Developing, Testing and Scaling with Apache Camel

Matt Raible • http://raibledesigns.com

Photos by McGinity Photo

Apache Camel™

RAIBLE DESIGNS
ENTERPRISE OPEN SOURCE CONSULTING

Who is **Matt Raible**?

Father, Skier, Mountain Biker, Whitewater Rafter

Web Framework Connoisseur

Founder of AppFuse

Bus Lover

Blogger on raibledesigns.com

What about **YOU**?

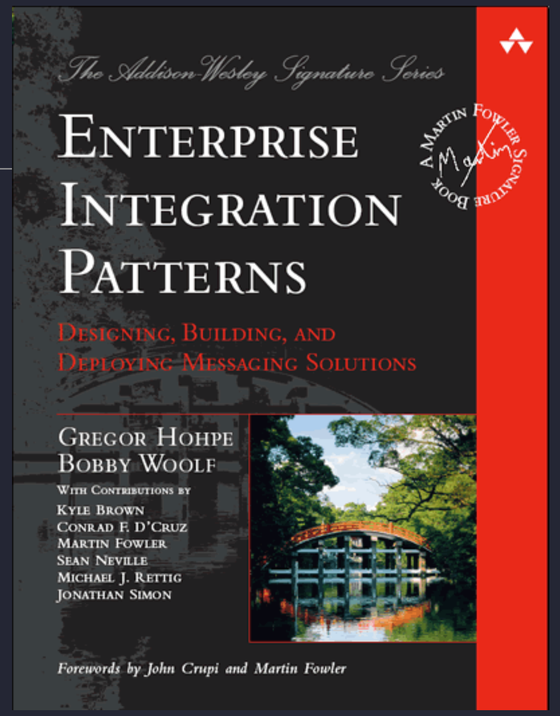
Are you familiar with Enterprise Integration Patterns?

Have you used Apache Camel?

What about Spring Integration?

XML or JavaConfig?

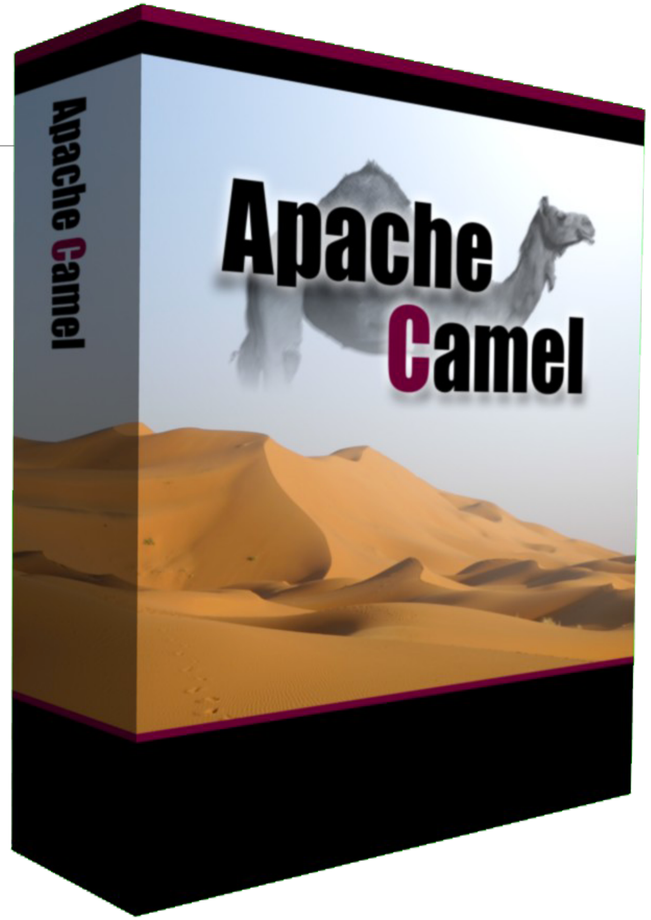
How do you feel about testing?



Do you want to take a Camel ride?

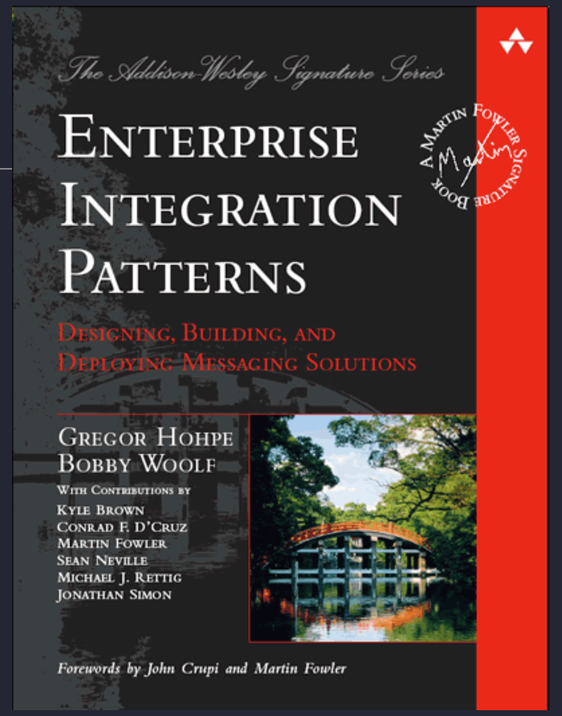
What is Apache Camel?

Apache Camel is a versatile open-source integration framework based on known Enterprise Integration Patterns.



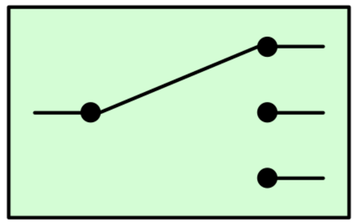
Enterprise Integration

- Why is it necessary?
  - Businesses need to integration different systems
- Why a framework?
  - To allow you to focus on the business logic
  - Others have done it before, don’t reinvent the wheel
  - Simplified testing

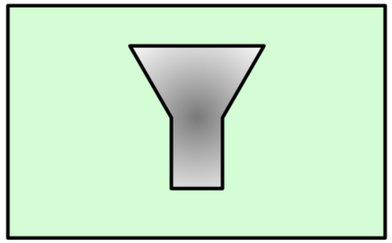


Camel EIPs

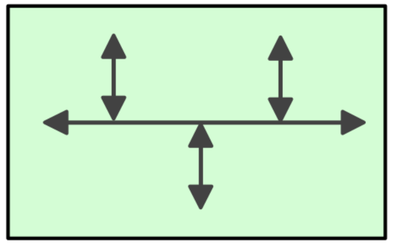
http://camel.apache.org/eip



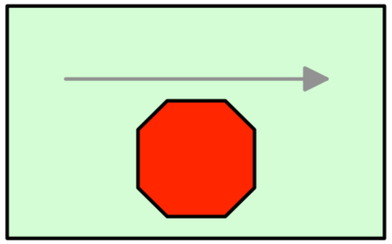
Content Based Router



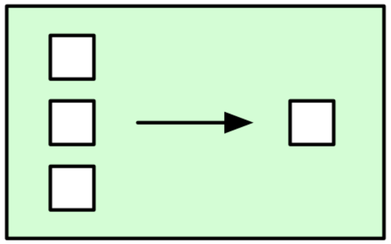
Message FIlter



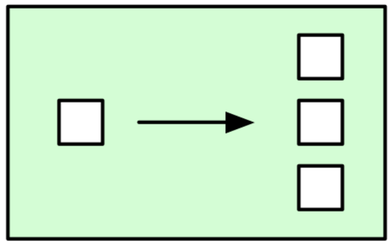
Message Bus



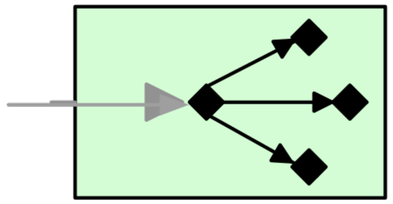
Dead Letter Channel



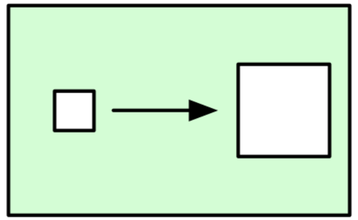
Aggregator



Splitter

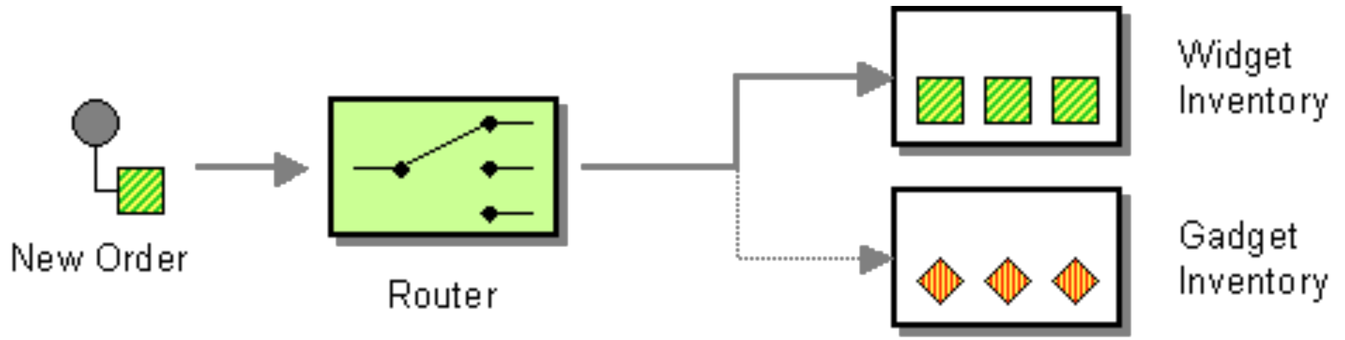


Message Dispatcher



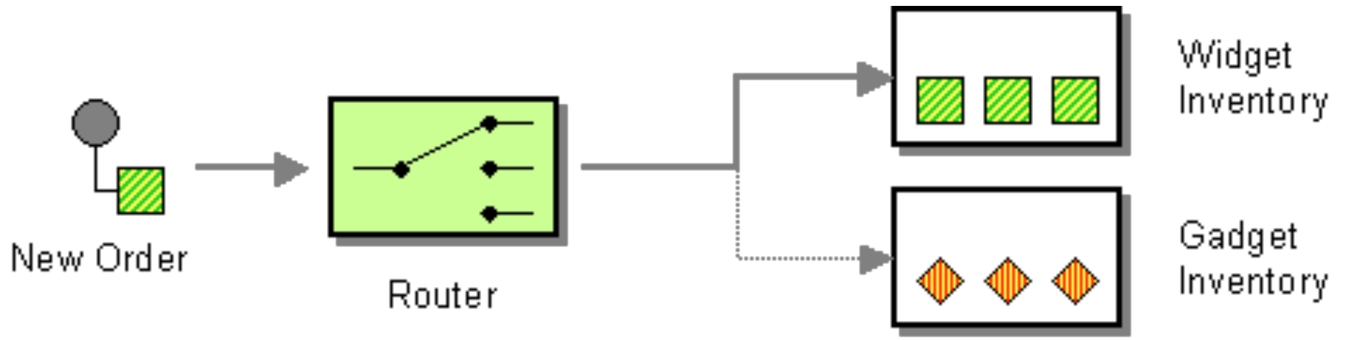
Content Enricher

Content Based Router



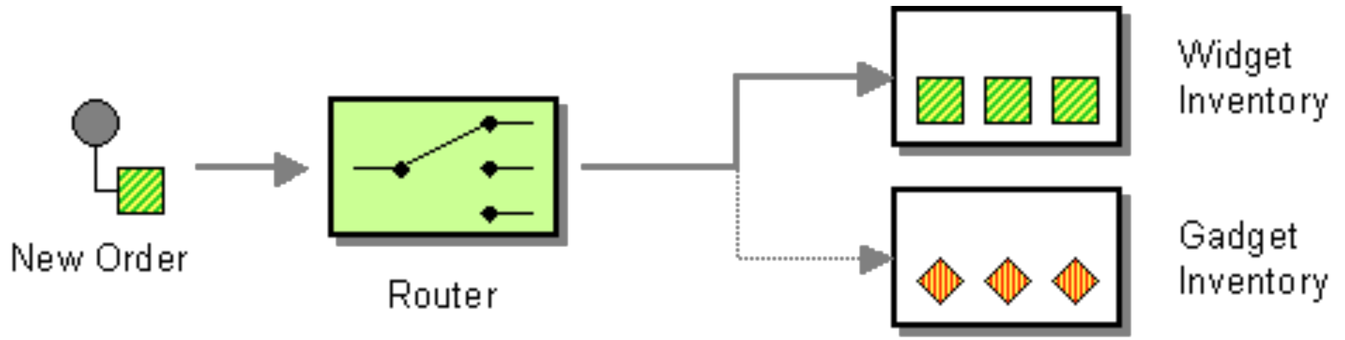

```
from newOrder
    choice
        when isWidget to widget
        otherwise to gadget
```

Content Based Router



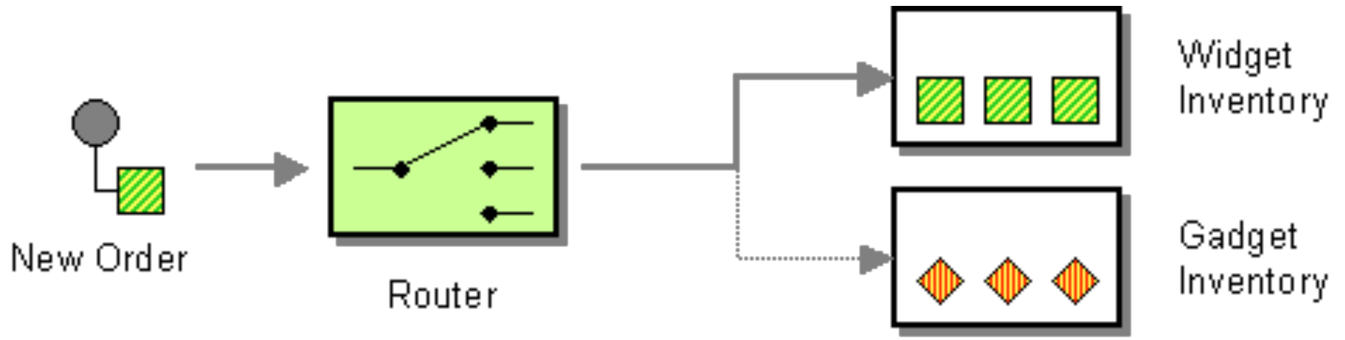

```
from(newOrder)
    choice()
        when(isWidget) to(widget)
        otherwise to(gadget)
```

Content Based Router




```
from(newOrder)
    .choice()
        .when(isWidget).to(widget)
        .otherwise().to(gadget);
```

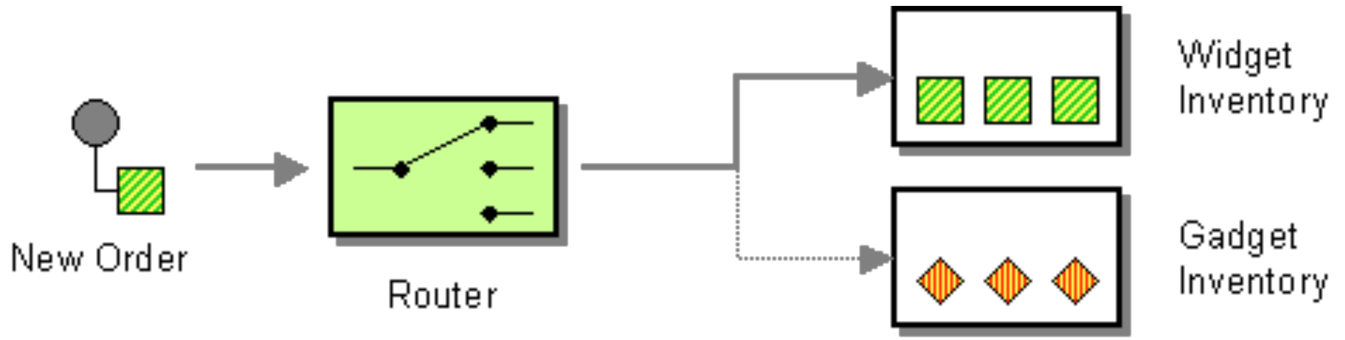
Content Based Router




```
Endpoint newOrder = endpoint(“activemq:queue:newOrder");
```

```
from(newOrder)
    .choice()
        .when(isWidget).to(widget)
        .otherwise().to(gadget);
```

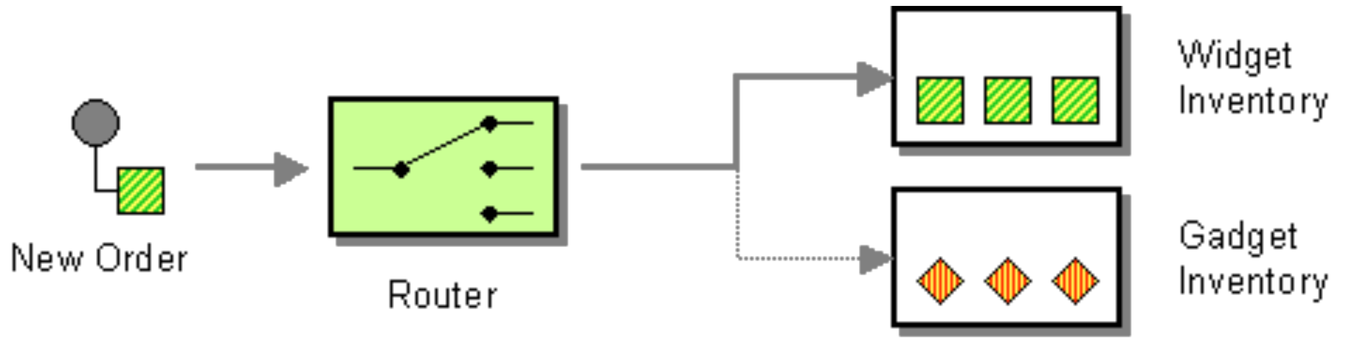
Content Based Router




```
Endpoint newOrder = endpoint("activemq:queue:newOrder");
Predicate isWidget = xpath("/order/product = 'widget");
```

```
from(newOrder)
    .choice()
        .when(isWidget).to(widget)
        .otherwise().to(gadget);
```

Content Based Router




```
Endpoint newOrder = endpoint("activemq:queue:newOrder");
Predicate isWidget = xpath("/order/product = 'widget");
Endpoint widget = endpoint("activemq:queue:widget");
Endpoint gadget = endpoint("activemq:queue:gadget");

from(newOrder)
    .choice()
        .when(isWidget).to(widget)
        .otherwise().to(gadget);
```

Content Based Router

```
import org.apache.camel.Endpoint;
import org.apache.camel.Predicate;
import org.apache.camel.builder.RouteBuilder;

public class MyRoute extends RouteBuilder {

    public void configure() throws Exception {
        Endpoint newOrder = endpoint("activemq:queue:newOrder");
        Predicate isWidget = xpath("/order/product = 'widget");
        Endpoint widget = endpoint("activemq:queue:widget");
        Endpoint gadget = endpoint("activemq:queue:gadget");

        from(newOrder)
            .choice()
                .when(isWidget).to(widget)
                .otherwise().to(gadget)
            .end();
        }
    }
}
```

Content Based Router: Java DSL

```
import org.apache.camel.builder.RouteBuilder;

public class MyRoute extends RouteBuilder {

    public void configure() throws Exception {
        from("activemq:queue:newOrder")
            .choice()
                .when(xpath("/order/product = 'widget"))
                    .to("activemq:queue:widget")
                .otherwise()
                    .to("activemq:queue:gadget")
            .end();
        }
    }
}
```

Content Based Router: XML DSL

```
<route>
    <from uri="activemq:queue:newOrder"/>
    <choice>
        <when>
            <xpath>/order/product = 'widget'</xpath>
            <to uri="activemq:queue:widget"/>
        </when>
        <otherwise>
            <to uri="activemq:queue:gadget"/>
        </otherwise>
    </choice>
</route>
```

Spring XML

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
    <!-- read input from the console using the stream component -->
    <from uri="stream:in?promptMessage=Enter something: "/>
    <!-- transform the input to upper case using the simple language -->
    <transform>
      <simple>${body.toUpperCase()}</simple>
    </transform>
    <!-- and then print to the console -->
    <to uri="stream:out"/>
  </route>
</camelContext>
```

public static void main

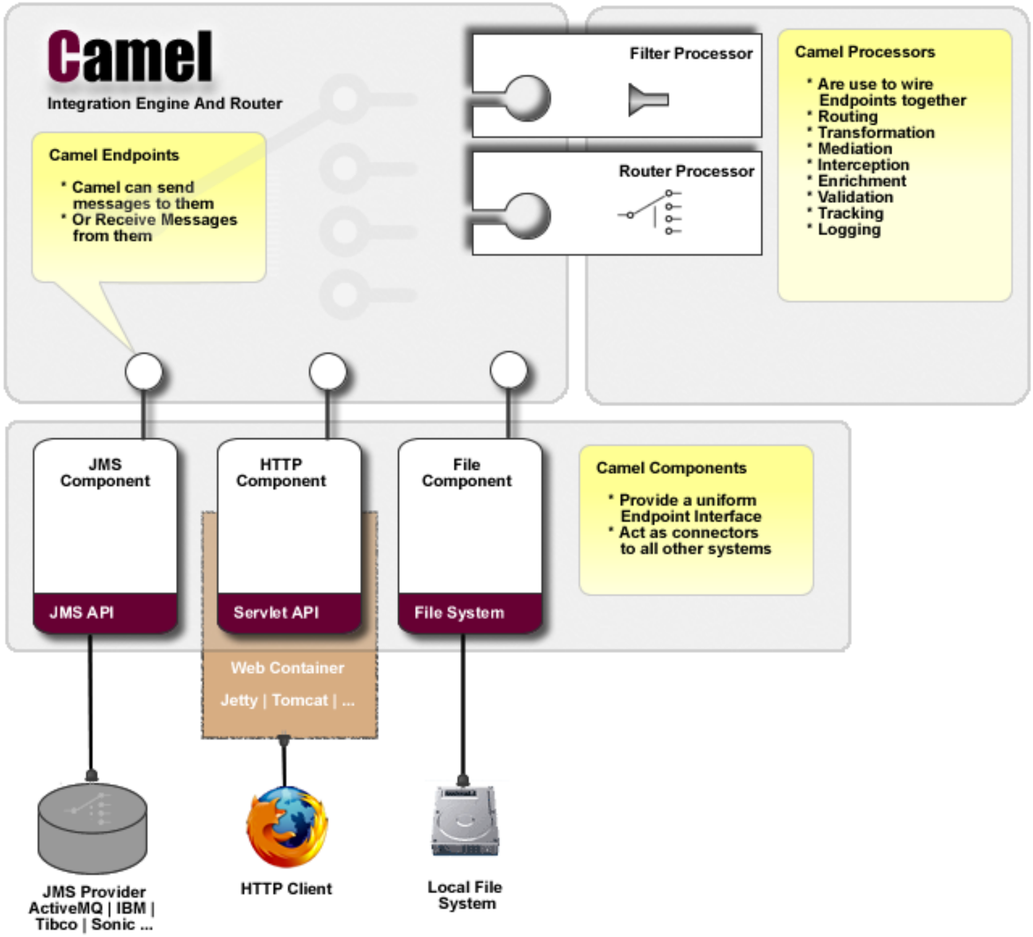
```java
import org.apache.camel.spring.Main;

/**
 * A main class to run the example from your editor.
 */
public final class CamelConsoleMain {

    private CamelConsoleMain() {}

    public static void main(String[] args) throws Exception {
        // Main makes it easier to run a Spring application
        Main main = new Main();
        // configure the location of the Spring XML file
        main.setApplicationContextUri("META-INF/spring/camel-context.xml");
        // enable hangup support allows Camel to detect when the JVM is terminated
        main.enableHangupSupport();
        // run and block until Camel is stopped (or JVM terminated)
        main.run();
    }
}
```

Camel Architecture



Camel Components

Task Automation: Timer, Quartz2

Amazon: AWS-CW, AWS-DDB, AWS-SES, AWS-S3

Basics: Bean, Class, Context, Data Format, Exec, Language, Printer

Chat: IRC, XMPP

Endpoints: Direct, Direct-VM, Disruptor, SEDA

ESB: JBI, NMR, Vert.x

JMS: ActiveMQ, JMS, RabbitMQ, Stomp, ZeroMQ

Apache Camel Components

Automating Tasks

Component / ArtifactId / URI	Description
Timer / camel-core in org.apache.activemq timer:timerName[?options]	Used to generate message exchanges when a timer fires. Events can only be consumed from this endpoint.
Quartz / camel-quartz quartz://groupName/timerName[?options]	Provides a scheduled delivery of messages using the Quartz 1.x scheduler.
Quartz2 / camel-quartz2 quartz2://groupName/timerName[?options]	Provides a scheduled delivery of messages using the Quartz 2.x scheduler.

Amazon

Component / ArtifactId / URI	Description
AWS-CW / camel-aws aws-cw://clientName[?options]	For working with Amazon's CloudWatch (CW).
AWS-DDB / camel-aws aws-ddb://tableName[?options]	For working with Amazon's DynamoDB (DDB).
AWS-SDB / camel-aws aws-sdb://domainName[?options]	For working with Amazon's SimpleDB (SDB).
AWS-SES / camel-aws aws-ses://from[?options]	For working with Amazon's Simple Email Service (SES).
AWS-SNS / camel-aws aws-sns://topicName[?options]	For Messaging with Amazon's Simple Notification Service (SNS).
AWS-SQS / camel-aws aws-sqs://queueName[?options]	For Messaging with Amazon's Simple Queue Service (SQS).
AWS-S3 / camel-aws aws-s3://bucketName[?options]	For working with Amazon's Simple Storage Service (S3).

Basics

Component / ArtifactId / URI	Description
Bean / camel-core bean:beanName[?options]	Uses the Bean Binding to bind message exchanges to beans in the Registry. Is also used for exposing and invoking Plain Old Java Objects (POJOs).
Class / camel-core class:className[?options]	Uses the Bean Binding to bind message exchanges to beans based on the class name. Is also used for exposing and invoking POJOs.
Context / camel-context context:camelContextId:localEndpointName[?options]	Used to refer to endpoints within a separate CamelContext to provide a simple black box composition approach so that routes can be combined into a CamelContext and then used as a black box component inside other routes in other CamelContext.
Data Format / camel-core dataformat:name:(marshal|unmarshal)[?options]	For working with Data Formats as if it was a regular Component supporting Endpoints and URIs.
Exec / camel-exec exec://executable[?options]	For executing system commands.
Language / camel-core language://languageName[:script][?options]	Executes Languages scripts.
Printer / camel-printer lpr://hostName[:port]/printerPath[?options]	Facilitates creation of printer endpoints to local, remote and wireless printers. The endpoints provide the ability to print camel directed payloads when utilized on camel routes.
Properties / camel-core properties://key[?options]	Facilitates using property placeholders directly in endpoint URI definitions.
Ref / camel-core ref:name	Component for lookup of existing endpoints bound in the Registry.

Chat

Component / ArtifactId / URI	Description
IRC / camel-irc irc:[login@]hostName[:port]/#room[?options]	For Internet Relay Chat (IRC) communication.
XMPP / camel-xmpp xmpp://[login@]hostName[:port][/participant][?options]	Working with the Extensible Messaging and Presence Protocol (XMPP).

Clusters

Component / ArtifactId / URI	Description
JGroups / camel-jgroups in camel-extra jgroups:clusterName[?options]	Provides exchange of messages between Camel infrastructure and JGroups clusters.
ZooKeeper / camel-zookeeper zookeeper://zookeeperServer[:port]/path[?options]	Working with ZooKeeper clusters.

Content Repositories

Component / ArtifactId / URI	Description
CMIS / camel-cmis cmis://cmisServerUrl[?options]	Uses the Apache Chemistry client API to interface with Content Management Interoperability Services (CMIS).
JCR / camel-jcr jcr://user:password@repository/path/to/node[?options]	Storing a message in a Java Content Repository (JCR) compliant repository like Apache Jackrabbit.

Endpoint Communications

Component / ArtifactId / URI	Description
Direct / camel-core direct:someName[?options]	Synchronous call to another endpoint from same CamelContext.
Direct-VM / camel-core direct-vm:someName[?options]	Synchronous call to another endpoint in another CamelContext running in the same Java virtual machine (JVM).
Disruptor / camel-disruptor disruptor:someName[?options]	Provides asynchronous SEDA behavior much as the standard SEDA Component, but utilizes a Disruptor instead of a BlockingQueue.
Disruptor VM / camel-disruptor disruptor-vm:someName[?options]	Same as Disruptor, but the buffers of the Disruptor VM endpoints also provide support for communication across CamelContexts instances so you can use this mechanism to communicate across web applications.
SEDA / camel-core seda:someName[?options]	Asynchronous call to another endpoint in the same Camel Context, staged event-driven architecture (SEDA).
VM / camel-core vm:queueName[?options]	Asynchronous call to another endpoint in the same JVM.

ESB

Component / ArtifactId / URI	Description
JBI / servicemix-camel in org.apache.servicemix jbi:serviceName[?options]	For Java Business Integration (JBI) integration such as working with Apache ServiceMix.
NMR / apache-servicemix-nmr in org.apache.servicemix.nmr nmr://serviceName[?options]	Integration with the Normalized Message Router bus in Apache ServiceMix.
Vert.x / camel-vertx vertx:eventBusName	Working with the Vert.x event bus.

Feeds

Component / ArtifactId / URI	Description
Atom / camel-atom atom:atomUri[?options]	Working with Apache Abdera for atom integration, such as consuming an atom feed.
RSS / camel-rss rss:uri[?options]	Working with ROME for Rich Site Summary (RSS) integration.

File I/O and Transfer

Component / ArtifactId / URI	Description
File / camel-core file://nameOfFileOrDirectory[?options]	Sending messages to a file or polling a file or directory.
Flatpack / camel-flatpack flatpack:[fixed|delim]:configFileName[?options]	Processing fixed width or delimited files or messages using the FlatPack library.
FTP / camel-ftp ftp:contextPath[?options]	Sending and receiving files over File Transfer Protocol (FTP).
FTPS / camel-ftp ftps://[username@]hostName[:port]/directoryName[?options]	Sending and receiving files over FTP Secure (TLS and SSL).
HDFS / camel-hdfs hdfs://hostName[:port]/path[?options]	For reading/writing from/to an Hadoop Distributed File System (HDFS) filesystem.
SCP / camel-jsch scp://hostName[:port]/destination[?options]	Support for the scp protocol using the Client API of the Java Secure Channel (JSch) project.
SFTP / camel-ftp sftp://[username@]hostName[:port]/directoryName[?options]	Sending and receiving files over FTP Secure (FTP and SSH).
Stream / camel-stream stream:(in|out|err|file|header|url)[?options]	Read or write to an input/output/error/file stream rather like unix pipes.

Google

Component / ArtifactId / URI	Description
GAuth / camel-gae gauth:contextPath[?options]	Used by web applications to implement an OAuth consumer.
GHttp / camel-gae ghttp:contextPath[?options]	Provides connectivity to the URL fetch service of Google App Engine but can also be used to receive messages from servlets.
GLogin / camel-gae glogin://hostName[:port][?options]	Used by Camel applications outside Google App Engine (GAE) for programmatic login to GAE applications.
GTask / camel-gae gtask://queueName[?options]	Supports asynchronous message processing on Google App Engine by using the task queuing service as message queue.
GMail / camel-gae gmail://user@gleegle3mail-com[?options]	Supports sending of emails via the mail service of Google App Engine.
Guava EventBus / camel-guava-eventbus guava-eventbus:busName[?options]	The Google Guava EventBus allows publish-subscribe style communication between components without requiring the components to explicitly register with one another.

HTTP

Component / ArtifactId / URI	Description
AHC / camel-ahc ahc:http[s]://hostName[:port][/resourceUri][?options]	To call external HTTP services using Async Http Client.
Cometd / camel-cometd cometd://hostName:port/channelName[?options]	HTTP-based event routing bus used to deliver messages using the Jetty Cometd implementation of the Bayeux Protocol.
HTTP / camel-http http:hostName[:port][/resourceUri][?options]	For calling out to external HTTP servers using Apache HTTP Client 3.x.
HTTP4 / camel-http4 http4://hostName[:port][/resourceUri][?options]	For calling out to external HTTP servers using Apache HTTP Client 4.x.
Jetty / camel-jetty jetty:hostName[:port][/resourceUri][?options]	For exposing services over HTTP.
Servlet / camel-servlet servlet:relativePath[?options]	For exposing services over HTTP through the Servlet which is deployed into the Web container.

Java Message Service

Component / ArtifactId / URI	Description
ActiveMQ / activemq-camel in org.apache.activemq activemq:[topic:]destinationName	For JMS Messaging with Apache ActiveMQ.
ActiveMQ Journal / activemq-core activemq-journal:directoryName[?options]	Uses ActiveMQ's fast disk journaling implementation to store message bodies in a rolling log file.
JMS / camel-jms jms:[queue:|topic:]destinationName[?options]	Working with JMS provider.
RabbitMQ / camel-rabbitmq rabbitmq://hostName[:port]/exchangeName[?options]	For JMS Messaging with RabbitMQ.
SJMS / camel-sjms sjms:[queue:|topic:]destinationName[?options]	From the ground upwards implementation of a JMS client.
Stomp / camel-stomp stomp:queue:destinationName[?options]	For communicating with Stomp compliant message brokers, like Apache ActiveMQ or ActiveMQ Apollo.
ZeroMQ / camel-zeromq in camel-extra. zeromq:(tcp|ipc)://hostName:port[?options]	For JMS Messaging with ZeroMQ.

LDAP

Component / ArtifactId / URI	Description
LDAP / camel-ldap ldap:hostName[:port][?options]	Performing searches on Lightweight Directory Access Protocol (LDAP) servers.
Spring LDAP / camel-spring-ldap spring-ldap:springLdapTemplateBeanName[?options]	Camel wrapper for Spring LDAP. Spring LDAP is a Java library for simplifying LDAP operations, based on the pattern of Spring's JdbcTemplate.

Mail

Component / ArtifactId / URI	Description
IMAP / camel-mail imap://[username@]hostName[:port][?options]	Receiving email using Internet Message Access Protocol (IMAP).
IMAPS / camel-mail imaps://[username@]hostName[:port][?options]	Receiving email using secured IMAP.
POP3 / camel-mail pop3://[username@]hostName[:port][?options]	Receiving email using Post Office Protocol (POP3) and JavaMail.
POP3S / camel-mail pop3s://[username@]hostName[:port][?options]	Receiving email using secured POP3 and JavaMail.
SMTP / camel-mail smtp://[username@]hostName[:port][?options]	Sending email using Simple Mail Transfer Protocol (SMTP) and JavaMail.
SMTPS / camel-mail smtps://[username@]hostName[:port][?options]	Sending email using secured SMTP and JavaMail.

Maintenance and Monitoring

Component / ArtifactId / URI	Description
Browse / camel-core browse:someName	Provides a simple BrowsableEndpoint which can be useful for testing, visualisation tools or debugging. The exchanges sent to the endpoint are all available to be browsed.
ControlBus / camel-core controlbus:command[?options]	ControlBus EIP that allows to send messages to Endpoints for managing and monitoring your Camel applications.
JMX / camel-jmx jmx://platform[?options]	For working with Java Management Extensions (JMX) notification listeners.
Log / camel-core log:loggingCategory[?options]	Uses Jakarta Commons Logging to log the message exchange to some underlying logging system like log4j.
Nagios / camel-nagios nagios://hostName[:port][?options]	Sending passive checks to Nagios using JSendNSCA. Nagios supports IT Infrastructure Monitoring.
SNMP / camel-snmp snmp://hostName[:port][?options]	Polling OID values and receiving traps using SNMP via SNMP4J library.

Messaging

Component / ArtifactId / URI	Description
AMQP / camel-amqp amqp:[queue:|topic:]destinationName[?options]	For Messaging with the Advanced Message Queuing Protocol (AMQP).
Esper / camel-esper in camel-extra esper:name[?options]	Working with the Esper Library for Event Stream Processing.
JavaSpace / camel-javaspace javaspace:jini://hostName[?options]	Sending and receiving messages through JavaSpace.
JT/400 / camel-jt400 jt400://user:pwd@system/path_to_dtaq[?options]	For integrating with data queues on an AS/400 (i.e., System i, IBM i, i5, etc.) systems.
Kestrel / camel-kestrel kestrel://[address list/]queueName[?options]	For producing to or consuming from Kestrel queues.
MQTT / camel-mqtt mqtt:name[?options]	Component for communicating with MQ Telemetry Transport (MQTT) machine-to machine (M2M) message brokers.
SIP / camel-sip sip://user@hostName[:port][?options]	Publish/Subscribe communication capability using the Session Initiation Protocol (SIP) protocol.
SIPS / camel-sip sips://user@hostName[:port][?options]	Publish/Subscribe communication capability using the secured Session Initiation Protocol (SIP) protocol.
SMPP / camel-smpp smpp://[username@]hostName[:port][?options]	To send and receive SMS using Short Messaging Service Center using the JSMPP library.
SMPPS / camel-smpp smpps://[username@]hostName[:port][?options]	To send and receive secured SMS using Short Messaging Service Center using the JSMPP library.
Quickfix / camel-quickfix quickfix:configFileName[?options]	Implementation of the QuickFix for Java engine which allows to send/receive FIX messages.

Networking

Component / ArtifactId / URI	Description
DNS / camel-dns dns:operation[?options]	To lookup domain information and run Domain Name System (DNS) queries using DNS.Java.
HL7 / camel-hl7 mina2:tcp://hostName[:port][?options]	For working with the HL7 MLLP protocol and the HL7 model using the HAPI library.
MINA / camel-mina mina:(tcp|udp|vm):hostName[:port][?options]	Working with Apache Multipurpose Infrastructure or Network Applications (MINA) 2.x. Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Virtual Machine (VM) protocol are supported.
MINA2 / camel-mina2 mina2:(tcp|udp|vm):hostName[:port][?options]	Working with Apache MINA 2.x. Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Virtual Machine (VM) protocol are supported.
Netty / camel-netty netty:(tcp|udp):hostName[:port][?options]	Working with TCP and UDP protocols using Java NIO based capabilities offered by the Netty project.
Netty HTTP / camel-netty-http netty-http:http:[port]/contextPath[?options]	Netty HTTP server and client using the Netty project.

OSGi

Component / ArtifactId / URI	Description
EventAdmin / camel-eventadmin eventadmin:topic[?options]	Receiving OSGi EventAdmin events.
Pax Logging / camel-paxlogging paxlogging:appender	Receiving Pax-Logging events in OSGi.

Persistence

Component / ArtifactId / URI	Description
CouchDB / camel-couchdb couchdb:hostName[:port]/database[?options]	To integrate with Apache CouchDB.
DB4o / camel-db4o in camel-extra db4o://className[?options]	For using a db4o datastore as a queue via the db4o [database for objects] library.
EJB / camel-ejb ejb:ejbName[?options]	Uses the Bean Binding to bind message exchanges to Enterprise Java Beans (EJBs). It works like the Bean component but just for accessing EJBs. Supports EJB 3.0 onwards.
HBase / camel-hbase hbase://table[?options]	For reading/writing from/to an HBase store (Hadoop database).
Hibernate / camel-hibernate in camel-extra hibernate://entityName[?options]	For using a database as a queue via the Hibernate library.
iBATIS / camel-ibatis ibatis://statementName[?options]	Performs a query, poll, insert, update or delete in a relational database using Apache iBATIS.
JDBC / camel-jdbc jdbc:dataSourceName[?options]	For performing Java Database Connectivity (JDBC) queries and operations.
JPA / camel-jpa jpa://entityName[?options]	For using a database as a queue via the Java Persistence API (JPA) specification for working with OpenJPA, Hibernate or TopLink.
Krati / camel-krati krati://datastorePath[?options]	For producing to or consuming to Krati datastores.
MongoDB / camel-mongodb mongodb:connectionBean[?options]	Interacts with MongoDB databases and collections. Offers producer endpoints to perform CRUD-style operations and more against databases and collections, as well as consumer endpoints to listen on collections and dispatch objects to Camel routes.
MyBatis / camel-mybatis mybatis://statementName[?options]	Performs a query, poll, insert, update or delete in a relational database using MyBatis.
SQL / camel-sql sql:queryCommand[?options]	Performing Structured Query Language (SQL) queries using JDBC.
Spring Neo4j / camel-spring-neo4j in camel-extra spring-neo4j:http://hostName[:port]/database[?options]	Component for producing to Neo4j datastore using the Spring Data Neo4j library.

Platform Support

Component / ArtifactId / URI	Description
Activiti / activiti-camel activiti:camelProcess:serviceTask	For working with Activiti, a light-weight workflow and Business Process Management (BPM) platform which supports BPMN 2.
APNS / camel-apns apns:(notify|consumer)[?options]	For sending notifications to Apple iOS devices.
Salesforce / camel-salesforce salesforce:topic[?options]	To integrate with Salesforce.
SAP NetWeaver / camel-sap-netweaver sap-netweaver:hostName[:port][?options]	To integrate with SAP NetWeaver Gateway.

Remote Services

Component / ArtifactId / URI	Description
Avro / camel-avro avro:(transport):(hostName):(port)/messageName[?options]	Working with Apache Avro for data serialization.
JClouds / camel-jclouds jclouds:(blobstore|compute):(provider id)[?options]	For interacting with cloud compute & blobstore service via JClouds.
JCIFS / camel-jcifs in camel-extra smb://[username@hostName/share#name[?options]	This component provides access to remote file systems over the CIFS/SMB networking protocol by using the JCIFS library.
RMI / camel-rmi rmi://hostName[:port]#service[?options]	Working with Remote Method Invocation (RMI).
SSH component / camel-ssh ssh://username[:password]@hostName[:port][?options]	For sending commands to a SSH server.

Search Engines

Component / ArtifactId / URI	Description
ElasticSearch / camel-elasticsearch elasticsearch://clusterName[?options]	For interfacing with an ElasticSearch server.
Lucene / camel-lucene lucene:searcherName:(insert|query)[?options]	Uses Apache Lucene to perform Java-based indexing and full text based searches using advanced analysis/tokenization capabilities.
Solr / camel-solr solr://hostName[:port]/solr[?options]	Uses the Solrj client API to interface with an Apache Lucene Solr server.

Security

Component / ArtifactId / URI	Description
Crypto (Digital Signatures) / camel-crypto crypto:(sign|verify):name[?options]	Used to sign and verify exchanges using the Signature Service of the Java Cryptographic Extension.
XML Security / camel-xmlsecurity xmlsecurity:(sign|verify):name[?options]	Used to sign and verify exchanges using the XML signature specification.

Social Media

Component / ArtifactId / URI	Description
Facebook / camel-facebook facebook://endpointE[?options]	Providing access to all of the Facebook APIs accessible using Facebook4J.
Twitter / camel-twitter twitter://endpointE[?options]	Enables the most useful features of the Twitter API by encapsulating Twitter4J.
Yammer / camel-yammer yammer://functionE[?options]	Allows you to interact with the Yammer enterprise social network.

Special support

Component / ArtifactId / URI	Description
Cache / camel-cache cache://cacheName[?options]	The cache component facilitates creation of caching endpoints and processors using EHCache as the cache implementation.
Geocoder / camel-geocoder geocoder:(address|latlng:latitude-longitude)[?options]	Supports looking up geocoders for an address, or reverse lookup geocoders from an address.
Hazelcast / camel-hazelcast hazelcast://(type)?cacheName[?options]	Hazelcast is a data grid supporting map, multimap, SEDA, queue, set, atomic number and simple clusters.
RCode / camel-rcode in camel-extra rcode://hostName[:port]/operation[?options]	Uses Rserve to integrate Camel with the statistics environment R.
Routebox / camel-routebox routebox:routeBoxName[?options]	Facilitates the creation of specialized endpoints that offer encapsulation and a strategy/map based indirection service to a collection of camel routes hosted in an automatically created or user injected camel context.
Weather / camel-weather weather://name[?options]	Polls the weather information from Open Weather Map.

Spring

Component / ArtifactId / URI	Description
Spring Event / camel-spring spring-event://default	Working with Spring Application Events.
Spring Batch / camel-spring-batch spring-batch://jobName[?options]	To bridge Camel and Spring Batch.
Spring Integration / camel-spring-integration spring-integration:defaultChannelName[?options]	The bridge component of Camel and Spring Integration.
Spring Redis / camel-spring-redis spring-redis://hostName:port[?options]	Component for consuming and producing from a Redis key-value store.

Templates

Component / ArtifactId / URI	Description
FreeMarker / camel-freemarker freemarker:templateName[?options]	Generates a response using a FreeMarker template.
Mustache / camel-mustache mustache:templateName[?options]	Generates a response using a Mustache template.
MVEL / camel-mvel mvel:templateName[?options]	Generates a response using an MVFLEX Expression Language (MVEL) template.
Scalate / scalate-camel in org.fusesource.scalate scalate:templateName	Generates a response using a Scalate template.
StringTemplate / camel-stringtemplate string-template:templateName[?options]	Generates a response using a String Template.

Testing

Component / ArtifactId / URI	Description
DataSet / camel-core dataset:name[?options]	For load and soak testing, the DataSet provides a way to create huge numbers of messages for sending to components or asserting that they are consumed correctly.
Mock / camel-core mock:name[?options]	For testing routes and mediation rules using mocks.
Stub / camel-core stub:someOtherCamelUri[?options]	Allows you to stub out some physical middleware endpoint for easier testing or debugging.
Test / camel-spring test:expectedMessagesEndpointUri[?options]	Creates a Mock endpoint which expects to receive all the message bodies that could be polled from the given underlying endpoint.

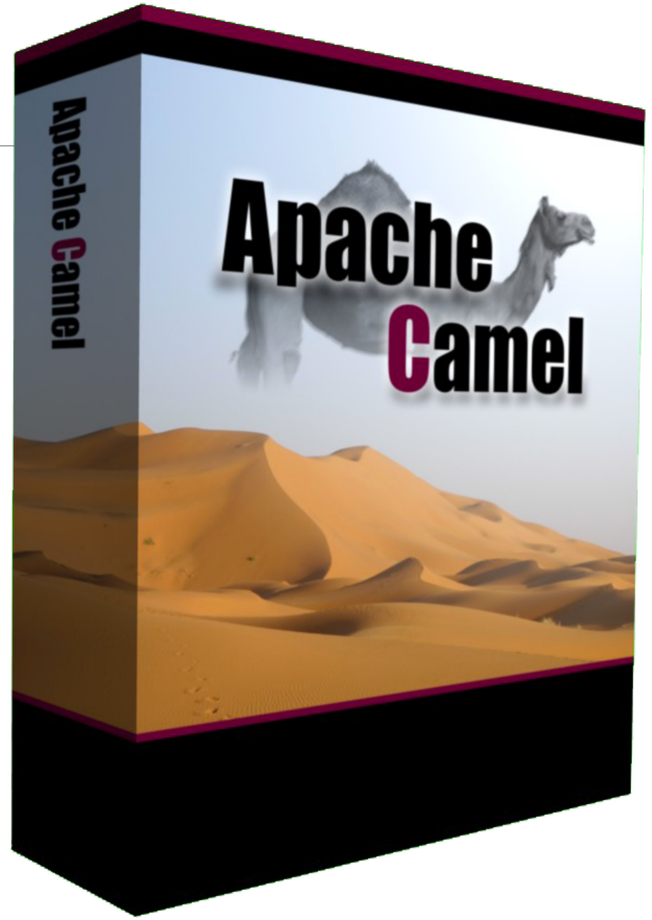
Web Services

Component / ArtifactId / URI	Description
CXF / camel-cxf cxf:bean:cxfEndpoint|//someAddress[?options]	Working with Apache CXF for web services integration.
CXF Bean / camel-cxf cxf:bean:serviceBeanRef[?options]	Processes the exchange using a JAX-WS or JAX-RS annotated bean from the registry.
CXFRS / camel-cxf cxfrs:bean:rsEndpoint|//someAddress[?options]	Working with Apache CXF for RESTful services integration.
Restlet / camel-restlet restlet:restletUri[?options]	Component for consuming and producing Restful resources using Restlet.
Spring Web Services / camel-spring-ws spring-ws:(mapping-type):someAddress[?options]	Client-side support for accessing web services, and server-side support for creating your own contract-first web services using Spring Web Services.
Websocket / camel-websocket websocket://hostName[:port][/resourceUri][?options]	Communicating with Websocket clients.

XML

Component / ArtifactId / URI	Description
Bean Validation / camel-bean-validator bean-validator:something[?options]	Validates the payload of a message using the Java Validation API (JSR 303). Hibernate Validator is the reference implementation.
FOP / camel-fop fop:outputFormat[?options]	Renders the message into different output formats using Formatting Objects Processor (FOP), which is driven by XSL formatting objects (XSL-FO).
MSV / camel-msv msv:someLocalOrRemoteResource[?options]	Validates the payload of a message using the MSV Library.
RNC / camel-jing rnc://relativeOrAbsoluteUri[?options]	Validates the payload of a message using RelaxNG Compact Syntax.
RNG / camel-jing rng://relativeOrAbsoluteUri[?options]	Validates the payload of a message using RelaxNG.
StAX / camel-stax stax:(contentHandlerClassName)[#myHandler]	Process messages through a Simple API for XML (SAX) ContentHandler.
XQuery / camel-saxon xquery:some|queryResource	Generates a response using an XQuery template.
XSLT / camel-core xslt:templateName[?options]	Generates a response using a Extensible Stylesheet Language Transformations (XSLT) template.
Validation / camel-core validation:someLocalOrRemoteResource[?options]	Validates the payload of a message using XML Schema and Java API for XML Processing (JAXP) Validation.

What is Apache Camel?



An integration framework

Supports Enterprise Integration Patterns (EIP)

Routing (many DSLs to choose from)

Easy Configuration (endpoints as URIs)

Just Java, XML, Scala, etc.

No Container Dependency

Hundreds of components

My Experience

First learned about Apache Camel from Bruce Snyder in 2008 at Colorado Software Summit

Asked to replace IBM Message Broker last year (2014) for a client

My Experience

James Strachan

Me: “Do you know of any particular guides about migrating from IBM Message Broker to Fuse?”

Strachan: “TBH the easiest thing really is to just start using Apache Camel for this kinda stuff”

Result: Recommended Apache Camel to my client and went to work

Getting Started

Started with a Camel Archetype

Used Java 7

Used Java DSL to define routes

Development Strategy

1. Write an integration test against existing service
2. Implement the service with Camel; unit test
3. Copy logic from step 1; write integration test

Legacy Integration Test

```
@Test
public void sendGPIRequestUsingSoapApi() throws Exception {
    SOAPElement bodyChildOne = getBody(message).addChildElement("gpiRequest", "m");
    SOAPElement bodyChildTwo = bodyChildOne.addChildElement("args0", "m");
    bodyChildTwo.addChildElement("NDC", "ax22").addTextNode("54561237201");
    SOAPMessage reply = connection.call(message, getUrlWithTimeout(SERVICE_NAME));
    if (reply != null) {
        Iterator itr = reply.getSOAPBody().getChildElements();
        Map resultMap = TestUtils.getResults(itr);
        assertEquals("66100525123130", resultMap.get("GPI"));
    }
}
```

Drug Service Implementation

```java
@WebService
public interface DrugService {
    @WebMethod(operationName = "gpiRequest")
    GpiResponse findGpiByNdc(GpiRequest request);
}
```

Spring Configuration

```
@Configuration
@ImportResource("classpath:META-INF/cxf/cxf.xml")
@ComponentScan("com.raibledesigns.camel")
public class CamelConfig extends CamelConfiguration {

    @Override
    protected void setupCamelContext(CamelContext camelContext) throws Exception {
        PropertiesComponent pc = new PropertiesComponent();
        pc.setLocation("classpath:application.properties");
        camelContext.addComponent("properties", pc);
        super.setupCamelContext(camelContext);
    }
}
```

CXF Servlet

```
    @Override
    public void onStartup(ServletContext servletContext) throws ServletException {
        servletContext.addListener(new ContextLoaderListener(getContext()));
        ServletRegistration.Dynamic servlet =
            servletContext.addServlet("CXFServlet", new CXFServlet());
        servlet.setLoadOnStartup(1);
        servlet.setAsyncSupported(true);
        servlet.addMapping("/api/*");
    }

    private AnnotationConfigWebApplicationContext getContext() {
        AnnotationConfigWebApplicationContext context =
            new AnnotationConfigWebApplicationContext();
        context.setConfigLocation("com.raibledesigns.camel.config");
        return context;
    }
}
```

Drug Route

```
@Component
public class DrugRoute extends RouteBuilder {
    private String uri = "cxf:/drugs?serviceClass=" + DrugService.class.getName();

    @Override
    public void configure() throws Exception {
        from(uri)
            .recipientList(simple("direct:${header.operationName}"));
        from("direct:gpiRequest").routeId("gpiRequest")
            .process(new Processor() {
                public void process(Exchange exchange) throws Exception {
                    // get the ndc from the input
                    String ndc = exchange.getIn().getBody(GpiRequest.class).getNDC();
                    exchange.getOut().setBody(ndc);
                }
            })
            .to("sql:{{sql.selectGpi}}")
            .to("log:output")
            .process(new Processor() {
                public void process(Exchange exchange) throws Exception {
                    // get the gpi from the input
                    List<HashMap> data = (ArrayList<HashMap>) exchange.getIn().getBody();
                    DrugInfo drug = new DrugInfo();
                    if (data.size() > 0) {
                        drug = new DrugInfo(String.valueOf(data.get(0).get("GPI")));
                    }
                    GpiResponse response = new GpiResponse(drug);
                    exchange.getOut().setBody(response);
                }
            });
    }
}
```

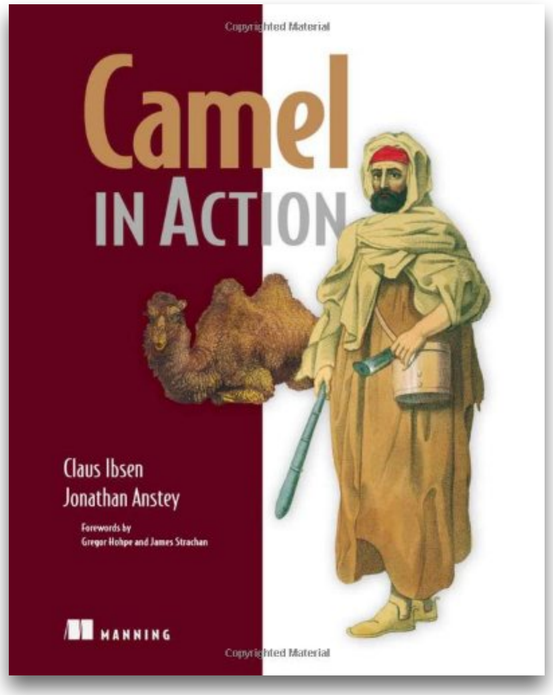
Unit Testing

Hardest part was figuring out Camel's testing support

Bought Camel in Action, read chapter 6

Went to work

- Eliminated dependency on datasource
- Modified route and intercepted SQL calls



Unit Testing: DrugRouteTests.java

```java
@RunWith(CamelSpringJUnit4ClassRunner.class)
@ContextConfiguration(loader = CamelSpringDelegatingTestContextLoader.class,
                      classes = CamelConfig.class)
@DirtiesContext(classMode = DirtiesContext.ClassMode.AFTER_EACH_TEST_METHOD)
@UseAdviceWith
public class DrugRouteTests {

    @Autowired
    CamelContext camelContext;

    @Produce
    ProducerTemplate template;

    @EndpointInject(uri = "mock:result")
    MockEndpoint result;

    static List<Map> results = new ArrayList<Map>() {{
        add(new HashMap<String, String>() {{
            put("GPI", "123456789");
        }});
    }};
    // continued on next slide
}
```

Unit Testing: @Before

```
@Before
public void before() throws Exception {
    camelContext.setTracing(true);

    ModelCamelContext context = (ModelCamelContext) camelContext;
    RouteDefinition route = context.getRouteDefinition("gpiRequest");
    route.adviceWith(context, new RouteBuilder() {
        @Override
        public void configure() throws Exception {
            interceptSendToEndpoint("sql:*").skipSendToOriginalEndpoint().process(new Processor() {
                @Override
                public void process(Exchange exchange) throws Exception {
                    exchange.getOut().setBody(results);
                }
            });
        }
    });
    route.to(result);
    camelContext.start();
}
```

Unit Testing: @Test

```
@Test
public void testMockSQLEndpoint() throws Exception {
    result.expectedMessageCount(1);
    GpiResponse expectedResult = new GpiResponse(new DrugInfo("123456789"));
    result.allMessages().body().contains(expectedResult);

    GpiRequest request = new GpiRequest();
    request.setNDC("123");
    template.sendBody("direct:gpiRequest", request);

    MockEndpoint.assertIsSatisfied(camelContext);
}
```

Code Coverage

```
<build>
    <plugins>
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>cobertura-maven-plugin</artifactId>
            <configuration>
                <instrumentation>
                    <excludes>
                        <exclude>**/model/*.class</exclude>
                        <exclude>**/AppInitializer.class</exclude>
                        <exclude>**/StoredProcedureBean.class</exclude>
                        <exclude>**/SoapActionInterceptor.class</exclude>
                    </excludes>
                </instrumentation>
                <check/>
            </configuration>
            <version>2.6</version>
        </plugin>
...
<reporting>
    <plugins>
        <plugin>
            <groupId>org.codehaus.mojo</groupId>
            <artifactId>cobertura-maven-plugin</artifactId>
            <version>2.6</version>
        </plugin>
```

Integration Testing

```
public class DrugRouteITest {

    private static final String URL = "http://localhost:8080/api/drugs";

    protected static DrugService createCXFClient() {
        JaxWsProxyFactoryBean factory = new JaxWsProxyFactoryBean();
        factory.setBindingId("http://schemas.xmlsoap.org/wsdl/soap12/");
        factory.setServiceClass(DrugService.class);
        factory.setAddress(getTestUrl(URL));
        return (DrugService) factory.create();
    }

    @Test
    public void findGpiByNdc() throws Exception {
        // create input parameter
        GpiRequest input = new GpiRequest();
        input.setNDC("54561237201");

        // create the webservice client and send the request
        DrugService client = createCXFClient();
        GpiResponse response = client.findGpiByNdc(input);

        assertEquals("66100525123130", response.getDrugInfo().getGPI());
    }
}
```

Integrating Spring Boot

Had to upgrade to Spring 4

Camel 2.13.1 didn’t support Spring 4

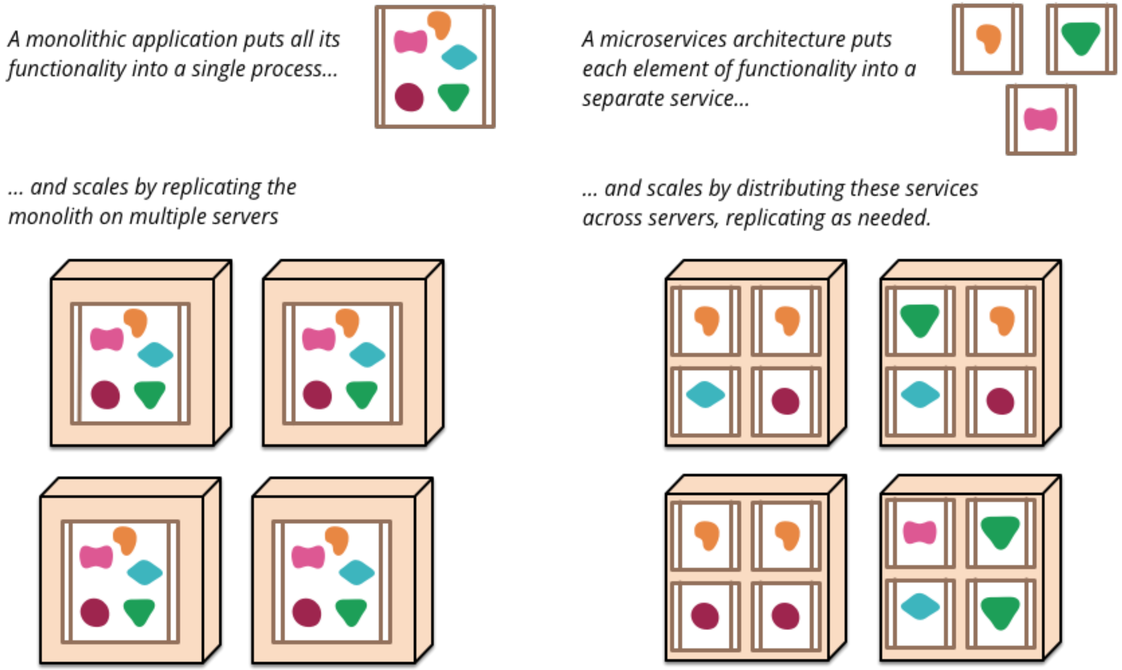
Camel 2.14-SNAPSHOT, CXF 3.0

Found issues with camel-test-spring, fixed, created pull request

```
    <properties>
      <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
      <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
-     <camel.version>2.13.1</camel.version>
-     <cxf.version>2.7.11</cxf.version>
-     <spring.version>3.2.8.RELEASE</spring.version>
+     <camel.version>2.14-SNAPSHOT</camel.version>
+     <cxf.version>3.0.0</cxf.version>
+     <spring.version>4.0.5.RELEASE</spring.version>
    </properties>
```

Learn more: http://raibledesigns.com/rd/entry/developing_services_with_apache_camel2

Microservices Deployment



http://martinfowler.com/articles/microservices.html

Camel 2.15 includes Spring Boot Support

Auto-configuration of Camel context

Auto-detects Camel routes

Registers key Camel utilities

- Producer Template
- Consumer Template
- Type Converter

Connects Spring Boot's external configuration with Camel properties

```
<dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-spring-boot</artifactId>
    <version>${camel.version}</version>
</dependency>
```

Load Testing with Gatling

- High Performance
  - Scala, Akka and Netty
- Nice HTML-based Reporting
- Scenario Recorder

```
val scn = scenario("Test the Blog entity")
    .exec(http("First unauthenticated request")
    .get("/api/account")
    .headers(headers_http)
    .check(status.is(401)))
    .pause(10)
    .exec(http("Authentication")
    .post("/api/authenticate")
    .headers(headers_http_authentication)
    .formParam("username", "admin")
    .formParam("password", "admin")
    .check(jsonPath("$.token").saveAs("x_auth_token")))
    .pause(1)
    .exec(http("Authenticated request")
    .get("/api/account")
    .headers(headers_http_authenticated)
    .check(status.is(200)))
    .pause(10)
    .repeat(2) {
        exec(http("Get all blogs")
        .get("/api/blogs")
        .headers(headers_http_authenticated)
        .check(status.is(200)))
        .pause(10 seconds, 20 seconds)
        .exec(http("Create new blog")
        .put("/api/blogs")
```

Gatling Approach

1. Write tests to run against current system. Find the number of concurrent requests that make it fall over.
2. Run tests against new system and tune accordingly.
3. Throttle requests if there are remote connectivity issues with 3rd parties. If I needed to throttle requests, I was planning to use Camel's Throttler.

My Gatling Strategy

Used Gatling's Recorder

- Listened on port 8000
- Changed DrugServiceTest to use same port
- Ran integration test

Created AbstractSimulation.scala to allow changing parameters

AbstractSimulation.scala

```scala
import io.gatling.core.scenario.Simulation
import io.gatling.http.Predef._

/**
 * Base Simulation class that allows passing in parameters.
 */
class AbstractSimulation extends Simulation {

  val host = System.getProperty("host", "localhost:8080")
  val serviceType = System.getProperty("service", "modern")
  val nbUsers = Integer.getInteger("users", 10).toInt
  val rampRate = java.lang.Long.getLong("ramp", 30L).toLong

  val httpProtocol = http
    .baseURL("http://" + host)
    .acceptHeader("text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8")
    .doNotTrackHeader("1")
    .acceptLanguageHeader("en-US,en;q=0.5")
    .acceptEncodingHeader("gzip, deflate")
    .userAgentHeader("Gatling 2.0")

  val headers = Map(
    """Cache-Control""" -> """no-cache""",
    """Content-Type""" -> """application/soap+xml; charset=UTF-8""",
    """Pragma""" -> """no-cache""")
}
```

DrugServiceSimulation.scala

```
import io.gatling.core.Predef._
import io.gatling.http.Predef._

import scala.concurrent.duration._

class DrugServiceSimulation extends AbstractSimulation {

  val service = if ("modern".equals(serviceType)) "/api/drugs"
                else "/axis2/services/DrugService"

  val scn = scenario("Drug Service :: findGpiByNdc")
    .exec(http(host)
    .post(service)
    .headers(headers)
    .body(RawFileBody("DrugServiceSimulation_request.xml")))

  setUp(scn.inject(ramp(nbUsers users) over (rampRate seconds))).protocols(httpProtocol)
}
```

Executing Simulations

Legacy service with 100 users over 60 seconds

```
mvn test -Dhost=legacy.server:7802 -Dservice=legacy -Dusers=100 -Dramp=60
```

Local drug service with 100 users over 30 seconds (defaults used)

```
mvn test -Dusers=100
```

Results

- Legacy service started failing at 400 requests per second (rps)
- Local service started throwing errors at 4000/rps

Data Feeders

JDBC Feeder allowed making requests contain unique data for each user

ELFileBody allows substituting a ${NDC} variable in XML file

```
val feeder = jdbcFeeder("jdbc:db2://server:50002/database", "username", "password",
    "SELECT NDC FROM GENERICS")

val scn = scenario("Drug Service")
        .feed(feeder)
        .exec(http(host)
        .post(service)
        .headers(headers)
        .body(ELFileBody("DrugServiceSimulation_request.xml")))
```

Performance Results

100 users over 30 seconds

- No failures
- Max response time: 389ms for legacy, 172ms for Camel service

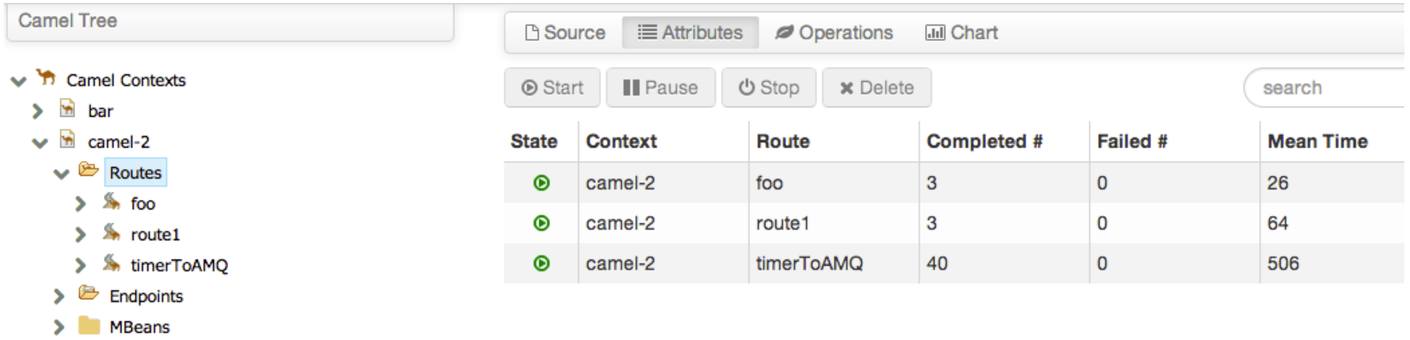
1000 users over 60 seconds

- Legacy service: 50% of requests failed, avg. response time over 40s
- New service: all requests succeeded, response mean time 100ms

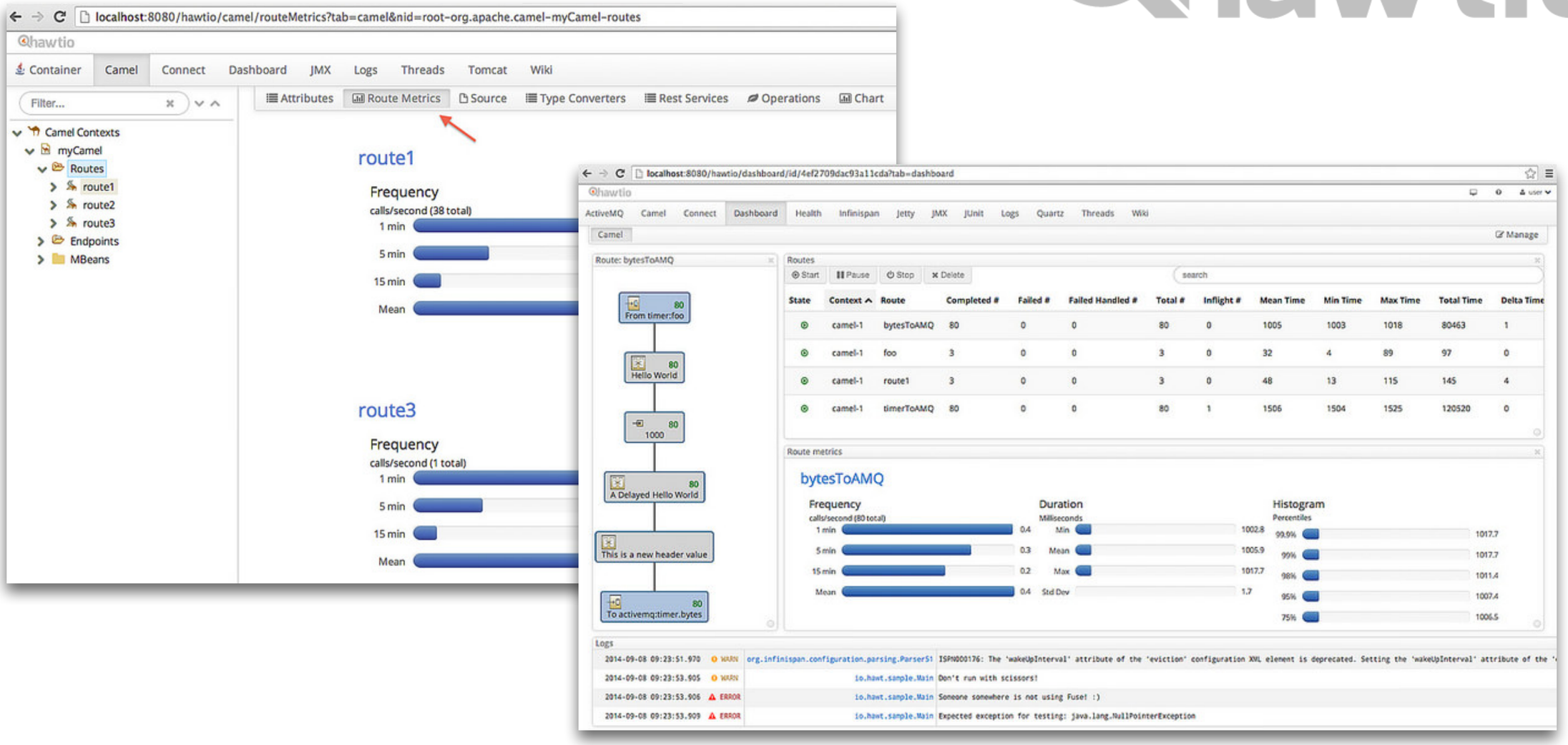
Monitoring

"a modular web console for managing your Java stuff"

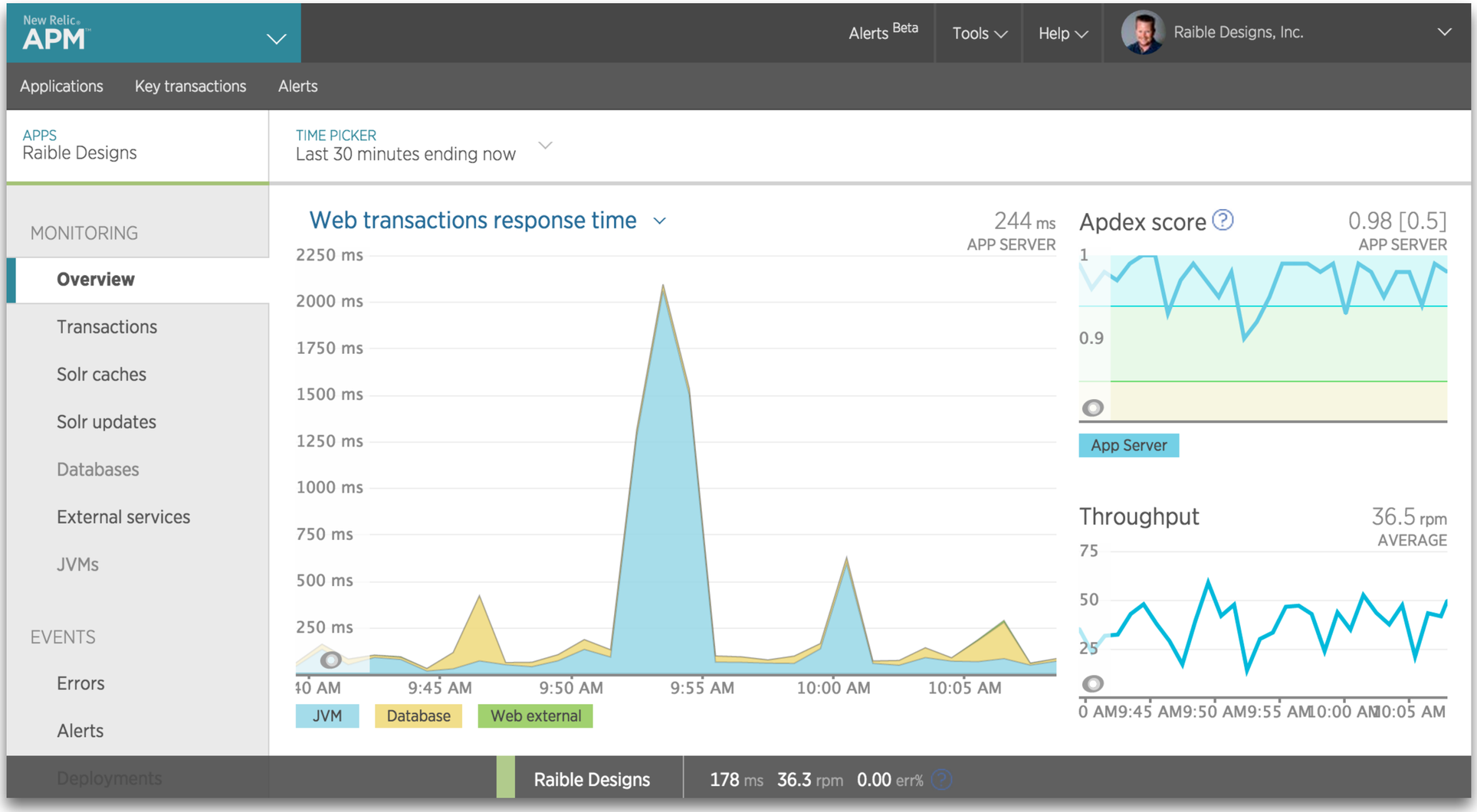
Camel plugin shows routes and metrics, route source editable (XML)



Monitoring


localhost:8080/hawtio/camel/routeMetrics?tab=camel&nid=root-org.apache.camel-myCamel-routes
hawtio
Container
Camel
Connect
Dashboard
JMX
Logs
Threads
Tomcat
Wiki
Filter...
Attributes
Route Metrics
Source
Type Converters
Rest Services
Operations
Chart
Camel Contexts
myCamel
Routes
route1
route2
route3
Endpoints
MBeans
route1
Frequency
calls/second (38 total)
1 min
5 min
15 min
Mean
route3
Frequency
calls/second (1 total)
1 min
5 min
15 min
Mean
localhost:8080/hawtio/dashboard/id/4ef2709dac93a11cda?tab=dashboard
ActiveMQ
Camel
Connect
Dashboard
Health
Infinispan
Jetty
JMX
JUnit
Logs
Quartz
Threads
Wiki
Manage
Route: bytesToAMQ
From timer:foo
Hello World
1000
A Delayed Hello World
This is a new header value
To activemq:timer.bytes
Routes
Start
Pause
Stop
Delete
search
State
Context
Route
Completed #
Failed #
Failed Handled #
Total #
Inflight #
Mean Time
Min Time
Max Time
Total Time
Delta Time
camel-1 bytesToAMQ 80 0 0 80 0 1005 1003 1018 80463 1
camel-1 foo 3 0 0 3 0 32 4 89 97 0
camel-1 route1 3 0 0 3 0 48 13 115 145 4
camel-1 timerToAMQ 80 0 0 80 1 1506 1504 1525 120520 0
Route metrics
bytesToAMQ
Frequency
calls/second (80 total)
1 min
5 min
15 min
Mean
Duration
Milliseconds
Min
Mean
Max
Std Dev
Histogram
Percentiles
99.9%
99%
98%
95%
75%
Logs
2014-09-08 09:23:51.970 WARN org.infinispan.configuration.parsing.ParserS1 ISPN000176: The 'wakeUpInterval' attribute of the 'eviction' configuration XML element is deprecated. Setting the 'wakeUpInterval' attribute of the
2014-09-08 09:23:53.905 WARN io.hawt.sample.Main Don't run with scissors!
2014-09-08 09:23:53.906 ERROR io.hawt.sample.Main Someone somewhere is not using Fuse! :)
2014-09-08 09:23:53.909 ERROR io.hawt.sample.Main Expected exception for testing: java.lang.NullPointerException

New Relic


New Relic APM
Alerts Beta
Tools
Help
Raible Designs, Inc.
Applications
Key transactions
Alerts
APPS
Raible Designs
TIME PICKER
Last 30 minutes ending now
MONITORING
Overview
Transactions
Solr caches
Solr updates
Databases
External services
JVMs
EVENTS
Errors
Alerts
Deployments
Web transactions response time
244 ms
APP SERVER
2250 ms
2000 ms
1750 ms
1500 ms
1250 ms
1000 ms
750 ms
500 ms
250 ms
9:45 AM
9:50 AM
9:55 AM
10:00 AM
10:05 AM
JVM
Database
Web external
Apdex score
0.98 [0.5]
APP SERVER
1
0.9
App Server
Throughput
36.5 rpm
AVERAGE
75
50
25
9:45 AM
9:50 AM
9:55 AM
10:00 AM
10:05 AM
Raible Designs
178 ms
36.3 rpm
0.00 err%

Summary

Open source wins again!

Pleasant experience developing with Apache Camel and Spring

Integration tests were essential to verifying functionality

Gatling Feeders helped discover edge cases with real data

What about Spring Integration?

- Never looked into it, Camel was good enough

Action!

Don’t be afraid to try new things

Try Apache Camel if you need Enterprise Integration

Spring Boot Rocks! Try it if you’re already using Spring

Test, Test, Test, then test some more

Gatling is a great way to find our performance and test with real data

Questions?

Contact Information

http://raibledesigns.com

@mraible

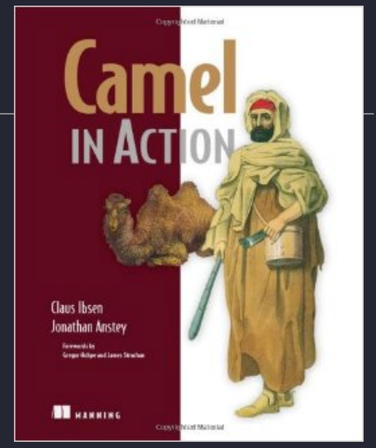
Presentations

http://slideshare.net/mraible

Code

http://github.com/mraible

Additional Information



Camel in Action

- 2nd edition on its way (summer 2016)!
- Microservices with Spring Boot, Dropwizard, Jetty, CDI
- Cloud, Docker, Kubernetes, Fabric8, Hawtio
- Reactive with RxJava and Vert.x

Follow its authors: @davsclaus and @jon_anstey

http://camel.apache.org

Devoxx4Kids Denver

Teaching Kids to Program

Java, Minecraft, robots, oh my!

Non-profit, looking for speakers!

http://www.meetup.com/Devoxx4Kids-Denver/