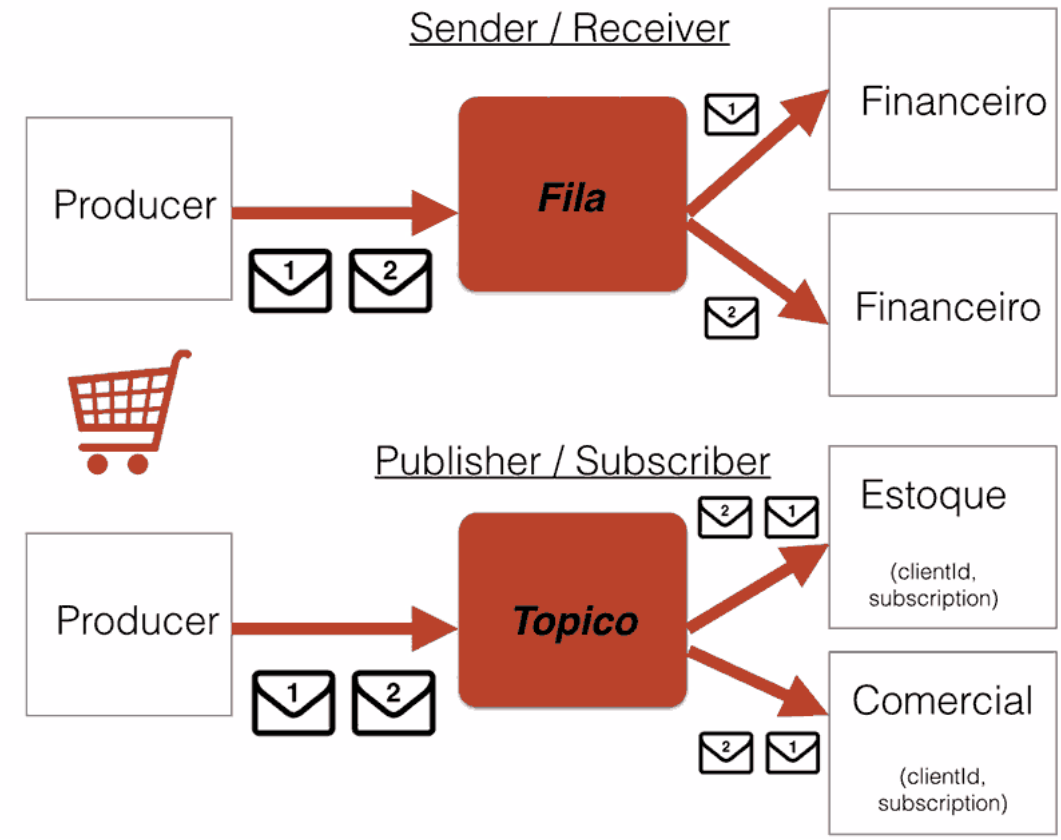
01

Prioridade e Tempo de vida da mensagem

Transcrição

A motivação inicial do uso do JMS era a nossa loja virtual onde é criado um pedido a partir de uma compra. Com o pedido em mãos, precisávamos nos integrar com outros sistemas e utilizamos um MOM para desacoplar nossos sistemas, sendo assim, a geração de um e-book e o processamento no sistema financeiro podem acontecer assíncronamente.

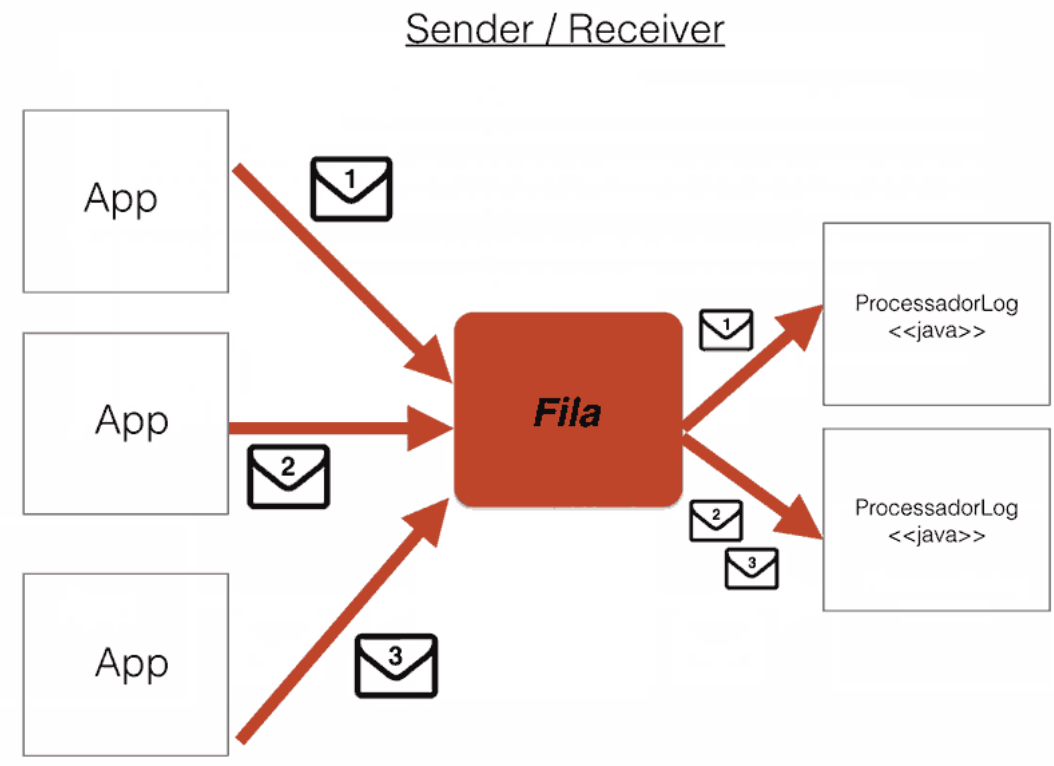
Dentro desse contexto, vimos dois modelos de entrega: a fila e o tópico. Na hora que estamos enviando a mensagem para o MOM, não há diferença entre fila e tópico, porém quando nosso MOM entrega nossas mensagens a coisa muda de figura. Enquanto a fila envia exatamente para apenas um, o tópico espalha a mensagem. Neste capítulo, vamos mudar um pouco o nosso caso de uso.



Um novo problema: vários logs

Arquivos de logs são comuns em sistema e é muito provável que todos aqui já tenham visto um. Alguns fornecem informações sobre o sistema outros o que está sendo executado e por aí vai. Cada aplicação de cada sistema gera seu arquivo de log e se quisermos saber o que esta acontecendo com cada um deles precisaríamos olhar diversos arquivos. A ideia agora é centrar o acesso aos arquivo de log em um único lugar.

Em nossa infraestrutura teremos uma aplicação com a responsabilidade de analisar os logs. Todos as aplicações enviaram uma mensagem JMS para um fila `log`, deixaremos de trabalhar com pedidos aqui. Os processadores agora poderão analisar os logs, tudo centralizado.



O problema é quando os processadores estão offline. A fila continuará a receber as mensagens de log de todas as aplicações gerando um volume muito grande de informações. Precisamos pensar como a fila lidará com esse grande volume de

informações.

Definindo o que é importante

Nossos log's possuem informações mais importantes e menos importantes. Mensagens menos importantes são as `INFO`. Vamos começar por elas.

Podemos informar para nosso ActiveMQ que mensagens de erro `INFO` podem ter um tratamento diferente.

Vamos alterar nosso `TesteProdutoFila.java`. No método `message.send`, vamos passar mais algumas parâmetros:

```
// TesteProdutorFila
// código anterior omitido
Message message = session.createTextMessage("INFO | ....");
producer.send(message, DeliveryMode.NON_PERSISTENT);
// código posterior omitido
```

Persistente vs não persistente

O padrão é `DeliveryMode.PERSISTENT` fazendo o ActiveMQ guardar a mensagem em um banco de dados. Caso ele tenha sido reiniciado, não haverá problema, pois ele recuperará a mensagem que foi salva. Mas no caso dos nossos log's do tipo INFO, não precisamos dessa persistência. Se o ActiveMQ reiniciar e perdermos essas informações, não será de grande importância.

Prioridade da mensagem

O próximo parâmetro é a prioridade. Podemos usar um número inteiro de 0 ate 9, na menor prioridade para a maior. Quando temos várias mensagens na fila, o ActiveMQ consegue reorganizá-la em termos de prioridade.

```
// código anterior omitido
producer.send(message, DeliveryMode.NON_PERSISTENT, 3);
// código posterior omitido
```

E o último parâmetro é o tempo de vida. Se ninguém consegue consumir a mensagem, podemos deixá-la viva por algum tempo e se esse tempo expirar, a mensagem é descartada. Vamos usar 5 segundos, mas em milissegundos:

```
// código anterior omitido
producer.send(message, DeliveryMode.NON_PERSISTENT, 3, 5000);
// código posterior omitido
```

Habilitando prioridades em activemq.xml

Podemos saber ainda mais sobre a parte de prioridade na própria documentação online do ActiveMQ (http://activemq.apache.org/how-can-i-support-priority-queues.html (http://activemq.apache.org/how-can-i-support-priority-queues.html)). Lá, vemos que por padrão o ActiveMQ não respeita a prioridade das mensagens, precisamos ativar esse recurso em sua configuração.

Precisamos colocar no XML do ActiveMQ a configuração:

```
<destinationPolicy>
 <policyMap>
   <policyEntries>
     <policyEntry queue=">" prioritizedMessages="true"/>
   ...
```

Vamos editar o arquivo `apache-activemq-5.12.0/conf/activemq.xml` e adicionar a configuração:

```
<policyEntry queue=">" prioritizedMessages="true"/>
```

Com a alteração feita, basta enviarmos uma mensagem e verificarmos no admin web do AtiveMQ se ela está na fila.

Agora, vamos **parar o ActiveMQ** subindo-o logo em seguida para que leve em consideração a nova configuração. Se olharmos a mensagem ela não está mais lá, porque não é persistente.

Agora, vamos enviar algumas mensagens com diferentes prioridades e vê-las no admin o ActiveMQ (*INFO, DEGUG, ERROR*).

Basta executar nosso consumidor e ver a ordem na qual ele consumirá as mensagens.