Servicio de mensajería

Tiempo estimado: 20min

Una característica muy valorada de Redis es su servicio de mensajería integrado. El objeto de la presente lección. Redis es un motor de bases de datos que proporciona servicio de tiempo real a través de su servicio de mensajería. Muy útil para todo tipo de aplicaciones, principalmente, webs y móviles.

Comenzamos la lección describiendo el concepto de servicio de mensajería y distinguiendo entre los modelos punto a punto y publicación/suscripción. Para a continuación presentar los comandos del modelo implementado por Redis.

Al finalizar la lección, el estudiante sabrá:

- Qué es un servicio de mensajería.
- Cuáles son las diferencias entre los servicios punto a punto y publicación/suscripción.
- Cómo utilizar el servicio publicación/suscripción implementado por Redis.

Introducción

Un servicio de mensajería (*messaging service*) es un sistema de comunicación para el intercambio de mensajes entre dos o más componentes de software. Sus principales usos son los siguientes:

- Implementación de chats.
- Notificación de información o datos en tiempo real como eventos, alertas, noticias, cambios de precio, registros, etc. Recibida la notificación, el cliente puede realizar una acción como, por ejemplo, su visualización o el envío de un correo.

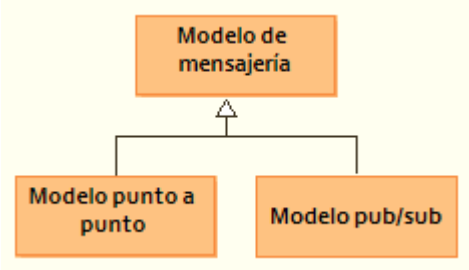
Las principales características por las que se usa un servicio o sistema de mensajería son:

- Desacoplamiento de los componentes.

  Los emisores de los mensajes no se conectan a los receptores, sino al intermediario. De igual manera actúan los receptores, se conectan al intermediario. Por lo que los extremos no se comunican directamente, manteniéndose desacoplados.

- Escalabilidad del sistema si la carga crece o decrece.

Generalmente, los servicios se implementan bajo uno de dos modelos de mensajería (*messaging models*), una manera de concebir el sistema. Define los componentes que lo forman, cómo se conectan y cómo se intercambian los mensajes. Estos modelos son el punto a punto y el basado en publicación/suscripción.



Bajo el modelo punto a punto (*point-to-point model*), los emisores envían sus mensajes a un intermediario, el cual se lo entregará a un único receptor. Mientras que en el modelo publicación/suscripción (*pub/sub model*), el mensaje puede llegar a varios receptores. Redis implementa el modelo publicación/suscripción de una manera muy fácil de usar, pero es interesante conocer lo que proporcionan ambos.

Modelo punto a punto

El modelo punto a punto (*point-to-point model*) describe el sistema de mensajería como un medio para que un mensaje remitido por un componente se entregue a un único receptor. Para ello, tiene varios componentes: el emisor, el consumidor, el intermediario y la cola de mensajes.

El intermediario (*broker*) es el componente que media entre los emisores y los consumidores de mensajes. La cola (*queue*) es el contenedor donde el intermediario almacena temporalmente los mensajes recibidos para su entrega. Los emisores (*senders*) son aquellos que envían o remiten mensajes. Se los envían al intermediario, para que éste se los entregue a los destinatarios, los consumidores (*consumers*) o receptores (*receivers*).

Generalmente se prefiere el término consumidor, antes que el de receptor. La razón es bien sencilla. Un mensaje podría ser entregado a varios receptores, pero el intermediario se asegura que sólo se lo entregará a *uno* de ellos.

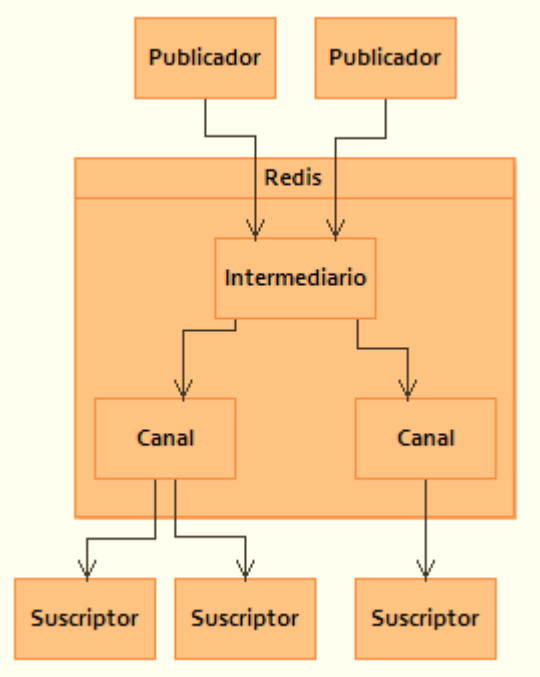
El modelo punto a punto se utiliza básicamente para procesar órdenes o tareas por parte de uno o más consumidores. Pero toda orden o tarea sólo debe ser procesada por un único consumidor. Por lo general, el sistema funciona de manera muy pero que muy sencilla. Cuando un consumidor termina con su orden de trabajo actual, lo que hace es remitirse al intermediario e indicarle que le proporcione el siguiente mensaje de su cola asociada. El intermediario se asegurará que, como puede haber varios consumidores, un mensaje sólo se entregará a un único consumidor.

Es importante tener claro que bajo el modelo punto a punto, aunque haya varios componentes consumidores leyendo de ella, el servicio asegura que nunca un mensaje se entregará a más de uno. Por lo que si es necesario que un mismo mensaje sea atendido por varios consumidores, el modelo punto a punto no permitirá hacerlo.

Modelo publicación/suscripción

En el modelo publicación/suscripción (*publish/subscribe model*), un mismo mensaje puede llegar a cero, uno o más receptores. Como bien indica su nombre, se basa en los conceptos de publicación y suscripción a canales.

Un canal (*channel*) o tema (*topic*) es un contenedor donde se publican mensajes. Es el medio a través del cual los extremos intercambian mensajes. El intermediario es el responsable de administrarlos. Una vez los ha entregado, los suprime automáticamente. Los emisores de los mensajes se los envían al intermediario indicándole a través de qué canales debe difundirlos. A esta operación se la conoce formalmente como publicación (*publication*). Mientras que cuando un componente desea recibir mensajes publicados por otros lo que hace es abonarse a los canales que son de su interés, lo que se conoce formalmente como suscripción (*subscription*). Cada vez que el intermediario publica un mensaje en un canal, cuando pueda se lo enviará a sus suscriptores y, finalizado, lo suprimirá del canal.



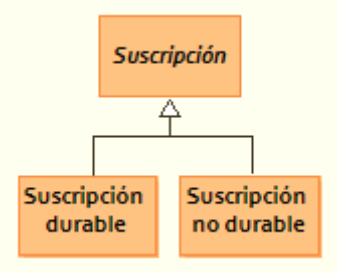
A los que emiten mensajes se les conoce como publicadores (*publishers*), mientras que a los que se les entrega como suscriptores (*subscribers*). Los publicadores pueden publicar sus mensajes en varios canales y de igual manera los suscriptores pueden abonarse a varios canales.

Por lo que vemos, el sistema es muy sencillo y permite que un mismo mensaje llegue a varios destinatarios, todos aquellos que se encuentran suscritos al canal a través del cual se publica.

Redis se basa en este modelo y su rol es el de intermediario. Se encarga de administrar los canales de comunicación, recibir los mensajes, publicarlos y entregarlos a sus suscriptores. Todo ello de manera muy sencilla.

Tipos de suscripción

Atendiendo a la duración de una suscripción, distinguimos entre durables y no durables.



Una suscripción durable (*durable subscription*) es aquella que perdura aun cuando el suscriptor se desconecta del intermediario, formal o informalmente. Es necesario que se dé de baja explícitamente. En cambio, una suscripción no durable (*non-durable subscription*) se da de baja cuando explícitamente así lo solicite el usuario o bien se desconecte, lo que primero ocurra. La cuestión es que finalizada la conexión, sea como sea, el cliente ya no tendrá suscripción, siendo necesario volver a solicitarla si la necesita.

En una suscripción durable, cuando el suscriptor se desconecta, el intermediario se compromete a guardar los mensajes que todavía no le haya entregado y que le lleguen hasta que vuelva a conectarse. En cambio, en una suscripción no durable, cuando el suscriptor se desconecta, automáticamente pierde los mensajes que no le ha entregado todavía y cualquier otro que pueda publicarse hasta que vuelva a renovar su suscripción explícitamente.

Las suscripciones de Redis son no durables.

Publicación de mensajes

Recordemos que la publicación de un mensaje hace referencia al envío del mensaje a un canal del intermediario para su difusión a todos sus suscriptores. Esta operación se realiza mediante el comando PUBLISH de Redis:

```
PUBLISH canal mensaje
```

Los mensajes deben ser de tipo cadena. Su contenido no tiene ningún formato especial, se delega a la aplicación. El comando devuelve el número de suscriptores a los que les entregará el mensaje.

He aquí un ejemplo ilustrativo:

```
127.0.0.1:6379> PUBLISH alta "{'correo': 'elvis@costello.com', 'nombre': 'Elvis'}"
(integer) 1
127.0.0.1:6379>
```

Redis no proporciona ningún comando especial para crear canales. Los crea a medida que lo necesita, concretamente con las suscripciones. Si se publica un mensaje en un canal inexistente, no se quejará, simplemente indicará que no se lo entregó a nadie. Cuando un canal deja de tener suscripciones, automáticamente lo suprimirá.

Suscripción a canales

Para suscribirnos a un canal, podemos utilizar los comandos SUBSCRIBE y PSUBSCRIBE:

```
SUBSCRIBE canal
SUBSCRIBE canal canal canal...
PSUBSCRIBE patrón
PSUBSCRIBE patrón patrón patrón...
```

SUBSCRIBE suscribe a uno o más canales según su nombre. Mientras que PSUBSCRIBE permite hacerlo a través de patrones de nombres. Así, por ejemplo, el patrón eco* representa cualquier canal que comience por eco.

Recordemos que las suscripciones en Redis son no durables, lo que quiere decir que si no damos de baja las suscripciones explícitamente, implícitamente lo hará el sistema al cerrar o perder la conexión con el cliente. Así pues, cada vez que un cliente vuelve a conectar con la instancia Redis, debe realizar de nuevo sus suscripciones.

Cuando ejecutamos un comando de este tipo mediante redis-cli, se bloqueará y se quedará a la espera de mensajes. Cada vez que reciba un mensaje, lo mostrará por la consola tal como se muestra a continuación:

```
127.0.0.1:6379> SUBSCRIBE alta
Reading messages... (press Ctrl-C to quit)
1) "subscribe"
2) "alta"
3) (integer) 1
1) "message"
2) "alta"
3) "{'correo': 'elvis@costello.com', nombre: 'Elvis Costello'}"
1) "message"
2) "alta"
3) "{'correo': 'ian@mcculloch.com', nombre: 'Ian McCulloch'}"
```

El comando muestra entradas de tres tipos: el proceso de suscripción, los mensajes recibidos y el proceso de baja de suscripción. Cada entrada es una lista de tres elementos:

- El primero indica el tipo de entrada: subscribe, suscripción; message, mensaje recibido; y unsubscribe, baja de suscripción.
- El segundo indica el nombre del canal.
- El tercero depende del tipo de entrada. En las entradas de mensaje, el texto del mensaje. En las suscripciones y bajas de suscripciones, el número de suscripciones actuales del cliente.

Generalmente, redis-cli no se utiliza para recibir mensajes, porque al suscribirse se bloquea. Se puede utilizar para publicar, pero es raro utilizarlo para suscripciones, a menos que estemos monitorizando el canal. Cuando se necesita reaccionar o ejecutar un controlador como consecuencia de la recepción del mensaje, se suele utilizar aplicaciones escritas, por ejemplo, en C/C++, Java, JavaScript o Python.

En Redis, cuando un cliente se suscribe a un canal, no puede ejecutar otros comandos que los relacionados con el sistema de mensajería. Si necesita consultar la base de datos, debe abrir otra conexión aparte y encauzar los comandos por ella.

Baja de suscripción

La baja de suscripción se lleva a cabo mediante los comandos siguientes:

```
UNSUBSCRIBE
UNSUBSCRIBE canal
UNSUBSCRIBE canal canal canal...
PUNSUBSCRIBE patrón
PUNSUBSCRIBE patrón patrón patrón...
```

UNSUBSCRIBE da de baja de uno o más canales. Cuando no se indica ninguno, de todos a los que el cliente está suscrito. Si se desea ser más explícito, se puede indicar los canales en cuestión, separándolos por espacios. PUNSUBSCRIBE es similar, pero utiliza patrones, en vez de nombres concretos.

Recordemos que si un cliente cierra o pierde la conexión con la instancia Redis, se realiza una baja implícita. Cuando se conecte el cliente de nuevo, será necesario que vuelva a suscribirse.

Comando PUBSUB

El comando PUBSUB se utiliza para obtener información del servicio de mensajería. Básicamente, los canales existentes y los suscriptores. Para conocer los canales, se utiliza la siguiente sintaxis:

```
PUBSUB CHANNELS
PUBSUB CHANNELS patrón
```

Si no se indica ningún patrón, mostrará todos los existentes actualmente. Si se indica un patrón, aquellos cuyo nombre coincida con el indicado. Ejemplo:

```
127.0.0.1:6379> PUBSUB CHANNELS
1) "baja"
2) "alta"
127.0.0.1:6379>
```

Para conocer el número de suscriptores de uno o más canales indicados, se utiliza:

```
PUBSUB NUMSUB canal
PUBSUB NUMSUB canal canal canal...
```

Este comando no contabiliza los suscriptores bajo patrones, esto es, mediante PSUBSCRIBE. Veamos un ejemplo de su resultado:

```
127.0.0.1:6379> PUBSUB NUMSUB alta baja
1) "alta"
2) (integer) 1
3) "baja"
4) (integer) 2
127.0.0.1:6379>
```

Para conocer los suscriptores bajo patrones, se utiliza la siguientes sintaxis:

```
PUBSUB NUMPAT
```

