Curso de Especialização em Desenvolvimento de Aplicações para Dispositivos Móveis e Cloud Computing

DM110

Desenvolvimento Java EE

Prof. Roberto Ribeiro Rocha
rrocha.roberto@gmail.com

Na aula de hoje

- Message-Driven Beans

Message-Driven Bean (MDB)

- Tipo de EJB usado para que as aplicações Java EE possam processar mensagens assincronamente
- Age como um listener de mensagens
  - Especificação JMS (Java Messaging System)
- Podem receber mensagens de uma fila ou tópico

Message-Driven Bean (MDB)

- Um MDB é similar a um Session Bean
  - Mas um cliente não o acessa via interface
- Todas as instâncias de um MDB são equivalentes.
- Um MDB pode processar mensagens de múltiplos clientes.

Características de um MDB

- É executado apenas quando recebe uma mensagem
- Execução assíncrona
- Possui vida relativamente curta
- Não representa diretamente o acesso a um BD
- Pode estar contido em uma transação
- É stateless
- Implementa a interface **`MessageListener`** → método **`onMessage(...)`**

JMS – Java Message Service API

- Permite o envio e recebimento de mensagens
  - Confiável, assíncrono e com baixo acoplamento
- Define um conjunto comum para comunicação
- Reduz o trabalho do programador
- Maximiza a portabilidade das aplicações
- Permite envio de mensagens sem ter uma resposta imediata

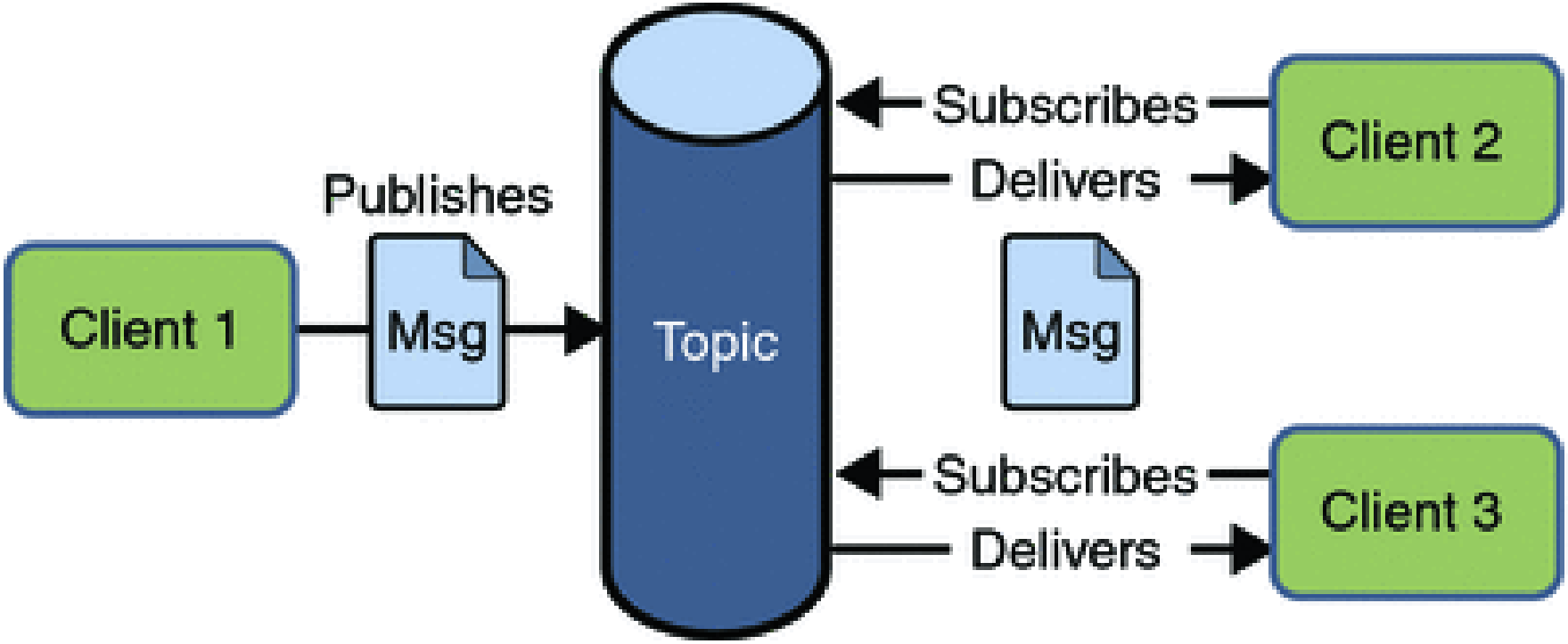
Exemplo: manufatura

1) O sistema de inventário envia uma mensagem para a fábrica quando o estoque está baixo.
2) A fábrica dispara uma mensagem para o setor de compras.
3) O setor de compras envia uma mensagem para o financeiro fazer o pagamento das novas peças.

Tópicos JMS

- Possui semântica *publish*/*subscribe*.
- Quando uma mensagem é publicada ela é recebida por todos os *subscribers* que tenham interesse.
- Zero ou mais *subscribers* receberão uma cópia da mensagem.
- Somente os *subscribers* ativos no momento em que a mensagem for publicada irão receber uma cópia da mensagem.

Tópicos JMS

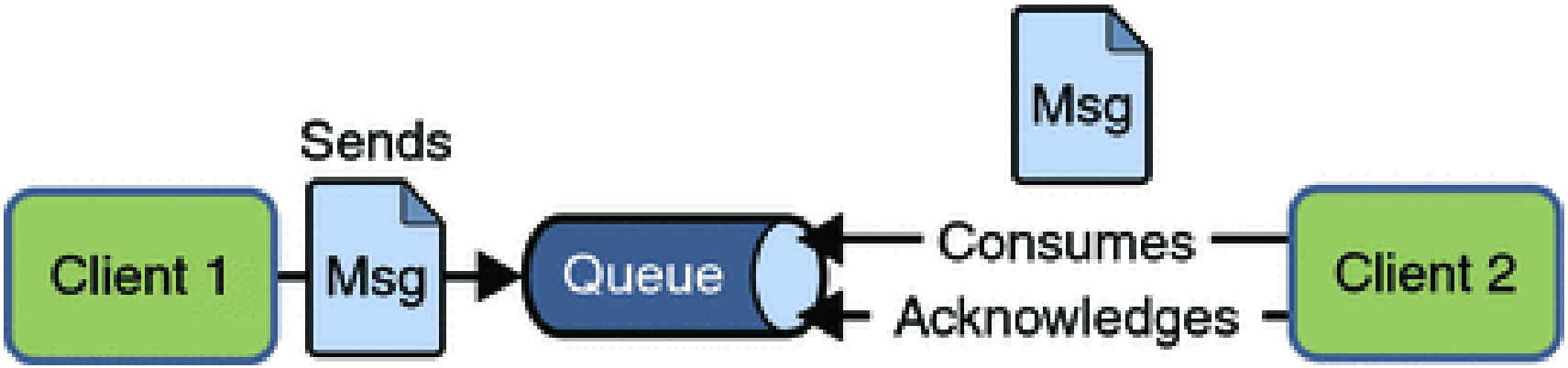


Fonte: https://docs.oracle.com

Filas JMS

- Implementa semântica de balanceamento de carga.
- Cada mensagem será recebida por exatamente um consumidor.
- Se nenhum consumidor estiver disponível, a mensagem será mantida na fila até que um consumidor possa processá-la.
- A fila pode ter muitos consumidores, mas as mensagens serão entregues de forma balanceada aos consumidores disponíveis.

Fila JMS

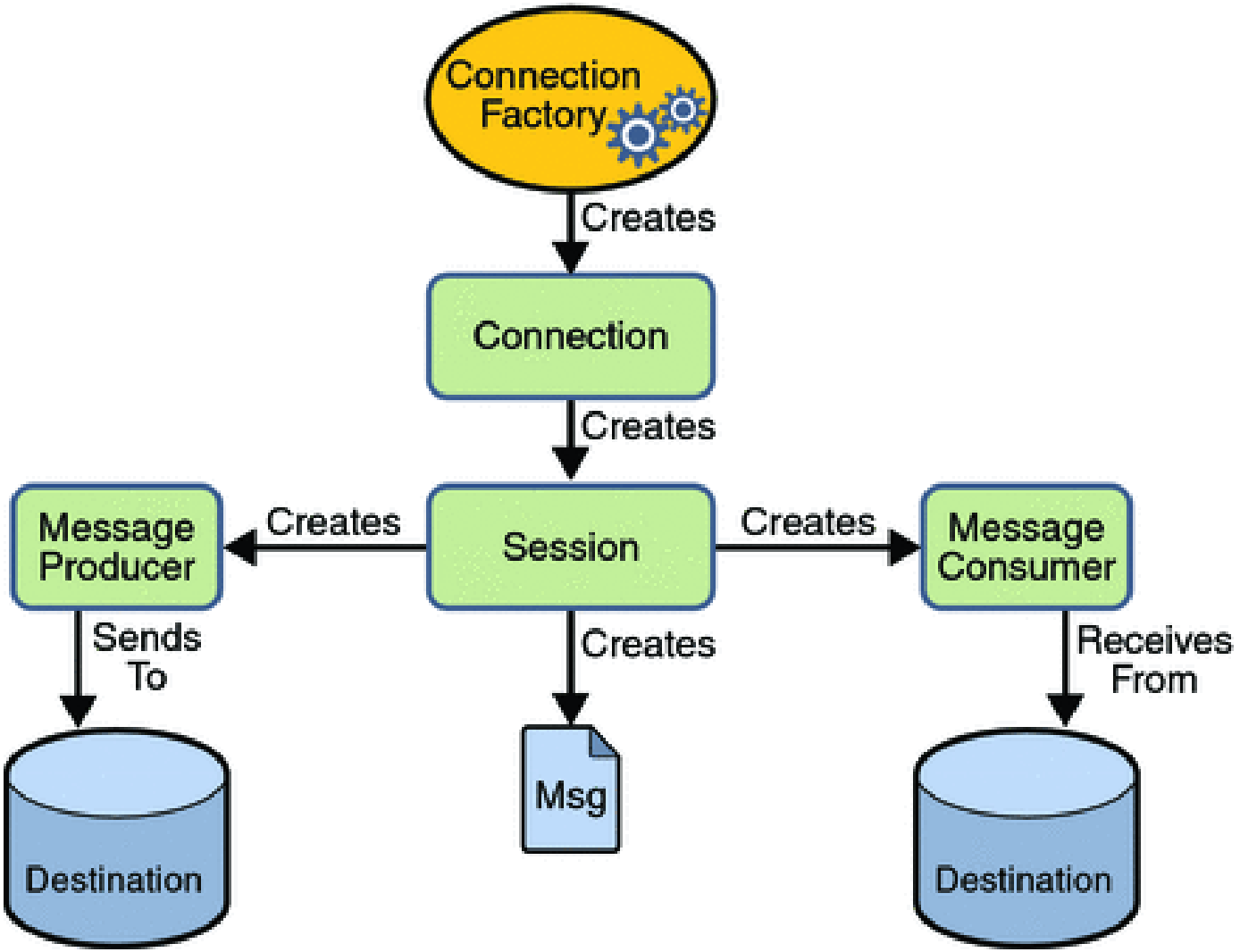


Fonte: https://docs.oracle.com

Exemplo de classe MDB

```
@MessageDriven(activationConfig = {
    @ActivationConfigProperty(propertyName = "destinationType",
                              propertyValue = "javax.jms.Queue"),
    @ActivationConfigProperty(propertyName = "destination",
                              propertyValue = "java:/jms/queue/dm110queue"),
    @ActivationConfigProperty(propertyName = "maxSession", propertyValue = "5")
})
public class HelloMDB implements MessageListener {
  @Override
  public void onMessage(Message message) {
    //processamento da mensagem
  }
}
```

Modelo de programação da JMS



Produzindo Mensagens

```
@Stateless
public class HelloMessageSender {
@Resource(lookup = "java:/ConnectionFactory")
private ConnectionFactory connectionFactory;
@Resource(lookup = "java:/jms/queue/dm110queue")
private Queue queue;
public void sendTextMessage(String text) {
  try ( Connection connection = connectionFactory.createConnection();
        Session session = connection.createSession();
        MessageProducer producer = session.createProducer(queue); ) {
    TextMessage textMessage = session.createTextMessage(text);
    producer.send(textMessage);
  } catch (JMSException e) { throw new RuntimeException(e);}
}
}
```

Enviando e recebendo “objetos”

- No transmissor:

```
ObjectMessage objMsg = session.createObjectMessage(stateTO);
producer.send(objMsg);
```

- No receptor:

```
public void onMessage(Message message) {
    if (message instanceof ObjectMessage) {
        ObjectMessage objMsg = (ObjectMessage) message;
        Object object = objMsg.getObject();
        if (object instanceof StateTO) {
            StateTO to = (StateTO) object;
            // usa o objeto “to”
```

Prática com JMS

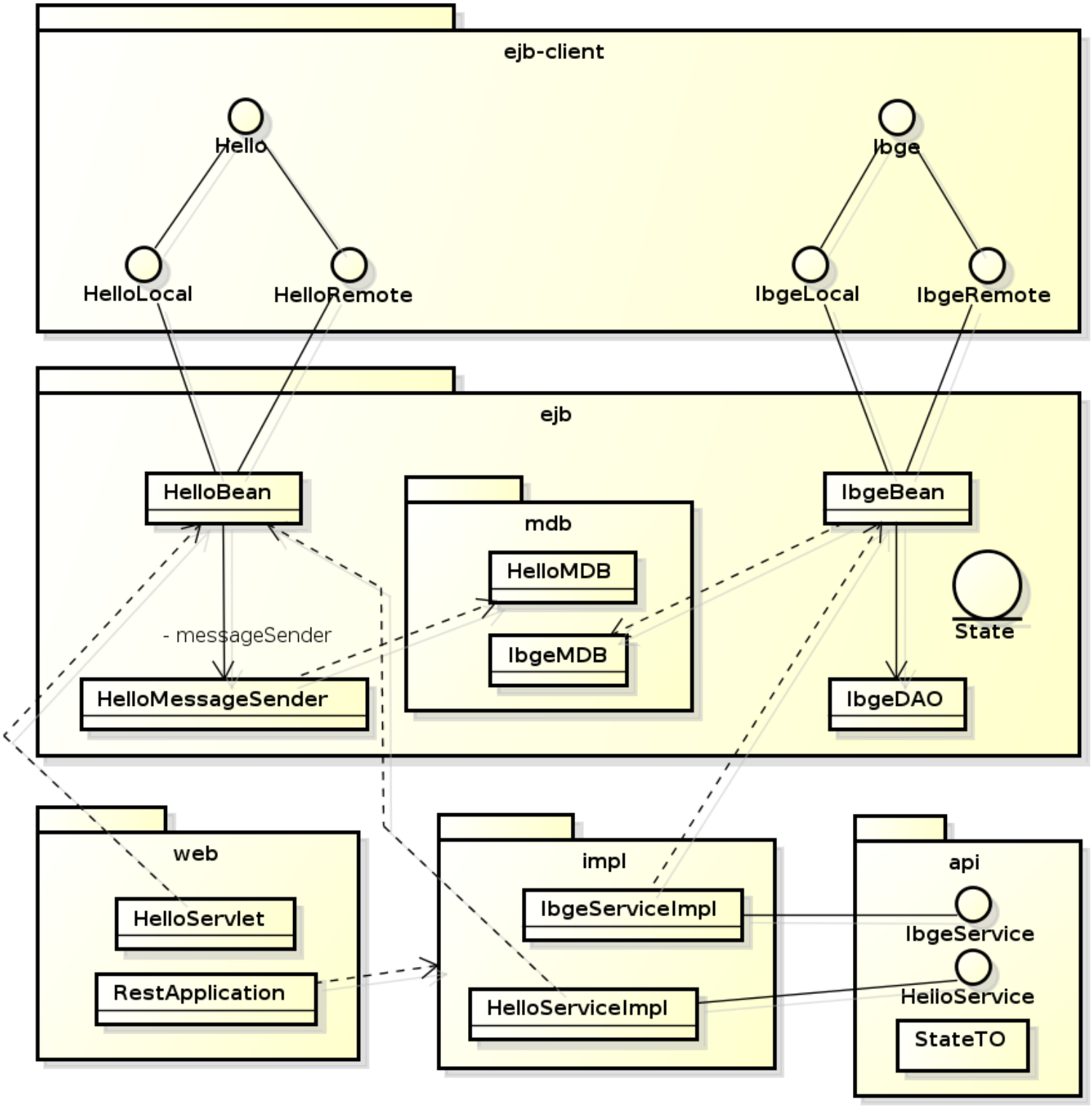
- Criar a classe que envia a mensagem
  - Chamar essa classe dentro do session bean
- Criar a classe que recebe a mensagem
- Configurar o JMS no servidor de aplicação
- Testar e verificar os logs.

Exercício

- Implementar e configurar um MDB para receber uma notificação quando um Estado é criado no sistema.
  - O MDB deve apenas imprimir as informações do estado no console.

Inatel

Visão geral das classes



Obrigado :)