**Servicios**

**Vista rápida**

* + Un servicio puede ejecutarse en segundo plano para realizar el trabajo, incluso mientras el usuario está en una aplicación diferente
  + Un servicio puede permitir otros componentes que se unen a ella, con el fin de interactuar con él y realizar la comunicación entre
  + Un servicio se ejecuta en el hilo principal de la aplicación que lo alberga, de forma predeterminada

**En este documento**

* + [Los puntos básicos.](http://developer.android.com/guide/topics/fundamentals/services.html#Basics)
    1. [La declaración de un servicio en el manifiesto](http://developer.android.com/guide/topics/fundamentals/services.html#Declaring)
  + [Creación de un servicio iniciado](http://developer.android.com/guide/topics/fundamentals/services.html#CreatingAService)
    1. [Ampliación de la clase IntentService](http://developer.android.com/guide/topics/fundamentals/services.html#ExtendingIntentService)
    2. [Ampliación de la clase de servicio](http://developer.android.com/guide/topics/fundamentals/services.html#ExtendingService)
    3. [A partir de un servicio](http://developer.android.com/guide/topics/fundamentals/services.html#StartingAService)
    4. [Detención de un servicio de](http://developer.android.com/guide/topics/fundamentals/services.html#Stopping)
  + [Creación de un servicio consolidado](http://developer.android.com/guide/topics/fundamentals/services.html#CreatingBoundService)
  + [El envío de notificaciones al usuario](http://developer.android.com/guide/topics/fundamentals/services.html#Notifications)
  + [Ejecución de un servicio en el primer plano](http://developer.android.com/guide/topics/fundamentals/services.html#Foreground)
  + [Gestión del Ciclo de Vida de un Servicio de](http://developer.android.com/guide/topics/fundamentals/services.html#Lifecycle)
    1. [La aplicación de las devoluciones de llamada de ciclo de vida](http://developer.android.com/guide/topics/fundamentals/services.html#LifecycleCallbacks)

**Clases de clave**

* + [Service](http://developer.android.com/reference/android/app/Service.html)
  + [IntentService](http://developer.android.com/reference/android/app/IntentService.html)

**Las muestras**

* + [ServiceStartArguments](http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/app/ServiceStartArguments.html)
  + [LocalService](http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/app/LocalService.html)

**Artículos.**

* + [Multitarea la Vía Android](http://developer.android.com/resources/articles/multitasking-android-way.html)
  + [Servicio de Cambios en el API de partida con Android 2.0](http://developer.android.com/resources/articles/service-api-changes-starting-with.html)

**Véase también**

* + [Servicios enlazados](http://developer.android.com/guide/topics/fundamentals/bound-services.html)

Un [Service](http://developer.android.com/reference/android/app/Service.html) es un componente de aplicación que puede realizar operaciones de larga duración en el fondo y no proporciona una interfaz de usuario. Otro componente de la aplicación se puede iniciar un servicio y que seguirá funcionando en segundo plano, incluso si el usuario cambia a otra aplicación. Además, un componente puede unirse a un servicio para interactuar con él e incluso realizar la comunicación entre procesos (IPC). Por ejemplo, un servicio podría llevar a cabo transacciones en la red, reproducir música, realizar operaciones con archivos de E / S, o interactuar con un proveedor de contenido, todo desde el fondo.

Un servicio esencial puede adoptar dos formas:

Iniciado

Un servicio está "en marcha" cuando un componente de la aplicación (por ejemplo, una actividad) que se inicia llamando [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) . Una vez iniciado, un servicio puede ejecutar en el fondo de forma indefinida, incluso si el componente que comenzó se destruye. Por lo general, un servicio iniciado realiza una sola operación y no devuelve un resultado a la persona que llama. Por ejemplo, puede descargar o cargar un archivo a través de la red. Cuando la operación se realiza, el servicio debe detenerse.

Obligado

Un servicio está "obligado", cuando un componente de aplicación se une a ella llamando al [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) . Un servicio consolidado ofrece una interfaz de cliente-servidor que permite a los componentes de interactuar con el servicio, enviar solicitudes, obtener resultados, e incluso hacerlo a través de procesos con la comunicación entre procesos (IPC). Un servicio consolidado sólo se ejecuta mientras otro componente de la aplicación está obligado a ello. Varios componentes se pueden enlazar con el servicio a la vez, pero cuando todos ellos desatar, el servicio se destruye.

A pesar de esta documentación en general, se analizan estos dos tipos de servicios por separado, su servicio puede funcionar en ambos sentidos, se puede iniciar (para ejecutar de forma indefinida) y también permite la unión. Es simplemente una cuestión de si se implementa un par de métodos de devolución de llamada: [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) para que los componentes para iniciarlo y [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) para permitir la unión.

Sin importar si su aplicación se inicia, obligado, o ambos, cualquiera de los componentes de aplicaciones pueden utilizar el servicio (incluso a partir de una solicitud por separado), de la misma manera que cualquier componente puede utilizar una actividad por partida con una [Intent](http://developer.android.com/reference/android/content/Intent.html) . Sin embargo, se puede declarar el servicio como privado, en el archivo de manifiesto, y bloquear el acceso desde otras aplicaciones. Esto se discute más en la sección acerca de [declarar el servicio en el manifiesto](http://developer.android.com/guide/topics/fundamentals/services.html#Declaring) .

**Precaución:** Un servicio se ejecuta en el hilo principal de su anfitrión proceso del 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 se va a hacer cualquier trabajo intensivo de la CPU o de operaciones de bloqueo (por ejemplo, la reproducción de MP3 o en red), debe crear un nuevo hilo en el servicio para hacer ese trabajo. Mediante el uso de un hilo separado, se reducirá el riesgo de aplicación que no responde (ANR) los errores y el hilo principal de la aplicación puede seguir siendo dedicado a la interacción del usuario con sus actividades.

**Los Fundamentos**

**En caso de utilizar un servicio o un hilo?**

Un servicio es simplemente un componente que puede ejecutarse en segundo plano, incluso cuando el usuario no está interactuando con su solicitud. Por lo tanto, debe crear un servicio sólo si eso es lo que usted necesita.

Si necesita realizar un trabajo fuera de su hilo principal, pero sólo mientras el usuario está interactuando con la aplicación, entonces usted probablemente debería en lugar de crear un hilo nuevo y no un servicio. Por ejemplo, si quieres jugar algo de música, pero sólo mientras la actividad se ejecuta, puede crear un hilo en [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate%28android.os.Bundle%29) , comienza a correr en [onStart()](http://developer.android.com/reference/android/app/Activity.html#onStart%28%29) , entonces se paran en [onStop()](http://developer.android.com/reference/android/app/Activity.html#onStop%28%29) . También considere usar [AsyncTask](http://developer.android.com/reference/android/os/AsyncTask.html) o [HandlerThread](http://developer.android.com/reference/android/os/HandlerThread.html) , en lugar de la tradicional [Thread](http://developer.android.com/reference/java/lang/Thread.html) de clase. Ver los [procesos y subprocesos](http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html#Threads) de documentos para obtener más información acerca de las discusiones.

Recuerde que si usted hace uso de un servicio, que aún se ejecuta en el hilo principal de la aplicación por defecto, por lo que aún debe crear un nuevo hilo en el servicio si se lleva a cabo operaciones intensivas o de bloqueo.

Para crear un servicio, debe crear una subclase de [Service](http://developer.android.com/reference/android/app/Service.html) (o una de sus subclases ya existentes). En su implementación, es necesario reemplazar algunos de los métodos de devolución de llamada que se ocupan de los aspectos clave del ciclo de vida del servicio y proporcionar un mecanismo para los componentes de enlazar con el servicio, en su caso. Los métodos de devolución de llamada más importantes que deben prevalecer son los siguientes:

[onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29)

El sistema llama a este método cuando otro componente, como una actividad, pide que el servicio sea iniciado, llamando al [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) . Una vez que este método se ejecuta, el servicio se inicia y puede ejecutarse en segundo plano de forma indefinida. Si decide implementar esta forma, es su responsabilidad para detener el servicio cuando su trabajo está hecho, llamando [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) o [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) . (Si sólo desea proporcionar vinculante, no es necesario para poner en práctica este método.)

[onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29)

El sistema llama a este método cuando quiere otro componente para enlazar con el servicio (por ejemplo, para llevar a cabo RPC), llamando al [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) . En la implementación de este método, usted debe proporcionar una interfaz que los clientes utilizan para comunicarse con el servicio, mediante la devolución de un [IBinder](http://developer.android.com/reference/android/os/IBinder.html) . Siempre se debe poner en práctica este método, pero si usted no quiere para permitir la unión, entonces usted debe devolver null.

[onCreate()](http://developer.android.com/reference/android/app/Service.html#onCreate%28%29)

El sistema llama a este método cuando el servicio se creó por primera vez, para llevar a cabo una sola vez los procedimientos de configuración (antes de llamar a cualquiera de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) o [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) ). Si el servicio ya se está ejecutando, este método no se conoce.

[onDestroy()](http://developer.android.com/reference/android/app/Service.html#onDestroy%28%29)

El sistema llama a este método cuando el servicio ya no se utiliza y está siendo destruido. Su servicio debe implementar este para limpiar los recursos, tales como las discusiones, los oyentes registrados, receptores, etc Esta es la última llamada el servicio recibe.

Si un componente se inicia el servicio llamando al [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) (que se traduce en una llamada a [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) ), el servicio sigue funcionando hasta que se detiene con [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) u otro componente que se detenga llamando [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) .

Si un componente llama [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) para crear el servicio (y [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) *no* se llama), el servicio sólo se ejecuta siempre y cuando el componente está ligado a ella. Una vez que el servicio está consolidado de todos los clientes, el sistema lo destruye.

El sistema Android obligará a parar un servicio, sólo cuando la memoria es baja y debe recuperar los recursos del sistema para la actividad que tiene el foco del usuario. Si el servicio está destinado a una actividad que tiene centrarse en el usuario, entonces es menos probable que se mató, y si el servicio se declara que [se ejecute en primer plano](http://developer.android.com/guide/topics/fundamentals/services.html#Foreground) (explicado más adelante), entonces casi nunca morir. De lo contrario, si el servicio se ha iniciado y es de larga duración, el sistema reducirá su posición en la lista de tareas en segundo plano en el tiempo y el servicio son altamente susceptibles a la muerte, si su servicio se ha iniciado, a continuación, debe diseñar de controlar correctamente se reinicia por el sistema. Si el sistema mata a su servicio, se reinicia tan pronto como se disponga de recursos una vez más (aunque esto también depende del valor que regrese de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) , tal como se discute más adelante). Para obtener más información acerca de cuando el sistema podría destruir un servicio, vea los [procesos y subprocesos](http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html) de documentos.

En las secciones siguientes, veremos cómo se puede crear cada tipo de servicio y de cómo utilizarlo a partir de componentes de otras aplicaciones.

**La declaración de un servicio en el manifiesto**

Al igual que las actividades (y otros componentes), debe declarar todos los servicios en el archivo de manifiesto de la aplicación.

Para declarar su servicio, agregar un [<service>](http://developer.android.com/guide/topics/manifest/service-element.html) elemento como un niño de la [<application>](http://developer.android.com/guide/topics/manifest/application-element.html) elemento. Por ejemplo:

<Manifiesta ... >

...

<Aplicación ... >

<Servicio de android: name = ".ExampleService "/>

...

</ Application>

</ Manifest>

Hay otros atributos que se pueden incluir en el [<service>](http://developer.android.com/guide/topics/manifest/service-element.html) elemento para definir propiedades tales como permisos necesarios para iniciar el servicio y el proceso en el que debe ejecutarse el servicio. El [android:name](http://developer.android.com/guide/topics/manifest/service-element.html#nm) atributo es el atributo que sólo se requiere especifica el nombre de la clase del servicio. Una vez que se publique su aplicación, no se debe cambiar este nombre, porque si lo hace, podría romper alguna funcionalidad en las intenciones explícitas se utilizan para hacer referencia a su servicio (leer la entrada del blog, [las cosas que no puede cambiar](http://www.google.com/url?q=http%3A%2F%2Fandroid-developers.blogspot.com%2F2011%2F06%2Fthings-that-cannot-change.html&sa=D&sntz=1&usg=AFQjCNFwvQmH1QFSjfQ_MZdv1B--Qbu-Ww) ).

Ver la [<service>](http://developer.android.com/guide/topics/manifest/service-element.html) elemento de referencia para obtener más información sobre la declaración de su servicio en el manifiesto.

Al igual que una actividad, un servicio puede definir los filtros de intención que permiten a los otros componentes para invocar el servicio con intenciones implícitas. Al declarar la intención de filtros, componentes de cualquier aplicación instalada en el dispositivo del usuario potencialmente puede iniciar el servicio si el servicio se declara un filtro de la intención que coincide con la intención de otra aplicación pasa a [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) .

Si usted planea usar su servicio sólo a nivel local (otras aplicaciones no lo uso), entonces no es necesario (ni deben) proporcionar los filtros de la intención. Sin ningún tipo de filtros de la intención, debe iniciar el servicio con la intención de que nombra explícitamente la clase de servicio. Más información acerca de [iniciar un servicio](http://developer.android.com/guide/topics/fundamentals/services.html#StartingAService) que se discuten a continuación.

Además, usted puede asegurarse de que su servicio es privado para su aplicación sólo si incluye el [android:exported](http://developer.android.com/guide/topics/manifest/service-element.html#exported) atributo y establezca a "false" . Esto es efectivo, incluso si su servicio suministra filtros intención.

Para obtener más información sobre cómo crear filtros de intención para su servicio, ver las [intenciones y la intención de filtros](http://developer.android.com/guide/topics/intents/intents-filters.html) de documentos.

**Creación de un servicio iniciado**

**Orientación de Android 1.6 o inferior**

Si usted está construyendo una aplicación para Android 1.6 o inferior, es necesario implementar [onStart()](http://developer.android.com/reference/android/app/Service.html#onStart%28android.content.Intent,%20int%29) , en lugar de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) (en Android 2.0, [onStart()](http://developer.android.com/reference/android/app/Service.html#onStart%28android.content.Intent,%20int%29) fue descartado en favor de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) ).

Para obtener más información sobre cómo proporcionar compatibilidad con las versiones de Android mayor que 2.0, consulte el [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) la documentación.

Un servicio de empezar es uno que otro componente se inicia llamando [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) , lo que resulta en una llamada al servicio de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) método.

Cuando se inicia un servicio, tiene un ciclo de vida que es independiente de la componente que se inició y el servicio puede ejecutarse en segundo plano de forma indefinida, incluso si el componente que se inicia destruido. Como tal, el servicio debería detenerse cuando su trabajo se hace llamando a [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) , u otro componente puede detenerlo llamando [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) .

Un componente de aplicación, como una actividad puede iniciar el servicio llamando al [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) y pasar una [Intent](http://developer.android.com/reference/android/content/Intent.html) que especifica el servicio e incluye a cualquier de datos para que utilice el servicio. El servicio recibe esta [Intent](http://developer.android.com/reference/android/content/Intent.html) en el [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) método.

Por ejemplo, supongamos que una actividad tiene que salvar algunos datos a una base de datos en línea. La actividad se puede iniciar un servicio de compañía y entregar los datos a salvo por el paso de una intención de [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) . El servicio recibe la intención de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) , se conecta a Internet y realiza la transacción de base de datos. Cuando la transacción se lleva a cabo, el servicio se detiene y se destruye.

**Precaución:** A los servicios se ejecuta en el mismo proceso que la aplicación en la que se declara y en el hilo principal de dicha solicitud, de forma predeterminada. Por lo tanto, si el servicio lleva a cabo operaciones intensivas o el bloqueo mientras el usuario interactúa con una actividad desde la misma aplicación, el servicio será más lento desempeño de la actividad. Para evitar el impacto en el rendimiento de aplicaciones, usted debe comenzar un nuevo hilo en el interior del servicio.

Tradicionalmente, hay dos clases que se pueden extender para crear un servicio de empezar:

[Service](http://developer.android.com/reference/android/app/Service.html)

Esta es la clase base para todos los servicios. Al extender esta clase, es importante que se crea un nuevo hilo en el que hacer todo el trabajo del servicio, debido a que el servicio utiliza hilo principal de la aplicación, por defecto, lo que podría retrasar la realización de cualquier actividad de su aplicación se está ejecutando.

[IntentService](http://developer.android.com/reference/android/app/IntentService.html)

Esta es una subclase de [Service](http://developer.android.com/reference/android/app/Service.html) que utiliza un subproceso de trabajo para manejar todas las peticiones de inicio, uno a la vez. Esta es la mejor opción si usted no requiere que el servicio de tramitar las solicitudes de forma simultánea. Todo lo que necesitas hacer es poner en práctica [onHandleIntent()](http://developer.android.com/reference/android/app/IntentService.html#onHandleIntent%28android.content.Intent%29) , que recibe la intención de cada solicitud de inicio para que pueda hacer el trabajo de fondo.

Las secciones siguientes describen cómo puede implementar su servicio a través de cualquiera de estas clases.

**Ampliación de la clase IntentService**

Dado que la mayoría comenzó a los servicios no es necesario para tramitar las solicitudes de forma simultánea (que en realidad puede ser un peligroso escenario multi-threading), probablemente es mejor si se implementa el servicio utilizando el [IntentService](http://developer.android.com/reference/android/app/IntentService.html) clase.

El [IntentService](http://developer.android.com/reference/android/app/IntentService.html) hace lo siguiente:

* Crea un subproceso de trabajo que se ejecuta por defecto todos los intentos entregados a [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) separados de hilo principal de la aplicación.
* Crea una cola de trabajo que pasa a una intención en el momento de su [onHandleIntent()](http://developer.android.com/reference/android/app/IntentService.html#onHandleIntent%28android.content.Intent%29) la aplicación, por lo que nunca tiene que preocuparse de multi-threading.
* Detiene el servicio después de todas las solicitudes de arranque han sido manipulados, por lo que nunca tiene que llamar a [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) .
* Proporciona la implementación predeterminada de [onBind()](http://developer.android.com/reference/android/app/IntentService.html#onBind%28android.content.Intent%29) que devuelve un valor nulo.
* Proporciona una implementación predeterminada de [onStartCommand()](http://developer.android.com/reference/android/app/IntentService.html#onStartCommand%28android.content.Intent,%20int,%20int%29) que envía la intención de la cola de trabajo y luego a su [onHandleIntent()](http://developer.android.com/reference/android/app/IntentService.html#onHandleIntent%28android.content.Intent%29) la aplicación.

Todo esto se suma el hecho de que todo lo que necesitas hacer es poner en práctica [onHandleIntent()](http://developer.android.com/reference/android/app/IntentService.html#onHandleIntent%28android.content.Intent%29) para hacer el trabajo proporcionado por el cliente. (Aunque, también es necesario para proporcionar un constructor pequeño para el servicio.)

He aquí un ejemplo de implementación de [IntentService](http://developer.android.com/reference/android/app/IntentService.html) :

public class extends HelloIntentService IntentService {

/ \*\*

\* Un constructor se requiere, y debe llamar a la super [IntentService(String)](http://developer.android.com/reference/android/app/IntentService.html#IntentService%28java.lang.String%29)

\* Constructor con un nombre para el subproceso de trabajo.

\* /

HelloIntentService pública () {

super ("HelloIntentService");

}

/ \*\*

\* El IntentService llama a este método desde el subproceso de trabajo por defecto con

\* La intención de que se inició el servicio. Cuando este método devuelve un valor, IntentService

\* Detiene el servicio, según corresponda.

\* /

@ Override

protected void onHandleIntent (intención de Intención) {

/ / Normalmente íbamos a hacer un trabajo aquí, como descargar un archivo.

/ / Para la muestra, sólo dormir durante 5 segundos.

a largo endTime = System.currentTimeMillis () + 5 \* 1000;

mientras que (System.currentTimeMillis () <endTime) {

synchronized (this) {

try {

esperar (endTime - System.currentTimeMillis ());

} Catch (Exception e) {

}

}

}

}

}

Eso es todo lo que necesita: un constructor y una implementación de [onHandleIntent()](http://developer.android.com/reference/android/app/IntentService.html#onHandleIntent%28android.content.Intent%29) .

Si decide reemplazar también otros métodos de devolución de llamada, como [onCreate()](http://developer.android.com/reference/android/app/IntentService.html#onCreate%28%29) , [onStartCommand()](http://developer.android.com/reference/android/app/IntentService.html#onStartCommand%28android.content.Intent,%20int,%20int%29) , o [onDestroy()](http://developer.android.com/reference/android/app/IntentService.html#onDestroy%28%29) , asegúrese de llamar a la implementación super, para que el [IntentService](http://developer.android.com/reference/android/app/IntentService.html) puede manejar adecuadamente la vida del subproceso de trabajo.

Por ejemplo, [onStartCommand()](http://developer.android.com/reference/android/app/IntentService.html#onStartCommand%28android.content.Intent,%20int,%20int%29) debe devolver a la aplicación por defecto (que es lo que la intención sea entregado a [onHandleIntent()](http://developer.android.com/reference/android/app/IntentService.html#onHandleIntent%28android.content.Intent%29) ):

@ Override

onStartCommand public int (intención de Intención, int banderas, int startID) {

Toast.makeText (this, "el servicio de partida", pan tostado.. LENGTH\_SHORT) show ();

volver super.onStartCommand (intención, banderas, startID);

}

Además [onHandleIntent()](http://developer.android.com/reference/android/app/IntentService.html#onHandleIntent%28android.content.Intent%29) , el único método del que no es necesario llamar a la superclase se [onBind()](http://developer.android.com/reference/android/app/IntentService.html#onBind%28android.content.Intent%29) (pero sólo es necesario para poner en práctica que si el servicio permite la unión).

En la siguiente sección, veremos cómo el mismo tipo de servicio se implementa cuando se amplíe la base de [Service](http://developer.android.com/reference/android/app/Service.html) clase, que es un código mucho más, pero que podría ser apropiada si es necesario para tramitar las solicitudes simultáneas de inicio.

**Ampliación de la clase de servicio**

Como se vio en la sección anterior, utilizando [IntentService](http://developer.android.com/reference/android/app/IntentService.html) hace que su puesta en práctica de un servicio comenzó muy simple. Sin embargo, si usted requiere el servicio para realizar multi-threading (en lugar de procesar las solicitudes de inicio a través de una cola de trabajo), entonces se puede ampliar el [Service](http://developer.android.com/reference/android/app/Service.html) la clase para manejar cada intento.

Por comparación, el código siguiente ejemplo es una aplicación del [Service](http://developer.android.com/reference/android/app/Service.html) la clase que realiza el trabajo exactamente igual que el ejemplo anterior usando [IntentService](http://developer.android.com/reference/android/app/IntentService.html) . Es decir, para cada solicitud de inicio, utiliza un subproceso de trabajo para realizar el trabajo y procesa sólo una solicitud a la vez.

HelloService public class Servicio {

privada mServiceLooper Looper;

mServiceHandler ServiceHandler privado;

/ / Manejador de que recibe los mensajes de la rosca

ServiceHandler privada última clase se extiende Handler {

pública ServiceHandler (Looper lanzadera) {

super (lanzadera);

}

@ Override

public void handleMessage (Mensaje msg) {

/ / Normalmente íbamos a hacer un trabajo aquí, como descargar un archivo.

/ / Para la muestra, sólo dormir durante 5 segundos.

a largo endTime = System.currentTimeMillis () + 5 \* 1000;

mientras que (System.currentTimeMillis () <endTime) {

synchronized (this) {

try {

esperar (endTime - System.currentTimeMillis ());

} Catch (Exception e) {

}

}

}

/ / Detener el servicio utilizando el startID, por lo que no se detienen

/ / El servicio en el centro de manejo de otro puesto de trabajo

stopSelf (msg.arg1);

}

}

@ Override

onCreate public void () {

/ / Inicia el hilo que ejecuta el servicio. Tenga en cuenta que se crea una

/ / Hilo separado porque el servicio normalmente se ejecuta en el proceso de

/ / Hilo conductor, que no queremos bloquear. También se hacen

/ / Fondo prioridad para el trabajo intensivo de la CPU no va a alterar nuestra interfaz de usuario.

Hilo HandlerThread = HandlerThread nueva ("ServiceStartArguments",

Proceso.THREAD\_PRIORITY\_BACKGROUND);

thread.start ();

/ / Obtener el Looper HandlerThread y utilizarlo para nuestro controlador

mServiceLooper thread.getLooper = ();

mServiceHandler = new ServiceHandler (mServiceLooper);

}

@ Override

onStartCommand public int (intención de Intención, int banderas, int startID) {

Toast.makeText (this, "el servicio de partida", pan tostado.. LENGTH\_SHORT) show ();

/ / Para cada solicitud de inicio, envíe un mensaje para iniciar un trabajo y entregar el

/ / Iniciar ID por lo que sabemos que la solicitud de que vamos a parar al terminar el trabajo

Mensaje msg = mServiceHandler.obtainMessage ();

msg.arg1 = startID;

mServiceHandler.sendMessage (msg);

/ / Si nos mataron, después de regresar de aquí, reinicie

volver START\_STICKY;

}

@ Override

pública IBinder onBind (intención de Intención) {

/ / Nosotros no proporcionamos vinculante, por lo que devolverá un valor nulo

return null;

}

@ Override

OnDestroy public void () {

Toast.makeText (this, "servicio hecho", pan tostado.. LENGTH\_SHORT) show ();

}

}

Como puede ver, es mucho más trabajo que el uso de [IntentService](http://developer.android.com/reference/android/app/IntentService.html) .

Sin embargo, debido a que manejan cada llamada a [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) usted mismo, usted puede realizar varias peticiones simultáneamente. Eso no es lo que hace este ejemplo, pero si eso es lo que usted desea, entonces usted puede crear un nuevo hilo para cada solicitud y ejecutar de inmediato (en lugar de esperar a que la petición anterior para terminar).

Observe que el [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) método debe devolver un entero. El número entero es un valor que describe cómo el sistema debe continuar con el servicio en caso de que el sistema que mata (como se mencionó anteriormente, la aplicación por defecto para [IntentService](http://developer.android.com/reference/android/app/IntentService.html) se encarga de esto por usted, a pesar de que son capaces de modificarlo). El valor de retorno de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) debe ser una de las siguientes constantes:

[START\_NOT\_STICKY](http://developer.android.com/reference/android/app/Service.html#START_NOT_STICKY)

Si el sistema mata el servicio después de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) devuelve, *no se* vuelva a crear el servicio, a menos que se espera de los intentos de entregar. Esta es la opción más segura para evitar la utilización de su servicio cuando no es necesario y cuando su aplicación simplemente puede reiniciar los trabajos sin terminar.

[START\_STICKY](http://developer.android.com/reference/android/app/Service.html#START_STICKY)

Si el sistema mata el servicio después de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) devuelve, vuelva a crear el servicio y la llamada [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) , pero *no* entregar de nuevo la intención última. En cambio, el sistema llama [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) con una intención nula, a menos que hubiera intenciones pendientes de iniciar el servicio, en cuyo caso, esos intentos se entregan. Esto es conveniente para los jugadores de los medios de comunicación (o similares) que no se ejecutan los comandos, pero el funcionamiento por tiempo indefinido ya la espera de un puesto de trabajo.

[START\_REDELIVER\_INTENT](http://developer.android.com/reference/android/app/Service.html#START_REDELIVER_INTENT)

Si el sistema mata el servicio después de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) devuelve, vuelva a crear el servicio y la llamada [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) con la intención última que se entregó al servicio. Las intenciones pendientes se entregan a su vez. Esto es conveniente para los servicios que se activa se realiza un trabajo que debe ser inmediatamente reanudó, como la descarga de un archivo.

Para más detalles sobre estos valores de retorno, consulte la documentación de referencia vinculados a cada constante.

**Inicio de un servicio**

Usted puede iniciar un servicio de una actividad o componente de otra aplicación haciendo pasar una [Intent](http://developer.android.com/reference/android/content/Intent.html) (especificando el inicio del servicio) para [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) . El sistema Android, llama al servicio de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) el método y le pasa la [Intent](http://developer.android.com/reference/android/content/Intent.html) . (Nunca se debe llamar a [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) directamente.)

Por ejemplo, una actividad puede iniciar el servicio de ejemplo, en la sección anterior ( HelloSevice ) con la intención explícita con [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) :

Intención de Intención = nuevo intento (esto, HelloService.class);

StartService (intención);

El [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) método devuelve inmediatamente y el sistema Android, llama al servicio de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) método. Si el servicio no se está ejecutando, el sistema de llamadas de primera [onCreate()](http://developer.android.com/reference/android/app/Service.html#onCreate%28%29) , a continuación, llama [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) .

Si el servicio no también proporcionan vinculante, la intención de entregar con [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) es el único modo de comunicación entre el componente de aplicación y el servicio. Sin embargo, si desea que el servicio para enviar un resultado de nuevo, el cliente que inicia el servicio puede crear un [PendingIntent](http://developer.android.com/reference/android/app/PendingIntent.html) de una emisión (con [getBroadcast()](http://developer.android.com/reference/android/app/PendingIntent.html#getBroadcast%28android.content.Context,%20int,%20android.content.Intent,%20int%29) ) y entregarlo al servicio de la [Intent](http://developer.android.com/reference/android/content/Intent.html) que se inicia el servicio. El servicio puede entonces utilizar la difusión para entregar un resultado.

Varias solicitudes para iniciar el servicio en el resultado de múltiples llamadas correspondientes al servicio de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) . Sin embargo, sólo una petición para detener el servicio (con [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) o [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) ) es necesario para detenerla.

**Detención de un servicio de**

Un servicio iniciado debe gestionar su propio ciclo de vida. Es decir, el sistema no se detiene o destruir el servicio a menos que se debe recuperar la memoria del sistema y el servicio sigue funcionando después de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) devuelve. Por lo tanto, el servicio debe pararse llamando [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) u otro componente puede detenerlo llamando [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) .

Una vez solicitada para detener a [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) o [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) , el sistema destruye el servicio tan pronto como sea posible.

Sin embargo, si el servicio se encarga de múltiples solicitudes de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) al mismo tiempo, entonces usted no debe detener el servicio cuando haya terminado de procesar una solicitud de inicio, porque es posible que desde entonces han recibido una solicitud de nuevo comienzo (con parada en el final de la primera petición pone fin el segundo). Para evitar este problema, puede utilizar [stopSelf(int)](http://developer.android.com/reference/android/app/Service.html#stopSelf%28int%29) para asegurarse de que su petición para detener el servicio se basa siempre en la solicitud de inicio más reciente. Es decir, cuando se llama a [stopSelf(int)](http://developer.android.com/reference/android/app/Service.html#stopSelf%28int%29) , se pasa el ID de la solicitud de inicio (el startId entregado a [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) ) a la que corresponde a su solicitud de parada. Entonces, si el servicio recibido una solicitud de nueva oportunidad antes de que fuera capaz de llamar a [stopSelf(int)](http://developer.android.com/reference/android/app/Service.html#stopSelf%28int%29) , entonces el ID no coincide y el servicio no se detendrá.

**Precaución:** Es importante que su aplicación deja sus servicios cuando se hace trabajando, para evitar el desperdicio de los recursos del sistema y el poder de consumir batería. Si es necesario, otros componentes pueden detener el servicio llamando [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) . Incluso si usted habilitar el enlace para el servicio, siempre se debe detener el servicio a sí mismo si es que alguna vez recibió una llamada a [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) .

Para obtener más información sobre el ciclo de vida de un servicio, vea la sección siguiente sobre [Gestión del Ciclo de Vida de un Servicio](http://developer.android.com/guide/topics/fundamentals/services.html#Lifecycle) .

**Creación de un servicio consolidado**

Un servicio de la envolvente es una que permite que los componentes de aplicación que se unen a ella llamando [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) con el fin de crear una conexión desde hace mucho tiempo (y por lo general no permiten que los componentes para *iniciar* llamando [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) ).

Usted debe crear un servicio consolidado cuando se quiere interactuar con el servicio de las actividades y otros componentes en su aplicación o para exponer algunas de las funciones de la aplicación para otras aplicaciones, a través de la comunicación entre procesos (IPC).

Para crear un servicio consolidado, debe implementar la [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) el método de devolución de llamada para devolver un [IBinder](http://developer.android.com/reference/android/os/IBinder.html) que define la interfaz para la comunicación con el servicio. Otros componentes de la aplicación puede llamar [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) para recuperar la interfaz y comenzar a llamar a métodos en el servicio. El servicio no vive más que para servir a la componente de la aplicación que está unido a ella, así que cuando no hay componentes vinculados con el servicio, el sistema se destruye *(no es* necesario para detener un servicio consolidado en la forma en que debe, cuando se inicia el servicio a través de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) ).

Para crear un servicio consolidado, lo primero que debes hacer es definir la interfaz que especifica cómo un cliente puede comunicarse con el servicio. Esta interfaz entre el servicio y un cliente debe ser una implementación de [IBinder](http://developer.android.com/reference/android/os/IBinder.html) y es lo que el servicio debe regresar de la [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) el método de devolución de llamada. Una vez que el cliente recibe el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) , puede empezar a interactuar con el servicio a través de esa interfaz.

Varios clientes pueden enlazar con el servicio a la vez. Cuando un cliente se lleva a cabo la interacción con el servicio, llama a [unbindService()](http://developer.android.com/reference/android/content/Context.html#unbindService%28android.content.ServiceConnection%29) para desenlazar. Una vez que no hay clientes vinculados al servicio, el sistema destruye el servicio.

Hay varias formas de implementar un servicio consolidado y la aplicación es más complicado que un servicio se ha iniciado, por lo que la discusión de servicios del que aparezca en un documento separado sobre [los servicios están consolidados](http://developer.android.com/guide/topics/fundamentals/bound-services.html) .

**El envío de notificaciones al usuario**

Una vez en funcionamiento, un servicio puede notificar al usuario de eventos mediante [Notificaciones Toast](http://developer.android.com/guide/topics/ui/notifiers/toasts.html) o [Notificaciones barra de estado](http://developer.android.com/guide/topics/ui/notifiers/notifications.html) .

Una notificación brindis es un mensaje que aparece en la superficie de la ventana actual por un momento y luego desaparece, mientras que una notificación de la barra de estado proporciona un icono en la barra de estado con un mensaje, que el usuario puede seleccionar con el fin de realizar una acción (por ejemplo, como iniciar una actividad).

Por lo general, una notificación de la barra de estado es la mejor técnica cuando un trabajo de fondo ha terminado (como una descarga de archivos completo) y el usuario puede ahora actuar en consecuencia. Cuando el usuario selecciona la notificación de la vista ampliada, la notificación se puede iniciar una actividad (por ejemplo, para ver el archivo descargado).

Ver las [notificaciones de pan tostado](http://developer.android.com/guide/topics/ui/notifiers/toasts.html) o [notificaciones de la barra de estado de](http://developer.android.com/guide/topics/ui/notifiers/notifications.html) guías de desarrollo para más información.

**Ejecución de un servicio en el primer plano**

Un servicio de primer plano es un servicio que se considera que es algo que el usuario es consciente de forma activa y no tanto a un candidato para el sistema de matar cuando se baja de la memoria. Un servicio de primer plano debe entregar una notificación de la barra de estado, que se coloca en el marco del "curso", título, lo que significa que la notificación no puede ser despedido a menos que el servicio se detiene o se retira del primer plano.

Por ejemplo, un reproductor de música que reproduce música de un servicio debe ser configurado para ejecutarse en primer plano, porque el usuario es explícitamente conscientes de su funcionamiento. La notificación en la barra de estado puede indicar la canción actual y permitir al usuario poner en marcha una actividad para interactuar con el reproductor de música.

Para solicitar que su servicio se ejecuta en primer plano, llame [startForeground()](http://developer.android.com/reference/android/app/Service.html#startForeground%28int,%20android.app.Notification%29) . Este método toma dos parámetros: un entero que identifica de forma única la notificación y la [Notification](http://developer.android.com/reference/android/app/Notification.html) de la barra de estado. Por ejemplo:

Notificación de la notificación = nueva notificación (R.drawable.icon, gettext (R.string.ticker\_text),

System.currentTimeMillis ());

NotificationIntent Intención = nuevo intento (esto, ExampleActivity.class);

PendingIntent PendingIntent = PendingIntent.getActivity (este, 0, notificationIntent, 0);

notification.setLatestEventInfo (esto, gettext (R.string.notification\_title),

gettext (R.string.notification\_message), pendingIntent);

startForeground (ONGOING\_NOTIFICATION, la notificación);

Para quitar el servicio desde el primer plano, llame [stopForeground()](http://developer.android.com/reference/android/app/Service.html#stopForeground%28boolean%29) . Este método toma un valor booleano, indicando si se debe eliminar la notificación de la barra de estado también. Este método *no* se detiene el servicio. Sin embargo, si se detiene el servicio mientras se está aún en marcha en el primer plano, a continuación, la notificación también se elimina.

**Nota:** Los métodos [startForeground()](http://developer.android.com/reference/android/app/Service.html#startForeground%28int,%20android.app.Notification%29) y [stopForeground()](http://developer.android.com/reference/android/app/Service.html#stopForeground%28boolean%29) se introdujeron en Android 2.0 (Nivel API 5). Con el fin de ejecutar el servicio en el primer plano en las versiones anteriores de la plataforma, debe utilizar el anterior setForeground() el método de ver la [startForeground()](http://developer.android.com/reference/android/app/Service.html#startForeground%28int,%20android.app.Notification%29) para obtener información acerca de cómo proporcionar compatibilidad con versiones anteriores.

Para obtener más información acerca de las notificaciones, consulte [Creación de notificaciones de estado de barra](http://developer.android.com/guide/topics/ui/notifiers/notifications.html) .

**Gestión del Ciclo de Vida de un Servicio de**

El ciclo de vida de un servicio es mucho más simple que el de una actividad. Sin embargo, es aún más importante que usted preste mucha atención a cómo su servicio se crea y destruye, porque un servicio puede ejecutarse en segundo plano sin que el usuario sea consciente.

El servicio de ciclo de vida, desde cuando se crea cuando se destruye, puede seguir dos caminos diferentes:

* Un servicio comenzó

El servicio se crea cuando se llama a otro componente [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) . El servicio a continuación, ejecuta de forma indefinida y deben pararse llamando [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) . Otro de los componentes también se puede detener el servicio llamando al [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) . Cuando se detiene el servicio, el sistema lo destruye ..

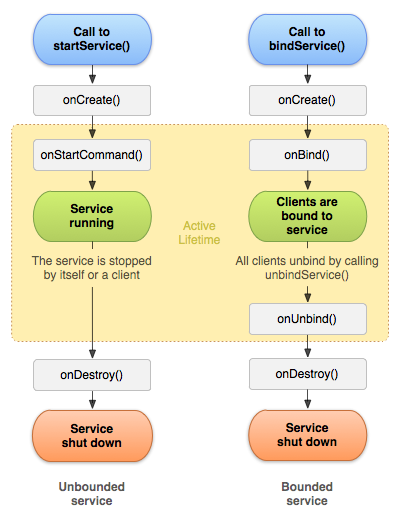
* Un servicio consolidado

El servicio se crea cuando otro componente (un cliente) llama [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) . El cliente se comunica con el servicio a través de un [IBinder](http://developer.android.com/reference/android/os/IBinder.html) interfaz. El cliente puede cerrar la conexión llamando a [unbindService()](http://developer.android.com/reference/android/content/Context.html#unbindService%28android.content.ServiceConnection%29) . Varios clientes pueden enlazar con el mismo servicio y cuando todos ellos desatar, el sistema destruye el servicio. (El servicio *no* tiene por qué detenerse.)

Estos dos caminos no son totalmente independientes. Es decir, usted puede unirse a un servicio que se inició ya con [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) . Por ejemplo, un servicio de música de fondo puede ser iniciado por vocación [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) con una [Intent](http://developer.android.com/reference/android/content/Intent.html) que identifica a la música para jugar. Más tarde, posiblemente cuando el usuario desea ejercer algún control sobre el jugador o para obtener información sobre la canción actual, una actividad que puede unirse a este servicio llamando al [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) . En casos como este, [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) o [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) en realidad no detener el servicio hasta que todos los clientes de desatar.

**La aplicación de las devoluciones de llamada de ciclo de vida**

Al igual que una actividad, un servicio dispone de métodos de retrollamada de ciclo que se pueden implementar para controlar los cambios en el estado del servicio y realizar el trabajo en los momentos adecuados. El servicio de esqueleto siguiente muestra cada uno de los métodos del ciclo de vida:



**FIGURA 2.** El ciclo de vida del servicio. El diagrama de la izquierda muestra el ciclo de vida cuando el servicio se crea con [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) y el diagrama de la derecha muestra el ciclo de vida cuando el servicio se crea con [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) .

ExampleService public class Servicio {

int mStartMode; / / indica cómo debe comportarse si el servicio es asesinado

IBinder mBinder; / / interfaz para los clientes que se unen

boolean mAllowRebind; / / indica si se debe utilizar onRebind

@Override

public void [onCreate](http://developer.android.com/reference/android/app/Service.html#onCreate%28%29) () {

/ / El servicio se está creando

}

@ Override

public int [onStartCommand](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) (intención de Intención, int banderas, int startID) {

/ / El servicio se inicia, debido a una llamada a [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29)

volver *mStartMode;*

}

@ Override

pública IBinder [onBind](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) (intención de Intención) {

/ / Cliente A es vinculante para el servicio con [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29)

volver *mBinder;*

}

@ Override

public boolean [onUnbind](http://developer.android.com/reference/android/app/Service.html#onUnbind%28android.content.Intent%29) (intención de Intención) {

/ / Todos los clientes tienen libre con [unbindService()](http://developer.android.com/reference/android/content/Context.html#unbindService%28android.content.ServiceConnection%29)

volver *mAllowRebind;*

}

@ Override

public void [onRebind](http://developer.android.com/reference/android/app/Service.html#onRebind%28android.content.Intent%29) (intención de Intención) {

/ / Cliente A es vinculante para el servicio con [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) ,

/ / Después de onUnbind () ya ha sido llamado

}

@ Override

public void [onDestroy](http://developer.android.com/reference/android/app/Service.html#onDestroy%28%29) () {

/ / El servicio ya no se utiliza y se está destruyendo

}

}

**Nota:** A diferencia de los métodos de devolución de llamada de la actividad del ciclo de vida, *no es* necesario llamar a la implementación de la superclase de estos métodos de devolución de llamada.

Mediante la implementación de estos métodos, puede controlar dos bucles anidados del ciclo de vida del servicio:

* La **vida entera** de un servicio que pasa entre el momento en [onCreate()](http://developer.android.com/reference/android/app/Service.html#onCreate%28%29) se llama y el tiempo [onDestroy()](http://developer.android.com/reference/android/app/Service.html#onDestroy%28%29) devuelve. Al igual que una actividad, un servicio realiza su configuración inicial, en [onCreate()](http://developer.android.com/reference/android/app/Service.html#onCreate%28%29) y libera todos los recursos que quedan en [onDestroy()](http://developer.android.com/reference/android/app/Service.html#onDestroy%28%29) . Por ejemplo, un servicio de reproducción de música podría crear el hilo donde la música que se reproduce en el [onCreate()](http://developer.android.com/reference/android/app/Service.html#onCreate%28%29) , y luego se detiene el hilo en [onDestroy()](http://developer.android.com/reference/android/app/Service.html#onDestroy%28%29) .

El [onCreate()](http://developer.android.com/reference/android/app/Service.html#onCreate%28%29) y [onDestroy()](http://developer.android.com/reference/android/app/Service.html#onDestroy%28%29) se llama a métodos para todos los servicios, si están creados por [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) o [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) .

* La **vida activa** de un servicio comienza con una llamada a cualquiera de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) o [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) . Cada método se entregó la [Intent](http://developer.android.com/reference/android/content/Intent.html) que se ha pasado a cualquiera de [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) o [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) , respectivamente.

Si se inicia el servicio, el tiempo de vida activa termina al mismo tiempo que los extremos de toda la vida (el servicio sigue activo incluso después de [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) devuelve). Si el servicio está obligado, el tiempo de vida activa termina cuando [onUnbind()](http://developer.android.com/reference/android/app/Service.html#onUnbind%28android.content.Intent%29) devuelve.

**Nota: A** pesar de un inicio del servicio se interrumpe por una llamada a cualquiera de [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) o [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) , no hay una devolución de llamada correspondiente para el servicio (no hay onStop() de devolución de llamada). Por lo tanto, a menos que el servicio está destinado a un cliente, el sistema lo destruye cuando el servicio se detiene- [onDestroy()](http://developer.android.com/reference/android/app/Service.html#onDestroy%28%29) es sólo la devolución de llamada recibida.

La figura 2 ilustra los métodos de devolución de llamada típicos para un servicio. Aunque la cifra que separa los servicios que son creados por [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) de los creados por [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) , tenga en cuenta que cualquier servicio, sin importar cómo se inició, lo que potencialmente puede permitir a los clientes que se unen a ella. Por lo tanto, un servicio que se inició al principio con [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) (por un cliente llamado [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) ) todavía puede recibir una llamada a [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) (cuando un cliente llama a [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) ).

Para obtener más información sobre cómo crear un servicio que proporciona vinculante, consulte el [Servicio Bound](http://developer.android.com/guide/topics/fundamentals/bound-services.html) documento, que incluye más información sobre el [onRebind()](http://developer.android.com/reference/android/app/Service.html#onRebind%28android.content.Intent%29) el método de devolución de llamada en el apartado de [Gestión del Ciclo de Vida de un Servicio de Bound](http://developer.android.com/guide/topics/fundamentals/bound-services.html#Lifecycle) .

[↑ Ir arriba](http://developer.android.com/guide/topics/fundamentals/services.html#top)

A excepción de lo señalado, este contenido está disponible bajo [Apache 2.0](http://www.google.com/url?q=http%3A%2F%2Fwww.apache.org%2Flicenses%2FLICENSE-2.0&sa=D&sntz=1&usg=AFQjCNFPaSHdvCi6VD7GwgEkKspvD3d_tw) . Para más detalles y restricciones, consulte la [Licencia de Contenido](http://developer.android.com/license.html) .

Android 4.0 r1 - 01 de mayo 2012 17:37

# Servicios enlazados

## Vista Rapida

* + Un servicio de la envolvente permite otros componentes que se unen a ella, con el fin de interactuar con él y realizar la comunicación entre
  + Un servicio unido se destruyen una vez que todos los clientes de desvincular, a menos que se inició el servicio también

## En este documento

* + [Los puntos básicos.](http://developer.android.com/guide/topics/fundamentals/bound-services.html#Basics)
  + [Creación de un servicio consolidado](http://developer.android.com/guide/topics/fundamentals/bound-services.html#Creating)
    1. [Ampliación de la clase Binder](http://developer.android.com/guide/topics/fundamentals/bound-services.html#Binder)
    2. [El uso de un mensajero](http://developer.android.com/guide/topics/fundamentals/bound-services.html#Messenger)
  + [Enlazar a un servicio](http://developer.android.com/guide/topics/fundamentals/bound-services.html#Binding)
  + [Gestión del Ciclo de Vida de un Servicio de Consolidada](http://developer.android.com/guide/topics/fundamentals/bound-services.html#Lifecycle)

## Clases de clave

* + [Service](http://developer.android.com/reference/android/app/Service.html)
  + [ServiceConnection](http://developer.android.com/reference/android/content/ServiceConnection.html)
  + [IBinder](http://developer.android.com/reference/android/os/IBinder.html)

## Las muestras

* + [RemoteService](http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/app/RemoteService.html)
  + [LocalService](http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/app/LocalService.html)

## Véase también

* + [servicios:](http://developer.android.com/guide/topics/fundamentals/services.html)

Un servicio consolidado es el servidor en una interfaz de cliente-servidor. Un servicio de la envolvente permite que los componentes (tales como las actividades) para enlazar con el servicio, enviar solicitudes, recibir respuestas, e incluso realizar la comunicación entre procesos (IPC). Un servicio ligado habitualmente vive sólo mientras éste se sirve a otro componente de la aplicación y no se ejecuta en segundo plano de forma indefinida.

En este documento se muestra cómo crear un servicio consolidado, incluyendo la manera de enlazar con el servicio de otros componentes de aplicación. Sin embargo, también debe referirse a la [Servicios de](http://developer.android.com/guide/topics/fundamentals/services.html) documento para obtener información adicional acerca de los servicios en general, tales como la forma de entregar las notificaciones de un servicio, configure el servicio se ejecute en primer plano, y mucho más.

## Los puntos básicos.

Un servicio consolidado es una implementación del [Service](http://developer.android.com/reference/android/app/Service.html) clase que permite que otras aplicaciones se unen a él e interactuar con él. Para proporcionar vinculante para un servicio, debe implementar la [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) el método de devolución de llamada. Este método devuelve un [IBinder](http://developer.android.com/reference/android/os/IBinder.html) objeto que define la interfaz de programación que los clientes pueden utilizar para interactuar con el servicio.

### Enlazar a un servicio iniciado

Como se discutió en el [Servicios de](http://developer.android.com/guide/topics/fundamentals/services.html) documento, puede crear un servicio que es al mismo tiempo comenzó y los consolidados. Es decir, el servicio se puede iniciar llamando al [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) , que permite que el servicio se ejecute de forma indefinida, y también permitir a un cliente que se unen a este servicio llamando al [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) .

Si se permite su servicio para ponerse en marcha y los consolidados, a continuación, cuando el servicio se ha iniciado, el sistema no destruye el servicio cuando todos los clientes de desatar. En su lugar, de forma explícita, debe detener el servicio, llamando [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) o [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) .

Aunque por lo general debe aplicar cualquiera [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) o [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) , a veces es necesario aplicar ambos. Por ejemplo, un reproductor de música puede resultar útil para permitir su servicio para ejecutar de forma indefinida y también proporcionar vinculante. De esta manera, una actividad que puede iniciar el servicio a jugar un poco de música y la música continúa reproduciéndose incluso si el usuario sale de la aplicación. Entonces, cuando el usuario vuelve a la aplicación, la actividad puede enlazar con el servicio para recuperar el control de la reproducción.

Asegúrese de leer la sección sobre la [Gestión del Ciclo de Vida de un Servicio de Bound](http://developer.android.com/guide/topics/fundamentals/bound-services.html#Lifecycle) , para obtener más información acerca del ciclo de vida de servicio al añadir el enlace a un inicio del servicio.

Un cliente puede enlazar con el servicio llamando al [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) . Cuando lo hace, debe proporcionar una implementación de [ServiceConnection](http://developer.android.com/reference/android/content/ServiceConnection.html) , que supervisa la conexión con el servicio. El [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) método devuelve inmediatamente, y sin un valor, pero cuando el sistema Android crea la conexión entre el cliente y el servicio, llama a [onServiceConnected()](http://developer.android.com/reference/android/content/ServiceConnection.html#onServiceConnected%28android.content.ComponentName,%20android.os.IBinder%29) en el [ServiceConnection](http://developer.android.com/reference/android/content/ServiceConnection.html) , para entregar el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) que el cliente puede utilizar para comunicarse con el servicio.

Varios clientes pueden conectar con el servicio a la vez. Sin embargo, el sistema llama a su servicio [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) para recuperar el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) sólo cuando el primer cliente se une. Entonces, el sistema ofrece la misma [IBinder](http://developer.android.com/reference/android/os/IBinder.html) a todos los clientes adicionales que se unen, sin llamar a [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) de nuevo.

Cuando el último cliente se suelta del servicio, el sistema destruye el servicio (a menos que se inició el servicio también por [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) ).

Cuando se implementa el servicio de la envolvente, la parte más importante es la definición de la interfaz que su [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) devuelve el método de devolución de llamada. Hay varias formas distintas de definir el servicio de [IBinder](http://developer.android.com/reference/android/os/IBinder.html) interfaz y la sección siguiente se examinan cada técnica.

## Creación de un servicio consolidado

Al crear un servicio que proporciona vinculante, debe proporcionar un [IBinder](http://developer.android.com/reference/android/os/IBinder.html) que proporciona la interfaz de programación que los clientes pueden utilizar para interactuar con el servicio. Hay tres maneras de definir la interfaz:

[Ampliación de la clase Binder](http://developer.android.com/guide/topics/fundamentals/bound-services.html#Binder)

Si el servicio es privado para su propia aplicación y se ejecuta en el mismo proceso que el cliente (que es común), debe crear la interfaz mediante la ampliación del [Binder](http://developer.android.com/reference/android/os/Binder.html) clase y devolver una instancia de ella desde [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) . El cliente recibe el [Binder](http://developer.android.com/reference/android/os/Binder.html) y se puede utilizar para acceder directamente a los métodos públicos disponibles ya sea en la [Binder](http://developer.android.com/reference/android/os/Binder.html) la aplicación o incluso el [Service](http://developer.android.com/reference/android/app/Service.html) .

Esta es la técnica preferida cuando el servicio es más que un trabajo de fondo para su propia aplicación. La única razón por la que no crearía su interfaz de esta manera es porque el servicio es utilizado por otras aplicaciones o en procesos separados.

[El uso de un mensajero](http://developer.android.com/guide/topics/fundamentals/bound-services.html#Messenger)

Si necesita que su interfaz para trabajar a través de diferentes procesos, se puede crear una interfaz para el servicio con un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) . De esta manera, el servicio define un [Handler](http://developer.android.com/reference/android/os/Handler.html) que responde a diferentes tipos de [Message](http://developer.android.com/reference/android/os/Message.html) objetos. Este [Handler](http://developer.android.com/reference/android/os/Handler.html) es la base de un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) que luego se puede compartir un [IBinder](http://developer.android.com/reference/android/os/IBinder.html) con el cliente, permitiendo que el cliente para enviar comandos al servicio a través de [Message](http://developer.android.com/reference/android/os/Message.html) los objetos. Además, el cliente puede definir un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) de su propia lo que el servicio puede enviar mensajes de un lado.

Esta es la forma más sencilla de llevar a cabo la comunicación entre procesos (IPC), debido a que los [Messenger](http://developer.android.com/reference/android/os/Messenger.html) colas de todas las solicitudes en un solo hilo de modo que usted no tiene que diseñar su servicio para ser thread-safe.

Uso de lisina digestible ileal aparente

Lisina digestible ileal aparente (Android Interface Definition Language) realiza todo el trabajo para descomponer los objetos en las primitivas que el sistema operativo pueda entender y Marshall a través de los procesos para llevar a cabo la CIA. La técnica anterior, con un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) , se basa realmente en lisina digestible ileal aparente ya que su estructura subyacente. Como se mencionó anteriormente, el [Messenger](http://developer.android.com/reference/android/os/Messenger.html) crea una cola de todas las solicitudes de los clientes en un solo hilo, por lo que el servicio recibe las solicitudes de una en una. Sin embargo, si usted quiere que su servicio para atender las peticiones de forma simultánea, entonces usted puede utilizar directamente en lisina digestible ileal aparente. En este caso, el servicio debe ser capaz de multi-threading y construir thread-safe.

Para utilizar directamente lisina digestible ileal aparente, se debe crear una .aidl archivo que define la interfaz de programación. Las herramientas del SDK de Android utilizar este archivo para generar una clase abstracta que implementa la interfaz y se ocupa de la IPC, que luego puede extenderse dentro de su servicio.

**Nota: La** mayoría de las aplicaciones **no deben** utilizar lisina digestible ileal aparente para crear un servicio consolidado, ya que pueden requerir capacidades de multithreading y puede resultar en una ejecución más complicada. Como tal, lisina digestible ileal aparente no es adecuado para la mayoría de las aplicaciones y este documento no se explica cómo utilizarlo para su servicio. Si está seguro de que es necesario utilizar lisina digestible ileal aparente directamente, consulte la [lisina digestible ileal aparente](http://developer.android.com/guide/developing/tools/aidl.html) documento.

### Ampliación de la clase Binder

Si su servicio es utilizado sólo por la aplicación local y no necesita trabajar a través de los procesos, entonces usted puede implementar su propio [Binder](http://developer.android.com/reference/android/os/Binder.html) clase que proporciona el acceso directo con el cliente a los métodos públicos de la administración.

**Nota:** Esto sólo funciona si el cliente y el servicio están en la misma aplicación y el proceso, que es más común. Por ejemplo, esto funciona bien para una aplicación de música que tiene que obligar a una actividad a su propio servicio que se está reproduciendo música en el fondo.

He aquí cómo configurarlo:

1. En su servicio, crear una instancia de [Binder](http://developer.android.com/reference/android/os/Binder.html) que:
   * contiene los métodos públicos que el cliente puede llamar
   * devuelve el actual [Service](http://developer.android.com/reference/android/app/Service.html) ejemplo, que tiene métodos públicos que el cliente puede llamar
   * o bien, devuelve una instancia de otra clase ofrecida por el servicio con los métodos públicos que el cliente puede llamar
2. Regresa esta instancia de [Binder](http://developer.android.com/reference/android/os/Binder.html) de la [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) el método de devolución de llamada.
3. En el cliente, recibirá el [Binder](http://developer.android.com/reference/android/os/Binder.html) de la [onServiceConnected()](http://developer.android.com/reference/android/content/ServiceConnection.html#onServiceConnected%28android.content.ComponentName,%20android.os.IBinder%29) el método de devolución de llamada y hacer llamadas al servicio de la envolvente utilizando los métodos proporcionados.

**Nota:** La razón del servicio y el cliente debe estar en la misma aplicación es que el cliente pueda convertir el objeto devuelto y bien llamar a su API. El servicio y el cliente también debe estar en el mismo proceso, ya que esta técnica no realiza ningún cálculo de referencias en todos los procesos.

Por ejemplo, aquí es un servicio que proporciona a los clientes acceso a los métodos en el servicio a través de un [Binder](http://developer.android.com/reference/android/os/Binder.html) ejecución:

LocalService public class Servicio {

/ / Cuaderno de atención a clientes

final privado IBinder mBinder = new LocalBinder ();

/ / Random Number Generator

final privado aleatorio mGenerator = new Random ();

/ \*\*

\* La clase utilizada para la Carpeta de cliente. Porque sabemos que este servicio siempre

\* Se ejecuta en el mismo proceso que sus clientes, no es necesario para hacer frente a la CIP.

\* /

pública LocalBinder clase extiende Cuaderno {

LocalService GetService () {

/ / Devuelve la instancia de LocalService que los clientes puedan llamar a los métodos públicos

volver LocalService.this;

}

}

@Override

pública IBinder onBind (intención de Intención) {

volver mBinder;

}

/ \*\* Método para los clientes \* /

getRandomNumber public int () {

volver mGenerator.nextInt (100);

}

}

El LocalBinder proporciona la getService() método para que los clientes recuperen la instancia actual de LocalService . Esto permite a los clientes a llamar a los métodos públicos de la administración. Por ejemplo, los clientes pueden llamar a getRandomNumber() del servicio.

He aquí una actividad que se une a LocalService y pide getRandomNumber() , cuando se hace clic en un botón:

BindingActivity public class Actividad {

LocalService MSERVICE;

boolean mBound = false;

@Override

protected void onCreate (paquete savedInstanceState) {

super.onCreate (savedInstanceState);

setContentView (R.layout.main);

}

@Override

protected void OnStart () {

super.onStart ();

/ / Enlazar a LocalService

Intención de Intención = nuevo intento (esto, LocalService.class);

bindService (intención, mConnection, Contexto.BIND\_AUTO\_CREATE);

}

@Override

protected void OnStop () {

super.onStop ();

/ / Separar del servicio

if (mBound) {

unbindService (mConnection);

mBound = false;

}

}

/ \*\* Se le llama cuando se hace clic en un botón (el botón en el archivo de diseño se une a

\* Este método con el androide: atributo onClick) \* /

public void onButtonClick (Ver v) {

if (mBound) {

/ / Llamar a un método de la LocalService.

/ / Sin embargo, si esta convocatoria fuera algo que se puede bloquear, a continuación, esta solicitud debe

/ / Se producen en un hilo separado para evitar ralentizar el rendimiento de la actividad.

int num = mService.getRandomNumber ();

Toast.makeText (this, "número:" + num, Toast.. LENGTH\_SHORT) show ();

}

}

/ \*\* Define las devoluciones de llamada de enlace de servicio, pasó a bindService () \* /

mConnection ServiceConnection privada ServiceConnection = new () {

@Override

public void onServiceConnected (className ComponentName,

IBinder servicio) {

/ / Hemos obligado a LocalService, emitir el IBinder y obtener instancia LocalService

LocalBinder ligante = (LocalBinder) de servicios;

MSERVICE binder.getService = ();

mBound = true;

}

@Override

public void onServiceDisconnected (ComponentName arg0) {

mBound = false;

}

};

}

El ejemplo anterior muestra cómo el cliente se une al servicio a través de una implementación de [ServiceConnection](http://developer.android.com/reference/android/content/ServiceConnection.html) y el [onServiceConnected()](http://developer.android.com/reference/android/content/ServiceConnection.html#onServiceConnected%28android.content.ComponentName,%20android.os.IBinder%29) de devolución de llamada. La siguiente sección proporciona más información acerca de este proceso de unirse al servicio.

**Nota:** El ejemplo anterior no se refiere explícitamente desvincular del servicio, pero todos los clientes deben dirimir en el momento oportuno (por ejemplo, cuando la actividad se detiene).

Para más código de ejemplo, vea el [LocalService.java](http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/app/LocalService.html) de clases y la [LocalServiceActivities.java](http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/app/LocalServiceActivities.html) clase en [ApiDemos](http://developer.android.com/resources/samples/ApiDemos/index.html) .

### El uso de un mensajero

#### En comparación con lisina digestible ileal aparente

Cuando usted necesita para llevar a cabo la IPC, con un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) para su interfaz es más simple de implementar con lisina digestible ileal aparente, porque [Messenger](http://developer.android.com/reference/android/os/Messenger.html) las colas de todas las llamadas al servicio, mientras que, una interfaz de lisina digestible ileal aparente pura envía solicitudes simultáneas para el servicio, el cual debe manejar multi-threading .

Para la mayoría de las aplicaciones, el servicio no se necesita realizar multi-threading, así que usar un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) permite que el servicio para manejar una llamada a la vez. Si es importante que el servicio de ser multi-hilo, entonces usted debe utilizar [lisina digestible ileal aparente](http://developer.android.com/guide/developing/tools/aidl.html) para definir la interfaz.

Si necesita que su servicio para comunicarse con los procesos remotos, entonces usted puede utilizar un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) para proporcionar la interfaz de su servicio. Esta técnica le permite realizar la comunicación entre procesos (IPC) sin la necesidad de utilizar lisina digestible ileal aparente.

He aquí un resumen de cómo utilizar un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) :

* El servicio implementa un [Handler](http://developer.android.com/reference/android/os/Handler.html) que recibe una devolución de llamada para cada llamada de un cliente.
* El [Handler](http://developer.android.com/reference/android/os/Handler.html) se utiliza para crear un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) objeto (que es una referencia al [Handler](http://developer.android.com/reference/android/os/Handler.html) ).
* El [Messenger](http://developer.android.com/reference/android/os/Messenger.html) crea un [IBinder](http://developer.android.com/reference/android/os/IBinder.html) que el servicio devuelve a los clientes de [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) .
* Los clientes usan el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) crear una instancia del [Messenger](http://developer.android.com/reference/android/os/Messenger.html) (que hace referencia al servicio [Handler](http://developer.android.com/reference/android/os/Handler.html) ), lo que el cliente utiliza para enviar [Message](http://developer.android.com/reference/android/os/Message.html) objetos para el servicio.
* El servicio recibe cada [Message](http://developer.android.com/reference/android/os/Message.html) en su [Handler](http://developer.android.com/reference/android/os/Handler.html) , específicamente, en el [handleMessage()](http://developer.android.com/reference/android/os/Handler.html#handleMessage%28android.os.Message%29) método.

De esta manera, no existen "métodos" para que el cliente llame en el servicio. En cambio, el cliente entrega "mensajes" ( [Message](http://developer.android.com/reference/android/os/Message.html) objetos) que el servicio recibe en su [Handler](http://developer.android.com/reference/android/os/Handler.html) .

He aquí un ejemplo de servicio simple que utiliza un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) de la interfaz:

public class MessengerService amplía el servicio de {

/ \*\* Comando al servicio para mostrar un mensaje \* /

static final int MSG\_SAY\_HELLO = 1;

/ \*\*

\* Controlador de mensajes entrantes de los clientes.

\* /

la clase se extiende IncomingHandler Handler {

@Override

public void handleMessage (Mensaje msg) {

switch (msg.what) {

caso MSG\_SAY\_HELLO:

Toast.makeText (getApplicationContext (), "hola!", Pan tostado.. LENGTH\_SHORT) show ();

[ break ; ]

default:

super.handleMessage (msg);

}

}

}

/ \*\*

\* Objetivo que publicamos para que los clientes envían mensajes a IncomingHandler.

\* /

Mensajero final de mMessenger = new Messenger (nueva IncomingHandler ());

/ \*\*

\* Cuando la unión con el servicio, volvemos una interfaz a nuestro mensajero

\* Para el envío de mensajes al servicio.

\* /

@Override

pública IBinder onBind (intención de Intención) {

Toast.makeText (getApplicationContext (), "vinculante", pan tostado.. LENGTH\_SHORT) show ();

volver mMessenger.getBinder ();

}

}

Observe que el [handleMessage()](http://developer.android.com/reference/android/os/Handler.html#handleMessage%28android.os.Message%29) método en el [Handler](http://developer.android.com/reference/android/os/Handler.html) es que el servicio recibe la entrada [Message](http://developer.android.com/reference/android/os/Message.html) y decide qué hacer, basada en el [what](http://developer.android.com/reference/android/os/Message.html#what) los miembros.

Todo lo que un cliente tiene que hacer es crear un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) basado en el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) devuelto por el servicio y enviar un mensaje usando [send()](http://developer.android.com/reference/android/os/Messenger.html#send%28android.os.Message%29) . Por ejemplo, aquí es una actividad simple que enlaza con el servicio y entrega el MSG\_SAY\_HELLO mensaje al servicio de:

pública ActivityMessenger clase extiende la actividad {

/ \*\* Mensajero para comunicarse con el servicio. \* /

Mensajero MSERVICE = null;

/ \*\* Bandera que indica si hemos llamado se unen en el servicio. \* /

boolean mBound;

/ \*\*

\* Clase para interactuar con la interfaz principal del servicio.

\* /

mConnection ServiceConnection privada ServiceConnection = new () {

public void onServiceConnected (className ComponentName, IBinder servicio) {

/ / Se llama cuando la conexión con el servicio ha sido

/ / Se ha establecido, que nos da el objeto que podemos utilizar para

/ / Interactuar con el servicio. Nos estamos comunicando con la

/ / Servicio a través de un mensajero, asi que aquí tenemos un cliente

/ / Representación de que desde el objeto IBinder prima.

MSERVICE = new Messenger (servicio);

mBound = true;

}

public void onServiceDisconnected (className ComponentName) {

/ / Se llama cuando la conexión con el servicio ha sido

/ / Desconecta inesperadamente - es decir, su proceso se estrelló.

MSERVICE = null;

mBound = false;

}

};

public void sayHello (Ver v) {

Si los númerosmBound) return;

/ / Crear y enviar un mensaje al servicio, con un apoyo "que" el valor

Mensaje msg = Message.obtain (null, MessengerService.MSG\_SAY\_HELLO, 0, 0);

try {

mService.send (msg);

} Catch (RemoteException e) {

e.printStackTrace ();

}

}

@Override

protected void onCreate (paquete savedInstanceState) {

super.onCreate (savedInstanceState);

setContentView (R.layout.main);

}

@Override

protected void OnStart () {

super.onStart ();

/ / Enlazar con el servicio

bindService (nuevo intento (esto, MessengerService.class), mConnection,

"Contexto"BIND\_AUTO\_CREATE);

}

@Override

protected void OnStop () {

super.onStop ();

/ / Separar del servicio

if (mBound) {

unbindService (mConnection);

mBound = false;

}

}

}

Tenga en cuenta que este ejemplo no muestra cómo el servicio puede responder al cliente. Si desea que el servicio responda, entonces usted necesita para crear también un [Messenger](http://developer.android.com/reference/android/os/Messenger.html) en el cliente. Luego, cuando el cliente recibe el [onServiceConnected()](http://developer.android.com/reference/android/content/ServiceConnection.html#onServiceConnected%28android.content.ComponentName,%20android.os.IBinder%29) de devolución de llamada, envía un [Message](http://developer.android.com/reference/android/os/Message.html) al servicio que incluye el cliente de [Messenger](http://developer.android.com/reference/android/os/Messenger.html) en el [replyTo](http://developer.android.com/reference/android/os/Message.html#replyTo) parámetro de la [send()](http://developer.android.com/reference/android/os/Messenger.html#send%28android.os.Message%29) método.

Puede ver un ejemplo de cómo proporcionar mensajería bidireccional en el [MessengerService.java](http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/app/MessengerService.html) (servicio) y [MessengerServiceActivities.java](http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/app/MessengerServiceActivities.html) muestras (cliente).

## Enlazar a un servicio

Los componentes de aplicaciones (clientes) pueden unirse a un servicio llamando [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) . El sistema Android a continuación, llama al servicio de [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) , que devuelve un [IBinder](http://developer.android.com/reference/android/os/IBinder.html) para interactuar con el servicio.

La unión es asíncrona. [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) devuelve inmediatamente y no devuelve el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) para el cliente. Para recibir el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) , el cliente debe crear una instancia de [ServiceConnection](http://developer.android.com/reference/android/content/ServiceConnection.html) y pasarlo a [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) . El [ServiceConnection](http://developer.android.com/reference/android/content/ServiceConnection.html) incluye un método de devolución de llamada que el sistema llama a entregar el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) .

**Nota:** Sólo las actividades, servicios y proveedores de contenido pueden unirse a un **servicio-no** se **puede** enlazar a un servicio de un receptor de radiodifusión.

Por lo tanto, para unirse a un servicio de su cliente, usted debe:

1. Implementar [ServiceConnection](http://developer.android.com/reference/android/content/ServiceConnection.html) .

Su implementación debe reemplazar dos métodos de devolución de llamada:

[onServiceConnected()](http://developer.android.com/reference/android/content/ServiceConnection.html#onServiceConnected%28android.content.ComponentName,%20android.os.IBinder%29)

El sistema llama a esto para entregar el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) devuelto por el servicio de [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) método.

[onServiceDisconnected()](http://developer.android.com/reference/android/content/ServiceConnection.html#onServiceDisconnected%28android.content.ComponentName%29)

El sistema Android llama a esto cuando la conexión al servicio se perdió inesperadamente, por ejemplo, cuando el servicio se ha estropeado o ha muerto. Esto no se llama cuando el cliente se desenlaza.

1. Llame [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) , pasando por el [ServiceConnection](http://developer.android.com/reference/android/content/ServiceConnection.html) ejecución.
2. Cuando el sistema llama a su [onServiceConnected()](http://developer.android.com/reference/android/content/ServiceConnection.html#onServiceConnected%28android.content.ComponentName,%20android.os.IBinder%29) el método de devolución de llamada, usted puede empezar a hacer llamadas al servicio, utilizando los métodos definidos por la interfaz.
3. Para desconectarse del servicio, llame a [unbindService()](http://developer.android.com/reference/android/content/Context.html#unbindService%28android.content.ServiceConnection%29) .

Cuando el cliente se destruye, se desvincula del servicio, pero siempre se debe desatar cuando haya terminado de interactuar con el servicio o cuando su actividad se detiene para que el servicio puede apagar cuando no la esté utilizando. (Momento oportuno, a Enlazar y desenlazar se discute más adelante).

Por ejemplo, el siguiente fragmento de código se conecta el cliente al servicio creado anteriormente por la [ampliación de la clase Binder](http://developer.android.com/guide/topics/fundamentals/bound-services.html#Binder) , así que todo lo que tiene que hacer es lanzar el regresó [IBinder](http://developer.android.com/reference/android/os/IBinder.html) a la LocalService clase y pedir a la LocalService ejemplo:

LocalService MSERVICE;

mConnection ServiceConnection privada ServiceConnection = new () {

/ / Se llama cuando la conexión con el servicio se ha establecido

public void onServiceConnected (className ComponentName, IBinder servicio) {

/ / Ya que han obligado a una explícita

/ / Servicio que se ejecuta en nuestro propio proceso, podemos

/ / Para ejercer el derecho IBinder a una clase concreta y directamente acceder a él.

LocalBinder ligante = (LocalBinder) de servicios;

MSERVICE binder.getService = ();

mBound = true;

}

/ / Se llama cuando la conexión con el servicio se desconecta inesperadamente

public void onServiceDisconnected (className ComponentName) {

Log.e (TAG, "onServiceDisconnected");

mBound = false;

}

};

Con este [ServiceConnection](http://developer.android.com/reference/android/content/ServiceConnection.html) , el cliente puede unirse a un servicio al pasar a esta [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) . Por ejemplo:

Intención de Intención = nuevo intento (esto, LocalService.class);

bindService (intención, mConnection, Contexto.BIND\_AUTO\_CREATE);

* El primer parámetro de [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) es una [Intent](http://developer.android.com/reference/android/content/Intent.html) que se nombra explícitamente que el servicio se unen (se cree podría ser la intención implícita).
* El segundo parámetro es el [ServiceConnection](http://developer.android.com/reference/android/content/ServiceConnection.html) objeto.
* El tercer parámetro es un indicador que muestra las opciones para el enlace. Por lo general, debe ser [BIND\_AUTO\_CREATE](http://developer.android.com/reference/android/content/Context.html#BIND_AUTO_CREATE) con el fin de crear el servicio si no es que ya vive. Otros valores posibles son [BIND\_DEBUG\_UNBIND](http://developer.android.com/reference/android/content/Context.html#BIND_DEBUG_UNBIND) y [BIND\_NOT\_FOREGROUND](http://developer.android.com/reference/android/content/Context.html#BIND_NOT_FOREGROUND) , o 0 para ninguno.

### Notas adicionales:

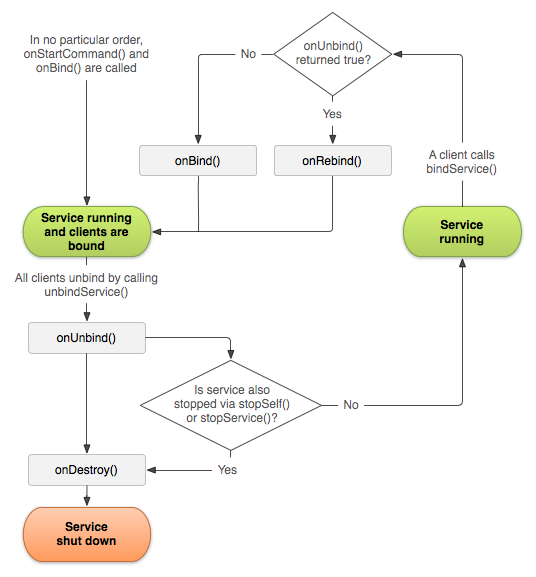
Aquí están algunas notas importantes acerca de la unión a un servicio:

* Usted siempre debe atrapar [DeadObjectException](http://developer.android.com/reference/android/os/DeadObjectException.html) excepciones, que se producen cuando la conexión se ha roto. Esta es la única excepción lanzada por métodos remotos.
* Los objetos son un contador de referencias en todos los procesos.
* Por regla general deberá vincular el atar y desatar en juego poner en marcha y desmontaje momentos del ciclo de vida del cliente. Por ejemplo:
  + Si usted sólo tiene que interactuar con el servicio, mientras que su actividad es visible, se debe vincular en [onStart()](http://developer.android.com/reference/android/app/Activity.html#onStart%28%29) y desenlazar en [onStop()](http://developer.android.com/reference/android/app/Activity.html#onStop%28%29) .
  + Si desea que su actividad para recibir las respuestas, incluso mientras está parado en el fondo, a continuación, puede enlazar en [onCreate()](http://developer.android.com/reference/android/app/Activity.html#onCreate%28android.os.Bundle%29) y desenlazar en [onDestroy()](http://developer.android.com/reference/android/app/Activity.html#onDestroy%28%29) . Tenga en cuenta que esto implica que su actividad tiene que utilizar el servicio durante todo el tiempo que se está ejecutando (aunque en el fondo), así que si el servicio se encuentra en otro proceso, entonces se aumenta el peso del proceso y se hace más probable que el sistema se acabar con él.

**Nota: No se** debe enlazar y desenlazar lo general durante su actividad de [onResume()](http://developer.android.com/reference/android/app/Activity.html#onResume%28%29) y [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause%28%29) , ya que estas devoluciones de llamada se producen en todas las transiciones del ciclo de vida y usted debe mantener el tratamiento que se produce en estas transiciones a un mínimo. Además, si múltiples actividades en su aplicación se unen al mismo servicio y no hay una transición entre dos de esas actividades, el servicio puede ser destruido y recreado como elimina la asignación de actividad actual (durante la pausa) antes de la próxima liga (en resumen). (Esta transición de la actividad como las actividades de coordinación de sus ciclos de vida se describe en la [Actividades de](http://developer.android.com/guide/topics/fundamentals/activities.html#CoordinatingActivities) documento).

Para más código de ejemplo que muestra cómo enlazar a un servicio, consulte el [RemoteService.java](http://developer.android.com/resources/samples/ApiDemos/src/com/example/android/apis/app/RemoteService.html) clase en [ApiDemos](http://developer.android.com/resources/samples/ApiDemos/index.html) .

## Gestión del Ciclo de Vida de un Servicio de Consolidada



**FIGURA I.** El ciclo de vida para un servicio que se inicia y también permite la unión.

Cuando un servicio no está consolidado de todos los clientes, el sistema Android que destruye (a menos que se inició también con [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) ). Como tal, usted no tiene que gestionar el ciclo de vida de su servicio si es puramente un servicio-la cota sistema Android que gestiona para usted en función de si se ve obligada a todos los clientes.

Sin embargo, si usted decide poner en práctica el [onStartCommand()](http://developer.android.com/reference/android/app/Service.html#onStartCommand%28android.content.Intent,%20int,%20int%29) el método de devolución de llamada, a continuación, de forma explícita, debe detener el servicio, porque el servicio se considera que debe iniciarse. En este caso, el servicio se ejecuta hasta que el servicio se detiene con [stopSelf()](http://developer.android.com/reference/android/app/Service.html#stopSelf%28%29) u otro componente llama [stopService()](http://developer.android.com/reference/android/content/Context.html#stopService%28android.content.Intent%29) , independientemente de si está destinado a todos los clientes.

Además, si el servicio se inicia y acepta vinculante, a continuación, cuando el sistema llama a su [onUnbind()](http://developer.android.com/reference/android/app/Service.html#onUnbind%28android.content.Intent%29) método, si lo desea, puede devolver true si a usted le gustaría recibir una llamada a [onRebind()](http://developer.android.com/reference/android/app/Service.html#onRebind%28android.content.Intent%29) la próxima vez que un cliente se une al servicio (en lugar de recibir una llamada a [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) ). [onRebind()](http://developer.android.com/reference/android/app/Service.html#onRebind%28android.content.Intent%29) devuelve void, pero el cliente todavía recibe el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) en su [onServiceConnected()](http://developer.android.com/reference/android/content/ServiceConnection.html#onServiceConnected%28android.content.ComponentName,%20android.os.IBinder%29) de devolución de llamada. A continuación, la figura 1 ilustra la lógica de este tipo de ciclo de vida.

Para más información sobre el ciclo de vida de un servicio de empezar, consulte el [Servicio](http://developer.android.com/guide/topics/fundamentals/services.html#Lifecycle) documento.

[↑ Ir arriba](http://developer.android.com/guide/topics/fundamentals/bound-services.html#top)

[← Volver a Servicios](http://developer.android.com/guide/topics/fundamentals/services.html)

A excepción de lo señalado, este contenido está disponible bajo [Apache 2.0](http://www.google.com/url?q=http%3A%2F%2Fwww.apache.org%2Flicenses%2FLICENSE-2.0&sa=D&sntz=1&usg=AFQjCNFPaSHdvCi6VD7GwgEkKspvD3d_tw) . Para más detalles y restricciones, consulte la [Licencia de Contenido](http://developer.android.com/license.html) .

Android 4.0 r1 - 01 de mayo 2012 17:37