Enterprise Messaging Workshop

Jeremy Deane
http://jeremydeane.net

Agenda

- Environment Setup
- Asynchronous Hello World!
- Messaging Foundations
- **Exercises**
- Messaging Networks
- Advanced Messaging
- Extensible Messaging
- **Exercises**

Environment Setup

- JDK 1.7+ Pre-Requisite ($JAVA_HOME in path)

- Unzip MessagingWorkshop.zip to $HOME

```
https://app.box.com/s/lx414bpwht0lutjo3jqs
```

- Unzip the following:
  - maven.zip (optional unpack workshop repo artifacts)
  - activemq.zip (includes additional Camel JARs and Configuration)
  - jetty.zip (includes pre-deployed magic-supplies.war)
  - magic-supplies.zip or https://github.com/jtdeane/magic-supplies
  - message-client.zip or https://github.com/jtdeane/message-client
  - camel-magic-router.zip or https://github.com/jtdeane/camel-magic-router
  - camel-standalone.zip or https://github.com/jtdeane/camel-standalone-router

- Add $JETTY_HOME & $MAVEN_HOME to path

- Windows Install ActiveMQ as Service:

./$ACTIVEMQ_HOME/win65/InstallService.bat

Asynchronous Hello World!

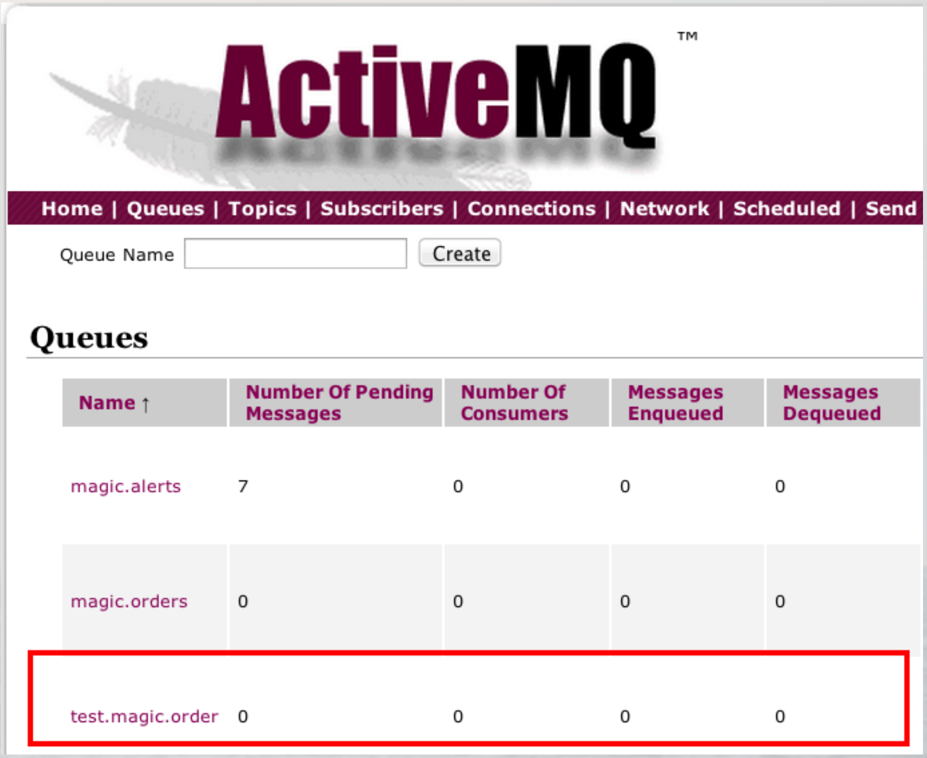
1. Start ActiveMQ

```
cd $ACTIVEMQ_HOME/bin
./activemq start OR ./activemq.bat
```

2. Build Magic Supplies Project

```
cd magic-supplies
mvn clean install
```

3. Import project into Java IDE


ActiveMQ™
Home | Queues | Topics | Subscribers | Connections | Network | Scheduled | Send
Queue Name
Create

Queues

Name ↑	Number Of Pending Messages	Number Of Consumers	Messages Enqueued	Messages Dequeued
magic.alerts	7	0	0	0
magic.orders	0	0	0	0
test.magic.order	0	0	0	0

4. Execute Producer

```
magic-supplies/src/test/java/cogito/online/messaging/JMSProducerFunctionalTest.java
```

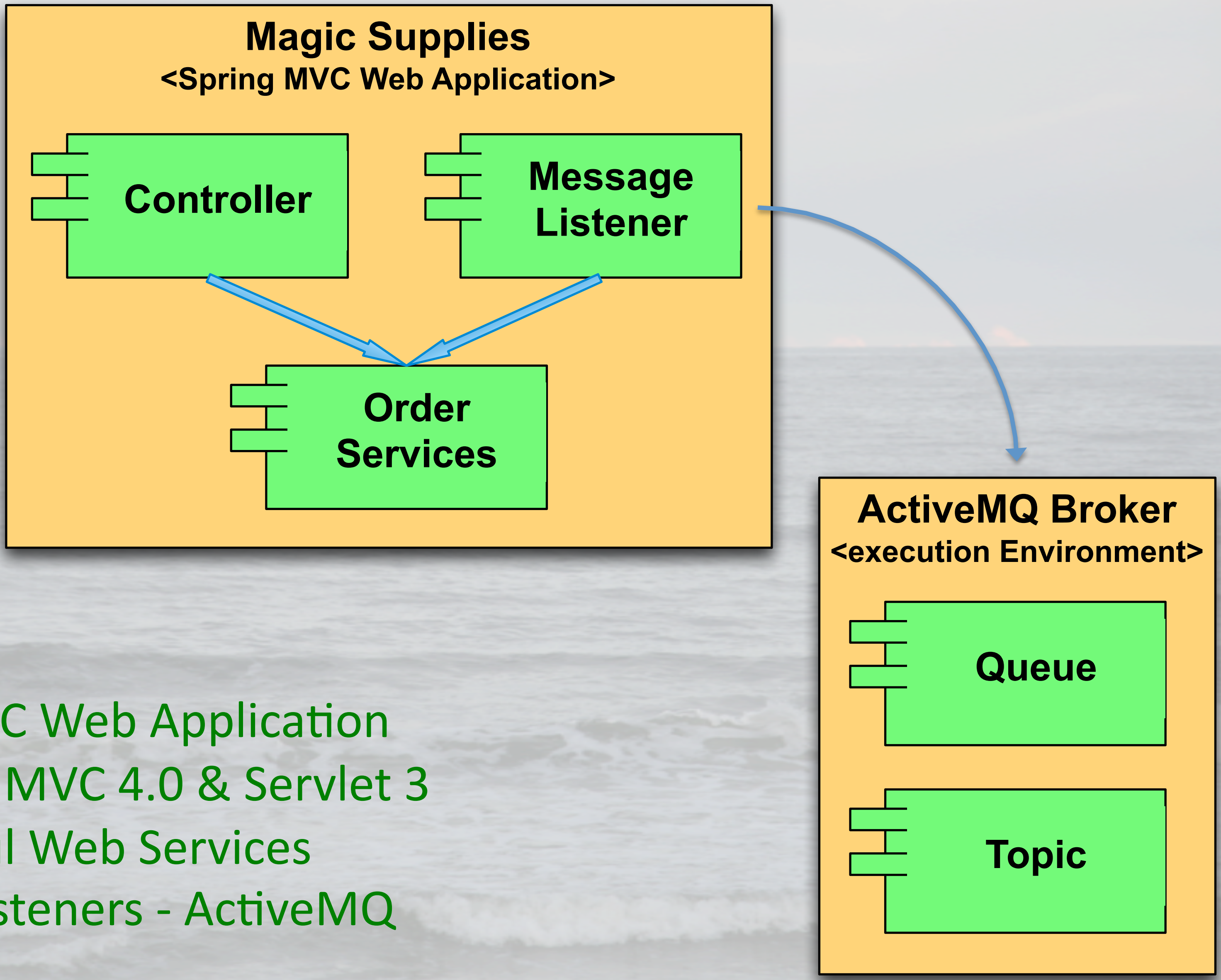
5. View ActiveMQ Console

```
Open http://localhost:8161/admin/ {admin/admin}
Open http://localhost:8161/admin/queues.jsp
```

6. Execute Consumer

```
magic-supplies/src/test/java/cogito/online/messaging/JMSConsumerFunctionalTest.java
```

Magic Supplies Web Application



- Spring MVC Web Application
  - Spring MVC 4.0 & Servlet 3
  - RESTful Web Services
  - JMS Listeners - ActiveMQ

Magic Supplies Web Services

1. Start ActiveMQ *

```
cd $ACTIVEMQ_HOME/bin
./activemq start OR ./activemq.bat
```

2. Start Jetty Web Server

```
cd $JETTY_HOME/bin
./jetty.sh start OR java -DSTOP.PORT=8079 -jar start.jar
```

3. Tail Jetty Log

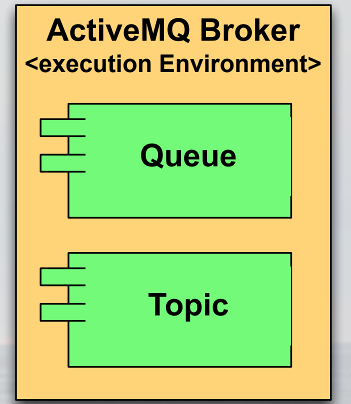
```
cd $JETTY_HOME/logs
tail –f {Current Log}
```

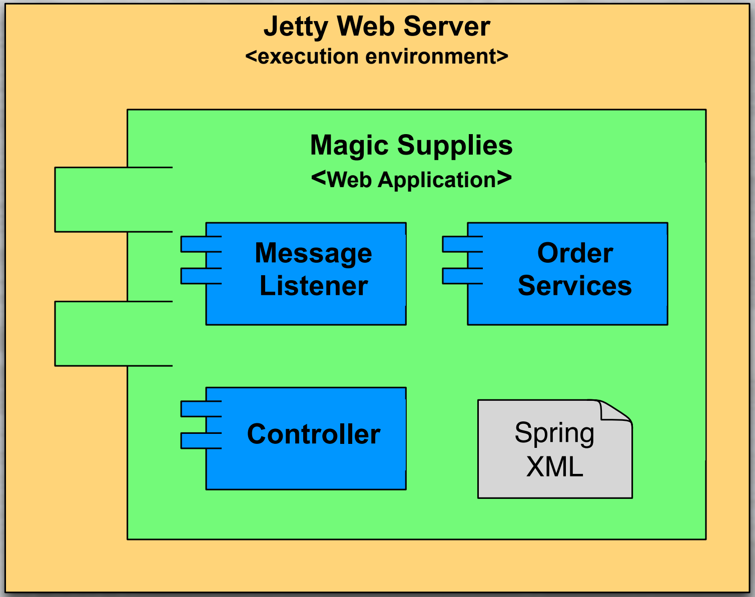
4. Open Browser

```
http://localhost:8080/magic-supplies/health
```

5. View Health Check Response

```
All Systems Go
```





* Web Application requires ActiveMQ started

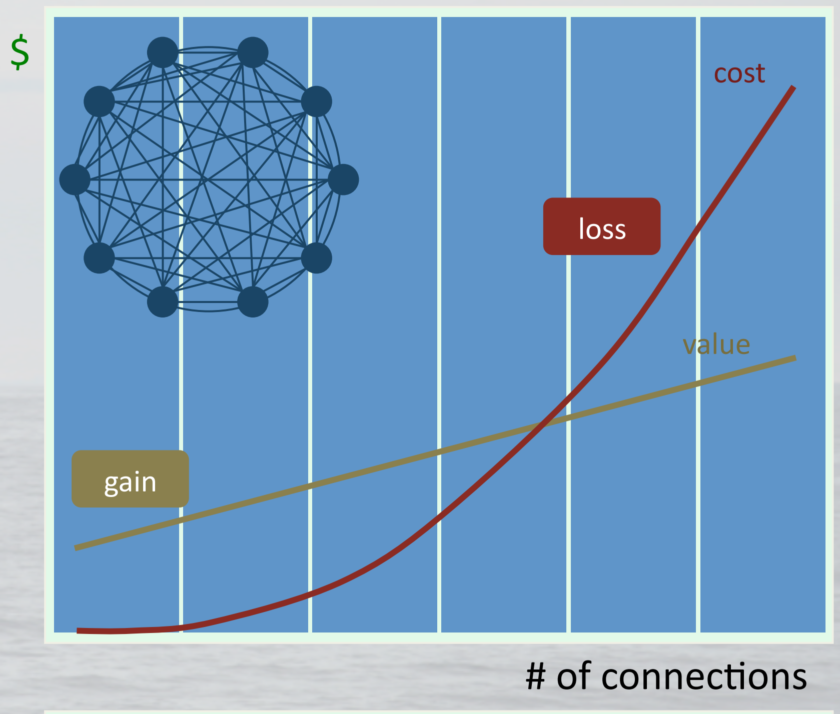
P2P Hidden Costs

Web services

A web service does **NOT** truly decouple the consumer and provider

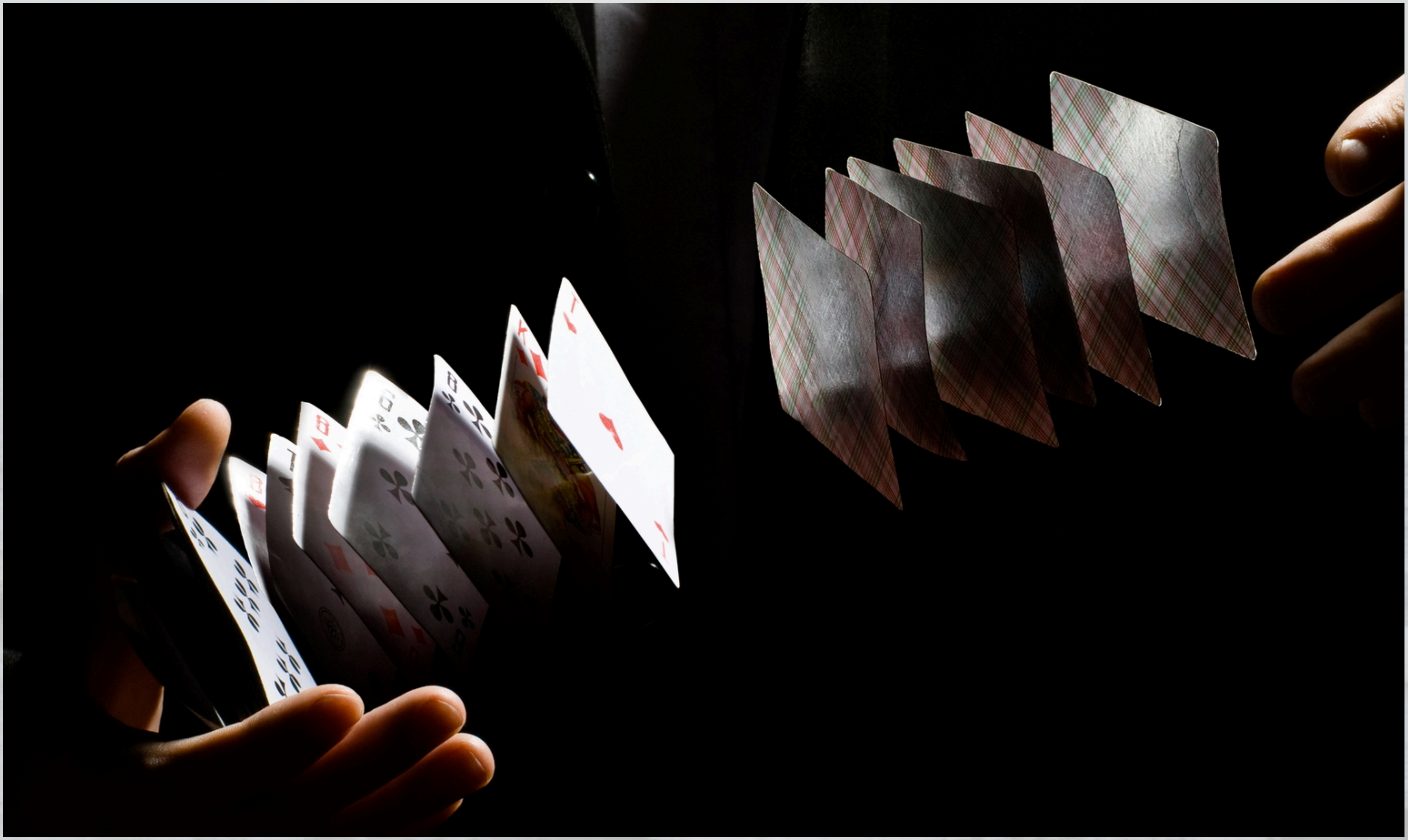
P2P Integrations

The cost of maintaining P2P integrations **increases exponentially** as the number of the connections increases



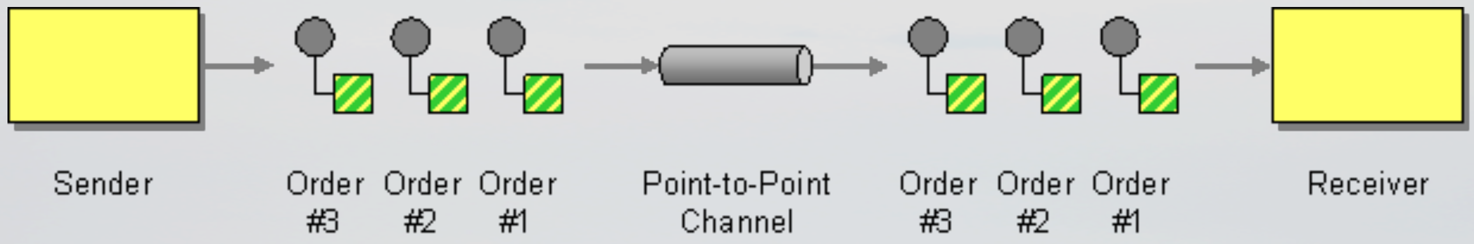
Bottom Line SOA
by Marc Rix

Messaging Foundations

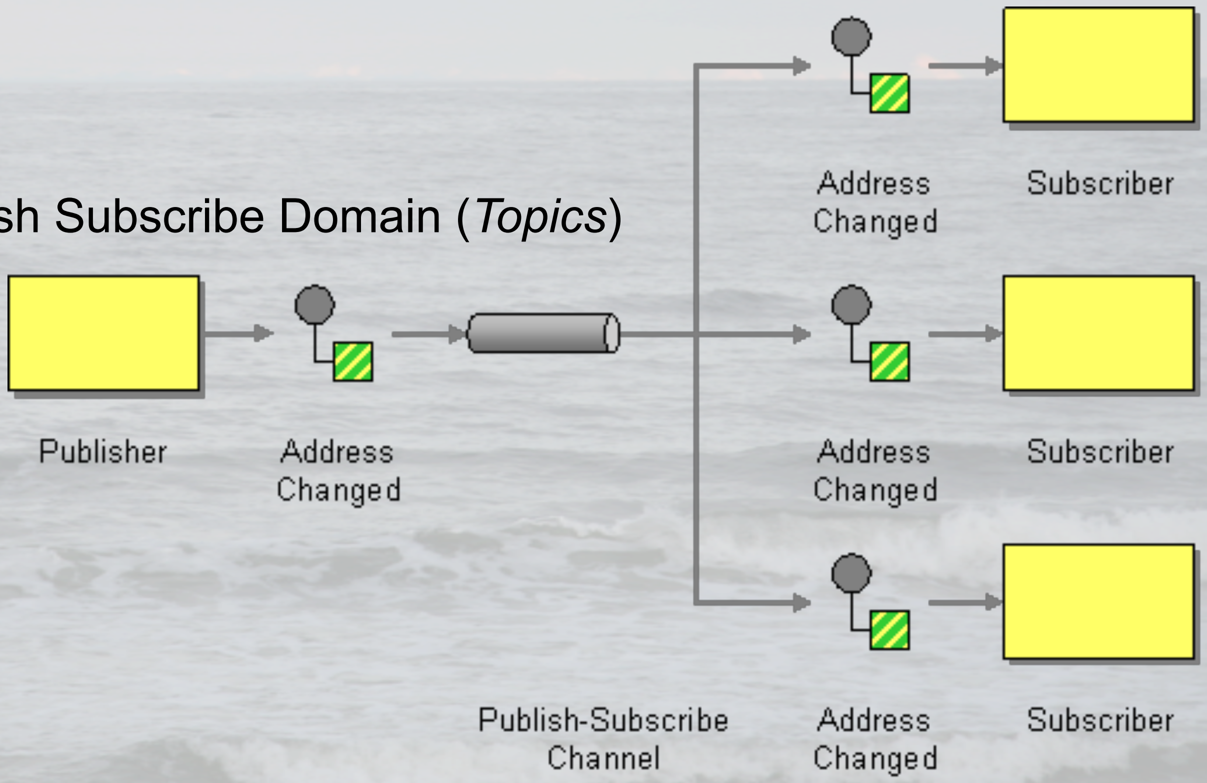


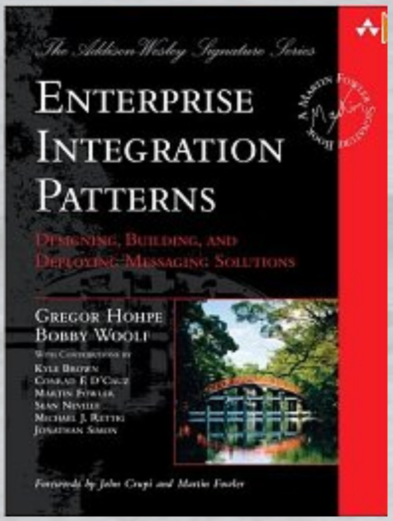
Enterprise Integration Patterns*

Point-to-Point Domain (*Queues*)



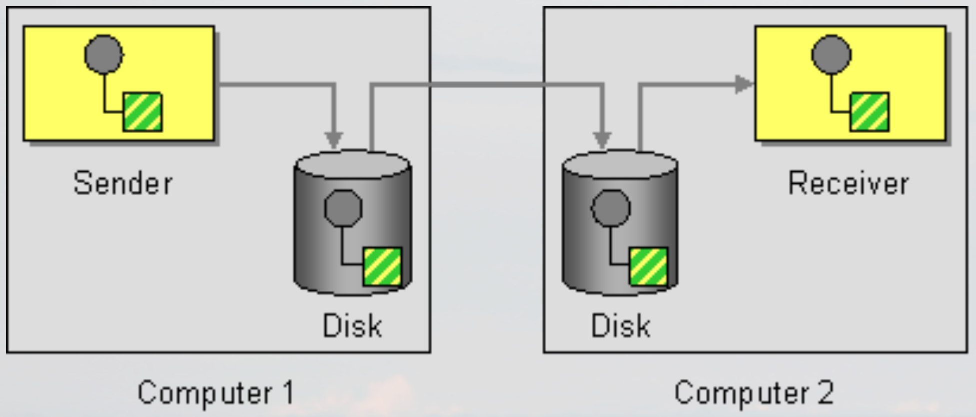
Publish Subscribe Domain (*Topics*)



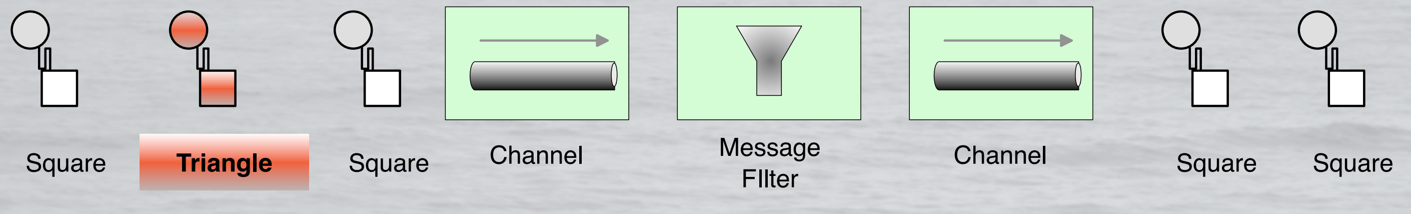

The Addison-Wesley Signature Series
Enterprise Integration Patterns
Designing, Building, and Deploying Messaging Solutions
Gregor Hohpe
Bobby Woolf

* A.K.A. Message Exchange Patterns (MEP)

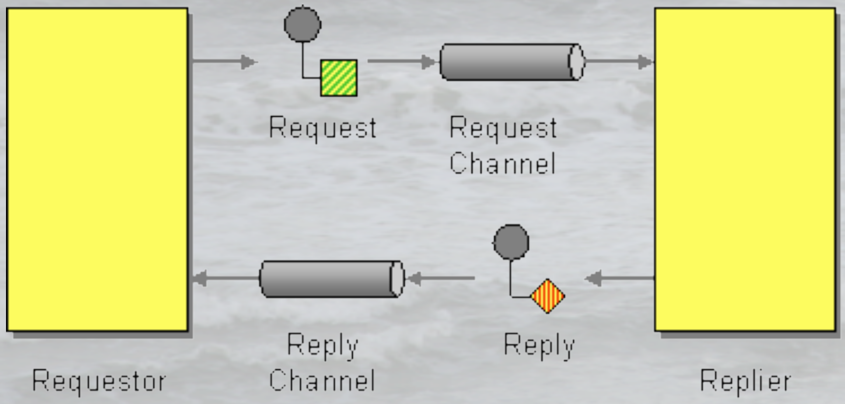
Enterprise Integration Patterns



Guaranteed Message Delivery

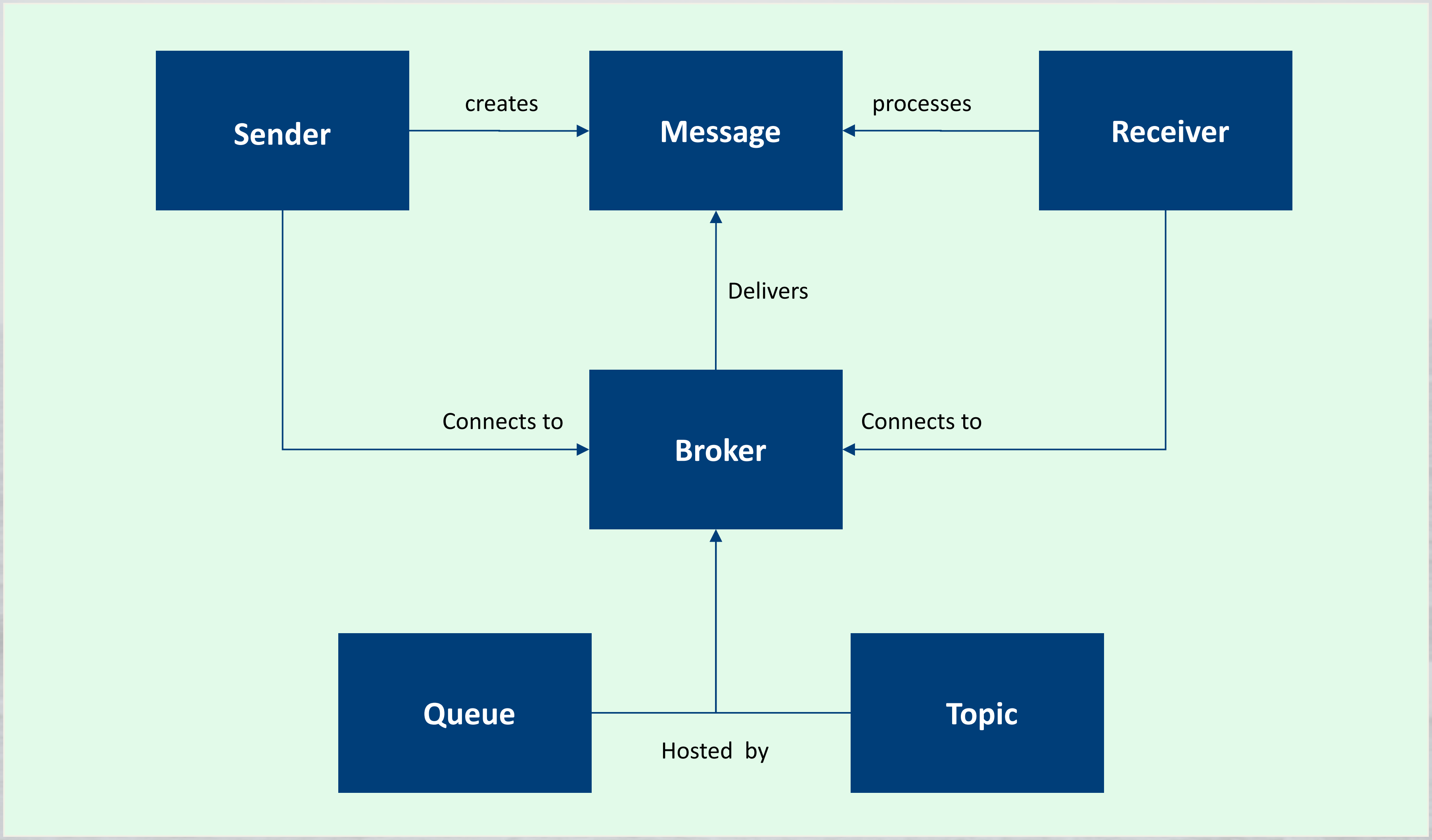


Filter Messages



Asynchronous Request Reply

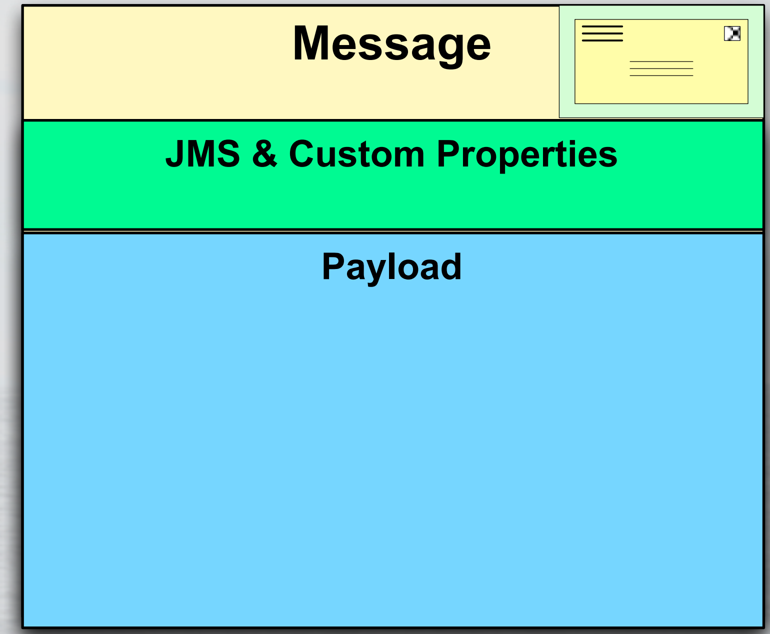
Message Oriented Architecture (MOA)



Java Message Service (JMS)*

Properties (a.k.a. Headers)

- JMS* (e.g. JMSType, JMSCorrelationID, JMSDeliveryMode, JMSExpiration)
- Custom (e.g. MimeType, Token)

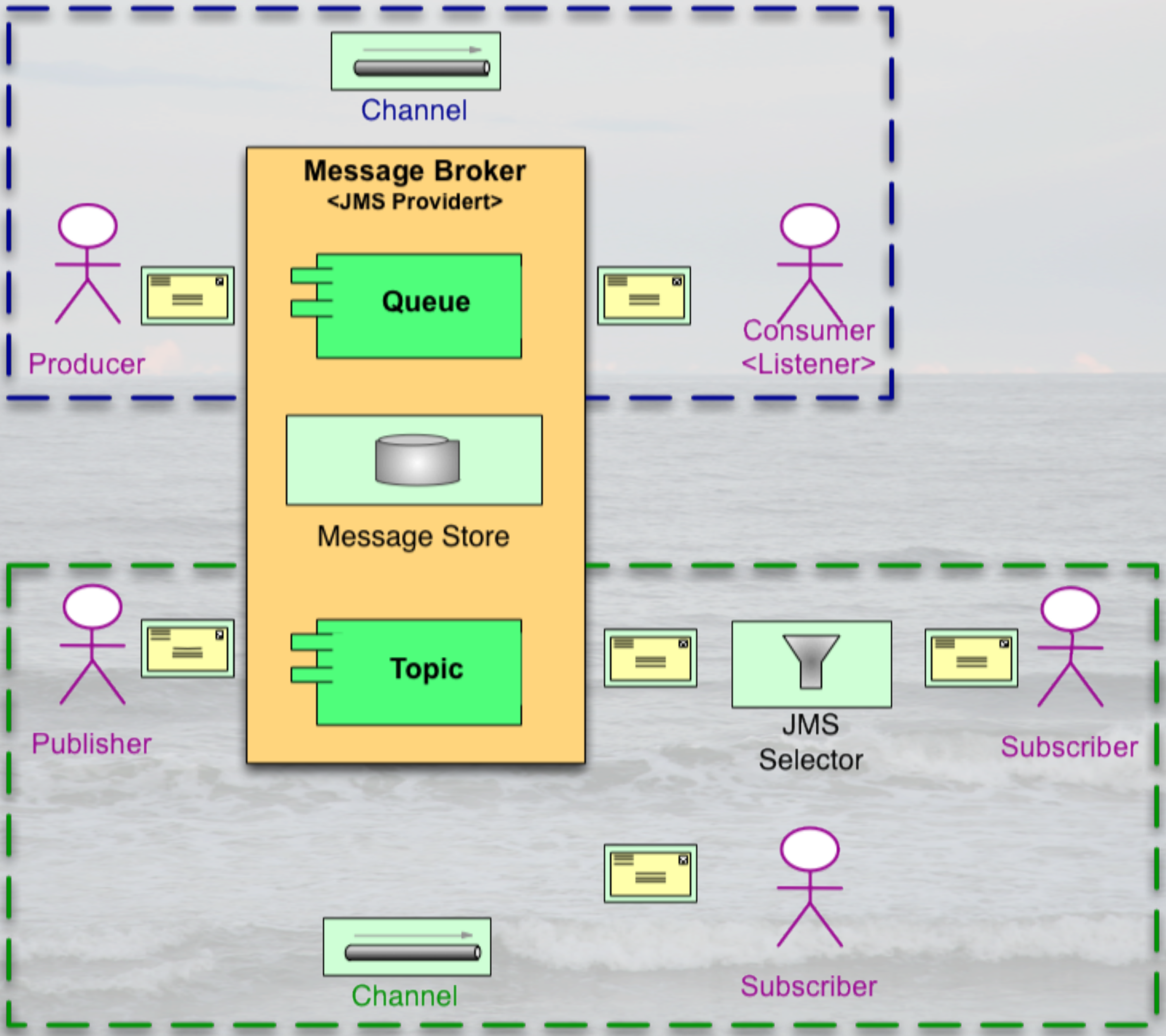


Payload (a.k.a. Body)

- Text
- Object
- Map
- Bytes
- Stream

* Alternative – Advanced Message Queuing Protocol (AMQP)

Java Message Service (JMS)



Apache ActiveMQ

Integration Options

Java Message Service (JMS)
Advanced Message Queuing Protocol (AMQP)

Deployment Flexibility

Stand-alone
Embedded

Advanced Topologies

Master-Slave High Availability (HA)
Federated Network

Support

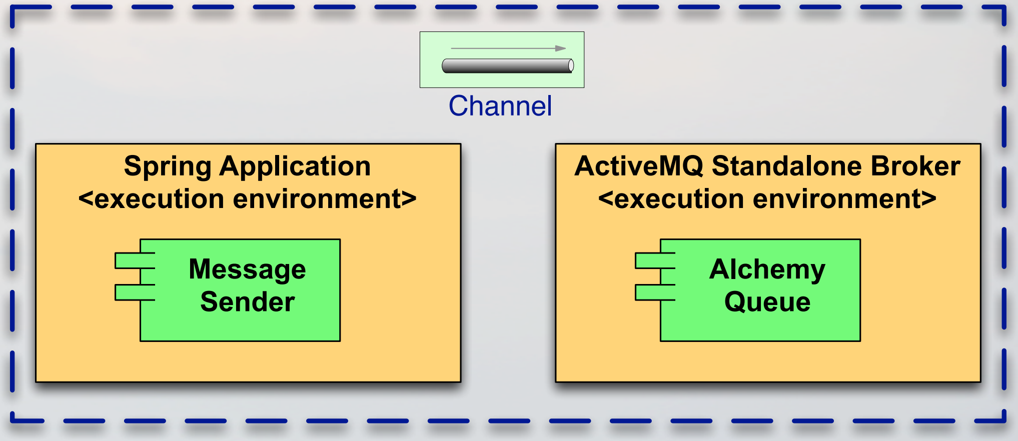
Active Open Source Community
Commercial 24X7 Options

Languages - Transport

- Java-Scala – TCP/NIO
- Ruby, Perl, Python – Stomp
- C# (NMS) –TCP/NIO

Messaging Exercises

Exercise: JMS Sender



1. Start ActiveMQ

```
cd $ACTIVEMQ_HOME/bin
./activemq start OR ./activemq.bat
```

2. Build Client [*Optional*]

```
mvn clean install
cd target
```

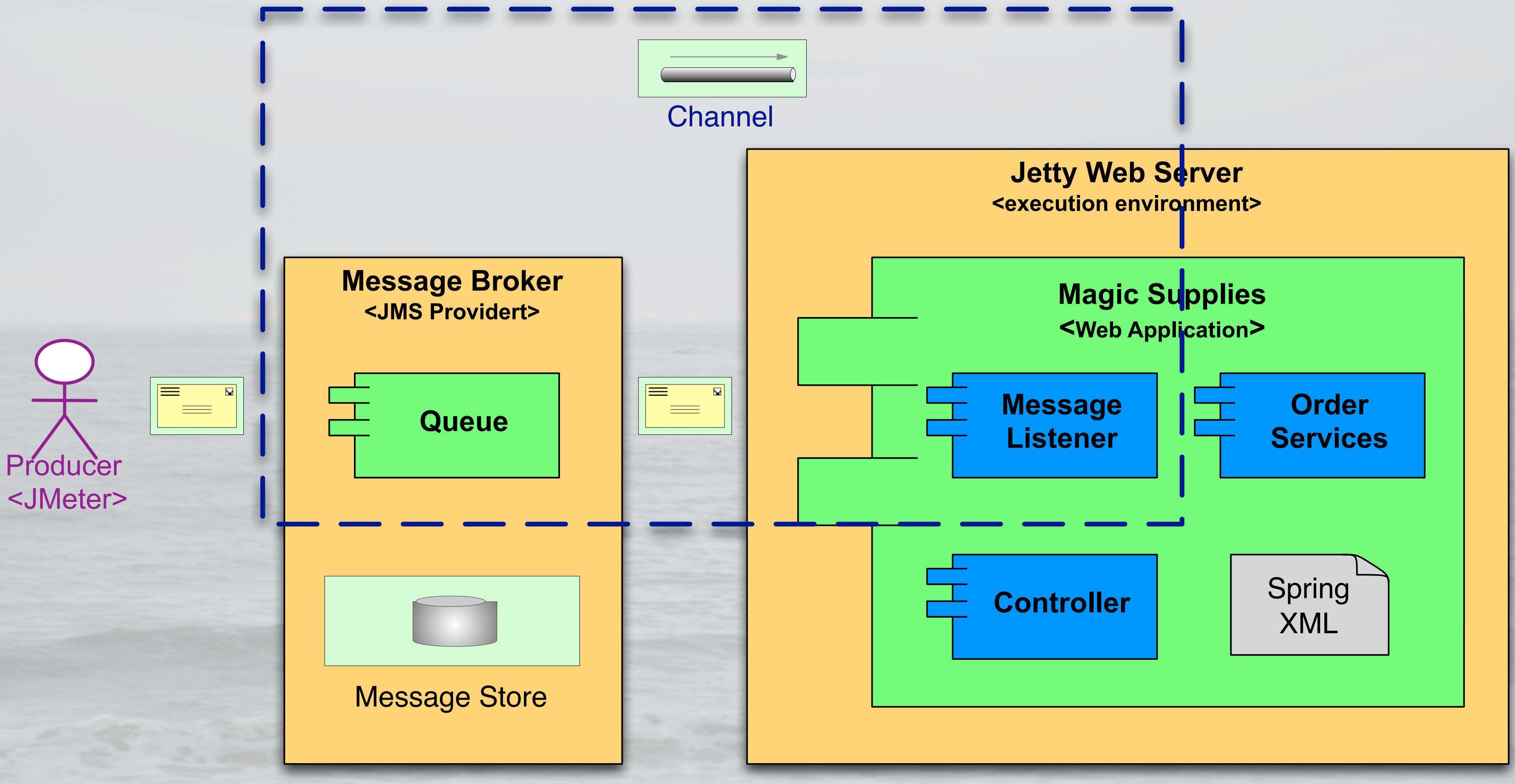
3. Execute JAR

```
java -jar message-client-1.0.0.jar
```

4. View Message from ActiveMQ Web Console (`test.alchemy`)

```
Open http://localhost:8161/admin/ {admin/admin}
Open http://localhost:8161/admin/queues.jsp
```

Demo: JMS Listeners



```
magic-supplies/src/main/webapp/WEB-INF/servlet-context.xml
magic-supplies/src/main/java/cogito/online/messaging/SingleOrderProcessingMessageListener
magic-supplies/src/main/java/cogito/online/messaging/OrderProcessingMessageListener
```

Exercise: JMS Listener

1. Start ActiveMQ

```
cd $ACTIVEMQ_HOME/bin
./activemq start OR ./activemq.bat
```

2. Start Jetty Web Server

```
cd $JETTY_HOME/bin
./jetty.sh start OR java -DSTOP.PORT=8079 -jar start.jar
```

3. Tail Jetty Log

```
cd $JETTY_HOME/logs
tail –f {Current Log}
```

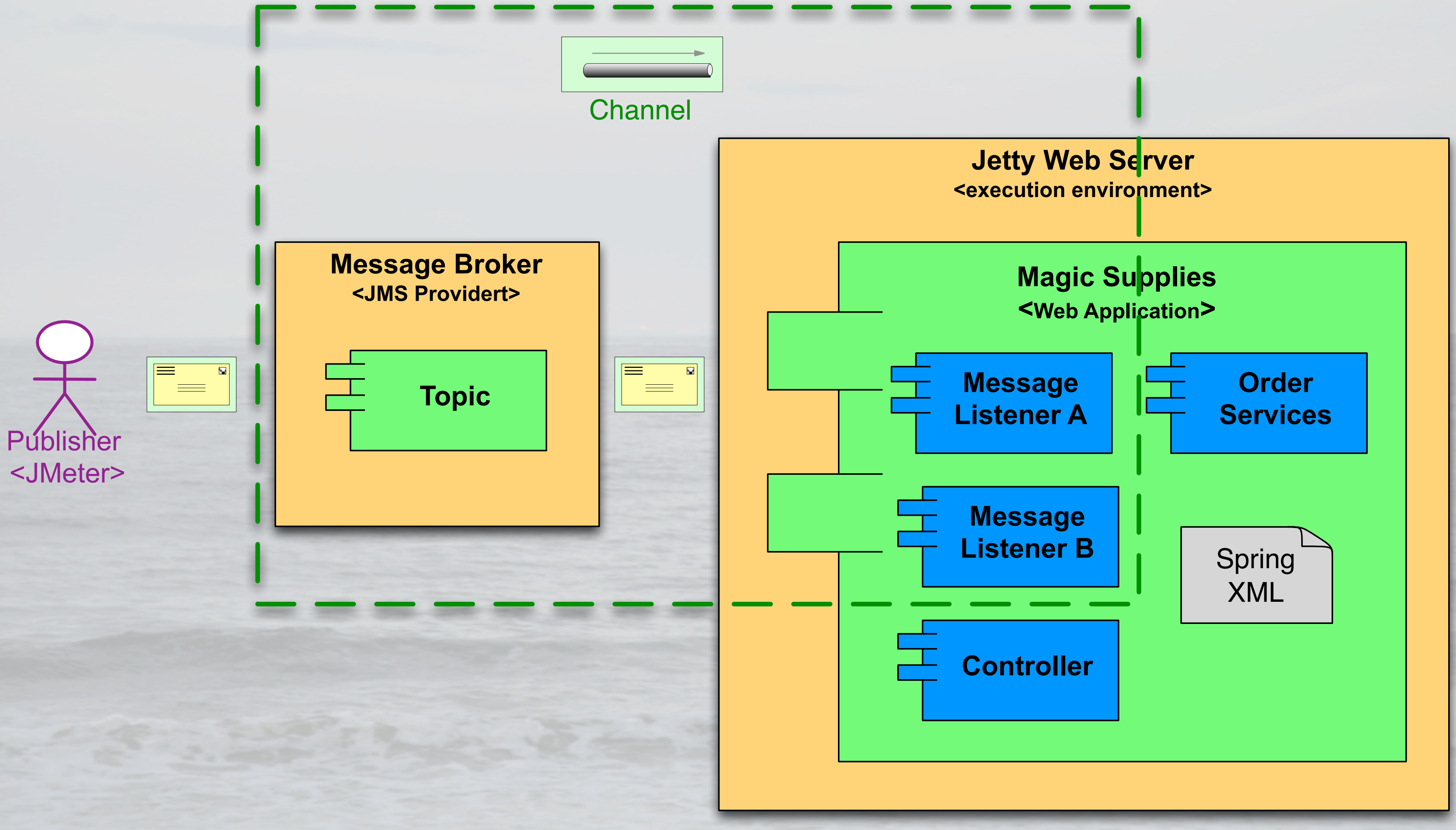
4. Execute JAR

```
java -jar message-client-1.0.0.jar magic.order
java -jar message-client-1.0.0.jar magic.orders
```

5. View Activity on ActiveMQ Web Console

```
Open http://localhost:8161/admin/ {admin/admin}
Open http://localhost:8161/admin/queues.jsp
```

Demo: JMS Subscribers



```
magic-supplies/src/main/webapp/WEB-INF/servlet-context.xml
magic-supplies/src/main/java/cogito/online/messaging/AlertProcessingTopicListener.java
```

Exercise: JMS Subscribers

1. Start ActiveMQ

```
cd $ACTIVEMQ_HOME/bin
./activemq start OR ./activemq.bat
```

2. Start Jetty Web Server

```
cd $JETTY_HOME/bin
./jetty.sh start OR java -DSTOP.PORT=8079 -jar start.jar
```

3. Tail Jetty Log

```
cd $JETTY_HOME/logs
tail –f {Current Log}
```

4. Execute JAR

```
java -jar message-client-1.0.0.jar magic.alerts
```

5. View Activity on ActiveMQ Web Console

```
Open http://localhost:8161/admin/ {admin/admin}
Open http://localhost:8161/admin/topics.jsp
```

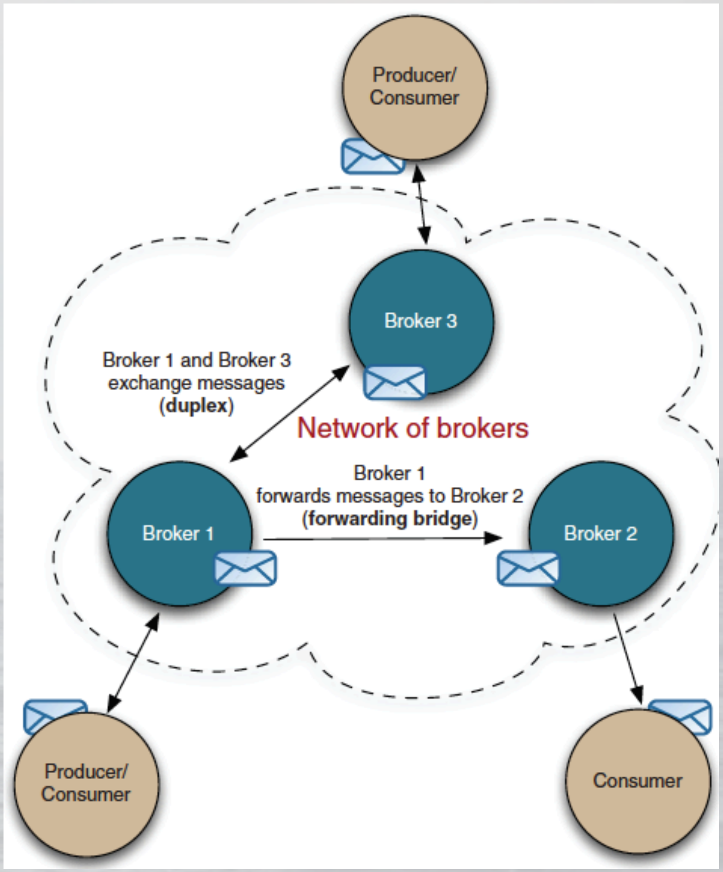
Messaging Networks

ActiveMQ Broker Topologies

- Embedded Broker (VM Transport)
- Standalone Broker (a.k.a. Client-Server)
- Load Balanced Brokers

```
failover:(tcp://BrokerA:61616,tcp://BrokerC:61616)?
randomize=true
```

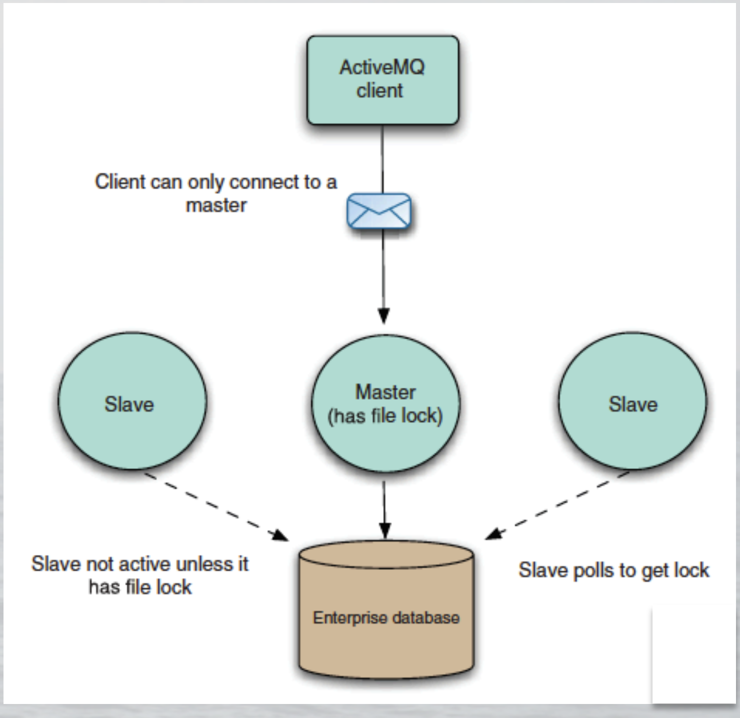
- Networked Brokers (a.k.a. Store and Forward)



ActiveMQ in Action by Bruce Snyder, Dejan Bosanac, and Rob Davies

ActiveMQ High Availability and Disaster Recovery

- Master Slave – File System
    - KahaDB
    - SAN-NFS4
- Master Slave – JDBC
    - RDBSM (e.g. PostgreSQL)
- Master Slave – Replicated
    - LevelDB & Zookeep
- Failover Protocol (*use domain aliases*)



ActiveMQ in Action by Bruce Snyder, Dejan Bosanac, and Rob Davies

```
failover:(tcp://PrimaryBroker:61616,tcp://SecondaryBroker:61616)?randomize=false
```

ActiveMQ Error Handling

- Dead Letter Queue (DLQ)
  - Review (Web Console)
  - Redeliver (Plug-in)
  - Remediate (Policy)
- Monitor DLQ Depth
  - Alerts (Splunk Example)
- Failed Message Transactions
  - Rollback
  - DLQ
- Throw or Log Exception

ActiveMQ Security

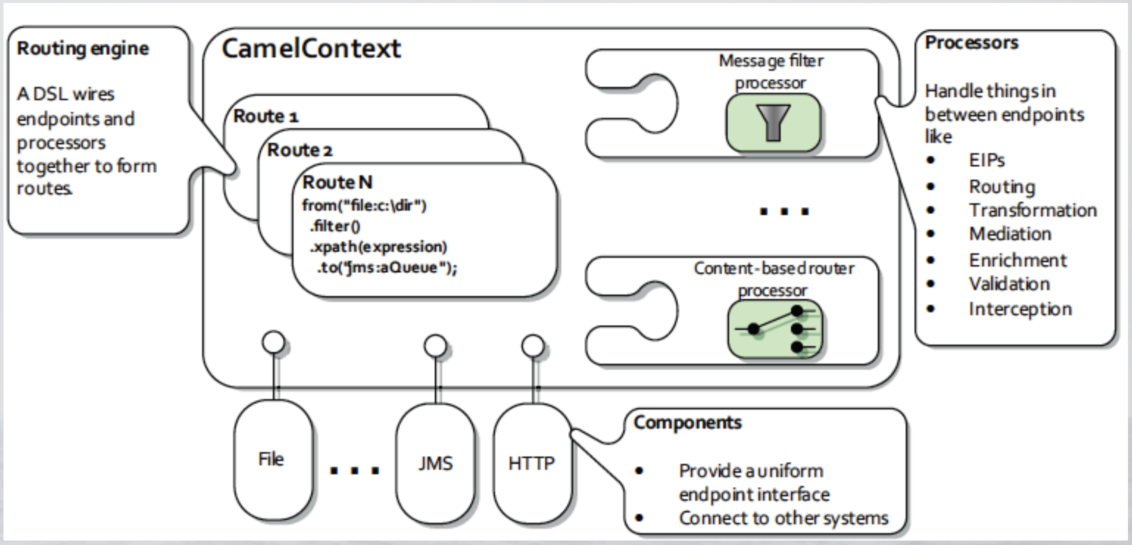
- Authentication and Authorization
  - Web Console
  - JMX Connector (Bind to specific Port #)
  - Queues and Topics

- Auditing
  - Monitor Logs
  - Periodically Audit Access

- Confidentiality
  - Transport (SSL)
  - Messages (Encryption)
  - KahaDB Files (Encryption)
  - Log Files (Encryption)

Advanced Messaging

Apache Camel – Enterprise Integration Library



[Camel In Action]() by Claus Ibsen and Jonathan Anstey

- Domain Specific Languages (**DSL**) – Java, Scala, Spring DSL

- Route Builders – create **Endpoints** (i.e. from & to) using **Components** (e.g. protocol) and **Processors** (e.g. Mediation, Enrichment

- **Route Engine** – Loads and executes **Routes**

Apache Camel Embedded – ActiveMQ Integration

1. Create and configure Camel ActiveMQ Component (camel.xml)

```
$ACTIVEMQ_HOME/conf/camel.xml
```

2. Configure ActiveMQ Configuration (activemq.xml) to import `camel.xml` {*update this file – uncomment import*}

```
cd $ACTIVEMQ_HOME/conf/activemq.xml

<!-- Uncomment to enable embedded camel routes
<import resource="camel.xml"/>-->
```

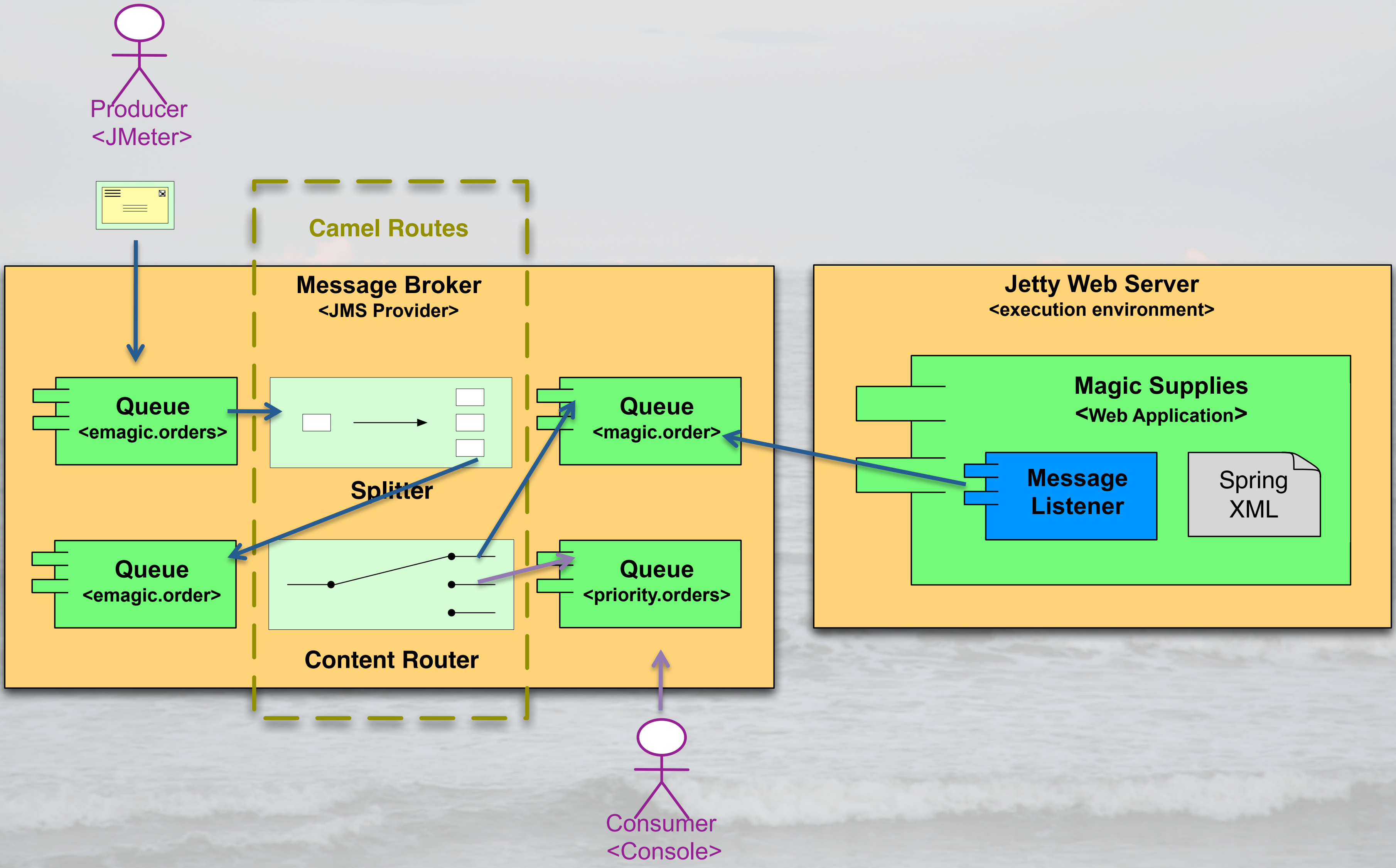
3. Add any required Camel JARs (e.g. camel-mail or camel-http)

```
$ACTIVEMQ_HOME/lib/camel
```

4. Deploy custom routers (i.e. Spring, Java, or Scala DSL) as JAR

```
$ACTIVEMQ_HOME/lib
```

Exercise: Content-Based Router



Exercise: Content-Based Router - Instructions

1. Extract camel-magic-router.zip contents to $HOME

2. Build project and import into IDE (e.g. Eclipse, IntelliJ)

```
mvn clean install
mvn eclipse:eclipse
```

3. Open MagicRouteBuilder.java

4. Implement Splitter Pattern

```
XPathBuilder splitXPath = new XPathBuilder (splitXpath);

from("activemq:emagic.orders").
     split(splitXPath).
     parallelProcessing().
to("activemq:emagic.order");
```

Exercise: Content-Based Router - Instructions

5. Implement Content Based Router Pattern

```
from("activemq:emagic.order").
choice().
     when().simple("${in.body} contains 'Houdini'").
          to("activemq:priority.order").
     otherwise().
          to("activemq:magic.order");
```

6. Update pom.xml to support Wagon deployment

```
<plugin>
     <groupId>org.codehaus.mojo</groupId>
     <artifactId>wagon-maven-plugin</artifactId>
…
     <configuration>
          <fromDir>${project.build.directory}</fromDir>
          <includes>*.jar</includes>
          <!-- Update to location of your ActiveMQ Lib Directory -->
          <url>file:///opt/servers/activemq</url>
          <toDir>lib</toDir>
     </configuration>
</plugin>
```

Exercise: Content-Based Router - Instructions

7. Build and deploy to ActiveMQ

```
mvn clean install
mvn wagon:upload
```

8. Stop/Start ActiveMQ and then Jetty (see ActiveMQ Exercises)

9. Execute Message Client

```
java -jar message-client-1.0.0.jar emagic.orders
```

10. Open ActiveMQ Web Console and view `priority.order` queue

```
Open http://localhost:8161/admin/ {admin/admin}
Open http://localhost:8161/admin/queues.jsp
```

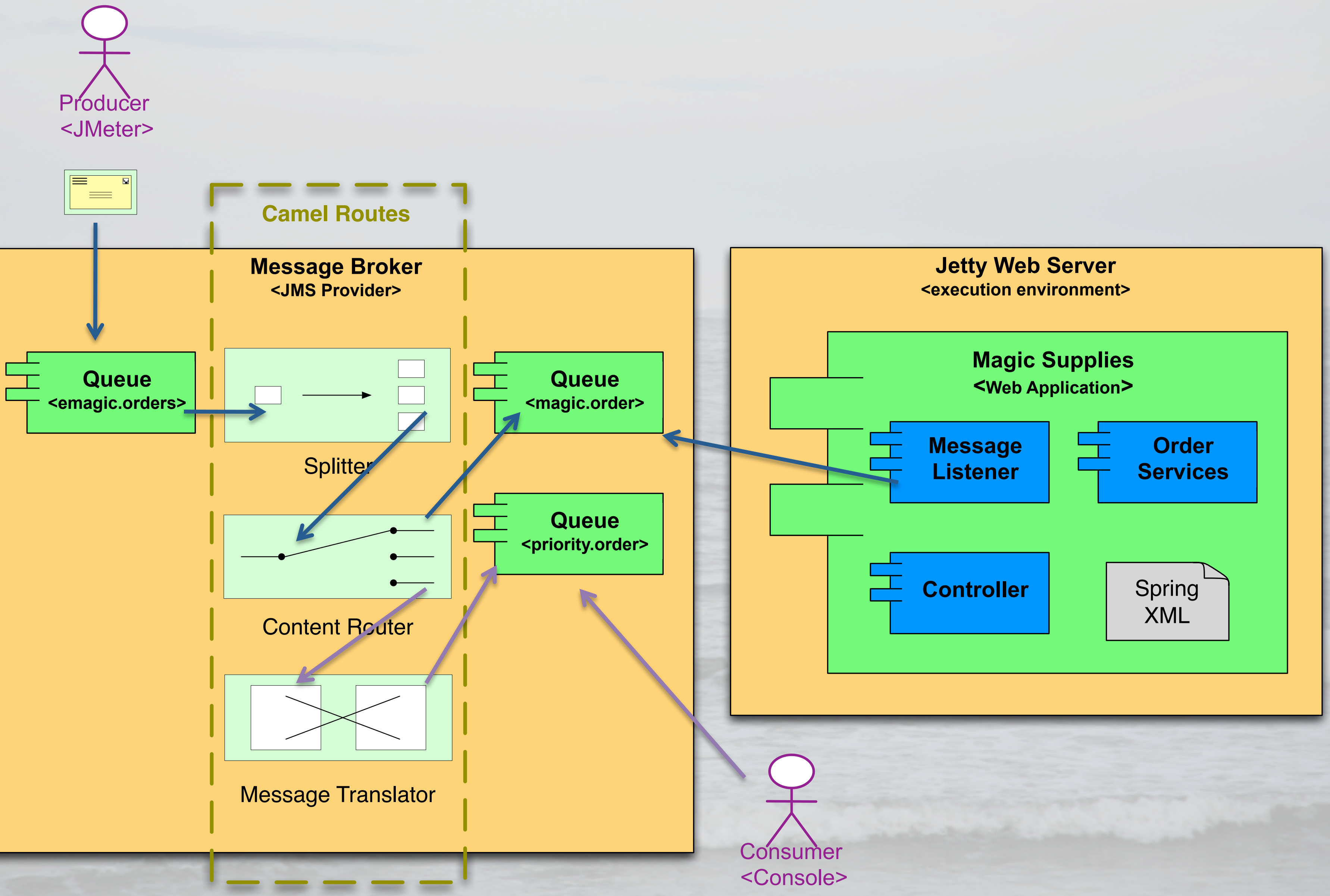
11. View Jetty log for orders processed via `magic.order` queue

```
cd $JETTY_HOME/logs
tail –f {Current Log}
```

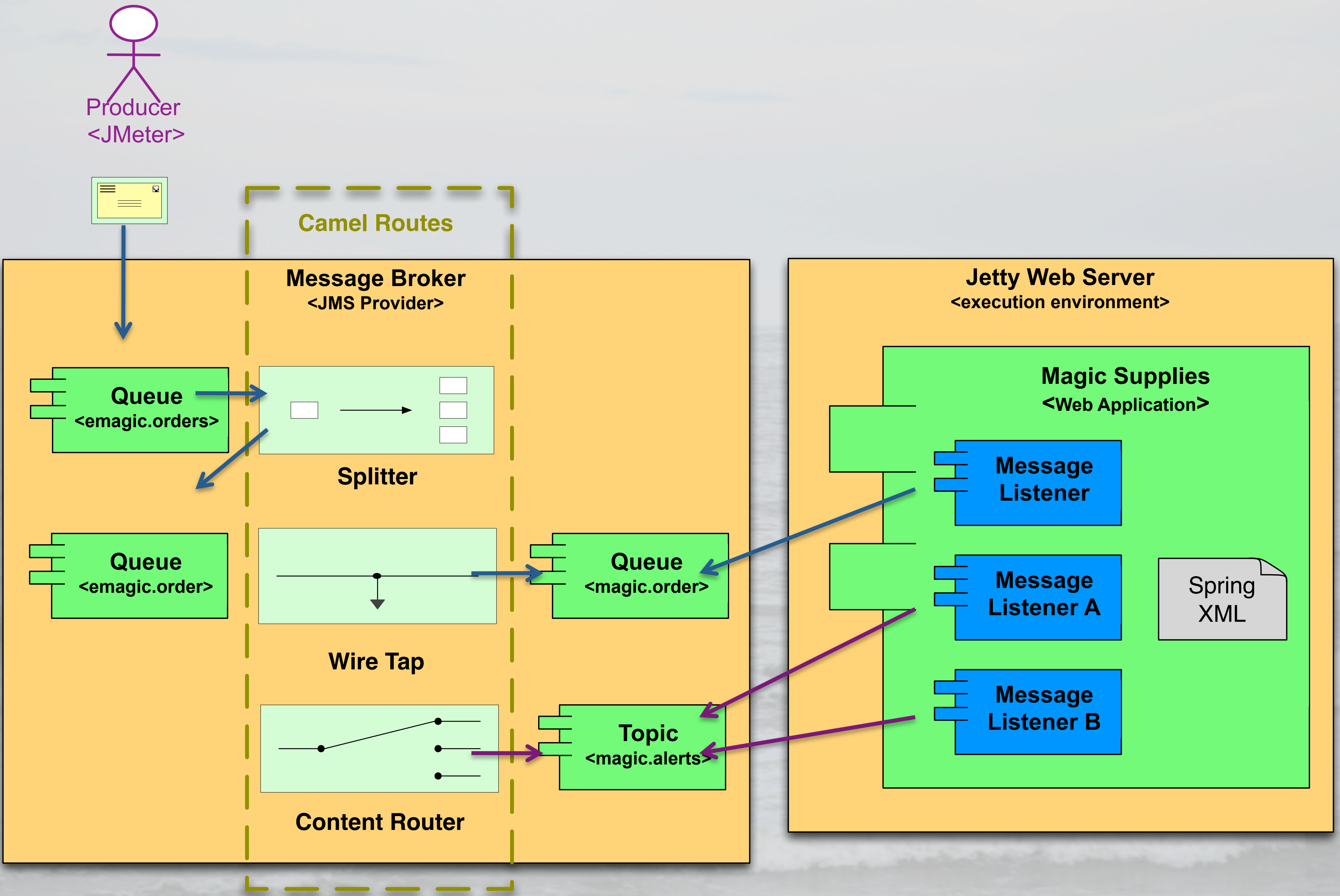
12. View ActiveMQ log for any issues

```
$ACTIVEMQ_HOME/data/activemq.log
```

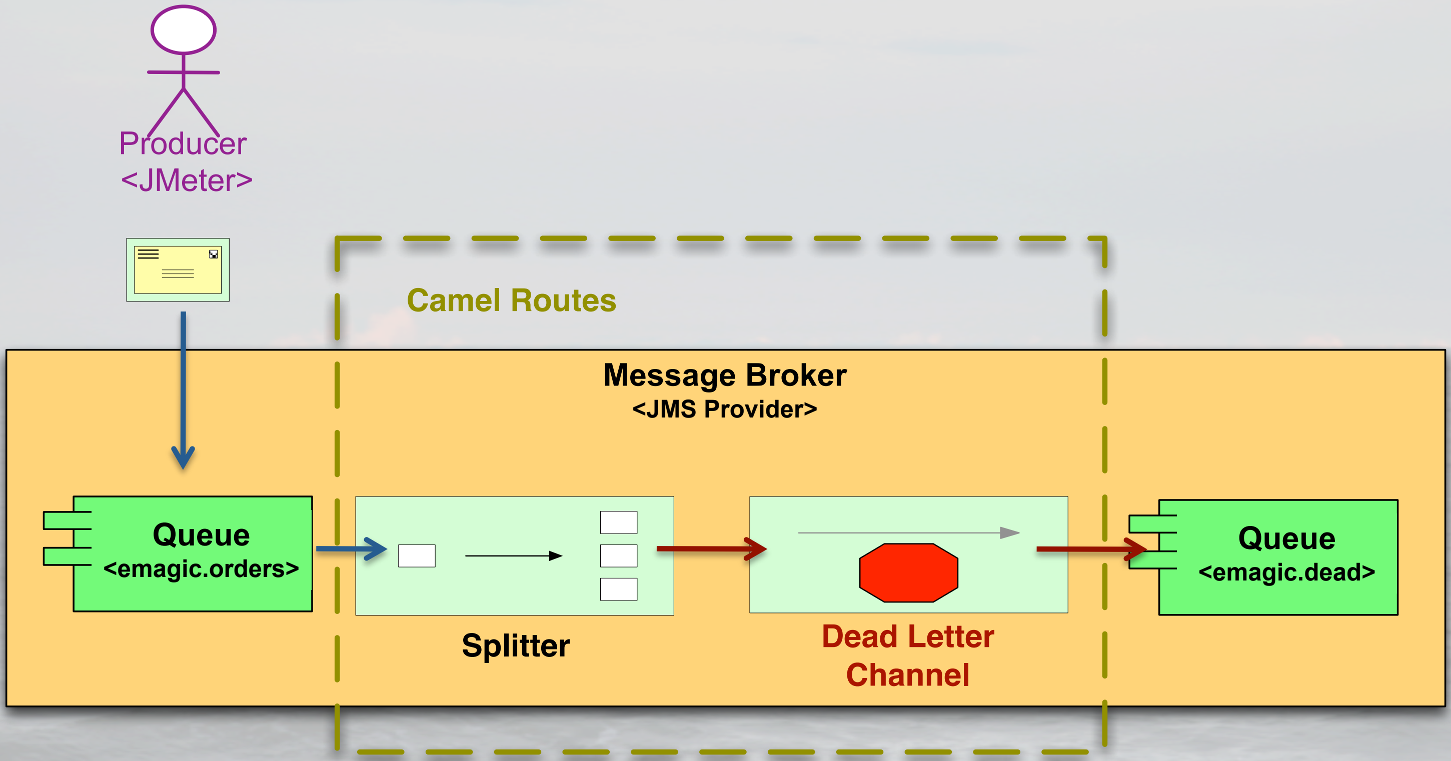
Demo: Mediation



Demo: Wire Tap



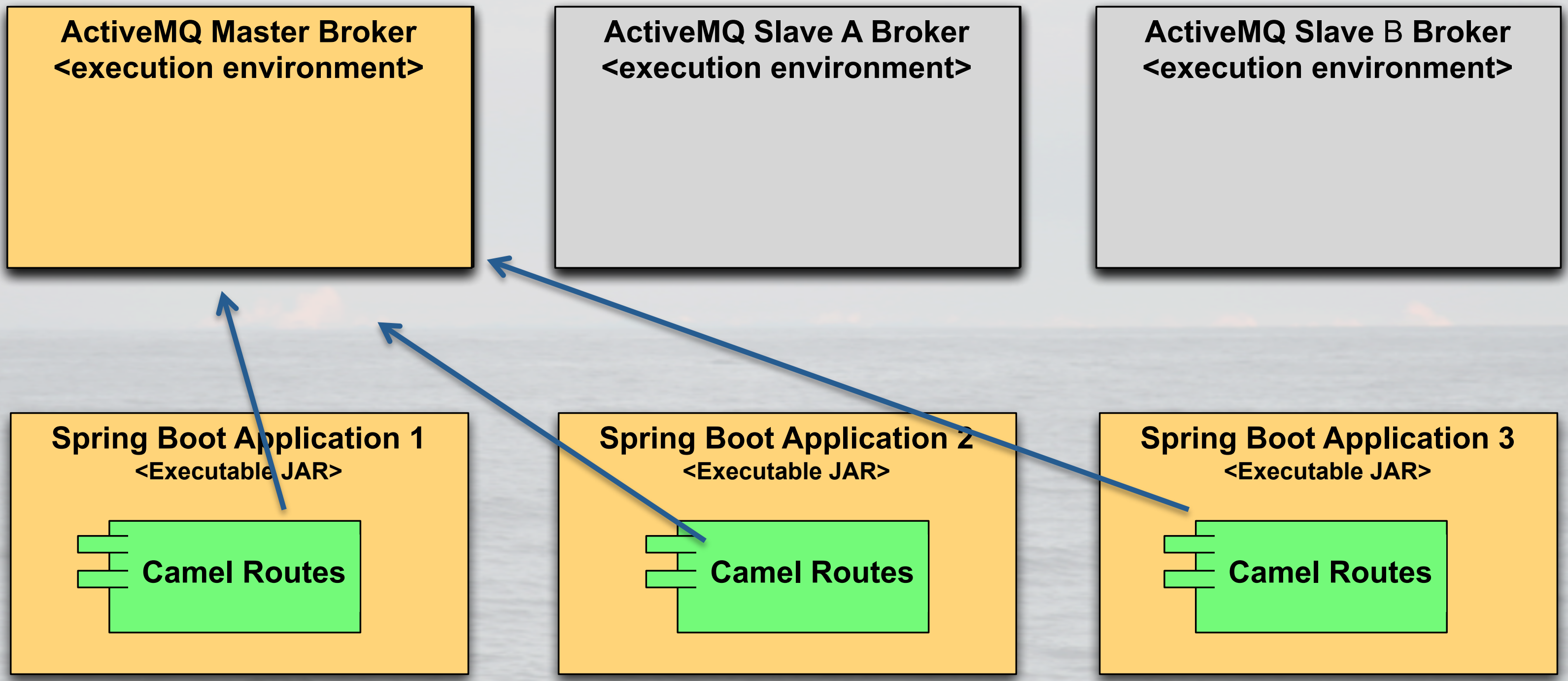
Demo: Dead Letter Channel



```
errorHandler(deadLetterChannel("activemq:emagic.dead").maximumRedeliveries(1).redeliveryDelay(1000));
```

Extensible Messaging

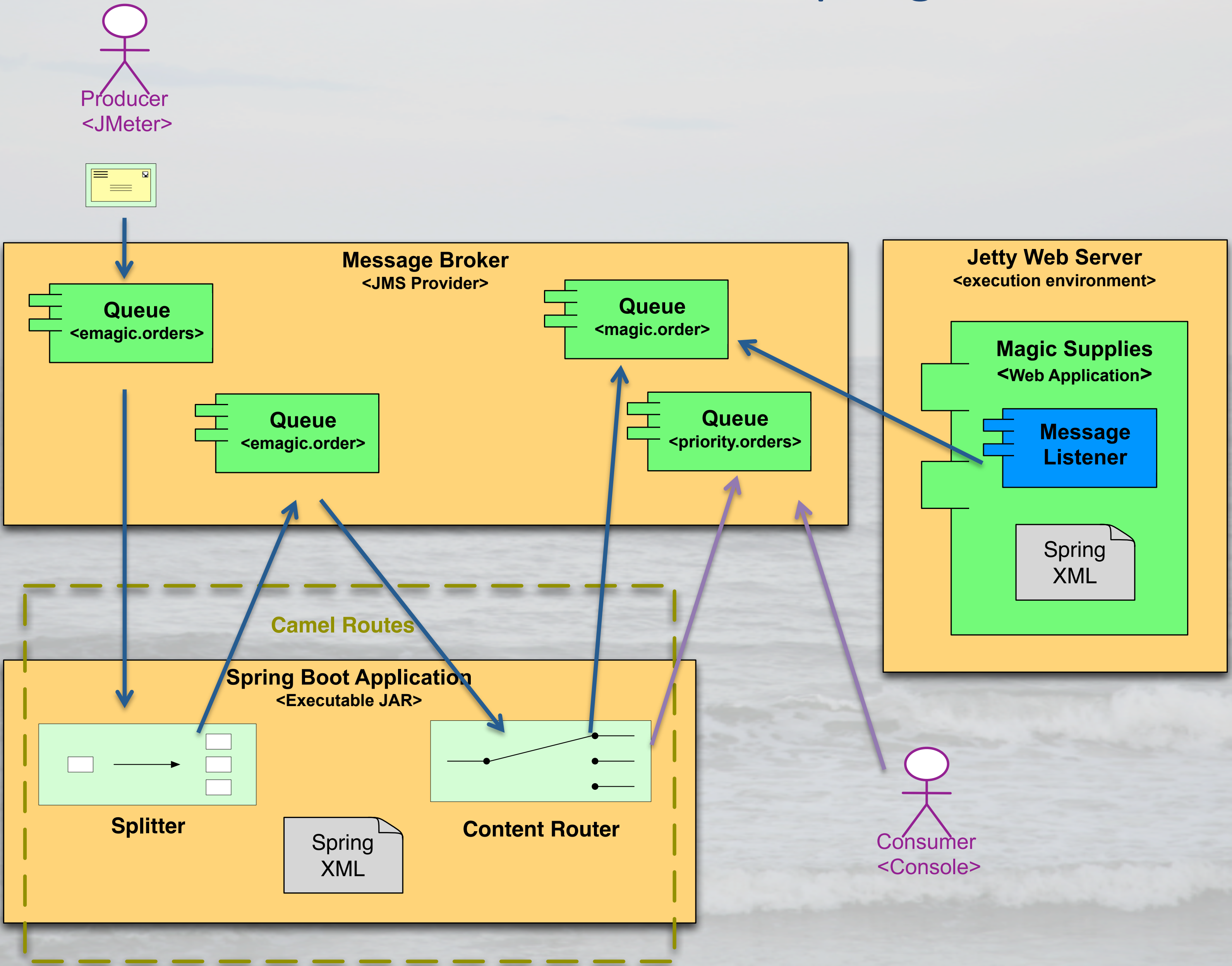
Apache Camel Standalone – ActiveMQ Integration



Primary Benefits:

- Resiliency – Camel Route failure does not take down Broker
- Extensibility – Swap out Message Broker or EIP Framework
- Continuous Development – No Broker or Web Server downtime

Exercise: Camel Standalone Router – Spring Boot



Exercise: Camel Standalone Router - Instructions

1. Extract camel-standalone-router.zip contents to $HOME

2. Build project and import into IDE (e.g. Eclipse, IntelliJ)

```
mvn clean install
mvn eclipse:eclipse
```

3. View MagicRouteBuilder.java & MagicRouterApplication.java

4. View camel-route-spring.xml

5. Update ActiveMQ Configuration (activemq.xml) and comment out camel.xml import

```
cd $ACTIVEMQ_HOME/config
<!-- <import resource="camel.xml"/> -->
```

6. Restart ActiveMQ & Jetty (see ActiveMQ Exercises)

Exercise: Camel Standalone - Instructions

7. Execute Spring Boot Application

```
mvn spring-boot:run
```

8. Execute Message Client

```
java -jar message-client-1.0.0.jar emagic.orders
```

9. Open ActiveMQ Web Console and view `priority.order` queue

```
Open http://localhost:8161/admin/ {admin/admin}
Open http://localhost:8161/admin/queues.jsp
```

10. View Jetty log for orders processed via `magic.order` queue

```
cd $JETTY_HOME/logs
tail –f {Current Log}
```

11. View Camel Standalone - JConsole

```
jconsole {pid}
```

Exercise: Camel Standalone - Instructions

12. Bonus Add: Wire-Tap Splitter

```
from("activemq:emagic.orders").split(splitXPath).parallelProcessing().wireTap("direct:ministry
").to("activemq:emagic.order");

from("direct:ministry").choice().when().simple("${in.body} contains 'Elder
Wand'").log("ILLEGAL MAGIC ALERT").to("activemq:topic:magic.alerts").otherwise().log("...off
into the ether");
```

13. Stop Spring Boot and Repeat Steps: #2, 7-8

14. Open ActiveMQ Web Console

```
Open http://localhost:8161/admin/ {admin/admin}
Open http://localhost:8161/admin/queues.jsp
Open http://localhost:8161/admin/topics.jsp
```

Questions & Feedback

My Contact information:

Jeremy Deane
Director of Architecture
NaviNet
jeremy.deane@gmail.com
http://jeremydeane.net

https://github.com/jtdeane/magic-supplies

https://github.com/jtdeane/message-client

https://github.com/jtdeane/camel-magic-router

https://github.com/jtdeane/camel-standalone-router