Spring 5.0

Enterprise and Mobile

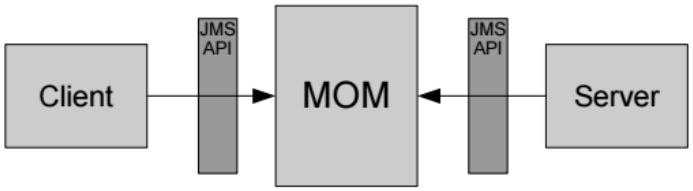
DE HOGESCHOOL
MET HET NETWERK

Hogeschool PXL – Dep. PXL-IT – Elfde-Liniestraat 26 – B-3500 Hasselt
www.pxl.be - www.pxl.be/facebook

Java Messaging Service

JMS

- Asynchroon messaging systeem
- Via tussenliggende server: middleware server
  - Message Oriented Middleware (MOM)
  - Voordelen:
    - Loose coupling
    - Asynchroon: zender (en ontvanger) kunnen doorgaan
    - Ontvanger moet niet online zijn
- Uniform binnen Java via JMS-API



Afbeelding 44: De JMS-API

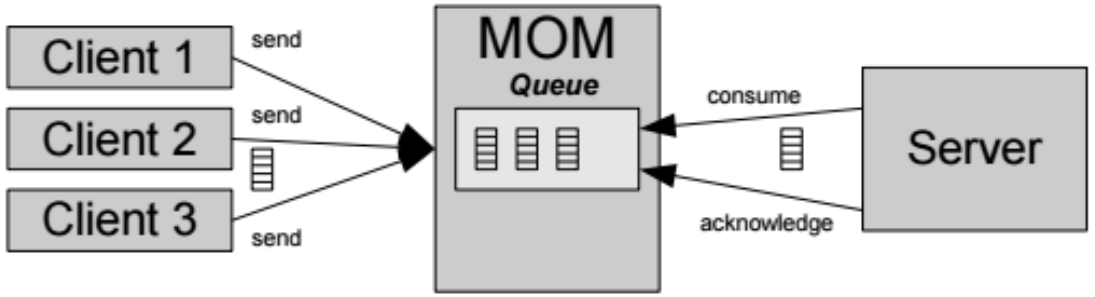
JMS Architectuur

- Twee opties
  - Point-to-Point domein
  - Publisher-Subsribe domein

PXL IT

JMS Point-to-Point

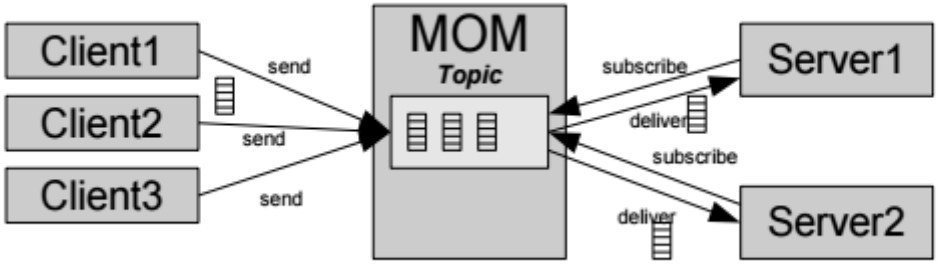
- Tussen twee vaste punten (via MOM)
- MOM houdt ***queue*** bij
- Client**s** sturen messages naar queue
- Server haalt messages van queue
- Message uit de queue na bevestiging door server



PXL IT

JMS Publisher-Subscriber

- MOM is ‘prikbord’ met ***topics***
- Berichten op topic geplaatst
- ALLE actieve subscribers halen bericht van topic



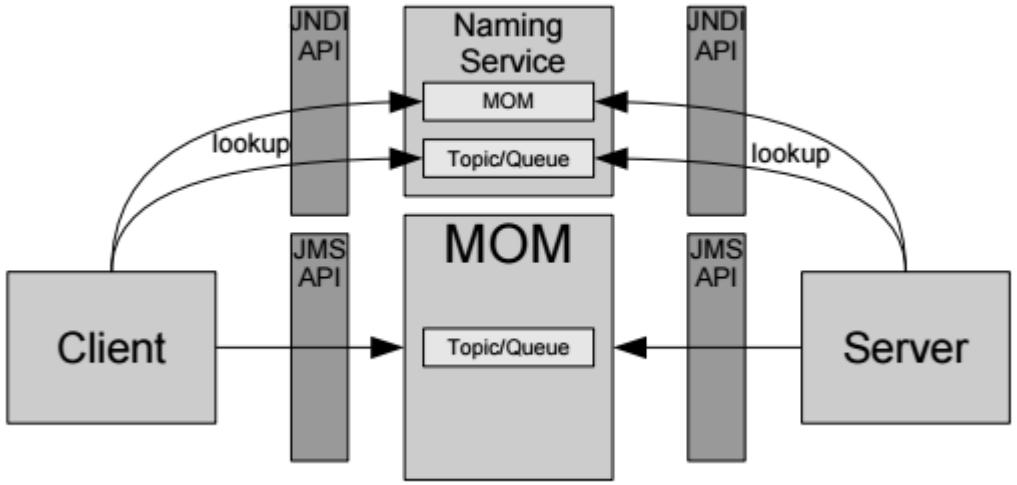
Synchroon vs Asynchroon

- Ontvanger kan pollen (synchroon)
  - Method die opvraag doet **blokkeert**
- Ontvanger registreert *listener* (asynchroon)
  - Via event op de hoogte gebracht (onMessage)

PXL IT

Naming service

- Client en server moeten toegang krijgen tot MOM (queues en topics)
- Opzoeking via JNDI (Java Naming and Directory Interface)

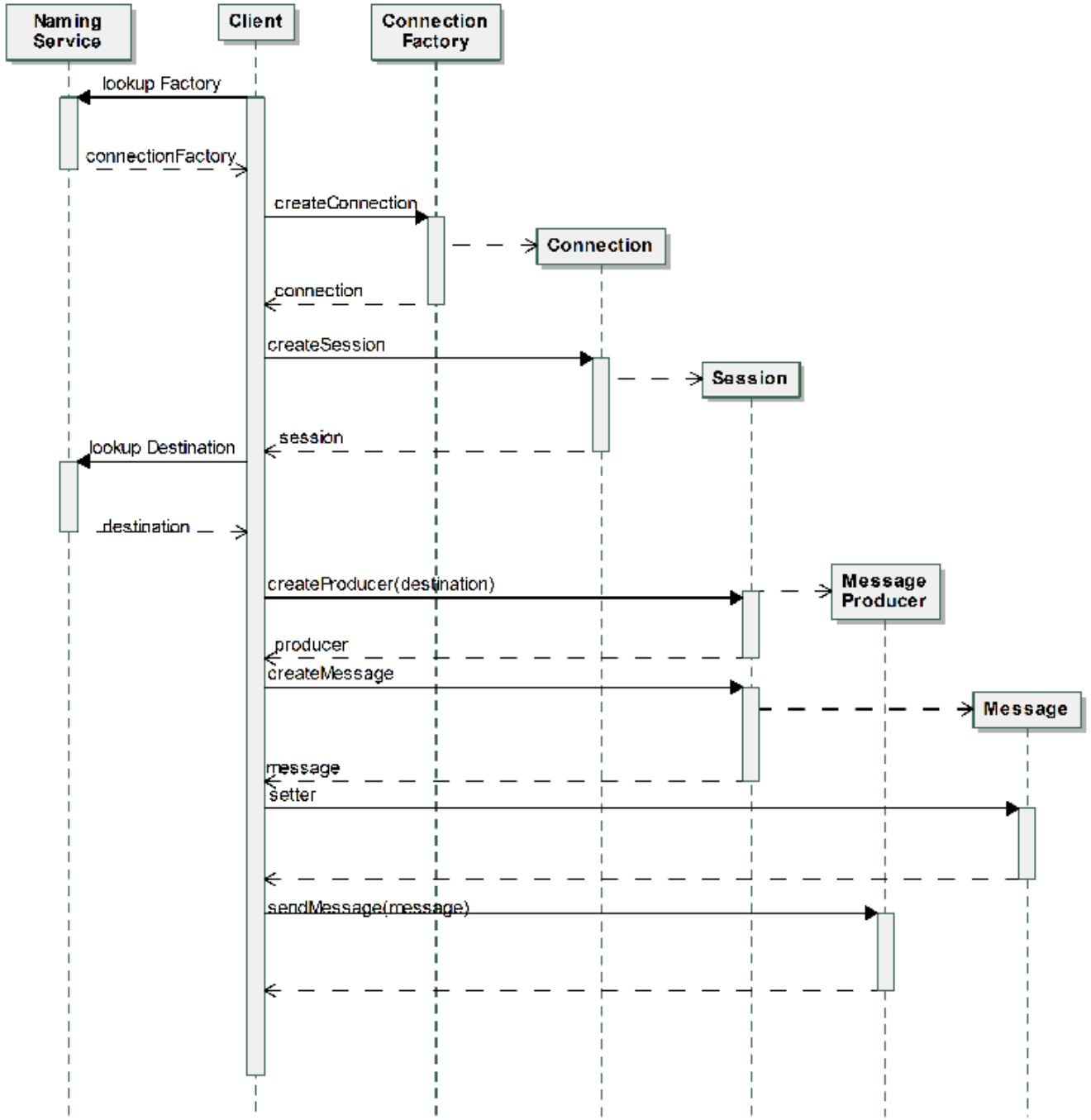


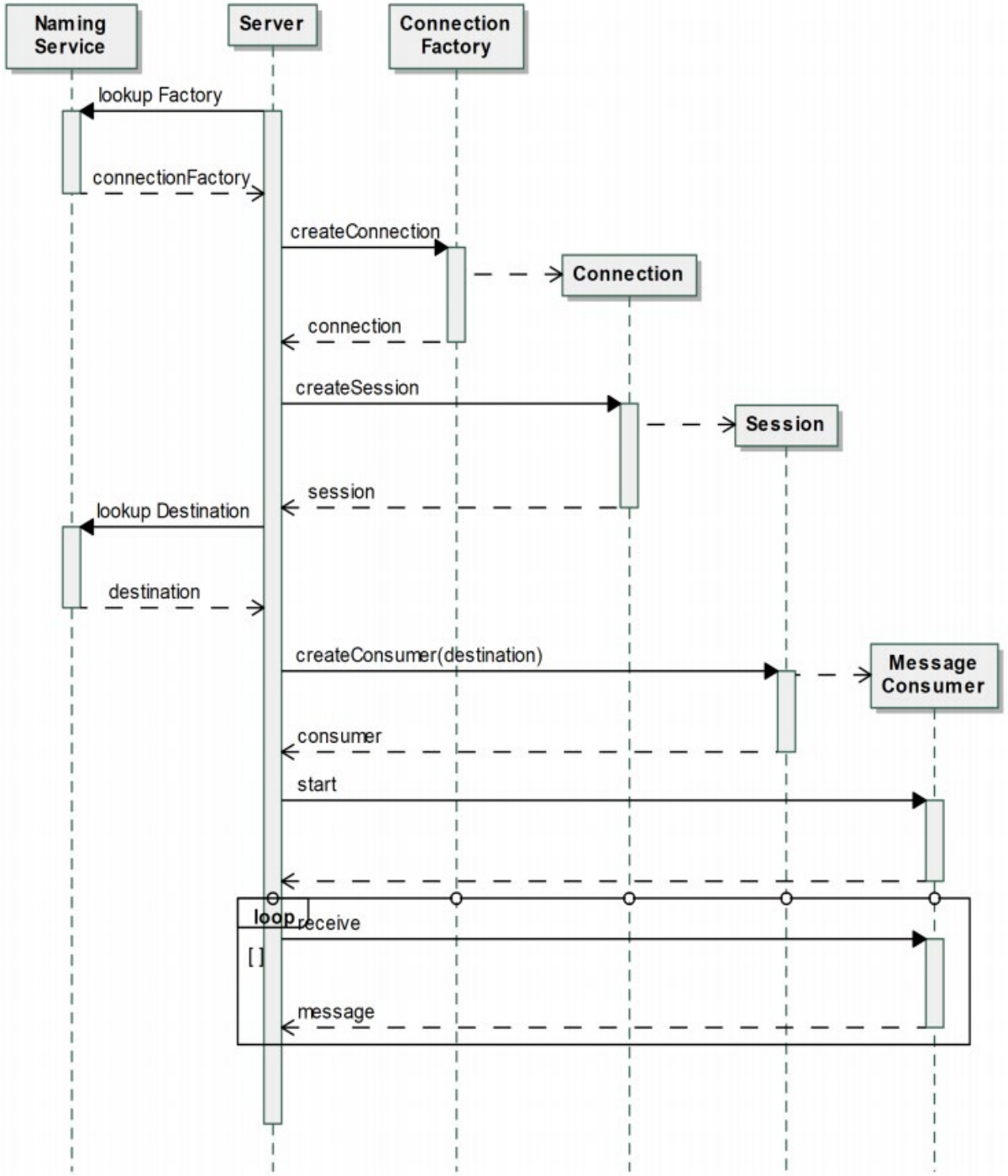
JMS API

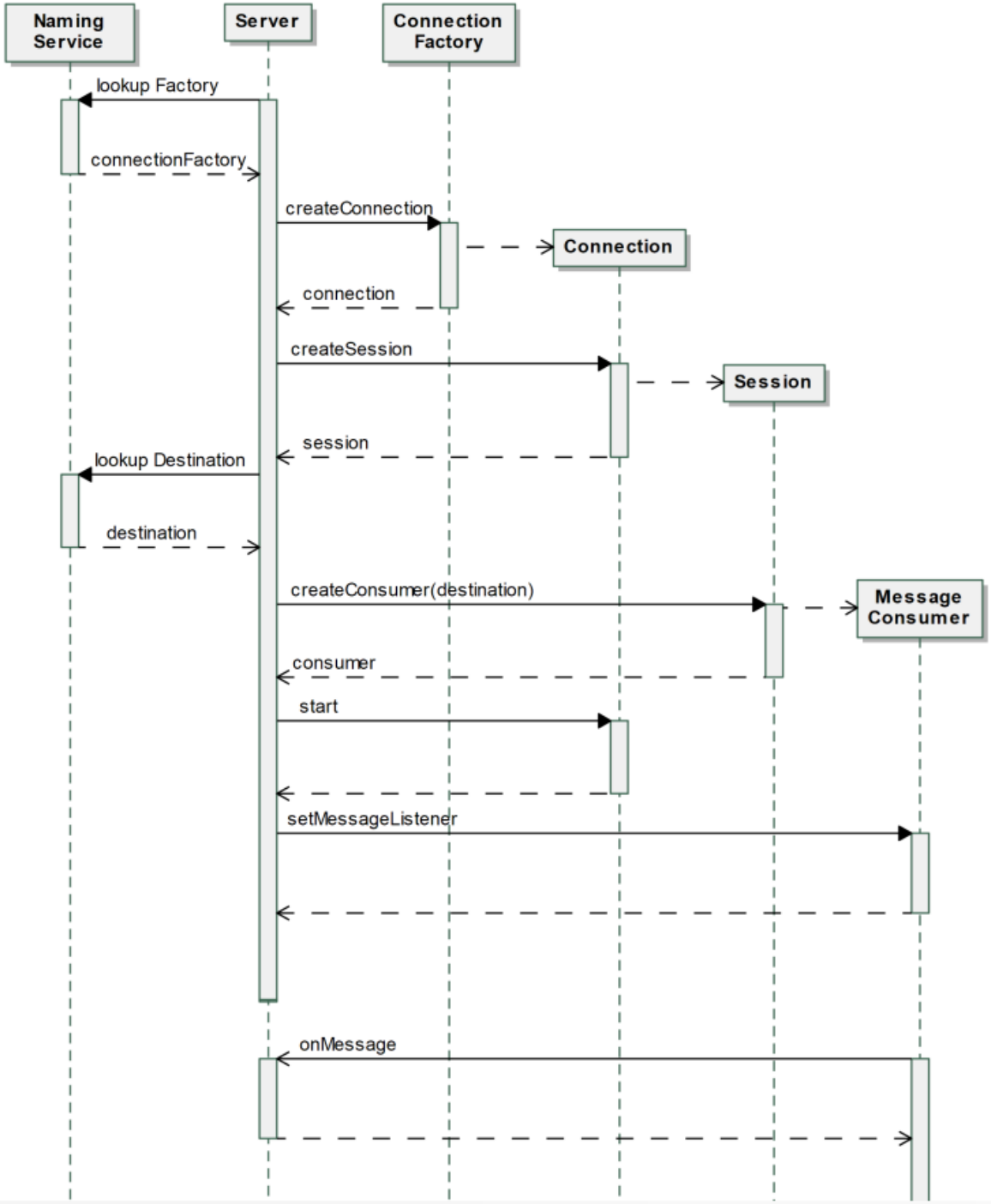
- Onderdeel van javax.jms (vooral interfaces)
- Twee objecten moeten geregistreerd worden
  - ConnectionFactory (connectie naar MOM)
  - Destination (Queue of Topic)

  -> via JNDI
- Verloop:
  - ConnectionFactory maakt Connection object (naar MOM)
  - Connection maakt Session (single threaded context)
  - Session maakt Producer of Consumer (en Messages)

PXL IT


Naming Service
Client
Connection Factory
lookup Factory
connectionFactory
createConnection
Connection
connection
createSession
Session
session
lookup Destination
destination
createProducer(destination)
Message Producer
producer
createMessage
Message
message
setter
sendMessage(message)


Naming Service
Server
Connection Factory
lookup Factory
connectionFactory
createConnection
Connection
connection
createSession
Session
session
lookup Destination
destination
createConsumer(destination)
Message Consumer
consumer
start
loop
receive
[ ]
message


Naming Service
Server
Connection Factory
lookup Factory
connectionFactory
createConnection
Connection
connection
createSession
Session
session
lookup Destination
destination
createConsumer(destination)
Message Consumer
consumer
start
setMessageListener
onMessage

JMS Message

- Object dat interface Message implementeert
- Hoofding
  - Identificatie velden (meeste automatisch)
- Properties (custom, optioneel)
- Body (eigenlijk bericht, optioneel)

Interface	***Omschrijving***
`Message`	Gewone boodschap zonder inhoud.
`TextMessage`	Boodschap met tekst (*string*) als inhoud.
`MapMessage`	Boodschap met als inhoud een set van naam-waarde-paren (map).
`BytesMessage`	Boodschap met als inhoud een reeks van bytes.
`StreamMessage`	Boodschap met als inhoud een *stream* van primitieve waarden.
`ObjectMessage`	Boodschap met als inhoud een geserialiseerd Java-object.

PXL IT

MOM

- ActiveMQ
  - http://activemq.apache.org/

```
#spring.activemq.broker-url=tcp://noelvaes.eu:61616
spring.activemq.broker-url=tcp://localhost:61616

spring.activemq.user=admin
spring.activemq.password=admin
```

PXL IT

JMS in Spring

- POM: **spring-boot-starter-activemq**
  - Spring Boot: Twee beans automatisch beschikbaar
    - ConnectionFactory
    - JmsTemplate
  - ConnectionFactory wordt geïnjecteerd in JmsTemplate!
    - JmsTemplate gebruiken in onze *beans*

PXL IT

JMS client in Spring

```
@Component
public class HelloSender {
    @Autowired
    private JmsTemplate jmsTemplate;

    public void sendHello(String text) {
        jmsTemplate.send("HelloQueue", new MessageCreator() {
            public Message createMessage(Session session)
                throws JMSException {
                return session.createTextMessage(text);
            }
        });
    }
}
```

PXL IT

JMS client in Spring

```
@Component
public class HelloSender {
    @Autowired
    private JmsTemplate jmsTemplate;

    public void sendHello(String text) {
        jmsTemplate.convertAndSend("HelloQueue",text);
    }
}
```

String → TextMessage
array → BytesMessage
Map → MapMessage
Serializable → ObjectMessage

PXL IT

JMS client in Spring

- Standaard Point-to-Point (Queue)
- Voor publish-subscribe:

```
jmsTemplate.setPubSubDomain(true);
jmsTemplate.convertAndSend("HelloQueue",text);
```

PXL IT

JMS server in Spring

- In de hoofdapp:
  - @EnableJms
    - Extra bean:
      - JmsListenerContainerFactory -> maakt DefaultMessageListenerContainer
  - Vervolgens eender welke bean:
    - @JmsListener(destination="EenQueue")

PXL IT

JMS server in Spring

```
@Service
public class HelloJmsReceiver {

    @JmsListener(destination="HelloTopic")
    public void onMessage(Message msg) {
        try {
            if(msg instanceof TextMessage) {
                String text = ((TextMessage)msg).getText();
                System.out.println("Hello " + text);
            }
        }
        catch (JMSException e) {
            e.printStackTrace();
        }
    }
}
```

JMS server in Spring

- Opnieuw: standaard Point-to-Point (Queue)
- Voor publish-subscribe (in hoofd app):

```
@Autowired
public void configure(DefaultJmsListenerContainerFactory fact) {
   fact.setPubSubDomain(true);
}
```

PXL IT

Links Messaging

- https://www.oracle.com/technetwork/articles/java/introjms-1577110.html
- http://activemq.apache.org

- .NET integration:
  - http://activemq.apache.org/nms/ (.NET Messaging API)

PXL IT