DESARROLLO DE APLICACIONES EN ANDROID

SAMSUNG

Abraham Gutiérrez Rodríguez

Samsung TECH INSTITUTE

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Un servicio es un componente de aplicación que puede realizar operaciones de larga ejecución en segundo plano y no proporciona una interfaz de usuario.
  - Cualquier componente de una aplicación puede iniciar un servicio y continuará funcionando en segundo plano, incluso si el usuario cambia a otra aplicación.
  - Además, cualquier componente puede unirse a un servicio para interactuar con él e incluso haciendo uso de comunicación entre procesos (IPC).
    - Un servicio puede declararse como privado en el archivo de manifiesto, y bloquear el acceso desde otras aplicaciones.
  - Por ejemplo, un servicio puede manejar las transacciones de red, reproducir música, realizar operaciones con archivos de E/S, o interactuar con un proveedor de contenido, todo desde un segundo plano.

Samsung TECH INSTITUTE

Android: Conceptos Básicos
Servicios

- Un servicio esencialmente puede adoptar dos formas:
  - **Iniciado**. Un servicio es "iniciado" cuando un componente de una aplicación (por ejemplo, una actividad) lo inicia llamando al método startService(). Una vez iniciado, un servicio puede ejecutarse en segundo plano de forma indefinida, incluso si el componente que se inició, se destruye.
    - Por lo general, un servicio iniciado realiza una sola operación y no devuelve un resultado al proceso que lo llamó. Por ejemplo, puede descargar o cargar un archivo en la red. Cuando se realiza la operación, el servicio debería pararse.
  - **Enlazado**. Un servicio es "enlazado" cuando un componente de la aplicación se une a él llamando al método bindService(). Un servicio de enlace ofrece una interfaz cliente-servidor que permite a los componentes para interactuar con el servicio, enviar solicitudes, obtener resultados, e incluso lo hacen a través de comunicación entre procesos (IPC).
    - Un servicio de enlace sólo se ejecuta mientras que un componente de una aplicación está enlazado con él. Múltiples componentes pueden enlazarse con un servicio a la vez, cuando todos ellos se desenlazan se destruye el servicio.

Samsung TECH INSTITUTE

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Un servicio se ejecuta en el hilo principal del proceso que lo contiene. Un servicio no crea su propio hilo y no se ejecuta en un proceso separado (a menos que se especifique lo contrario). Esto significa que, si el servicio va a hacer algún trabajo intensivo de la CPU u operaciones bloqueantes (como la reproducción de MP3 o de redes), se debe crear un nuevo hilo dentro del servicio para hacer ese trabajo.
  - Mediante el uso de un hilo separado, se reducirá el riesgo de que la aplicación no responda (errores ANR) y el hilo principal de la aplicación podrá seguir dedicado a la interacción del usuario con sus actividades.
  - La plataforma Android ofrece una gran cantidad de servicios predefinidos, disponibles regularmente a través de la clase *Manager*. De esta manera, en nuestras actividades podremos acceder a estos servicios a través del método getSystemService().

Android: Conceptos Básicos
Servicios

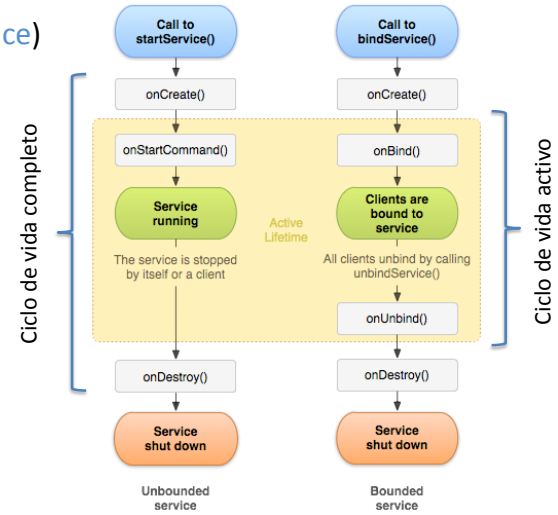
- Servicios (Service)
  - Para crear un servicio, se debe crear una subclase de Service (o de una de sus subclases existentes). En la implementación, se deben redefinir algunos métodos de devolución de llamada que se encargan de los aspectos clave del ciclo de vida del servicio y proporcionan los mecanismos para que los componentes se unan al servicio, si corresponde.
  - Los métodos de devolución de llamada más importantes que debe reemplazar son:
    - onStartCommand(). El sistema llama a este método cuando otro componente, tal como una actividad, solicita que se inicie el servicio, llamando startService(). Una vez que este método se ejecuta, se inicia el servicio y puede ejecutarse en segundo plano de forma indefinida. Es responsabilidad del código detener el servicio cuando se lleve a cabo su labor, llamando a los métodos stopSelf() o stopService(). Si sólo se desea proporcionar el enlace con el servicio no es necesario reescribir este método.

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Los métodos de devolución de llamada más importantes que debe reemplazar son:
    - onBind(). El sistema llama a este método cuando otro componente quiere enlazar con el servicio (por ejemplo, para realizar RPC), llamando al método bindService(). En la implementación de este método, se debe proporcionar una interfaz que los clientes utilizan para comunicarse con el servicio, devolviendo una IBinder. Siempre hay que implementar este método, pero si no se desea permitir el enlace con el servicio se *null*.
    - onCreate(). El sistema llama a este método cuando se crea el servicio por primera vez, para llevar a cabo los procedimientos de configuración una sola vez (antes de llamar a onStartCommand() o onBind()). Si el servicio ya está en funcionamiento este método no se llama.
    - onDestroy(). El sistema llama a este método cuando ya no se usa el servicio y se está destruyendo. Se debe implementar este método para limpiar los recursos, tales como hilos, escuchadores registrados, receptores, etc Esta es la última llamada que el servicio recibe.

Android: Conceptos Básicos
Servicios

- Servicios (Service)



Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Cuando se inicia un servicio para realizar alguna tarea en segundo plano, el proceso donde se ejecuta podría ser eliminado ante una situación de baja memoria. Podemos configurar la forma en que el sistema reaccionará ante esta circunstancia según el valor que devolvamos en onStartCommand().
  - Existen varios modos principales:
    - Devolveremos START_STICKY si queremos que el sistema trate de crear de nuevo el servicio cuando disponga de memoria suficiente.
    - Devolveremos START_NOT_STICKY si queremos que el servicio sea creado de nuevo solo cuando llegue una nueva solicitud de creación.
    - Devolveremos START_REDELIVER_INTENT si queremos que el servicio sea creado de nuevo solo cuando existen intenciones enviadas que no han sido procesadas. Recrea el servicio con la última intención que fue enviada al servicio.

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Teniendo en cuenta que los servicios pueden estar largo tiempo en ejecución, el ciclo de vida del proceso que contiene nuestro servicio es un asunto de gran importancia. Conviene aclarar que en situaciones donde el sistema necesite memoria conservar un servicio siempre será menos prioritario que la actividad visible en pantalla, aunque más prioritario que otras actividades en segundo plano.
    - Dado que el número de actividades visibles es siempre reducido, un servicio solo será eliminado en situaciones de extrema necesidad de memoria.
    - Por otra parte, si un cliente visible está conectado a un servicio, el servicio también será considerado como visible, siendo tan prioritario como el cliente.
    - En el caso de un proceso que contenga varios componentes, por ejemplo una actividad y un servicio, su prioridad se obtiene como el máximo de sus componentes.

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Al igual que las actividades (y otros componentes), los servicios se deben declarar en el archivo de manifiesto de la aplicación. Para ello se utiliza el elemento <service> como un elemento hijo del elemento <application>.

```
<manifest ... >
  ...
  <application ... >
      <service android:name=".ExampleService" />
      ...
  </application>
</manifest>
```

  - El atributo *android:name* es el único atributo obligatorio. Hay otros atributos que se pueden incluir en el elemento <service> para definir propiedades tales como permisos requeridos para iniciar el servicio y el proceso en el que debe ejecutarse el servicio.

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Para asegurar que la aplicación es segura, hay que utilizar siempre una intención explícita cuando se inicia o se enlaza con el servicio y no declarar filtros de intención para el servicio.
    - Si es crítico permitir cantidad de ambigüedad en cuanto a qué servicio se inicia, se pueden suministrar filtros de intención en los servicios y no incluir el nombre del componente en la intención, pero en ese caso se debe configurar el paquete para la intención con setPackage(), que proporciona desambiguación suficiente para el servicio de destino.
  - Además, se puede asegurar que un servicio está disponible sólo para una aplicación estableciendo el atributo *android:exported* a "*false*". Esto evita que otras aplicaciones inicien el servicio, incluso cuando se utiliza una intención explícita.

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Declarando un servicio en el archivo de manifiesto simplemente con la etiqueta <service> estamos permitiendo un acceso global al mismo. Podemos definir permisos para restringir su acceso. En este caso, las aplicaciones han de declarar este permiso, con el correspondiente <uses-permission> en su propio manifiesto.
    - Podemos definir permisos para arrancar, parar o conectarse a un servicio. De forma adicional, podemos restringir el acceso a funciones específicas de las ofertadas por un servicio. Para este propósito, podemos llamar al principio de nuestra función a checkCallingPermission() para verificar si el cliente dispone de un permiso en concreto.

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Un servicio iniciado debe ser arrancado por otro componente llamando al método startService(), lo que resulta en una llamada al método del servicio onStartCommand().
    - Cuando se inicia un servicio, tiene un ciclo de vida que es independiente del componente que lo arrancó y el servicio puede ejecutarse en segundo plano de forma indefinida, incluso si el componente que lo inició, se destruye. Como tal, el servicio debe pararse cuando su trabajo se hace llamando stopSelf(), u otro componente puede detenerlo llamando stopService().
    - El componente de la aplicación que inicia el servicio llamando a startService() proporciona a este método como parámetro una intención que especifica el servicio a iniciar, e incluye todos los datos necesarios para que el servicio realice su función. El servicio recibe esta intención en el método onStartCommand().

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Tradicionalmente, hay dos clases que se pueden extender para crear un servicio de tipo *iniciado*:
    - Service. Esta es la clase base para todos los servicios. Al ampliar esta clase, es importante que se cree un nuevo hilo en el que hacer todo el trabajo del servicio, debido a que el servicio utiliza el subproceso principal de la aplicación, de forma predeterminada, lo que podría disminuir el rendimiento de cualquier actividad se ejecuta la aplicación.
    - IntentService. Esta es una subclase de servicio que utiliza un hilo independiente de trabajo para manejar todas las peticiones de inicio una a una. Esta es la mejor opción si no se requiere que el servicio maneje múltiples peticiones simultáneamente. Sólo hay que implementar el método onHandleIntent(), que recibe el intento de cada solicitud de puesta en marcha para que pueda hacer el trabajo en segundo plano.

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Al crear un servicio de enlace se debe proporcionar un IBinder que proporciona la interfaz de programación que los clientes pueden utilizar para interactuar con el servicio.
  - Hay tres formas en que puede definir esta interfaz:
    - **Extendiendo la clase Binder**. Si el servicio es privado para la propia aplicación y se ejecuta en el mismo proceso que el cliente (que es lo más común), se debe crear la interfaz mediante la extensión de la clase Binder y devolver una instancia de ella desde onBind(). El cliente recibe el Binder y puede utilizarlo para acceder directamente a los métodos públicos disponibles, ya sea en el Binder o incluso en el servicio.
      Esta es la técnica a utilizar cuando el servicio no es más que un trabajador en segundo plano para la aplicación. La única razón por la que no debería crear la interfaz de esta manera es porque el servicio sea utilizado por otras aplicaciones o a través de procesos separados.

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Hay tres formas en que puede definir esta interfaz:
    - **El uso de un mensajero (Messenger).** Si necesita que la interfaz pueda trabajar a través de diferentes procesos, se debe crear la interfaz para el servicio con un mensajero. De esta manera, el servicio define un controlador que responde a diferentes tipos de mensajes. Este controlador es la base para un mensajero que luego puede compartir un IBinder con el cliente, lo que permite que el cliente envíe comandos al servicio a través de los objetos de mensaje. Además, el cliente puede definir su propio mensajero por lo que el servicio puede enviar mensajes de respuesta.
      Esta es la forma más sencilla de realizar la comunicación entre procesos (IPC), porque el mensajero encola todas las solicitudes en un único hilo de modo que no hay que diseñar el servicio para que sea seguro para subprocesos (*thread-safe*).

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Hay tres formas en que puede definir esta interfaz:
    - **Usando AIDL (Android Interface Definition Language)**. AIDL realiza todo el trabajo para descomponer objetos en primitivas que el sistema operativo pueda entender y enviar a través de procesos para realizar IPC. La técnica anterior, el uso de un mensajero, se basa realmente en AIDL como su estructura subyacente. Como se mencionó anteriormente, el mensajero crea una cola con todas las solicitudes de cliente en un solo hilo, por lo que el servicio recibe las peticiones de una en una. Sin embargo, si se quiere que el servicio pueda manejar múltiples peticiones a la vez es necesario utilizar directamente AIDL. En este caso, el servicio debe ser capaz de realizar multitarea y se deberá construir seguro para subprocesos .

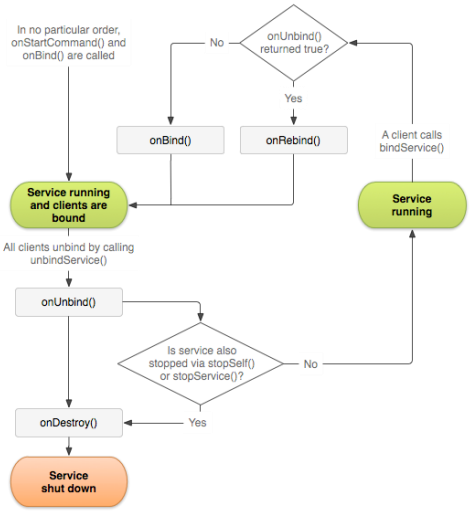
      Para utilizar AIDL directamente, se debe crear un archivo *.aidl* que define la interfaz de programación. Las herramientas del SDK de Android utilizan este archivo para generar una clase abstracta que implementa la interfaz y se ocupa de manejar la IPC, que luego se puede extender dentro del servicio.

Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Es posible crear un servicio en modo *iniciado*, el servicio se arranca con starService(), y permitir que un cliente se enlace con el servicio llamando a bindService().
  - Si se permite que un servicio sea iniciado o enlazado hay que tener en cuenta que el sistema no destruye el servicio cuando todos los clientes se desvinculan, sino que debe detener de forma explícita llamando a stopSelf() o stopService().
  - A pesar de que normalmente sólo implementaremos onBind() o onStartCommand(), a veces es necesario implementar ambos.
    - Por ejemplo, a un reproductor de música le puede resultar útil que su servicio se ejecute indefinidamente y también proporcionan el enlace con el mismo. De esta manera, una actividad puede iniciar el servicio para escuchar música, que continuará reproduciéndose incluso si el usuario sale de la aplicación. Y cuando el usuario vuelva a la aplicación, la actividad puede enlazar con el servicio para recuperar el control de la reproducción.

Android: Conceptos Básicos
Servicios

- Servicios (Service)



Android: Conceptos Básicos
Servicios

- Servicios (Service)
  - Un servicio en primer plano es un servicio que se considera debe hacer algo que el usuario necesita siempre activo, por lo que no será un candidato para ser destruido por el sistema cuando haya problemas de memoria.
  - Debe proporcionar una notificación en la barra de estado, que se coloca bajo la cabecera "permanente" (*Ongoing*), lo que significa que la notificación no puede ser eliminada a menos que el servicio sea detenido o eliminado.
    - Por ejemplo, un reproductor de música que reproduce música de un servicio puede ser configurado para ejecutarse en primer plano, ya que el usuario es explícitamente consciente de su funcionamiento. La notificación en la barra de estado puede indicar la canción actual y permitir al usuario iniciar una actividad para interactuar con el reproductor de música.
  - Para solicitar que un servicio se ejecute en el primer plano, se llama a startForeground(). Este método toma dos parámetros: un entero que identifica unívocamente la notificación y la notificación de la barra de estado. Para eliminar un servicio en primer plano hay que llamar a stopForeground().

Android: Conceptos Básicos
Componentes básicos

- Receptor de difusión (Broadcast Receiver):
  - Se utilizan para lanzar alguna ejecución dentro de la aplicación actual cuando un determinado evento se produzca (generalmente abrir un componente, una actividad o un servicio).
  - Este componente no tiene interfaz de usuario asociada.
    - Puede utilizar el API Notification Manager para avisar al usuario del evento producido a través de la barra de estado del dispositivo móvil.
  - Son el modo de responder a notificaciones externas o alarmas y de responder las intenciones enviadas.
  - Se implementan como una subclase de BroadcastReceiver y cada difusión se realiza como un objeto Intent.
  - Más info:
    http://developer.android.com/reference/android/content/BroadcastReceiver.html

Android: Conceptos Básicos
Componentes básicos

- Receptor de difusión (Broadcast Receiver):
  - Por último, mencionar que hay dos tipos de broadcasts:
  - Broadcasts normales: Son completamente asíncronos, de forma que todos los receptores corren en un orden desconocido, incluso al mismo tiempo. Es más eficiente, pero perdemos control en el orden de las acciones a realizar.
  - Broadcasts ordenados: En este caso, el mensaje se envía a un sólo receptor a la vez. Cada receptor tendrá un turno en el cual recibirá el mensaje, y propagará el resultado al siguiente receptor o incluso cancelar el mensaje. Podremos controlar el orden con prioridades, pero perderemos eficiencia.

Android: Conceptos Básicos
Servicios

- Receptores de difusión (BroadcastReceiver)
  - Un receptor de difusión es un componente Android que permite el registro de eventos del sistema. Todos los receptores registrados para un evento serán notificados por Android una vez que éstos ocurran.
    - Por ejemplo, Android permite que aplicaciones puedan registrarse al ACTION_BOOT_COMPLETED que es un evento que lanza el sistema una vez que ha completado el proceso de arranque.
  - Un receptor de difusión extiende de la clase BroadcastReceiver y es registrado como un receptor, <receiver>, en una aplicación a través del fichero de manifiesto.
  - Alternativamente a este tipo de registro, podemos registrar un receptor de difusión dinámicamente a través del método Context.registerReceiver().

Android: Conceptos Básicos
Servicios

- Receptores de difusión (BroadcastReceiver)
  - Si el evento para el que el receptor de difusión se ha registrado ocurre, el sistema Android llama al método onReceive() y le pasa un Intent para su tratamiento.
  - Después de que finalice la ejecución del método onReceive(), el sistema considerará que el receptor de difusión ya no está activo, por lo que el sistema podrá reciclarlo si lo considera necesario.
    - A partir del nivel 11 de API no es posible realizar tareas asíncronas en el método onReceive(). Si tenemos operaciones potencialmente largas deberíamos lanzar un servicio.
    - Desde API Android 11 se puede llamar al método goAsync(). Este método devuelve un objeto del tipo PendingResult. El sistema Android considera el receptor con vida hasta que se llama a PendingResult.finish() sobre el objeto. Con esta opción se puede activar el procesamiento asincrónico en un receptor. Tan pronto como el hilo ha terminado, su tarea llama a finish() para indicar al sistema de Android que este componente puede ser reciclado.

Samsung TECH INSTITUTE

Android: Conceptos Básicos
Servicios

- Receptores de difusión (BroadcastReceiver)
  - Hay dos clases principales de emisiones que se pueden recibir:
    - **Emisiones normales** (enviadas con Context.sendBroadcast) que son completamente asíncronas. Todos los receptores de la emisión se ejecutan en un orden definido, a menudo al mismo tiempo . Esto es más eficiente, pero significa que los receptores no pueden utilizar el resultado o abortar APIs incluidas aquí .
    - **Emisiones ordenadas** (enviadas con Context.sendOrderedBroadcast) que se entregan a un solo receptor a la vez. Como cada receptor se ejecuta en un turno, se puede propagar un resultado al siguiente receptor, o se puede abortar completamente la emisión de manera que no se puede pasar a otros receptores. El orden de ejecución de los receptores se puede controlar con el atributo *android:priority* del filtro de intenciones correspondiente; los receptores con la misma prioridad se ejecutan en un orden arbitrario.

Samsung TECH INSTITUTE

Android: Conceptos Básicos
Servicios

- Receptores de difusión (BroadcastReceiver)
  - Incluso en el caso de emisiones normales, el sistema puede en algunas situaciones revertir la entrega de la emisión de un receptor a la vez. En particular, para los receptores que requieran la creación de un proceso sólo se puede ejecutar de uno en uno para evitar sobrecargar el sistema con nuevos procesos.
    - En esta situación, sin embargo, la semántica no-ordenada se sostiene: estos receptores no pueden devolver resultados o abortar la difusión.
  - El método sendBroadcast() no permite desencadenar emisiones del sistema, el sistema Android evita esto. Pero sí podemos definir filtros de intención para nuestras propias acciones y emitirlas con el método sendBroadcast().

Android: Conceptos Básicos
Servicios

- Receptores de difusión (BroadcastReceiver)
  - A partir de la versión Android 3.1 el sistema Android por defecto excluye todo receptor de difusión de recibir intenciones si la correspondiente aplicación nunca ha sido iniciada por el usuario o si el usuario explícitamente detuvo la aplicación desde el menú Android.
  - Esta es una característica de seguridad adicional ya que el usuario puede estar seguro de que sólo las aplicaciones que él empezó recibirá intenciones de difusión.
  - Los permisos de acceso pueden ser impuestos por el remitente o el receptor de una emisión. Para hacer cumplir un permiso al enviar, se debe proporcionar un argumento permiso no nulo, de modo que sólo los receptores a los que se ha concedido este permiso (mediante la etiqueta <uses-permission> en su manifiesto) podrán recibir la emisión.

Android: Conceptos Básicos

Servicios

- Receptores de difusión (BroadcastReceiver)
  - Hay que tener en cuenta que, aunque la clase Intent se utiliza para enviar y recibir estas emisiones, el mecanismo de difusión de la intención aquí es totalmente independiente de las intenciones que se utilizan para iniciar actividades con Context.startActivity().
  - No hay manera de que un receptor de emisiones vea o capture las intenciones utilizadas con startActivity(); del mismo modo, cuando se emite una intención, nunca se deberá iniciar una actividad.
  - Estas dos operaciones son semánticamente muy diferentes: iniciar una actividad con una Intención es una operación de primer plano que modifica el componente con el que el usuario está interactuando; difundir/emitir un intención es una operación en segundo plano de la que el usuario normalmente no es consciente.

Samsung TECH INSTITUTE