Servicios

Los servicios son componentes Android que no tienen interfaz gráfica. Funcionan de forma invisible para el usuario ejecutando un tratamiento o una tarea. Permiten ejecutar tratamientos bastante largos que no dependen de las interacciones del usuario.

Una de las particularidades de los servicios reside en la prioridad asignada por el sistema Android. Un servicio que ejecuta una tarea en segundo plano es prioritario respecto a una actividad que ejecuta también una tarea en segundo plano, lo que hace que los servicios estén menos expuestos a la liberación de recursos por el sistema.

Un servicio es (sólo) menos prioritario que una aplicación que se ejecuta en primer plano, lo cual reduce la probabilidad de que el sistema mate sus servicios.

1. Crear y utilizar un servicio

Para declarar un servicio hay que realizar dos pasos:

- En primer lugar, cree una clase que extienda la clase **Service**.
- Después, declare el servicio en el archivo del manifiesto de su aplicación.

La nueva clase que sobrecargue a la clase **Service** implementa los siguientes dos métodos:

- **onCreate**: inicialización del servicio y de su entorno.
- **onBind**: permite asociar un servicio a una actividad (véase la sección Asociar un servicio a una actividad).

Con lo que se obtiene:

```
public class MyFirstService extends Service {



   @Override

   public IBinder onBind(Intent intent) {
```

```
        return null;
    }

    @Override
    public void onCreate() {
        super.onCreate();
    }

}
```

Sin olvidar añadir la declaración del servicio en el manifiesto de la aplicación.

```
<application android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

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

</application>
```

La ejecución de un servicio (método **startService**) corresponde a la llamada al método **onStartCommand**. Este método sirve para ejecutar el tratamiento que debe realizar su servicio.

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {

    return super.onStartCommand(intent, flags, startId);
}
```

Este método recibe tres parámetros:

- El primer parámetro representa una instancia de la clase **Intent**, puede tener valores distintos según el valor de retorno del método **onStartCommand** (**START_STICKY/START_NOT_STICKY/START_REDELIVER_INTENT**).
- El segundo parámetro permite saber las condiciones de ejecución del servicio (0, **START_FLAG_REDELIVERY**,**START_FLAG_RETRY**).
- El tercer parámetro especifica la acción que se deberá ejecutar.

El método **onStartCommand** devuelve un valor entero que sirve para especificar el comportamiento de su servicio. Este entero puede tener los siguientes valores:

- **START_STICKY**: significa que, si el sistema mata el servicio, automáticamente se reiniciará si hay recursos disponibles. Se llamará automáticamente al método **onStartCommand** (no se conservará el estado del servicio).
- **START_NOT_STICKY**: significa que, si el sistema mata al servicio, no se reiniciará automáticamente aunque haya recursos disponibles.
- **START_REDELIVER_INTENT**: significa que, si el sistema mata al servicio, se reiniciará automáticamente si hay recursos disponibles que reciben como parámetro el estado (**intent**) anterior del servicio. Esto será así hasta que el servicio llame al método **stopSelf**.

Para **ejecutar** un servicio, puede utilizar el método **startService**:

```
Intent serviceIntent = new Intent(this, MyFirstService.class);

startService(serviceIntent);
```

Si el método **startService** se invoca en el UI Thread, la ejecución del servicio se realizará en el UI Thread. Deberá crear un nuevo **Thread** o una **AsyncTask** antes de comenzar a ejecutar el tratamiento que realizará su servicio. Por ejemplo:

```
@Override

public int onStartCommand(Intent intent, int flags, int startId)

{

    new Thread(new Runnable() {
```

```
        @Override
        public void run() {
            doServiceWork();
        }
    }).run();
    return START_STICKY;
}
```

Hay dos modos de detener un servicio:

- **Desde el exterior del servicio**: un componente Android puede detener un servicio mediante el método **stopService**.

```
Intent serviceIntent = new Intent(this, MyFirstService.class);
stopService(serviceIntent);
```

- **Desde el interior del servicio**: un servicio puede poner fin a su ejecución mediante el método **stopSelf**.

Puede combinar un servicio con un Broadcast Receiver (**BOOT_COMPLETED**) para iniciar automáticamente el servicio tras el arranque del dispositivo.

2. Asociar un servicio a una actividad

Un servicio puede necesitar actualizar una actividad. Por ejemplo, la reproducción de música puede actualizar el nombre del título reproducido cuando la aplicación está en primer plano.

Esta asociación se puede realizar gracias al método **onBind** implementado en la sobrecarga de la clase **Service**.

Debe comenzar creando una clase que herede de **Binder**. Esta clase servirá para asociar una actividad a un servicio.

```
public class MyActivityBinder extends Binder {
   MyFirstService getMyService() {
```

```
      return MyFirstService.this;

   }

}
```

Después, cree una instancia de la clase **MyActivityBinder**.

```
private IBinder myBinder = new MyActivityBinder();
```

Ahora, modifique el método **onBind** para devolver la instancia que acaba de crear.

```
@Override

public IBinder onBind(Intent intent) {

    return myBinder;

}
```

Para finalizar, cree una variable de tipo **ServiceConnection** en su actividad. Con ello se podrá gestionar la conexión(**onServiceConnected**) y la desconexión (**onServiceDisconnected**) de la actividad con el servicio.

```
private MyFirstService myService;



private ServiceConnection myServiceConnection = new

ServiceConnection() {



    @Override

    public void onServiceDisconnected(ComponentName name) {

       myService = null;

    }



    @Override

    public void onServiceConnected(ComponentName name, IBinder
```

```
service) {

       myService = ((MyFirstService.MyActivityBinder)

service).getMyService();

   }

};
```

El último paso consiste en asociar y ejecutar el servicio a partir de la actividad.

```
Intent bindIntent = new Intent(this, MyFirstService.class);

bindService(bindIntent, myServiceConnection,

Context.BIND_AUTO_CREATE);
```

Una vez que el servicio se ha asociado a la actividad, todos los métodos y todos los atributos públicos del servicio serán accesibles desde la actividad.

Puede utilizar la clase **IntentService** para crear un servicio que pueda reaccionar a eventos asíncronos.