

Services

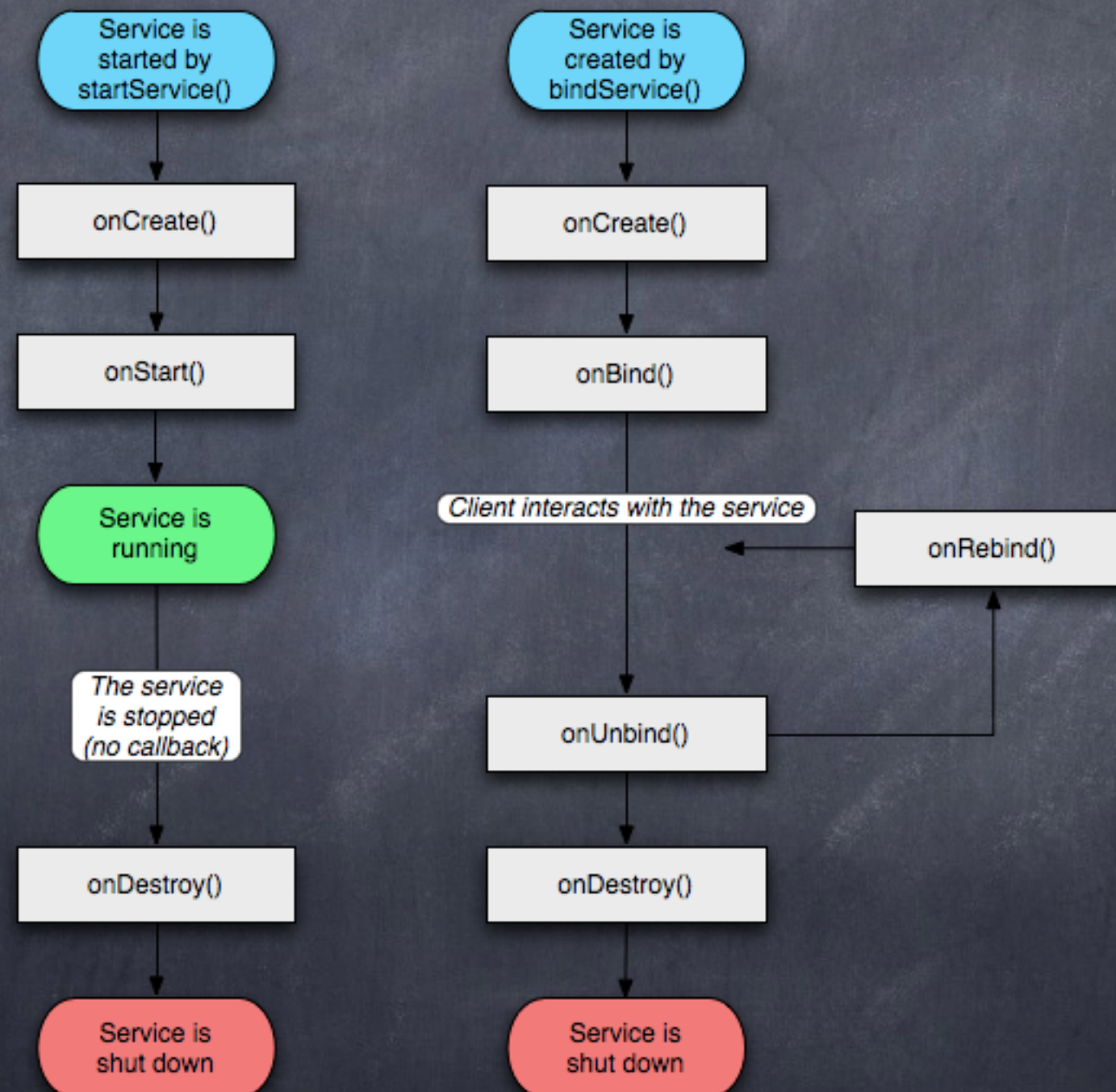
¿Qué es?

- Son un componente principal de Android
- Representa el deseo de una aplicación para realizar una operación de larga duración en background.
- La posibilidad de que una aplicación pueda exponer su funcionalidad a otras aplicaciones.

¿Qué no es?

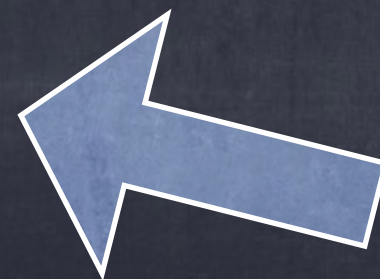
- No es un proceso separado. Corre en el mismo proceso que la aplicación de la cual es parte.
- No es un hilo aparte, por lo que corre en el hilo principal.

Ciclo de vida



A tomar en cuenta

- Cuando el servicio se encuentre en ejecución, es considerado menos importante que cualquier proceso que se encuentre visible en pantalla, pero más importante que cualquier procesos que sea no visible.
- Se puede utilizar el `startForeground(int, Notification)` para disminuir las probabilidades que sea finalizado.



No malutilizar

Dos maneras de uso

- Simple proceso que corre en background
- Conectarse al proceso y enviarle mensajes

Proceso simple

Service

```
public class NormalService extends Service {
    Thread hilo;
    boolean parar = false;
    @Override
    public void onCreate() {
        super.onCreate();

        Log.i("sesion6", "Se creo el servicio");
    }

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        Log.i("sesion6", "Se inicio el servicio");

        hilo = new Thread(){
            public void run(){
                int cont = 0;
                while(!parar){

                    Log.i("sesion6", "Contador: "+ cont++);
                    try {
                        Thread.sleep(5000);
                    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    }
                }
            }
        };
        hilo.start();

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

    @Override
    public void onDestroy() {
        parar = true;
        Log.i("sesion6", "Se finalizo el servicio");
    }
}
```


Declararlo en el Manifest

```
<application
    android:allowBackup="true"
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
    <activity>
    <service android:name=".NormalService" />
</application>
```

Cliente

```
@Override
    public void onClick(View v) {
        switch(v.getId()){
            case R.id.butIniciar:
                intent = new Intent(MainActivity.this, NormalService.class);
                startService(intent);
                break;
            case R.id.butFinalizar:
                stopService(intent);
                break;
        }
    }
```


Conectarse (bind) al
service

El Service

```
public class LocalService extends Service {
    private NotificationManager mNM;
    private int NOTIFICATION = 1000;

    private final IBinder mBinder = new LocalBinder();

    public IBinder onBind(Intent intent) {
        return mBinder;
    }
    @Override
    public void onCreate() {
        mNM = (NotificationManager) getSystemService(NOTIFICATION_SERVICE);

        mostrarNotificacion();
    }

    @Override
    public void onDestroy() {
        // Cancela la notificacion
        mNM.cancel(NOTIFICATION);

        Toast.makeText(this, "Se paro el servicio", Toast.LENGTH_SHORT).show();
    }

    private void mostrarNotificacion(){
        CharSequence text = "Ejemplo";
        // Seteamos la notificacion
        Notification notification = new Notification(android.R.drawable.ic_notification_overlay, text,
            System.currentTimeMillis());
        // PendingIntent que lanzara un activity cuando se haga click en la notificacion
        PendingIntent nextIntent = PendingIntent.getActivity(this, 0,
            new Intent(this, MainActivity.class), 0);
        // Setea la notificacion que saldra en la barra
        notification.setLatestEventInfo(this, "Llego una notificacion",
            text, nextIntent);
        // Envia la notificacion
        mNM.notify(NOTIFICATION, notification);
    }
    public class LocalBinder extends Binder {
        LocalService getService() {
            return LocalService.this;
        }
    }
}
```



```

public class ConectarseActivity extends Activity implements OnClickListener {
    Button butConectarse;
    Button butDesconectarse;
    private boolean isBound = false;
    private LocalService localService;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.conectarse);
        butConectarse = (Button) findViewById(R.id.butConectarse);
        butDesconectarse = (Button) findViewById(R.id.butDesconectarse);
        butConectarse.setOnClickListener(this);
        butDesconectarse.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        switch (v.getId()){
            case R.id.butConectarse:
                doBindService();
                break;
            case R.id.butDesconectarse:
                doUnBindService();
                break;
        }
    }
    private ServiceConnection mConnection = new ServiceConnection() {
        @Override
        public void onServiceDisconnected(ComponentName name) {
            localService = null;
            Toast.makeText(ConectarseActivity.this, "Servicio desconectado", Toast.LENGTH_LONG).show();
        }
        @Override
        public void onServiceConnected(ComponentName name, IBinder service) {
            localService = ((LocalService.LocalBinder)service).getService();
            Toast.makeText(ConectarseActivity.this, "Servicio conectado", Toast.LENGTH_LONG).show();
        }
    };
    private void doBindService(){
        bindService(new Intent(ConectarseActivity.this, LocalService.class), mConnection, Context.BIND_AUTO_CREATE);
        isBound = true;
    }
    private void doUnBindService(){
        if (isBound){
            unbindService(mConnection);
            isBound = false;
        }
    }
}

```