11

Recebendo o e-mail assíncrono Parte 1

Transcrição

Estamos bem próximos de finalizar e vermos o resultado. Falta-nos uma última configuração que é a do *Destination*, para que nosso servidor entenda que o destino é um tópico e para onde ele será enviado para todo mundo que se registra nele. Além do emissor/produtor da mensagem, temos também aquele que escuta a mensagem, o *listener* e precisamos de alguém no meio que faça essa comunicação. Precisamos lembrar que este é um processo assíncrono, ou seja, quem está enviando a mensagem não o faz diretamente para o receptor, mas sim para alguém que está no meio e recebe esta mensagem. Este que está no meio precisa saber quem precisa ouvir a mensagem. Quem precisar receber deverá avisar quem está no meio falando que quer receber, assim ele sai enviando para cada um dos listeners. É assim que funciona o JMS. Este destino central é conhecido como *destination config* ou *JMS Destination*.

Para que isso funcione precisamos de uma Classe, que chamaremos de `ConfigureJMSDestination`, dentro do pacote de configuração `br.com.casadocodigo.loja.conf`:

```
package br.com.casadocodigo.loja.conf;

public class ConfigureJMSDestination {

}
```

Essa Classe de configuração não terá nenhum código, não temos como fugir disso. Estamos ganhando em configuração, uma vez que não precisamos abrir XML, mas alguns códigos ficam estranhos. Precisaremos configurar um `@JMSDestinationDefinition`, o qual será um *array* de definições:

```
package br.com.casadocodigo.loja.conf;

import javax.jms.JMSDestinationDefinition;

@JMSDestinationDefinition(
    name="java:/jms/topics/CarrinhoComprasTopico",
    interfaceName="javax.jms.Topic"
)
public class ConfigureJMSDestination {

}
```

É no `interfaceName` que avisamos que é um tópico de fato. Dessa forma temos nossa configuração de destino. Se subirmos o servidor aparecerão alguns erros, sendo um deles um WARN referenciando `destinationType=null`. Isso acontece porque também precisamos configurar o *listener*. Dentro do `@MessageDriven` fazemos:

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(
        propertyName="destinationLookup",
        propertyValue="java:/jms/topics/CarrinhoComprasTopico"),
    @ActivationConfigProperty(
        propertyName="destinationType",
```

```
            propertyValue="javax.jms.Topic")
    })
```

Subindo o servidor novamente, não retorna nenhum erro.

Antigamente o Wildfly utilizava um framework JMS de fila chamado HornetQ, porém ele foi passado para o grupo da Apache. Agora o Wildfly está usando o ActiveMQ.

Vamos testar nossa aplicação fazendo uma operação de compra colocando no campo de e-mail o "alura.springmvc@gmail.com". Logo mais veremos porque deu erro.