07

Entendendo o message driven

Transcrição

Vamos tentar capturar a mensagem que estamos enviando de maneira que possamos, de fato, enviar o e-mail. Para isso usaremos a Classe `EnviaEmailCompra`. Ela será bastante modificada em relação a como está agora. ela tem uma configuração diferente para que se consiga receber as mensagens. Já discutimos alguns conceitos do JMS, como o assíncrono que o servidor executa e aquele que liberamos o usuário para que outras operações sejam executadas em background. Focaremos agora em resolver de fato o *listener*. A primeira implementação será anotar a Classe com `@MessageDriven()`:

```
import javax.ejb.MessageDriven;

@MessageDriven()
public class EnviaEmailCompra {
    //...
}
```

O pacote dele é "javax.ejb". O `@MessageDriven()` é um EJB especial através do qual conseguimos escutar uma determinada mensagem,podendo ser uma fila ou um tópico. Precisamos indicar quem ele vai, de fato, ouvir, ou seja, o `Destination`. Passamos o caminho já configurado anteriormente como a propriedade `activationConfig` e a anotation `@ActivationConfigProperty` com chave e valor. Este é o mesmo do `@Resource`:

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

O `propertyValue` é aquele que "ouviremos" e, como nesse caso estaremos ouvindo um tópico e para o `EnviaEmailCompra` não faz diferença aquilo que ele estará ouvindo, só precisamos realmente conseguir receber a mensagem. Mas como o `@MessageDriven` sabe que a Classe `EnviaEmailCompra` possui o método `enviar(String uuid)`? Não tem como saber, portanto precisamos implementar isso utilizando o MessageListener do pacote javax.jms:

```
public class EnviaEmailCompra implements MessaListener {

    //...
}
```

Ocorrerá um erro na Classe, nos obrigando a implementar um método:

```
public class EnviaEmailCompra implements MessaListener {

    //...

    @Override
    public void onMessage(Message message) {
        //TODO Auto-generated method stub
```

```
    }
}
```

Nós só precisamos da assinatura dele, então o apagamos e substituímos a assinatura no lugar do método `enviar()`:

```
public void onMessage(Message message) {
    Compra compra = compraDao.buscaPorUuid(uuid);
    String messageBody = "Sua compra foi realizada com sucesso, com o número de pedido " + compr

    mailSender.send("compras@casacodigo.com.br", compra.getUsuario().getEmail(), "Nova Compra na
}
```

O que nós queremos dele agora é o *uuid*, o qual já estamos colocando para buscar. Porém o Message não vem como texto. Para isso precisamos adicionar um `TextMessage`, o queal irá nos dar uma mensagem específica em String:

```
public void onMessage(Message message) {
    TextMesssage textMessage = (TextMessage)message;
    //...
}
```

Só precisamos pegar o `TextMessage` para obter o texto:

```
public void onMessage(Message message) {
    TextMesssage textMessage = (TextMessage)message;

    Compra compra = compraDao.buscaPorUuid(textMessage.getText());
    //...
}
```

Dessa forma, de fato, estamos obtendo o uuid que estamos enviando na Classe `PagamentoService`. Precisamos fazer ainda o try/catch:

```
public void onMessage(Message message) {
    try {
        TextMessage textMessage = (TextMessage) message;

        Compra compra = compra = compraDao.buscaPorUuid(textMessage.getText());
        String messageBody = "Sua compra foi realizada com sucesso," + "com número de pedido " +

        mailSender.send("compras@casadocodigo.com.br, compra.getUsuario().getEmail(), "Nova Comp
    } catch (JMSException e) {
        e.printStackTrace();
    }
}
```

Em relação ao erro podemos logar a mensagem, modificar através de uma outra forma avisando que o e-mail não pode ser enviado, mas não há muito que fazer, pois é assíncrono. E, como falamos anteriormente, algo assíncrono não mandamos resposta para o usuário no mesmo instante. O usuário já foi embora, já terminou a compra, ou seja, talvez

nem veja a informação de compra. Então no `e.printStackTrace()` não conseguimos dar um retorno específico para ele. Vamos deixar do jeito que está.

O `MessageDriven` ouvirá o destino que é o mesmo local onde estamos enviando na Classe `PagamentoService`. Porém "jms/topics/CarrinhoComprastopico" não existe no servidor, não podemos nem afirmar que ele é de fato um tópico. Veremos como solucionar isso a seguir.