**Intenciones y filtros de Intención**

**En este documento**

1. [Objetos Intent](http://developer.android.com/guide/topics/intents/intents-filters.html#iobjs)
2. [Intención de Resolución](http://developer.android.com/guide/topics/intents/intents-filters.html#ires)
3. [Filtros de Intención](http://developer.android.com/guide/topics/intents/intents-filters.html#ifs)
4. [Los casos más comunes](http://developer.android.com/guide/topics/intents/intents-filters.html#ccases)
5. [Utilizando la intención coincidente](http://developer.android.com/guide/topics/intents/intents-filters.html#imatch)
6. [Nota Ejemplo Pad](http://developer.android.com/guide/topics/intents/intents-filters.html#npex)

**Clases de clave**

1. [Intent](http://developer.android.com/reference/android/content/Intent.html)
2. [IntentFilter](http://developer.android.com/reference/android/content/IntentFilter.html)
3. [BroadcastReceiver](http://developer.android.com/reference/android/content/BroadcastReceiver.html)
4. [PackageManager](http://developer.android.com/reference/android/content/pm/PackageManager.html)

Tres de los componentes básicos de una aplicación - las actividades, servicios y receptores de radio - se activan a través de mensajes, llamada *intenciones.* Intención de mensajería es una instalación para finales de tiempo de ejecución de la unión entre los componentes de las mismas o diferentes aplicaciones. La misma intención, una [Intent](http://developer.android.com/reference/android/content/Intent.html) objeto, es una estructura de datos pasiva la celebración de una descripción abstracta de una operación para llevar a cabo - o, a menudo en el caso de las emisiones, una descripción de algo que ha sucedido y anunció que se. Hay mecanismos separados para la entrega de los intentos para cada tipo de componente:

* Un objeto se pasa a la intención [Context.startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity%28android.content.Intent%29) o [Activity.startActivityForResult()](http://developer.android.com/reference/android/app/Activity.html#startActivityForResult%28android.content.Intent,%20int%29) para poner en marcha una actividad o conseguir una actividad existente de hacer algo nuevo. (También se puede pasar a [Activity.setResult()](http://developer.android.com/reference/android/app/Activity.html#setResult%28int,%20android.content.Intent%29) para devolver información a la actividad que se llama startActivityForResult() .)
* Un objeto se pasa a la intención [Context.startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) para iniciar un servicio o entregar nuevas instrucciones a un servicio continuo. De manera similar, una intención se puede pasar a [Context.bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) para establecer una conexión entre el componente de llamada y un servicio de destino. Opcionalmente, puede iniciar el servicio si no está ya en ejecución.
* Intent objetos pasa a cualquiera de los métodos de difusión (como [Context.sendBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendBroadcast%28android.content.Intent%29) , [Context.sendOrderedBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendOrderedBroadcast%28android.content.Intent,%20java.lang.String%29) , o [Context.sendStickyBroadcast()](http://developer.android.com/reference/android/content/Context.html#sendStickyBroadcast%28android.content.Intent%29) ) se entregan a todos los receptores de radiodifusión interesadas. Muchos tipos de emisiones se originan en el código del sistema.

En cada caso, el sistema Android encuentra la actividad apropiada, servicio o conjunto de receptores de radiodifusión para responder a la intención, instanciarlos si es necesario. No hay superposición dentro de estos sistemas de mensajería: los intentos de difusión se entregarán solamente a los receptores de difusión, nunca a las actividades o servicios. Un intento de pasar a startActivity() se entrega sólo a una actividad, nunca a un receptor del servicio o de difusión, y así sucesivamente.

Este documento comienza con una descripción de los objetos Intent. A continuación, se describen las reglas para Android utiliza para asignar las intenciones de los componentes - la forma en que se resuelve el componente que debe recibir un mensaje de intención. Para propósitos que no nombran explícitamente un componente de destino, este proceso consiste en comprobar el objeto Intención contra la *intención filtros* asociados con los objetivos potenciales.

**Objetos Intent**

Una [Intent](http://developer.android.com/reference/android/content/Intent.html) objeto es un conjunto de información. Contiene información de interés para el componente que recibe la intención (como la adopción de medidas y los datos para actuar en), además de información de interés para el sistema Android (por ejemplo, la categoría de componente que debe manejar la intención y las instrucciones sobre cómo poner en marcha una actividad de destino). Principalmente, se puede contener lo siguiente:

**Nombre del Componente**

El nombre del componente que debe manejar el intento. Este campo es un [ComponentName](http://developer.android.com/reference/android/content/ComponentName.html) objeto - una combinación del nombre de clase completo del componente de destino (por ejemplo, " com.example.project.app.FreneticActivity ") y el nombre del paquete establecido en el archivo de manifiesto de la aplicación en la que reside el componente (por ejemplo," com.example.project "). La parte del paquete del nombre del componente y el nombre del paquete establecido en el manifiesto no necesariamente tienen que coincidir.

El nombre del componente es opcional. Si se establece, el objeto de Intención se entrega a una instancia de la clase designada. Si no se establece, Android utiliza otro tipo de información en el objeto de la intención de localizar un blanco adecuado - ver [Resolución intención](http://developer.android.com/guide/topics/intents/intents-filters.html#ires) , más adelante en este documento.

El nombre del componente se ajusta [setComponent()](http://developer.android.com/reference/android/content/Intent.html#setComponent%28android.content.ComponentName%29) , setClass() , o [setClassName()](http://developer.android.com/reference/android/content/Intent.html#setClassName%28java.lang.String,%20java.lang.String%29) y leído por [getComponent()](http://developer.android.com/reference/android/content/Intent.html#getComponent%28%29) .

**acciones**

Una cadena con el nombre de la acción a realizar - o, en el caso de los intentos de difusión, la acción que llevó a cabo y se está informando. La clase de Intención define una serie de constantes acciones, incluyendo los siguientes:

|  |  |  |
| --- | --- | --- |
| **Constante** | **Objetivo de componentes** | **acciones** |
| ACTION\_CALL  | actividad  | Iniciar una llamada telefónica.  |
| ACTION\_EDIT  | actividad  | Visualización de los datos para el usuario que desea editar.  |
| ACTION\_MAIN  | actividad  | Puesta en marcha como la actividad inicial de una tarea, con la entrada de datos que no y no hay salida de regresar.  |
| ACTION\_SYNC  | actividad  | Sincronización de datos en un servidor con los datos en el dispositivo móvil.  |
| ACTION\_BATTERY\_LOW  | difusión del receptor  | Una advertencia de que la batería está baja.  |
| ACTION\_HEADSET\_PLUG  | difusión del receptor  | Un auricular se ha conectado en el dispositivo, o se desconecta de ella.  |
| ACTION\_SCREEN\_ON  | difusión del receptor  | La pantalla se ha encendido.  |
| ACTION\_TIMEZONE\_CHANGED  | difusión del receptor  | La configuración de la zona horaria ha cambiado.  |

Ver la [Intent](http://developer.android.com/reference/android/content/Intent.html) descripción de la clase para una lista de constantes predefinidas para acciones genéricas. Otras acciones se definen en otras partes de la API de Android. Usted también puede definir sus propias cadenas de acción para la activación de los componentes en su aplicación. Los que inventan debe incluir el paquete de aplicaciones como prefijo - por ejemplo: " com.example.project.SHOW\_COLOR ".

La acción determina en gran medida de cómo el resto de la intención está estructurada - en particular los [datos](http://developer.android.com/guide/topics/intents/intents-filters.html#data) y [extras](http://developer.android.com/guide/topics/intents/intents-filters.html#extras) campos - tanto como un nombre de método determina un conjunto de argumentos y un valor de retorno. Por esta razón, es una buena idea utilizar nombres de las acciones que sean lo más específico posible, y para acoplarlas con fuerza a los otros campos de la intención. En otras palabras, en lugar de definir una acción de forma aislada, definir un protocolo completo de los objetos Intent sus componentes pueden manejar.

La acción en un objeto Intención es fijado por el [setAction()](http://developer.android.com/reference/android/content/Intent.html#setAction%28java.lang.String%29) método y leído por [getAction()](http://developer.android.com/reference/android/content/Intent.html#getAction%28%29) .

**'DATA'**

El URI de los datos para que se actúe sobre y el tipo MIME de los datos. Las diferentes acciones se combinan con diferentes tipos de especificaciones de los datos. Por ejemplo, si el campo de acción es ACTION\_EDIT , el campo de datos que contiene el URI del documento que se mostrará para su edición. Si la acción es ACTION\_CALL , el campo de datos sería un tel: URI con el número para llamar. Del mismo modo, si la acción es ACTION\_VIEW y el campo de datos es un http: URI, la actividad que reciba serían llamados para descargar y mostrar todos los datos que el URI se refiere.

Cuando se especifique la intención de un componente que es capaz de manejar los datos, a menudo es importante saber el tipo de datos (su tipo MIME), además de su URI. Por ejemplo, un componente capaz de mostrar datos de la imagen no debe ser llamada a desempeñar un archivo de audio.

En muchos casos, el tipo de datos se puede deducir de la URI - sobre todo content: URI, que indican que se encuentran los datos en el dispositivo y controlada por un proveedor de contenido (ver la [discusión por separado de los proveedores de contenido](http://developer.android.com/guide/topics/providers/content-providers.html) ). Sin embargo, el tipo también se puede establecer explícitamente en el objeto de Intención. El [setData()](http://developer.android.com/reference/android/content/Intent.html#setData%28android.net.Uri%29) método especifica los datos sólo como un URI, [setType()](http://developer.android.com/reference/android/content/Intent.html#setType%28java.lang.String%29) que especifica sólo como un tipo MIME, y [setDataAndType()](http://developer.android.com/reference/android/content/Intent.html#setDataAndType%28android.net.Uri,%20java.lang.String%29) que especifica tanto como una URI y un tipo MIME. El URI es leído por [getData()](http://developer.android.com/reference/android/content/Intent.html#getData%28%29) y el tipo de [getType()](http://developer.android.com/reference/android/content/Intent.html#getType%28%29) .

**Categoría**

Una cadena que contiene información adicional sobre el tipo de componente que debe manejar el intento. Cualquier número de descripciones de la categoría se puede colocar en un objeto Intención. Al igual que para las acciones, la clase de Intención define varias constantes de categoría, incluyendo los siguientes:

|  |  |
| --- | --- |
| **Constante** | **Sentido** |
| CATEGORY\_BROWSABLE  | La actividad de destino puede ser con seguridad invocada por el navegador para mostrar los datos referenciados por un enlace, por ejemplo, una imagen o un mensaje de correo electrónico.  |
| CATEGORY\_GADGET  | La actividad puede ser embebido dentro de otra actividad que organiza los gadgets.  |
| CATEGORY\_HOME  | La actividad muestra la pantalla de inicio, la primera pantalla que ve el usuario cuando el dispositivo está encendido o cuando el botón se pulsa *Inicio.*  |
| CATEGORY\_LAUNCHER  | La actividad puede ser la actividad inicial de una tarea y se enumeran en el lanzador de aplicaciones de nivel superior.  |
| CATEGORY\_PREFERENCE  | La actividad de destino es un panel de preferencias.  |

Ver la [Intent](http://developer.android.com/reference/android/content/Intent.html) descripción de la clase para la lista completa de categorías.

El [addCategory()](http://developer.android.com/reference/android/content/Intent.html#addCategory%28java.lang.String%29) método coloca en una categoría de un objeto, la intención [removeCategory()](http://developer.android.com/reference/android/content/Intent.html#removeCategory%28java.lang.String%29) elimina una categoría previamente agregado, y [getCategories()](http://developer.android.com/reference/android/content/Intent.html#getCategories%28%29) obtiene el conjunto de todas las categorías actualmente en el objeto.

**Extras**

Pares clave-valor para la información adicional que debe ser entregado a la componente de manejo de la intención. Al igual que algunas acciones se combinan con determinados tipos de URIs de datos, algunos están vinculados con los suplementos particulares. Por ejemplo, un ACTION\_TIMEZONE\_CHANGED la intención tiene un " time-zone "extra que identifica a la nueva zona horaria, y ACTION\_HEADSET\_PLUG tiene un " state "extra que indica si el auricular ya está conectado o desconectado, así como un" name "extra para el tipo de auriculares. Si se va a inventar un SHOW\_COLOR acción, el valor del color se encuentra en un extra par clave-valor.

El objeto de Intención tiene una serie de put...() los métodos para la inserción de diversos tipos de datos adicionales y un conjunto similar de get...() los métodos para leer los datos. Estos métodos son paralelos a los de [Bundle](http://developer.android.com/reference/android/os/Bundle.html) objetos. De hecho, los extras se puede instalar y leer como un paquete mediante el [putExtras()](http://developer.android.com/reference/android/content/Intent.html#putExtras%28android.content.Intent%29) y [getExtras()](http://developer.android.com/reference/android/content/Intent.html#getExtras%28%29) métodos.

**Banderas**

Banderas de distintos tipos. Muchos instruyen al sistema Android de cómo iniciar una actividad (por ejemplo, que la tarea de la actividad debe pertenecer a) y cómo tratarla después de que se puso en marcha (por ejemplo, si pertenece a la lista de las actividades recientes). Todos estos indicadores se definen en la clase de intenciones.

El sistema Android y las aplicaciones que vienen con la plataforma de utilizar objetos Intent tanto para enviar por el sistema se originaron las emisiones y para activar los componentes definidos por el sistema. Para ver la forma de estructurar la intención de activar un componente del sistema, consulte la [lista de las intenciones](http://developer.android.com/guide/appendix/g-app-intents.html) de la referencia.

**Intención de Resolución**

Intentos se puede dividir en dos grupos:

* *Las intenciones explícitas de* designar el componente de destino por su nombre (el [campo de nombre del componente](http://developer.android.com/guide/topics/intents/intents-filters.html#cname) , mencionado anteriormente, tiene un conjunto de valores). Dado que los nombres de los componentes que generalmente no se conoce a los desarrolladores de otras aplicaciones, las intenciones explícitas se utilizan normalmente para la aplicación de mensajes internos - como una actividad de inicio de un servicio subordinado o el lanzamiento de una actividad hermana.
* *Intenciones implícitas* no nombrar a un objetivo (el campo para el nombre del componente está en blanco). Intentos implícitos se utilizan a menudo para activar componentes en otras aplicaciones.

Android ofrece una intención explícita de una instancia de la clase de destino designado. Nada de lo dispuesto en el objeto que no sea la intención de los asuntos de los componentes de nombres para determinar qué componentes deben tener la intención.

Una estrategia diferente es necesario para las intenciones implícitas. En ausencia de un objetivo designado, el sistema Android tiene que encontrar el mejor componente (o componentes) para manejar la intención - una sola actividad o servicio para realizar la acción requerida, o la serie de los receptores de radiodifusión para responder a la convocatoria de difusión. Lo hace comparando el contenido del objeto intención de *filtros intención,* estructuras asociadas con los componentes que potencialmente pueden recibir intentos. Filtros de publicidad de las capacidades de un componente y delimitar las intenciones que puede manejar. Abren el componente de la posibilidad de recibir las intenciones implícitas del tipo anunciado. Si un componente no tiene la intención de los filtros, puede recibir sólo las intenciones explícitas. Uno de los componentes con los filtros pueden recibir ambos intentos explícitos e implícitos.

Sólo tres de los aspectos de un objeto Intención son consultados cuando el objeto se compara con un filtro de intención:

acciones
los datos (tanto tipo de URI y datos)
categoría

Los suplementos y las banderas no desempeñan ningún papel en la resolución el componente que recibe una intención.

**Filtros de Intención**

Para informar al sistema que las intenciones implícitas que pueden manejar, actividades, servicios y receptores de radiodifusión puede tener uno o más filtros de intención. Cada filtro se describe una capacidad del componente, un conjunto de intentos que el componente está dispuesto a recibir. Es, en efecto, los filtros en las intenciones de un tipo deseado, filtrando los intentos no deseados-, pero las intenciones implícitas sólo no deseados (los que no nombra una clase de destino). Un intento explícito se entrega siempre a su objetivo, no importa lo que contiene, el filtro no es consultado. Pero una intención implícita es entregado a un componente sólo si puede pasar a través de uno de los filtros del componente.

Un componente dispone de filtros independientes para cada puesto de trabajo que puede hacer, cada cara se puede presentar al usuario. Por ejemplo, la actividad NoteEditor de la aplicación de ejemplo de Notas tiene dos filtros, uno para la puesta en marcha con una nota específica que el usuario puede ver o editar, y otro para comenzar con una nueva nota en blanco que el usuario puede rellenar y guardar . (Todos los filtros Pad Nota se describen en la [Nota Ejemplo almohadilla](http://developer.android.com/guide/topics/intents/intents-filters.html#npex) sección, más adelante.)

**Los filtros y de seguridad**

Un filtro de intención no puede ser invocado por la seguridad. Si bien se abre un componente de recibir sólo cierto tipo de intenciones implícitas, no hace nada para prevenir los intentos explícitos de la focalización del componente. A pesar de que un filtro restringe las intenciones se pidió un componente para manejar a ciertas acciones y fuentes de datos, siempre hay alguien que podía armar una intención explícita, con una acción diferente y una fuente de datos, y el nombre del componente como el objetivo.

Un filtro de intención es una instancia de la [IntentFilter](http://developer.android.com/reference/android/content/IntentFilter.html) clase. Sin embargo, dado que el sistema Android tiene que saber acerca de las capacidades de un componente antes de que pueda poner en marcha este componente, los filtros de la intención general no se estableció en el código de Java, pero en el archivo de manifiesto de la aplicación (AndroidManifest.xml) como [<intent-filter>](http://developer.android.com/guide/topics/manifest/intent-filter-element.html) elementos . (La única excepción sería filtros para receptores de radiodifusión que se registran dinámicamente llamando [Context.registerReceiver()](http://developer.android.com/reference/android/content/Context.html#registerReceiver%28android.content.BroadcastReceiver,%20android.content.IntentFilter,%20java.lang.String,%20android.os.Handler%29) , sino que están creados directamente como objetos IntentFilter.)

Un filtro tiene campos paralelos a la acción, los datos y los campos de la categoría de un objeto Intención. La intención implícita se compara con el filtro en las tres áreas. Para ser entregado al componente que posee el filtro, debe pasar las tres pruebas. Si falla uno solo de ellos, el sistema Android no lo entregará al componente - al menos no sobre la base de ese filtro. Sin embargo, puesto que un componente puede tener varios filtros de intención, la intención de que no pasa por uno de los filtros de un componente puede hacerlo a través de otro.

Cada una de las tres pruebas se describe en detalle a continuación:

**Acción de la prueba**

Un [<intent-filter>](http://developer.android.com/guide/topics/manifest/intent-filter-element.html) elemento en el archivo de manifiesto enumera las acciones que [<action>](http://developer.android.com/guide/topics/manifest/action-element.html) subelementos. Por ejemplo:

<Intención de filtro. . . >

 <Action: name = "com.example.project.SHOW\_CURRENT "/>

 <Action: name = "com.example.project.SHOW\_RECENT "/>

 <Action: name = "com.example.project.SHOW\_PENDING "/>

 . . .

</ Intención de filtro>

Como muestra el ejemplo, mientras que un objeto Intención nombres simplemente una sola acción, un filtro puede mostrar más de uno. La lista no puede estar vacío, un filtro debe contener al menos un <action> elemento, o que bloqueará todos los intentos.

Para pasar esta prueba, la acción especificada en el objeto de Intención debe coincidir con una de las acciones enumeradas en el filtro. Si el objeto o el filtro no especifica una acción, los resultados son como sigue:

* Si el filtro falla a la lista de las acciones, no hay nada que la intención de hacer coincidir, por lo que todos los intentos fallan la prueba. No hay intenciones puede conseguir a través del filtro.
* Por otro lado, un objeto Intención que no especifica una acción automáticamente pasa la prueba - siempre y cuando el filtro contiene al menos una acción.

**Categoría de prueba**

Un [<intent-filter>](http://developer.android.com/guide/topics/manifest/intent-filter-element.html) elemento también se enumeran las categorías como subelementos. Por ejemplo:

<Intención de filtro. . . >

 <Categoría de android: name = "android.intent.category.MORA.

 <Categoría de android: name = "android.intent.category.Navegable "/>

 . . .

</ Intención de filtro>

Tenga en cuenta que las constantes descritas anteriormente para las acciones y las categorías no se utilizan en el archivo de manifiesto. Los valores de la cadena completa se utilizan en su lugar. Por ejemplo, el " android.intent.category.BROWSABLE "cadena en el ejemplo anterior corresponde a la CATEGORY\_BROWSABLE constante se mencionó anteriormente en este documento. Del mismo modo, la cadena " android.intent.action.EDIT "corresponde a la ACTION\_EDIT constante.

Para un intento de pasar la prueba de categoría, todas las categorías en el objeto de Intención debe coincidir con una categoría en el filtro. El filtro puede incluir categorías adicionales, pero no se puede omitir cualquiera que esté en el intento.

En principio, por lo tanto, un objeto Intención con ninguna categoría siempre debe pasar esta prueba, independientemente de lo que está en el filtro. Eso es todo cierto. Sin embargo, con una excepción, Android trata todas las intenciones implícitas pasan a [startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity%28android.content.Intent%29) como si contuvieran al menos una categoría: " android.intent.category.DEFAULT "(el CATEGORY\_DEFAULT constante). Por lo tanto, las actividades que están dispuestos a recibir las intenciones implícitas debe incluir " android.intent.category.DEFAULT "en los filtros de sus intenciones. (Filtros con " android.intent.action.MAIN "y" android.intent.category.LAUNCHER "ajustes son la excepción. Ellos marcan las actividades que se inician nuevas tareas y que están representados en la pantalla de inicio. Pueden incluir " android.intent.category.DEFAULT "en la lista de categorías, pero no es necesario.) Ver [Uso de coincidencia de intenciones](http://developer.android.com/guide/topics/intents/intents-filters.html#imatch) , más tarde, para más información sobre estos filtros.)

**Los datos de prueba**

Al igual que la acción y categorías, la especificación de datos para un filtro intención está contenida en un subelemento. Y, como en esos casos, el subelemento puede aparecer varias veces, o nada en absoluto. Por ejemplo:

<Intención de filtro. . . >

 <Datos de Android: mimeType = "video / mpeg" android: scheme = "http". . . />

 <Datos de Android: mimeType = "audio / mpeg" android: scheme = "http". . . />

 . . .

</ Intención de filtro>

Cada [<data>](http://developer.android.com/guide/topics/manifest/data-element.html) elemento puede especificar un URI y un tipo de datos (tipo de contenido MIME). Hay atributos separados, scheme , host , port , y la path - para cada parte de la URI:

scheme://host:port/path

Por ejemplo, en la siguiente URI,

content://com.example.project:200/folder/subfolder/etc

el esquema es " content ", el anfitrión es" com.example.project ", el puerto es" 200 ", y el camino es" folder/subfolder/etc ". El host y el puerto en conjunto constituyen la *autoridad* URI, si un host no se especifica, el puerto se ignora.

Cada uno de estos atributos es opcional, pero no son independientes entre sí: para que una autoridad que tenga sentido, un sistema también debe ser especificado. Por un camino que tenga sentido, tanto en el régimen y la autoridad debe ser especificado.

Cuando el URI en un objeto Intención se compara con una especificación URI en un filtro, se compararon sólo a las partes de la URI realmente mencionadas en el filtro. Por ejemplo, si un filtro especifica sólo un esquema, todos los URI con dicho régimen coincide con el filtro. Si un filtro especifica un esquema y una autoridad, pero no hay camino, todos los URI con el mismo esquema y el partido de la autoridad, independientemente de sus caminos. Si un filtro especifica un esquema, una autoridad, y un camino, sólo URI con el mismo esquema, la autoridad, y el partido camino. Sin embargo, una especificación de ruta en el filtro puede contener comodines para requerir sólo una coincidencia parcial de la ruta.

El type atributo de un <data> elemento especifica el tipo MIME de los datos. Es más común en los filtros que un URI. Tanto el objeto de Intención y el filtro se puede utilizar un comodín "\*" para el campo de subtipo - por ejemplo, " text/\* "o" audio/\* "- indica los partidos subtipo.

La prueba de los datos compara el URI y el tipo de datos en el objeto de Intención para un tipo de URI y los datos especificados en el filtro. Las reglas son como sigue:

1. Un objeto Intención que no contiene ni un URI, ni un tipo de datos pasa la prueba si el filtro del mismo modo no se especifica ningún tipo URI o de datos.
2. Un objeto Intención que contiene un URI, pero ningún tipo de datos (y un tipo que no se puede deducir de la URI) pasa la prueba si su URI coincide con una URI en el filtro y el filtro del mismo modo no se especifique el tipo. Este será el caso sólo para los URIs como mailto: y el tel: que no se refieren a los datos reales.
3. Un objeto Intención que contiene un tipo de datos pero no es un URI pasa la prueba si el filtro muestra el mismo tipo de datos y de manera similar no especifica un URI.
4. Un objeto Intención que contiene un URI y un tipo de datos (o un tipo de datos se puede deducir de la URI) pasa a la parte de tipo de datos de la prueba sólo si su tipo concuerda con un tipo que figuran en el filtro. Pasa a la parte URI de la prueba, ya sea si su URI coincide con una URI en el filtro o si tiene un content: o file: URI y el filtro no se especifica un URI. En otras palabras, un componente se presume que apoyar content: y file: datos si sus listas de filtros sólo un tipo de datos.

Si la intención puede pasar a través de los filtros de más de una actividad o servicio, el usuario puede pedir el componente para activar. Se produce una excepción si no se meta se puede encontrar.

**Los casos más comunes**

La última regla de arriba para la prueba de los datos, la regla (d), refleja la expectativa de que los componentes son capaces de obtener datos locales de un archivo o un proveedor de contenido. Por lo tanto, los filtros se pueden enumerar sólo un tipo de datos y no es necesario mencionar explícitamente el content: y el file: esquemas. Este es un caso típico. Un <data> elemento como el siguiente, por ejemplo, dice que el componente de Android puede obtener datos de la imagen de un proveedor de contenido y mostrarlo:

datos> android:mimeType="image/\*" />

Como la mayoría de los datos disponibles se distribuye los proveedores de contenido, filtros que especifican un tipo de datos, pero no un URI son, quizás, el más común.

Otra configuración común son los filtros con un esquema y un tipo de datos. Por ejemplo, un <data> elemento como el siguiente Android dice que el componente puede obtener los datos de vídeo de la red y mostrar que:

datos> android:scheme="http" android:type="video/\*" />

Consideremos, por ejemplo, lo que hace la aplicación del navegador cuando el usuario sigue un enlace en una página web. En primer lugar, trata de mostrar los datos (como podría si el enlace era una página HTML). Si no puede visualizar los datos, que reúne una intención implícita con el tipo de sistema y de datos y trata de iniciar una actividad que puede hacer el trabajo. Si no hay compradores, se pregunta el gestor de descargas para descargar los datos. Esto se pone bajo el control de un proveedor de contenido, por lo que una piscina potencialmente más amplio de actividades (aquellos con filtros que sólo nombrar un tipo de datos) puede responder.

La mayoría de las aplicaciones también tienen una forma de empezar de cero, sin referencia a ningún dato en particular. Actividades que pueden iniciar las aplicaciones tienen filtros con " android.intent.action.MAIN "definida como la acción. Si van a estar representados en el lanzador de aplicaciones, que también se especifica el " android.intent.category.LAUNCHER "la categoría:

<Intención de filtro. . . >

 <Action: name = "android.intent.action código.PRINCIPAL

 <Categoría de android: name = "Código android.intent.category.LANZADOR "/>

</ Intención de filtro>

**Utilizando la intención coincidente**

Intenciones se comparan con filtros de la intención no sólo de descubrir un componente de destino para activar, pero también para descubrir algo sobre el conjunto de componentes en el dispositivo. Por ejemplo, el sistema Android rellena el lanzador de aplicaciones, la pantalla de nivel superior que muestra las aplicaciones que están disponibles para el usuario para poner en marcha, mediante la búsqueda de todas las actividades de los filtros con la intención que se especifica el " android.intent.action.MAIN "la acción y la" android.intent.category.LAUNCHER "categoría (como se ilustra en la sección anterior). A continuación, muestra los iconos y las etiquetas de esas actividades en el lanzador. Del mismo modo, se descubre la pantalla de inicio mediante la búsqueda de la actividad con " android.intent.category.HOME "en su filtro.

La aplicación puede utilizar coincidencia de intención es una manera similar. El [PackageManager](http://developer.android.com/reference/android/content/pm/PackageManager.html) tiene un conjunto de query...() los métodos que devuelven todos los componentes que pueden aceptar una intención particular, y una serie similar de resolve...() métodos que determinan el mejor componente para responder a una intención. Por ejemplo, [queryIntentActivities()](http://developer.android.com/reference/android/content/pm/PackageManager.html#queryIntentActivities%28android.content.Intent,%20int%29) devuelve una lista de todas las actividades que pueden llevar a cabo la intención de pasar como argumento, y [queryIntentServices()](http://developer.android.com/reference/android/content/pm/PackageManager.html#queryIntentServices%28android.content.Intent,%20int%29) devuelve una lista similar de servicios. Ninguno de estos métodos se activan los componentes, sino que simplemente una lista de los que pueden responder. No hay un método similar, [queryBroadcastReceivers()](http://developer.android.com/reference/android/content/pm/PackageManager.html#queryBroadcastReceivers%28android.content.Intent,%20int%29) , para los receptores de radiodifusión.

**Nota Ejemplo Pad**

El Bloc de Notas aplicación de ejemplo permite a los usuarios navegar a través de una lista de notas, ver los detalles de las distintas partidas de la lista, editar los artículos, y añadir un nuevo elemento a la lista. Esta sección trata de los filtros de la intención declarada en el archivo de manifiesto. (Si usted está trabajando fuera de línea en el SDK, puede encontrar todos los archivos de código fuente de esta aplicación de ejemplo, incluyendo su archivo de manifiesto, en <sdk>/samples/NotePad/index.html . Si usted está viendo la documentación en línea, los archivos de origen están en los [Tutoriales y código de ejemplo de](http://developer.android.com/resources/samples/index.html) la sección [aquí](http://developer.android.com/resources/samples/NotePad/index.html) .)

En su archivo de manifiesto, la aplicación Notas declara tres actividades, cada una con filtro de la intención al menos uno. También declara un proveedor de contenidos que gestiona los datos de la nota. Aquí está el archivo de manifiesto en su totalidad:

Menores xmlns: android = manifiestos "http://schemas.android.com/apk/res/android~~HEAD=NNS"

 paquete = "com.example.android.notepad">

 <Aplicaciones android: icon = "@ dibujable o app\_notes"

 android: label = "@ cadena / app\_name">

 <Proveedor de android: name = "NotePadProvider"

 android: las autoridades = "com.google.provider.Bloc de notas "/>

 <activity android:name="NotesList" android:label="@string/title\_notes\_list">

 <intent-filter>

 <Action: name = "android.intent.action.PRINCIPAL

 <Categoría de android: name = "android.intent.category.LANZADOR "/>

 </ Intención de filtro>

 <intent-filter>

 <Action: name = "android.intent.action."Vistas";

 <Action: name = "android.intent.action.EDITAR

 <Action: name = "android.intent.action.Escoger...

 <Categoría de android: name = "android.intent.category.MORA.

 datos> android:mimeType="vnd.android.cursor.dir/vnd.google.note" />

 </ Intención de filtro>

 <intent-filter>

 <Action: name = "android.intent.action.Get\_content "/>

 <Categoría de android: name = "android.intent.category.MORA.

 datos> android:mimeType="vnd.android.cursor.item/vnd.google.note" />

 </ Intención de filtro>

 </ Actividad>

 <Actividad de android: name = "NoteEditor"

 Android: tema = "@ android: estilo / tema.LUZ

 android: label = "@ cadena / title\_note">

 <intent-filter android:label="@string/resolve\_edit">

 <Action: name = "android.intent.action."Vistas";

 <Action: name = "android.intent.action.EDITAR

 <Action: name = "com.android.notepad.action.[ editar ] Notas

 <Categoría de android: name = "android.intent.category.MORA.

 datos> android:mimeType="vnd.android.cursor.item/vnd.google.note" />

 </ Intención de filtro>

 <intent-filter>

 <Action: name = "android.intent.action.INSERT

 <Categoría de android: name = "android.intent.category.MORA.

 datos> android:mimeType="vnd.android.cursor.dir/vnd.google.note" />

 </ Intención de filtro>

 </ Actividad>

 <Actividad de android: name = "TitleEditor"

 android: label = "@ cadena / title\_edit\_title"

 Android: tema = "@ android: estilo / tema.Diálogo

 <intent-filter android:label="@string/resolve\_title">

 <Action: name = "com.android.notepad.action.EDIT\_TITLE "/>

 <Categoría de android: name = "android.intent.category.MORA.

 <Categoría de android: name = "android.intent.category.ALTERNATIVA "/>

 <Categoría de android: name = "android.intent.category.SELECTED\_ALTERNATIVE "/>

 datos> android:mimeType="vnd.android.cursor.item/vnd.google.note" />

 </ Intención de filtro>

 </ Actividad>

 </ Application>

</manifest>

La primera actividad, NotesList, se distingue de las otras actividades por el hecho de que opera en un directorio de las notas (la lista de notas) en lugar de una sola nota. Por lo general, serviría como la interfaz de usuario inicial en la aplicación. Se puede hacer tres cosas tal como se describe por sus tres filtros de la intención:

1. <intent-filter>
2. <Action: name = "android.intent.action.PRINCIPAL
3. <Categoría de android: name = "android.intent.category.LANZADOR "/>
4. </ Intención de filtro>

Este filtro se declara el punto de entrada principal en la aplicación Notas. La norma MAIN acción es un punto de entrada que no requiere de ningún otro tipo de información en el intento (sin especificación de datos, por ejemplo), y el LAUNCHER categoría dice que este punto de entrada deben ser listados en el lanzador de aplicaciones.

1. <intent-filter>
2. <Action: name = "android.intent.action."Vistas";
3. <Action: name = "android.intent.action.EDITAR
4. <Action: name = "android.intent.action.Escoger...
5. <Categoría de android: name = "android.intent.category.MORA.
6. datos> android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
7. </ Intención de filtro>

Este filtro declara las cosas que la actividad se puede hacer en un directorio de las notas. Puede permitir al usuario ver o editar el directorio (a través de la VIEW y EDIT acciones), o para recoger una nota en particular desde el directorio (a través de la PICK acción).

El mimeType atributo de la [<data>](http://developer.android.com/guide/topics/manifest/data-element.html) elemento especifica el tipo de datos que operan en estas acciones. Se indica que la actividad puede obtener un cursor por encima de cero o más elementos ( vnd.android.cursor.dir ) de un proveedor de contenido que posee los datos de Notas ( vnd.google.note ). El objeto de la intención que pone en marcha la actividad incluiría un content: URI que especifica los datos exactos de este tipo que la actividad se debe abrir.

Nótese también la DEFAULT categoría suministra en este filtro. Está ahí porque el [Context.startActivity()](http://developer.android.com/reference/android/content/Context.html#startActivity%28android.content.Intent%29) y [Activity.startActivityForResult()](http://developer.android.com/reference/android/app/Activity.html#startActivityForResult%28android.content.Intent,%20int%29) los métodos de tratar a todos los intentos como si contuvieran el DEFAULT categoría - con sólo dos excepciones:

* + Intenciones que mencionen explícitamente la actividad de destino
	+ Intenciones que consisten en la MAIN acción y LAUNCHER categoría

Por lo tanto, el DEFAULT categoría se *requiere* para todos los filtros - a excepción de aquellos con el MAIN acción y LAUNCHER categoría. (Filtros de intenciones no se consultó a las intenciones explícitas.)

1. <intent-filter>
2. <Action: name = "android.intent.action.Get\_content "/>
3. <Categoría de android: name = "android.intent.category.MORA.
4. datos> android:mimeType="vnd.android.cursor.item/vnd.google.note" />
5. </ Intención de filtro>

Este filtro se describe la capacidad de la actividad de devolver una nota seleccionada por el usuario sin requerir ninguna especificación del directorio el usuario debe elegir. El GET\_CONTENT acción es similar a la PICK acción. En ambos casos, la actividad devuelve el URI para una nota seleccionada por el usuario. (En cada caso, se devuelve a la actividad que se llama [startActivityForResult()](http://developer.android.com/reference/android/app/Activity.html#startActivityForResult%28android.content.Intent,%20int%29) para iniciar la actividad NoteList.) Aquí, sin embargo, la persona que llama especifica el tipo de datos deseados en lugar del directorio de datos que el usuario va a recoger a partir.

El tipo de datos, vnd.android.cursor.item/vnd.google.note , indica el tipo de datos de la actividad puede volver - un URI para una sola nota. Desde la URI devuelto, la persona que llama puede obtener un cursor de exactamente un elemento ( vnd.android.cursor.item ) del proveedor de contenido que posee los datos de Notas ( vnd.google.note ).

En otras palabras, para la PICK acción en el filtro anterior, el tipo de datos indica el tipo de datos de la actividad podría mostrar al usuario. Para el GET\_CONTENT filtro, se indica el tipo de datos de la actividad puede volver a la llamada.

Teniendo en cuenta estas capacidades, los intentos siguientes se resolverá en la actividad NotesList:

acción: android.intent.action.MAIN

Inicia la actividad sin datos especificados.

acción: android.intent.action.MAIN
categoría: android.intent.category.LAUNCHER

Inicia la actividad con no especifica los datos seleccionados. Esta es la intención real utilizado por el lanzador para llenar su alto nivel de lista. Todas las actividades con filtros que coincidan con esta acción y la categoría se añaden a la lista.

acción: android.intent.action.VIEW
datos: content://com.google.provider.NotePad/notes

Pide a la actividad para mostrar una lista de todas las notas en content://com.google.provider.NotePad/notes . El usuario puede navegar a través de la lista y obtener información sobre los elementos de la misma.

acción: android.intent.action.PICK
datos: content://com.google.provider.NotePad/notes

Pide a la actividad para mostrar una lista de las notas en content://com.google.provider.NotePad/notes . El usuario puede entonces elegir una nota de la lista, y la actividad volverá el URI para que el tema de nuevo a la actividad que se inició la actividad NoteList.

acción: android.intent.action.GET\_CONTENT
tipo de datos: vnd.android.cursor.item/vnd.google.note

Pide a la actividad de suministro de un solo elemento de los datos de la aplicación Notas.

La segunda actividad, NoteEditor, muestra a los usuarios una entrada sola nota y les permite editarlo. Se pueden hacer dos cosas tal como se describe por sus filtros intención dos:

1. <intent-filter android:label="@string/resolve\_edit">
2. <Action: name = "android.intent.action."Vistas";
3. <Action: name = "android.intent.action.EDITAR
4. <Action: name = "com.android.notepad.action.[ editar ] Notas
5. <Categoría de android: name = "android.intent.category.MORA.
6. datos> android:mimeType="vnd.android.cursor.item/vnd.google.note" />
7. </ Intención de filtro>

El primero, el objetivo principal de esta actividad es permitir al usuario interactuar con una sola nota - ya sea a VIEW la nota o EDIT ella. (El EDIT\_NOTE categoría es sinónimo de EDIT ). El propósito sería contener la URI de datos que corresponden a los tipos MIME vnd.android.cursor.item/vnd.google.note - es decir, el URI de una sola nota, específico. Normalmente sería un URI que fue devuelto por los PICK o GET\_CONTENT acciones de la actividad NoteList.

Como antes, este filtro muestra el DEFAULT categoría, de modo que la actividad puede ser lanzado por los intentos de que no se especifica explícitamente la clase NoteEditor.

1. <intent-filter>
2. <Action: name = "android.intent.action.INSERT
3. <Categoría de android: name = "android.intent.category.MORA.
4. datos> android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
5. </ Intención de filtro>

El segundo propósito de esta actividad es permitir al usuario crear una nueva nota, que se INSERT en un directorio existente de notas. El propósito sería contener la URI de datos que corresponden a los tipos MIME vnd.android.cursor.dir/vnd.google.note - es decir, el URI para el directorio donde se debe colocar la nota.

Teniendo en cuenta estas capacidades, los intentos siguientes se resolverá en la actividad NoteEditor:

acción: android.intent.action.VIEW
datos: content://com.google.provider.NotePad/notes/*ID*

Pide a la actividad para mostrar el contenido de la nota identificada por *ID* . (Para obtener más información acerca de cómo content: URI especifica los miembros individuales de un grupo, vea [Proveedores de Contenido](http://developer.android.com/guide/topics/providers/content-providers.html) ).

acción: android.intent.action.EDIT
datos: content://com.google.provider.NotePad/notes/*ID*

Pide a la actividad para mostrar el contenido de la nota identificada por *ID* , y para permitir al usuario editar. Si el usuario guarda los cambios, las actualizaciones de los datos de actividad de la nota en el proveedor de contenido.

acción: android.intent.action.INSERT
datos: content://com.google.provider.NotePad/notes

Pide a la actividad para crear una nota nueva y vacía en la lista de notas en content://com.google.provider.NotePad/notes y permiten al usuario para editarlo. Si el usuario guarda la nota, el URI se devuelve a la persona que llama.

La última actividad, TitleEditor, permite al usuario editar el título de una nota. Esto podría ser implementado por la invocación directa de la actividad (estableciendo explícitamente su nombre del componente en el intento), sin necesidad de utilizar un filtro de intención. Pero aquí tenemos la oportunidad de demostrar cómo las operaciones de publicación alternativas sobre los datos existentes:

<intent-filter android:label="@string/resolve\_title">

 <Action: name = "com.android.notepad.action.EDIT\_TITLE "/>

 <Categoría de android: name = "android.intent.category.MORA.

 <Categoría de android: name = "android.intent.category.ALTERNATIVA "/>

 <Categoría de android: name = "android.intent.category.SELECTED\_ALTERNATIVE "/>

 datos> android:mimeType="vnd.android.cursor.item/vnd.google.note" />

</ Intención de filtro>

El filtro de la intención única de esta actividad utiliza una acción personalizada denominada " com.android.notepad.action.EDIT\_TITLE ". Debe ser invocado en una nota específica (tipo de datos vnd.android.cursor.item/vnd.google.note ), al igual que la anterior VIEW y EDIT acciones. Sin embargo, aquí la actividad muestra el título contenida en los datos de nota, no el contenido de la misma nota.

Además de apoyar a la habitual DEFAULT categoría, el editor de títulos también es compatible con otras dos categorías estándar: [ALTERNATIVE](http://developer.android.com/reference/android/content/Intent.html#CATEGORY_ALTERNATIVE) y [SELECTED\_ALTERNATIVE](http://developer.android.com/reference/android/content/Intent.html#CATEGORY_SELECTED_ALTERNATIVE) . Estas categorías identificar las actividades que se pueden presentar a los usuarios en un menú de opciones (tanto como el LAUNCHER categoría identifica las actividades que deben ser presentadas al usuario en el lanzador de aplicaciones). Tenga en cuenta que el filtro también suministra una etiqueta explícita (a través de android:label="@string/resolve\_title" ) para controlar mejor lo que ven los usuarios cuando se les presenta esta actividad como una acción alternativa a los datos que está viendo actualmente. (Para más información sobre estas categorías y menús de opciones de construcción, ver las [PackageManager.queryIntentActivityOptions()](http://developer.android.com/reference/android/content/pm/PackageManager.html#queryIntentActivityOptions%28android.content.ComponentName,%20android.content.Intent%5B%5D,%20android.content.Intent,%20int%29) y [Menu.addIntentOptions()](http://developer.android.com/reference/android/view/Menu.html#addIntentOptions%28int,%20int,%20int,%20android.content.ComponentName,%20android.content.Intent%5B%5D,%20android.content.Intent,%20int,%20android.view.MenuItem%5B%5D%29) métodos.)

Teniendo en cuenta estas capacidades, la siguiente intención se resolverá en la actividad TitleEditor:

acción: com.android.notepad.action.EDIT\_TITLE
datos: content://com.google.provider.NotePad/notes/*ID*

Pide a la actividad para mostrar el título asociado con el *ID* de nota, y permitir al usuario editar el título.

# Los procesos y subprocesos

## Vista Rapida

* Cada aplicación se ejecuta en su propio proceso y todos los componentes de la aplicación se ejecute en ese proceso, de forma predeterminada
* Las operaciones lentas, el bloqueo en una actividad debe hacerse en un nuevo tema, para evitar ralentizar la interfaz de usuario

## En este documento

1. [Procesos](http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html#Processes)
	1. [Proceso del ciclo de vida](http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html#Lifecycle)
2. [Hilos(Threads)](http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html#Threads)
	1. [Los subprocesos de trabajo](http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html#WorkerThreads)
	2. [Thread-safe métodos](http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html#ThreadSafe)
3. [Comunicación entre procesos](http://developer.android.com/guide/topics/fundamentals/processes-and-threads.html#IPC)

Cuando un componente de aplicación se inicia y la aplicación no tiene ningún otro componente en funcionamiento, el sistema Android inicia un nuevo proceso para la aplicación de Linux con un solo hilo de ejecución. Por defecto, todos los componentes de la misma aplicación se ejecute en el mismo proceso y el hilo (llamado el "principal" de rosca). Si un componente de aplicación se inicia y existe ya un procedimiento para que la aplicación (debido a que otro componente de la aplicación existe), a continuación, el componente se inició dentro de ese proceso y utiliza el mismo hilo de ejecución. Sin embargo, usted puede hacer arreglos para diferentes componentes en su aplicación se ejecute en procesos separados, y usted puede crear subprocesos adicionales para cualquier proceso.

Este documento analiza cómo los procesos y subprocesos trabajar en una aplicación Android.

## Procesos

Por defecto, todos los componentes de la misma aplicación se ejecute en el mismo proceso y la mayoría de las aplicaciones no deben cambiar esta situación. Sin embargo, si usted encuentra que usted necesita para controlar los proceso de un determinado componente pertenece, puede hacerlo en el archivo de manifiesto.

La entrada de manifiesto para cada tipo de componente de elemento- [<activity>](http://developer.android.com/guide/topics/manifest/activity-element.html) , [<service>](http://developer.android.com/guide/topics/manifest/service-element.html) , [<receiver>](http://developer.android.com/guide/topics/manifest/receiver-element.html) y [<provider>](http://developer.android.com/guide/topics/manifest/provider-element.html) -soporta un android:process atributo que se puede especificar un proceso en el que dicho componente se debe ejecutar. Puede establecer este atributo para que cada componente se ejecuta en su propio proceso, o que algunos componentes de compartir un proceso, mientras que otros no lo hacen. También puede configurar android:process para que los componentes de las diferentes aplicaciones se ejecutan en el mismo proceso-siempre que las aplicaciones comparten el mismo ID de usuario y Linux están firmados con los mismos certificados.

El [<application>](http://developer.android.com/guide/topics/manifest/application-element.html) elemento también es compatible con un android:process atributo, para establecer un valor predeterminado que se aplica a todos los componentes.

Android podría decidir cerrar un proceso en algún momento, cuando queda poca memoria y requeridos por otros procesos que están más directamente al servicio del usuario. Los componentes de aplicaciones que se ejecutan en el proceso que ha matado, en consecuencia destruido. Un proceso se inicia de nuevo para aquellos componentes que no se volverá a trabajar para ellos.

Al decidir que los procesos de matar, el sistema Android pesa su importancia relativa para el usuario. Por ejemplo, más fácilmente se cierra un proceso de alojamiento actividades que ya no son visibles en la pantalla, en comparación con un proceso de alojamiento actividades visibles. La decisión de terminar un proceso, por lo tanto, depende del estado de los componentes que se ejecutan en ese proceso. Las reglas que se usan para decidir qué poner fin a los procesos que se discuten a continuación.

### Proceso del ciclo de vida

El sistema Android trata de mantener un proceso de solicitud durante el mayor tiempo posible, pero con el tiempo necesario para eliminar los procesos viejos para recuperar la memoria para los procesos nuevos o más importante. Para determinar qué procesos para mantener, y que matar, el sistema coloca cada proceso en una "jerarquía importante", basada en los componentes que se ejecutan en el proceso y el estado de los componentes. Los procesos con la menor importancia se eliminan primero, entonces aquellos con la menor importancia siguiente, y así sucesivamente, si es necesario para recuperar los recursos del sistema.

Hay cinco niveles en la jerarquía de importancia. La siguiente lista muestra los diferentes tipos de procesos en orden de importancia (el primer proceso es más importante y es asesinado el pasado):

1. **Primer plano el proceso**

Un proceso que se requiere para lo que el usuario está haciendo actualmente. Un proceso se considera en el primer plano si alguna de las siguientes condiciones:

* + Se organiza una [Activity](http://developer.android.com/reference/android/app/Activity.html) que el usuario está interactuando con (la [Activity](http://developer.android.com/reference/android/app/Activity.html) 's [onResume()](http://developer.android.com/reference/android/app/Activity.html#onResume%28%29) método ha sido llamado).
	+ Alberga un [Service](http://developer.android.com/reference/android/app/Service.html) que está destinado a la actividad que el usuario está interactuando.
	+ Alberga un [Service](http://developer.android.com/reference/android/app/Service.html) que se está ejecutando "en el primer plano", el servicio ha llamado [startForeground()](http://developer.android.com/reference/android/app/Service.html#startForeground%28int,%20android.app.Notification%29) .
	+ Alberga un [Service](http://developer.android.com/reference/android/app/Service.html) que se está ejecutando uno de sus devoluciones de llamada de ciclo de vida ( [onCreate()](http://developer.android.com/reference/android/app/Service.html#onCreate%28%29) , [onStart()](http://developer.android.com/reference/android/app/Service.html#onStart%28android.content.Intent,%20int%29) , o [onDestroy()](http://developer.android.com/reference/android/app/Service.html#onDestroy%28%29) ).
	+ Alberga una [BroadcastReceiver](http://developer.android.com/reference/android/content/BroadcastReceiver.html) que está ejecutando su [onReceive()](http://developer.android.com/reference/android/content/BroadcastReceiver.html#onReceive%28android.content.Context,%20android.content.Intent%29) método.

Generalmente, sólo unos pocos procesos en primer plano existir en cualquier momento dado. Son muertos sólo como un último recurso, si la memoria no es tan baja que no todos pueden seguir funcionando. Generalmente, en ese punto, el dispositivo ha alcanzado un estado de paginación de memoria, lo que eliminar algunos procesos en primer plano se requiere para mantener la interfaz de usuario sensible.

1. **Proceso visible**

Un proceso que no tiene ningún componente de primer plano, pero todavía puede afectar a lo que el usuario ve en pantalla. Un proceso se considera que es visible si cualquiera de las siguientes condiciones:

* + Se organiza una [Activity](http://developer.android.com/reference/android/app/Activity.html) que no está en primer plano, pero todavía es visible para el usuario (su [onPause()](http://developer.android.com/reference/android/app/Activity.html#onPause%28%29) método ha sido llamado). Esto podría ocurrir, por ejemplo, si la actividad de primer plano inició un diálogo, que permite que la actividad anterior para ser visto detrás de él.
	+ Alberga un [Service](http://developer.android.com/reference/android/app/Service.html) que está unido a un visible (o primer plano) la actividad.

Un proceso visible es considerado muy importante y no se mató a menos que al hacerlo se requiere para mantener todos los procesos en primer plano en ejecución.

1. **Servicio de proceso de**

Un proceso que se está ejecutando un servicio que se ha iniciado con la [startService()](http://developer.android.com/reference/android/content/Context.html#startService%28android.content.Intent%29) método y no cae en ninguna de las dos categorías más altas. Aunque los procesos de servicios que no están directamente vinculados a cualquier cosa que ve el usuario, por lo general hacer las cosas que el usuario se preocupa (por ejemplo, la reproducción de música en el fondo o la descarga de datos en la red), por lo que el sistema sigue en funcionamiento a menos que no hay suficiente memoria para retenerlos junto con todos los procesos de primer plano y visibles.

1. **Antecedentes proceso de**

Un proceso que mantiene una actividad que no es actualmente visible para el usuario (de la actividad [onStop()](http://developer.android.com/reference/android/app/Activity.html#onStop%28%29) método ha sido llamado). Estos procesos no tienen un impacto directo en la experiencia del usuario y el sistema puede matar en cualquier momento para recuperar la memoria de un primer plano, visible o proceso de servicio. Por lo general hay muchos procesos de fondo que ejecutan, por lo que se mantienen en un LRU (menos utilizado recientemente) lista para asegurarse de que el proceso con la actividad que fue visto más recientemente por el usuario es el último en ser asesinado. Si una actividad práctica sus métodos de ciclo de vida correctamente, y guarda su estado actual, matando a su proceso no tendrá un efecto visible en la experiencia del usuario, ya que cuando el usuario navega de regreso a la actividad, la actividad restaura la totalidad de su estado de visibilidad. Ver la [Actividades](http://developer.android.com/guide/topics/fundamentals/activities.html#SavingActivityState) documento para obtener información acerca de cómo guardar y restaurar el estado.

1. **Proceso de vacío**

Un proceso que no tiene ninguno de los componentes de aplicaciones activas. La única razón para mantener este tipo de proceso vivo es con fines de almacenamiento en caché, para mejorar el tiempo de inicio la próxima vez que un componente tiene que ejecutar en el mismo. El sistema mata a menudo a estos procesos con el fin de equilibrar los recursos globales del sistema entre los cachés de los procesos y los cachés del núcleo subyacente.

Android ocupa un proceso en el nivel más alto que puede, basado en la importancia de los componentes activos actualmente en el proceso. Por ejemplo, si un proceso aloja un servicio y una actividad visible, el proceso se clasificó como un proceso visible, no un proceso de servicio.

Además, la clasificación de un proceso puede ser mayor debido a otros procesos dependen de ella-un proceso que está sirviendo a otro proceso no puede ser clasificado más bajo que el proceso se está cumpliendo. Por ejemplo, si un proveedor de contenido en el proceso A está sirviendo un cliente en el proceso B, o si un servicio en el proceso A está unido a un componente en el proceso B, el proceso A siempre se considera por lo menos tan importante como el proceso B.

Debido a que un proceso que se ejecuta un servicio está en el puesto más alto que un proceso con las actividades de fondo, una actividad que se inicia una operación de larga duración harían bien en comenzar un [servicio](http://developer.android.com/guide/topics/fundamentals/services.html) para esa operación, en lugar de simplemente crear un subproceso de trabajo, sobre todo si la operación es probable se sobrevivir a la actividad. Por ejemplo, una actividad que está subiendo una foto a un sitio web debe iniciar un servicio para realizar la carga de modo que la carga pueda seguir en el fondo incluso si el usuario sale de la actividad. El uso de un servicio garantiza que la operación tendrá por lo menos "proceso de servicio" prioridad, independientemente de lo que ocurre con la actividad. Esta es la misma razón que los receptores de radiodifusión deben emplear los servicios en lugar de simplemente poner que consumen mucho tiempo en un hilo.

## Hilos(Threads)

Cuando se inicia una aplicación, el sistema crea un hilo de ejecución para la aplicación, llamada "principal". Este hilo es muy importante porque es el encargado de despacho de eventos a los widgets de la interfaz de usuario apropiados, incluidos los eventos de dibujo. También es el hilo en el que la aplicación interactúa con los componentes del kit de herramientas de interfaz de usuario para Android (los componentes de la [android.widget](http://developer.android.com/reference/android/widget/package-summary.html) y [android.view](http://developer.android.com/reference/android/view/package-summary.html) paquetes). Como tal, el hilo principal es también llamado a veces el hilo de la interfaz de usuario.

El sistema no crea un hilo para cada instancia de un componente. Todos los componentes que se ejecutan en el mismo proceso se ejecute en el hilo de la interfaz de usuario, y pide a cada componente del sistema se envían desde ese hilo. En consecuencia, los métodos que responden a las devoluciones de llamada del sistema (como [onKeyDown()](http://developer.android.com/reference/android/view/View.html#onKeyDown%28int,%20android.view.KeyEvent%29) para informar las acciones del usuario o un método de devolución de llamada del ciclo de vida) siempre se ejecutan en el subproceso de interfaz de usuario del proceso.

Por ejemplo, cuando el usuario toca un botón en la pantalla, los despachos de su aplicación de interfaz de usuario de rosca del evento táctil para el widget, que a su vez establece su estado presionado y publica una solicitud invalida a la cola de eventos. El hilo de la interfaz de usuario quita de la cola de la solicitud y notifica el widget que se debería volver a dibujar.

Cuando su aplicación lleva a cabo una intensa labor en respuesta a la interacción del usuario, este modelo solo hilo puede dar malos resultados a menos que poner en práctica su solicitud correctamente. En concreto, si todo lo que está sucediendo en el hilo de la interfaz de usuario, la realización de las operaciones de largo, como acceso a la red o las consultas de base de datos se bloqueará toda la interfaz de usuario. Cuando el hilo está bloqueado, no hay eventos pueden ser enviados, incluyendo los eventos de dibujo. Desde la perspectiva del usuario, la aplicación se bloquea. Lo que es peor, si el subproceso de interfaz de usuario está bloqueada por más de unos segundos (unos 5 segundos en la actualidad) se presenta al usuario con el infame " [la aplicación no responde](http://developer.android.com/guide/practices/design/responsiveness.html) "(ANR) de diálogo. Luego, el usuario puede decidir salir de la aplicación y desinstalarlo si no están contentos.

Además, la interfaz de usuario Andoid guía no es thread-safe. Por lo tanto, no debe manipular la interfaz de usuario de un subproceso de trabajo, usted debe hacer todo tipo de manipulación de la interfaz de usuario desde el subproceso de interfaz de usuario. Por lo tanto, no son más que dos reglas en el modelo de hilo de Android:

1. No bloquear el subproceso de interfaz de usuario
2. No acceder a la interfaz de usuario de Android kit de herramientas desde el exterior del hilo de la interfaz de usuario

### Los subprocesos de trabajo

Debido al modelo de hilo se ha descrito anteriormente, es vital para la capacidad de respuesta de la interfaz de usuario de la aplicación que no bloquea el subproceso de interfaz de usuario. Si usted tiene operaciones a realizar que no son instantáneos, usted debe asegurarse de hacerlas en hilos separados ("fondo" o "hilos" de los trabajadores).

Por ejemplo, a continuación es algo de código para un oyente clic con el botón que descarga una imagen de un hilo separado y lo muestra en una [ImageView](http://developer.android.com/reference/android/widget/ImageView.html) :

public void onClick (View v) {

 Tema nuevo (new Runnable () {

 public void run () {

 Bitmap b = loadImageFromNetwork ("http://example.com/image.png");

 mImageView.setImageBitmap (b);

 }

 Empezado

}

Al principio, esto parece funcionar bien, ya que crea un nuevo hilo para manejar la operación de la red. Sin embargo, se viola la regla segunda del modelo de un único subproceso: no acceder a la interfaz de usuario de Android conjunto de herramientas de fuera de la interfaz de usuario de hilo de esta muestra, se modifica el [ImageView](http://developer.android.com/reference/android/widget/ImageView.html) desde el subproceso de trabajo en lugar del hilo de la interfaz de usuario. Esto puede resultar en un comportamiento indefinido e inesperado, que puede ser difícil y requerir mucho tiempo para rastrear.

Para solucionar este problema, Android ofrece varias formas de acceder el hilo de interfaz de usuario de otros subprocesos. Aquí está una lista de métodos que pueden ayudar:

* [Activity.runOnUiThread(Runnable)](http://developer.android.com/reference/android/app/Activity.html#runOnUiThread%28java.lang.Runnable%29)
* [View.post(Runnable)](http://developer.android.com/reference/android/view/View.html#post%28java.lang.Runnable%29)
* [View.postDelayed(Runnable, long)](http://developer.android.com/reference/android/view/View.html#postDelayed%28java.lang.Runnable,%20long%29)

Por ejemplo, usted puede fijar el código anterior utilizando el [View.post(Runnable)](http://developer.android.com/reference/android/view/View.html#post%28java.lang.Runnable%29) método:

public void onClick (View v) {

 Tema nuevo (new Runnable () {

 public void run () {

 de mapa de bits de mapa de bits final = loadImageFromNetwork ("http://example.com/image.png");

 mImageView.post (new Runnable () {

 public void run () {

 mImageView.setImageBitmap (mapa de bits);

 }

 });

 }

 Empezado

}

Ahora bien, esta aplicación es segura para subprocesos: el funcionamiento de la red se realiza a partir de un hilo separado, mientras que el [ImageView](http://developer.android.com/reference/android/widget/ImageView.html) se manipula desde el subproceso de interfaz de usuario.

Sin embargo, como la complejidad de la operación crece, este tipo de código puede conseguir complicado y difícil de mantener. Para manejar las interacciones más complejas con un subproceso de trabajo, es posible considerar el uso de un [Handler](http://developer.android.com/reference/android/os/Handler.html) en el subproceso de trabajo, para procesar los mensajes enviados desde el subproceso de interfaz de usuario. Tal vez la mejor solución, sin embargo, es extender la [AsyncTask](http://developer.android.com/reference/android/os/AsyncTask.html) clase, lo que simplifica la ejecución de las tareas de subproceso de trabajo que necesitan para interactuar con la interfaz de usuario.

#### Usando AsyncTask

[AsyncTask](http://developer.android.com/reference/android/os/AsyncTask.html) le permite realizar el trabajo asincrónico en la interfaz de usuario. Se lleva a cabo las operaciones de bloqueo en un subproceso de trabajo y, a continuación se publican los resultados en el hilo de la interfaz de usuario, sin necesidad de manejar temas y / o manipuladores de ti mismo.

Para usarlo, debe subclase [AsyncTask](http://developer.android.com/reference/android/os/AsyncTask.html) y aplicar el [doInBackground()](http://developer.android.com/reference/android/os/AsyncTask.html#doInBackground%28Params...%29) método de devolución de llamada, que se ejecuta en un grupo de subprocesos de fondo. Para actualizar la interfaz de usuario, se deben implementar [onPostExecute()](http://developer.android.com/reference/android/os/AsyncTask.html#onPostExecute%28Result%29) , que ofrece el resultado de [doInBackground()](http://developer.android.com/reference/android/os/AsyncTask.html#doInBackground%28Params...%29) y se ejecuta en el subproceso de interfaz de usuario, por lo que con seguridad se puede actualizar la interfaz de usuario. A continuación, puede ejecutar la tarea llamando a [execute()](http://developer.android.com/reference/android/os/AsyncTask.html#execute%28Params...%29) del hilo de interfaz de usuario.

Por ejemplo, puede implementar el ejemplo anterior utilizando [AsyncTask](http://developer.android.com/reference/android/os/AsyncTask.html) de esta manera:

public void onClick (View v) {

 . nueva DownloadImageTask () ejecutar ("http://example.com/image.png");

}

DownloadImageTask clase privada se extiende AsyncTask <String, Void, Bitmap> {

 / \*\* El sistema llama a este para realizar un trabajo en un subproceso de trabajo y

 \* Que proporciona los parámetros dados a AsyncTask.execute () \* /

 de mapa de bits protegidos doInBackground (String. urls ..) {

 volver loadImageFromNetwork (urls [0]);

 }

 / \*\* El sistema llama a este para realizar un trabajo en el subproceso de interfaz de usuario y entrega

 \* El resultado de doInBackground () \* /

 protected void OnPostExecute (resultado de mapa de bits) {

 mImageView.setImageBitmap (resultado);

 }

}

Ahora la interfaz de usuario es segura y el código es más sencillo, ya que separa el trabajo en la parte que se debe hacer en un subproceso de trabajo y la parte que se debe hacer en el subproceso de interfaz de usuario.

Usted debe leer la [AsyncTask](http://developer.android.com/reference/android/os/AsyncTask.html) de referencia para una comprensión completa de cómo utilizar esta clase, pero aquí es una visión rápida de cómo funciona:

* Usted puede especificar el tipo de los parámetros, los valores de progreso, y el valor final de la tarea, el uso de genéricos
* El método [doInBackground()](http://developer.android.com/reference/android/os/AsyncTask.html#doInBackground%28Params...%29) se ejecuta de forma automática en un subproceso de trabajo
* [onPreExecute()](http://developer.android.com/reference/android/os/AsyncTask.html#onPreExecute%28%29) , [onPostExecute()](http://developer.android.com/reference/android/os/AsyncTask.html#onPostExecute%28Result%29) , y [onProgressUpdate()](http://developer.android.com/reference/android/os/AsyncTask.html#onProgressUpdate%28Progress...%29) se invocan en el subproceso de interfaz de usuario
* El valor devuelto por [doInBackground()](http://developer.android.com/reference/android/os/AsyncTask.html#doInBackground%28Params...%29) es enviado a [onPostExecute()](http://developer.android.com/reference/android/os/AsyncTask.html#onPostExecute%28Result%29)
* Usted puede llamar a [publishProgress()](http://developer.android.com/reference/android/os/AsyncTask.html#publishProgress%28Progress...%29) en cualquier momento en [doInBackground()](http://developer.android.com/reference/android/os/AsyncTask.html#doInBackground%28Params...%29) para ejecutar [onProgressUpdate()](http://developer.android.com/reference/android/os/AsyncTask.html#onProgressUpdate%28Progress...%29) en el subproceso de interfaz de usuario
* Usted puede cancelar la tarea en cualquier momento, desde cualquier subproceso

**Atención:** Otro de los problemas que pueden surgir cuando se utiliza un subproceso de trabajo es reinicios inesperados en su actividad debido a un [cambio de configuración en tiempo de ejecución](http://developer.android.com/guide/topics/resources/runtime-changes.html) (por ejemplo, cuando el usuario cambia la orientación de la pantalla), que pueden destruir su subproceso de trabajo. Para ver cómo puede persistir su tarea durante uno de estos se reinicia y cómo cancelar correctamente la tarea cuando la actividad se destruye, ver el código fuente para el [Estantes](http://code.google.com/p/shelves/) aplicación de ejemplo.

### Thread-safe métodos

En algunas situaciones, los métodos que implementan podría ser llamado desde más de un hilo, y por lo tanto debe ser escrito para ser thread-safe.

Esto es principalmente cierto para los métodos que se pueden llamar de forma remota, tales como métodos en un [servicio de la envolvente](http://developer.android.com/guide/topics/fundamentals/bound-services.html) . Cuando recibe una llamada de un método implementado en un [IBinder](http://developer.android.com/reference/android/os/IBinder.html) se origina en el mismo proceso en el que el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) se está ejecutando, el método se ejecuta en el hilo de la persona que llama. Sin embargo, cuando la llamada se origina en otro proceso, el método se ejecuta en un hilo elegida entre un grupo de hilos que el sistema mantiene en el mismo proceso que el [IBinder](http://developer.android.com/reference/android/os/IBinder.html) (no es ejecutado en el hilo de interfaz de usuario del proceso). Por ejemplo, mientras que un servicio de [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) método se llama desde el subproceso de interfaz de usuario del proceso del servicio, los métodos implementados en el objeto que [onBind()](http://developer.android.com/reference/android/app/Service.html#onBind%28android.content.Intent%29) devuelve (por ejemplo, una subclase que implementa los métodos de RPC) será llamado desde las discusiones en el piscina. Debido a que un servicio puede tener más de un cliente, más de un hilo piscina puede realizar la misma [IBinder](http://developer.android.com/reference/android/os/IBinder.html) método al mismo tiempo. [IBinder](http://developer.android.com/reference/android/os/IBinder.html) métodos deben, por tanto, ponerse en práctica para ser thread-safe.

Del mismo modo, un proveedor de contenido puede recibir solicitudes de datos que se originan en otros procesos. Aunque el [ContentResolver](http://developer.android.com/reference/android/content/ContentResolver.html) y [ContentProvider](http://developer.android.com/reference/android/content/ContentProvider.html) clases de ocultar los detalles de cómo la comunicación entre procesos se gestiona, [ContentProvider](http://developer.android.com/reference/android/content/ContentProvider.html) métodos que responden a las peticiones-los métodos [query()](http://developer.android.com/reference/android/content/ContentProvider.html#query%28android.net.Uri,%20java.lang.String%5B%5D,%20java.lang.String,%20java.lang.String%5B%5D,%20java.lang.String%29) , [insert()](http://developer.android.com/reference/android/content/ContentProvider.html#insert%28android.net.Uri,%20android.content.ContentValues%29) , [delete()](http://developer.android.com/reference/android/content/ContentProvider.html#delete%28android.net.Uri,%20java.lang.String,%20java.lang.String%5B%5D%29) , [update()](http://developer.android.com/reference/android/content/ContentProvider.html#update%28android.net.Uri,%20android.content.ContentValues,%20java.lang.String,%20java.lang.String%5B%5D%29) , y [getType()](http://developer.android.com/reference/android/content/ContentProvider.html#getType%28android.net.Uri%29) -se llaman a partir de un grupo de subprocesos en el proceso del proveedor de contenidos no, el hilo de interfaz de usuario para el proceso. Debido a que estos métodos pueden llamar desde cualquier número de hilos al mismo tiempo, ellos también deben ponerse en práctica para ser thread-safe.

## Comunicación entre procesos

Android ofrece un mecanismo de comunicación entre procesos (IPC) con llamadas a procedimientos remotos (RPC), en el que se llama un método por una actividad o un componente de otra aplicación, pero ejecutados de forma remota (en otro proceso), con cualquier resultado que devuelve al llamador. Esto implica la descomposición de una llamada de método y de sus datos a un nivel del sistema operativo puede entender, transmitir que en el proceso local y espacio de direcciones del proceso a distancia y el espacio de direcciones, entonces y volver a montar recreando la llamada allí. Los valores de retorno se transmiten entonces en la dirección opuesta. Android proporciona todo el código para realizar estas operaciones de la CIA, para que pueda centrarse en la definición e implementación de la interfaz de programación de RPC.

Para llevar a cabo la IPC, la aplicación debe unirse a un servicio, utilizando [bindService()](http://developer.android.com/reference/android/content/Context.html#bindService%28android.content.Intent,%20android.content.ServiceConnection,%20int%29) . Para obtener más información, consulte el [Servicios de](http://developer.android.com/guide/topics/fundamentals/services.html) guía para el desarrollador.