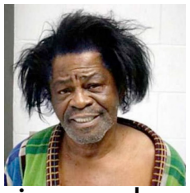
How to Ride with Apache Camel to Get Over Development Humps

Jeff Genender
CTO

savoir
TECHNOLOGIES

Bio - Who Am I?



jgenender

savoir TECHNOLOGIES

Java Community Process

JSR 366 - Java EE 8

Java™ Champions

JavaOne 2013 Rock Star

The Apache Software Foundation
http://www.apache.org/

Apache Camel

Apache TomEE™

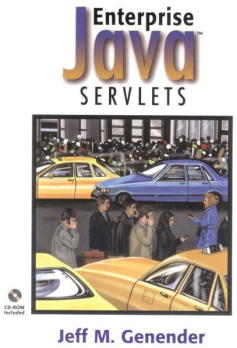
Apache CXF

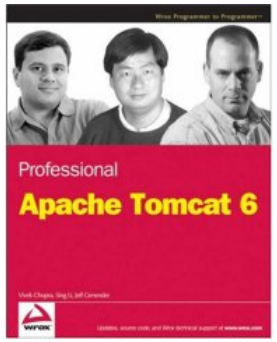
ServiceMix

Apache MINA

APACHE GERONIMO

mojo



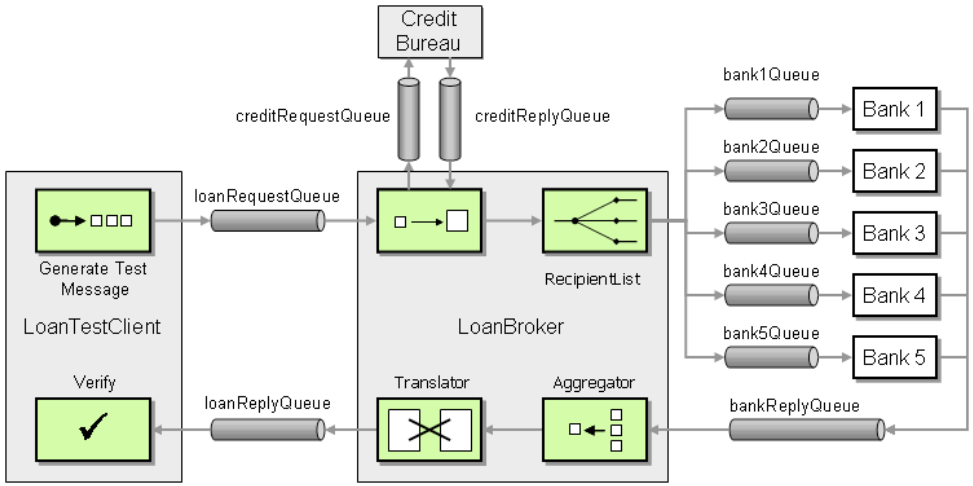


What is Apache Camel?

- A light weight integration framework.
  - Implements EIPs.
  - Standardizes communications
  - Flexible deployment models
  - Very large component library
  - Variety of domain-specific languages
  - Best of all…. Open Source!

What is Apache Camel?

- A plethora of EIPs (Enterprise Integration Patterns)
  - Most of the EIPs are supplied via processors
  - Examples:
    - Splitter
    - Multicast
    - Aggregator
    - Wiretap
    - Throttle
    - CBR (Content Based Routing)
    - And the list goes on……..
  - Custom/Developer produced processor
    - Makes the possibilities endless



What is Apache Camel?

- **Standardized Communications**
  - Camel provides an ever growing list of components (183 at last count)
  - Examples:
    - JMS/ActiveMQ
    - JMX
    - File
    - HTTP
    - CXF
    - http://camel.apache.org/components.html (for a more complete list)

What is Apache Camel?

- **Domain specific languages (DSL)**
  - **Java**

```
from(“amq:queue.order.pizza”).wireTap(“seda:cook.pizza”).to(“amq:queue.billing.pizza”)
```

  - **XML (Spring or Blueprint)**

```
<route>
   <from uri=“amq:queue.order.pizza”/>
   <wireTap uri=“seda:cook.pizza”/>
   <to uri=“amq:queue.billing.pizza”/>
</route>
```

  - **Scala**

```
"amq:queue.order.pizza" wireTap("seda:cook.pizza") to "amq:queue.billing.pizza"
```

What is Apache Camel?

- Flexible deployment models
  - Stand alone applications
  - Web servers
    - Jetty
    - Tomcat
  - Application servers
    - JBoss
    - Websphere
  - OSGI
    - Karaf
  - Messaging
    - ActiveMQ
    - JBossMQ

Why Apache Camel?

- **Simplify code and reducing effort**

This? or This?

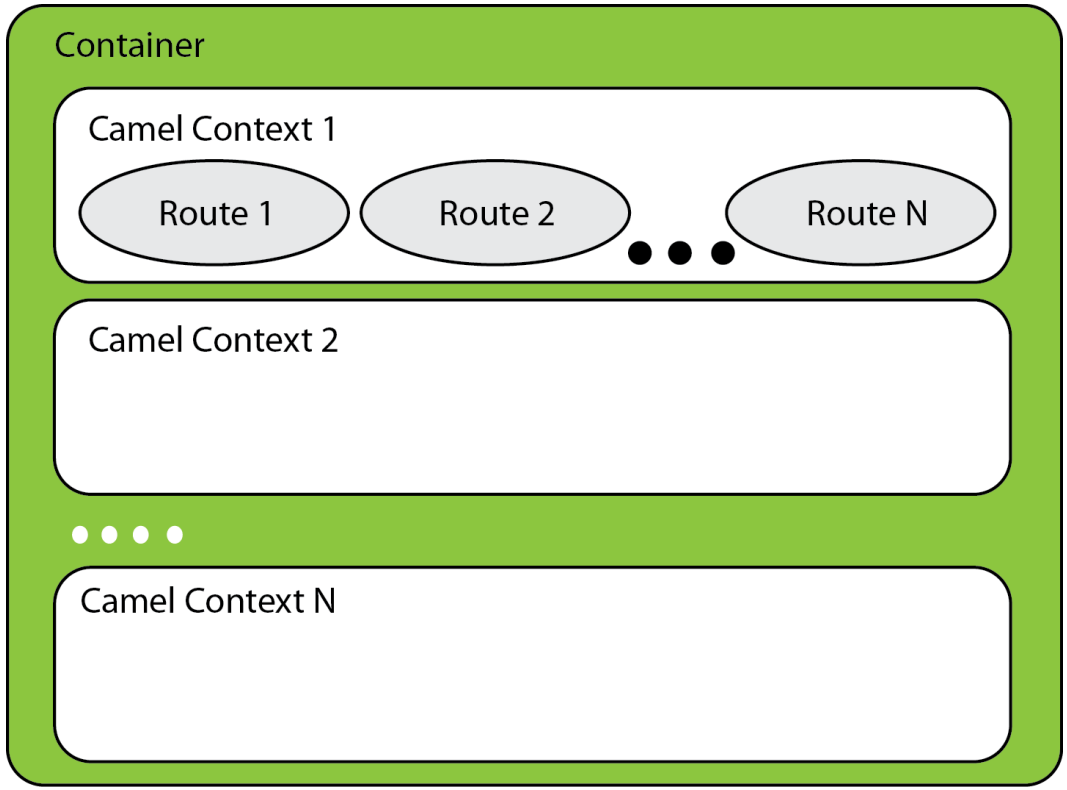
```
try {

        // Create a ConnectionFactory
        ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory("vm://localhost");

        // Create a Connection
        Connection connection = connectionFactory.createConnection();
        connection.start();

        connection.setExceptionListener(this);

        // Create a Session
        Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

        // Create the destination (Topic or Queue)
        Destination destination = session.createQueue("TEST.FOO");

        // Create a MessageConsumer from the Session to the Topic or Queue
        MessageConsumer consumer = session.createConsumer(destination);

        // Wait for a message
        Message message = consumer.receive(1000);

        if (message instanceof TextMessage) {
            TextMessage textMessage = (TextMessage) message;
            String text = textMessage.getText();
            System.out.println("Received: " + text);
        } else {
            System.out.println("Received: " + message);
        }

        consumer.close();
        session.close();
        connection.close();
    } catch (Exception e) {
        System.out.println("Caught: " + e);
        e.printStackTrace();
    }
```

```
from ("amq:queue:TEST.FOO")...
```

How Apache Camel works

Camel Context

- N number Camel Context in a Container
- Defining N number of route builder(s)
- Done in code or XML (Spring or Blueprint)



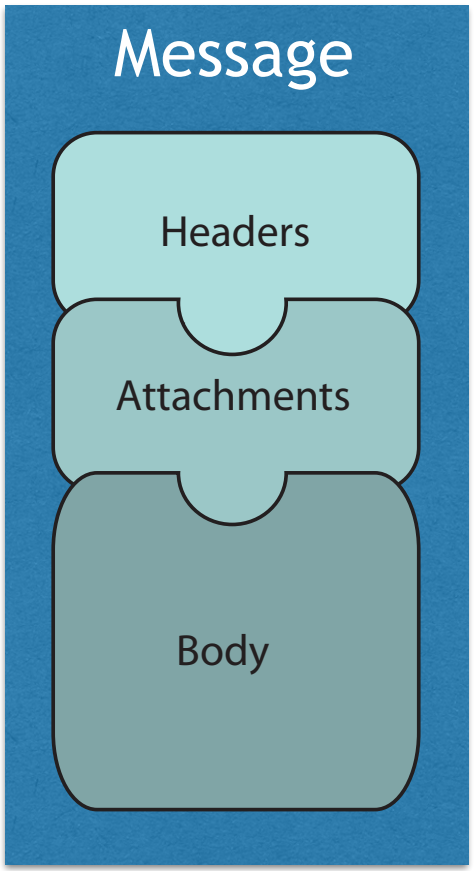
How Apache Camel works

- **Initializing a Camel Context**
  - **Using XML (Spring or Blueprint)**

```
<camelContext id="camel1" xmlns="http://camel.apache.org/schema/spring">
   <route>
      <from uri="direct:one"/>
      <to uri="mock:result"/>
   </route>
</camelContext>
```

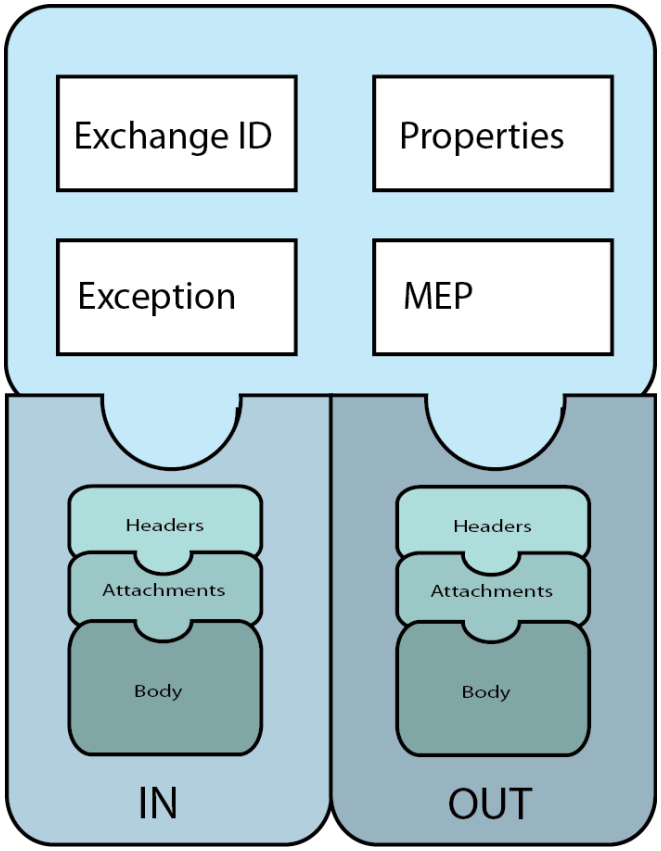
Message

- Message Model
  - Header
  - Attachments
  - Body



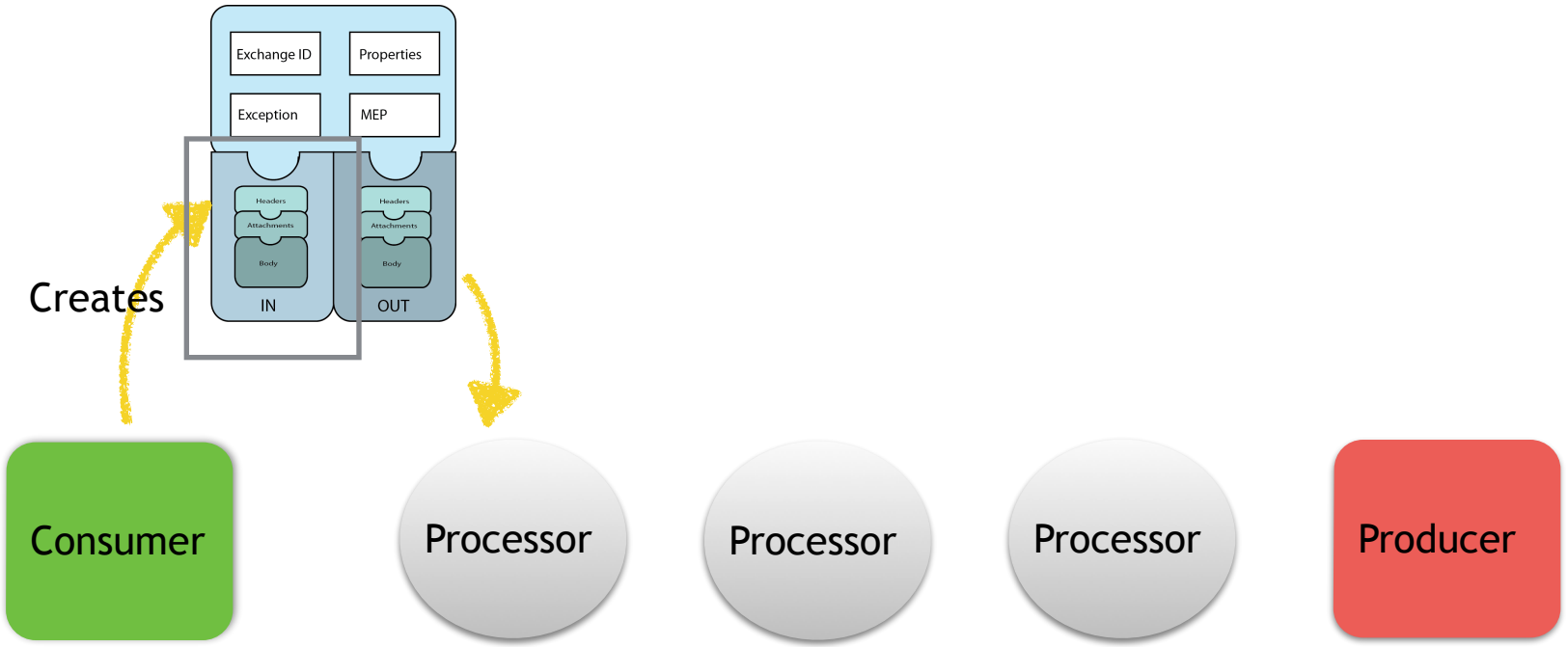
Exchange

- Message container during routing
- Contents of the Exchange
  - Headers
    - Exchange ID
    - Properties
    - Exception
    - Message Exchange Pattern
  - In Message
  - Out Message



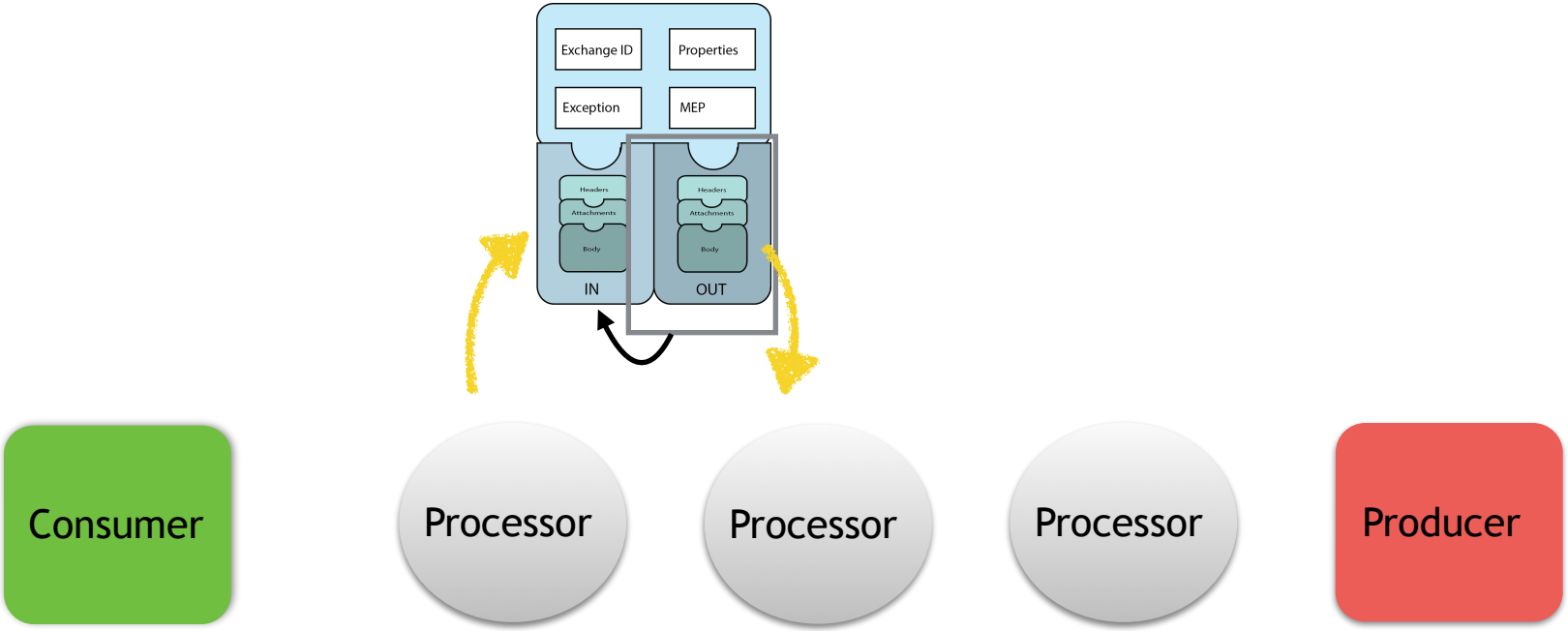
How the Exchange is used

- In Message
  - Consumer will create the exchange
  - It will put the message in the IN of the new exchange
  - And call the next process in the route.



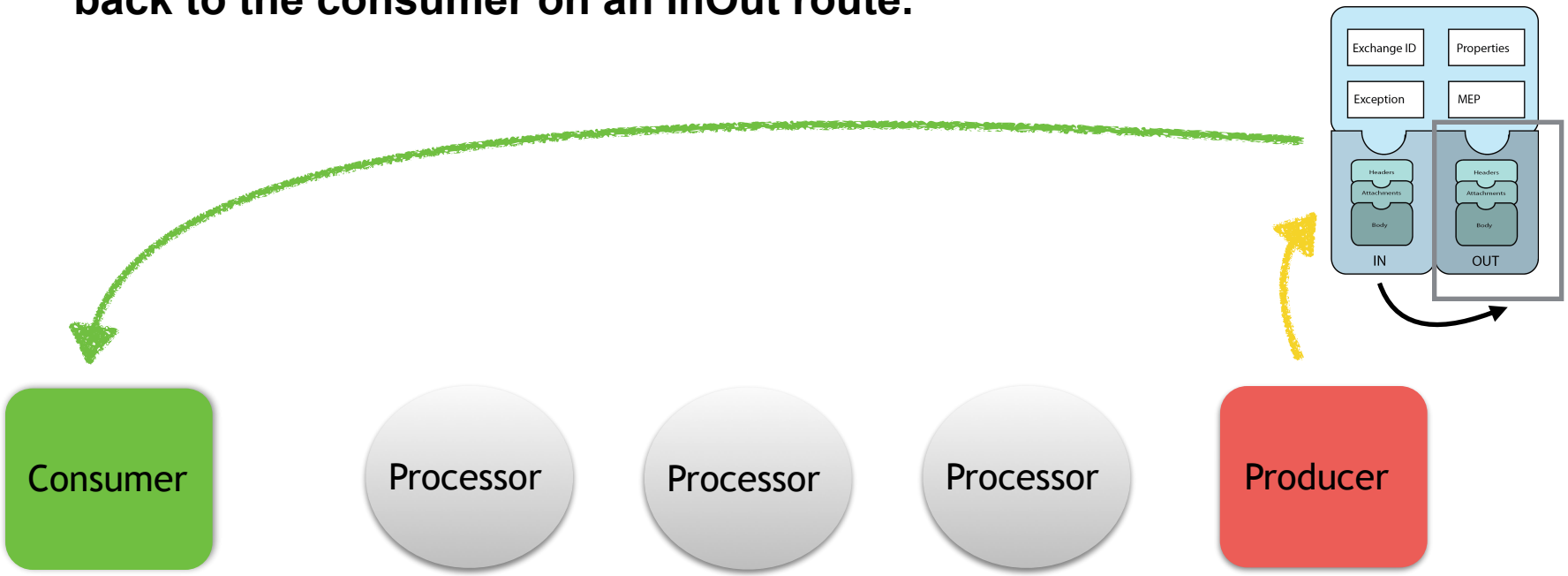
How the Exchange is used

- IN Message
  - The IN message is modified by each process in the route.
  - A processor may set the message in the OUT, but in this case, Camel will copy the OUT message to the IN, and set the OUT to null, before giving it to the next processor



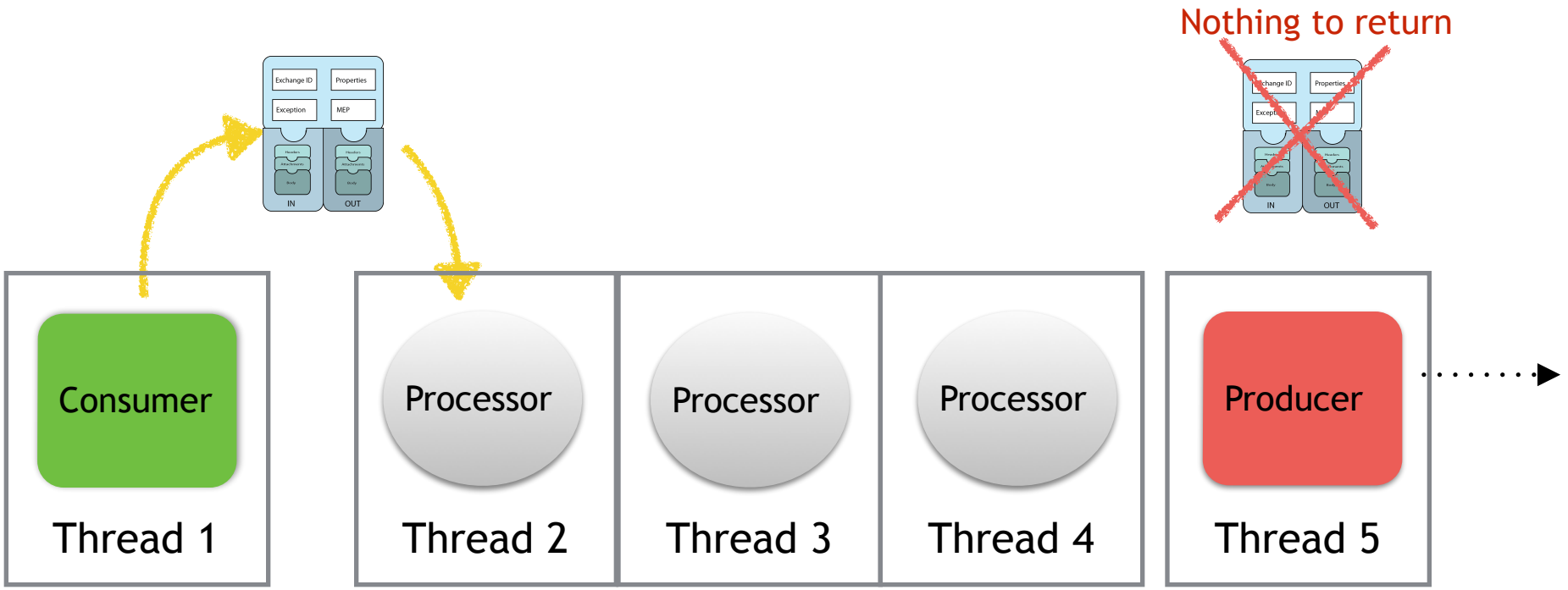
How the Exchange is used

- OUT Message
  - The OUT message is populated by the last producer in the route when the MEP is InOut.
  - If the last producer in the route does not populate the OUT message, then Camel will take the IN and copy it to the OUT before sending it back to the consumer on an InOut route.



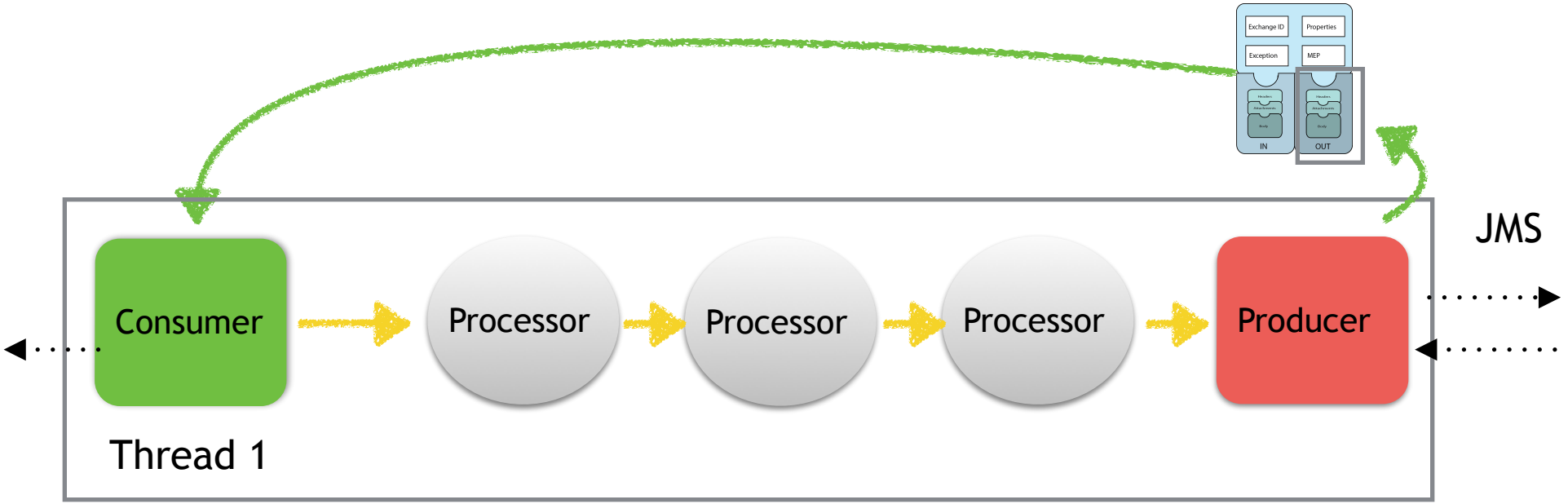
How the Message Exchange Pattern is used

- MEP - InOnly (Asynchronous)
  - Consumer will create the exchange
  - Each process runs in its own thread
  - After passing the exchange to the next process the Consumer will return and process the next message
  - Consumer/Producer does not wait on a return



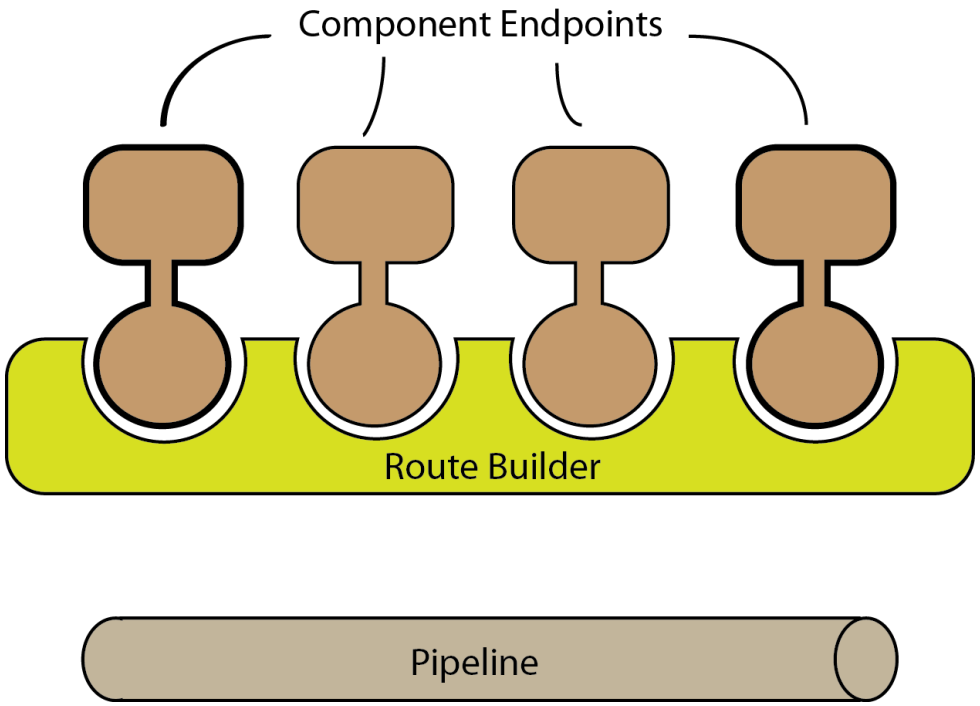
How the Message Exchange Pattern is used

- MEP - InOut (Synchronous)
  - Consumer will create the exchange
  - Each process runs in a single thread (thread per exchange)
  - After passing the exchange to the next process the Consumer will wait for a return message
  - A Producer will also wait for a response (Example: JMS will create a Temp queue, wait for the response, populate the out and return)



How Apache Camel works

- **Routing Builder**
  - wiring together processors and endpoints
  - Can use XML or JavaDSL or ScalaDSL

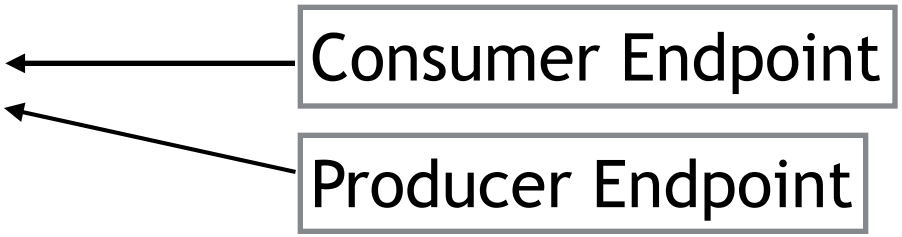


How Apache Camel works

Routing Builder

- **Example XML for defining a Route.**

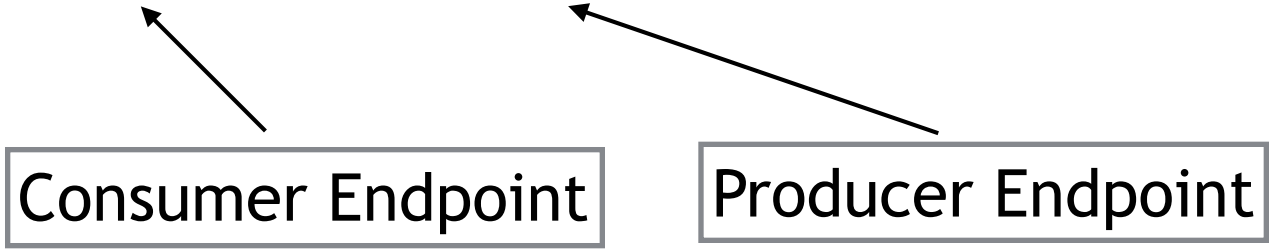
```
<camelContext id="camel1" xmlns="http://camel.apache.org/schema/spring">
   <route>
      <from uri="direct:one"/>
      <to uri="mock:result"/>
   </route>
</camelContext>
```



How Apache Camel works

- **Routing Builder**
  - **Example JavaDSL for defining a Route.**

```
from("direct:one").to("mock:result")
```



How Apache Camel works

- **Processors**
  - **Manipulate and mediate messages**
  - **EIPs are implemented as processors**

    from(“direct:one”).**split(xpath(“/orders/order”)**.to(“mock:result”)

  - **Custom processor**

```
public class MyProcessor implements Processor {
  public void process(Exchange exchange) throws Exception {
    // do something…
    Message in = exchange.getIn();
    in.setBody(“New body”);
  }
}
```

```
<bean id="myProcessor" class="com.acme.MyProcessor"/>

from(“direct:one").to("myProcessor");
```

Running Apache Camel (Testing)

Maven Plugin

- The easiest way to start and test a Camel route is to add the following to your <build> <plugins> section of your pom file.

```
<plugin>
  <groupId>org.apache.camel</groupId>
  <artifactId>camel-maven-plugin</artifactId>
  <configuration>
    <applicationContextUri>
                META-INF/spring/*.xml;<your file in classpath>.xml
    </applicationContextUri>
  </configuration>
</plugin>
```

- Start the route from Maven command line

```
mvn camel:run
```

LIVE CODING!

Questions?

Thank You!

Code for presentation:
https://github.com/jgenender/camel-rider