01

Enviando mensagens e Competição entre Consumidores

Transcrição

Começando deste ponto? Você pode fazer o [DOWNLOAD][1] completo do projeto do capítulo anterior e continuar seus estudos a partir deste capítulo.

No capítulo anterior falamos como consumir mensagens com JMS. Vimos como criar uma `Connection`, `Session`, trabalhar com `Consumer` e tratar as mensagens com o `MessageListener`. Nosso objetivo neste capítulo é trabalhar com o envio de mensagens para a nossa fila.

Criando o MessageProducer

Também precisaremos de `ConnectionFactory`, `Connection`, `Session` e um `Destination` portanto podemos reaproveitar esse código do nosso `TesteConsumidor` com a diferença que precisaremos de um `MessageProducer` em vez de um `MessageConsumer`!

```
IntialContext context = new InitialContext();
ConnectionFactory factory = (ConnectionFactory) context.lookup("ConnectionFactory");

Connection connection = factory.createConnection();
connection.start();
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

Destination fila = (Destination) context.lookup("financeiro");
MessageProducer producer = session.createProducer(fila);
```

Bem parecido com um `MessageConsumer` não? Com o `MessageProducer` em mãos podemos chamar o método `send` que recebe uma instância de alguma das subinterfaces de `Message`, por exemplo `TextMessage` ou `ObjectMessage`, etc.

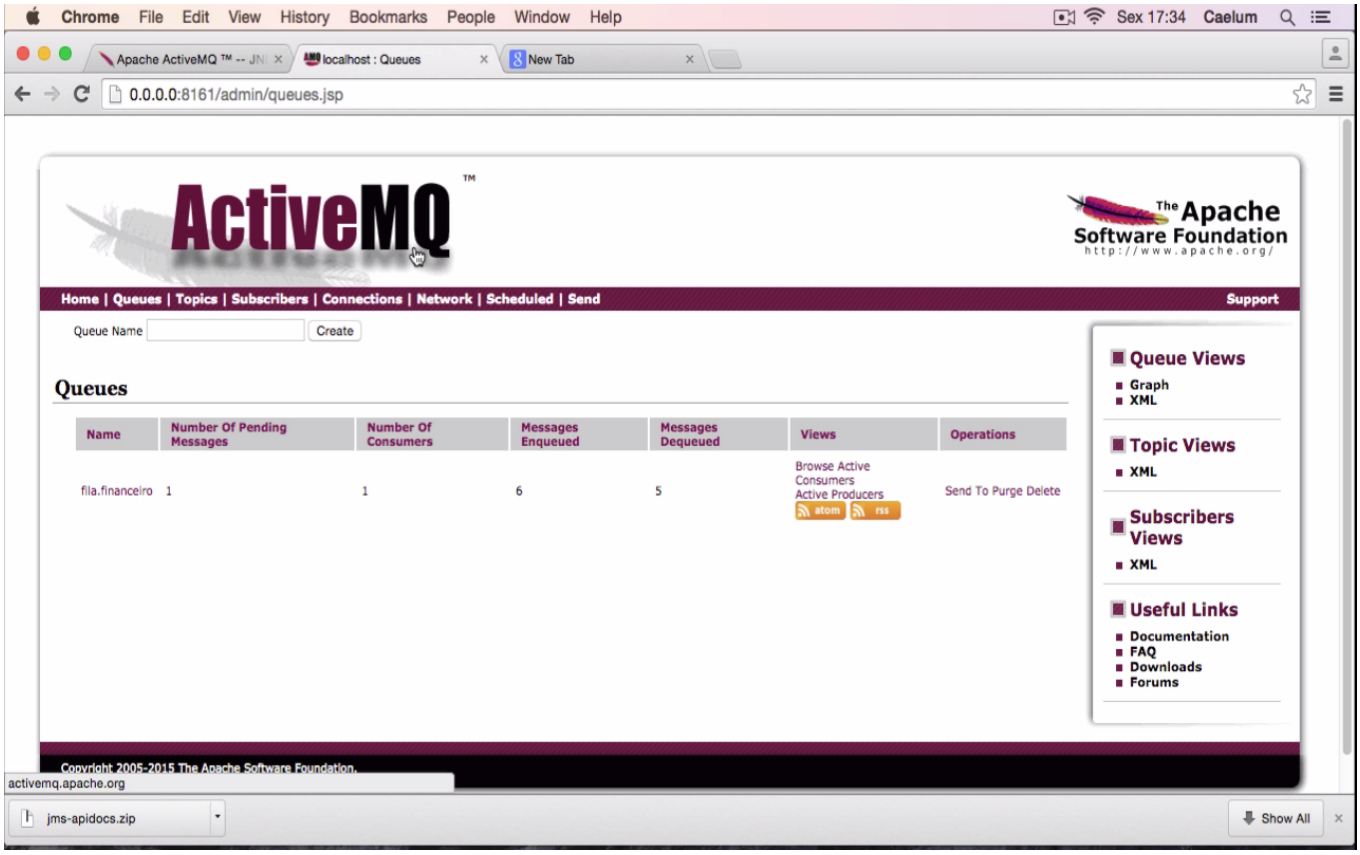
```
MessageProducer producer = session.createProducer(fila);
producer.send(...);
```

Para criar a mensagem, usamos a `Session` que é o objeto que conhece os detalhes de criação dos nossos componentes JMS. Podemos fazer algo assim:

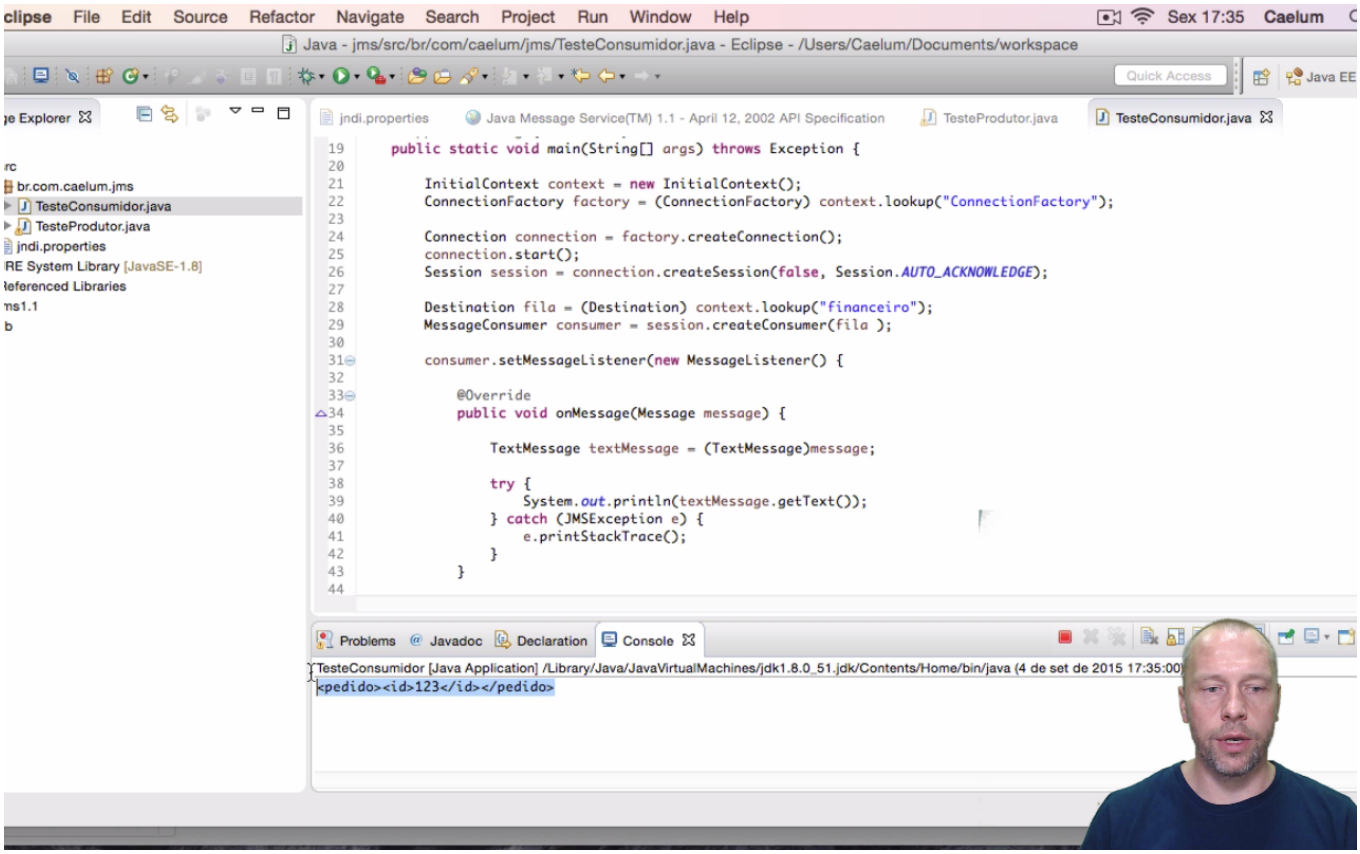
```
MessageProducer producer = session.createProducer(fila);
Message message = session.createTextMessage("<pedido><id>123</id></pedido>");

producer.send(message);
```

Pronto! Agora que tudo está pronto, vamos enviar e checar no painel do ActiveMQ se o envio ocorreu no ActiveMQ:



Podemos ver agora que algumas mensagens já foram enfileiradas. Vamos então iniciar nosso `Consumer` e ver que ele receberá o nosso XML:



Competição entre Consumidores

Perfeito! Vamos agora entender um pouco mais sobre como o *MOM* distribui as mensagens para os consumidores, para isso faremos um pequeno *teste de stress* no nosso produtor. Vamos criar um laço para enviar mil mensagens à fila para analisar como as mensagens serão distribuídas:

```
MessageProducer producer = session.createProducer(fila);

for(int i = 0; i < 1000; i ++) {
    Message message = session.createTextMessage("<pedido><id>" + i + "</id></pedido>");
    producer.send(message);
}
```

Após executarmos esse código concluímos que todas as mensagens chegaram corretamente, perfeito! O que testaremos agora é colocar dois consumidores para a mesma fila e analisarmos o modelo de entrega caso haja mais de um consumidor online. Ou seja, vamos executar mais uma máquina virtual rodando o `TesteConsumidor` e então vamos rodar novamente o `TesteProdutor` para enviar as mil mensagens.

Repare que o primeiro consumidor recebe apenas as mensagens de id **0, 2, 4, 6, ...** e não recebe mais todas as mensagens como no exemplo anterior. E o segundo consumidor recebe justamente as mensagens que o primeiro não recebeu. Ou seja, como a fila funciona? Ela entrega as mensagens apenas para um consumidor. Quando chega a mensagem, se há dois consumidores online (como no nosso caso) essa mensagem é entregue apenas para um dos dois (e nunca para os dois). Isso que o ActiveMQ faz é o que chamamos de **balanceamento de carga**.

Vamos testar agora com mais um consumidor online. Execute novamente a `TesteConsumidor` e reenvie as mensagens (`TesteProdutor`). Perceba que os consumidores agora recebem mensagens sempre de três em três. O primeiro recebe **0, 3, 6, ...** e o último **2, 5, 8, ...**. Ou seja, foi feito um balanceamento de carga. Ele realmente entrega uma mensagem apenas para **um dos três** (e nunca para os três). Esse é a ideia do modelo de distribuição de **fila**.