
25
26
27
28
29          {
30              try {
31                  Thread.sleep(1000);
32              } catch(InterruptedException e) {}
33          }
34      }
35
36
37
38
```

Como podéis comprobar también he añadido un nuevo tipo de mensaje broadcast (ACTION_FIN), esta vez sin datos adicionales, para comunicar a la aplicación principal la finalización de la tarea en segundo plano.

Además de la implementación del servicio, recordemos que también tendremos que declararlo en el *AndroidManifest.xml*, dentro de la sección <application>:

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

Y con esto ya tendríamos implementado nuestro servicio. El siguiente paso será llamar al servicio para comenzar su ejecución. Esto lo haremos desde una actividad principal de ejemplo en la que tan sólo colocaremos una barra de progreso y un botón para lanzar el servicio. El código del botón para ejecutar el servicio será muy sencillo, tan sólo tendremos que crear un nuevo intent asociado a la clase MiIntentService, añadir los datos de entrada necesarios mediante putExtra() y ejecutar el servicio llamando a startService() pasando como parámetro el intent de entrada. Como ya dijimos, el único dato de entrada que pasaremos será el número de iteraciones a ejecutar.

```
1
2   btnEjecutar.setOnClickListener(new OnClickListener() {
3       @Override
4       public void onClick(View v) {
5           Intent msgIntent = new Intent(MainActivity.this, MiIntentService.class);
6           msgIntent.putExtra("iteraciones", 10);
7           startService(msgIntent);
8       }
9   });
```

Con esto ya podríamos ejecutar nuestra aplicación y lanzar la tarea, pero no podríamos ver el progreso de ésta ni saber cuándo ha terminado porque aún no hemos creado el BroadcastReceiver necesario para capturar los mensajes broadcast que envía el servicio durante su ejecución.

Para ello, como clase interna a nuestra actividad principal definiremos una nueva clase que extienda a BroadcastReceiver y que implemente su método onReceive() para gestionar los mensajes ACTION_PROGRESO y ACTION_FIN que definimos en nuestro IntentService. En el caso de recibirse ACTION_PROGRESO extraeremos el nivel de progreso del intent recibido y actualizaremos consecuentemente la barra de progreso mediante setProgress(). En caso de recibirse ACTION_FIN mostraremos un mensaje Toast informando de la finalización de la tarea.

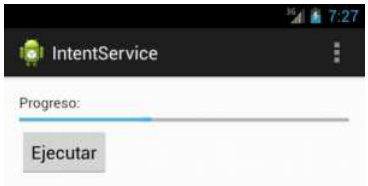
```
1
2   public class ProgressReceiver extends BroadcastReceiver {
3       @Override
4       public void onReceive(Context context, Intent intent) {
5           if(intent.getAction().equals(MiIntentService.ACTION_PROGRESO)) {
6               int prog = intent.getIntExtra("progreso", 0);
7               pbarProgreso.setProgress(prog);
8           }
9           else if(intent.getAction().equals(MiIntentService.ACTION_FIN)) {
10              Toast.makeText(MainActivity.this, "Tarea finalizada!", Toast.LENGTH_SHORT).show();
11          }
```

```
12 }
13
```

Pero aún no habríamos terminado dado, ya que aunque hayamos implementado nuestro BroadcastReceiver, éste no tendrá ningún efecto a menos que lo registremos con la aplicación y lo asociemos a los tipos de mensaje que deberá tratar (mediante un IntentFilter). Para hacer esto, al final del método onCreate() de nuestra actividad principal crearemos un IntentFilter al que asociaremos mediante addAction() los dos tipos de mensaje broadcast que queremos capturar, instanciaremos nuestro BroadcastReceiver y lo registraremos mediante registerReceiver(), al que pasaremos la instancia creada y el filtro de mensajes.

```
1 IntentFilter filter = new IntentFilter();
2 filter.addAction(MiIntentService.ACTION_PROGRESO);
3 filter.addAction(MiIntentService.ACTION_FIN);
4 ProgressReceiver rcv = new ProgressReceiver();
5 registerReceiver(rcv, filter);
```

Y con esto sí habríamos concluido nuestra aplicación de ejemplo. Si ejecutamos la aplicación en el emulador y pulsamos el botón de comenzar la tarea veremos cómo la barra de progreso comienza a avanzar hasta el final, momento en el que deberá aparecer el mensaje toast indicando la finalización de la tarea.




Android IntentService

Puedes consultar y/o descargar el código completo de los ejemplos desarrollados en este artículo accediendo a la página del curso en GitHub.

**Curso de Programación Android en PDF**

¿Te ha sido de utilidad el **Curso de Programación Android**? ¿Quieres colaborar de forma económica con el proyecto? Puedes contribuir con cualquier cantidad, unos céntimos, unos euros, cualquier aportación será bienvenida. Además, si tu aportación es superior a una pequeña cantidad simbólica recibirás como agradecimiento un documento con la última versión del curso disponible en formato PDF. Sea como sea, muchas gracias por colaborar!

Más información:

- Descubre cómo conseguir el curso en PDF
- Preguntas frecuentes