Spring Integration & Apache Camel

Spring Integration in Action

Grégory Boissinot
@gboissinot

Ecosystème Spring

Spring Integration: une API pour les EIP

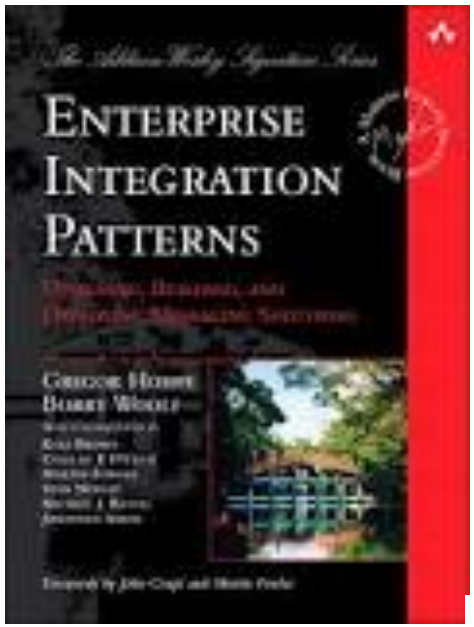
Mark Fisher

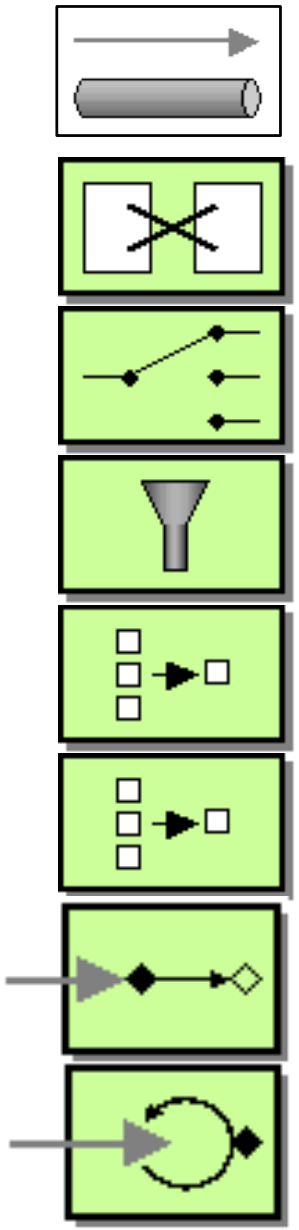
Read

Write

Implements

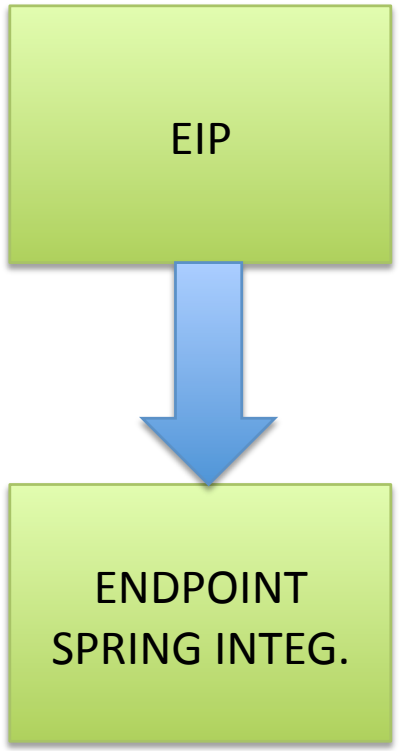
Integration



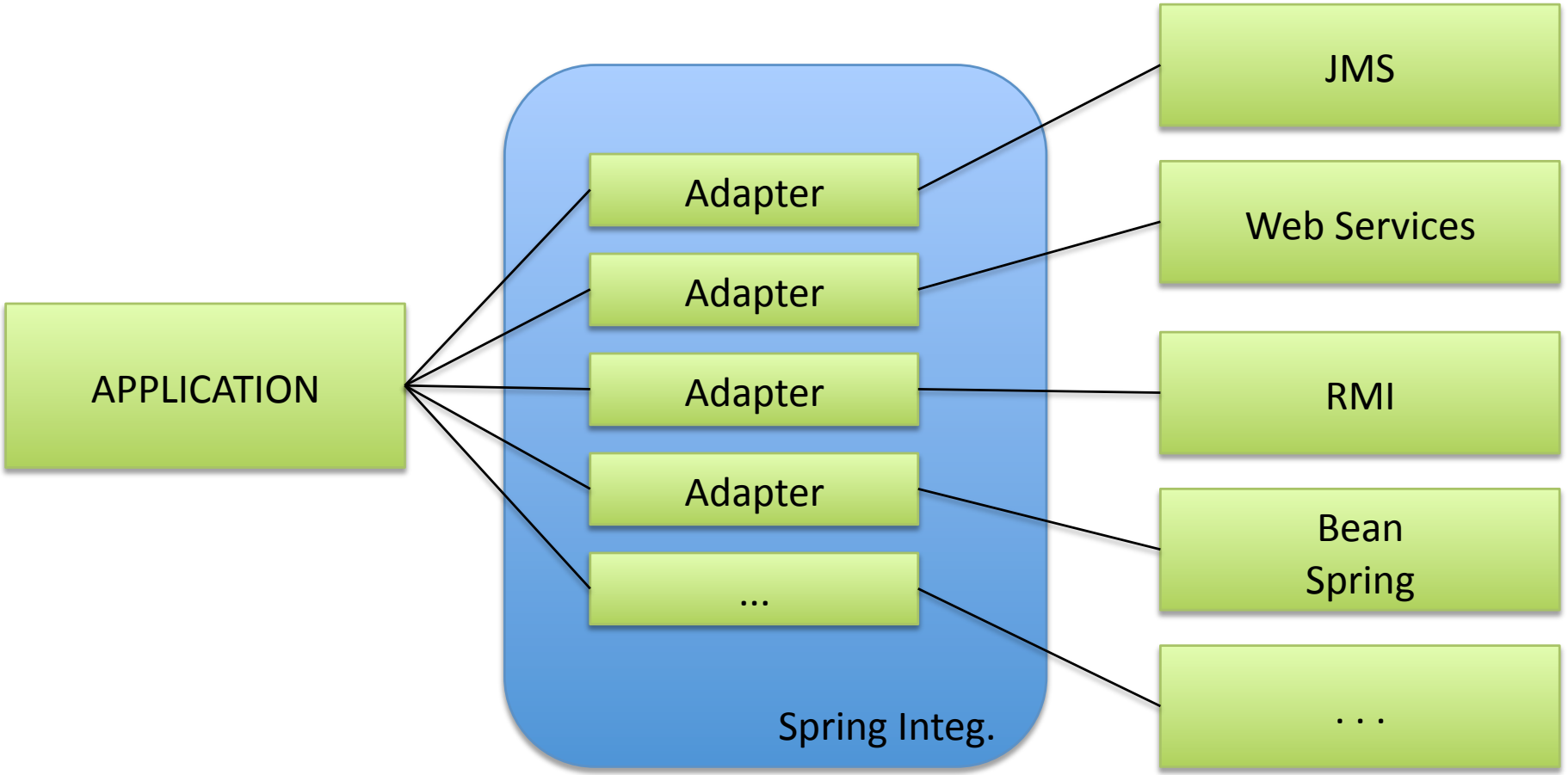


Les Endpoints Spring Integration

- Spring Integration fournit le support des principaux EIP à travers un modèle 1-1
  - Pas de concepts supplémentaires à apprendre



Une plateforme d'intégration à base d'adaptateurs



http://www.springsource.org/extensions/se-sia

Les principaux adaptateurs sur étagère

Un modèle déclaratif à travers un namespace

```
<!-- Fichier Spring Application Context -->
...
xmlns:int="http://www.springframework.org/schema/
integration"
xsi:schemaLocation="http://www.springframework.org/
schema/integration http://www.springframework.org/
schema/integration/spring-integration-2.1.xsd"
...

<int:channel id="inputChannel"/>
```

Chaque composant est un bean Spring

```
<int:channel id="inputChannel'/>
```

```
inputChannel =
  cxt.getBean("inputChannel", MessageChannel.class);
```

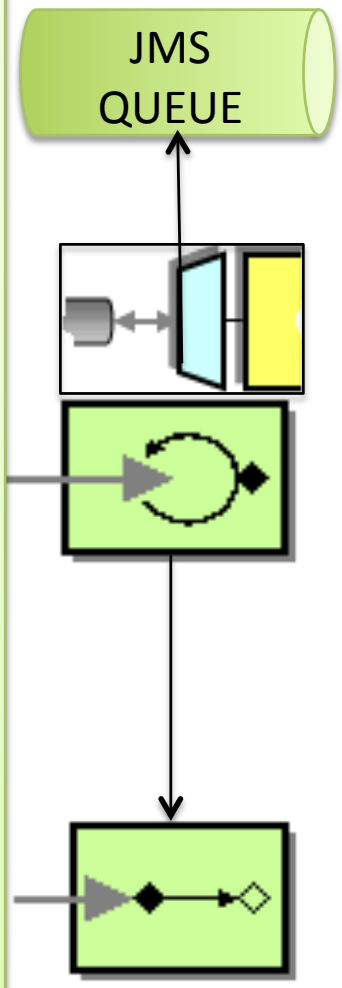
Réutilisation du modèle Spring

```
<int-jms:inbound-channel-adapter
      channel="inputChannel"
      destination="${input.queue}">
       <int:poller task-executor="pool"  fixed-delay="1000">
            <int:transactional />
        </int:poller>
</int-file:ibound-channel-adapteur>

<!– Pool de threads -->
 <task:executor id="pool" pool-size="10"/>

<!– Pollable Channel -->
 <int:channel id="inputChannel" />

<!– Call Spring Java Bean -->
<int:service-activator
            input-channel="inputChannel"
            ref="myServiceBean" />
```

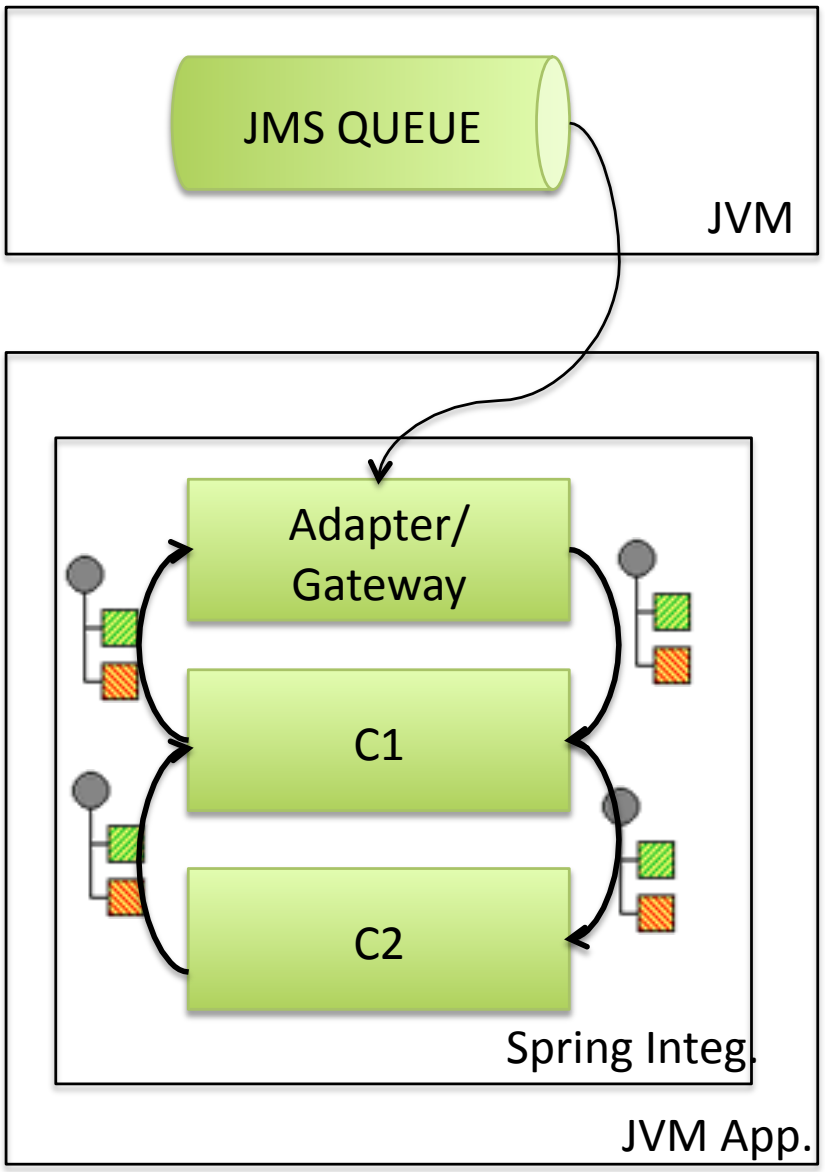
Spring Dependency Injection

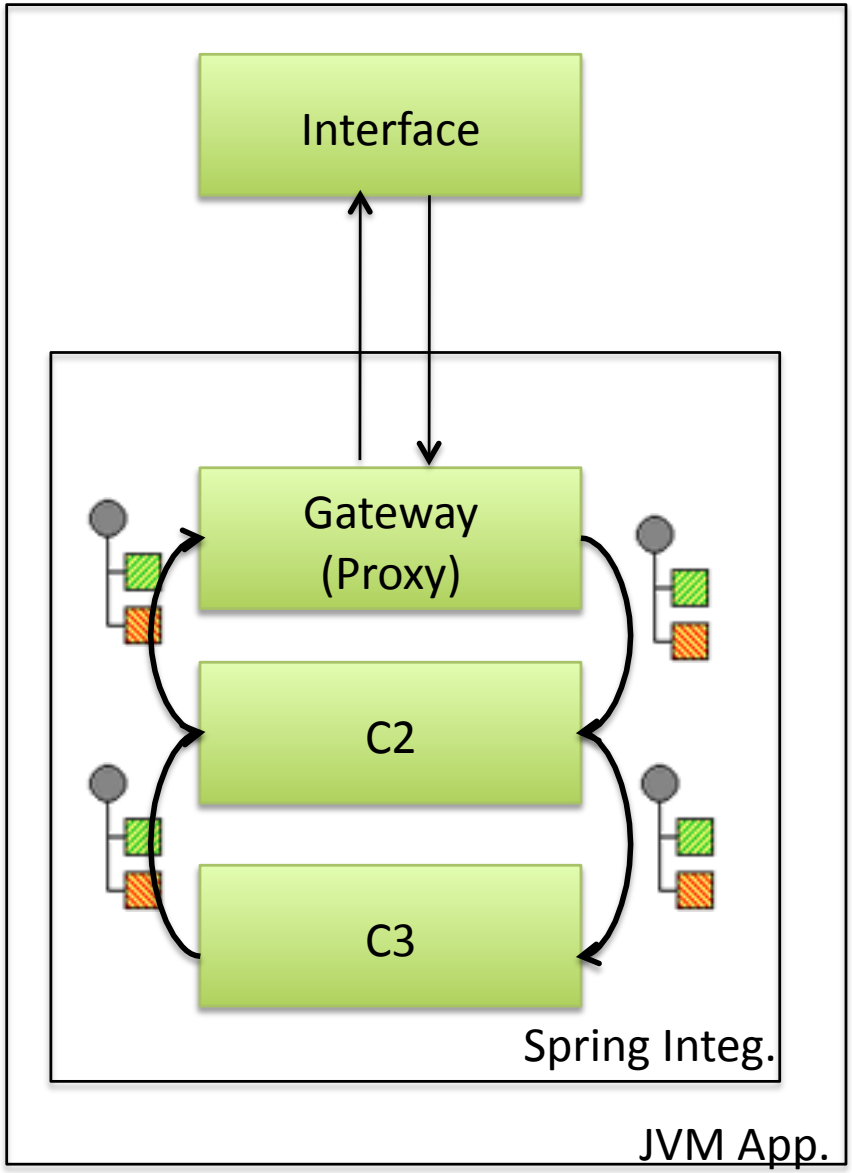
Spring Task Scheduling

Method Invocation



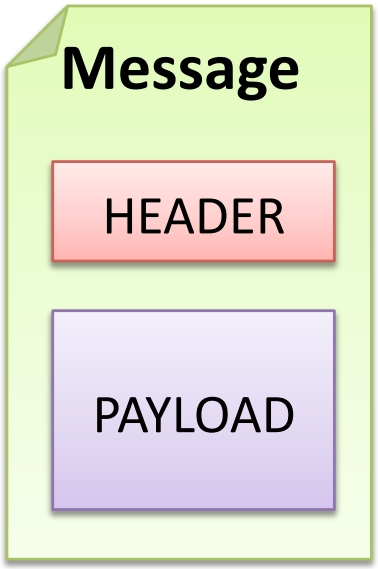
Inter et Intra Process par l'exemple





Spring Integ. Intra Process

Un message Spring Integration



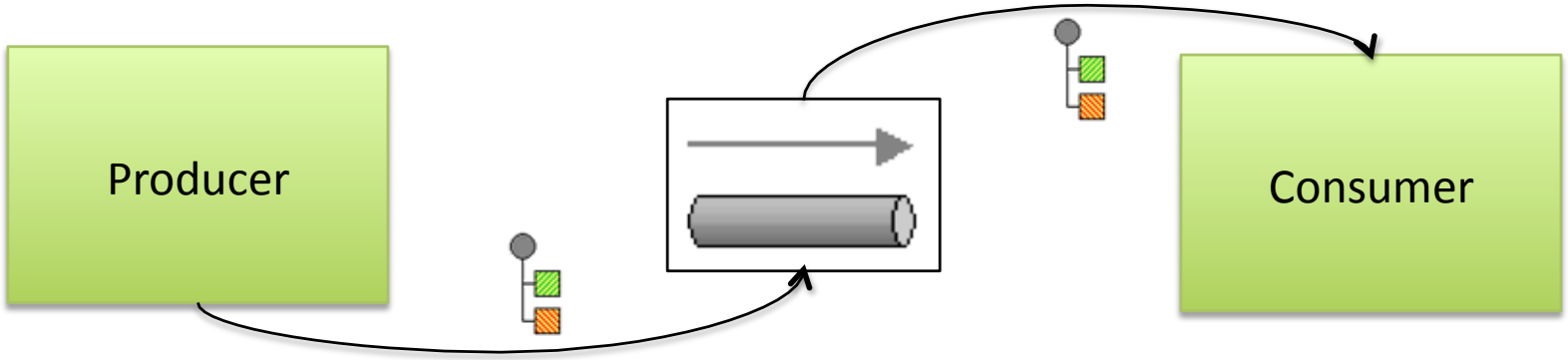

```
public interface Message<T> {

    MessageHeaders getHeaders();

    T getPayload();
}
```

```
public final class MessageHeaders
implements  Map<String, Object>,
                    Serializable {

}
```

Les Headers du Message

MESSAGE_ID
TIMESTAMP
CORRELATION_ID
PRIORITY
EXPIRATION_DATE
ERROR_CHANNEL
REPLY_CHANNEL
SEQUENCE_NUMBER
SEQUENCE_SIZE
...
MY_FUNCTIONAL_HEADER1
MY_FUNCTIONAL_HEADER2
MY_FUNCTIONAL_HEADER3
...

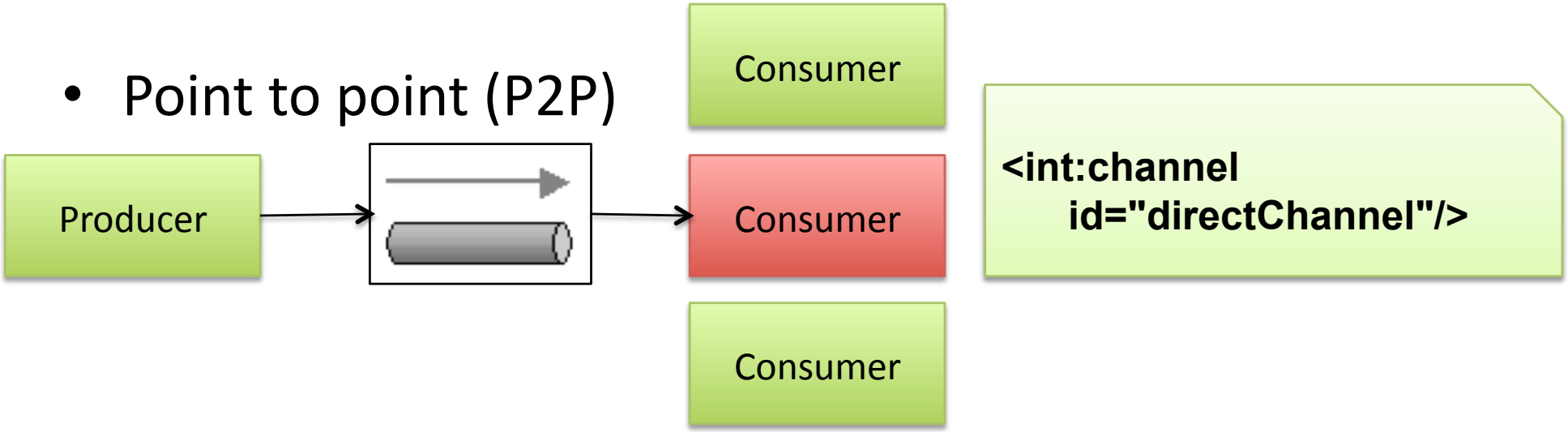
L'importance du Channel



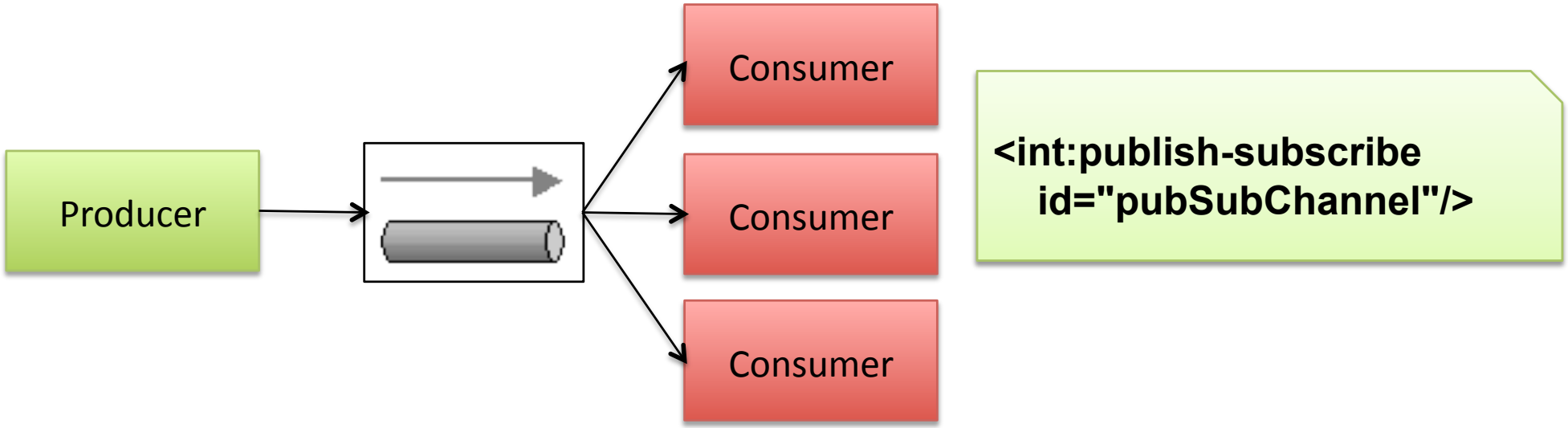
- Elément central
- En mémoire par défaut
  - Backend possible (JMS, JDBC store, etc)
- Facilite le buffering et l'interception

Le mode de destination du channel

- Point to point (P2P)

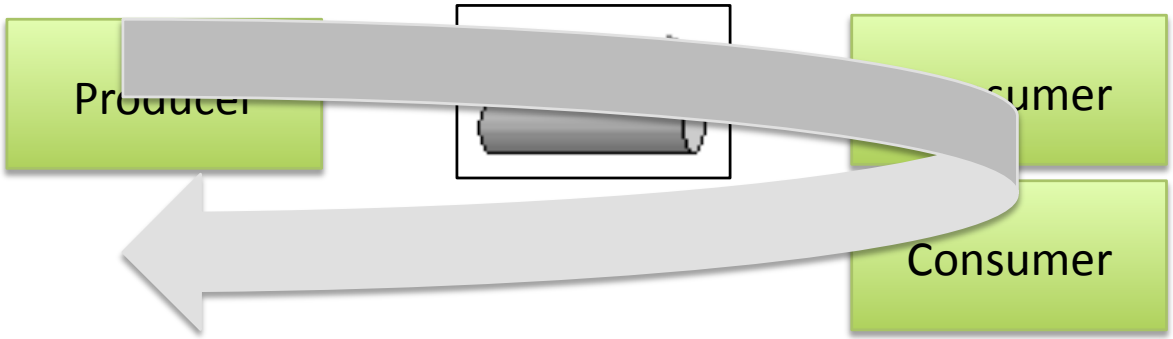


- Publish Subscribe (Pub/Sub)



Les modes de transfert du channel

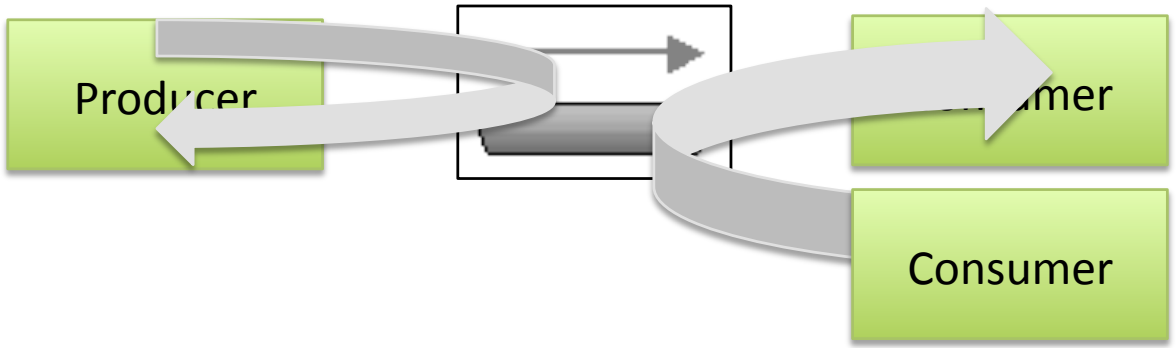
- Synchronous Handoff (Sync)




```
<int:channel
      id="directChannel"/>
```

```
<int:publish-subscribe
    id="pubSubChannel"/>
```

- Asynchronous Handoff (Async)



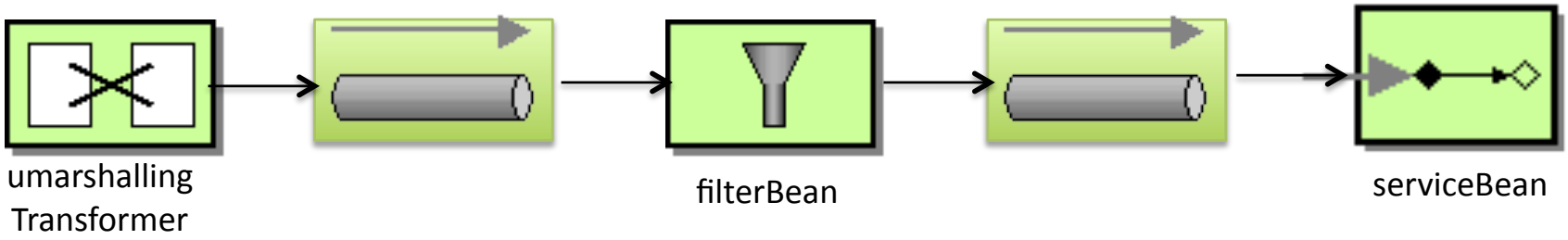

```
<int:channel
      id="directChannel">
<int:dispatcher
 task-executor="executor"/>
</int:channel>
```

```
<int:publish-subscribe
    id="pubSubChannel"
task-executor="executor"/>
```

Les différents types de Channel

- PollableChannel (P2P)
  - QueueChannel
    - Priority Channel
    - Rendezvous Channel
  - NullChannel
- SubscribableChannel
  - DirectChannel (P2P et sync)
  - Publish-subscribe Channel (Pub/sub – sync/async)
  - ExecutorChannel (P2P et async)

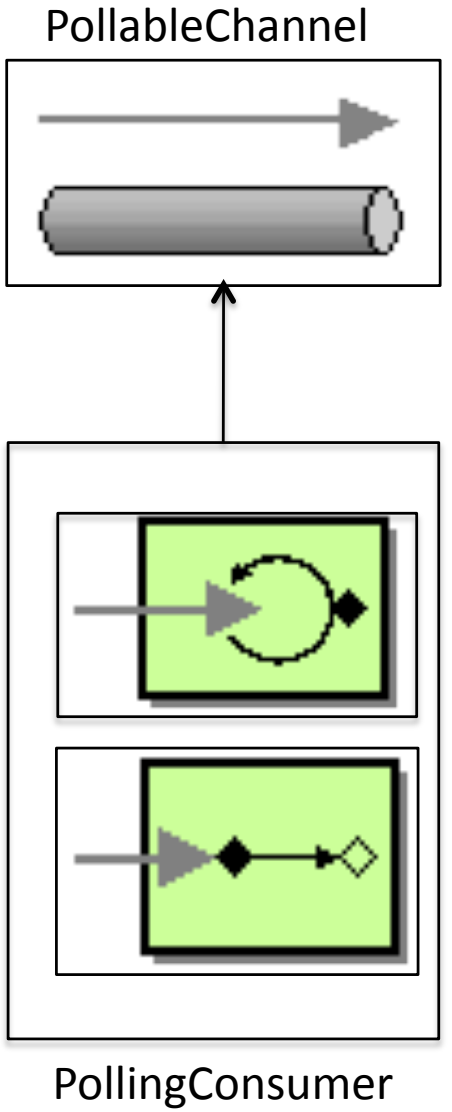
Chaining Endpoints




```
<int:chain input-channel="inputChannel"
           output-channel="outputChain">
    <int:transformer ref="umarshallingTransformerBean"/>
    <int:filter ref="filterBean"/>
    <int:service-activator ref="serviceBean" />
</int:chain>
```

Une API programmatique riche
PollingConsumer (1/2)

```
<int:channel id="queueChannel">
    <int:queue capacity="50"/>
</int:channel>

<int:service-activator
       input-channel="queueChannel"
       output-channel="outputChannel"
       expression="payload.toUpperCase()">
      <int:poller fixed-delay="1000"
                  max-messages-per-poll="5"/>
</int:service-activator>
```



Une API programmatique riche PollingConsumer (2/2)

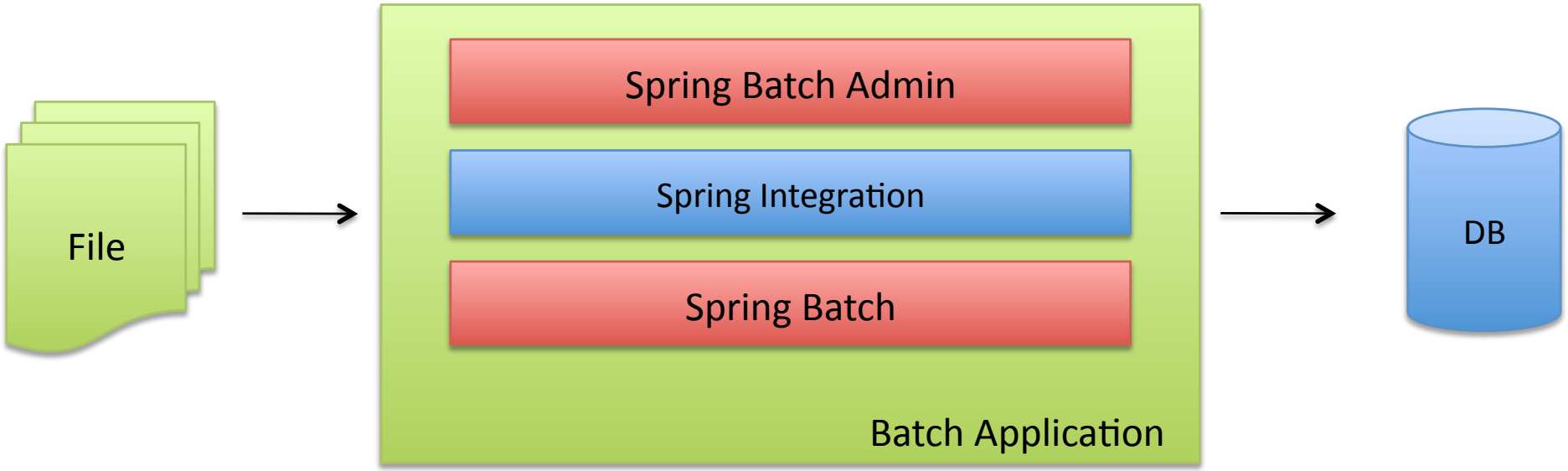
```
<int:channel id="queueChannel">
    <int:queue capacity="50"/>
</int:channel>
```

```
pollableChannel =
    cxt.getBean("queueChannel", QueueChannel.class);
PollingConsumer pollingConsumer =
    new PollingConsumer(pollableChannel, messageHandler);
PeriodicTrigger periodicTrigger =
    new PeriodicTrigger(1, TimeUnit.SECONDS);
pollingConsumer.setBeanFactory(applicationContext);
pollingConsumer.setTrigger(periodicTrigger);
pollingConsumer.setMaxMessagesPerPoll(5);
pollingConsumer.start();
```

Une API programmatique riche
Création de messages

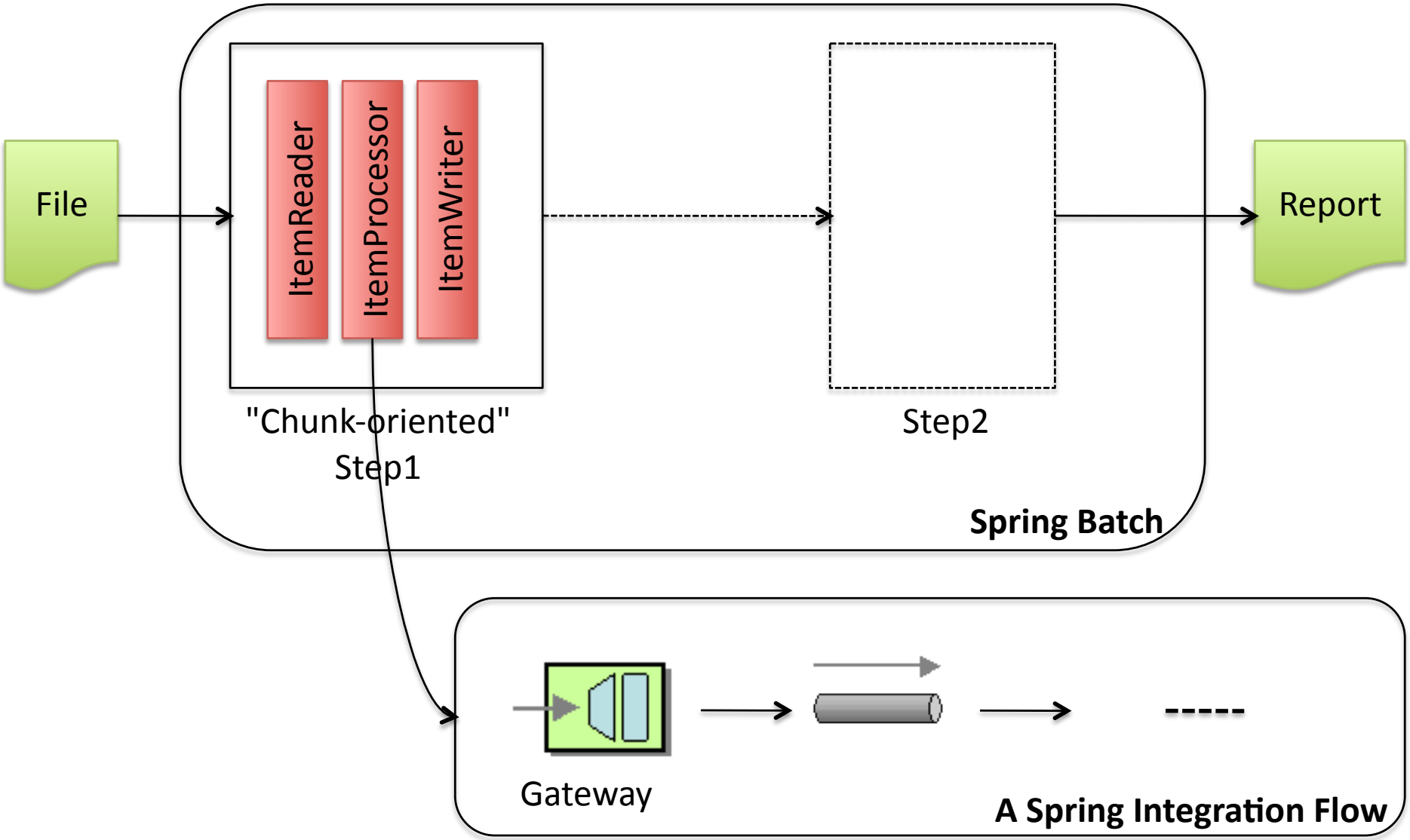
```
MessageHandler messageHandler =
    new MessageHandler() {
      public void handleMessage(Message<?> message)
            throws MessagingException {
     final String payload = (String) message.getPayload();
     outputChannel.send(
       MessageBuilder.withPayload(payload.toUpperCase())
                     .setHeader("key","value")
                     .build());
      }
};
```

Spring Integration et Spring Batch



Spring Integration en tant que brique additionnelle à Spring Batch

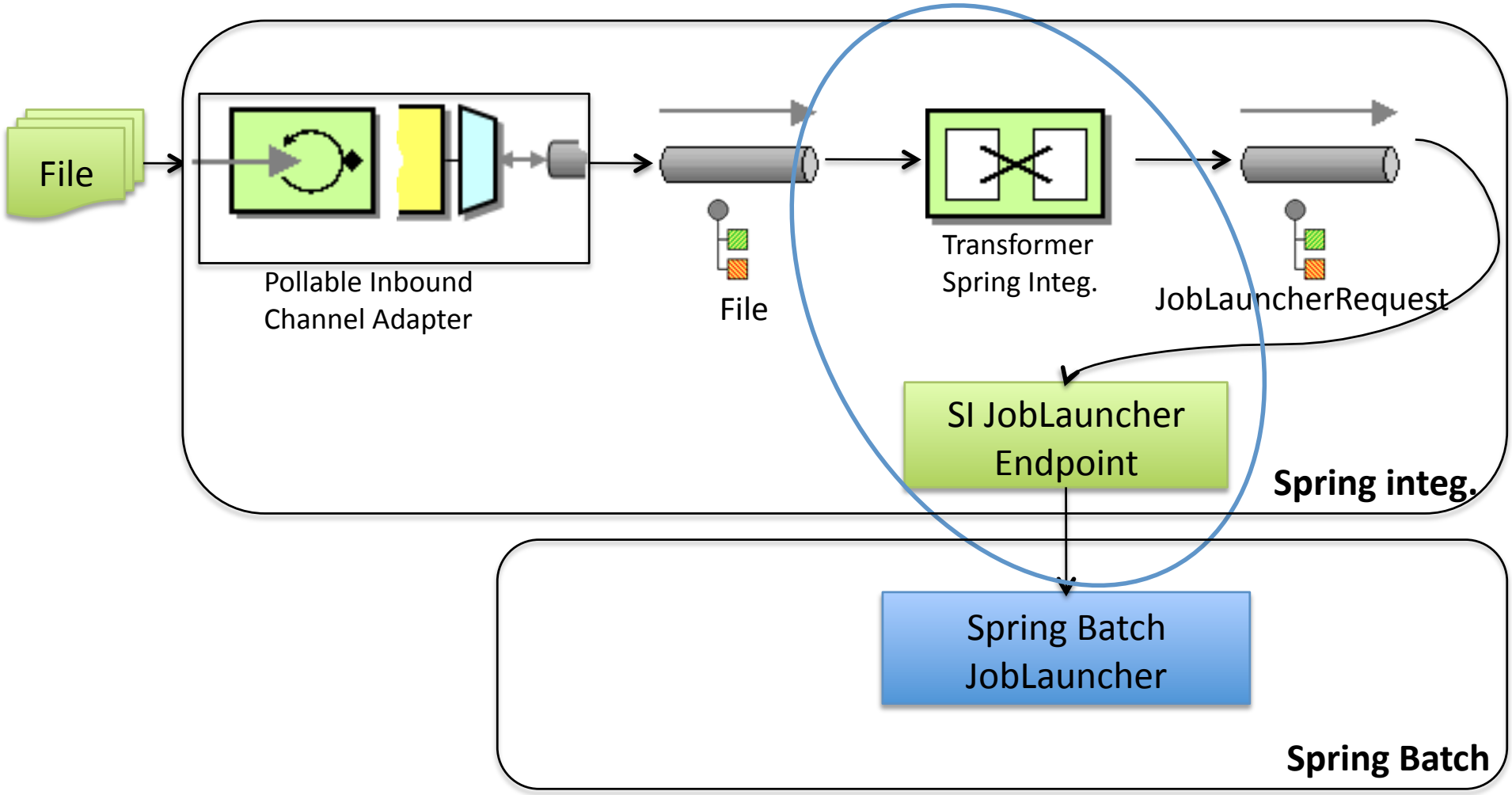
Spring Integration et Spring Batch
Modèle de délégation (1/2)



Spring Integration et Spring Batch
Modèle de délégation (2/2)

```
<int:gateway id="processor"
   service-interface=
                "org.springframework.batch.item.ItemProcessor"
   default-request-channel="inputChannel"
   default-reply-channel="outputChannel"
   default-reply-timeout="2"/>

<int:channel id="inputChannel"/>
<int:channel id="outputChannel"/>
```

Spring Batch et Spring Intégration
Modèle par "Évènement"



Exposition en JMX des endpoints et channels

Java Monitoring & Management Console

Connection Window Help

pid: 1461 com.intellij.rt.execution.application.AppMain com.boissinot.core.file.Main

Overview | Memory | Threads | Classes | VM Summary | MBeans

- JMImplementation
- com.sun.management
- java.lang
- java.util.logging
- spring.application
  - IntegrationMBeanExporter
  - ManagedEndpoint
    - org.springframework.integ
      - endpoint
  - MessageChannel
    - errorChannel
    - inputFilesChannel
      - Attributes
        - MaxSendDuration
        - MeanErrorRate
        - MeanErrorRatio
        - MeanSendDuration
        - MeanSendRate
        - MinSendDuration
        - SendCount
        - SendErrorCount
        - StandardDeviationSend
        - TimeSinceLastSend
      - Operations
      - Notifications
    - nullChannel
  - MessageHandler
  - MessageSource

Attribute values

Name	Value
MaxSendDuration	1.0
MeanErrorRate	0.0
MeanErrorRatio	0.0
MeanSendDuration	0.4736842105263158
MeanSendRate	0.04500256709050285
MinSendDuration	0.0
SendCount	2
SendErrorCount	0
StandardDeviationSendDuration	0.49930699897395464
TimeSinceLastSend	42.663

Refresh

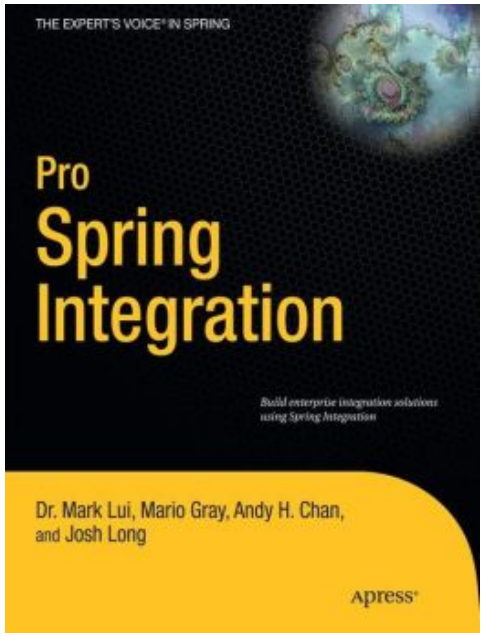
Spring Integration Version log

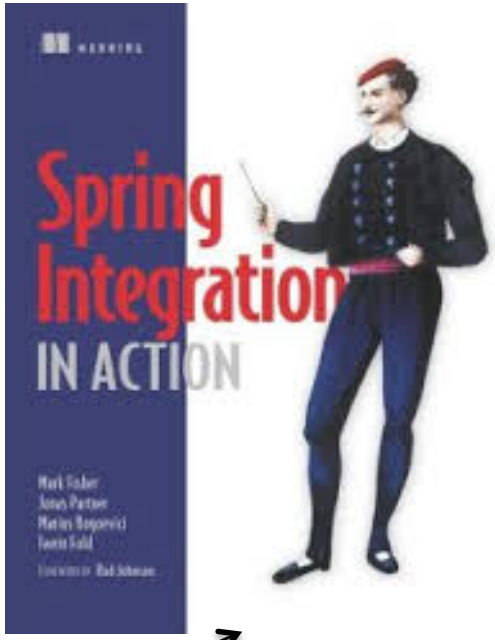
- 1.0.0: Nov. 2008
- 2.0.0: Nov. 2010
- 2.1.0: Janv. 2012
- 2.2.0: Mai 2013
- ...
- 2.2.5: Sept. 2013
- 3.3.0M3: Sept. 2013

Support dans les IDE

Spring Tool Suite (STS) fournit un très bon support Spring Integration

Les livres du marché





Le plus complet

Apache Camel

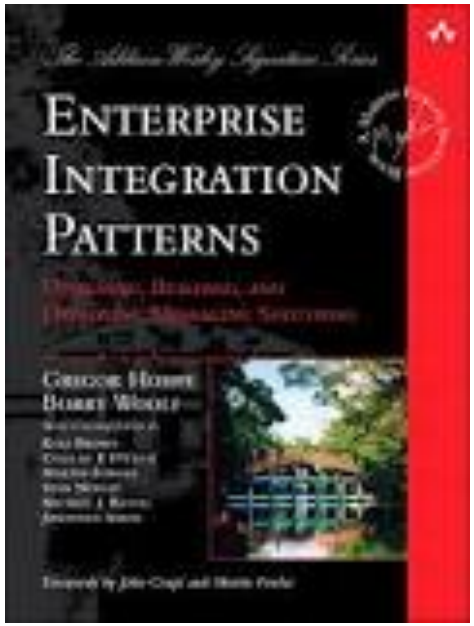
Apprivoisez le chameau

Guillaume Giamarchi

@ggiamarchi

James Strachan

Read



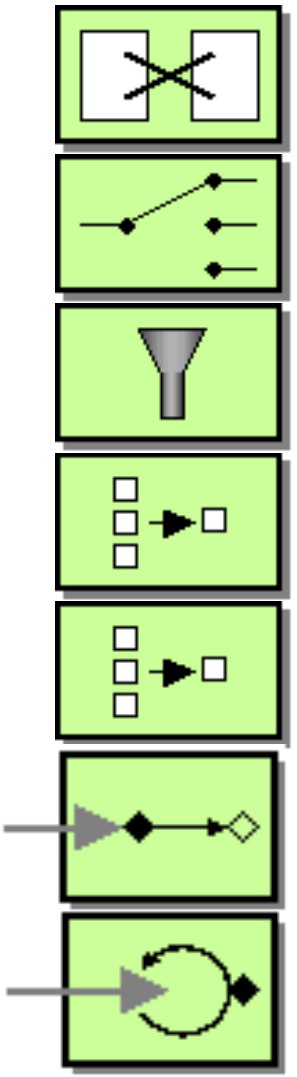
Write

Implements

Apache Camel

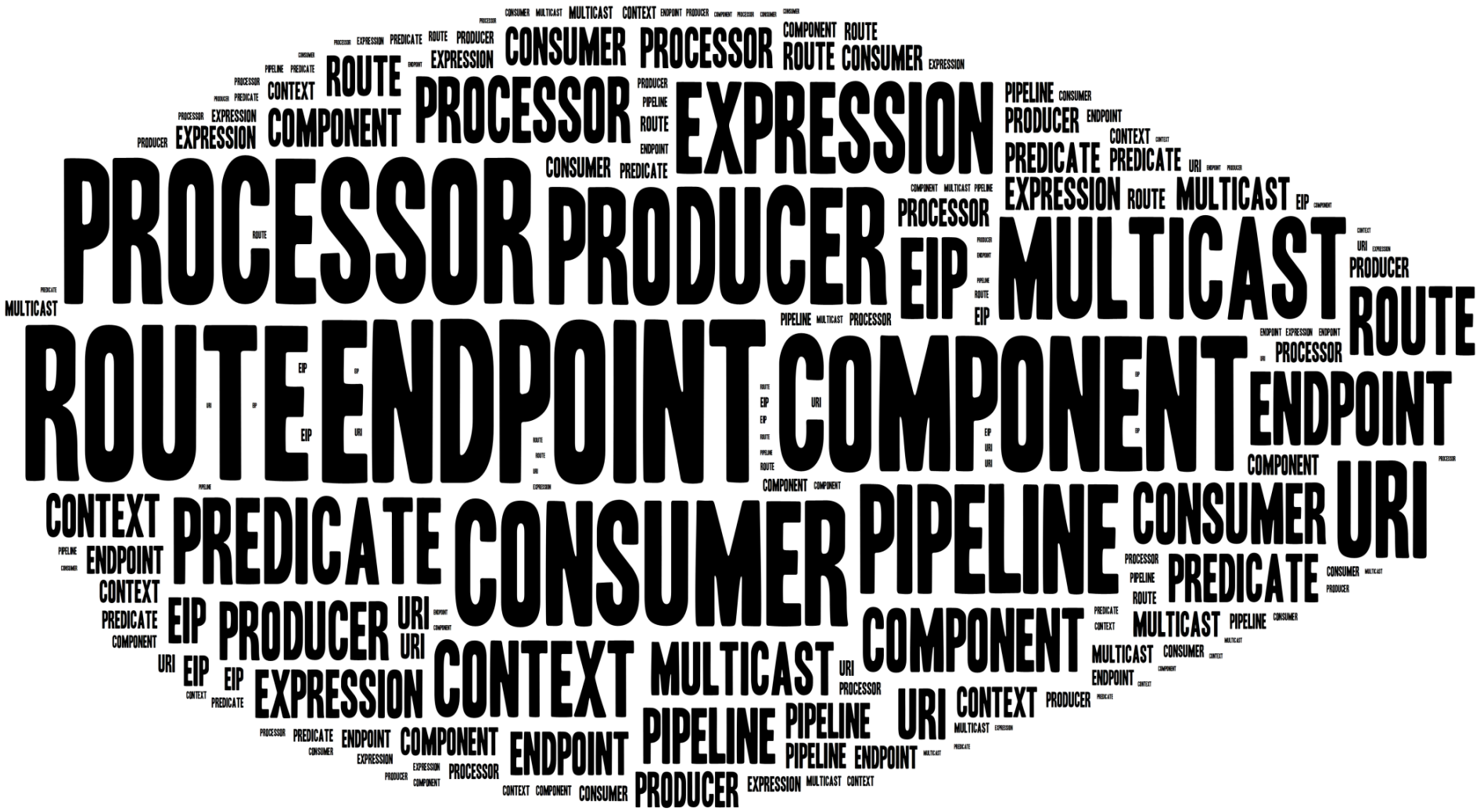
Leads

Claus Ibsen

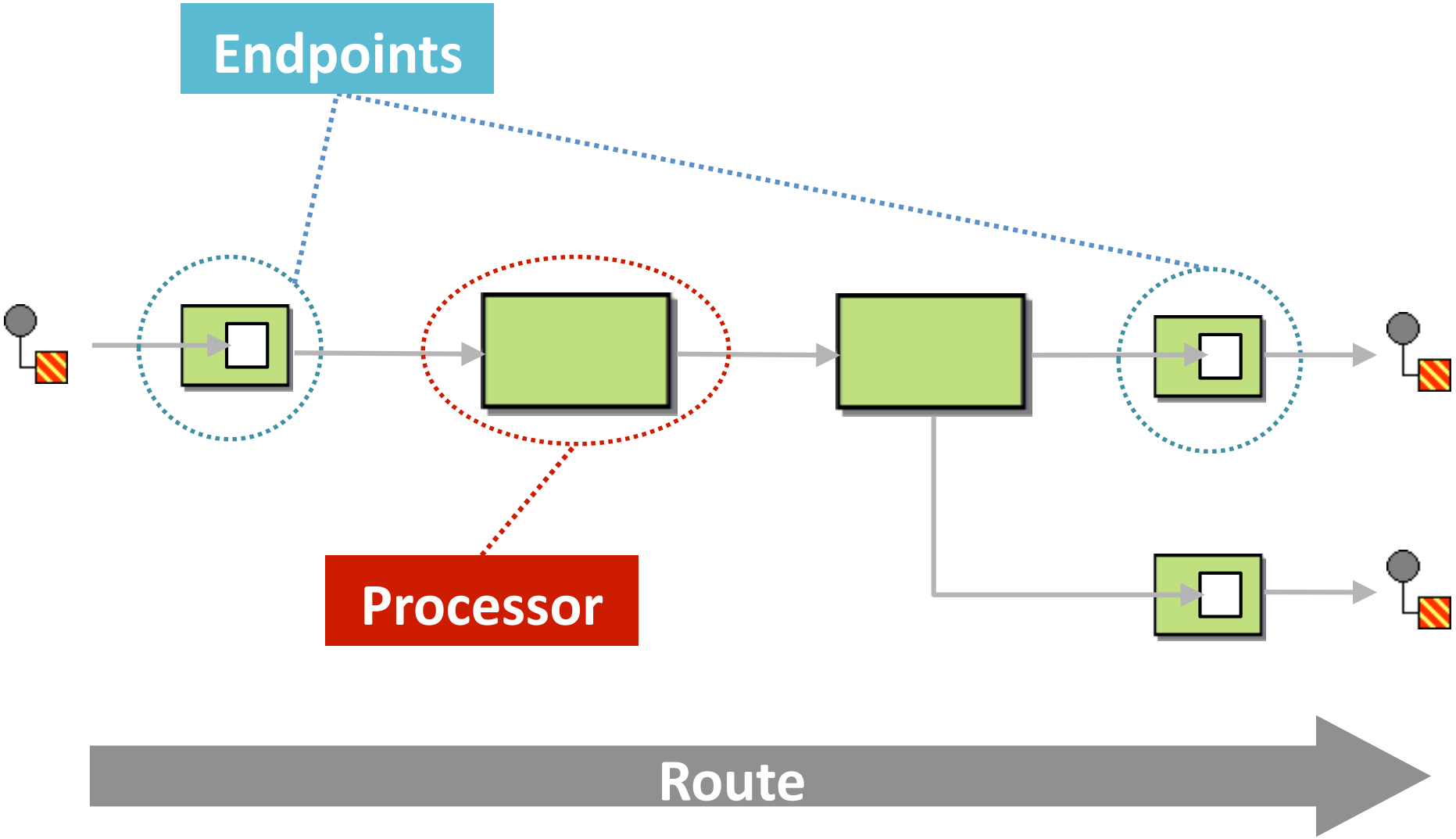


- Framework Java Open Source
- Implémentation des EIP
- Java DSL (Domain Specific Language)
- Très bonne intégration Spring

Apache Camel



En route…



Apache Camel


```java
import com.zenika.camel;

import org.apache.camel.builder.RouteBuilder;

public class HelloWorldRoute extends RouteBuilder {
    @Override
    public void configure() throws Exception {

        from("jms:queue:in").to("bean:myServiceBean");

    }
}
```

```xml
<camelContext xmlns="http://camel.apache.org/schema/spring">
    <package>com.zenika.camel</package>
</camelContext>

<bean id="myServiceBean" class="..." />
```


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

    <camelContext xmlns="http://camel.apache.org/schema/spring">
        <route>
            <from uri="jms:queue:in" />
            <to uri="bean:myServiceBean" />
        </route>
    </camelContext>

    <bean id="myServiceBean" class="..." />

</beans>
```

Dans les coulisses…

Les Endpoints

- Jouent le rôle d'interfaces
- Sont Exprimés par des URI

```
scheme:adresse[?options]
```

Type de composant

Dans les coulisses…

Les Endpoints

- Physiques

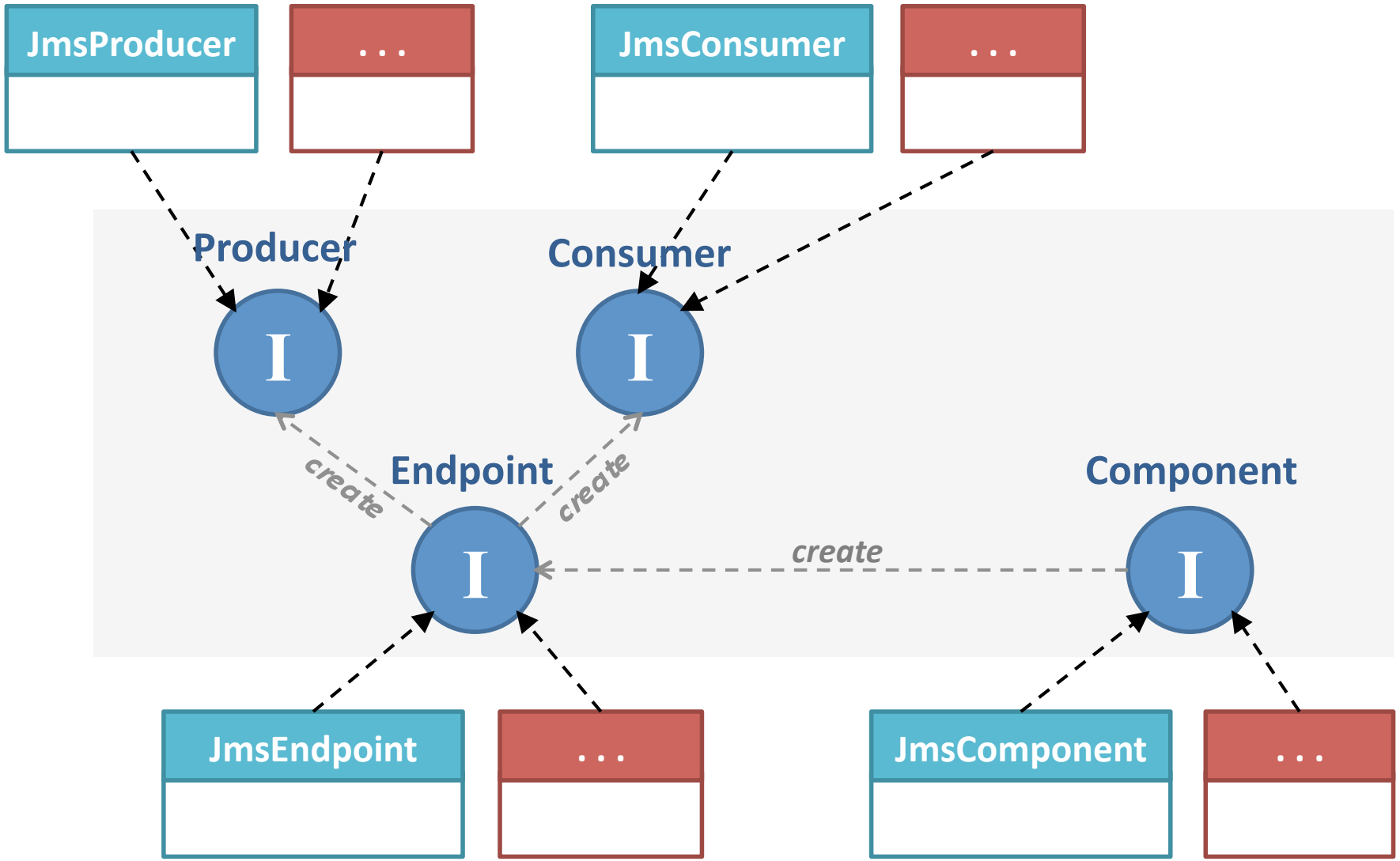
```
jms:queue:input
file:/home/guillaume/camel/out?delay=3000
http://www.zenika.com
```

- Logiques

```
from("jms:queue:in")
    .to("direct:callBean");

from("direct:callBean")
    .to("bean:myServiceBean");
```

Dans les coulisses…

Composants et Endpoints



Des composants

très classiques…

HTTP	FTP	SSH	JMS	File
JDBC	SMTP	IMAP	POP3	AMQP
LDAP	TCP	JMX	XSLT	. . .

Des composants

un peu moins classiques…

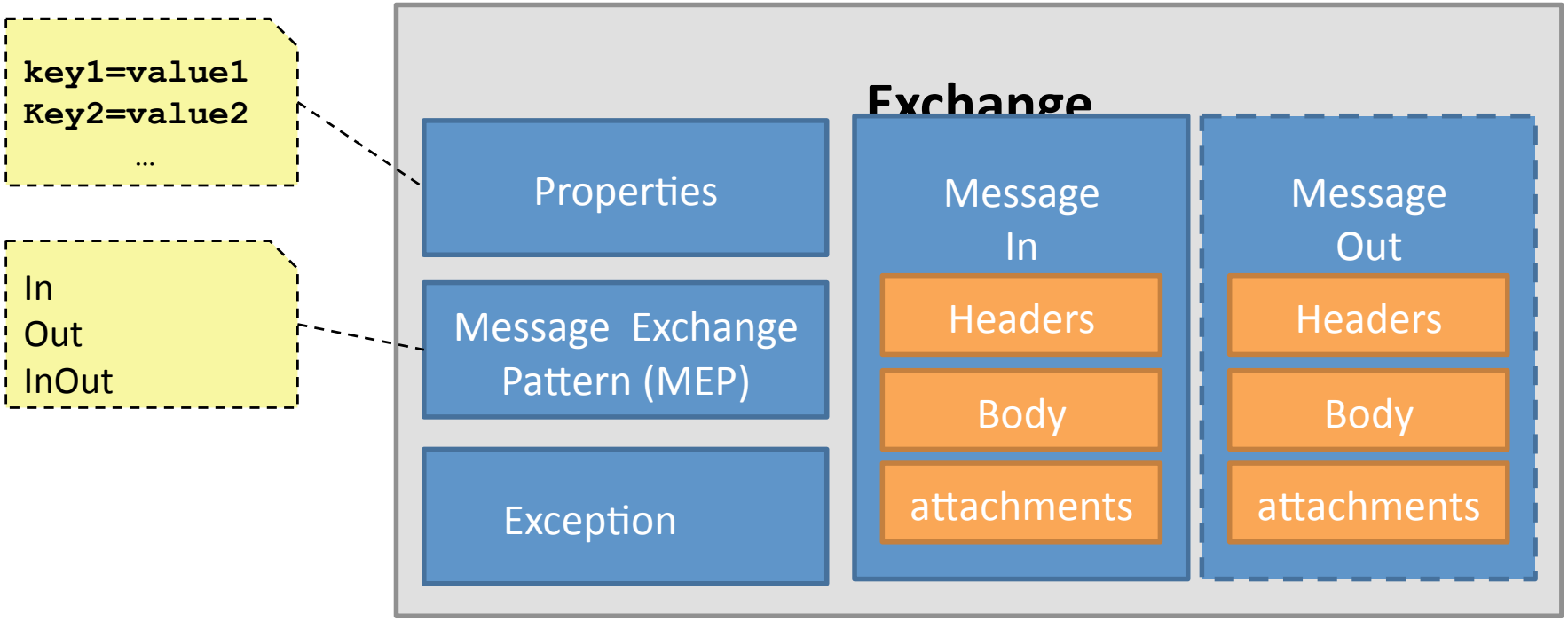
MyBatis	CouchDB	Hazelcast	Vert.x	MongoDB
Twitter	GMail	RabbitMQ	Nagios	Elasticsearch
Facebook	AWS-S3	Freemarker	Lucene	. . .

Dans les coulisses…

Les messages

Exchanges

- Les ~~messages~~ transitent sur les routes



Les Processors


```
public class MyProcessor implements Processor {

  public void process(Exchange exchange) throws Exception {
    Message in = exchange.getIn();
    String body = in.getBody(String.class);
    in.setBody(body + "abc")
  }

}
```

```
from("jms:queue:in")
    .process(new MyProcessor())
    .to("jms:queue:out");
```

Apache Camel

Cas pratique

User notification for Bob

Administrator notification, error level

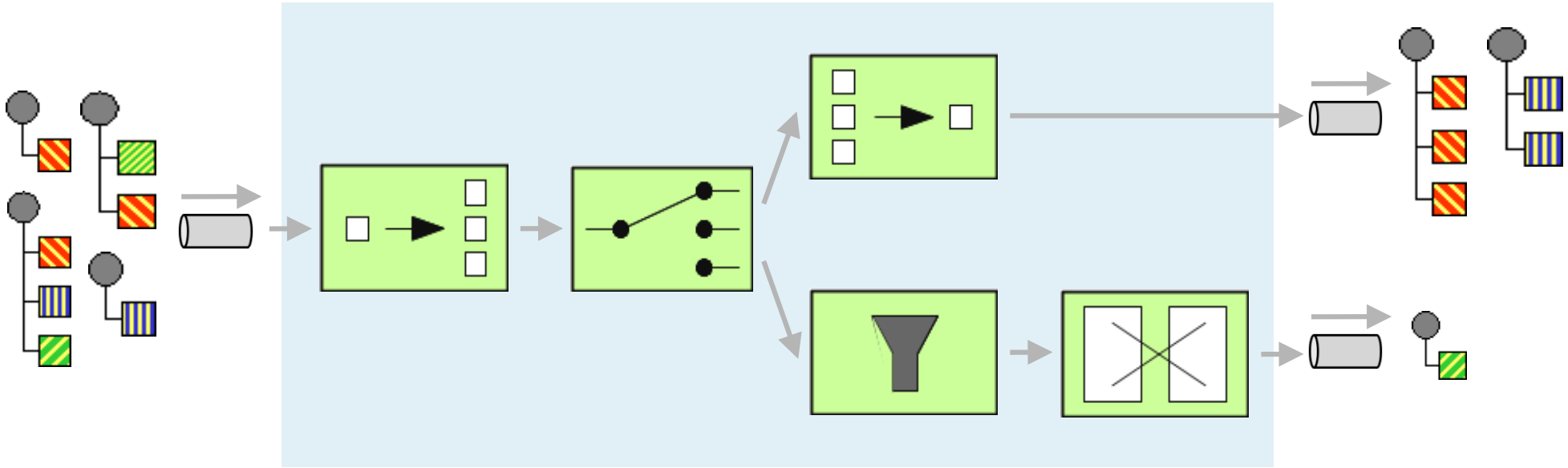
User notification for Alice

Administrator notification, info level

Cas pratique

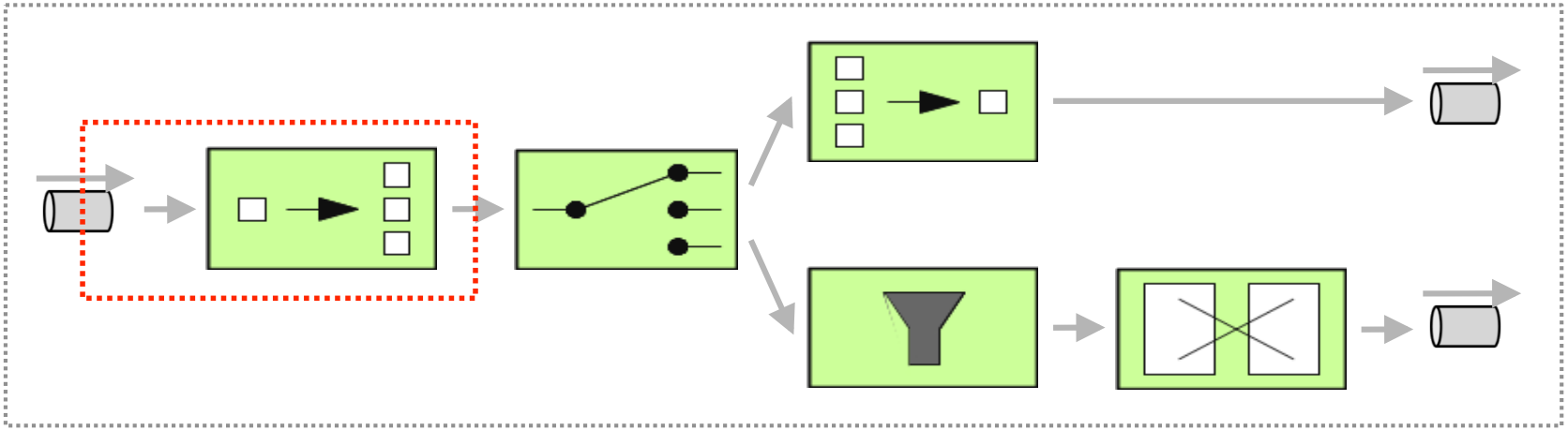

```xml
<notifications xmlns="http://www.zenika.com/camel/notif/model">
    <admin>
        <severity>ERROR</severity>
        <message>Exception during processing...</message>
    </admin>
    <user>
        <type>INVITATION</type>
        <from>Guillaume</from>
        <to>Bob</to>
        <title>Football match next sunday</title>
        <text>Hi all, who is available ?</text>
    </user>
    <user>
        <type>INVITATION</type>
        <from>Guillaume</from>
        <to>Alice</to>
        <title>Trip arround the world</title>
        <text>We plan an amazing trip! Are you in ?</text>
    </user>
</notifications>
```

Cas pratique



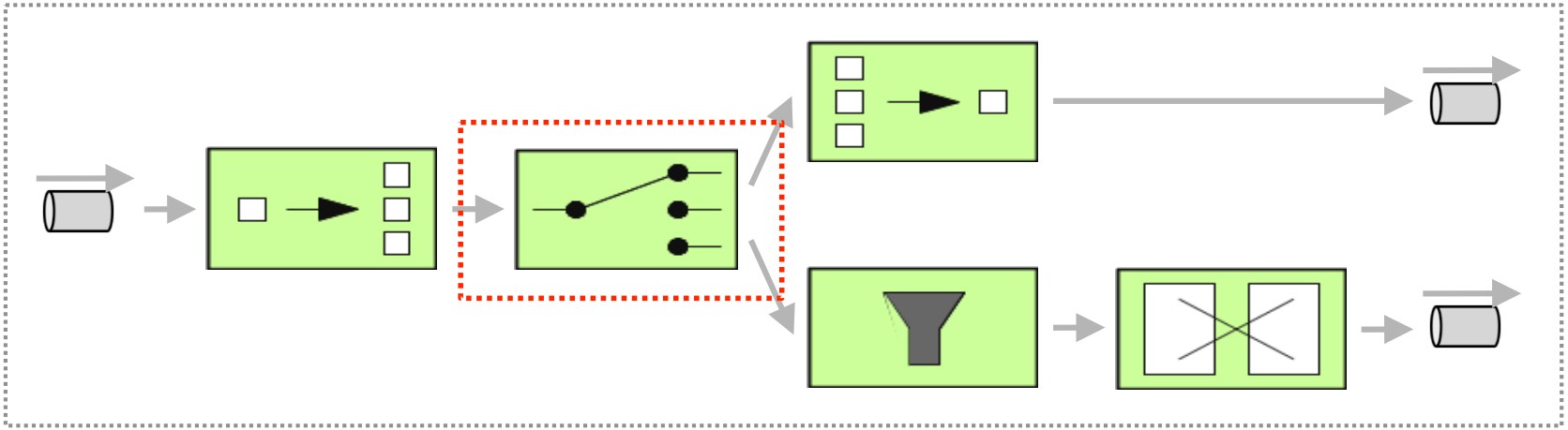
Cas pratique


```
Namespaces NS = new Namespaces("n", "...");

from(ENDPOINT_URI_IN)
    .split()
        .xpath("/n:notifications/child::*", NS)
    .to(ENDPOINT_URI_ROUTER);
```



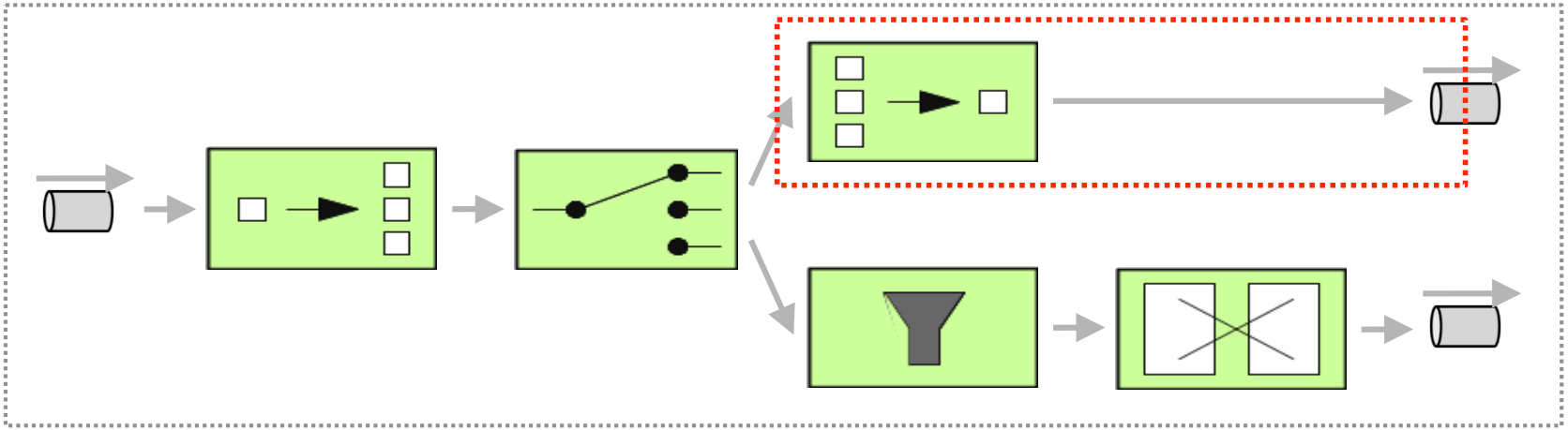
Cas pratique


```
from(ENDPOINT_URI_ROUTER)
    .choice()
        .when().xpath("/n:user", NS)
            .to(ENDPOINT_URI_USER_AGGREGATOR)
        .when().xpath("/n:admin", NS)
            .to(ENDPOINT_URI_ADMIN_FILTER)
        .otherwise()
            .to(ENDPOINT_URI_UNKNOWN_NOTIFICATION);
```



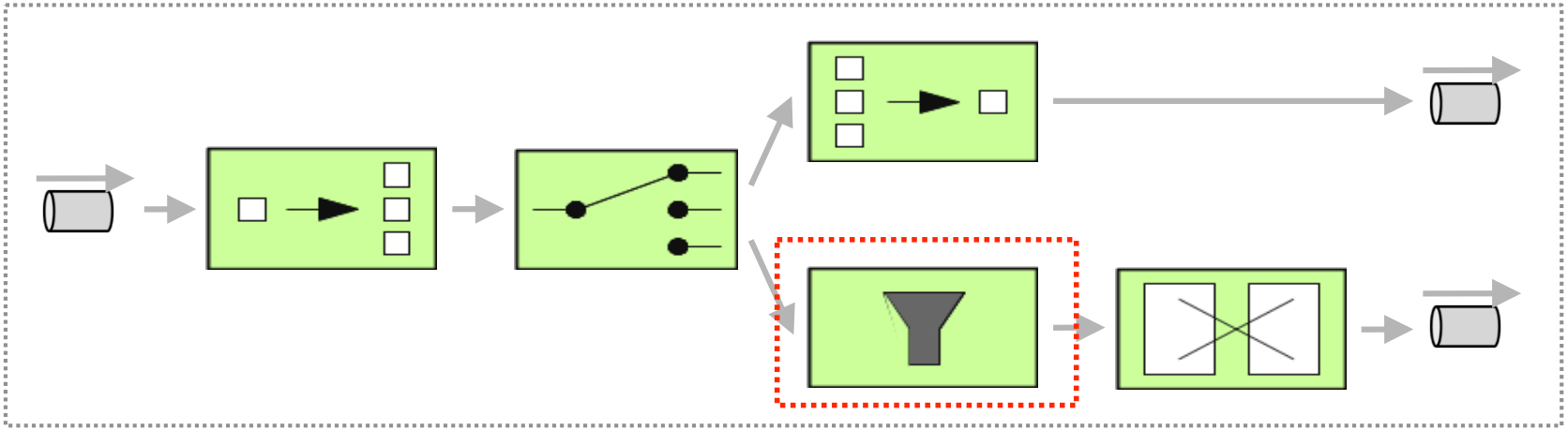
Cas pratique


```
from(ENDPOINT_URI_USER_AGGREGATOR)
    .aggregate()
        .xpath("/n:user/n:to", NS)
        .aggregationStrategy(aggStrategy)
        .completionSize(10)
        .completionTimeout(5000)
    .convertBodyTo(String.class)
    .to(ENDPOINT_URI_USER_OUT);
```



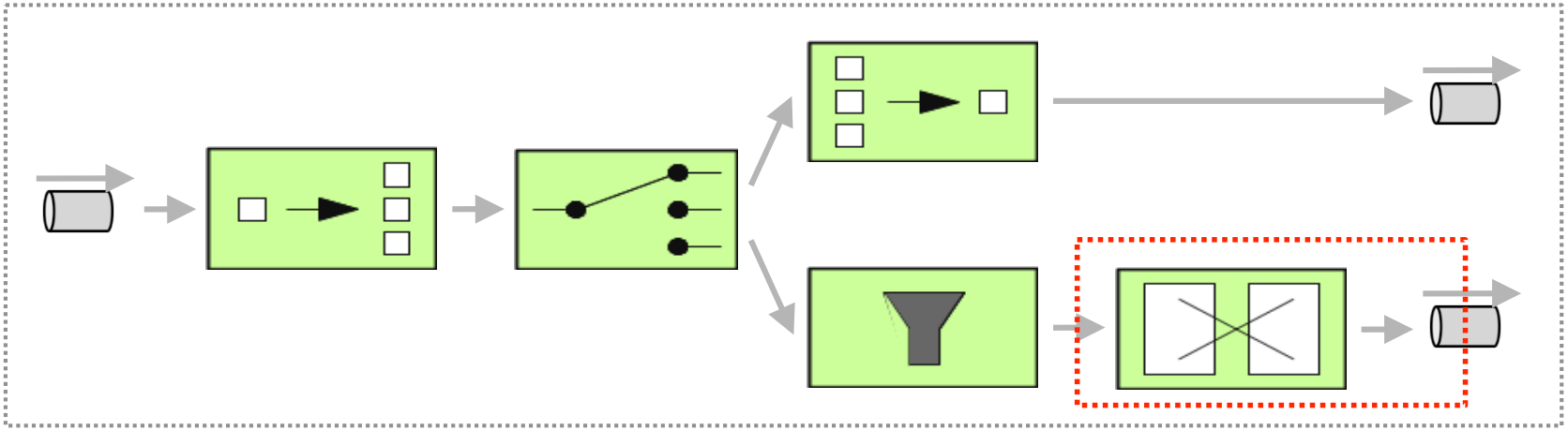
Cas pratique


```
from(ENDPOINT_URI_ADMIN_FILTER)
    .filter()
        .xpath("/n:admin/n:severity/text() = 'ERROR'", NS)
.to(ENDPOINT_URI_ADMIN_TRANSLATOR);
```



Cas pratique


```
from(ENDPOINT_URI_ADMIN_TRANSLATOR)
    .to("bean:notificationTransformer")
    .to(ENDPOINT_URI_ADMIN_OUT);
```

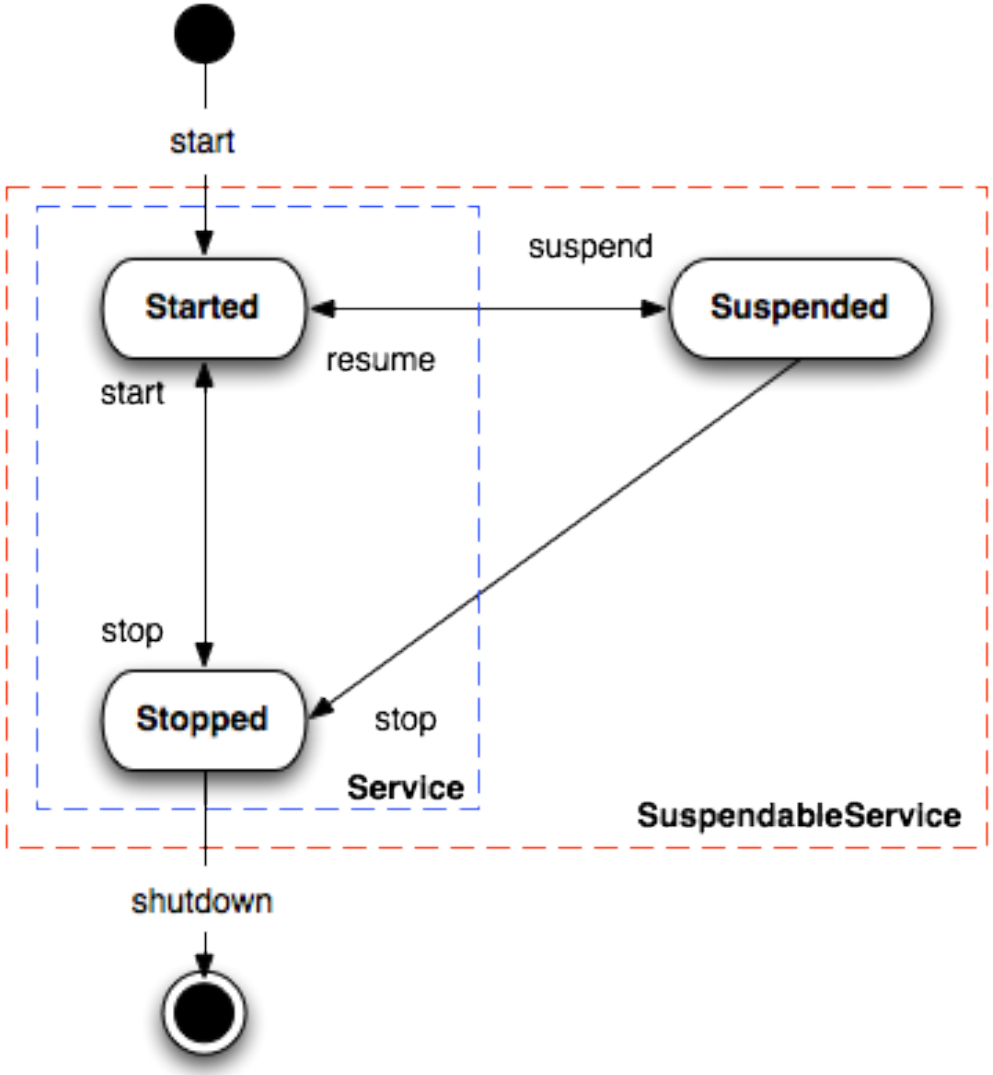


Cas pratique

https://github.com/ggiamarchi/EIP-Camel

Management & Monitoring

Cycle de vie




```java
package org.apache.camel;

public interface Service {

  void start() throws Exception;
  void stop() throws Exception;

}
```

```java
package org.apache.camel;

public interface SuspendableService
extends Service {

  void suspend() throws Exception;
  void resume() throws Exception;
  boolean isSuspended();

}
```

Management & Monitoring

Overview | Memory | Threads | Classes | VM Summary | MBeans

- JMImplementation
- com.sun.management
- java.lang
- java.util.logging
- org.apache.camel
  - components
    - Guil/contextTP1
      - "file"
  - consumers
    - Guil/contextTP1
      - FileConsumer(0x1c8321c8)
  - context
  - endpoints
    - Guil/contextTP1
      - "file://C:/Camel/resanet/resa_in_tp1"
      - "file://C:/Camel/resanet/resa_out_tp1"
      - "spring-event://default"
  - errorhandlers
  - processors
    - Guil/contextTP1
  - routes
    - Guil/contextTP1
      - "route1"
        - Attributes
        - Operations
        - Notifications
  - services
  - threadpools
  - tracer

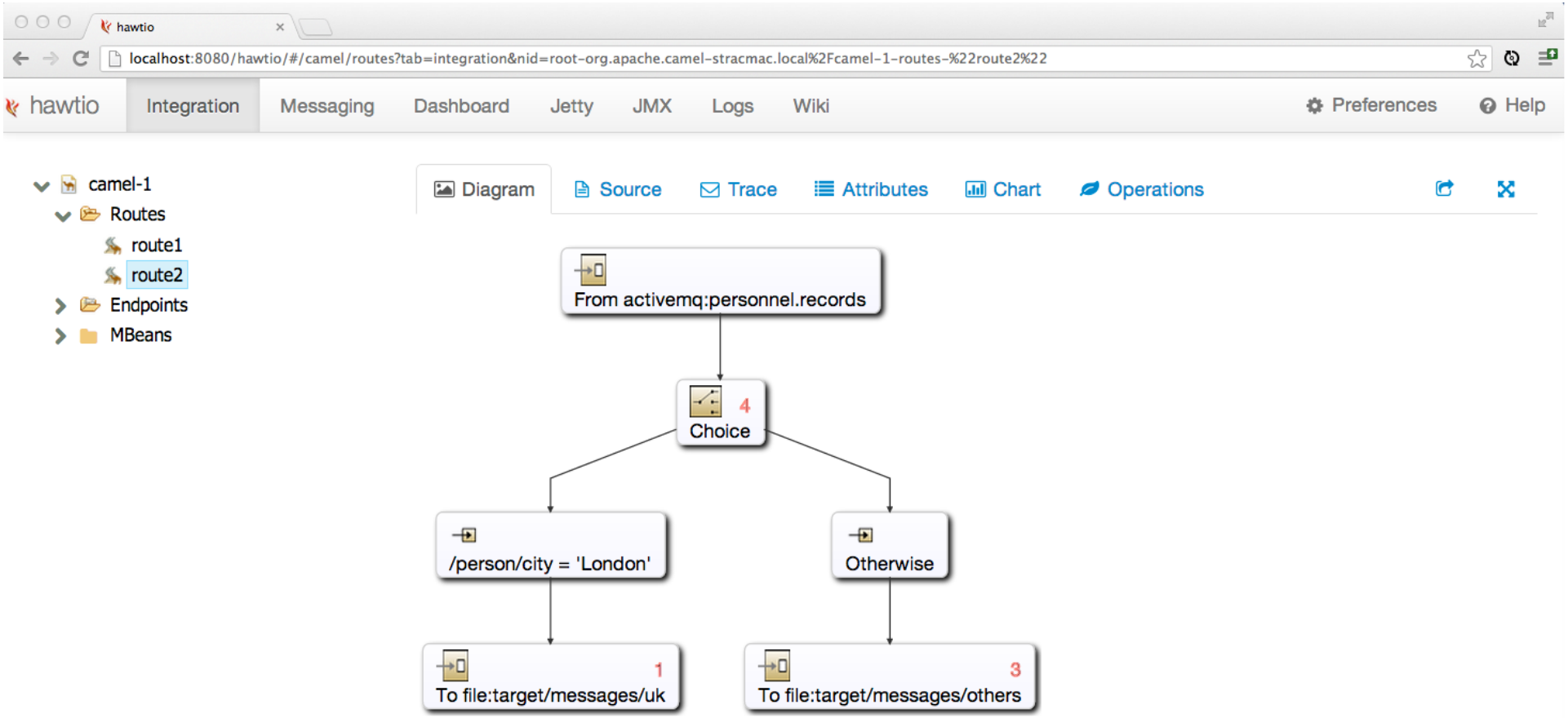
Attribute values

Name	Value
CamelId	contextTP1
DeltaProcessingTime	**0**
Description	EventDrivenConsumerRoute[Endpoint[file://C:/Camel/resanet/resa_in_tp1] -> Instrumenta...
EndpointUri	file://C:/Camel/resanet/resa_in_tp1
ExchangesCompleted	**0**
ExchangesFailed	**0**
ExchangesTotal	**0**
ExternalRedeliveries	**0**
FailuresHandled	**0**
FirstExchangeCompletedExchangeId	
FirstExchangeCompletedTimestamp	
FirstExchangeFailureExchangeId	
FirstExchangeFailureTimestamp	
InflightExchanges	**0**
LastExchangeCompletedExchangeId	
LastExchangeCompletedTimestamp	
LastExchangeFailureExchangeId	
LastExchangeFailureTimestamp	
LastProcessingTime	**0**
Load01	0,00
Load05	0,00
Load15	0,00
MaxProcessingTime	**0**
MeanProcessingTime	**0**
MinProcessingTime	**0**
Redeliveries	**0**
ResetTimestamp	Mon Oct 07 20:08:00 CEST 2013
RouteId	route1
RoutePolicyList	
State	Started
StatisticsEnabled	true
TotalProcessingTime	**0**

Refresh

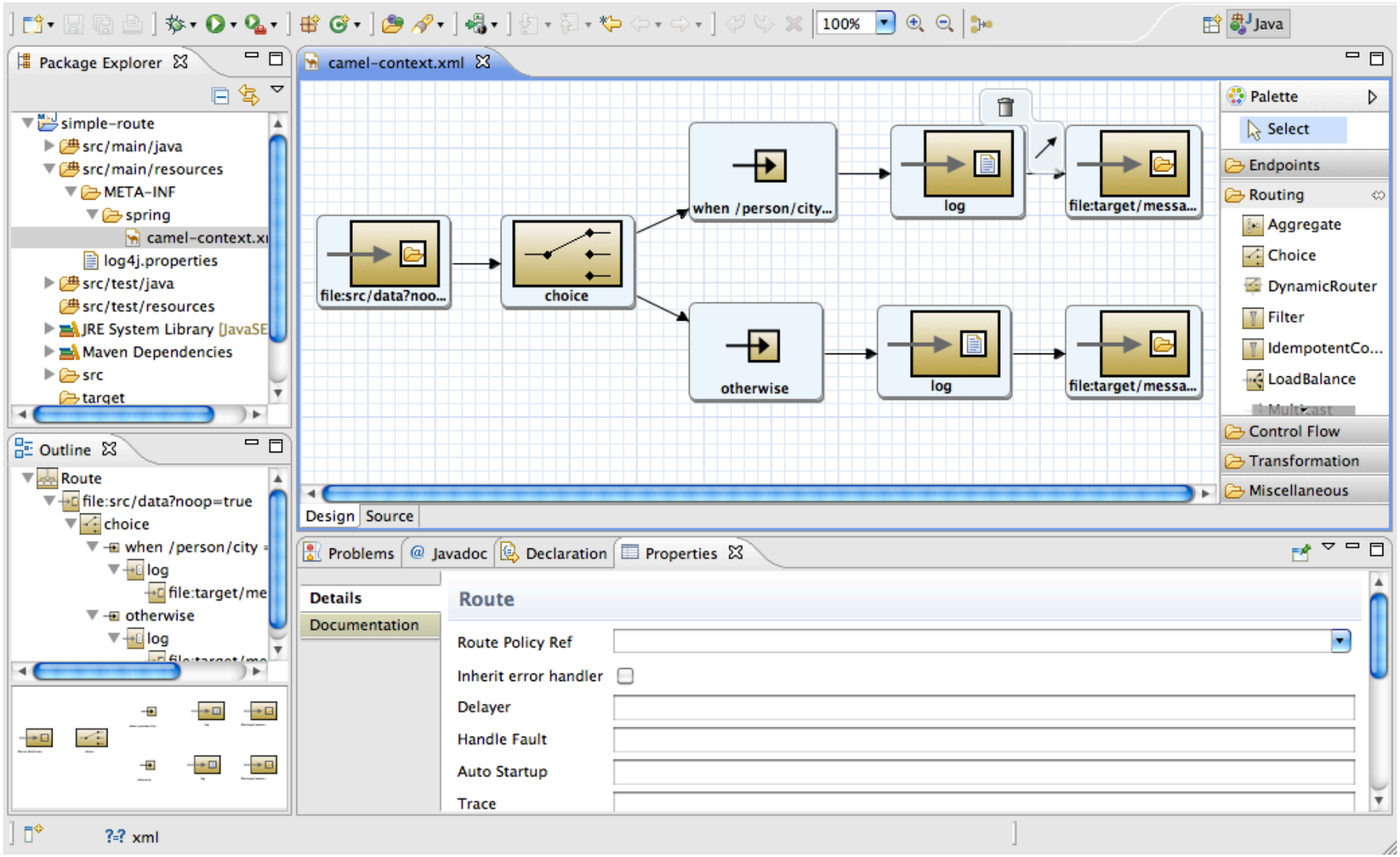
Management & Monitoring

http://hawt.io/

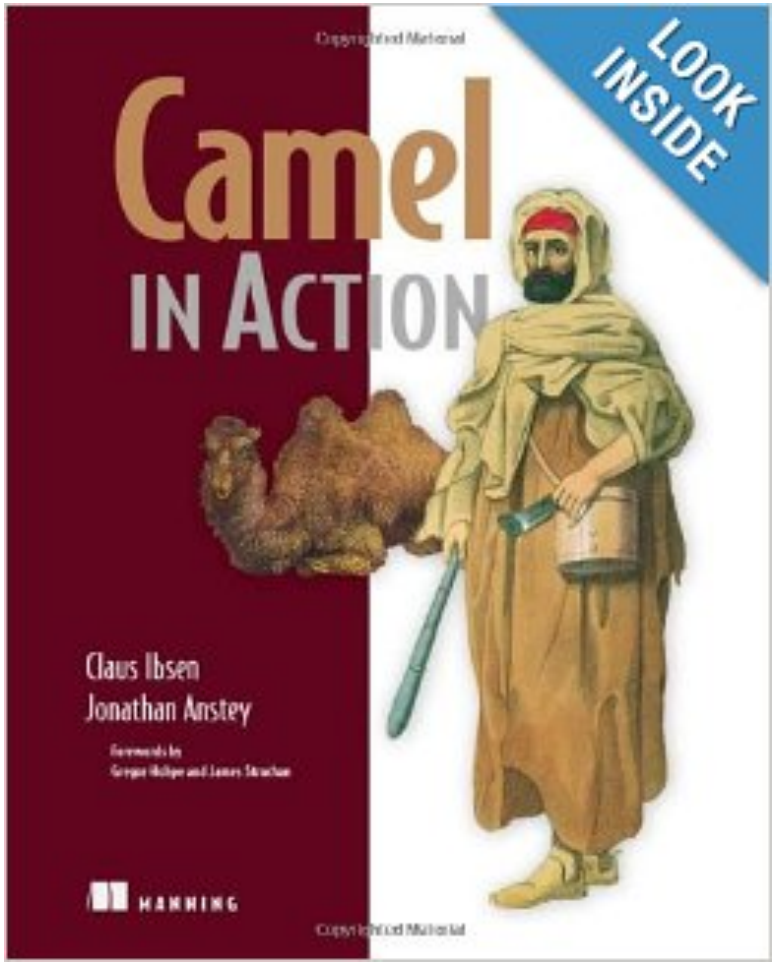


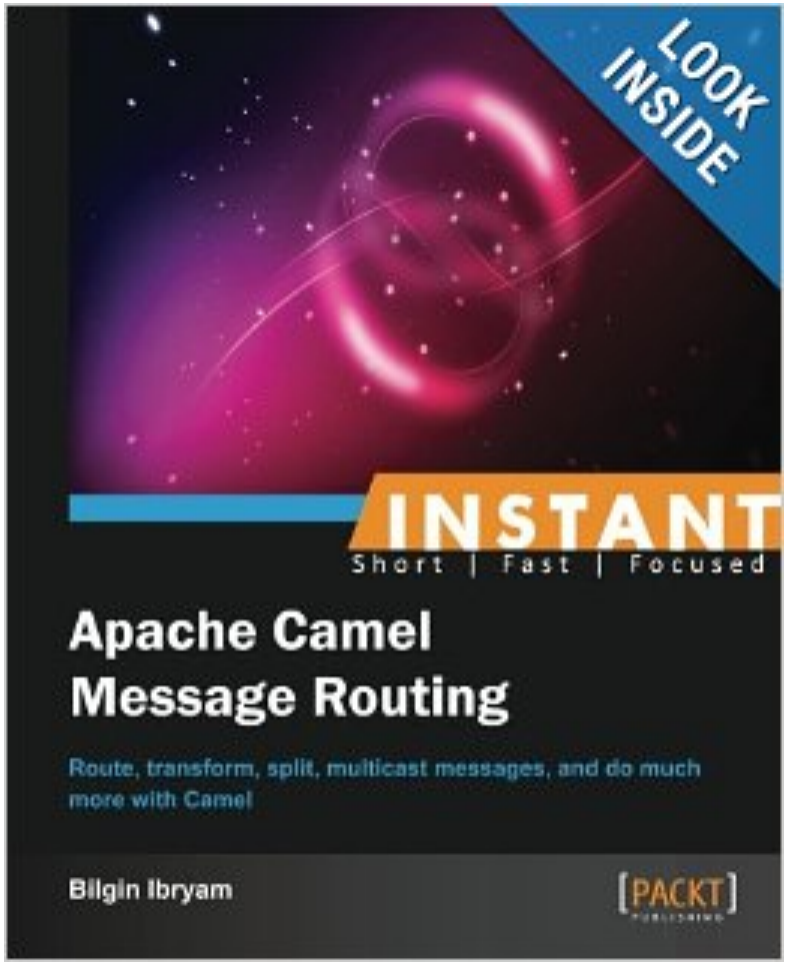
Fuse IDE

http://fusesource.com/products/fuse-ide/



Livres


LOOK INSIDE
Camel
IN ACTION
Claus Ibsen
Jonathan Anstey
MANNING


LOOK INSIDE
INSTANT
Short | Fast | Focused
Apache Camel
Message Routing
Route, transform, split, multicast messages, and do much more with Camel
Bilgin Ibryam
[PACKT]

Apache Camel

Conclusion

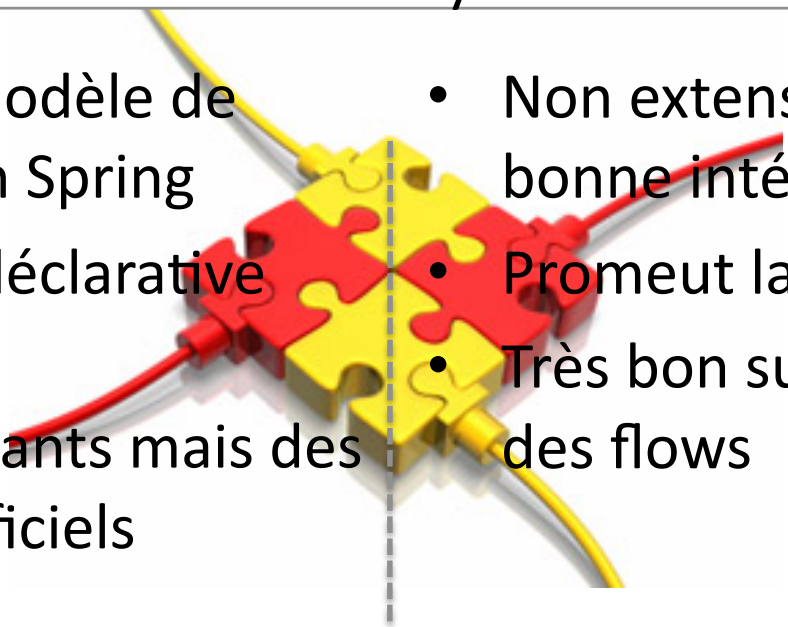
"Spring Integration vs Apache Camel"
2 approches pour un même objectif

Spring Integration

Apache Camel

- Brique d'intégration légère et non invasive supportant les EIP
- Simple à introduire dans son système d'information

- Extension du modèle de programmation Spring
- Configuration déclarative des endpoints
- Peu de composants mais des composants officiels



- Non extension mais très bonne intégration Spring
- Promeut la vision Route
- Très bon support des tests des flows