Red Hat

SYSTEM INTEGRATION AND APACHE CAMEL

VILIAM KASALA

vkasala@redhat.com

ME

VILIAM KASALA

- Quality Engineer at ***RedHat***
- Fuse product
  - Integration frameworks Camel and JMS
  - CXF Webservice framework -JAX-RS/JAX-WS
- vkasala@redhat.com

AGENDA

- System Integration
- Apache Camel
  - Brief Introduction
  - Camel Architecture
  - Camel Components
  - Data Transformation
  - Other Topics
  - Camel 3 & Future

AGENDA

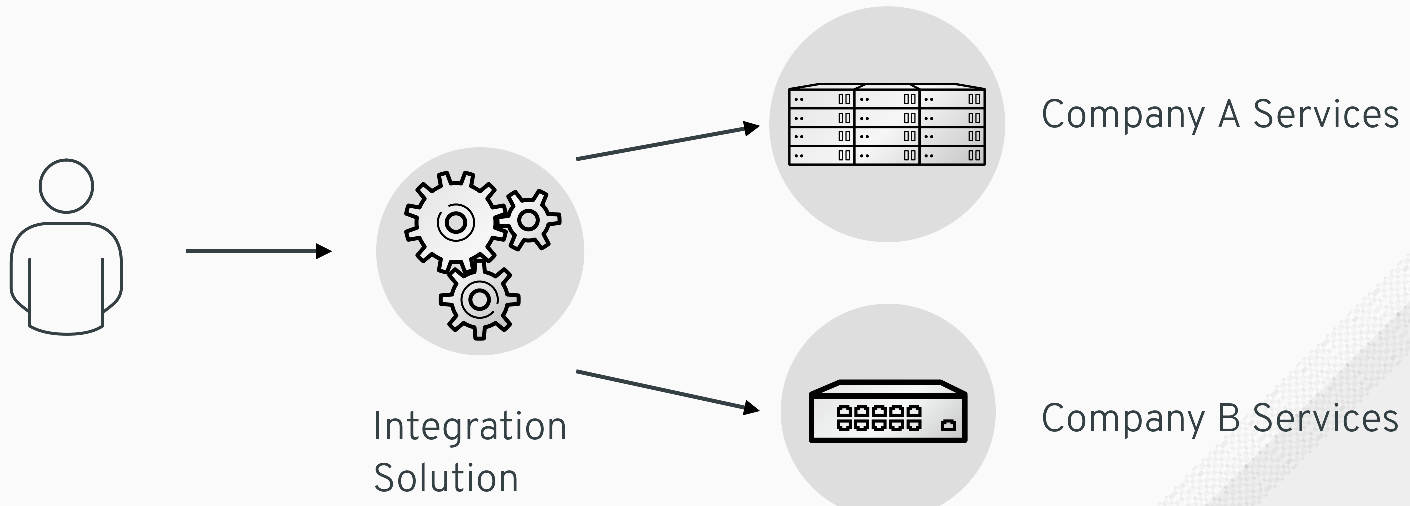
- **System Integration**
- Apache Camel
  - Brief Introduction
  - Camel Architecture
  - Camel Components
  - Data Transformation
  - Other Topics
  - Camel 3 & Future

SYSTEM INTEGRATION

The process of bringing together the component subsystems into one system and ensuring that the subsystems function together as a system.

WHY INTEGRATION ?

- Growth of an enterprise by:
  - acquisitions and fusions
- Different subsystems use different technologies or languages
- New values are created by combinations of existing products
- Data transformation
- Incremental legacy application replacements

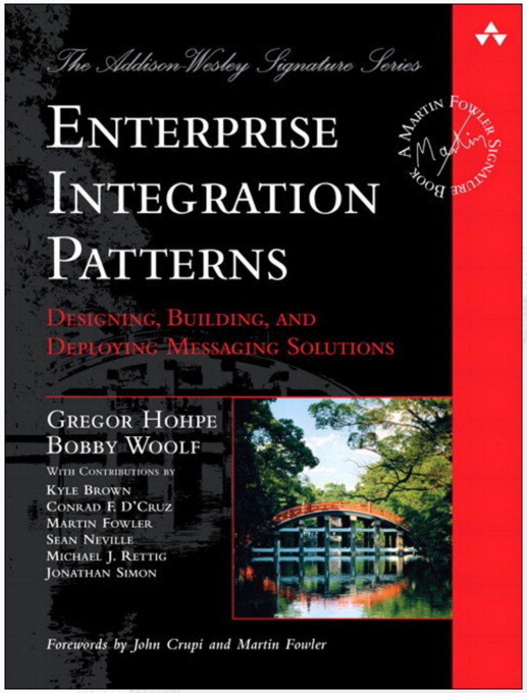


INTEGRATION STYLES

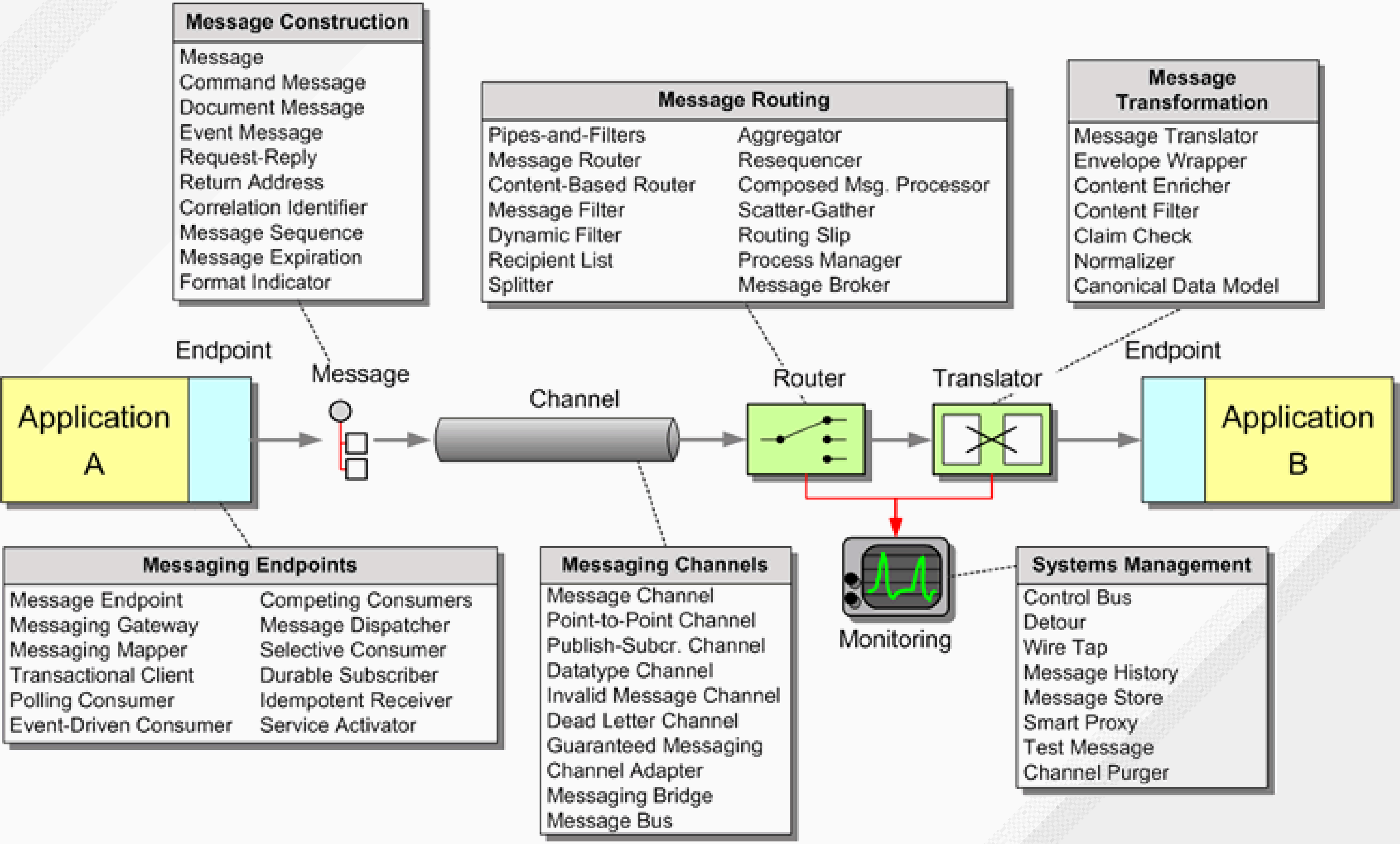
- File Transfer
  - each application produce files of shared data for others to consume, and consume files that others have produced
- Shared Database
  - the applications store the data they wish to share in a common database
- Remote Procedure Invocation
  - each application expose some of its procedures so that they can be invoked remotely, and have applications invoke those to run behavior and exchange data
- Messaging
  - each application connect to a common messaging system, and exchange data and invoke behavior using messages

ENTERPRISE INTEGRATION PATTERNS

- Proven design patterns and recipes for common integration problems
- Patterns were "Harvested" from a study of thousands of Integration projects.
- Used as the basic for all the major integration products
- Describes integration problems, solutions and also provide common vocabulary and diagram notations
- Message Centric
- A book by Gregor Hohpe and Bobby Woolf
- http:/www.eaipatterns.com


The Addison-Wesley Signature Series
Enterprise Integration Patterns
Designing, Building, and Deploying Messaging Solutions
A Martin Fowler Signature Book
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

ENTERPRISE INTEGRATION PATTERNS



Message Construction

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

Message Routing

Pipes-and-Filters	Aggregator
Message Router	Resequencer
Content-Based Router	Composed Msg. Processor
Message Filter	Scatter-Gather
Dynamic Filter	Routing Slip
Recipient List	Process Manager
Splitter	Message Broker

Message Transformation

- Message Translator
- Envelope Wrapper
- Content Enricher
- Content Filter
- Claim Check
- Normalizer
- Canonical Data Model

Messaging Endpoints

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

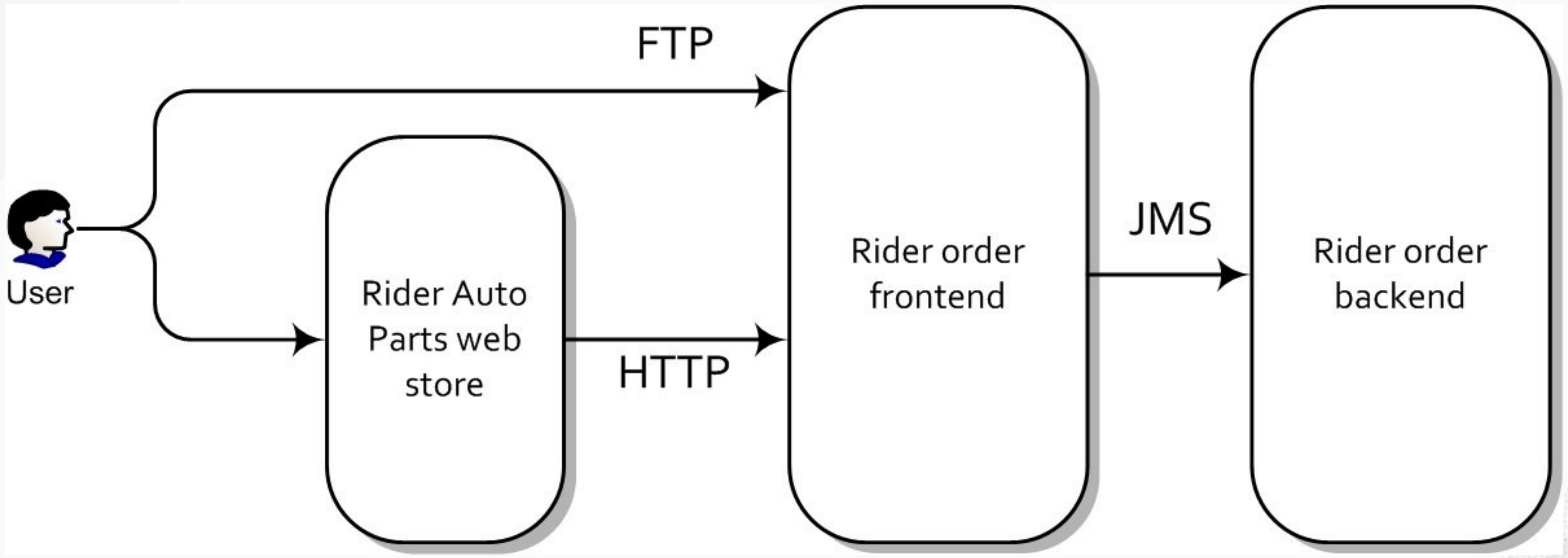
Messaging Channels

- Message Channel
- Point-to-Point Channel
- Publish-Subcr. Channel
- Datatype Channel
- Invalid Message Channel
- Dead Letter Channel
- Guaranteed Messaging
- Channel Adapter
- Messaging Bridge
- Message Bus

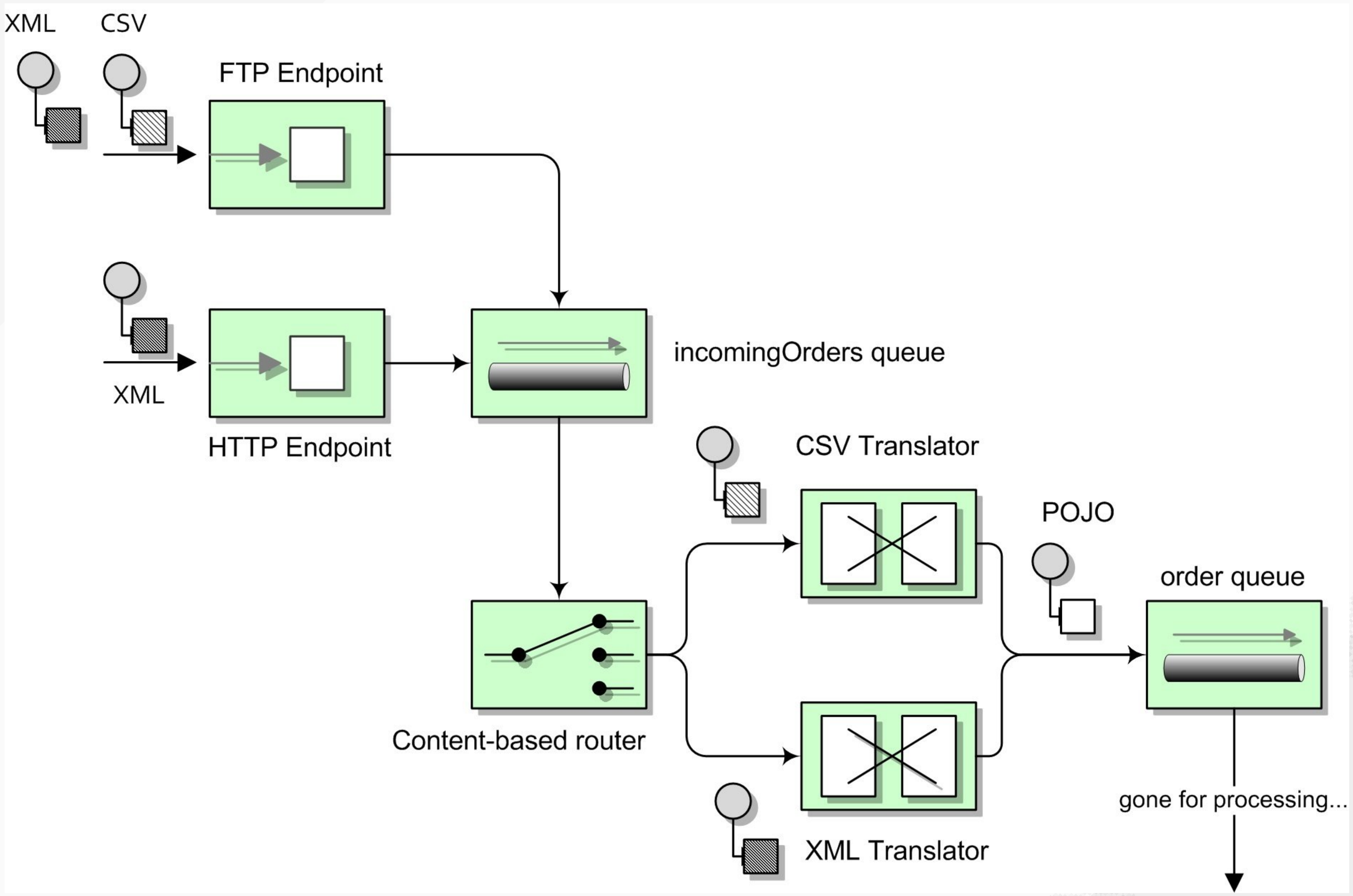
Systems Management

- Control Bus
- Detour
- Wire Tap
- Message History
- Message Store
- Smart Proxy
- Test Message
- Channel Purger

RIDERS AUTO PART EXAMPLE



RIDERS AUTO PART EXAMPLE EIP

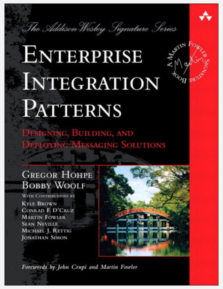


WHY INTEGRATION FRAMEWORK ?

- Don't reinvent the wheel
- It makes your life easier
- As a developer don't have to think about low level code
- Implements common Enterprise Integration Patterns

Apache Camel vs. spring vs. muleESB™

IMPLEMENTS


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

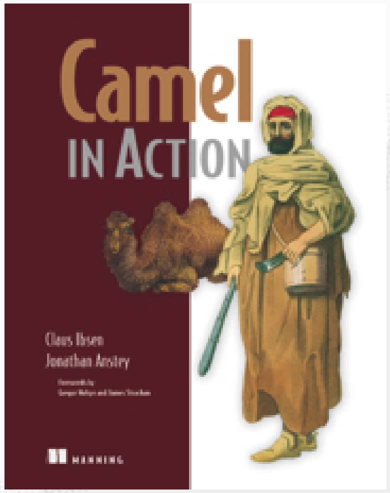
Forewords by John Crupi and Martin Fowler

AGENDA

- System Integration
- **Apache Camel**
  - **Brief Introduction**
  - Camel Architecture
  - Camel Components
  - Data Transformation
  - Other Topics
  - Camel 3 & Future

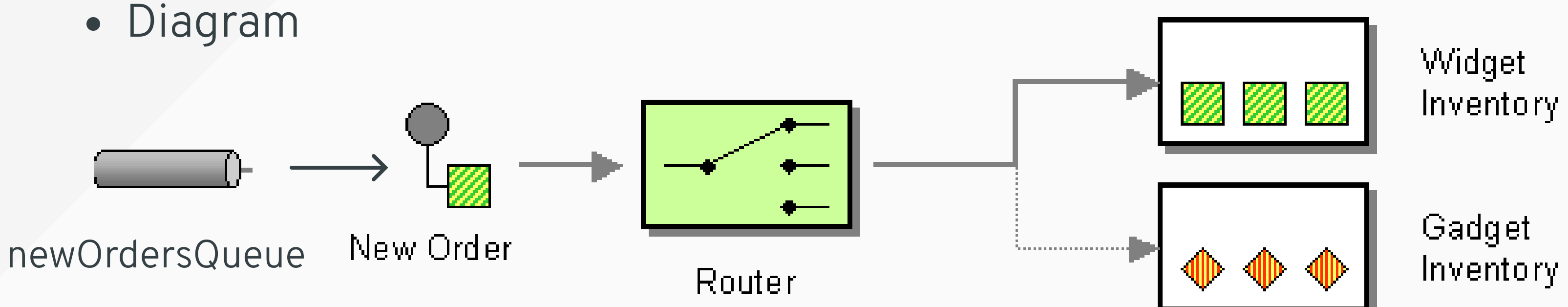
WHAT IS APACHE CAMEL 2?

- open source Java framework that focuses on making integration easier and more accessible to developers.
- Provides:
  - concrete implementations of all the widely used EIPs
  - connectivity to a great variety of transports and APIs - Extensive Component Library
  - easy to use Domain Specific Languages (DSLs) to wire EIPs and transports together
  - routing engine for moving of messages based on routes
  - payload-agnostic router - any kind of payload XML, JSON, Binary
  - POJO as a first-class citizens
  - modular and pluggable architecture
  - lightweight core ideal for microservices
  - easy configuration of Endpoints through URIs
  - automatic Type Converters
  - test kit
  - cloud ready



CAMEL IN PRACTICE

- Use Case
  - Receive orders from ActiveMQ queue and based on the type of message forward to appropriate queue (ActiveMQ widget or Websphere MQ gadget)
- Diagram

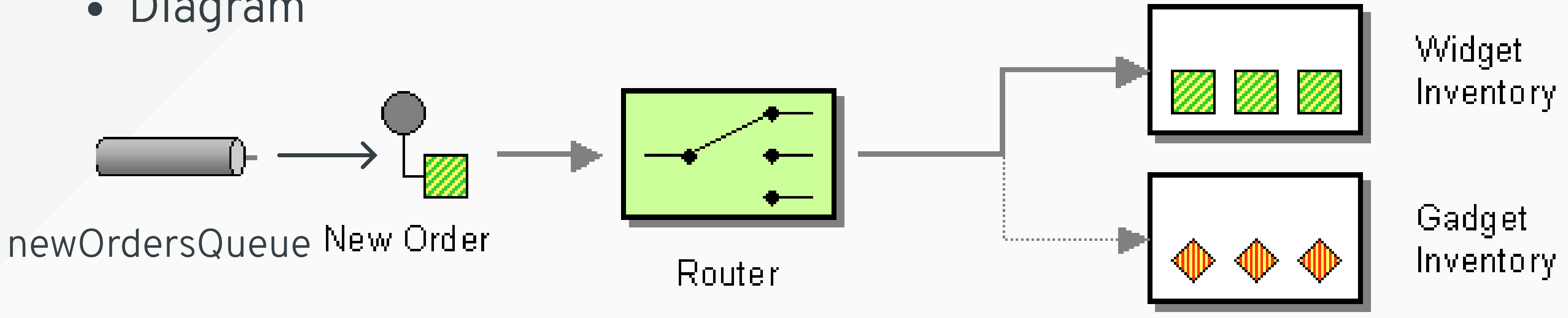


- DSL

  -
    ```
    from newOrderQueue
        choice
            when isWidget to widgetQueue
            otherwise to gadgetQueue
    ```

CAMEL IN PRACTICE

- Use Case
  - Receive orders from ActiveMQ queue and based on the type of message forward to appropriate queue (ActiveMQ widget or Websphere MQ gadget)
- Diagram



- DSL

  -
    ```
    from(newOrdersQueue)
        .choice()
        .when(isWidget).to(widget)
        .otherwise().to(gadget)
    ```

CAMEL IN PRACTICE - JAVA DSL

```java
import org.apache.camel.CamelContext;
import org.apache.camel.impl.DefaultCamelContext;
import org.apache.camel.builder.RouteBuilder;

public class BasicIntegrationExample {

    public static void main(String args[]) throws Exception {
        CamelContext context = new DefaultCamelContext();
        context.addRoutes(new RouteBuilder() {
            public void configure() throws Exception {
                from("jms:newOrdersQueue")
                    .choice()
                        .when(xpath("/order/product = 'widget'"))
                            .log("Widget")
                            .to("jms:widgetQueue")
                        .otherwise()
                            .log("Gadget")
                            .to("jms:gadgetQueue");
            }
        );
        context.start();
        Thread.sleep(10000); // Sleep main thread
        context.stop();
}
```

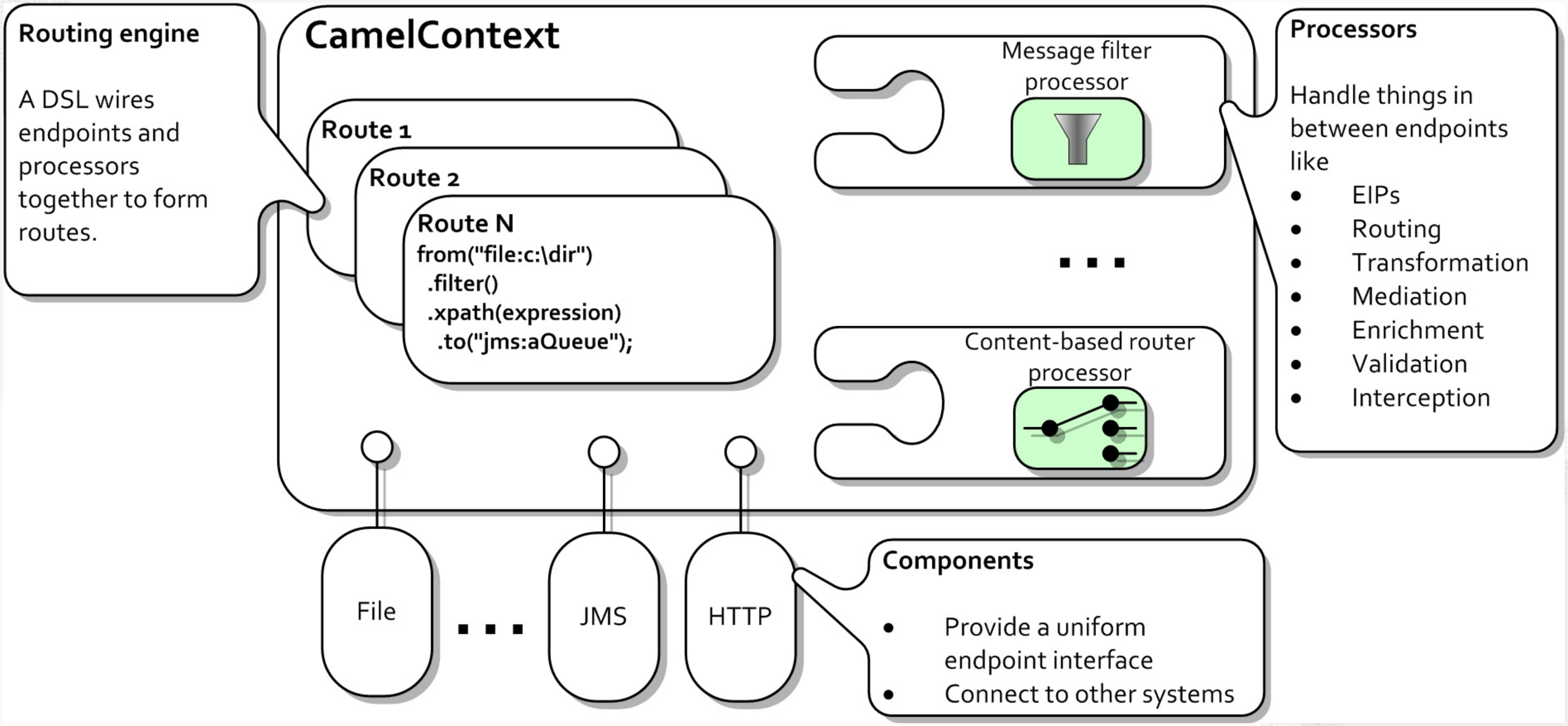
CAMEL SPRING XML DSL

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="...">
  <camelContext xmlns="http://camel.apache.org/schema/spring">
    <route>
      <from uri="jms:newOrdersQueue"/>
      <choice>
        <when>
          <xpath>/order/product = 'widget'</xpath>
          <log message="Widget message"/>
          <to uri="jms:widgetQueue"/>
        </when>
        <otherwise>
          <log message="Gadget message"/>
          <to uri="jms:gadgetQueue"/>
        </otherwise>
      </choice>
    </route>
  </camelContext>
</beans>
```

AGENDA

- System Integration
- **Apache Camel**
  - Brief Introduction
  - **Camel Architecture**
  - Camel Components
  - Data Transformation
  - Other Topics
  - Camel 3 & Future

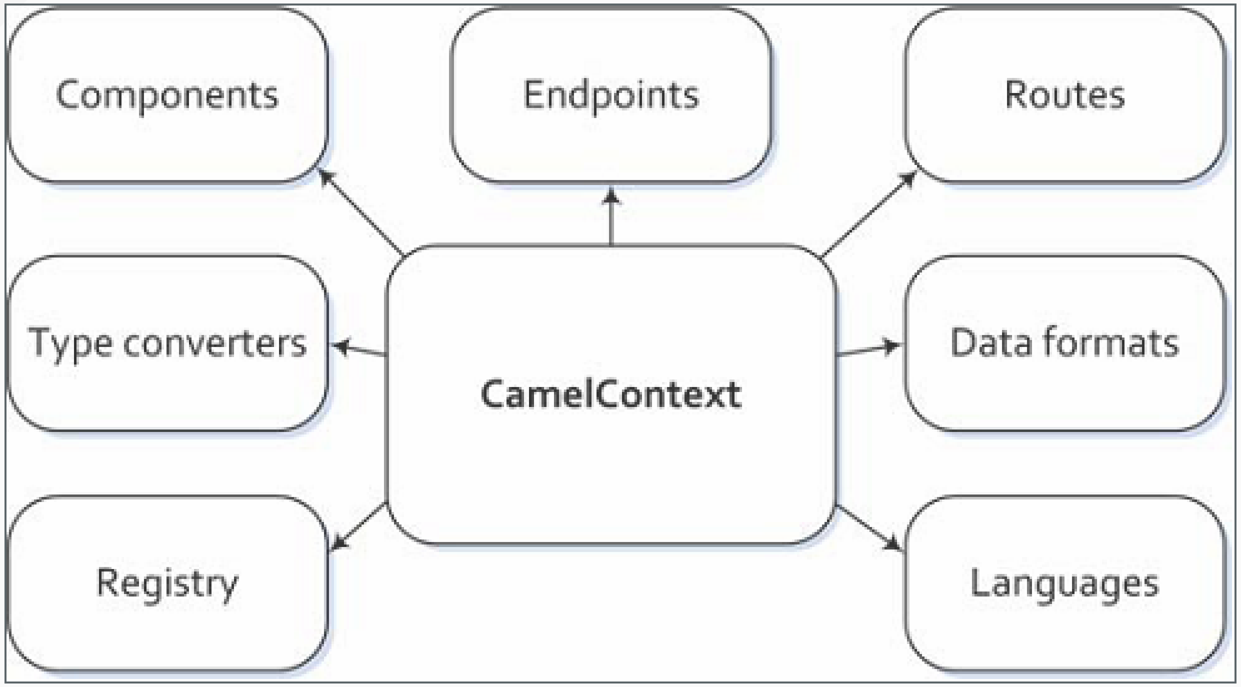
CAMEL ARCHITECTURE


Routing engine
A DSL wires endpoints and processors together to form routes.
CamelContext
Route 1
Route 2
Route N
Message filter processor
...
Content-based router processor
Processors
Handle things in between endpoints like
• EIPs
• Routing
• Transformation
• Mediation
• Enrichment
• Validation
• Interception
File
...
JMS
HTTP
Components
• Provide a uniform endpoint interface
• Connect to other systems

```
from("file:c:\dir")
 .filter()
 .xpath(expression)
  .to("jms:aQueue");
```

CAMEL CONTEXT

- Container of many Camel services, which keeps all the pieces together

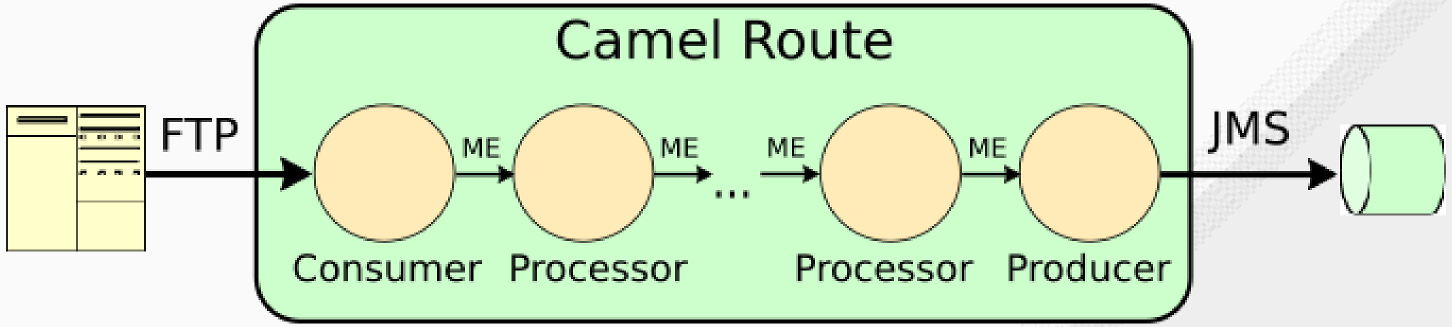


CAMEL ROUTE

- Core abstraction, which is defined in Java DSL, XML, Scala DSL, Groovy
- Chain of processors, components:
  - From a **Consumer** - listening endpoint
  - Through a zero or more processing components - e.g. EIP, processors
  - To a **Producer** - target endpoint
- Each route has a unique ID for logging, debugging, monitoring and managing purposes
- Example:

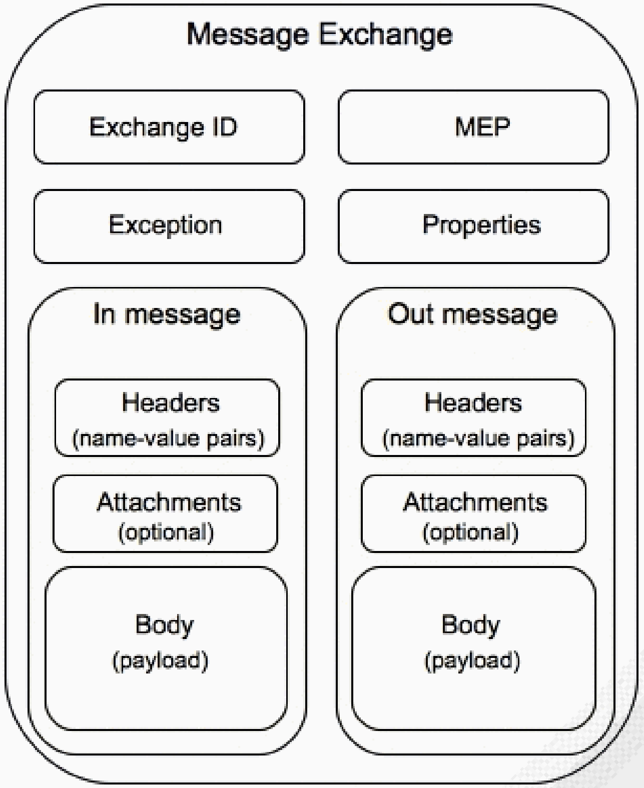
```
from("ftp://foo@myserver?password=secret&ftpClientConfig=#myConfig")
     .to("jms:queue:foo");
```

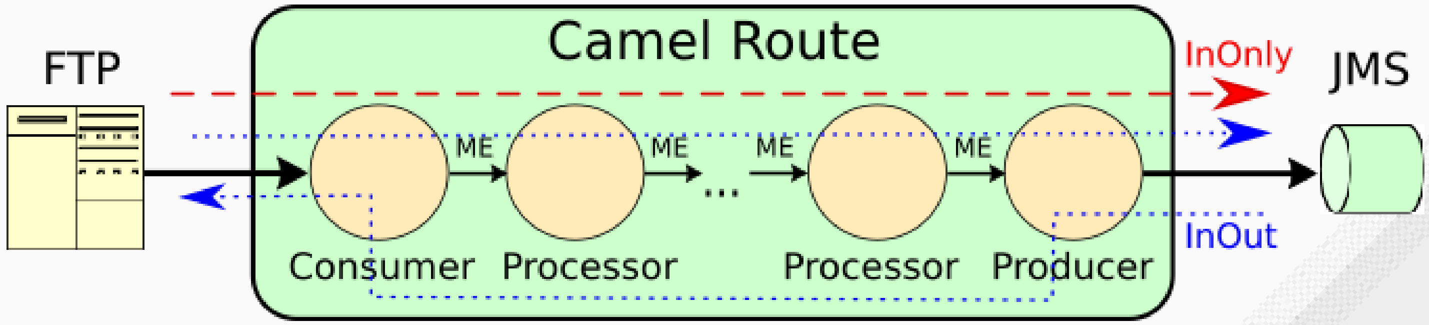
- Diagram:



CAMEL MESSAGE MODEL

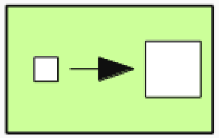
- **Message**
  - basic structure for moving data over a route
  - first created by consumer
- **Message Exchange - ME**
  - message container during routing
  - link between producer and consumer
  - two options for Message Exchange Pattern - MEP:
    - *InOnly* (fire & forget - e.g. JMS message)
    - *InOut* (request-response - e.g. HTTP request)

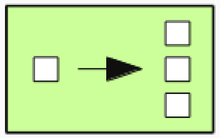


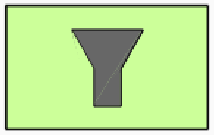


CAMEL PROCESSOR

- Perform actions on the message - modify, use, create, enrich, transform, validate, intercept, etc.
- Implements the actions of the EIP between the producer/consumer endpoint
- Processors can be linked in pipeline flow
- Processor examples:


Content Enricher


Splitter

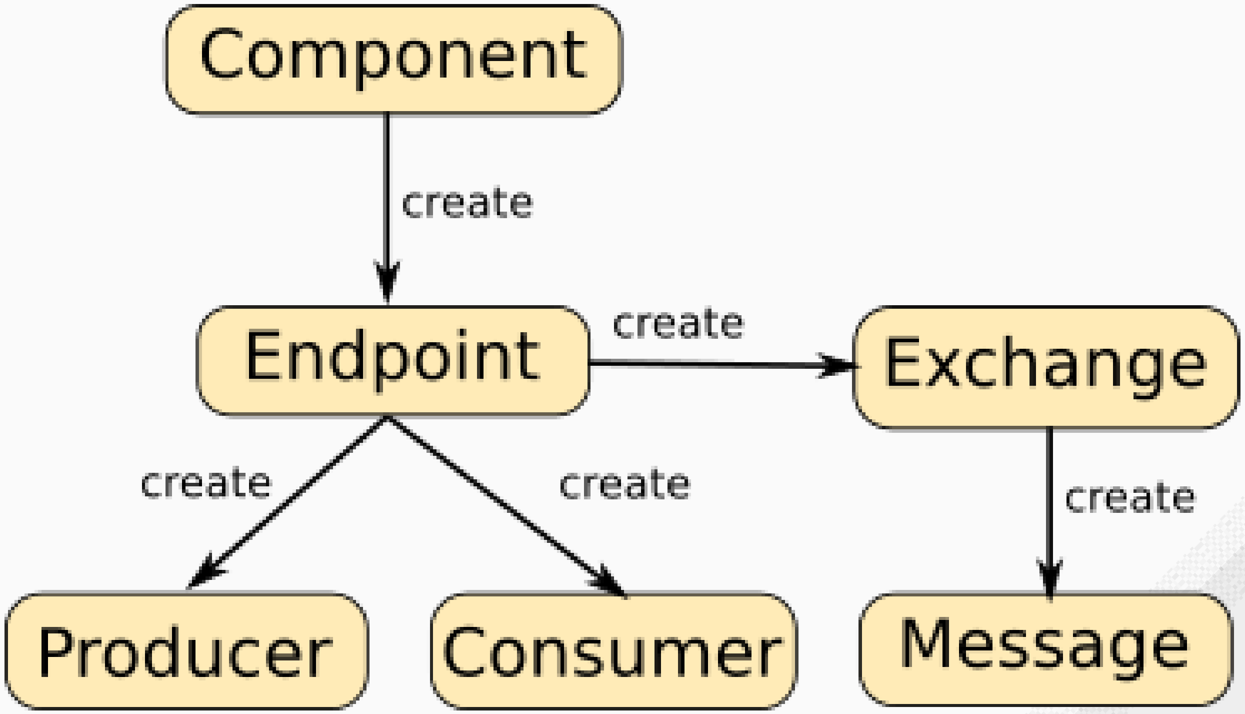

Message Filter

- Custom processor example:

```
from("activemq:myQueue").process(
    new Processor() {
        void process(Exchange exchange) throws Exception {
            exchange.getIn().setBody("Changed body");
        }
    }
).to("file://inbox/orders");
```

CAMEL COMPONENT

- Main extension point in Camel
- Contains configurations for Endpoints
- Factory for *Endpoint* instances
- Component Model:



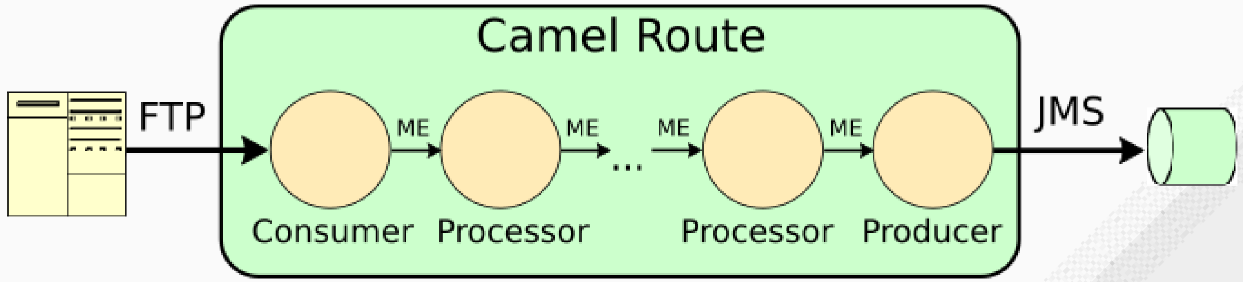
CAMEL ENDPOINT

- Represents endpoint which is capable of sending and receiving (producing and consuming) messages e.g. FTP server, a Web Service or a JMS broker
- Described by URIs:
  - **schema:context/path?options**
    - schema = identifies component
    - context/path = identifies location of a resource or destination
    - options = setup of properties for component, list of name/value pairs
  - examples:
    - file:inbox/orders?delete=true
    - ftp://john@localhost/ftp?password=nhoj
    - activemq:queue:MyQueue
    - timer://myTimer?period=2000

CAMEL ENDPOINT ROLE

- **Consumer** (from)
  - receives messages from an external source and creates a message exchange object, which the routing rule processes
  - *event-driven consumer* - waits until message arrives e.g. JMS, HTTP, tcp, udp
  - *polling consumer* - actively checks for new messages e.g. FTP, file, email
- **Producer** (to)
  - sends the current message wrapped in the message exchange object to an external target destination

```
from("ftp://john@localhost/ftp?password=nhoj")
    .to("xslt:MyTransform.xslt").to("activemq:queue:MyQueue")
```



EXPRESSIONS AND PREDICATES

- Camel supports different 15+ different expression languages
  - EL, Simple, XQuery, Xpath, JavaScript, Ruby, Python, PHP, etc
- Expression
  - returns the value of the expression on the given exchange
- Predicate
  - evaluates the predicate on the message exchange and returns true if this exchange matches the predicate

```
<route>
  <from uri="direct:start"/>
  <transform>
    <simple>${in.body} extra data!</simple>
  </transform>
  <to uri="mock:end"/>
</route>
```

Expression example

```
<from uri="direct:orders">
   <filter>
       <simple>${in.header.foo}</simple>
       <to uri="file:fooOrders"/>
   </filter>
</from>
```

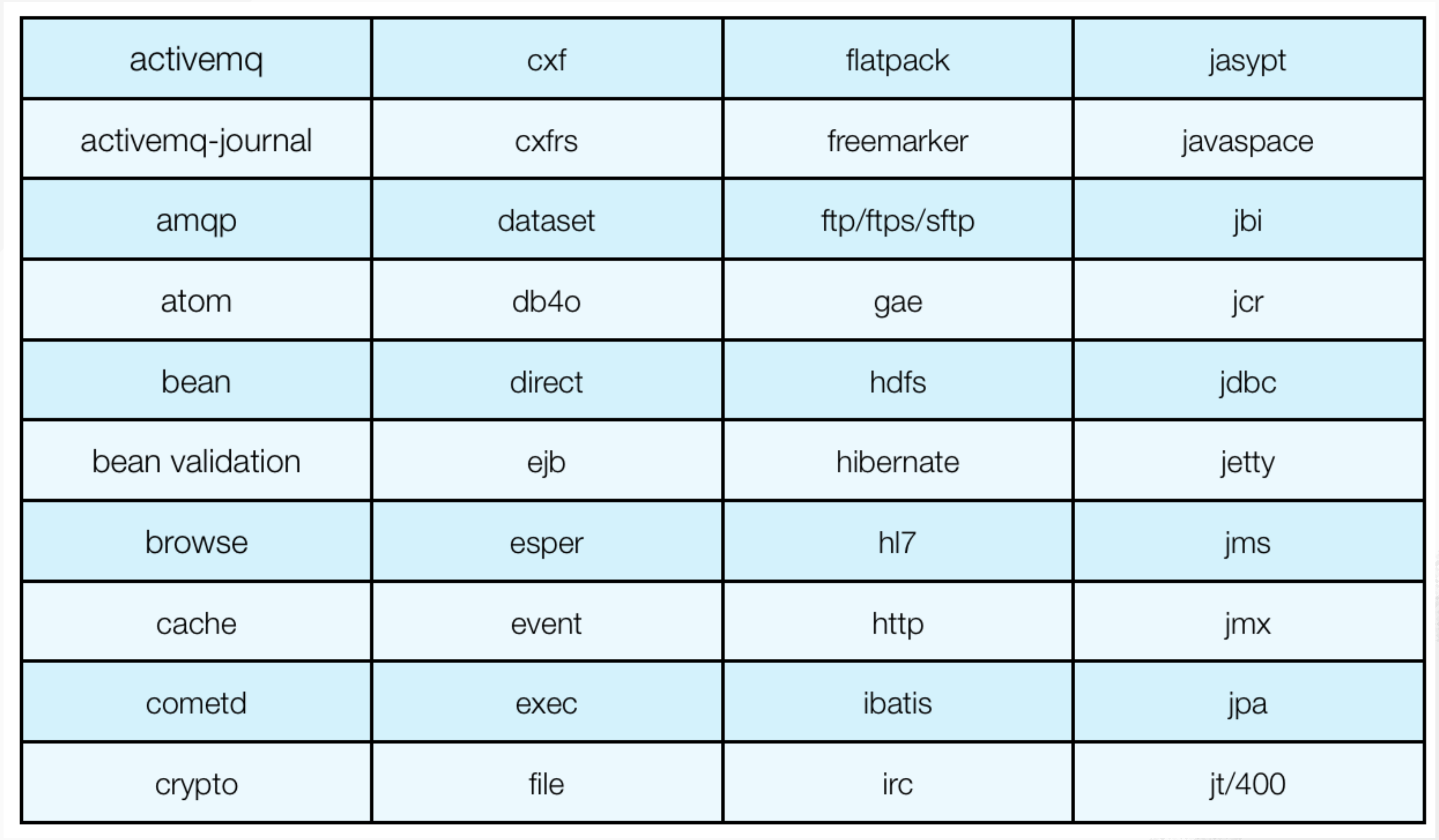
Predicate example

AGENDA

- System Integration
- **Apache Camel**
  - Brief Introduction
  - Camel Architecture
  - **Camel Components**
  - Data Transformation
  - Other Topics
  - Camel 3 & Future

CAMEL COMPONENTS

- 280 Components



activemq	cxf	flatpack	jasypt
activemq-journal	cxfrs	freemarker	javaspace
amqp	dataset	ftp/ftps/sftp	jbi
atom	db4o	gae	jcr
bean	direct	hdfs	jdbc
bean validation	ejb	hibernate	jetty
browse	esper	hl7	jms
cache	event	http	jmx
cometd	exec	ibatis	jpa
crypto	file	irc	jt/400

CAMEL COMPONENTS

language	properties	seda	stream
ldap	quartz	servlet	string-template
mail/imap/pop3	quickfix	sip	test
mina	ref	smooks	timer
mock	restlet	smpp	validation
msv	rmi	snmp	velocity
nagios	rnc	spring-integration	vm
netty	rng	spring-security	xmpp
nmr	rss	spring-ws	xquery
printer	scalate	sql	xslt

BEAN COMPONENT

- Allows to use methods of Java object for processing of a message
- URI pattern:
  - **bean:beanId?options**
- Useful options:
  - method - the name of the method of Java Class

```java
public class MyBean {
   public void process(String msg) {
      System.out.println("Message: " + msg);
   }
}
```

Java Bean

```java
from("file:src/data?noop=true")
   .bean(new MyBean(), "process");
```

Java DSL

FILE COMPONENT

- Allows access to file system
- URI pattern:
  - file:directoryName[?options]
- Useful options:
  - noop - if true files are nor moved not deleted after the procession finishes
  - include - regular expression of file names to be processed
  - exclude - regular expression

```
from("file:src/data?noop=true")
    .bean(new MyBean(), "process");
```

Java DSL

JETTY COMPONENT

- Provides endpoints based on HTTP transport protocol
- URI pattern:
  - jetty:http://hostname[:port][/resourceUri][?options]
- Useful options:
  - httpMethodRestrict - list of allow HTTP method e.g. GET, POST, PUT

```
from("jetty:http://localhost:8123/path")
    .to("file:target/messages");
```

Java DSL

RESTLET COMPONENT

- Provides endpoints based on REST HTTP
- URI pattern:
  - restlet:protocol://hostname[:port][/resourcePattern][?options]
- Useful options:
  - restletMethods - list of allow HTTP method e.g. GET, POST, PUT

Java DSL

```
from("restlet:http://localhost:8080/rest-end?restletMethods=POST")
    .log("POST: ${body}");
```

REST DSL

```
restConfiguration().component("restlet").host("localhost").port(8080);

rest("/rest-end").consumes("text/plain").produces("text/plain").post()
    .to("direct:post");

from("direct:post").log("POST: ${body}");
```

ACTIVEMQ-JMS COMPONENT

- Provides endpoint for communication with destinations (queues and topics) with various protocols
  - MQTT - ActiveMQ-MQTT component
  - JMS - ActiveMQ-JMS component
- URI pattern:
  - **activemq:[queue:|topic:]destinationName[?options]**
  - **jms:[queue:|topic:]destinationName[?options]**
  - **mqtt:name[?options]**
- Useful options:
  - replyTo - provides explicit ReplyTo destination

DATABASE COMPONENTS

- Camel provides multiple components to integrate with DBs:
  - JDBC component
  - SQL component
  - JPA/Hibernate component
  - iBatis component

```
<!-- route that generate new orders and insert them in the database -->
<route id="generateOrder-route">
    <from uri="direct:generateOrder"/>
    <to uri="hibernate:org.camel.examples.hibernate.Order"/>
    <log message="Inserted new order ${body.id}"/>
</route>
```

AGENDA

- System Integration
- **Apache Camel**
  - Brief Introduction
  - Camel Architecture
  - Camel Components
  - **Data Transformation**
  - Other Topics
  - Camel 3 & Future

DATA TRANSFORMATION OVERVIEW

- Data format transformation
  - the data format of message body is transformed from e.g.
  - *CSV to formatted XML*
- Data type transformation
  - the data type of the message body is transformed
  - *java.lang.String -> javax.jms.TextMessage*
  - automatic type converter mechanism

DATA TRANSFORMATION

1. In routes
   - *Processor*, *Beans*, *<transform>*
2. Using components
   - e.g. XSLT component for XML transformation
3. Using data formats
   - transform data back and forth between well-know formats
   - e.g. CSV, JAXB, Jackson, Zip
4. Using templates
   - components for transforming using templates - e.g. *Apache Velocity*, *FreeMarker*
5. Using camel's type converter mechanism

DATA FORMAT EXAMPLE

- unmarshalling/deserialization

```
JaxbDataFormat jaxb = new JaxbDataFormat("com.redhat.brq.integration");
from("file:src/data?noop=true")
    .unmarshal(jaxb)
    .bean(new MyBean(), "process");
```

- marshalling/serialization

```
JaxbDataFormat jaxb = new JaxbDataFormat("com.redhat.brq.integration");
from("direct:javaObject")
    .marshal(jaxb)
    .to("file:src/xmlData")
```

AGENDA

- System Integration
- **Apache Camel**
  - Brief Introduction
  - Camel Architecture
  - Camel Components
  - Data Transformation
  - **Other Topics**
  - Camel 3 & Future

TESTING CAMEL APPLICATIONS

- CamelTestSupport - JUnit framework extension
- PaxExam

ERROR HANDLING

- Camel provides *Exception Clause* to specify error handling per exception type
- Two scopes:
  - global level
  - route specific level

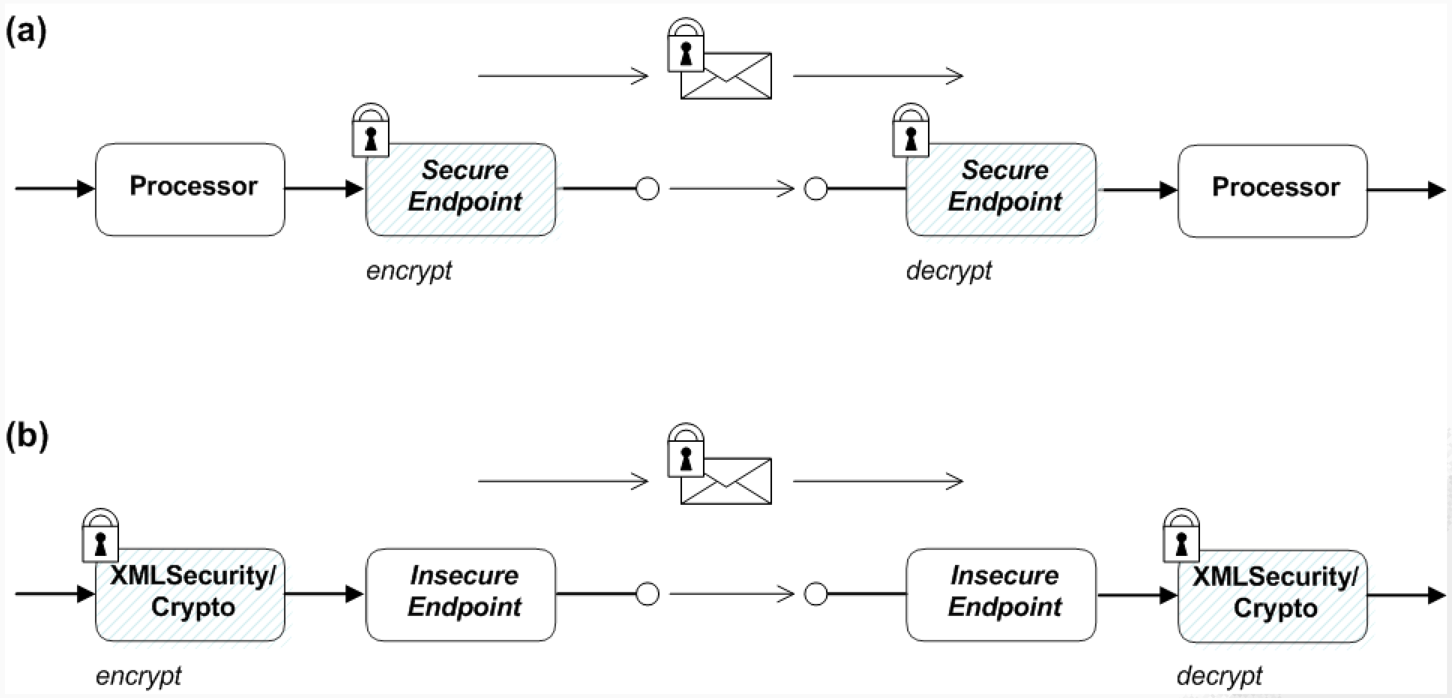
```
onException(ValidationException.class).to("activemq:validationFailed");
onException(ShipOrderException.class).to("activemq:shipFailed");
```

SECURITY #1

- Route Security
  - Authentication and Authorization services to proceed on a route or route segment
  - Apache Shiro or Spring Security
- Configuration Security
  - Camel allows to crypt/decrypt configuration files containing sensitive information

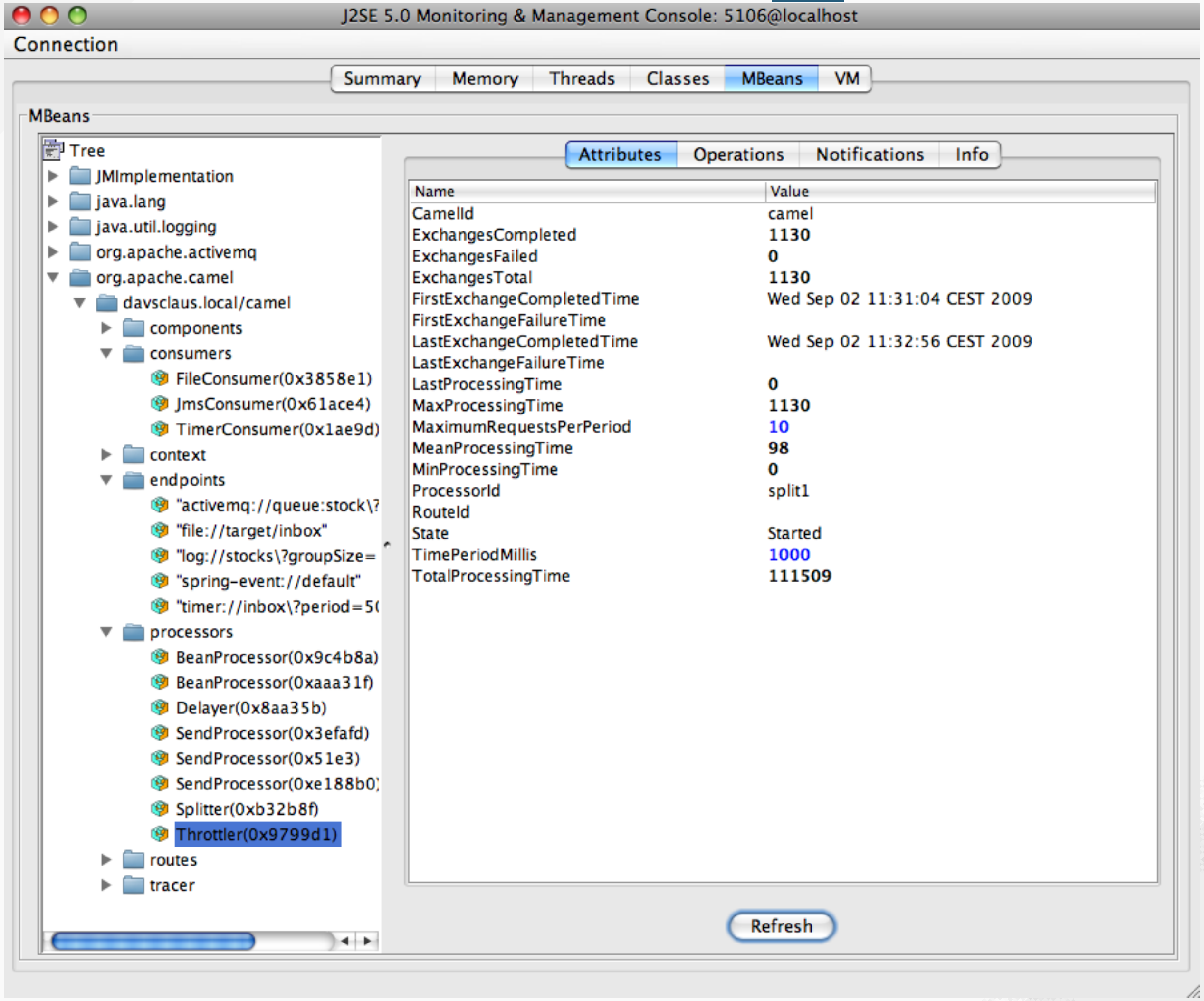
SECURITY #2

- Endpoint Security
  - Security offered by components through URI associated with the component
- Payload Security
  - Data Formats that offer encryption/decryption services at the payload level



MANAGING CAMEL

- At JVM level Camel exposes its managed beans through JMX



Name	Value
CamelId	camel
ExchangesCompleted	1130
ExchangesFailed	0
ExchangesTotal	1130
FirstExchangeCompletedTime	Wed Sep 02 11:31:04 CEST 2009
FirstExchangeFailureTime	
LastExchangeCompletedTime	Wed Sep 02 11:32:56 CEST 2009
LastExchangeFailureTime	
LastProcessingTime	0
MaxProcessingTime	1130
MaximumRequestsPerPeriod	10
MeanProcessingTime	98
MinProcessingTime	0
ProcessorId	split1
RouteId	
State	Started
TimePeriodMillis	1000
TotalProcessingTime	111509

LOGGING

- Log component
  - logs message content

```
from("direct:start").to("log:cz.company.order?level=DEBUG").to("bean:foo");
```

- Tracer
  - Trace log message flows

```
<camelContext trace="true" xmlns="...camel/schema/spring">...</camelContext>
```

- Log DSL

```
from("direct:start").log("Processing ${id}").to("bean:foo");
```

- From Java code using *Bean* or *Processor*

DEPLOYING CAMEL

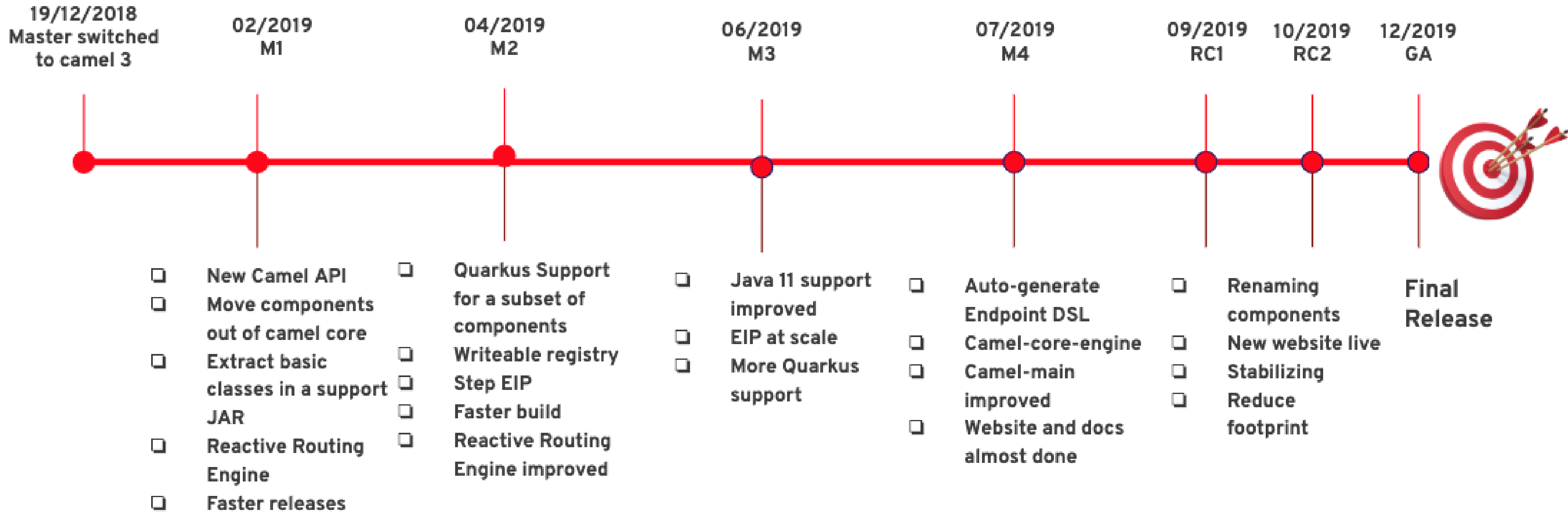
- Standalone JAR
- WAR - Servlet Container, e.g. *Apache Tomcat*, *Jetty*
- Spring - *Spring Boot*
- Java EE - *e.g. Wildfly, Glassfish, WebLogic, WebSphere*
- OSGi Container - *e.g. Apache Karaf, ServiceMix*
- Cloud - *e.g. Google Compute Engine, Amazon EC2*

AGENDA

- System Integration
- **Apache Camel**
  - Brief Introduction
  - Camel Architecture
  - Camel Components
  - Data Transformation
  - Other Topics
  - **Camel 3 & Future**

Camel 3 - Timeline

5500+ commits (Camel 3.x)



19/12/2018
Master switched to camel 3

02/2019
M1

- New Camel API
- Move components out of camel core
- Extract basic classes in a support JAR
- Reactive Routing Engine
- Faster releases

04/2019
M2

- Quarkus Support for a subset of components
- Writeable registry
- Step EIP
- Faster build
- Reactive Routing Engine improved

06/2019
M3

- Java 11 support improved
- EIP at scale
- More Quarkus support

07/2019
M4

- Auto-generate Endpoint DSL
- Camel-core-engine
- Camel-main improved
- Website and docs almost done

09/2019
RC1

- Renaming components
- New website live
- Stabilizing
- Reduce footprint

10/2019
RC2

12/2019
GA

Final Release

Apache Camel 3 - Projects

Camel

Integration Framework

"Swizz army of integration"

Camel K

Lightweight Integration Framework. Camel on Kubernetes & Knative.

Camel Quarkus

Camel extensions for Quarkus. Native compiled Java (GraalVM).

Resources

- [1] Enterprise Integration Patterns [book]
  - http://www.enterpriseintegrationpatterns.com/
- [2] Camel in Action
  - https://www.manning.com/books/camel-in-action
- [3] JBoss Fuse - Security Guide
  - https://access.redhat.com/documentation/en-US/Fuse_ESB_Enterprise/7.1/html/Security_Guide
- Fuse Training [RedHat Slides]
  - http://www.slideshare.net/adriangigante9/red-hat-open-day-jboss-fuse
- Riders Auto Part Use Case
  - https://dzone.com/articles/open-source-integration-apache
- Camel Simple Examples
  - https://github.com/vkasala/course-sys-int-camel-lecture
- Camel 3 Project
  - https://camel.apache.org/blog/Camel3-2monthsaway/

Red Hat

THANK YOU!