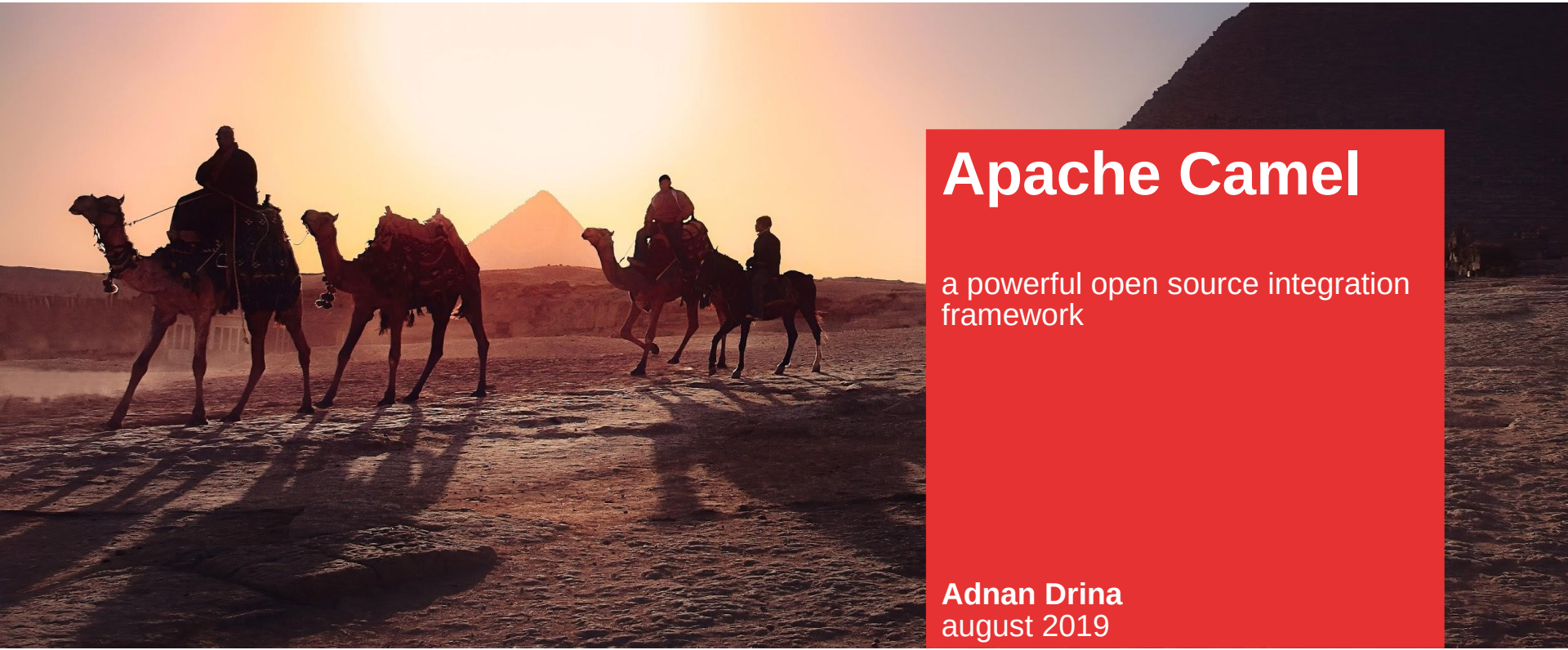
CONCLUSION



Apache Camel

a powerful open source integration framework

Adnan Drina
august 2019

Agenda

- An introduction to Camel
- Camel's main features
- Camel's architecture and concepts
- Some Examples
- Dinner
- Running Camel
- Camel in the Cloud
- What is next?
- Workshop (Your first Camel ride)

What is Apache Camel?

Apache Camel is a powerful **Open Source Integration Framework** based on **Enterprise Integration Patterns**

Apache Camel 2018 Numbers

The Camel project was **started in early 2007** and now is a mature open source project with a **strong community**.

Number of releases in 2018: **12**
Number of posts on Camel user forum in 2018: **1266**
Number of gitter chat users at end of 2018: **428**
Number of commits in 2018: **3600** (`git shortlog -ns --since 2018-01-01 --until 2019-01-01 | cut -c1-7 | awk '{ SUM += $1} END { print SUM }'`)

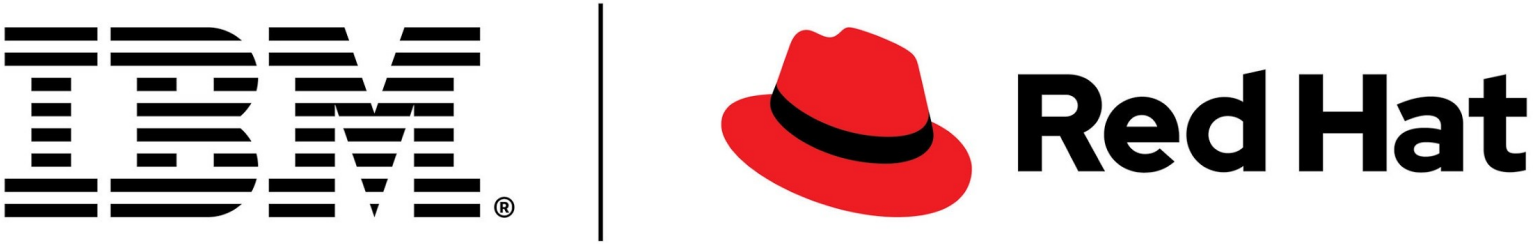
Total number of JIRA tickets created at end of 2018: **13033**
Number of JIRA tickets created in 2018: **924**
Number of JIRA tickets resolved in 2018: **766**

Stackoverflow number of questions at end of 2018: **8375**
Stackoverflow number of watchers at end of 2018: **1.8k**

Number of stars on github at end of 2018: **2303**
Total number of commits at end of 2018: **34431**
Total number of contributors on github at end of 2018: **447**
Number of closed pull requests at end of 2018: **2674**
Number of closed pull requests in 2018: **280** (`is:pr is:closed merged:>=2018-01-01`)
Number of committers doing commits in 2018: **184** (`git shortlog --since 2018-01-01 --until 2019-01-01 -ns | wc -l`).

The Apache Software Foundation recently posted a summary of the most active projects in 2018 and **Apache Camel was ranked 4th by commits**.

Red Hat offers Fuse



Red Hat offers **Fuse**, an **enterprise integration platform based on Apache Camel** and other open source integration projects, with developer and **production support**.

Camel has been used as the internal engine

Powering projects such as:

- Apache ServiceMix ESB
- Apache ActiveMQ
- Talend ESB
- JBoss Switchyard
- Syndesis
- SAP HANA
- Netflix
- **And many other frameworks**

APACHE Camel

PATTERN BASED INTEGRATION

Apache Camel, a powerful pattern-based integration engine with a comprehensive set of connectors and data formats to tackle any integration problem.

ENTERPRISE INTEGRATION PATTERNS

Build integrations using enterprise best practices.

200+ COMPONENTS

Batch, messaging, web services, cloud, APIs, and more ...

BUILT-IN DATA TRANSFORMATION

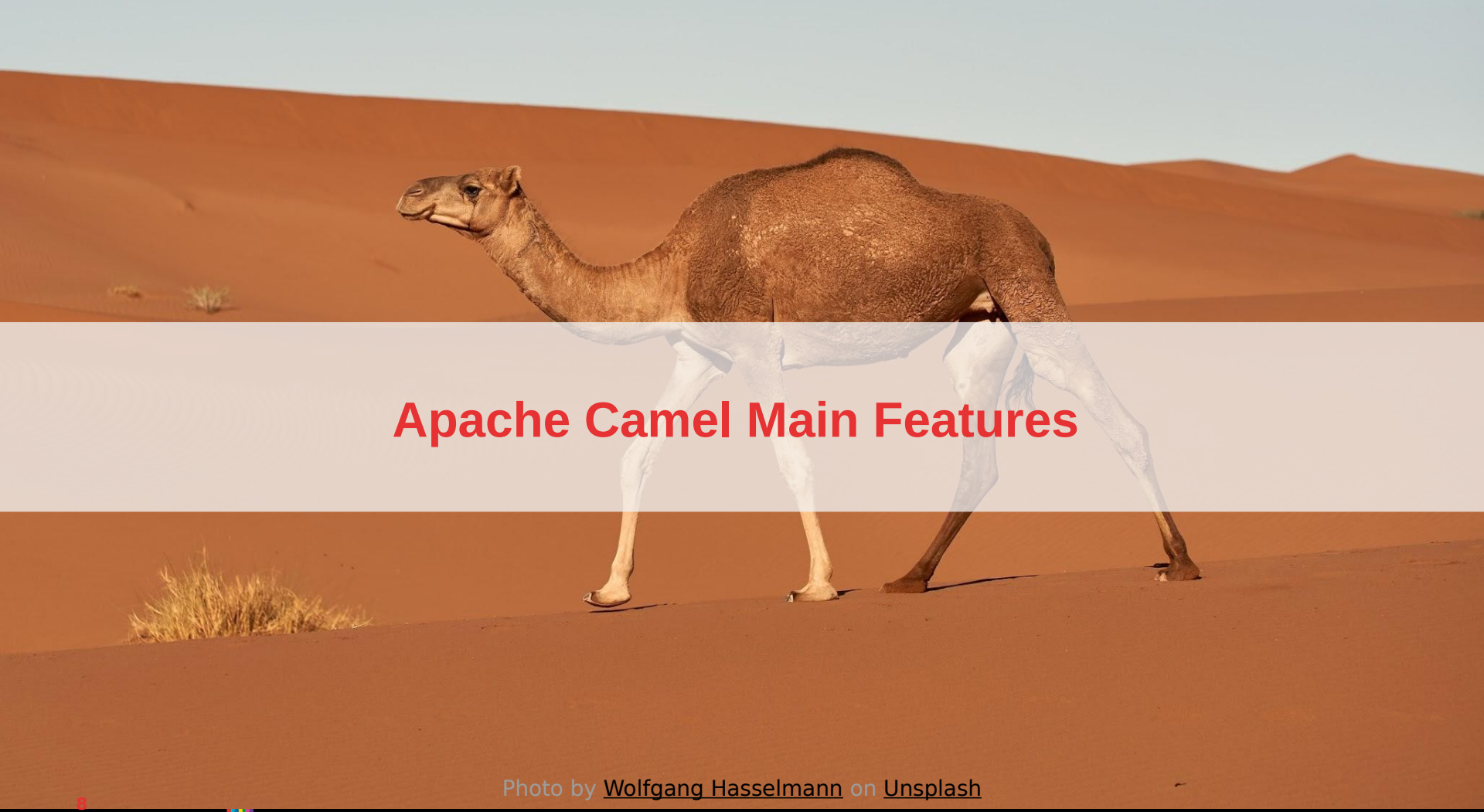
JSON, XML, HL7, YAML, SOAP, Java, CSV, and more ...

INTUITIVE ROUTING

Develop integrations quickly in Java or XML.

NATIVE REST SUPPORT

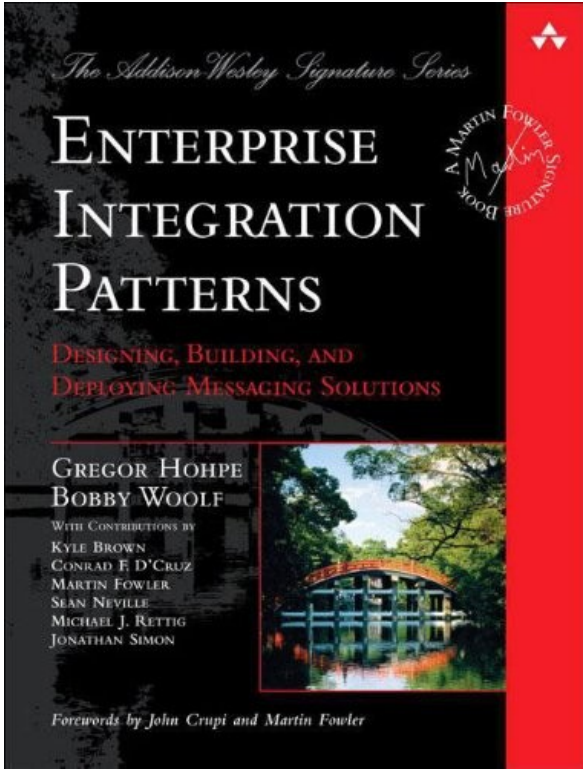
Create, connect, and compose APIs with ease.



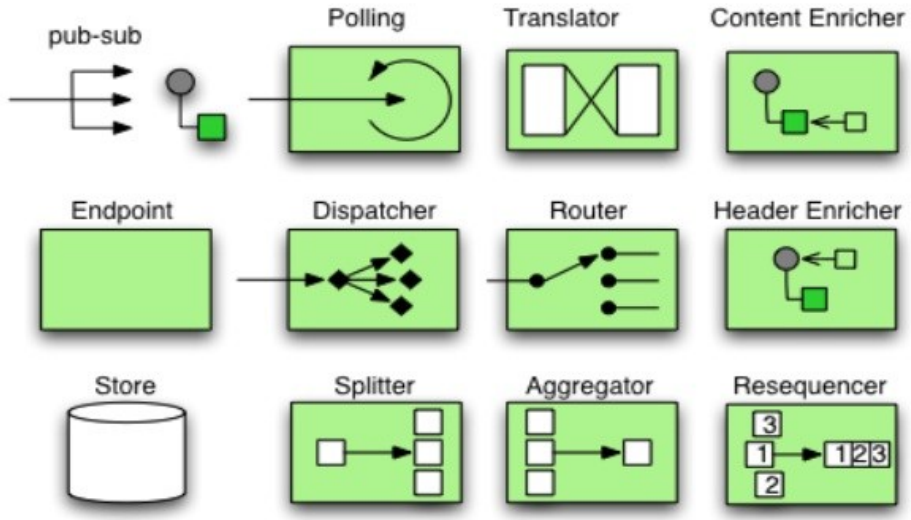
Apache Camel Main Features

Photo by Wolfgang Hasselmann on Unsplash

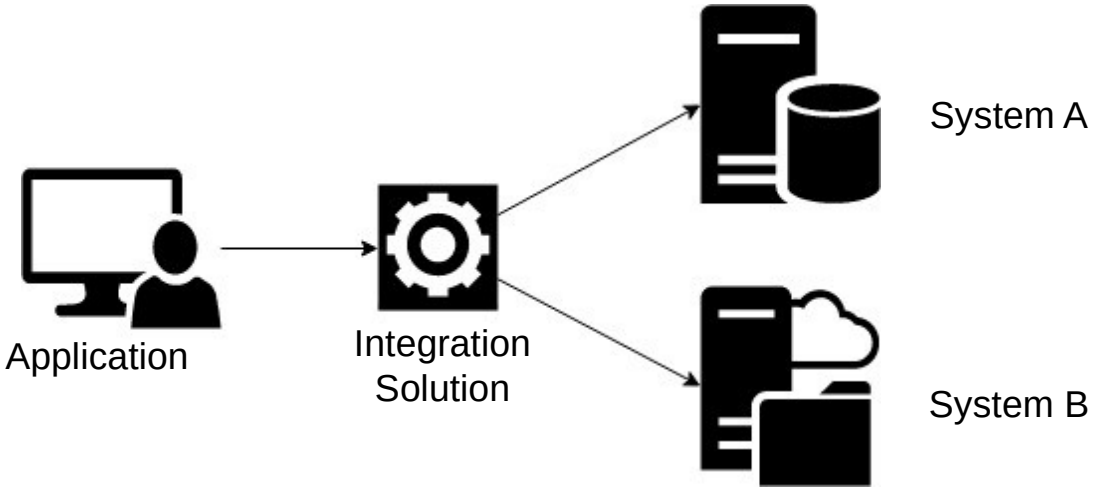
Enterprise Integration Patterns



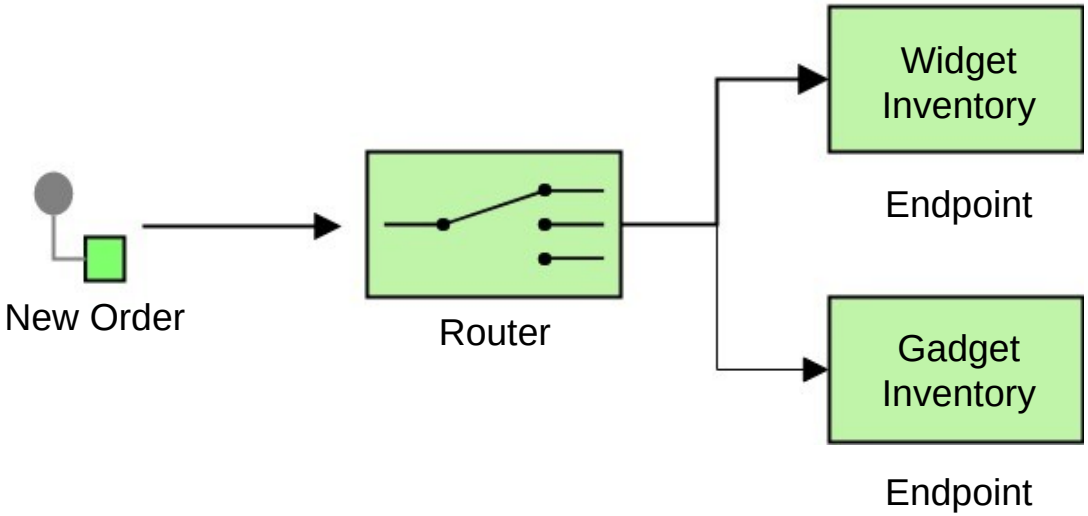
- A book by Gregor Hohpe and Bobby Woolf (October 10, 2003)
- Recipes for solving integration problems
- Based on study of thousands of Integration projects.
- Provide common vocabulary and diagram notations



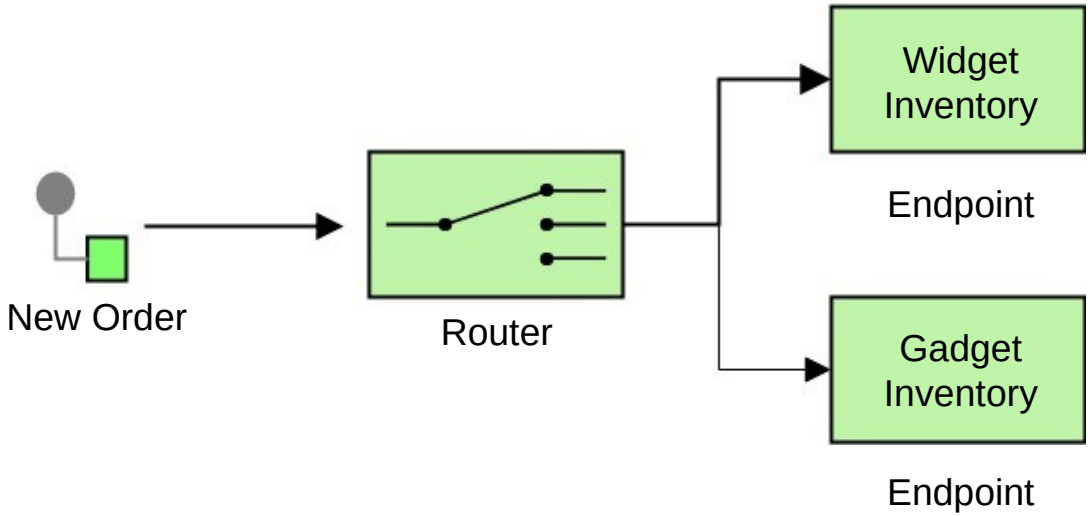
System Integration



Content Based Router

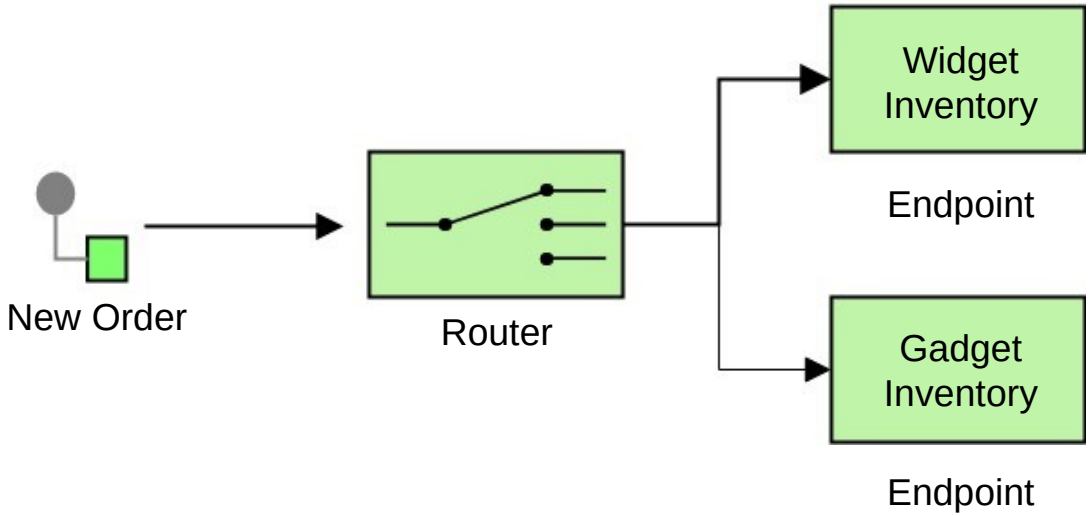


Content Based Router



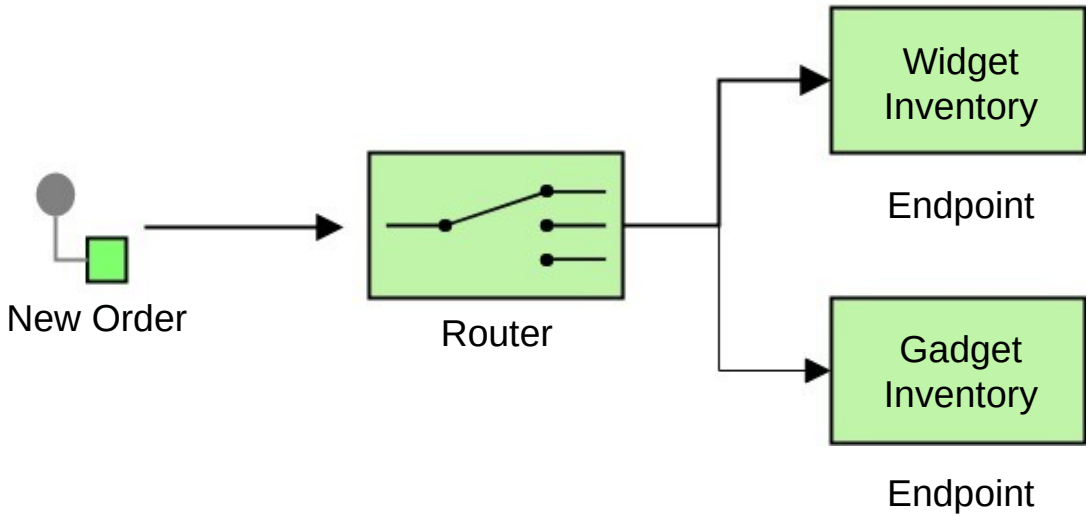
from **newOrder**
choice
when **isWidget** to **widget**
otherwise to **gadget**

Content Based Router




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

Content Based Router




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

JAVA Code


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

public class ExampleCamelRoute 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);
        }
    }
```

JAVA Code


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

public class ExampleCamelRoute 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");
        }
    }
```

JAVA Code

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

public class ExampleCamelRoute extends RouteBuilder {

    public void configure() throws Exception {

        from("file:inbox/orders?delete=true")
          .choice()
            .when(xpath("/order/product = 'widget'"))
              .to("activemq:queue:widget")
            .otherwise()
              .to("activemq:queue:gadget");
        }
}
```

Components

Number of Components: 298 in 235 JAR artifacts (0 deprecated)

Component	Available From	Description
ActiveMQ (camel-activemq) `activemq:destinationType:destinationName`	1.0	The activemq component allows messages to be sent to (or consumed from) Apache ActiveMQ. This component extends the Camel JMS component.
AHC (camel-ahc) `ahc:httpUri`	2.8	To call external HTTP services using Async Http Client.
AHC Websocket (camel-ahc-ws) `ahc-ws:httpUri`	2.14	To exchange data with external Websocket servers using Async Http Client.
AMQP (camel-amqp) `amqp:destinationType:destinationName`	1.2	Messaging with AMQP protocol using Apache QPid Client.
Apache Flink (camel-flink) `flink:endpointType`	2.18	The flink component can be used to send DataSet jobs to Apache Flink cluster.
Apache Pulsar (camel-pulsar) `pulsar:persistence://tenant/namespace/topic`	2.24	Camel Apache Pulsar Component
Apache Spark (camel-spark) `spark:endpointType`	2.17	The spark component can be used to send RDD or DataFrame jobs to Apache Spark cluster.
APNS (camel-apns) `apns:name`	2.8	For sending notifications to Apple iOS devices.
AS2 (camel-as2) `as2:apiName/methodName`	2.22	Component used for transferring data secure and reliable over the internet using the AS2 protocol.
Asterisk (camel-asterisk) `asterisk:name`	2.18	The asterisk component is used to interact with Asterisk PBX Server.
Atmos (camel-atmos) `atmos:name/operation`	2.15	The atmos component is used for integrating with EMC's Atomos Storage.
Atmosphere Websocket (camel-atmosphere-websocket) `atmosphere-websocket:servicePath`	2.14	To exchange data with external Websocket clients using Atmosphere.
Atom (camel-atom) `atom:feedUri`	1.2	The atom component is used for consuming Atom RSS feeds.
Atomix Map (camel-atomix) `atomix-map:resourceName`	2.20	The atomix-map component is used to access Atomix's distributed map.
Atomix Messaging (camel-atomix) `atomix-messaging:resourceName`	2.20	The atomix-messaging component is used to access Atomix's group messaging.
Atomix MultiMap (camel-atomix)	2.20	The atomix-multimap component is used to

https://github.com/apache/camel/tree/m

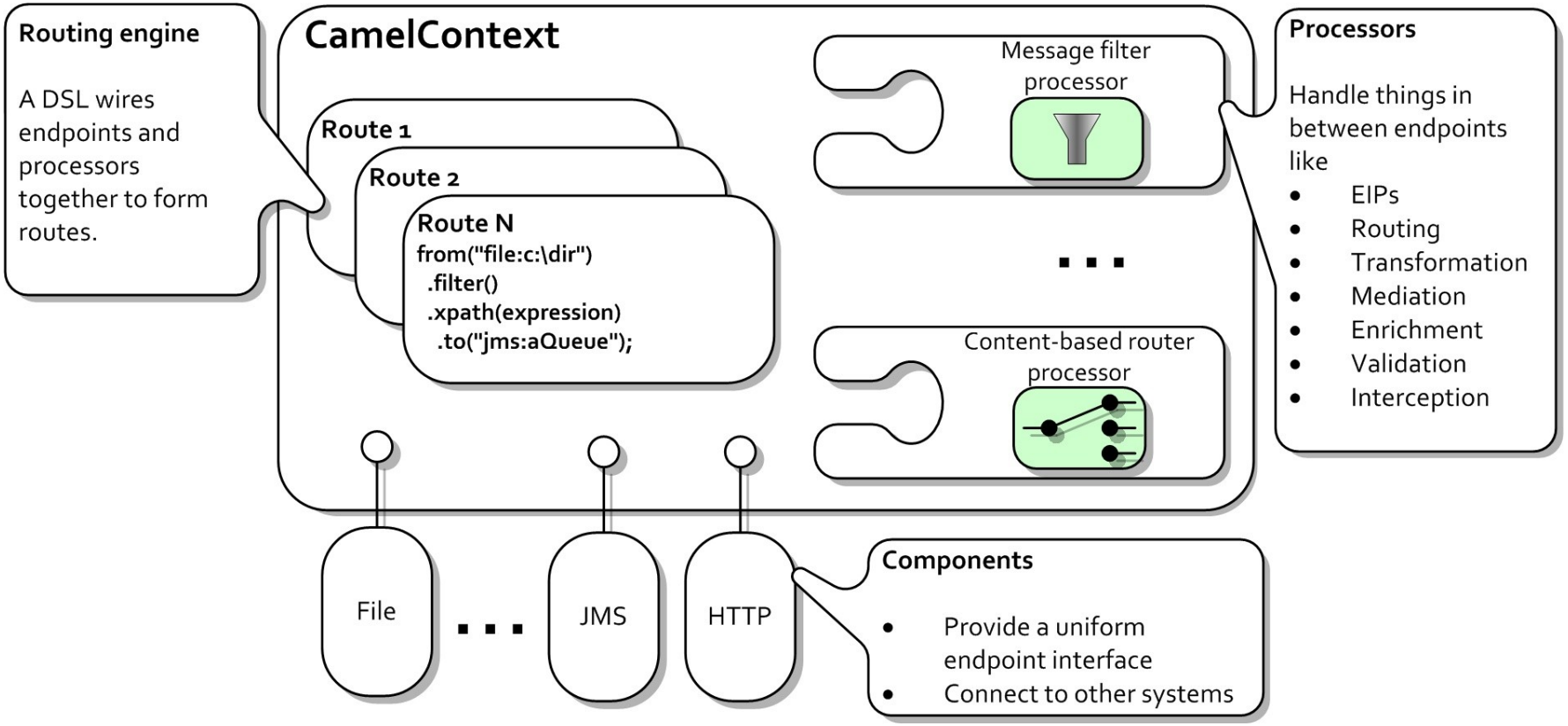
Java Code with Camel and without Camel


```
public class CopyFilesCamel extends RouteBuilder {

   public void configure() throws Exception {

      from("file:data/input")
            .to("file:data/output");
   }
}
```

```
public class CopyFilesWithoutCamel {

   public static void main(String[] args) throws IOException {

      //Step 1 - Create a file Object for the the directories to read and write.
      File inputDirectory = new File("data/input");
      File outputDirectory = new File("data/output");
      outputDirectory.mkdir();

      //Step 2 -  Read the files and Iterate the files
      File[] files = inputDirectory.listFiles();
      for (File source : files) {
         if (source.isFile()) {
            File dest = new File(
                  outputDirectory.getPath()
                        + File.separator
                        + source.getName());

            //Step 3 : Create a Output Stream to write the files.
            OutputStream oStream = new FileOutputStream(dest);
            byte[] buffer = new byte[(int) source.length()];
            FileInputStream iStream = new FileInputStream(source);
            iStream.read(buffer);
            try {
               oStream.write(buffer);
            } finally {
               oStream.close();
               iStream.close();
            }
         }
      }
```

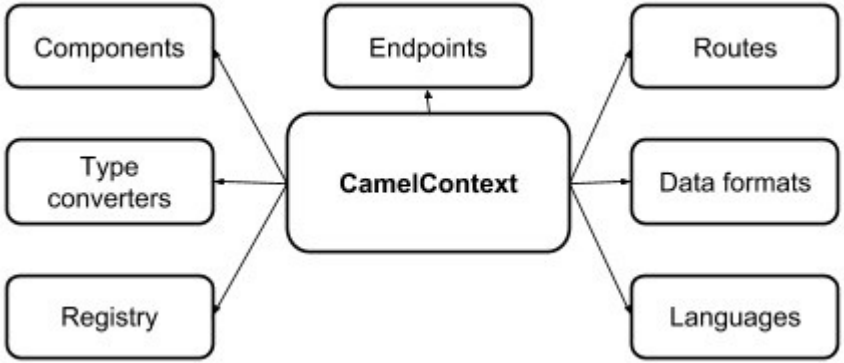
Apache Camel Architecture

Apache Camel Architecture



Apache Camel Context

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



Apache Camel Route

Integration pipeline between an Consumer and Producers

```
from("file:work/cbr/input")
  .split(xpath("//orders"))
    .choice()
      .when(xpath("/order:order/order:type = 'E'"))
        .to("amqp:queue:electronic/us")
      .otherwise()
        .recipientList(simple("http4://otherservice"));
```

Apache 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 (Configuration)
    - **options** = setup of properties for component, list of name/value pairs (Parameters)

- Examples:
  - **kafka:my-topic?brokers=localhost:9092**
  - **mongodb3:mongo?database=demo&collection=orders&operation=insert**
  - **sql-stored:classpath:stored-proc/CreateOrder.sql?dataSource=#dataSource**
  - **ftp://john@localhost/ftp?password=welcome**
  - **timer://myTimer?period=2000**

Apache Camel Endpoint Roles

- Consumer (**from**)
  - receives messages from an external source and creates a message exchange object
  - 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

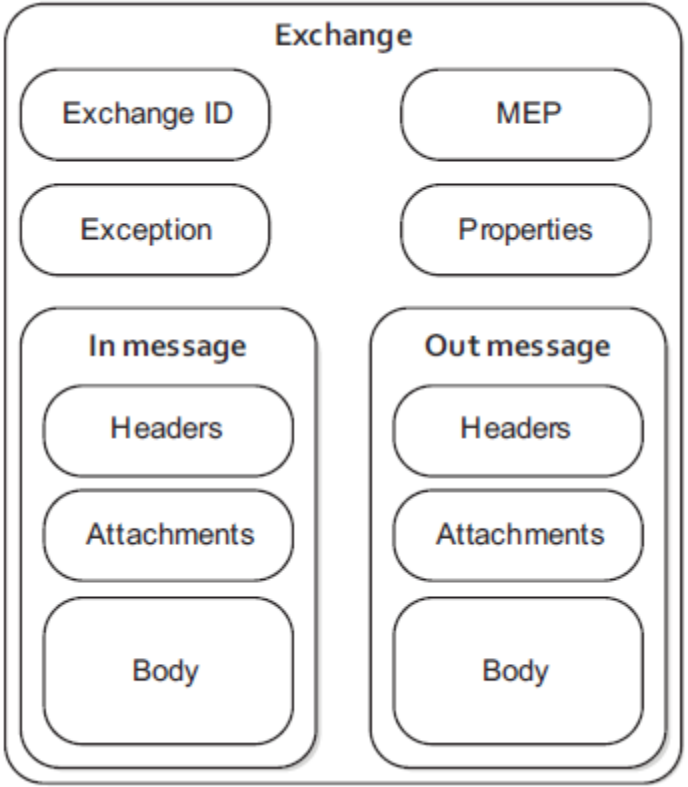
Apache Camel REST DSL


```
rest("/partyInteractionManagement/v1")
    .get("/partyInteraction")
        .id("retrievePartyInteractions")
        .produces("application/json")
        .param()
            .name("customerId")
            .type(RestParamType.query)
            .dataType("string")
            .required(true)
            .description("To retrieve interactions related with a specific customer")
        .endParam()
        .param()
            .name("status")
            .type(RestParamType.query)
            .dataType("string")
            .required(true)
            .description("To obtain interactions with a specific status")
        .endParam()
        .param()
            .name("duration")
            .type(RestParamType.query)
            .dataType("integer")
            .required(false)
            .description("Limit historical data sets to past N days")
        .endParam()
        .to("direct:retrievePartyInteractions")
```

Apache Camel Message Model

- Message
  - basic structure for moving data over a route
  - first created by producer

- Message Exchange - ME
  - message container during routing
  - link between producer and consumer
  - Message Exchange Pattern (MEP):
    - InOnly (fire & forget: JMS message)
    - InOut (request-response: HTTP request)



Apache Camel Processor

Processor is a Message consumer of Message Exchange.

- 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 myProcessor = new Processor() {
    public void process(Exchange exchange) {
        exchange.getBody();
        //e.g do something with Body..
    }
};

from("file:data/inbox").filter(header("foo").isEqualTo("bar"))
    .process(myProcessor).to("jms:queue:order")
```

Apache Camel Processor

aggregate()	idempotentConsumer()	multicast()	removeProperty()	sort()	unmarshal()
aop()	inheritErrorHandler()	onCompletion()	resequence()	split()	validate()
bean()	inOnly()	onException()	rollback()	stop()	wireTap()
choice()	inOut()	pipeline()	routingSlip()	threads()	
convertBodyTo()	loadBalance()	policy()	sample()	throttle()	
delay()	log()	pollEnrich()	setBody()	throwException()	
doTry()	loop()	process()	setExchangePattern()	to()	
end()	markRollbackOnly()	recipientList()	setHeader()	toF()	
enrich()	markRollbackOnlyLast()	removeHeader()	setOutHeader()	transacted()	
filter()	marshal()	removeHeaders()	setProperty()	transform()	

Apache Camel Examples

REST DSL

The Rest DSL is a facade that builds Rest endpoints as consumers for Camel routes. The actual REST transport is leveraged by using Camel REST components such as Restlet, Spark-rest, and others that has native REST integration.

Components supporting Rest DSL

- **camel-netty-http** (also supports Swagger Java)
- **camel-netty4-http** (also supports Swagger Java)
- **camel-jetty** (also supports Swagger Java)
- **camel-restlet** (also supports Swagger Java)
- **camel-servlet** (also supports Swagger Java)
- **camel-spark-rest** (also supports Swagger Java from Camel 2.17)
- **camel-undertow** (also supports Swagger Java from Camel 2.17)

REST DSL


```
public final class OrderManagementAPI extends RouteBuilder {
  public void configure() {

    restConfiguration()
        .component("jetty").port(8080).bindingMode(json);

    rest("/orderManagement/v1")
        .get("/order").id("retrieveAllOrders")
          .produces("application/json")
          .to("direct:get-orders")
        .get("/order/{id}").id("retrieveOrder")
          .produces("application/json")
          .to("direct:get-order")
        .post("/order").id("createOrder")
          .consumes("application/json")
          .produces("application/json")
          .to("direct:post-order")
        .patch("/order").id("modifyOrder")
          .consumes("application/json")
          .produces("application/json")
          .to("direct:patch-order");
  }
```

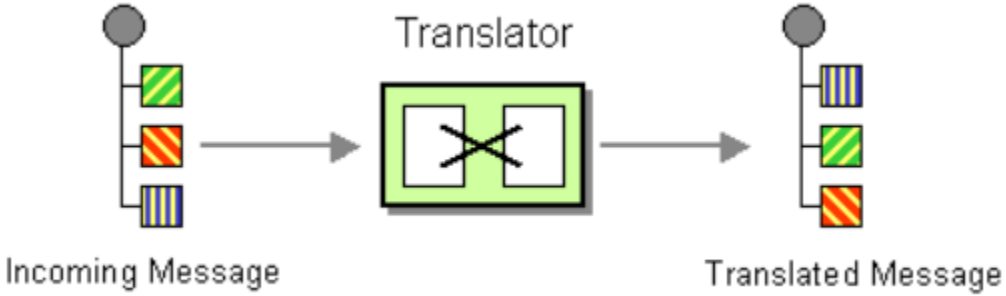
Configures Rest DSL to use camel-jetty component

Uses REST verbs to define four REST services with GET, POST, PUT and DELETE

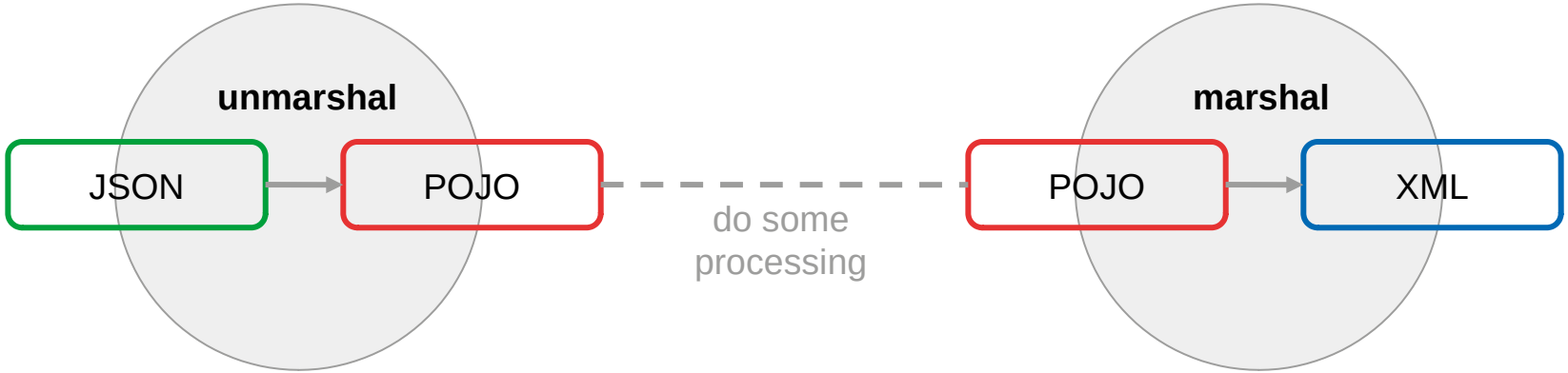
```
from("direct:post-order")
    .transform()
    .to("mongodb3:mongo?database=demo&collection=orders&operation=insert");
```

Data Transformation

Data Transformation means transforming data from one form to another. For example, XML to CSV, XML to JSON, etc. Data Transformation in Apache Camel can be achieved by using Processor in routing logic, using bean, or by using Transform() in DSL.



Data Transformation (Marshalling/Unmarshalling)



Sample Data Transformation

Input JSON

```
{"root": {
        "child1": "text1",
        "child2": "text2"
    }
}
```


Camel route

```
...
from("direct:post-order")
  .unmarshal().json()
  .process(...)
  .marshal().jaxb()
  .to("direct:post-orderXML");
...
```


Output XML

```
<root> <child1>text1</
child1> <child2>text2</
child2> </root>
```

Data Transformation

Suppose you need to prepare text for HTML formatting by replacing all line breaks with a
 tag. You can do this with a built-in Camel expression that searches and replaces using regular expressions:

```
from("direct:start")
        .transform(body().regexReplaceAll("\n", "<br/>"))
    .to("direct:continue");
```

Sample Data Transformation


```
from("direct:post-order")
        .marshal().json()
        .bean(basexUtils, "runXquery(${body},JSON,JSON,'Input2Output.xq')")
        .log(LoggingLevel.INFO, "BaseX","The data is transformed,"${body}"")
        .to("https4://mydomain/order-system-api/1.0/order?bridgeEndpoint=true");
```

Input2Output.xq

```
{
  "output_p1" : $jsonMap(input_p3),
  "output_p2": $jsonMap(input_1),
  "output_p3" : $jsonMap(input_p2)
}
```

Using Camel Message Headers


```
public void configure() throws Exception {

   from("direct:post-order").id("createOrder")
         .marshal().json(JsonLibrary.Jackson, InputOrder.class)
         .bean(basexUtils, "runXquery(${body},JSON,JSON,'Input2Output.xq')")
         .unmarshal().json(JsonLibrary.Jackson, OutputOrder.class)
         .log(LoggingLevel.INFO, "BaseX","The data is transformed,"${body}"")

         .setHeader("p_order_id", simple("${body.p_order_id}"))
         .setHeader("p_order_type", simple("${body.p_order_type}"))
         .setHeader("p_order_notes", simple("${body.p_order_notes}"))
         .setHeader("p_customer_id", simple("${body.p_customer_id}"))
         .setHeader("p_product_id", simple("${body.p_product_id}"))
         .setHeader("p_status", simple("${body.p_status}"))
         .setHeader("p_order_date", simple("${body.p_order_date}"))
         .setHeader("p_last_modified", simple("${body.p_last_modified}"))

         .setBody().simple("resource:classpath:stored-proc/CreateOrder.sql")
         .to("sql-stored:classpath:stored-proc/CreateOrder.sql?dataSource=#dataSource")
         .log(LoggingLevel.INFO, "CreateProc-Response","${body}");
```

Using Camel Exchange Properties


```
from("direct:post-order").id("createOrder")
        .process(exchange -> {
            Order order = exchange.getIn().getBody(Order.class);
            exchange.setProperty("ORDER", order)
            exchange.setProperty("STATUS", order.getType());
        })
        .choice()
            .when(simple("${exchangeProperty[STATUS]} == \"new\""))
            .log("processing order: ${exchangeProperty[ORDER]}")
            .setBody().simple("{\"id\": \"${exchangeProperty[ORDER.getId]}\"}")
            .to("mongodb3:mongo?database=demo&collection=orders&operation=insert")
            .log("order processed : ${body}")
        .otherwise()
            .log("Order event has unexpected status")
        .endChoice()
        .end();
```

Error Handling


```
//send the exception back to the client (rarely used, clients need a meaningful response)
onException(ClientException.class)
      .handled(false)    //default
      .to("log:GeneralError?level=ERROR");


//send a readable error message back to the client and handle the error internally
onException(HandledException.class)
      .handled(true)
      .setBody(constant("error"))
      .to("activemq:queue:ErrorQueue");


//ignore the exception and continue the route (can be dangerous, use wisely)
onException(ContinuedException.class)
      .continued(true);
```

Error Handling

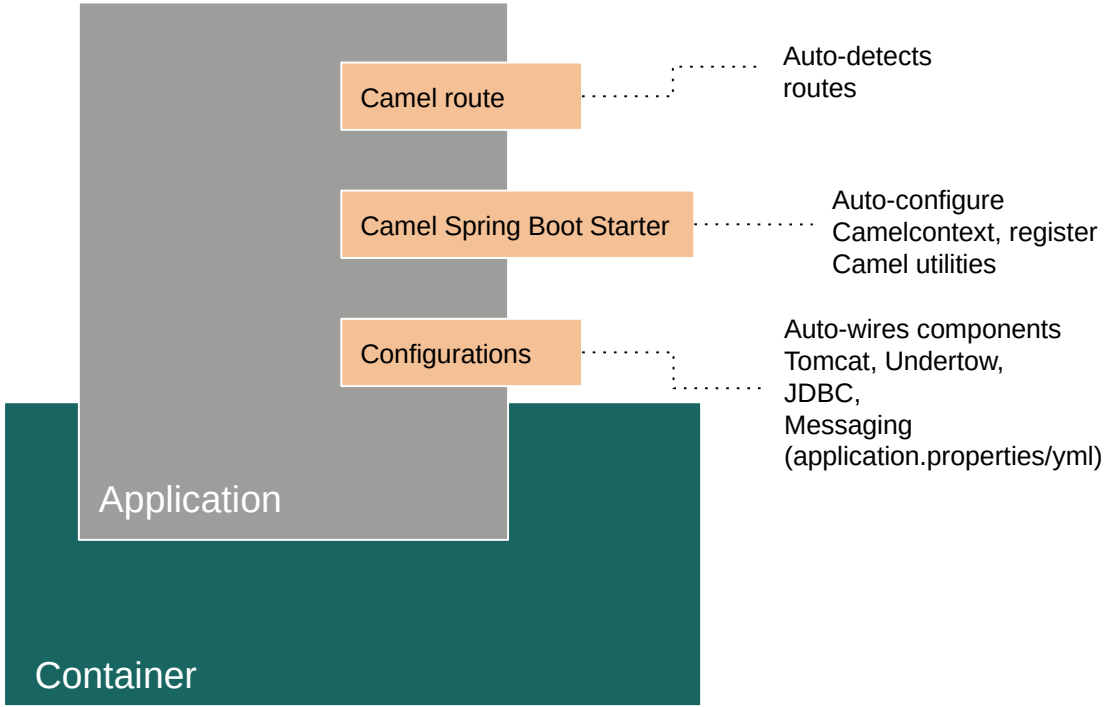

```
public class WorkflowRouter extends RouteBuilder {

  public void configure() throws Exception {

        onException(Exception.class)
              .handled(true)
              .log(LoggingLevel.ERROR, "body: ${body} + headers: ${headers}")
              .setHeader(Exchange.HTTP_RESPONSE_CODE, constant(500))
              .setBody(simple(""))
              .end();

        from("direct:post-order")
              .marshal().json(JsonLibrary.Jackson, Input.class)
              .bean(basexUtils, "runXquery(${body},JSON,JSON,'Input2Output.xq')")
              .log(LoggingLevel.INFO, "BaseX","The data is transformed,"${body}"")
              .to("https4://mydomain/order-system-api/1.0/order?bridgeEndpoint=true")
              .setHeader(Exchange.HTTP_RESPONSE_CODE, constant(200))
              .end();
        }
}
```

Running Apache Camel

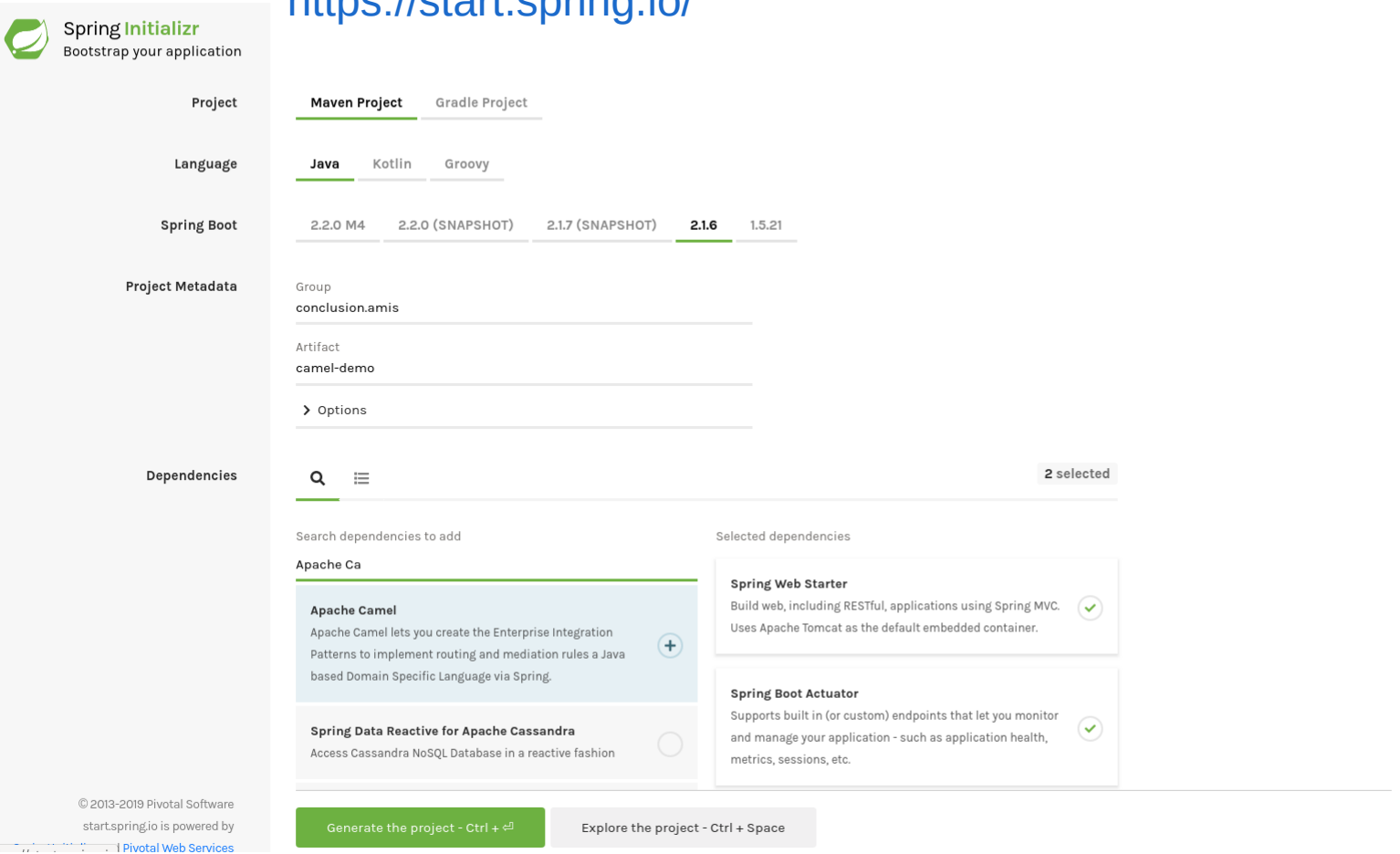
Deploying Apache Camel

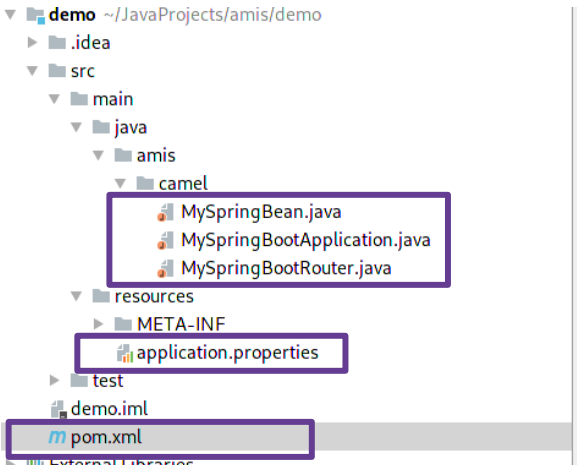
- Standalone JAR
- WAR - Servlet Container, e.g. Apache Tomcat, Jetty
- Spring - Spring Boot
- Java EE - e.g. Wildfly, Glassfish, WebLogic, WebSphere
- OSGi Container - e.g. Apache Karaf, ServiceMix
- Cloud - e.g. OpenShift, Kubernetes, Google Compute Engine, Amazon EC2

Spring Boot


Camel route
Auto-detects routes
Camel Spring Boot Starter
Auto-configure Camelcontext, register Camel utilities
Configurations
Auto-wires components Tomcat, Undertow, JDBC, Messaging (application.properties/yml)
Application
Container

https://start.spring.io/






```
<dependencies>

  <!-- Spring Boot -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
    <exclusions>
      <exclusion>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-tomcat</artifactId>
      </exclusion>
    </exclusions>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-undertow</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-actuator</artifactId>
  </dependency>

  <!-- Camel -->
  <dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-spring-boot-starter</artifactId>
  </dependency>
  <dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-stream-starter</artifactId>
  </dependency>

  <!-- Test -->
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.apache.camel</groupId>
    <artifactId>camel-test-spring</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

Camel Component Starters

camel-activemq-starter	Revert "Use property replacement instead of version management for ca…
camel-ahc-starter	Revert "Use property replacement instead of version management for ca…
camel-ahc-ws-starter	Revert "Use property replacement instead of version management for ca…
camel-amqp-starter	Revert "Use property replacement instead of version management for ca…
camel-apns-starter	Revert "Use property replacement instead of version management for ca…
camel-as2-starter	Revert "Use property replacement instead of version management for ca…
camel-asn1-starter	Revert "Use property replacement instead of version management for ca…
camel-asterisk-starter	Revert "Use property replacement instead of version management for ca…
camel-atmos-starter	Revert "Use property replacement instead of version management for ca…
camel-atmosphere-websocket-starter	Revert "Use property replacement instead of version management for ca…
camel-atom-starter	Revert "Use property replacement instead of version management for ca…
camel-atomix-starter	Revert "Use property replacement instead of version management for ca…
camel-avro-starter	Revert "Use property replacement instead of version management for ca…
camel-aws-cw-starter	CAMEL-13723 - Better docs, CW
camel-aws-ddb-starter	CAMEL-13723 - Improved docs for DDB components, and harmonize region ..
camel-aws-ec2-starter	CAMEL-13723 - Better docs, EC2
camel-aws-ecs-starter	CAMEL-13723 - Better docs, ECS
camel-aws-eks-starter	CAMEL-13723 - Better docs, EKS
camel-aws-iam-starter	CAMEL-13723 - Better docs, IAM
camel-aws-kinesis-starter	CAMEL-13723 - Better docs, Kinesis
camel-aws-kms-starter	CAMEL-13723 - Better docs, KMS
camel-aws-lambda-starter	CAMEL-13723 - Better docs, Lambda
camel-aws-mq-starter	CAMEL-13723 - Better docs, MQ
camel-aws-msk-starter	CAMEL-13723 - Better docs, MSK
camel-aws-s3-starter	CAMEL-13723 - Better docs, S3
camel-aws-sdb-starter	Revert "Use property replacement instead of version management for ca…
camel-aws-ses-starter	CAMEL-13723 - Better docs, SES

There is a camel-xxx-starter module for each Camel component.

Starters are created to meet the following requirements:

- Allow the auto-configuration of the component through the native spring-boot configuration system
- Manage transitive logging dependencies to better integrate with spring-boot logging system
- Include additional dependencies and align transitive ones to minimize the effort of creating a working spring-boot application
- Each starter has its own integration test that verifies its compatibility with the current release of spring-boot.

MySpringBootRouter.java


```java
package amis.camel;

import ...

/**
 * A simple Camel route that triggers from a timer and calls a bean and prints to system out.
 * <p/>
 * Use <tt>@Component</tt> to make Camel auto detect this route when starting.
 */
@Component
public class MySpringBootRouter extends RouteBuilder {

    @Override
    public void configure() {
        from( uri: "timer:hello?period={{timer.period}}").routeId("hello")
            .transform().method( bean: "myBean",  method: "saySomething")
            .filter(simple( value: "${body} contains 'foo'"))
                .to("log:foo")
            .end()
            .to("stream:out");
    }

}
```

MySpringBean.java


```java
package amis.camel;

import ...

/**
 * A bean that returns a message when you call the {@link #saySomething()} method.
 * <p/>
 * Uses <tt>@Component("myBean")</tt> to register this bean with the name <tt>myBean</tt>
 * that we use in the Camel route to lookup this bean.
 */
@Component("myBean")
public class MySpringBean {

    @Value("${greeting}")
    private String say;

    public String saySomething() { return say; }

}
```

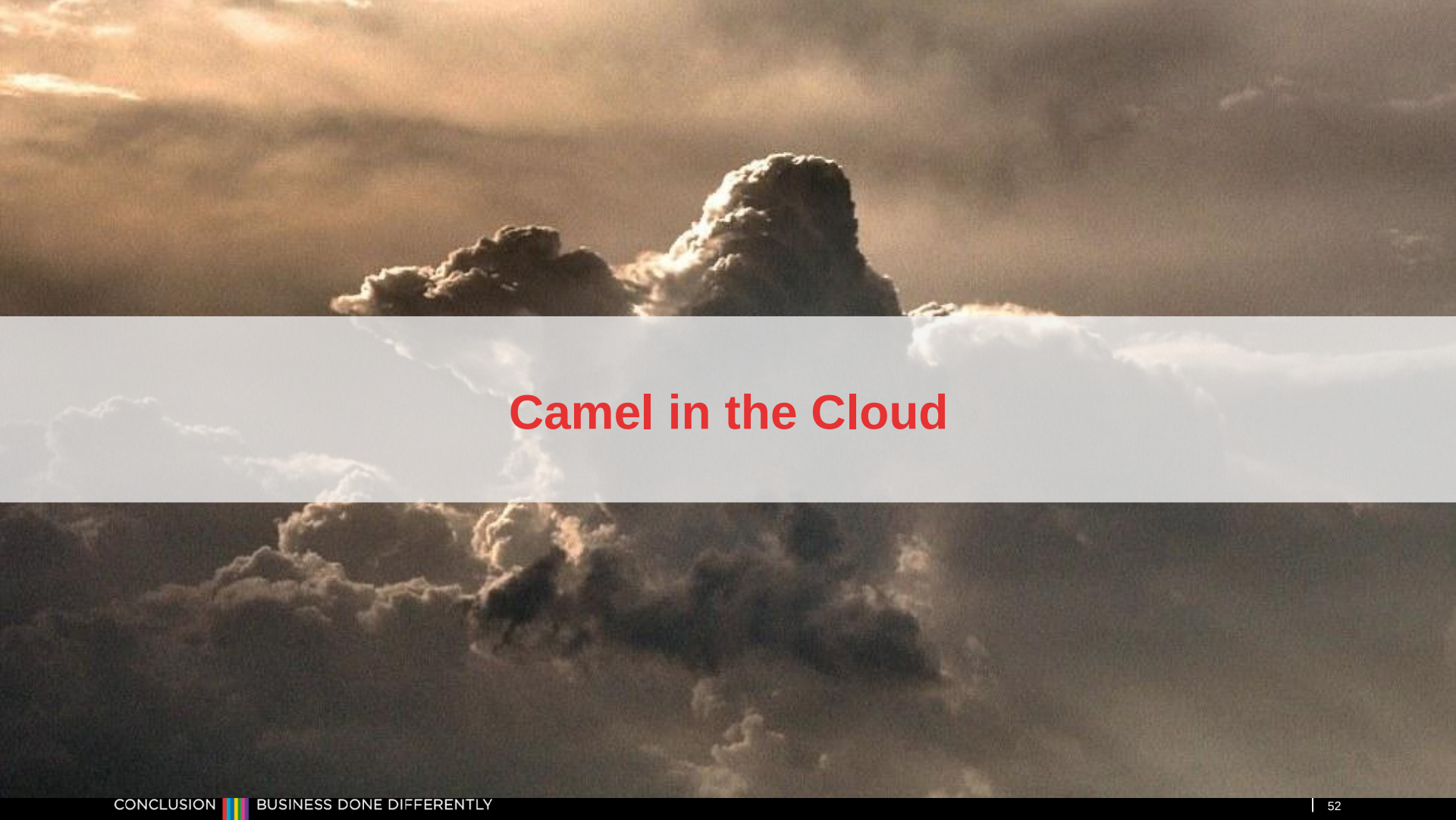
application.properties


```
# the name of Camel
camel.springboot.name = MyCamel

# what to say
greeting = Hello World

# how often to trigger the timer
timer.period = 2000

# to automatic shutdown the JVM after a period of time
#camel.springboot.duration-max-seconds=60
#camel.springboot.duration-max-messages=100

# add for example: &repeatCount=5 to the timer endpoint to make Camel idle
#camel.springboot.duration-max-idle-seconds=15

# expose actuator endpoint via HTTP
management.endpoints.web.exposure.include=info,health,camelroutes

# turn on actuator health check
management.endpoint.health.enabled = true

# allow to obtain basic information about camel routes (read only mode)
management.endpoint.camelroutes.enabled = true
management.endpoint.camelroutes.read-only = true

```

MySpringBootAplication.java

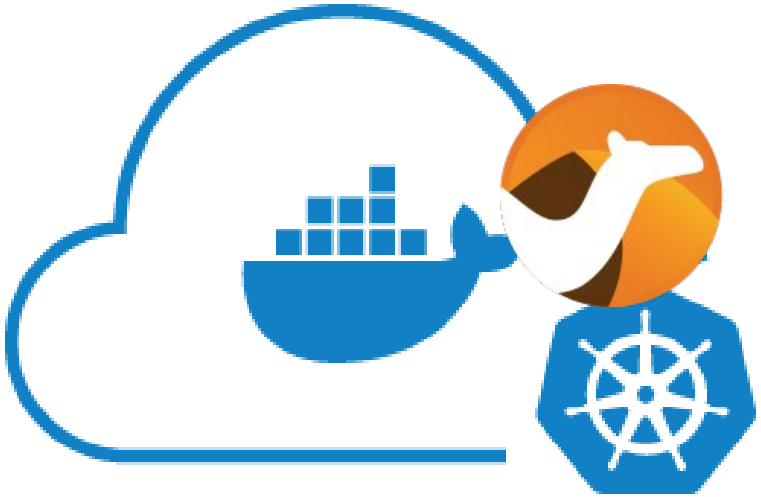

```java
package amis.camel;

import ...

@SpringBootApplication
public class MySpringBootApplication {

    /**
     * A main method to start this application.
     */
    public static void main(String[] args) { SpringApplication.run(MySpringBootApplication.class, args); }

}

```

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::        (v2.0.6.RELEASE)

2019-07-28 13:46:51.081  INFO 9024 --- [           main] amis.camel.MySpringBootApplication       : Starting MySpringBootApplication on localhost.localdomain with PID 9024

2019-07-28 13:46:53.159  INFO 9024 --- [           main] io.undertow.servlet                      : Initializing Spring embedded WebApplicationContext
2019-07-28 13:46:53.159  INFO 9024 --- [           main] o.s.web.context.ContextLoader            : Root WebApplicationContext: initialization completed in 2015 ms
2019-07-28 13:46:53.600  INFO 9024 --- [           main] o.a.c.i.converter.DefaultTypeConverter   : Type converters loaded (core: 195, classpath: 1)
2019-07-28 13:46:53.915  INFO 9024 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'characterEncodingFilter' to: [/*]
2019-07-28 13:46:53.916  INFO 9024 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'webMvcMetricsFilter' to: [/*]
2019-07-28 13:46:53.916  INFO 9024 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2019-07-28 13:46:53.916  INFO 9024 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'httpPutFormContentFilter' to: [/*]
2019-07-28 13:46:53.916  INFO 9024 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'requestContextFilter' to: [/*]
2019-07-28 13:46:53.917  INFO 9024 --- [           main] o.s.b.w.servlet.FilterRegistrationBean   : Mapping filter: 'httpTraceFilter' to: [/*]
2019-07-28 13:46:53.917  INFO 9024 --- [           main] o.s.b.w.servlet.ServletRegistrationBean  : Servlet dispatcherServlet mapped to [/]

2019-07-28 13:46:54.827  INFO 9024 --- [           main] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 3 endpoint(s) beneath base path '/actuator'

2019-07-28 13:46:54.902  INFO 9024 --- [           main] o.s.j.e.a.AnnotationMBeanExporter        : Registering beans for JMX exposure on startup
2019-07-28 13:46:54.947  INFO 9024 --- [           main] o.a.camel.spring.boot.RoutesCollector    : Loading additional Camel XML routes from: classpath:camel/*.xml
2019-07-28 13:46:54.947  INFO 9024 --- [           main] o.a.camel.spring.boot.RoutesCollector    : Loading additional Camel XML rests from: classpath:camel-rest/*.xml
2019-07-28 13:46:54.948  INFO 9024 --- [           main] o.a.camel.spring.SpringCamelContext      : Apache Camel 2.22.2 (CamelContext: MyCamel) is starting

2019-07-28 13:46:55.163  INFO 9024 --- [           main] o.a.camel.spring.SpringCamelContext      : Route: hello started and consuming from: timer://hello?period=2000
2019-07-28 13:46:55.163  INFO 9024 --- [           main] o.a.camel.spring.SpringCamelContext      : Total 1 routes, of which 1 are started
2019-07-28 13:46:55.164  INFO 9024 --- [           main] o.a.camel.spring.SpringCamelContext      : Apache Camel 2.22.2 (CamelContext: MyCamel) started in 0.216 seconds
2019-07-28 13:46:55.232  INFO 9024 --- [           main] o.s.b.w.e.u.UndertowServletWebServer     : Undertow started on port(s) 8080 (http) with context path ''
2019-07-28 13:46:55.236  INFO 9024 --- [           main] amis.camel.MySpringBootApplication       : Started MySpringBootApplication in 4.476 seconds (JVM running for 4.936)
Hello World
Hello World
Hello World
```


Camel in the Cloud

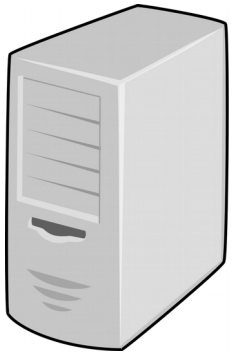
Camel runs everywhere

Application Servers



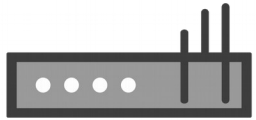
Docker and Kubernetes

Camel connects everything



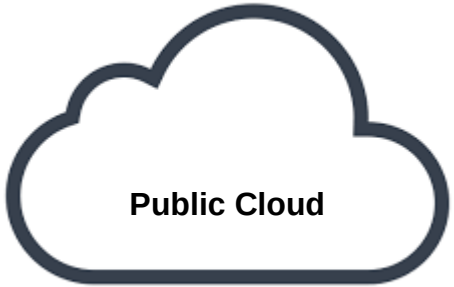
Legacy Systems

- File
- FTP
- JMS
- JDBC
- SQL
- TCP/UDP
- Mail
- JPA
- ...

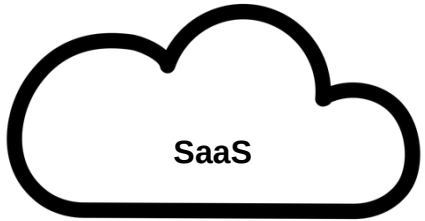


IoT

- CoAP
- MQTT
- PubNub

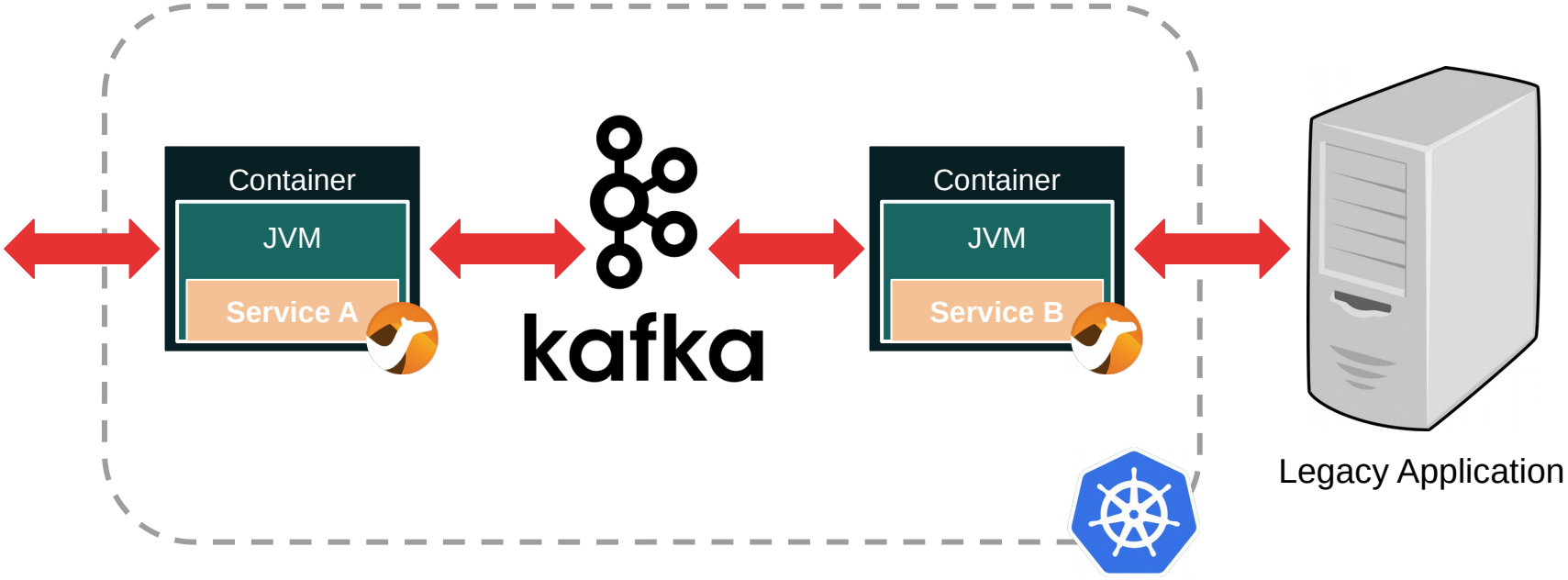


- Azure
  - azure-blob
  - azure-queue
- AWS
  - aws-s3
  - aws-kinesis
  - aws-sqs
  - ...
- Google
  - bigquery
  - pubsub



- Twitter
- Facebook
- LinkedIn
- SAP
- Salesforce
- ServiceNow
- ...

Agile Integration



Best Practice

extract the state:

camel-infinispan
camel-ignite
camel-hazelcast
kubernetes statefulsets
...

tracing:

camel-zipkin
camel-opentracing

service discovery:

camel-consul
camel-ribbon
camel-zookeeper
...

externalize config:

Kubernetes-ConfigMap
Kubernete-Secrets

resilience:

camel-hystrix

Three Pillars of Distributed Integration

DISTRIBUTED INTEGRATION

Lightweight
Patternbased
Event oriented

Flexibility

CONTAINERS

Cloud-native solutions
Independently deployable
Highly scalable

Scalability

API's

Management
Security and Optimization
Insights

Re-usability

What is next?

Showing posts with label **roadmap**. Show all posts

2019-06-25

Apache Camel 3 - camel-core vs camel-core-engine (smaller core)

The Camel team is very busy currently working on Apache Camel 3. A lot of work has already been implemented and we have released 3 milestone releases so far. The next milestone release number 4 has some great new innovative features which I will blog about in the following months.

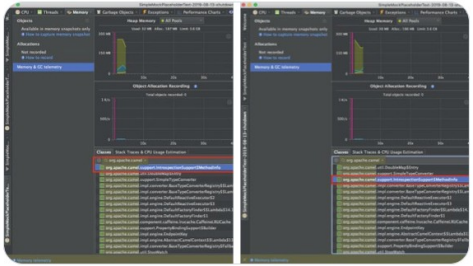
The topic of this blog is the work we have been doing on splitting up camel-core into smaller modules which you can now easily pick exactly only what you need.

If we take a look at the dependency tree of the camel-core JAR you can see that it's been split up into many modules as shown below:

```
[INFO] +- org.apache.camel:camel-core:jar:3.0.0-SNAPSHOT:compile
[INFO] |  +- org.apache.camel:camel-api:jar:3.0.0-SNAPSHOT:compile
[INFO] |  +- org.apache.camel:camel-base:jar:3.0.0-SNAPSHOT:compile
[INFO] |  +- org.apache.camel:camel-jaxp:jar:3.0.0-SNAPSHOT:compile
[INFO] |  +- org.apache.camel:camel-management-api:jar:3.0.0-
```

Claus Ibsen @davsclaus · 1d

Working on more optimizing @ApacheCamel core to reduce footprint on startup. Property placeholder now use fast source code generated configurer classes avoiding class introspection. The before vs after screenshots shows reduction of reflection caches in Camel #ApacheCamel



2 9 54

Show this thread

APACHE CAMEL K

Apache Camel K is a lightweight integration framework built from Apache Camel that runs natively on Kubernetes and is specifically designed for serverless and microservice architectures.

Users of Camel K can instantly run integration code written in Camel DSL on their preferred cloud (Kubernetes or OpenShift).

HOW IT WORKS

Just write a *helloworld.groovy* integration file with the following content:

```
from('timer:tick?period=3s')
  .setBody().constant('Hello world from Camel K')
  .to('log:info')
```

You can then execute the following command:

```
kamel run helloworld.groovy
```

The integration code immediately runs in the cloud. Nothing else is needed.

CAMEL QUARKUS

APACHE CAMEL EXTENSIONS FOR QUARKUS

This project hosts the efforts to port and package the 280+ Camel components as Quarkus extensions.

Quarkus is a Java platform offering fast boot times and low memory footprint. It targets both stock JVMs (OpenJDK in the first place) and GraalVM.

For more information, please refer to the Camel Quarkus Documentation or check the Camel Quarkus code on GitHub.

Edit this Page

Overview

- News
- Components
- Download
- Getting started
- FAQ

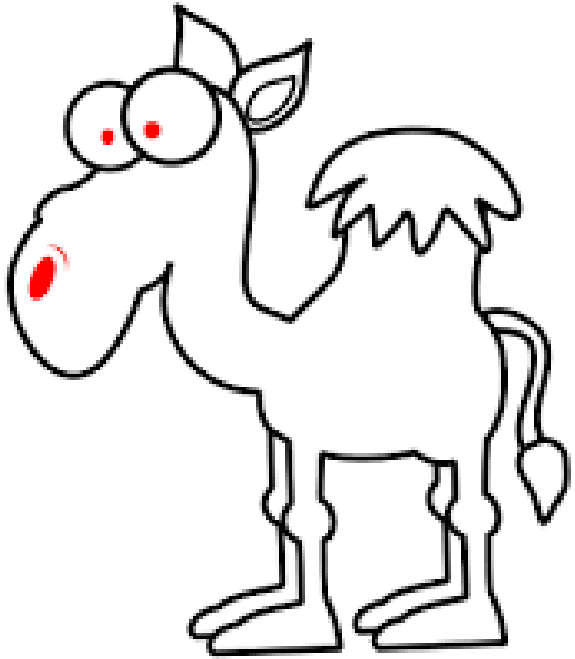
Community

- Support
- Contributing
- User stories
- Articles
- Books
- Team
- Camel extra

About

- Acknowledgments
- Apache Events
- License
- Security
- Sponsorship
- Thanks

Questions



CONCLUSION