J530 - Enterprise JavaBeans

10 Message-driven Beans

Helder da Rocha (helder@acm.org)

O que é um Message-Driven Bean

- Bean guiado por mensagens, ou eventos
  - Um event-handler distribuído em forma de EJB
- MDBs consomem mensagens enviadas a filas e canais JMS
  - Não é possível enviar uma mensagem diretamente a um message-driven bean (envia-se mensagens a um canal que o bean escuta)
  - Acoplamento entre clientes e MDB resume-se à conhecimento de canal comum de comunicações
- Um message-driven bean é um EJB mas
  - Não possui interfaces Home ou Remote
  - Possui apenas um método que recebe qualquer tipo de mensagem
  - Não devolve tipos ou exceções ao cliente
  - Não participa de contexto transacional iniciado no cliente
  - Não possui estado (é stateless)
  - Pode ser ouvinte de uma fila ou assinante durável ou não-durável de um canal (Topic) JMS

Desenvolvendo MDB

- *Message-driven beans implementam duas interfaces*

```
public interface javax.jms.MessageListener
public interface javax.ejb.MessageDrivenBean
                    extends javax.ejb.EnterpriseBean
```

- *Métodos de MessageDrivenBean e da especificação*
  - **`ejbRemove()`**: *método da interface. Chamado pelo container quando o message-driven bean está sendo removido do pool*
  - **`ejbCreate()`**: *método definido na especificação (não está na interface). É chamado quando um novo bean é criado pelo container*
  - **`setMessageDrivenContext(MessageDrivenContext ctx)`**: *método da interface. Chamado antes do ejbCreate() quando o bean está sendo adicionado ao pool*
- *Método de MessageListener*
  - **`onMessage(Message m)`**: *chamado cada vez que uma mensagem é enviada para o canal do bean (se o bean estiver ativado).*

Exemplo: implementação de um MDB

```
import javax.ejb.*;
import javax.jms.*;

public class LogBean implements MessageDrivenBean, MessageListener {

    protected MessageDrivenContext ctx;
    public void setMessageDrivenContext(MessageDrivenContext ctx) {
        this.ctx = ctx;
    }
    public void ejbCreate() {}
    public void ejbRemove() {}

    public void onMessage(Message msg) {
        TextMessage tm = (TextMessage) msg;
        try {
            String text = tm.getText();
            System.err.println("Received new message : " + text);
        } catch (JMSException e) {
            throw new EJBException();
        }
    }
}
```

Métodos da interface MessageDrivenBean ou do contrato com a especificação

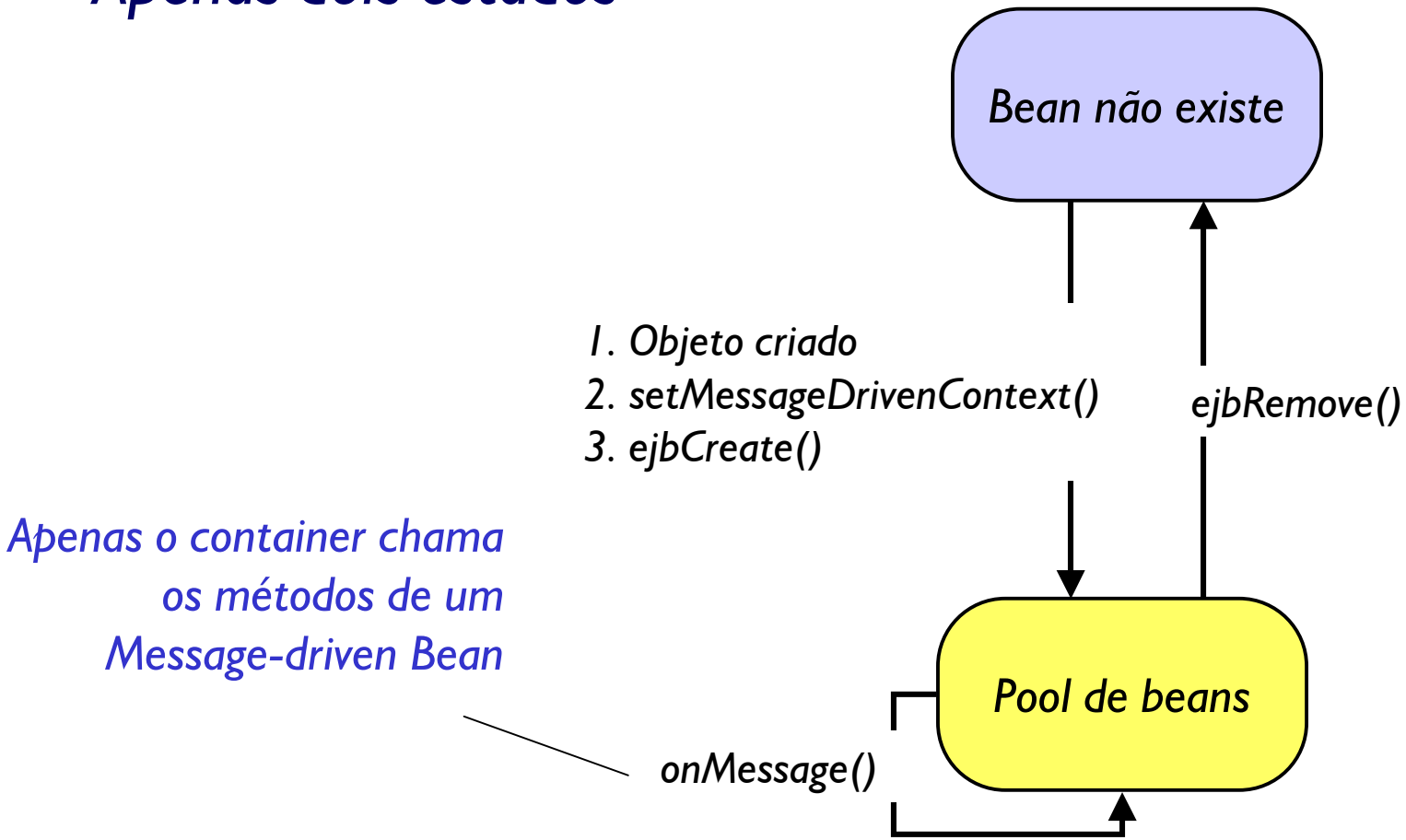
Método da interface MessageListener

RuntimeException para container capturar! Só faça isto se houver chance de corrigir erro: falta de acknowledgement pode causar loop infinito!

Este bean simplesmente imprime o texto contido na mensagem recebida

Ciclo de Vida

- Ciclo de vida muito simples
  - Apenas dois estados



Fonte: [1]

Deployment Descriptor

```
<ejb-jar>
  <enterprise-beans>
    <message-driven>

      <ejb-name>LogEJB</ejb-name>
      <ejb-class>examples.LogBean</ejb-class>
      <transaction-type>Container</transaction-type>

      <message-driven-destination>
        <destination-type>javax.jms.Topic</destination-type>
        <subscription-durability>NonDurable</subscription-durability>
      </message-driven-destination>

    </message-driven>
  </enterprise-beans>
</ejb-jar>
```

Canal (apenas a classe e tipo)

Usado apenas em Topics

Outros sub-tags de <message-driven>

- <message-selector>
  - Usado para incluir um filtro SQL para cabeçalhos de mensagens. Apenas mensagens que combinarem com o filtro serão recebidas pelo bean. Exemplo:

```
<message-selector><![CDATA[
      Formato LIKE '%Imagem%'
      AND JMSExpiration > 0
      AND Valor IS NOT NULL
]]></message-selector>
```

- <acknowledge-mode>
  - Permite definir o modo de acknowledgment. Pode ser Auto-acknowledge ou Dups-ok-acknowledge. O último permite o recebimento de mensagens duplicadas
  - Client-acknowledge não é suportado

Vendor-specific: JBoss

- *Os nomes JNDI dos recursos que serão acessados globalmente são definidos na configuração do fabricante.*
- *No JBoss:*

```
<jboss>
    <enterprise-beans>
        <message-driven>
            <ejb-name>LogEJB</ejb-name>
            <destination-jndi-name>
              topic/testTopic
            </destination-jndi-name>
        </message-driven>
    </enterprise-beans>

    <resource-managers>
       <resource-manager>
          <res-name>jms/logtopic</res-name>
          <res-jndi-name>topic/testTopic</res-jndi-name>
       </resource-manager>
    </resource-managers>
</jboss>
```

Nome JNDI global do destino compartilhado pelo bean e pelos clientes

Referência do ENC java:comp/env

Mapeamento do nome do ENC para nome JNDI global

Cliente é um cliente JMS comum

- *Pode ser outro bean, cliente Web ou standalone*

```
public class Client {

    public static void main(String[] args) throws Exception {

        Context ctx = new InitialContext(System.getProperties());
        TopicConnectionFactory factory =
             (TopicConnectionFactory) ctx.lookup("ConnectionFactory");
        TopicConnection connection =
             factory.createTopicConnection();
        TopicSession session =
             connection.createTopicSession(
                  false, Session.AUTO_ACKNOWLEDGE);

        Topic topic = (Topic) ctx.lookup("topic/testTopic");
        TopicPublisher publisher = session.createPublisher(topic);
        TextMessage msg = session.createTextMessage();
        msg.setText("This is a test message.");
        publisher.publish(msg);
    }
}
```

Cliente externo (fora do ENC) usa nomes JNDI globais

Execução dos exemplos

- Diretório cap10/exemplos/mejb2/
- Para montar e instalar a aplicação, use

```
> ant jboss.deploy
```

- Uma vez instalada, rode o cliente:

```
> ant run.jboss.client
```

- A execução não deve causar nenhuma reação no cliente, já que foi assíncrona "one-way".
- No servidor, aparece a mensagem enviada:

```
[STDERR] Received new message : This is a test message.
```

- Veja também o exemplo do J2EE Tutorial (usando Queue) no subdiretório cap10/exemplos/sun/

Exercício

- 1. Implemente um bean que receba as mensagens enviadas pelo cliente que você desenvolveu no capítulo anterior
  - O nome JNDI da fila é queue/A
  - O bean deve filtrar as mensagens e receber apenas aquelas cujo cabeçalho Quantidade contiver mais que 200 itens e menos que 400
- 2. Faça com que o Session Bean AdminLoja envie mensagens que serão recebidas por este bean
  - Defina a referência loja/jms/operacoes como o nome da fila no ENC (java:comp/env)
  - Mapeie o nome a fila queue/A do JBoss

Fontes

[1] *Ed Roman*. *Mastering EJB 2. Chapter 8 - Introduction to Message Driven beans*.

[2] *J2EE Tutorial*. *Message-driven Beans*.

[3] *Monson-Haefel*. *Enterprise JavaBeans. Chapter 13 - Message-driven beans*.

Curso J530: Enterprise JavaBeans

Revisão 2.0 - Junho de 2003

(helder@acm.org)

argonavis.com.br