Android Essentials

Servicios y Receivers

Servicios

FOLDER IT

Un `Service` es un componente de una aplicación que puede realizar operaciones de larga ejecución en segundo plano y que no proporciona una interfaz de usuario.

Otro componente de la aplicación puede iniciar un servicio y continuará ejecutándose en segundo plano aunque el usuario cambie a otra aplicación.

Además, un componente puede enlazarse con un servicio para interactuar con él e incluso realizar una comunicación entre procesos (IPC).

Por ejemplo, un servicio puede manejar transacciones de red, reproducir música, realizar I/O de archivos o interactuar con un proveedor de contenido, todo en segundo plano.

Servicios

FOLDER IT

Son similares a los servicios de windows o los demonios de Unix, es decir procesos que ejecutan en un segundo plano y que pueden comunicarse con nuestras aplicaciones.

Los servicios de android siempre pueden estar disponibles, pero no tiene que estar activamente haciendo algo. Tienen un ciclo de vida separado de las actividades a las que le dan soporte.

Existen servicios locales que sólo son accesibles para la aplicación que lo contiene y no son accesibles para otras aplicaciones aunque ejecuten en el mismo dispositivo. Generalmente se utilizan para realizar operaciones de soporte sobre la actividad principal.

Y servicios remotos, que se definen para clientes externos ejecutando en el mismo dispositivo empleando AIDL (Android Interface Definition Language).

Servicio iniciado

Un servicio es "iniciado" (Started) cuando un componente de aplicación (como una actividad) lo inicia llamando a `startService()`.

Una vez iniciado, un servicio puede ejecutarse en segundo plano de manera indefinida, incluso si se destruye el componente que lo inició.

Por lo general, un servicio iniciado realiza una sola operación y no devuelve un resultado al emisor. Por ejemplo, puede descargar o cargar un archivo a través de la red. Cuando la operación está terminada, el servicio debe detenerse por sí mismo.

Servicio enlazado

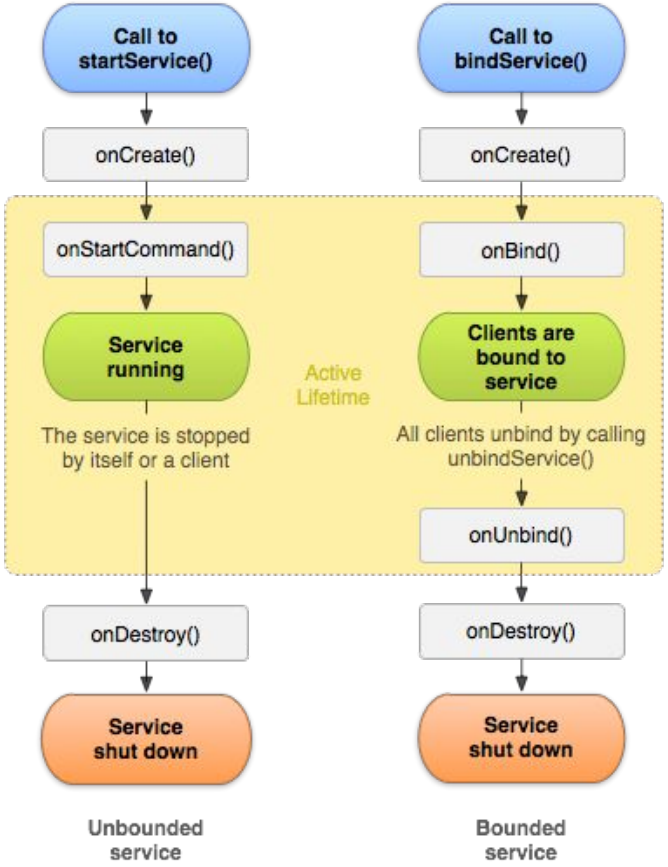
Un servicio es “enlazado” (Bound) cuando un componente de la aplicación se vincula a él llamando a `bindService()`

Un servicio de enlace ofrece una interfaz cliente-servidor que permite que los componentes interactúen con el servicio, envíen solicitudes, obtengan resultados e incluso lo hagan en distintos procesos con la comunicación entre procesos (IPC).

Un servicio de enlace se ejecuta solamente mientras otro componente de aplicación está enlazado con él. Se pueden enlazar varios componentes con el servicio a la vez, pero cuando todos ellos se desenlazan, el servicio se destruye.

Ciclo de vida de los servicios

FOLDER IT



Service vs IntentService

FOLDER IT

Cuando usarlos?

- Un `Service` puede ser usado como una tarea sin UI, pero no debe ser larga (10 seg máximo). Para tareas largas necesitamos hilos
- Un `IntentService` puede ser usado para tareas largas, sin comunicación con el hilo principal. Para comunicarse podemos usar `Handler's`

Cómo iniciarlos?

- Un `Service` es iniciado con el método `startService()`
- Un `IntentService` es iniciado con un `Intent`, crea un nuevo hilo de trabajo y se llama a `onHandleIntent()` en el hilo

Iniciado desde

- Ambos pueden ser iniciados desde cualquier hilo, `Activity` u otro componente

Service vs IntentService

FOLDER IT

Donde ejecutan?

- Un `Service` corre en background pero en el hilo principal
- Un `IntentService` corre en su propio hilo

Limitaciones

- Un `Service` puede bloquear el Hilo principal
- Un `IntentService` no puede ejecutar tareas en paralelo. Todos los `Intent` van a parar al mismo hilo secundario, se encolan y se ejecutan secuencialmente

Cuando terminan?

- En un `Service` es tu responsabilidad pararlo cuando finalice, llamando a `stopSelf()` o `stopService()`
- Un `IntentService` finaliza cuando todas las peticiones de `startService(intent)` han sido procesadas, no hay necesidad de llamar a `stopSelf()`

Broadcast Receiver

FOLDER IT

Un `BroadcastReceiver` es un componente de Android que nos permite registrarnos a eventos de aplicaciones o del sistema

Todos los Receivers registrados a un evento son notificados por Android cada vez que el evento ocurre

Puede ser registrado en el Manifest al igual que los componentes ya vistos, o puede ser registrado estáticamente con el método `Context.registerReceiver()`

Luego del `onReceive()` , el receiver termina, Android tiene permitido reciclarlo

Alarm Manager

FOLDER IT

Es un servicio del sistema que sirve para lanzar eventos

Estos eventos pueden ser lanzados en un momento específico o periódicamente

```
public void setAlarm() {
   AlarmManager manager = (AlarmManager) getSystemService(Context.ALARM_SERVICE);
   int interval = 8000;
   manager.setInexactRepeating(AlarmManager.RTC_WAKEUP, System.currentTimeMillis(),
           interval, pendingIntent);
   Toast.makeText(this, "Alarm Set", Toast.LENGTH_SHORT).show();
}
```

Receivers y Servicios de Larga Duración

FOLDER IT

Tanto los servicios como los receivers ejecutan en el hilo principal.

El tiempo máximo que se puede utilizar para procesamiento es 10 segundos

Si usamos hilos, Android asumirá que el hilo principal finalizó y termina el receiver

Necesitamos una forma de ejecutar en un hilo separado para evitar mensajes de ANR

Necesitamos iniciar un "worker thread", un hilo de trabajo secundario

Mientras dure este hilo en ejecución será necesario adquirir un `WakeLock` para que el dispositivo no vaya a modo "SLEEP"

WakeLocks

FOLDER IT

Un `WakeLock` es una forma de mantener a la CPU "Despierta"

Cualquier aplicación que utilice un WakeLock, debe requerir el permiso

```
<uses-permission android:name="android.permission.WAKE_LOCK" />
```

Lo obtenemos de esta forma:

```
PowerManager powerManager = (PowerManager) getSystemService(POWER_SERVICE);
WakeLock wakeLock = powerManager.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK, "MyWakelockTag");
wakeLock.acquire();
```

El `PARTIAL_WAKE_LOCK` nos asegura que la CPU queda ejecutando, y que la pantalla y la luz del teclado (en caso de tenerlo) pueden apagarse.

WakeLocks en el BroadcastReceiver

El `WakeLock` debe ser obtenido en la parte principal del código del receptor del broadcast

Hacerlo en el servicio no sería de utilidad dado que puede pasar mucho tiempo desde que se invoque `startService()` que puede ser iniciado por el `BroadcastReceiver`, y `onStartCommand()` de un servicio que inicia la ejecución

Pero como estamos proponiendo que se cree un servicio, el cual puede ser detenido cuando sea necesario, si esto sucede es necesario adquirir un `WakeLock` nuevamente

Cuando el hilo de trabajo termina necesita detener el servicio

BroadcastReceiver de larga duración

FOLDER IT

1. Obtener un `WakeLock` parcial en el método `onReceive()` de un `BroadcastReceiver`
    - Este `WakeLock` tiene que ser `static` para permitir el uso compartido entre el `BroadcastReceiver` y el `Service`
    - No hay otra forma de pasar una referencia a una variable `WakeLock`
2. Iniciar un servicio local, para que el proceso no sea “matado”
3. En el servicio iniciar un hilo de trabajo secundario que realizará las tareas de larga duración.
    - No ejecutar tareas de larga duración en `onStart()` dado que estaremos usando el hilo principal
4. Cuando el hilo de trabajo finalice, hacer que el servicio finalice
5. El servicio deberá encargarse de apagar el `WakeLock`