12.–15.09.2010
in Nürnberg

Herbstcampus

Wissenstransfer
par excellence

Ereignishaft

Event Driven Architecture mit .Net und Java

Thomas Haug

MATHEMA Software GmbH

About myself

- Senior Consultant, Architect and Trainer (MATHEMA Software GmbH)
- 12+ years Java Enterprise development
- 7+ years .Net development

- Main interests
  - Software architecture
  - Distributed systems
  - Object-Relational mapping

E-Mail: thomas.haug@mathema.de

AGENDA

- Motivation
- Tools Infrastructure
- Example
  - Simple Messaging
  - Routing & Enrichment
  - Multicasting
  - Scalability and Reliability
  - Availability
- Conclusion

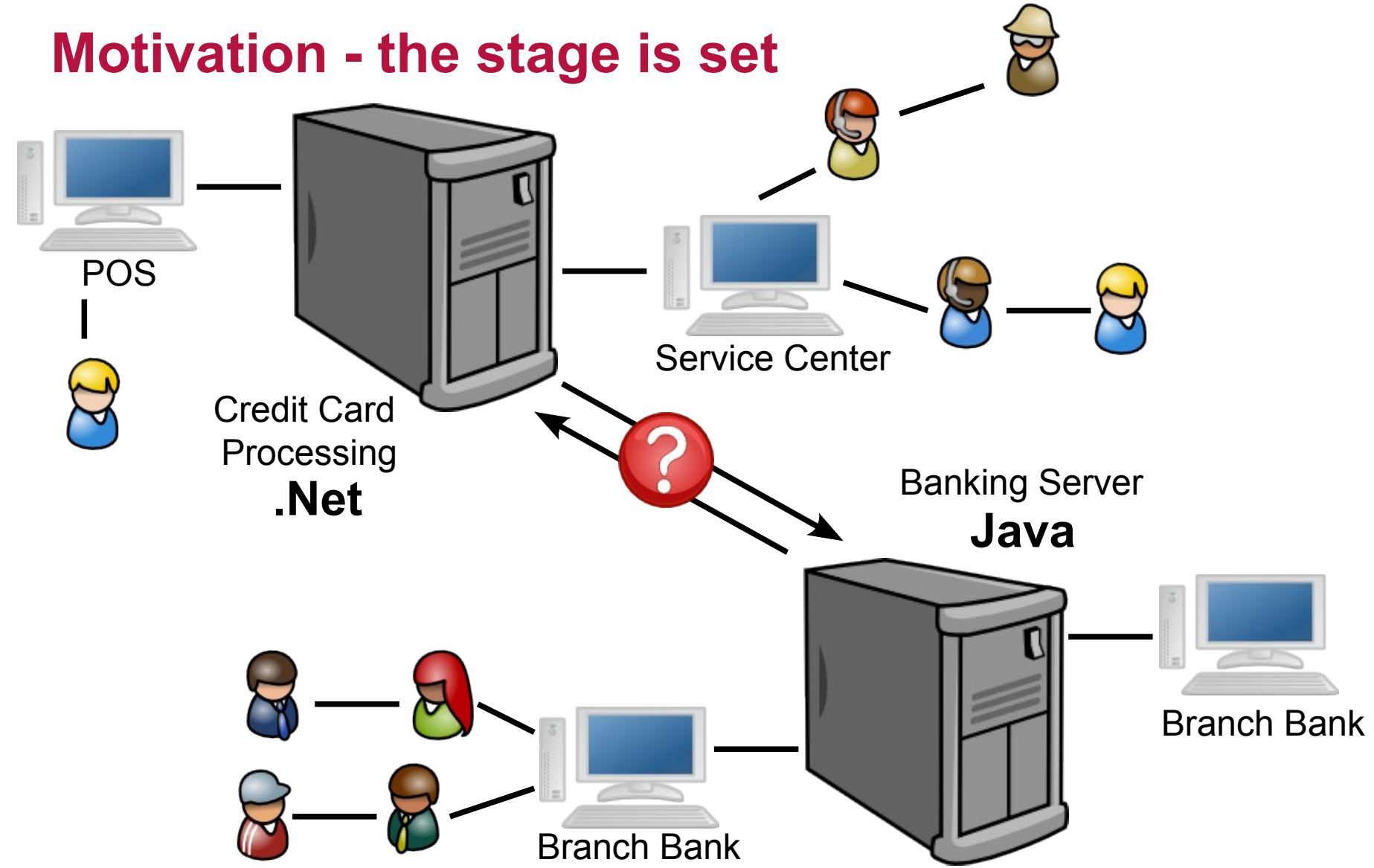
AGENDA

- Motivation
- Tools Infrastructure
- Example
  - Simple Messaging
  - Routing & Enrichment
  - Multicasting
  - Scalability and Reliability
  - Availability
- Conclusion

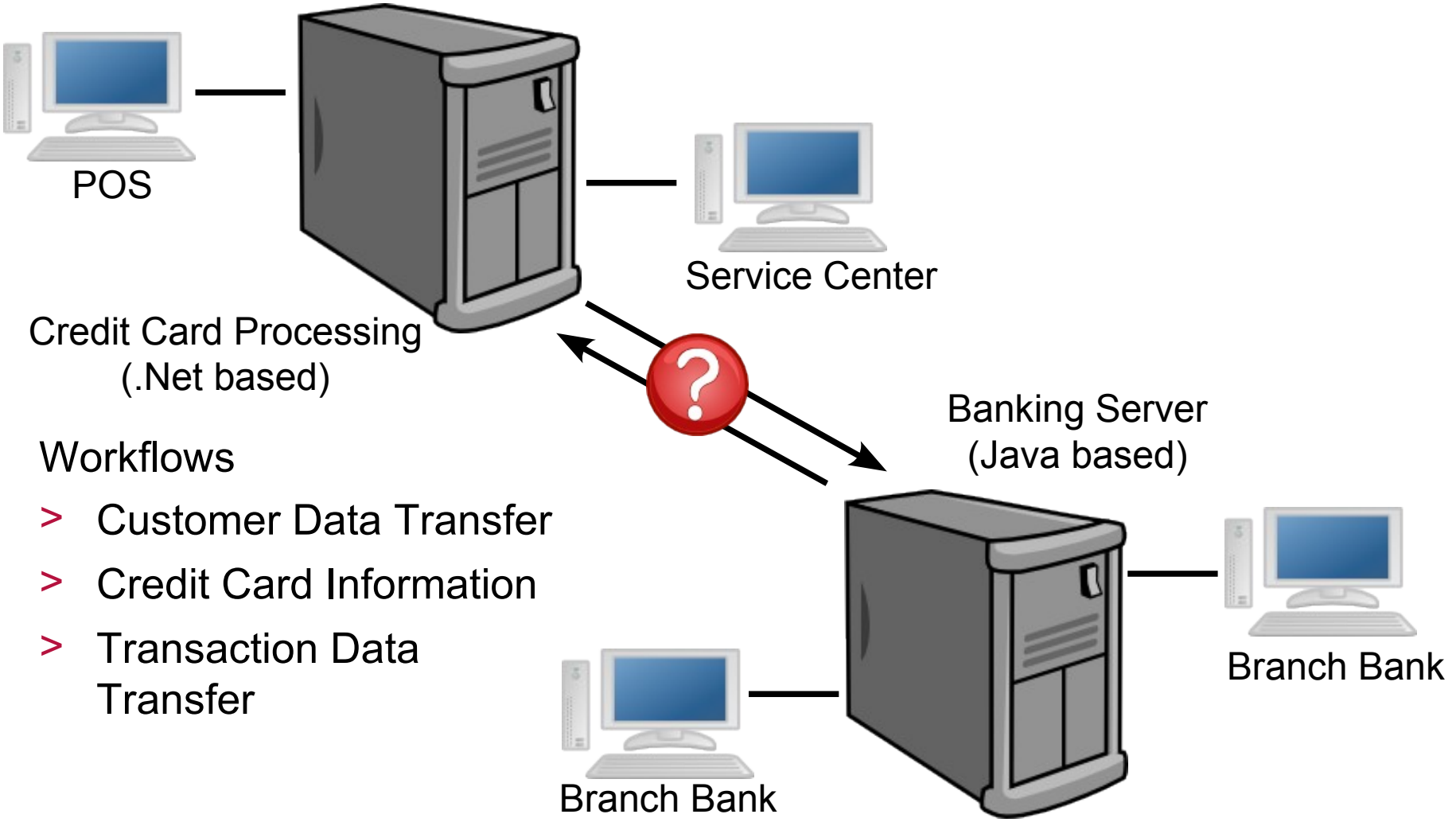
Motivation - Integration

Bridge of Millau, crossing the valley of Tarn

Motivation - the stage is set



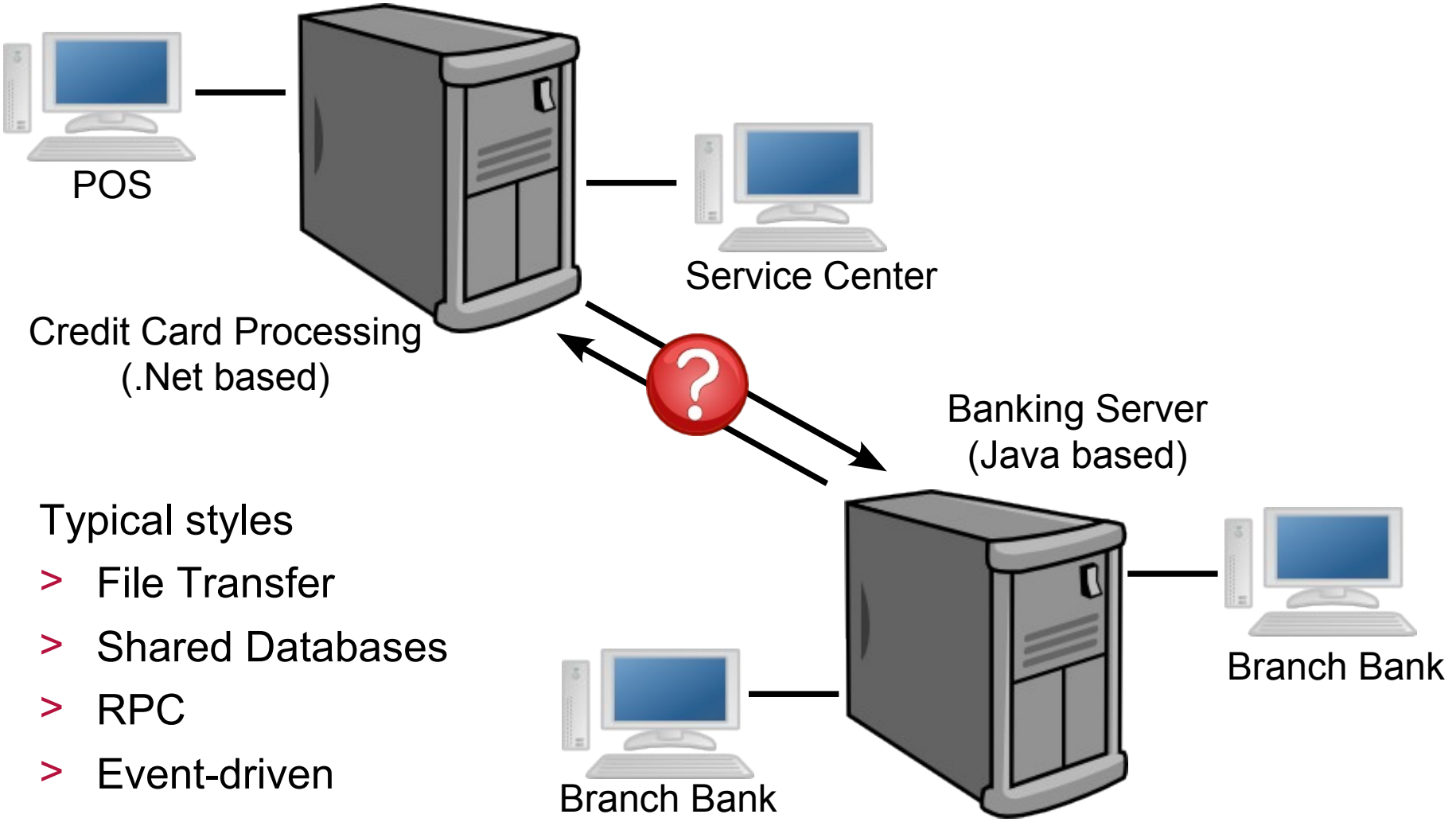
Motivation - Supported “Workflows“



Workflows

- Customer Data Transfer
- Credit Card Information
- Transaction Data Transfer

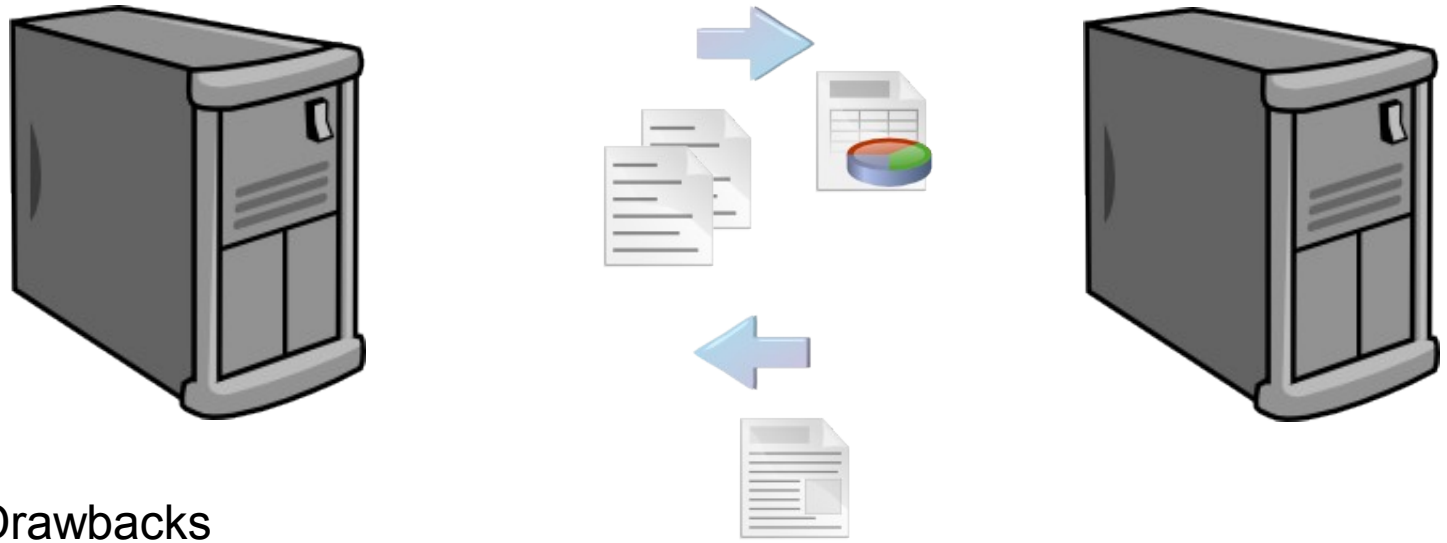
Motivation - Integration styles



Typical styles

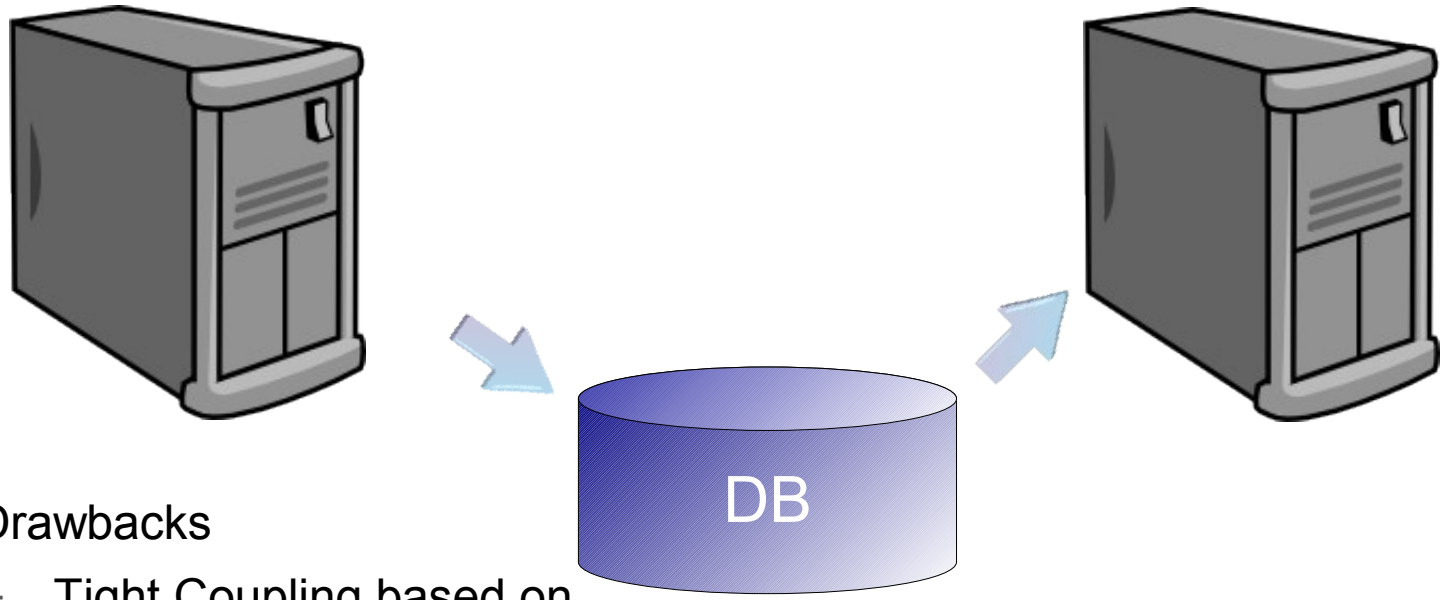
- File Transfer
- Shared Databases
- RPC
- Event-driven

Motivation - File Transfer style



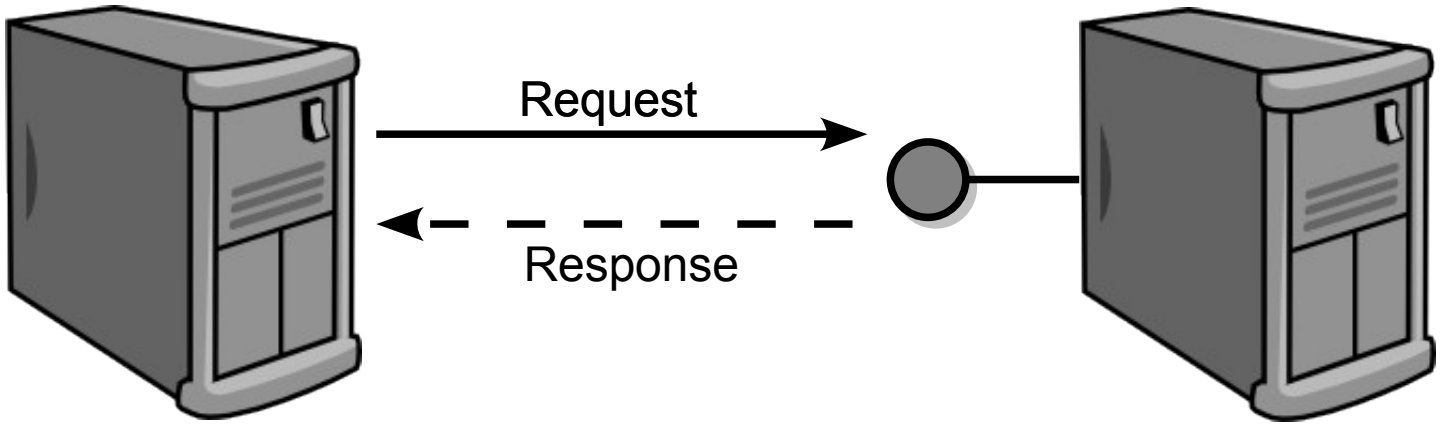
- Drawbacks
  - File naming conventions
  - Encoding of data
  - Updates tend to occur infrequently
  - How to detect / prevent lost files?
  - Security Issues

Motivation – Shared Databased



- Drawbacks
  - Tight Coupling based on Database schema
  - Difficult to define a unified schema
  - External programs accessing unified schema

Motivation - Remote Procedure Call style



- Drawbacks
  - Tight coupling to interface and location
  - Synchronous invocation
  - Extensibility is difficult to achieve

Motivation - Event-driven / Message-driven style

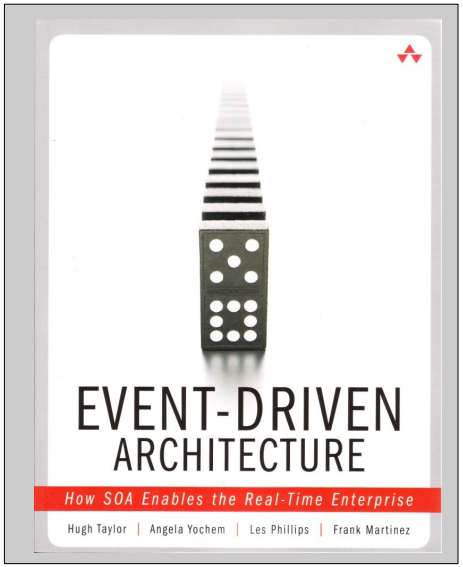
- Drawbacks
  - Complex programming model
  - Sequencing of messages
  - Monitoring of system might be difficult

Event Driven Architecture - Definition

- One Definition

  *"In an event-driven architecture, a **notable thing happens** inside or outside your business, which **disseminates immediately** to all interested parties (human or automated). The interested parties **evaluate the event**, and optionally **take action**."*

  (*Brenda M. Michelson* 2006)



- Messaging vs. Event-driven Architectures
  - Messaging is about routing, transformation and storage of messages.
  - Event-driven architecture are usually build on top of messaging frameworks.

Event Processing Styles

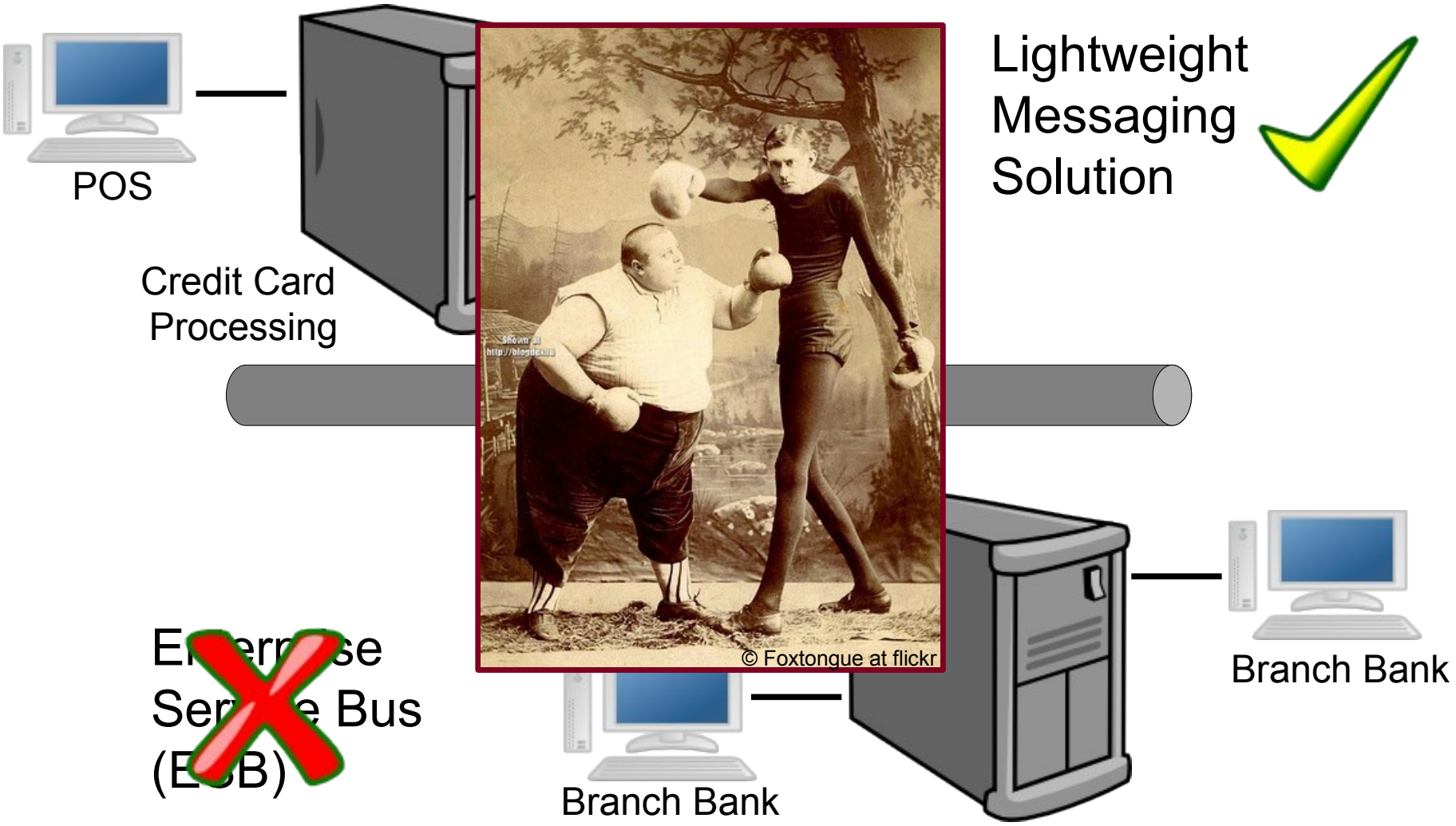
- Simple Event Processing
  - *a event initiates downstream action(s)*
- *Stream Event Processing*
  - *Consumer receives many event but reacts only to notable*
- *Complex Event Processing (CEP)*
  - *Consumer reacts on multiple events under certain logical conditions*
  - *correlation may be casual, temporal, or spatial.*

a

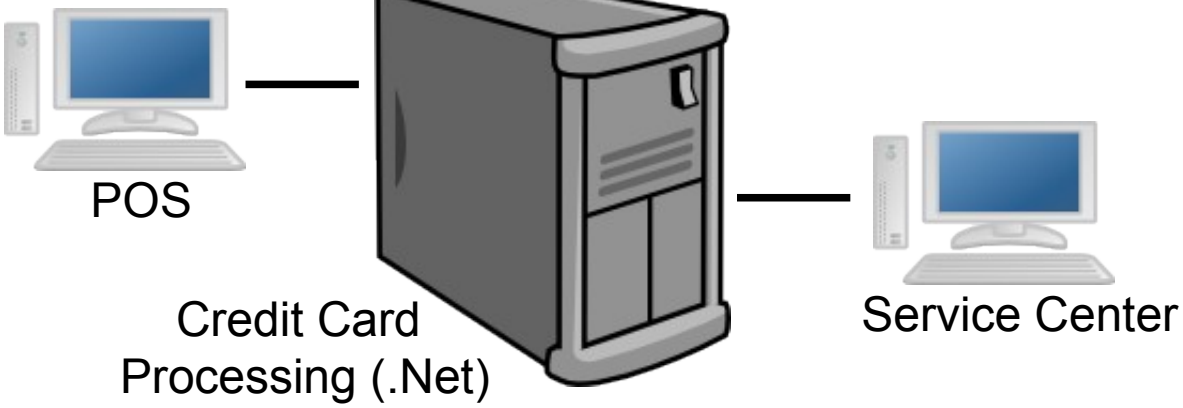
AGENDA

- Motivation
- Tools Infrastructure
- Example
  - Simple Messaging
  - Routing & Enrichment
  - Multicasting
  - Scalability and Reliability
  - Availability
- Conclusion

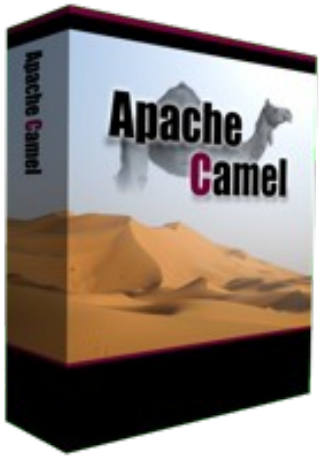
Example - Which Solution ?



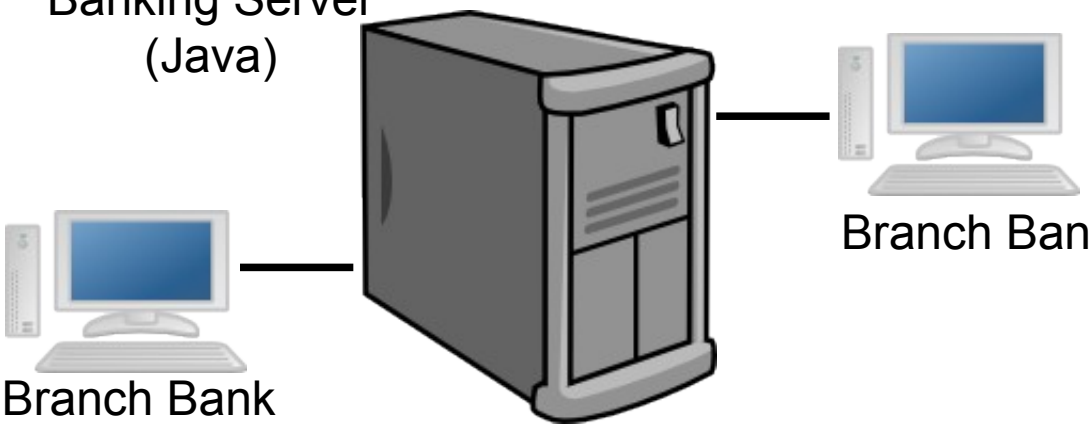
Example



Channel



Banking Server
(Java)



Apache ActiveMQ

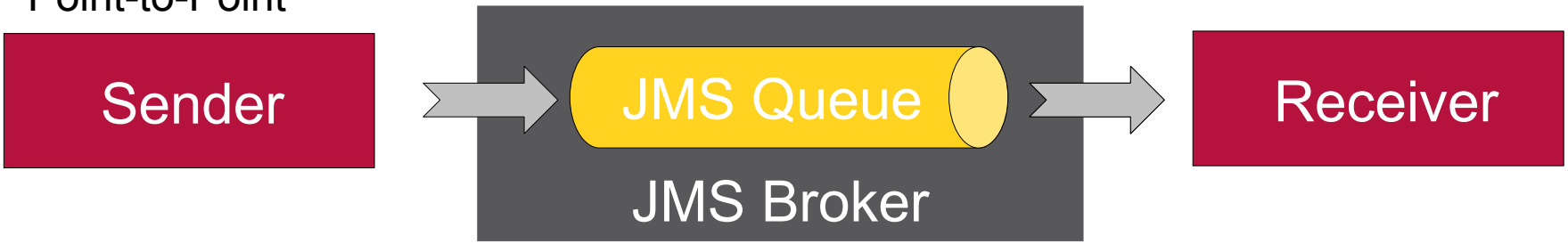
- ActiveMQ is an Enterprise Message Broker
  - Based on Java Messaging Service (JMS) 1.1
- Current Version 5.4.0
- Asynchronous communication
- Simple Messaging *(according to EDA definition)*
- Provides Quality of Service (QoS) like
  - Transaction handling
  - Guaranteed delivery
  - Location transparency

Apache ActiveMQ

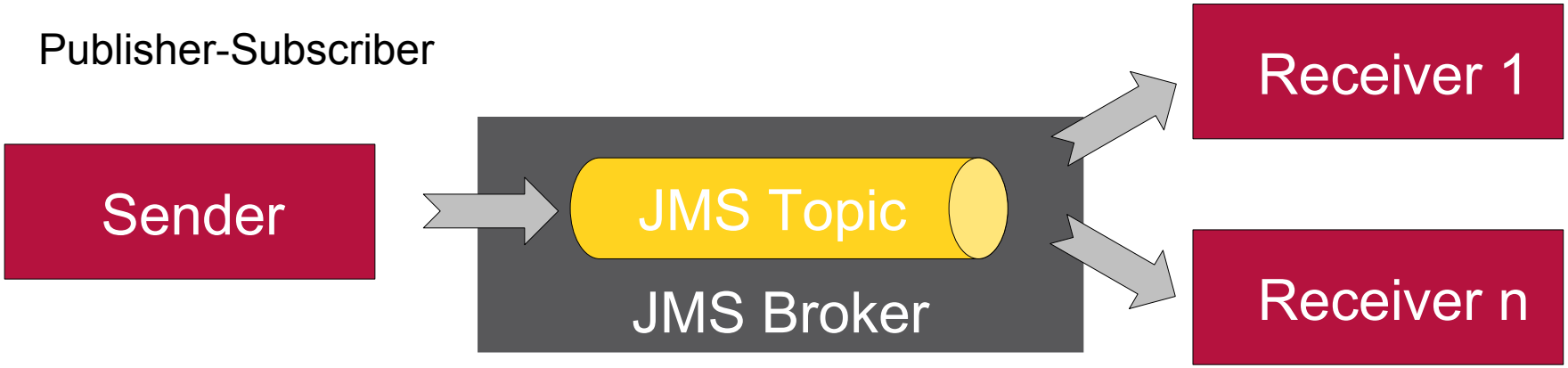
- Supports various Transport protocols
  - VM, TCP, NIO, UDP, HTTP
  - SSL, HTTPS
- Protocols
  - AMQP, OpenWire, REST, Stomp, WS Notification, Extensible Messaging and Presence Protocol (XMPP)
- Cross Platform Clients
  - C, C++, Perl, PHP, Pike, Python, Ruby
  - .Net Messaging API (NMS)
- Integrated Camel Support

Apache ActiveMQ - JMS Recap

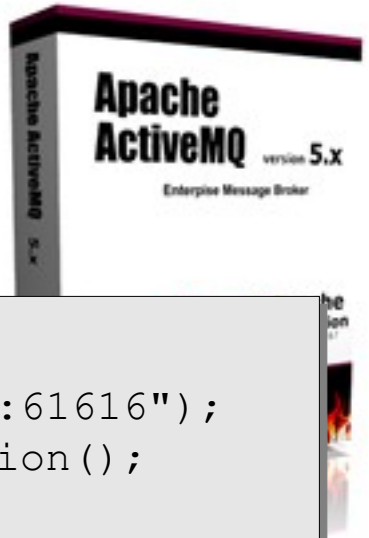
> Point-to-Point



> Publisher-Subscriber



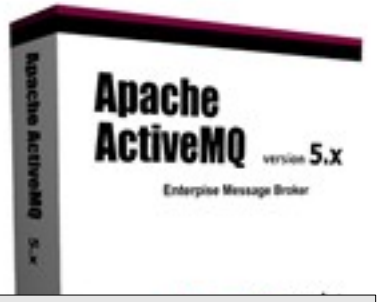
Apache ActiveMQ



- Java Publisher Example

```
1:   ConnectionFactory factory = new
        ActiveMQConnectionFactory("tcp://localhost:61616");
     Connection connection = factory.createConnection();

2:   Session jmsSession = connection.createSession(false,
                                   Session.AUTO_ACKNOWLEDGE);
     Topic topic = jmsSession.createTopic("DemoTopic");

3:   MessageProducer producer =
                              jmsSession.createProducer(topic);

4:   producer.send(
        jmsSession.createTextMessage("Hello")
       );
```

Apache ActiveMQ



> C# Subscriber Example

```
1:    using Apache.NMS.ActiveMQ;
      using Apache.NMS;
      namespace NMSSubscriber {
         class Subscriber {
           static void Main(string[] args) {
2:            ConnectionFactory factory =
                 new ConnectionFactory("tcp://localhost:61616");
              IConnection connection =
                                        factory.CreateConnection();

3:            ISession jmsSession =
 connection.CreateSession(AcknowledgementMode.AutoAcknowledge);
              ITopic topic = jmsSession.GetTopic("DemoTopic");

4:            IMessageConsumer consumer =
                                    jmsSession.CreateConsumer(topic);
```

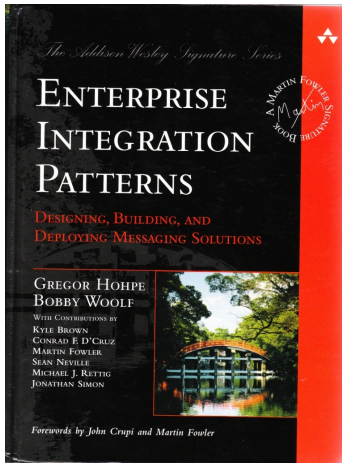
Apache ActiveMQ

> C# Subscriber Example

```
5:              consumer.Listener += new MessageListener(
                                          new Client().OnMessage);
                connection.Start();
            }

6:          public void OnMessage(IMessage message) {
              try {
                 if (message is ItextMessage) {
                   ITextMessage msg = message as ITextMessage;
                   Console.WriteLine("Nachricht : " + msg.Text);
                 }
              }
              catch (NMSException e) {
                 Console.WriteLine(e);
              }
            }
```

Apache Camel

- Powerful open source integration framework
- Current version 2.4
- Sub project of ActiveMQ
- Based on the well-known Enterprise Integration Patterns (EIP)
  (http://www.eaipatterns.com)



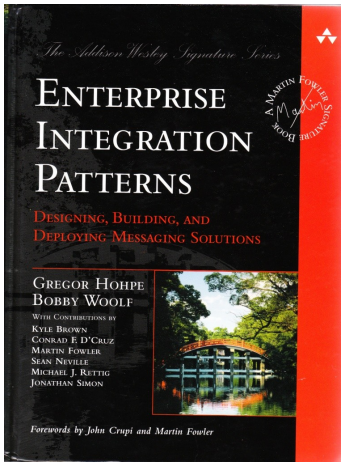
- Implement routing and mediation rules via
  - Java based Domain Specific Language
  - Spring based XML configuration
  - Scala DSL

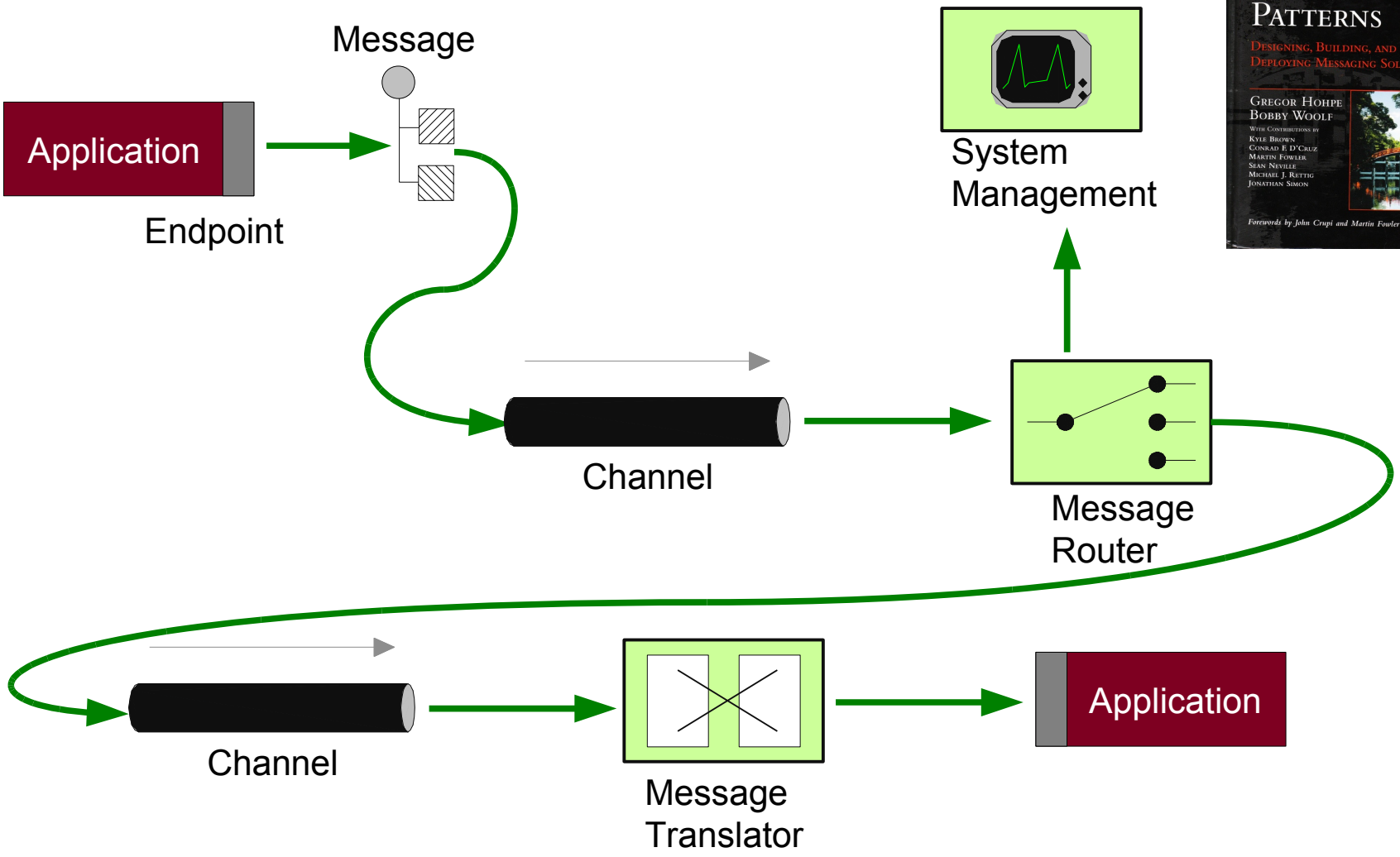
Apache Camel

- API is transport protocol transparent

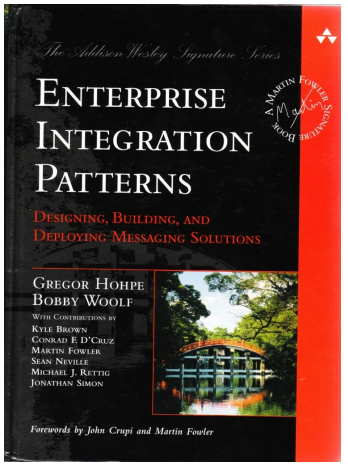
- Apache Camel supports:
  - Apache ActiveMQ (JMS Provider)
  - Apache CXF (JAX-WS implementation)
  - Apache MINA (networking framework)
  - Apache ServiceMix (ESB and JBI)

Apache Camel - Enterprise Integration Patterns





Apache Camel - EIP



8 Patterns

System Management

Message

10 Patterns

Application

Endpoint

12 Patterns

14 Patterns

Channel

Message Router

10 Patterns

Channel

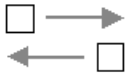
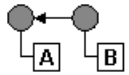
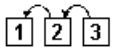
Message Translator

7 Patterns

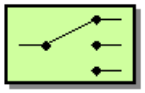
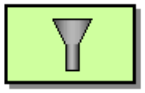
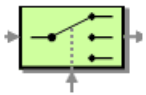
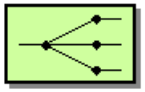
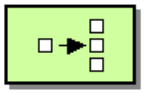
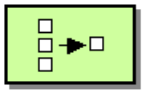
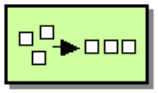
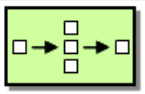
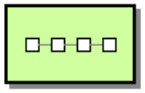
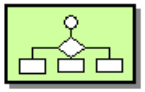
Application

Apache Camel - EIP

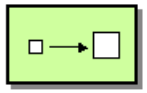
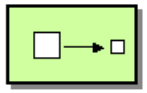
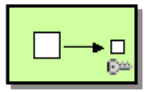
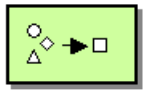
Message Construction

	Introduction to Message Construction
	Command Message
	Document Message
	Event Message
	Request-Reply
	Return Address
	Correlation Identifier
	Message Sequence
	Message Expiration
	Format Indicator

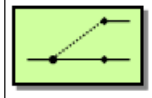
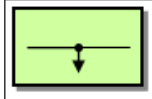
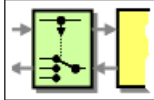
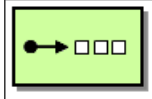
Message Routing

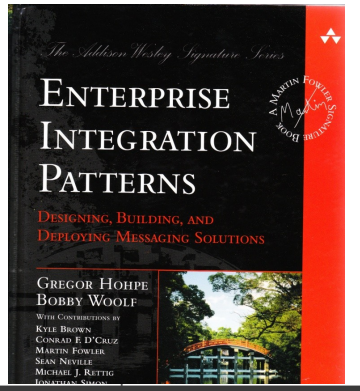
	Introduction to Message Routing
	Content-Based Router
	Message Filter
	Dynamic Router
	Recipient List
	Splitter
	Aggregator
	Resequencer
	Composed Message Processor
	Scatter-Gather
	Routing Slip
	Process Manager
	Message Broker

Message Transformation

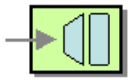
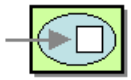
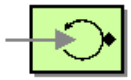
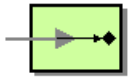
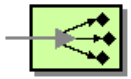
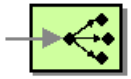
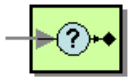
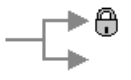
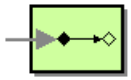
	Introduction to Message Transformation
	Envelope Wrapper
	Content Enricher
	Content Filter
	Claim Check
	Normalizer
	Canonical Data Model

System Management

	Introduction to System Management
	Control Bus
	Detour
	Wire Tap
	Message History
	Message Store
	Smart Proxy
	Test Message



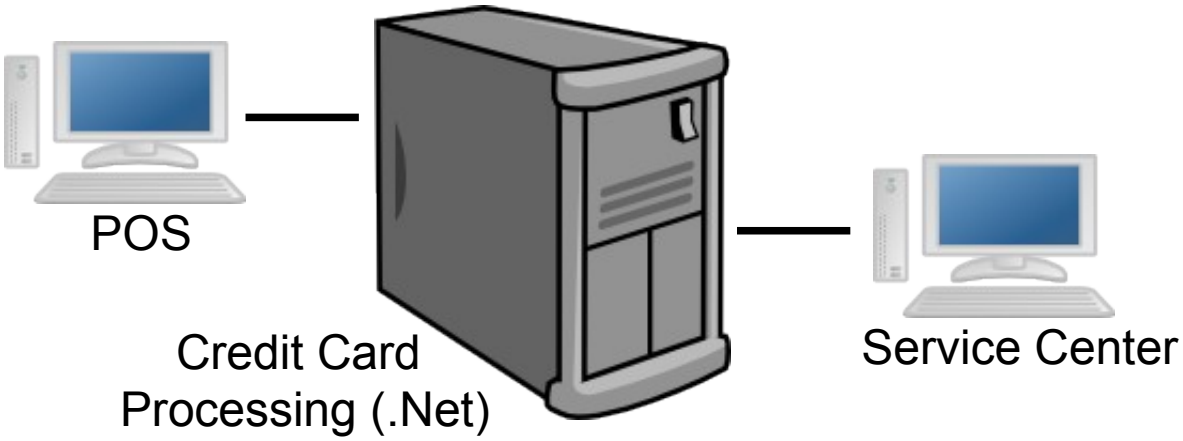
Messaging Endpoints

	Introduction to Messaging Endpoints
	Messaging Gateway
	Messaging Mapper
	Transactional Client
	Polling Consumer
	Event-Driven Consumer
	Competing Consumers
	Message Dispatcher
	Selective Consumer
	Durable Subscriber
	Idempotent Receiver
	Service Activator

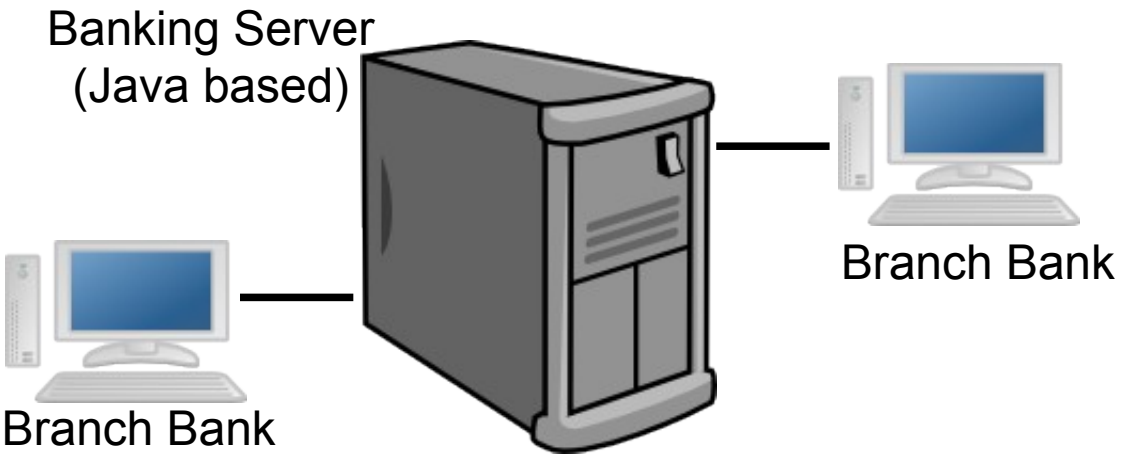
AGENDA

- Motivation
- Tools Infrastructure
- Example
  - Simple Messaging
  - Routing & Enrichment
  - Multicasting
  - Scalability and Reliability
  - Availability
- Conclusion

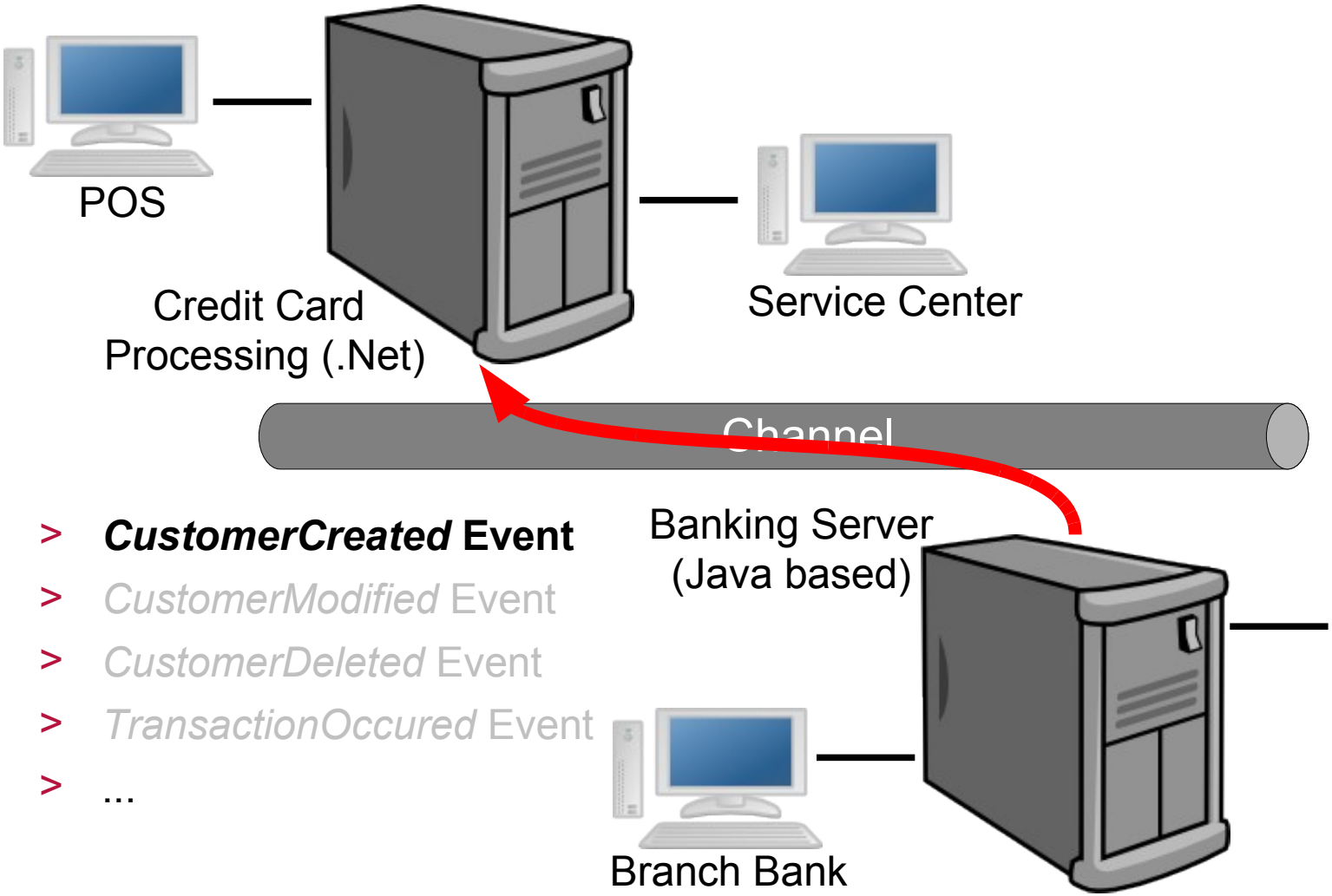
Example - Transfer Details



- *CustomerCreated* Event
- *CustomerModified* Event
- *CustomerDeleted* Event
- *TransactionOccured* Event
- ...

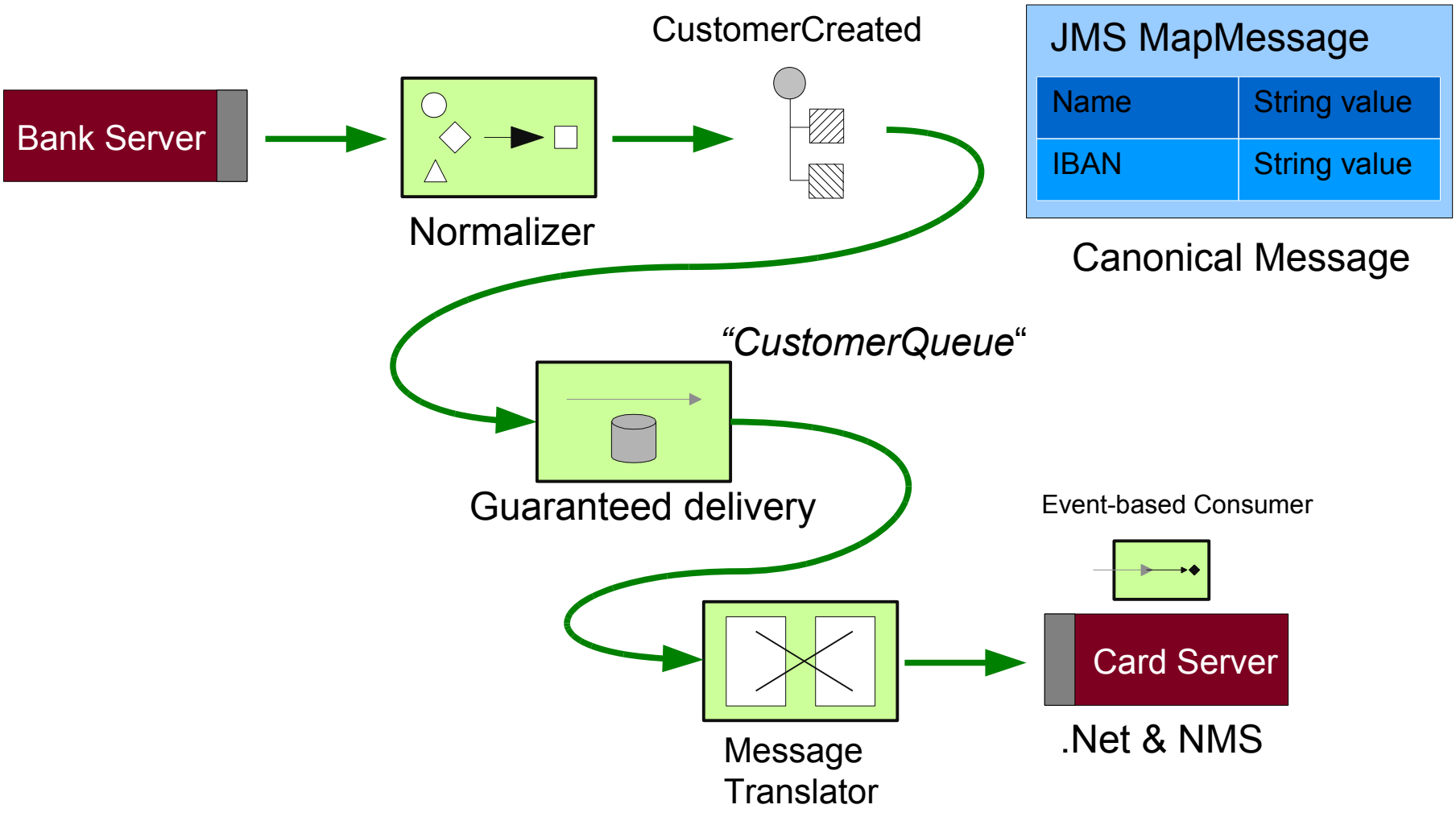


Example - Transfer Details



- ***CustomerCreated* Event**
- *CustomerModified* Event
- *CustomerDeleted* Event
- *TransactionOccured* Event
- ...

CustomerCreated Event



Implementation - Card Server

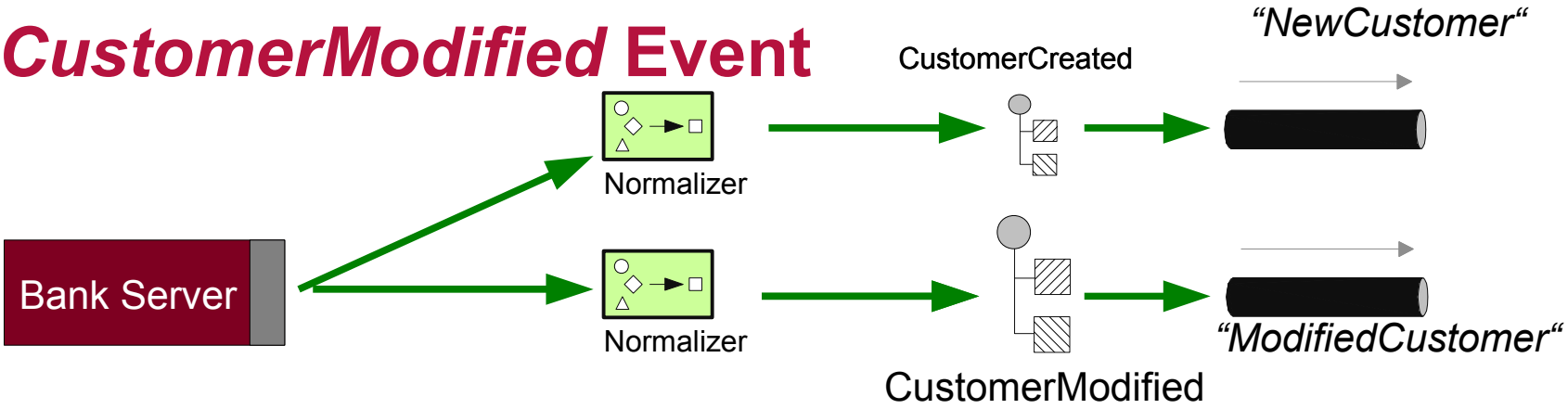
> Handling *CustomerCreated* Event

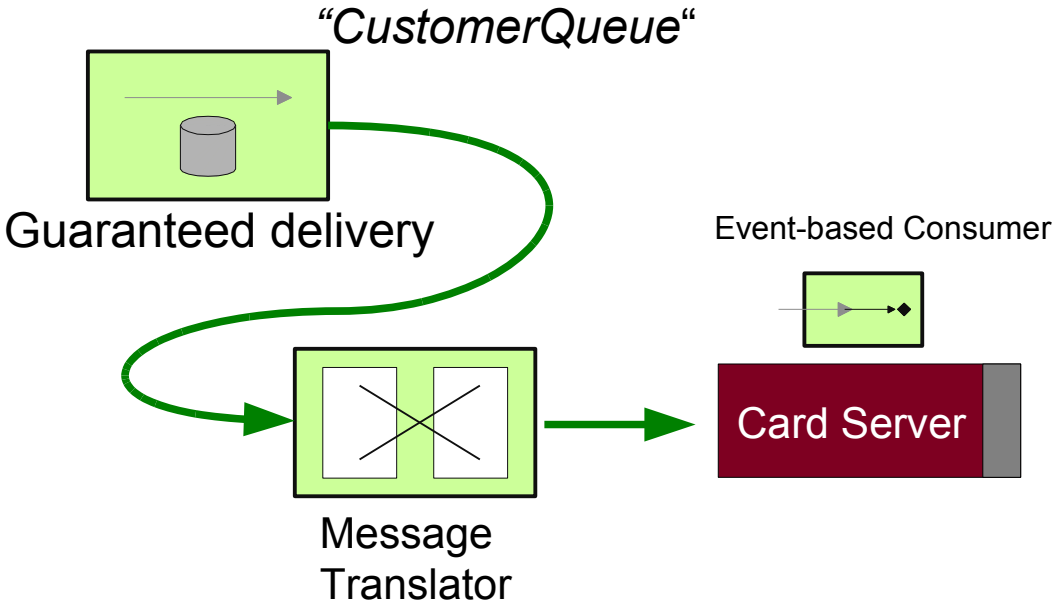
```
1:  public void OnMessage(IMessage message) {
        try {
            if (message is IMapMessage) {
2:              IMapMessage msg = (ImapMessage) message;
                IPrimitiveMap map = message.Body;

                Customer customer = new Customer();
                customer.IBAN = map.GetString(CUSTOMER_IBAN);
                customer.Name = map.GetString(CUSTOMER_NAME);

3:              customerDao.SaveNewCustomer(customer);
                Console.WriteLine("Saving customer '{0}'
                                   succeeded", customer);
            }
        }
    }
```

AGENDA

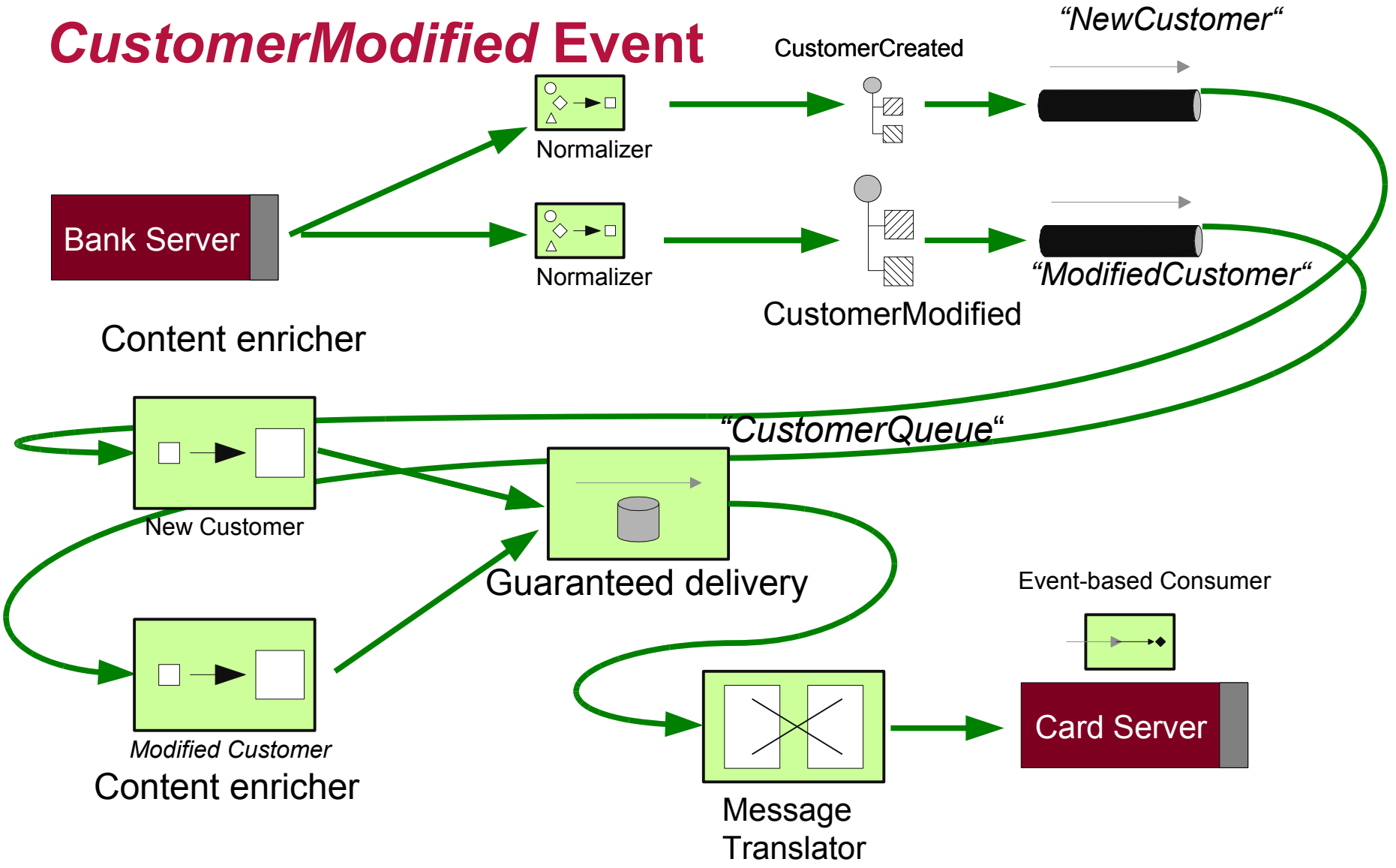
- Motivation
- Tools Infrastructure
- Example
  - Simple Messaging
  - Routing & Enrichment
  - Multicasting
  - Scalability and Reliability
  - Availability
- Conclusion

CustomerModified Event

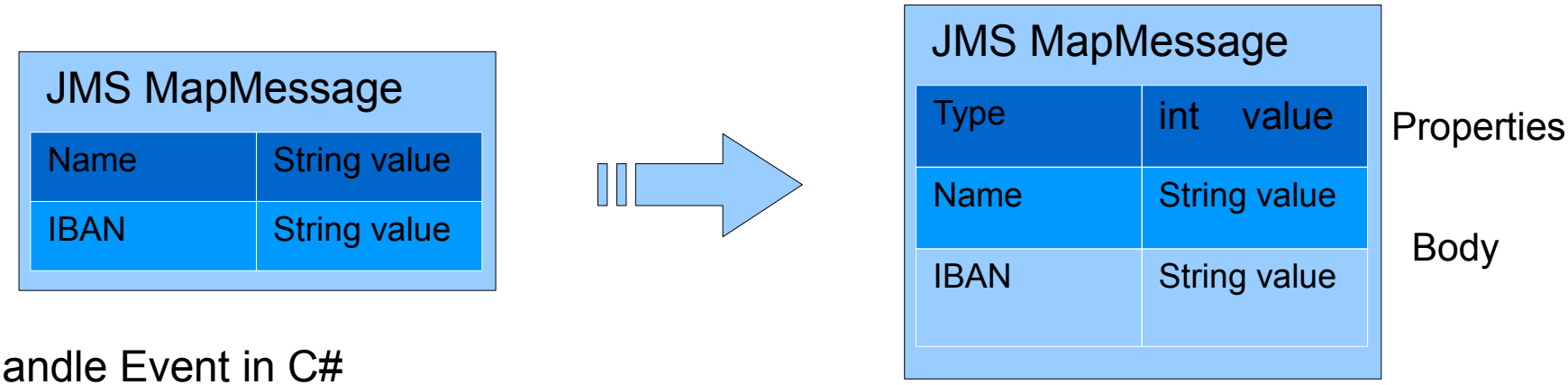




CustomerModified Event



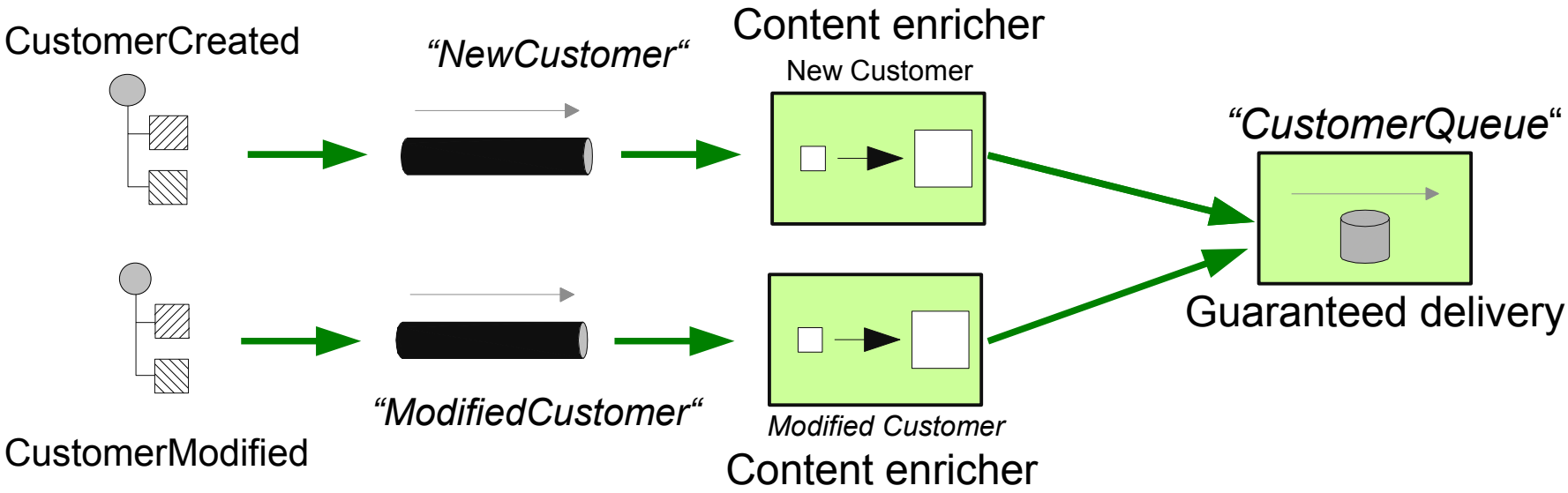
Implementation - Normalized Message



> Handle Event in C#

```
1:  public void OnMessage(IMessage message) {
        try {
            if (message is ImapMessage) {
2:            IMapMessage msg = (ImapMessage)message;

3:            if (msg.Properties.GetInt(REQUEST_TYPE) ==
                  REQUEST_NEW_CUSTOMER){
                IPrimitiveMap map = message.Body;
                // save customer
              }
```

Implementation - Routing & Enrichment



- Apache Camel provides the means to accomplish that
  - transparently to the application
  - embedded into Apache ActiveMQ

Implementation – Routing & Enrichment

- Activate Apache Camel inside Apache ActiveMQ
  - activemq.xml (in conf Directory)

```
<!--
   Uncomment to enable Camel
   Take a look at camel.xml for more details
  -->
 <import resource="camel.xml"/>
```

Implementation – Routing & Enrichment

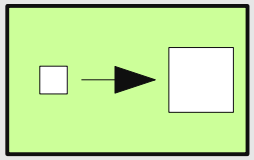
- Adding a Camel route to camel.xml (in the conf directory)

```
<beans
   xmlns="http://www.springframework.org/schema/beans"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://camel.apache.org/schema/spring
        http://camel.apache.org/schema/spring/camel-spring.xsd
        http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

   <camelContext id="camel"
                 xmlns="http://camel.apache.org/schema/spring">
      <route>
         <from uri="activemq:NewCustomer"/>
         <to uri="activemq:CustomerQueue"/>
      </route>
   </camelContext>
<beans>
```

Implementation – Routing & Enrichment

- Adding a Camel Enrichment to camel.xml (in the conf directory)

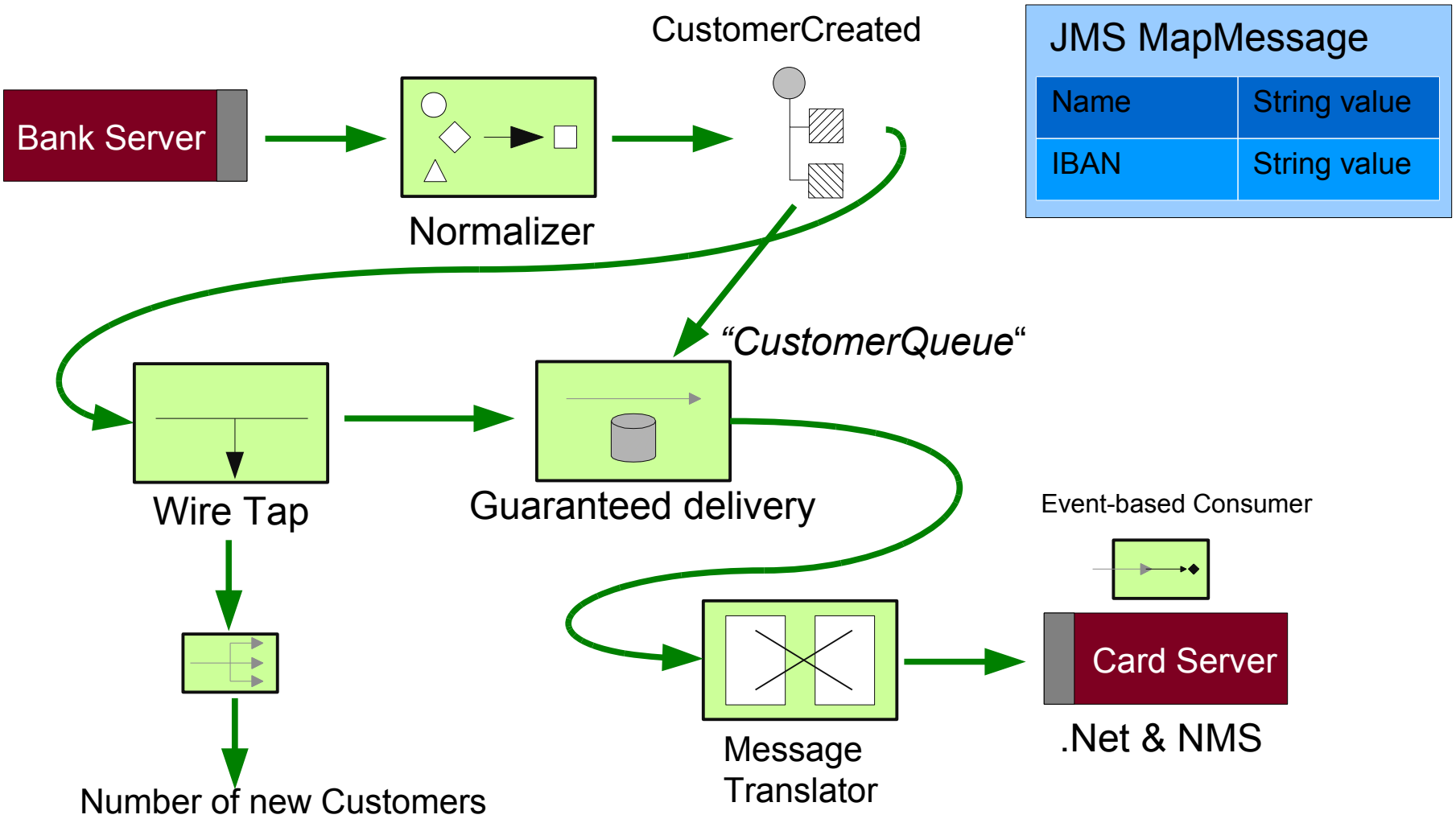
```
<beans
   <camelContext id="camel"
                 xmlns="http://camel.apache.org/schema/spring">
      <route>
         <from uri="activemq:NewCustomer"/>

         <setHeader headerName="Typ">
            <constant>NewCustomer</constant>
         </setHeader>

         <to uri="activemq:CustomerQueue"/>
      </route>
   </camelContext>
<beans>
```



AGENDA

- Motivation
- Tools Infrastructure
- Example
  - Simple Messaging
  - Routing & Enrichment
  - Multicasting
  - Scalability and Reliability
  - Availability
- Conclusion

Wire Tap CustomerCreated Event



Wire Tap CustomerCreated Event

- Wire Tap to JMS Topic with camel.xml

```
<beans
   <camelContext id="camel"
                 xmlns="http://camel.apache.org/schema/spring">
      <route>
         <from uri="activemq:CARD_NEW_CUSTOMER"/>
         <multicast>
            <pipeline>
               <setHeader headerName="Type">
                  <constant>NewCustomer</constant>
               </setHeader>
               <to uri="activemq:CARD_GLOBAL_QUEUE"/>
            </pipeline>

            <to uri="activemq:topic:CARD_BAM"/>
         </multicast>
      </route>
```

Wire Tap CustomerCreated Event

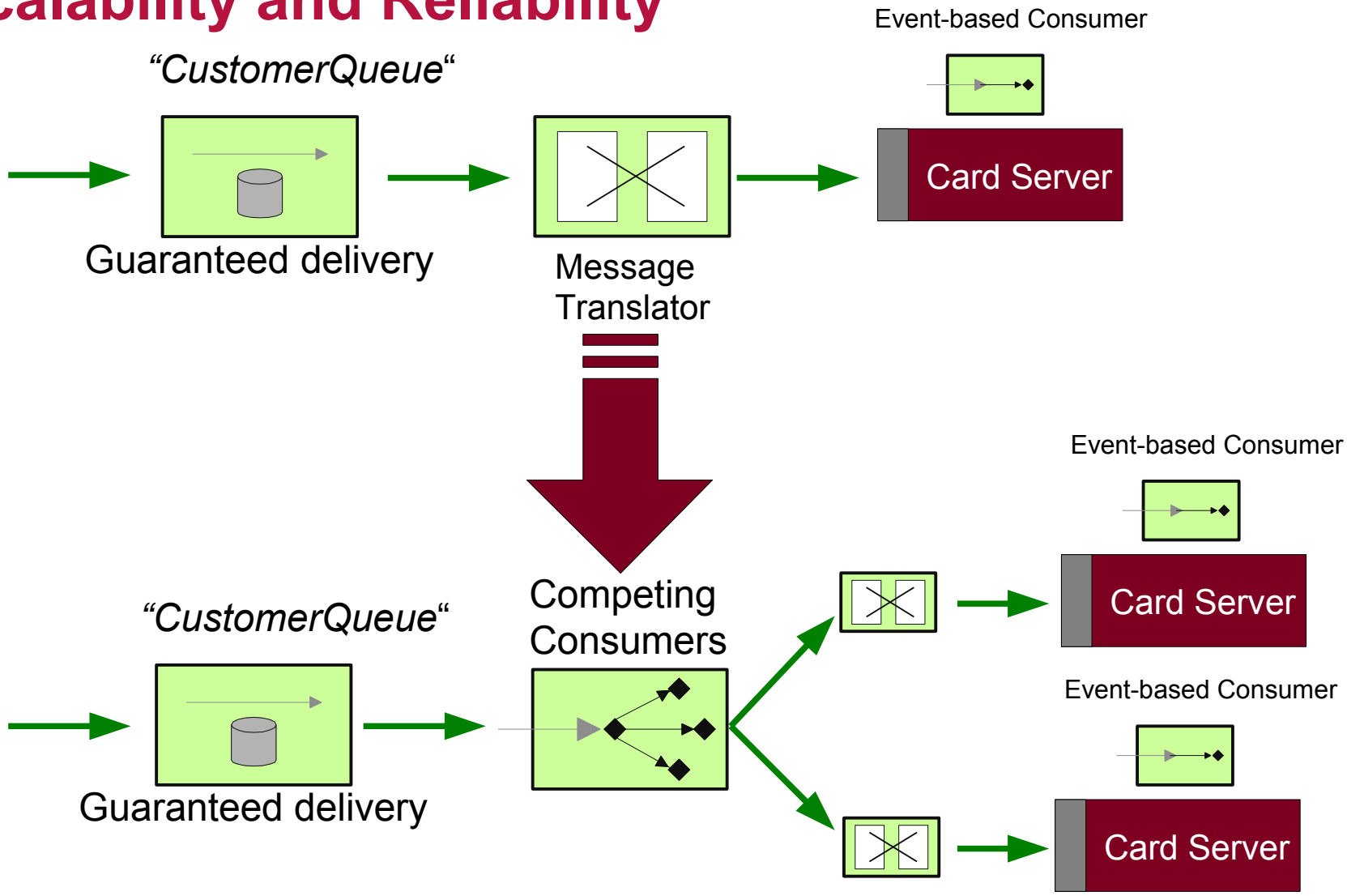
- Wire Tap to JMS Topic with camel.xml

```
<beans
   <camelContext id="camel"
                 xmlns="http://camel.apache.org/schema/spring">
   <route>
      <from uri="activemq:CARD_NEW_CUSTOMER"/>

      <setHeader headerName="Typ">
          <constant>NewCustomer</constant>
      </setHeader>

      <to uri="activemq:CARD_GLOBAL_QUEUE"/>

      <wireTap uri="activemq:CARD_BAM">
         <body>
            <constant>noch eine Kundenkarte</constant>
         </body>
      </wireTap>
   </route>
```

AGENDA

- Motivation
- Tools Infrastructure
- Example
  - Simple Messaging
  - Routing & Enrichment
  - Multicasting
  - Scalability and Reliability
  - Availability
- Conclusion

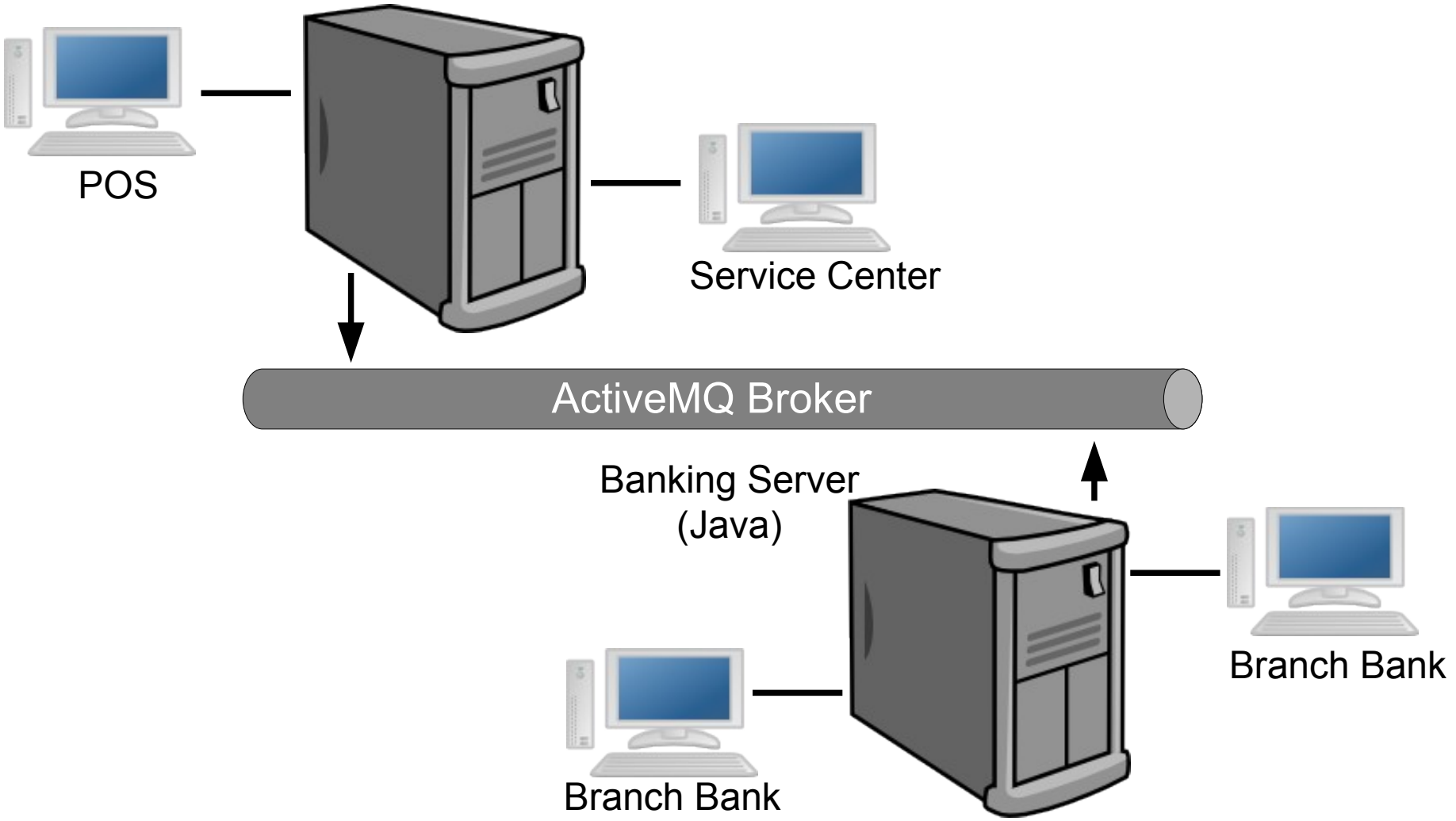
Scalability and Reliability



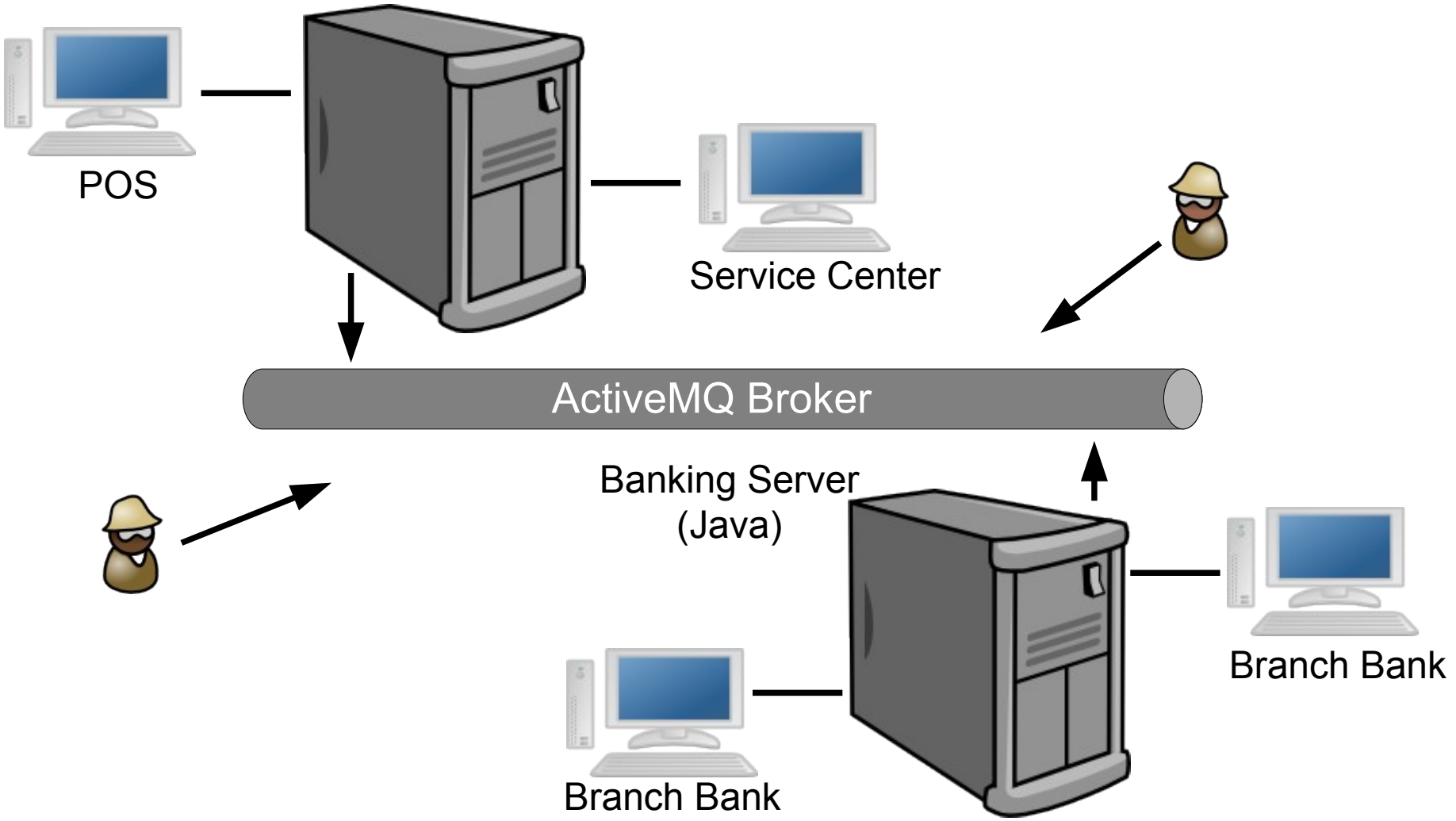
AGENDA

- Motivation
- Tools Infrastructure
- Example
  - Simple Messaging
  - Routing & Enrichment
  - Multicasting
  - Scalability and Reliability
  - Security
  - Availability
- Conclusion

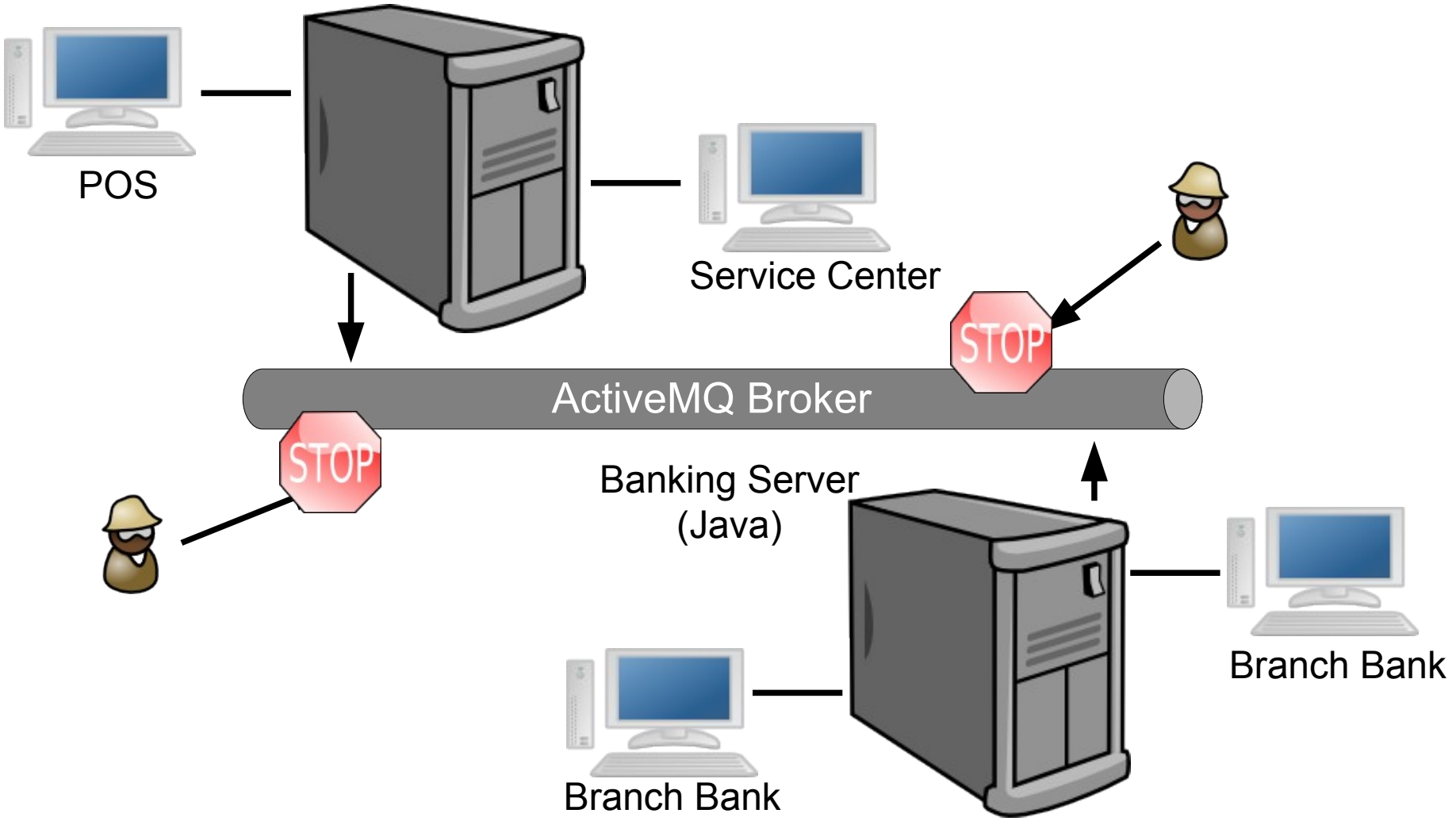
Security


POS
Service Center
ActiveMQ Broker
Banking Server
(Java)
Branch Bank
Branch Bank

Security


POS
Service Center
ActiveMQ Broker
Banking Server
(Java)
Branch Bank
Branch Bank

Security


POS
Service Center
STOP
ActiveMQ Broker
STOP
Banking Server
(Java)
Branch Bank
Branch Bank

Security - Using Secure Socket Layer

- Securing the Broker
  - activemq.xml (in the conf directory)

```
<beans ...>
   <broker xmlns="http://activemq.apache.org/schema/core"
           brokerName="localhost"
           dataDirectory="${activemq.base}/data"
           destroyApplicationContextOnStop="true">
   <sslContext>
      <sslContext keyStore="file:path/to/broker.ks"
                  keyStorePassword="ks_pwd"
                  trustStore="file:path/to/broker.ts"
                  trustStorePassword="ts_pwd"/>
   </sslContext>

   <transportConnectors>
      <transportConnector name="ssl"
                          uri="ssl://localhost:61617"/>
   </transportConnectors>
```

Security - Using Secure Socket Layer

- Securing the Java Server
  - ActiveMQ Broker Certificate (public key) must be stored in Clients Truststore
  - Start Java Server with following system properties

    javax.net.ssl.keyStore=path/to/client.ks

    javax.net.ssl.keyStorePassword=client_pwd

    javax.net.ssl.trustStore=path/to/client.ts

- Replace Broker URL in Java Server

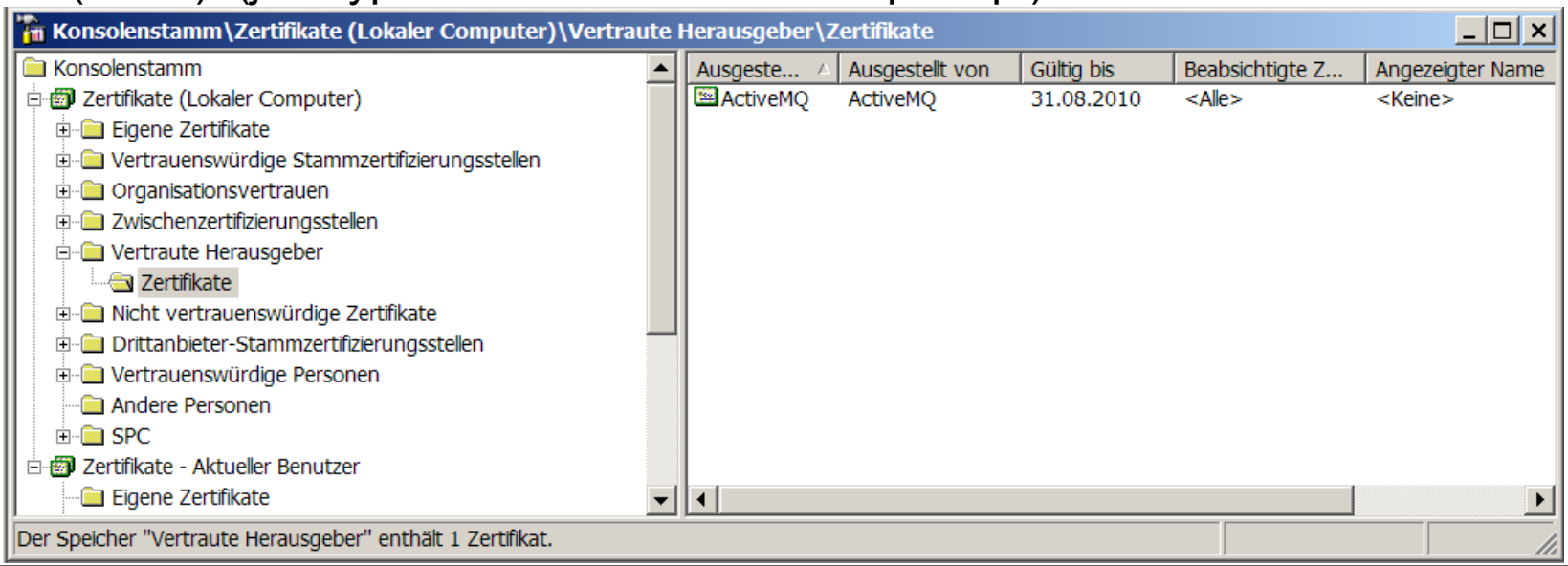
```
String url = "ssl://theBroker:61617";
```

- SSL Debugging

```
-Djavax.net.debug=ssl
```

Security - Using Secure Socket Layer

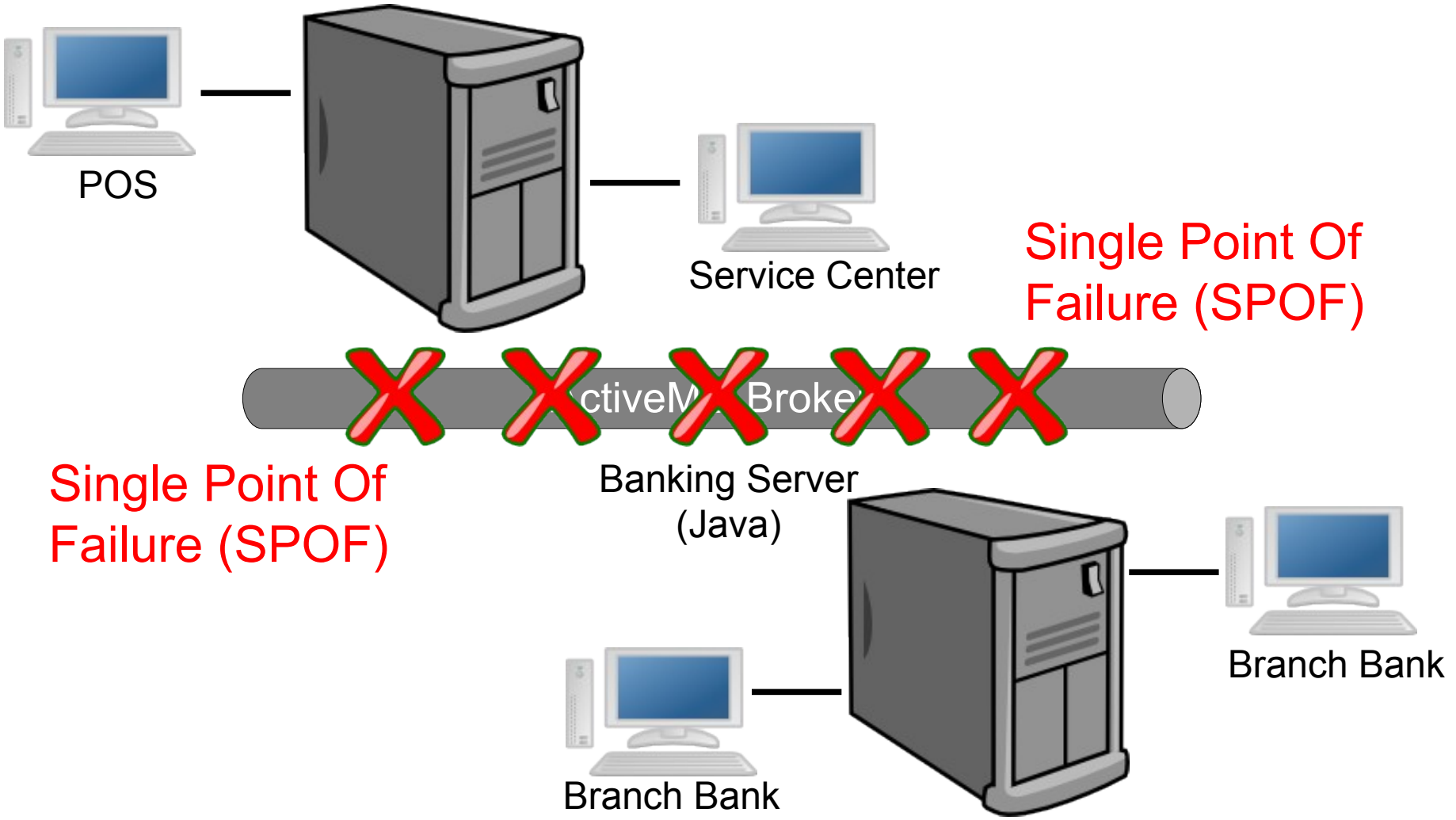
- Securing the .Net Server
  - Requires NMS > 1.3
  - Requires .Net 3.5 (at least 2.0 does not work)
  - Import ActiveMQ Certificate (public key) into the servers truststore by using the Certificate Snap-in in the Microsoft Management Console (MMC) (just type *mmc* at the command prompt)



AGENDA

- Motivation
- Tools Infrastructure
- Example
  - Simple Messaging
  - Routing & Enrichment
  - Multicasting
  - Scalability and Reliability
  - Security
  - Availability
- Conclusion

Availability



Availability

- Reconnecting consumers and producers
  - Failover protocol
  - Simple modify your broker URL

```
String url = "failover:(tcp://localhost:61616)";
```

  - Works for Java and .Net

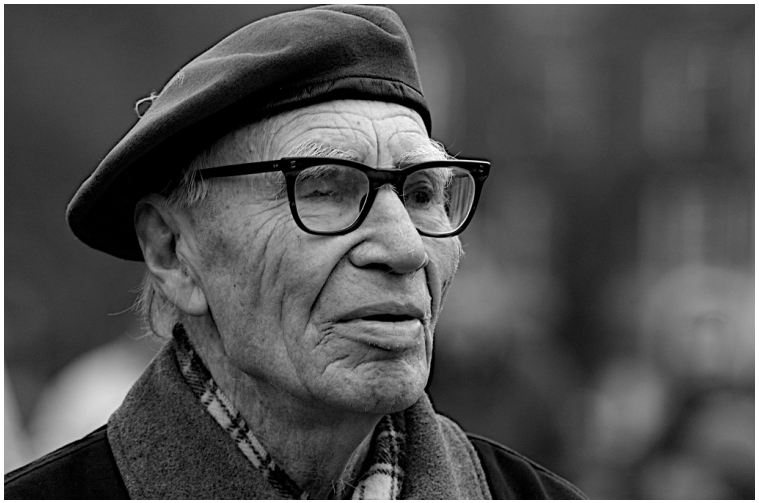
- Reconnection to Master-Slave Broker Configuration

```
String url = "failover:(tcp://localhost:61616,
                        tcp://localhost:61618)?randomize=false";
```

AGENDA

- Motivation
- Tools Infrastructure
- Example
  - Simple Messaging
  - Routing & Enrichment
  - Multicasting
  - Scalability and Reliability
  - Security
  - Availability
- Conclusion

Conclusion



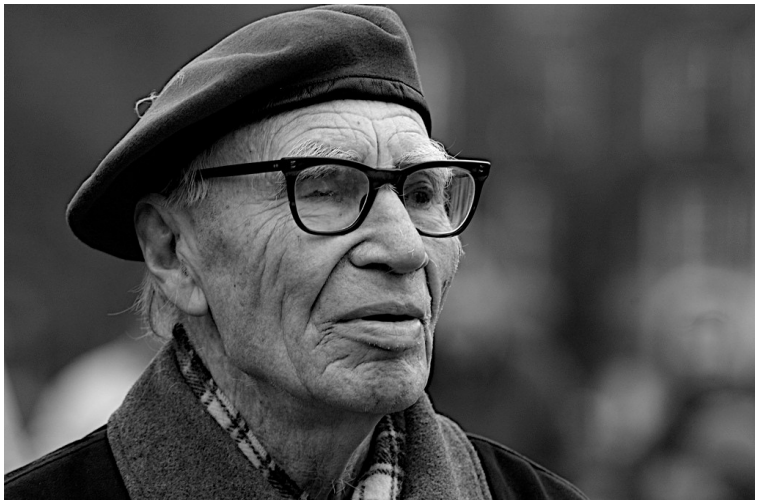
© by Steve Punter at flickr

- By using
  - Apache ActiveMQ,
  - Apache AciveMQ NMS
  - Apache Camel

  we are able to integrate .Net with Java

- We gained
  - Loose-coupling
  - Guaranteed Delivery
  - Extensibility
  - Scalability and Reliability
  - Security
  - *Availability*

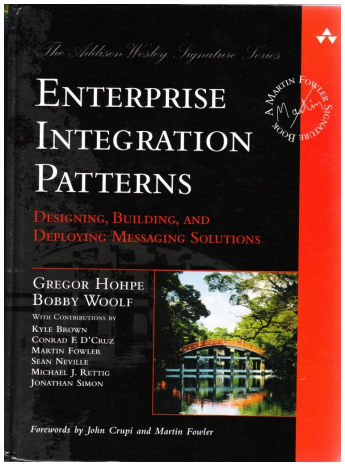
Conclusion



© by Steve Punter at flickr

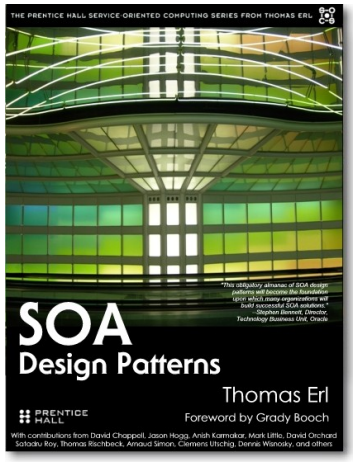
- We only saw a very small part of Apache ActiveMQ and Camel
- Infrastructure is kind of lightweight
  - Intermediate broker
- Obstracles
  - Documentation is partly available
  - Programming Model is more complex
  - Debugging is more demanding
  - No “real” .Net Transactions (System.Transaction namespace)

Literature

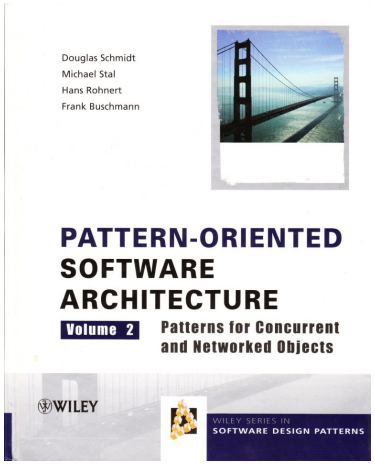

The Addison Wesley Signature Series
ENTERPRISE INTEGRATION PATTERNS
DESIGNING, BUILDING, AND DEPLOYING MESSAGING SOLUTIONS
GREGOR HOHPE
BOBBY WOOLF
WITH CONTRIBUTIONS BY
KYLE BROWN
CONRAD F D'CRUZ
MARTIN FOWLER
SEAN NEVILLE
MICHAEL J. RETTIG
JONATHAN SIMON
Forewords by John Crupi and Martin Fowler

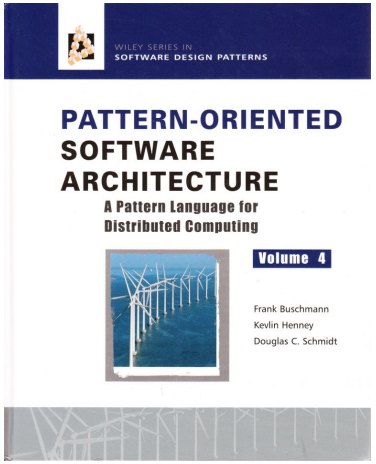

EVENT-DRIVEN ARCHITECTURE
How SOA Enables the Real-Time Enterprise
Hugh Taylor | Angela Yochem | Les Phillips | Frank Martinez

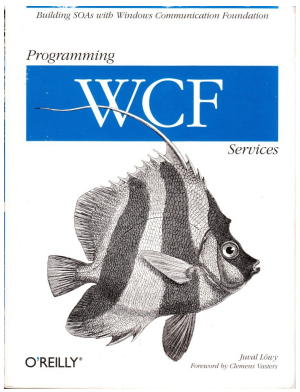

daniel LIEBHART
guido SCHMUTZ
marcel LATTMANN
markus HEINISCH
michael KÖNINGS
mischa KÖLLIKER
perry PAKULL
peter WELKENBACH
INTEGRATION ARCHITECTURE BLUEPRINT
LEITFADEN ZUR KONSTRUKTION VON INTEGRATIONSLÖSUNGEN
trivadis
HANSER


THE PRENTICE HALL SERVICE-ORIENTED COMPUTING SERIES FROM THOMAS ERL
SOA Design Patterns
Thomas Erl
Foreword by Grady Booch
PRENTICE HALL


Nicolai Josuttis
SOA in der Praxis
System-Design für verteilte Geschäftsprozesse
dpunkt.verlag


Douglas Schmidt
Michael Stal
Hans Rohnert
Frank Buschmann
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
Volume 2
Patterns for Concurrent and Networked Objects
WILEY
WILEY SERIES IN SOFTWARE DESIGN PATTERNS


WILEY SERIES IN SOFTWARE DESIGN PATTERNS
PATTERN-ORIENTED SOFTWARE ARCHITECTURE
A Pattern Language for Distributed Computing
Volume 4
Frank Buschmann
Kevlin Henney
Douglas C. Schmidt


Building SOAs with Windows Communication Foundation
Programming
WCF
Services
O'REILLY

Vielen Dank!

Thomas Haug

MATHEMA Software GmbH