Easy Enterprise Integration Patterns
with Apache Camel,
ActiveMQ and ServiceMix

James Strachan
http://macstrac.blogspot.com/

http://open.iona.com/
What are Enterprise Integration Patterns?
Book by Gregor & Bobby!
A selection of some of the patterns...

### Message Routing

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Based Router</td>
<td>How do we handle a situation where the implementation of a single logical function (e.g., inventory check) is spread across multiple physical systems?</td>
</tr>
<tr>
<td>Message Filter</td>
<td>How can a component avoid receiving uninteresting messages?</td>
</tr>
<tr>
<td>Recipient List</td>
<td>How do we route a message to a list of dynamically specified recipients?</td>
</tr>
<tr>
<td>Splitter</td>
<td>How can we process a message if it contains multiple elements, each of which may have to be processed in a different way?</td>
</tr>
<tr>
<td>Aggregator</td>
<td>How do we combine the results of individual, but related messages so that they can be processed as a whole?</td>
</tr>
<tr>
<td>Resequencer</td>
<td>How can we get a stream of related but out-of-sequence messages back into the correct order?</td>
</tr>
<tr>
<td>Throttler</td>
<td>How can I throttle messages to ensure that a specific endpoint does not get overloaded, or we don't exceed an agreed SLA with some external service?</td>
</tr>
<tr>
<td>Delayer</td>
<td>How can I delay the sending of a message?</td>
</tr>
</tbody>
</table>
What is Camel?
Why?
Aims of Camel

to make integration as simple as it can possibly be
What is Camel?

http://activemq.apache.org/camel/
What is Camel?

Spring based Enterprise Integration Patterns

A selection of some of the patterns...

**Message Routing**

<table>
<thead>
<tr>
<th>Component</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Based Router</td>
<td>How do we handle a situation where the implementation of a single logical function (e.g., inventory check) is spread across multiple physical systems?</td>
</tr>
<tr>
<td>Message Filter</td>
<td>How can a component avoid receiving uninteresting messages?</td>
</tr>
<tr>
<td>Recipient List</td>
<td>How do we route a message to a list of dynamically specified recipients?</td>
</tr>
<tr>
<td>Splitter</td>
<td>How can we process a message if it contains multiple elements, each of which may have to be processed in a different way?</td>
</tr>
<tr>
<td>Aggregator</td>
<td>How do we combine the results of individual, but related messages so that they can be processed as a whole?</td>
</tr>
<tr>
<td>Reseenuencer</td>
<td>How can we get a stream of related but out-of-sequence messages back into the correct order?</td>
</tr>
<tr>
<td>Throttler</td>
<td>How can I throttle messages to ensure that a specific endpoint does not get overloaded, or we don’t exceed an agreed SLA with some external service?</td>
</tr>
<tr>
<td>Delay</td>
<td>How can I delay the sending of a message?</td>
</tr>
</tbody>
</table>
Let's look at a pattern!
Message Filter
Message Filter : XML

```xml
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="activemq:topic:Quotes"/>
    <filter>
      <xpath>/quote/product = 'widget'</xpath>
      <to uri="mqseries:WidgetQuotes"/>
    </filter>
  </route>
</camelContext>
```
Message Filter : Spring XML

```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="http://www.springframework.org/schema/beans
                          http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
                          http://activemq.apache.org/camel/schema/spring
                          http://activemq.apache.org/camel/schema/spring/camel-spring.xsd">

  <camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
    <route>
      <from uri="activemq:topic:Quotes"/>
      <filter>
        <xpath>/quote/product = 'widget'</xpath>
        <to uri="mqseries:WidgetQuotes"/>
      </filter>
    </route>
  </camelContext>

</beans>
```
Message Filter: XML

```
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="activemq:topic:Quotes"/>
    <filter>
      <xpath>/quote/product = 'widget'</xpath>
      <to uri="mqseries:WidgetQuotes"/>
    </filter>
  </route>
</camelContext>
```
# Expressions & Predicates

<table>
<thead>
<tr>
<th>BeanShell</th>
<th>PHP</th>
</tr>
</thead>
<tbody>
<tr>
<td>EL</td>
<td>Python</td>
</tr>
<tr>
<td>Groovy</td>
<td>Ruby</td>
</tr>
<tr>
<td>JavaScript</td>
<td>SQL</td>
</tr>
<tr>
<td>JSR 223</td>
<td>XPath</td>
</tr>
<tr>
<td>OGNL</td>
<td>XQuery</td>
</tr>
</tbody>
</table>
# URIs, Endpoints and Components

http://activemq.apache.org/camel/components.html

<table>
<thead>
<tr>
<th>activemq</th>
<th>ibatis</th>
<th>mail</th>
<th>rmi</th>
<th>udp</th>
</tr>
</thead>
<tbody>
<tr>
<td>activemq.journal</td>
<td>imap</td>
<td>mina</td>
<td>rnc</td>
<td>validation</td>
</tr>
<tr>
<td>bean</td>
<td>irc</td>
<td>mock</td>
<td>rng</td>
<td>velocity</td>
</tr>
<tr>
<td>cxf</td>
<td>jdbc</td>
<td>msv</td>
<td>seda</td>
<td>vm</td>
</tr>
<tr>
<td>direct</td>
<td>jetty</td>
<td>multicast</td>
<td>sftp</td>
<td>xmpp</td>
</tr>
<tr>
<td>event</td>
<td>jbi</td>
<td>pojo</td>
<td>smtp</td>
<td>xquery</td>
</tr>
<tr>
<td>file</td>
<td>jms</td>
<td>pop</td>
<td>string-template</td>
<td>xslt</td>
</tr>
<tr>
<td>ftp</td>
<td>jpa</td>
<td>quartz</td>
<td>timer</td>
<td>webdav</td>
</tr>
<tr>
<td>http</td>
<td>log</td>
<td>queue</td>
<td>tcp</td>
<td></td>
</tr>
</tbody>
</table>
Message Filter : XML

```xml
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
    <route>
        <from uri="activemq:topic:Quotes"/>
        <filter>
            <xpath>/quote/product = 'widget'</xpath>
            <to uri="mqseries:WidgetQuotes"/>
        </filter>
    </route>
</camelContext>
```
Message Filter: Java

from("activemq:topic:Quotes").
  filter().xpath("/quote/product = 'widget'").
  to("mqseries:WidgetQuotes");
package com.acme.quotes;

import org.apache.camel.builder.RouteBuilder;

public class MyRouteBuilder extends RouteBuilder {

    public void configure() {

        // forward widget quotes to MQSeries
        from("activemq:topic:Quotes").
            filter().xpath("/quote/product = 'widget'").
            to("mqseries:WidgetQuotes");
    }
}

Create CamelContext in Java

CamelContext context = new DefaultCamelContext();
context.addRoutes(new MyRouteBuilder());
context.start();
Create CamelContext in Spring

<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <package>com.acme.quotes</package>
</camelContext>
More Patterns!
Content Based Router
Content Based Router

```xml
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <route>
    <from uri="activemq:NewOrders"/>
    <choice>
      <when>
        <xpath>/order/product = 'widget'</xpath>
        <to uri="activemq:Orders.Widgets"/>
      </when>
      <when>
        <xpath>/order/product = 'gadget'</xpath>
        <to uri="activemq:Orders.Gadgets"/>
      </when>
      <otherwise>
        <to uri="activemq:Orders.Bad"/>
      </otherwise>
    </choice>
  </route>
</camelContext>
```
Splitter
from("file://orders").
splitter(body().tokenize("\n")).
to("activemq:Order.Items");
Splitter using XQuery

from("file://orders").splitter().xquery("/order/items").to("activemq:Order.Items");
Aggregator

Inventory Item 1  Inventory Item 2  Inventory Item 3  Aggregator  Inventory Order
from("activemq:Inventory.Items").
aggregator().xpath("/order/@id").
to("activemq:Inventory.Order");
Message Translator

Incoming Message

Translated Message
Message Translator

from("file://incoming").
    to("xslt:com/acme/mytransform.xsl").
    to("http://outgoing.com/foo");
### Message Routing

<table>
<thead>
<tr>
<th>Component</th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content Based Router</td>
<td>How do we handle a situation where the implementation of a single logical function (e.g., inventory check) is spread across multiple physical systems?</td>
</tr>
<tr>
<td>Message Filter</td>
<td>How can a component avoid receiving uninteresting messages?</td>
</tr>
<tr>
<td>Recipient List</td>
<td>How do we route a message to a list of dynamically specified recipients?</td>
</tr>
<tr>
<td>Splitter</td>
<td>How can we process a message if it contains multiple elements, each of which may have to be processed in a different way?</td>
</tr>
<tr>
<td>Aggregator</td>
<td>How do we combine the results of individual, but related messages so that they can be processed as a whole?</td>
</tr>
<tr>
<td>Resequerener</td>
<td>How can we get a stream of related but out-of-sequence messages back into the correct order?</td>
</tr>
<tr>
<td>Throttler</td>
<td>How can I throttle messages to ensure that a specific endpoint does not get overloaded, or we don’t exceed an agreed SLA with some external service?</td>
</tr>
<tr>
<td>Delayer</td>
<td>How can I delay the sending of a message?</td>
</tr>
</tbody>
</table>
Beans
Bean as a Message Translator

```java
from("activemq:Incoming").
  beanRef("myBeanName").
  to("activemq:Outgoing");
```
public class Foo {
    public void someMethod(String name) {
        ...
    }
}
Bean as a Message Translator with method name

```java
from("activemq:Incoming").
  beanRef("myBeanName", "someMethod").
  to("activemq:Outgoing");
```
Type Conversion
Type Conversion

```java
package com.acme.foo.converters;

import org.apache.camel.Converter;
import java.io.*;

@Converter
public class IOConverter {

    @Converter
    public static InputStream toInputStream(File file) throws FileNotFoundException {
        return new BufferedInputStream(new FileInputStream(file));
    }
}

# META-INF/services/org/apache/camel/TypeConverter
com.acme.foo.converters
```
public class Foo {

    @MessageDriven(uri="activemq:cheese")
    public void onCheese(String name) {
        ...
    }
}
Binding Method Arguments

```java
public class Foo {

    public void onCheese(
            @XPath("/foo/bar") String name,
            @Header("JMSCorrelationID") String id) {
            ...
    }
}

for more annotations see
http://activemq.apache.org/camel/bean-integration.html
public class Foo {
    @EndpointInject(uri="activemq:foo.bar")
    ProducerTemplate producer;

    public void doSomething() {
        if (whatever) {
            producer.sendBody("<hello>world!</hello>"ainless);
        }
    }
}
Spring Remoting - Client Side

```xml
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <proxy id="sayService" serviceUrl="activemq:MyService"
    serviceInterface="com.acme.MyServiceInterface"/>
</camelContext>
```
Spring Remoting - Server Side

```xml
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  <export id="sayService" uri="activemq:MyService" serviceRef="sayImpl"
    serviceInterface="com.acme.MyServiceInterface"/>
</camelContext>

<bean id="sayImpl" class="com.acme.MyServiceImpl"/>
```
<camelContext xmlns="http://activemq.apache.org/camel/schema/spring">
  ...
</camelContext>

<bean id="activemq" class="org.apache.camel.component.jms.JmsComponent">
  <property name="connectionFactory">
    <bean class="org.apache.activemq.ActiveMQConnectionFactory">
      <property name="brokerURL" value="vm://localhost?broker.persistent=false"/>
    </bean>
  </property>
</bean>
Data Format

```java
from("activemq:QueueWithJavaObjects").marshal().jaxb().to("mqseries:QueueWithXmlMessages");
```
Business Activity Monitoring (BAM)
public class MyActivities extends ProcessBuilder {

    public void configure() throws Exception {

        // lets define some activities, correlating on an XPath on the message bodies
        ActivityBuilder purchaseOrder = activity("activemq:PurchaseOrders")
            .correlate(xpath("/purchaseOrder/@id").stringResult());

        ActivityBuilder invoice = activity("activemq:Invoices")
            .correlate(xpath("/invoice/@purchaseOrderId").stringResult());

        // now lets add some BAM rules
        invoice.starts().after(purchaseOrder.completes())
            .expectWithin(seconds(1))
            .errorIfOver(seconds(2)).to("activemq:FailedProcesses");
    }
}
Riding the camel
Where would I use Camel?

- standalone or in any Spring application
- inside ActiveMQ’s JMS client or the broker
- inside your ESB such as ServiceMix via the servicemix-camel Service Unit
- inside CXF either as a transport or reusing CXF inside Camel
Camel Riding from Java

- `/META-INF/spring/camelContext.xml`
- set the CLASSPATH
- `java org.apache.camel.spring.Main`
Maven Tooling

```xml
<project>
...
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.camel</groupId>
                <artifactId>camel-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
    <reporting>
        <plugins>
            <plugin>
                <groupId>org.apache.camel</groupId>
                <artifactId>camel-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </reporting>
</project>

mvn camel:run
Maven Plugin Site Report

org.apache.camel.example.docs.ContentBasedRoute

seda:cbr.input

XPath: /person/city = 'London'

Otherwise

seda:cbr.output.a, seda:cbr.output.b
Where do I get more info?

please do take Camel for a ride!

http://activemq.apache.org/camel/

don’t get the hump! :-)

![Apache Camel 1.0 Box](image)
Questions?
James Strachan

blog

http://macstrac.blogspot.com/
Camel v Mule

- A Camel can carry 4 times as much load as other beasts of burden!

http://activemq.apache.org/camel/why-the-name-camel.html
Camel v Mule

• Camel provides a higher level abstraction for EIP; making it simpler & easier & less XML

• Awesome integration with JMS + ActiveMQ (client and broker), JBI + ServiceMix and JAX-WS + CXF

• Great testing with Camel’s Mock Endpoint

• Business Activity Monitoring framework
Where do I get more info?

please do take Camel for a ride!

http://activemq.apache.org/camel/

don’t get the hump! :-}