APACHE Camel

Integrating your applications easily

‹from uri="whoami"›

```
<log id="ids" message="rmarting, jromanmartin" />
<log id="name" message="Jose Roman Martin Gil" />
<log id="role" message="Principal Middleware Architect" />
<log id="company" message="Red Hat" />
<log id="labels" message="father, husband, friend, runner,
curious, Red Hatter, developer, integrator (in any order)" />

<to uri="mailto:rmarting@redhat.com" />
<to uri="GitHub:https://github.com/rmarting" />
<to uri="Twitter:https://twitter.com/jromanmartin" />
<to uri="LinkedIn:https://www.linkedin.com/in/jromanmartin/" />
```

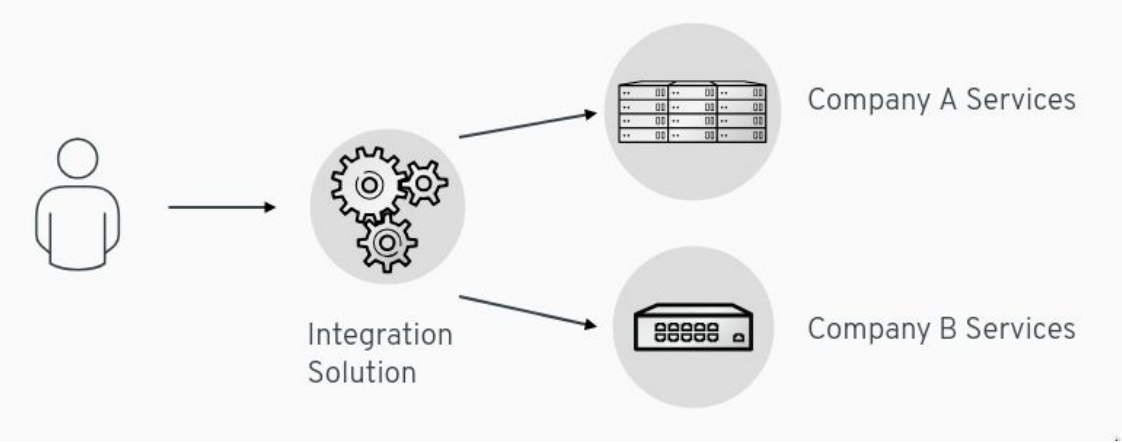
SYSTEM INTEGRATION

System Integration

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

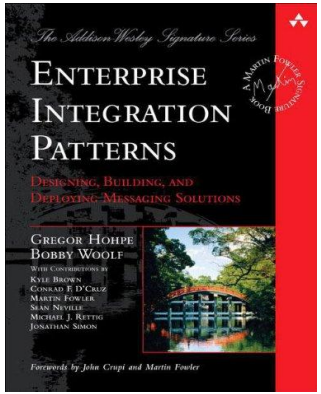
Why Integration?

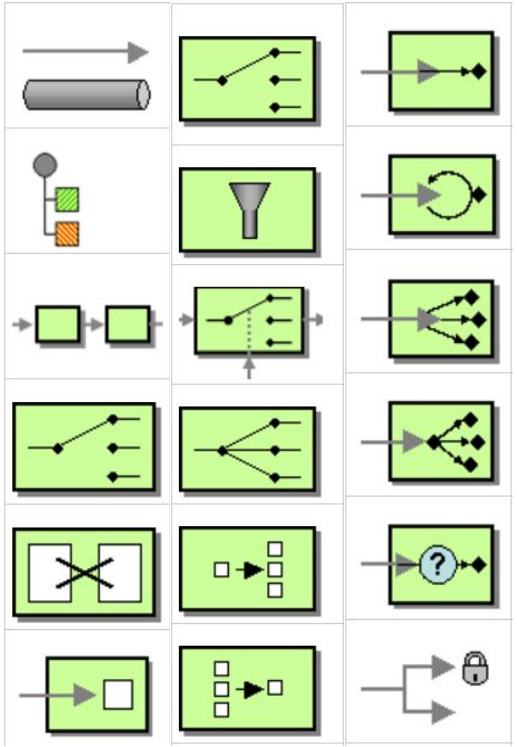
- Critical for business
- Growth of an enterprise by acquisitions and fusions
- New values are created by combinations of existing products
- Different subsystems use different technologies or languages
- Incremental legacy application replacements



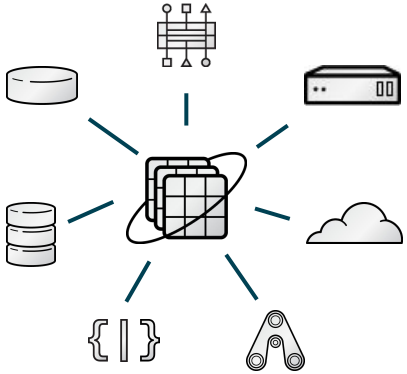
Enterprise Integration Patterns

- Recipes for solving integration problems
- Proven design patterns and recipes for common integration problems
- Patterns were "Harvested" from a study of thousands of Integration projects.
- Describes integration problems, solutions and also provide common vocabulary and diagram notations
- A book by Gregor Hohpe and Bobby Woolf
  - http:/www.eaipatterns.com



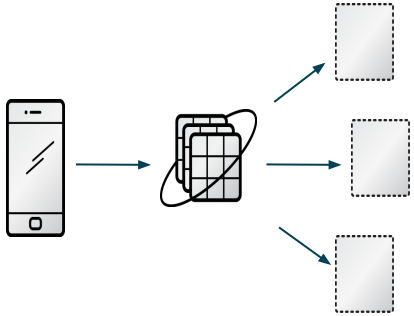


Multiple Integration Styles



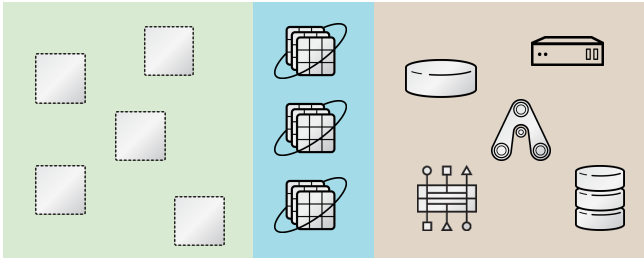
TRADITIONAL INTEGRATION

Pattern-oriented integration for on-premise and cloud-based resources.



INTEGRATION MICROSERVICES

Create and compose microservices using API and event-driven interactions.



TRANSITIONAL INTEGRATION

Blend greenfield and brownfield to deliver next generation services.

INTEGRATION FRAMEWORK

Why Integration Framework?

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

APACHE
Camel

APACHE CAMEL

What is Apache Camel?

- Versatile Open-Source integration framework based on known Enterprise Integration Patterns (Apache Camel Web Site)

- Open-Source Java framework that focused on making integration easier and more accessible to developers. It does this by providing:
  - Concrete implementations of all the widely used EIPS
  - Connectivity to a great variety of transports and APIs,
  - Easy to use Domain Specific Languages (DSLs) to wire EIPs and transports together

Jonathan Anstey (Author *Camel in Action*)

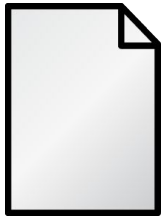
Apache Camel Provides

ENTERPRISE INTEGRATION PATTERNS

Build integrations using enterprise best practices.

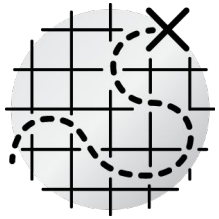
200+ COMPONENTS

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



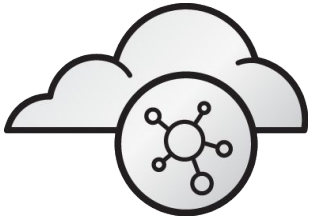
BUILT-IN DATA TRANSFORMATION

JSON, XML, HL7, YAML, SOAP, Java, CSV, Custom



INTUITIVE FRAMEWORK AND TOOLING

Build integrations quickly and easily without lock-in

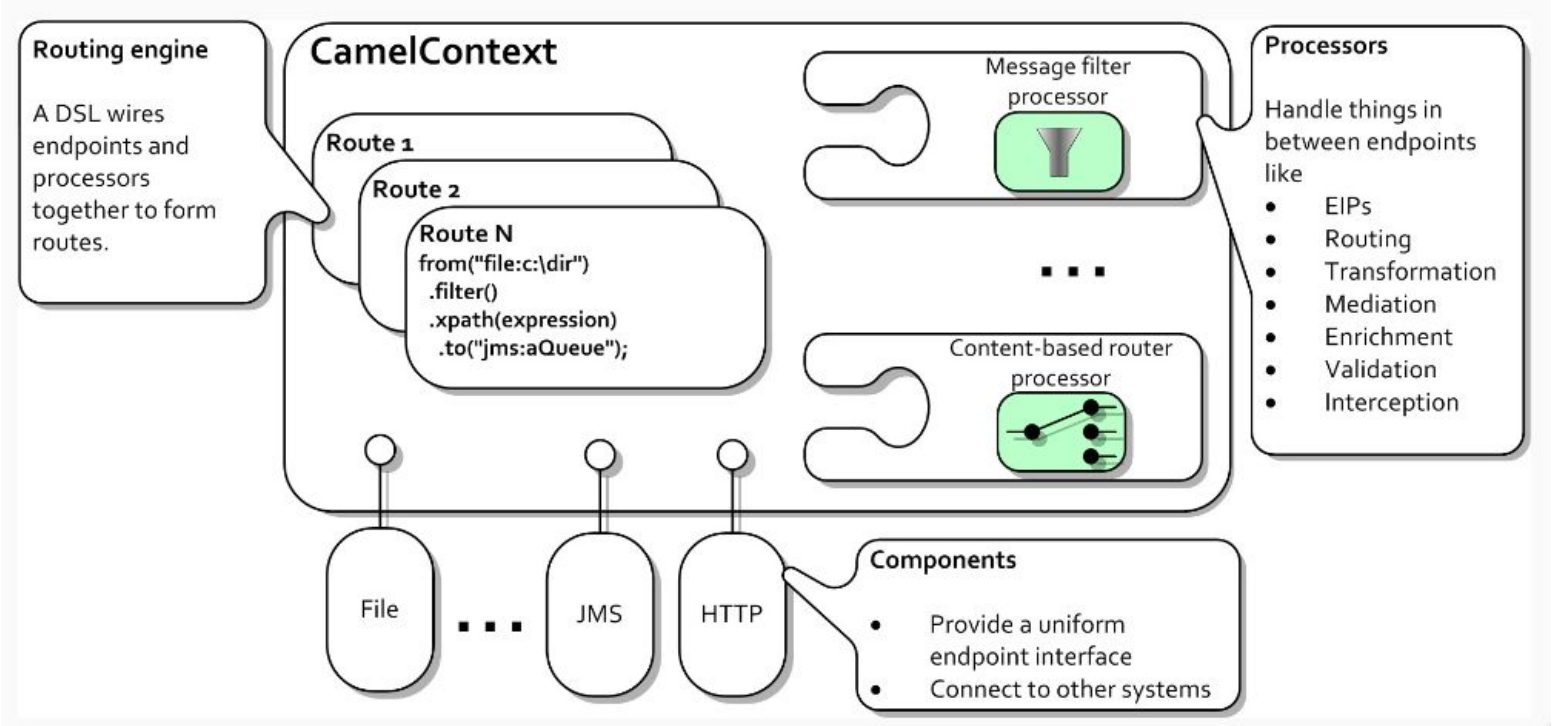


NATIVE REST SUPPORT

Create, connect, and compose APIs with ease.

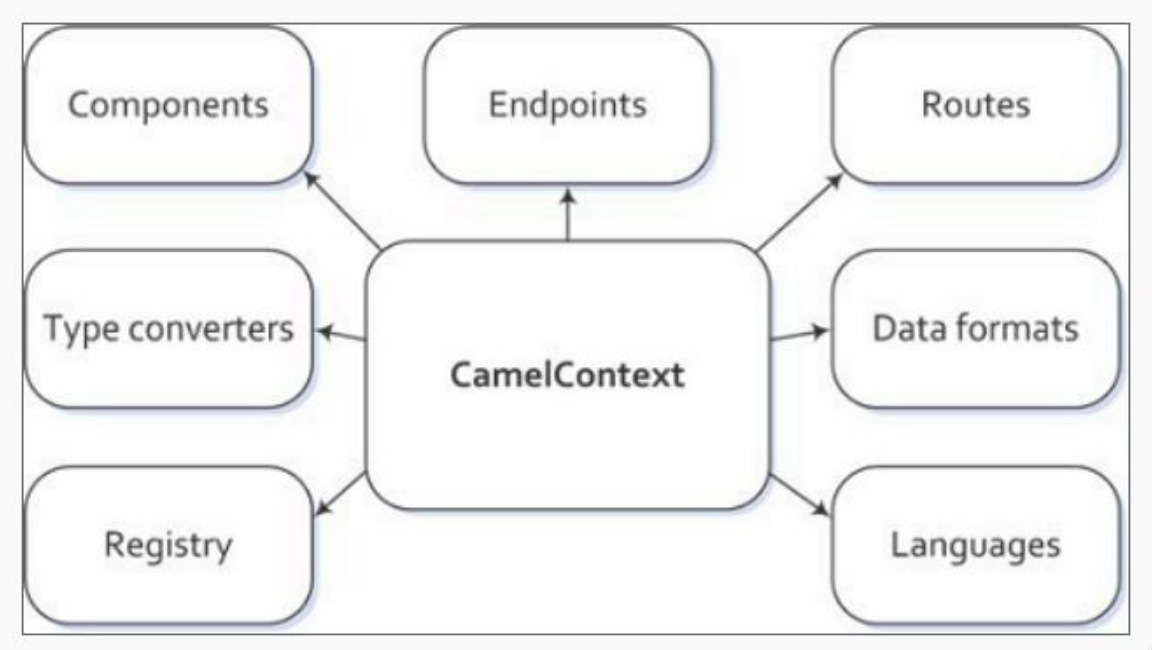
APACHE CAMEL ARCHITECTURE

Apache Camel Architecture



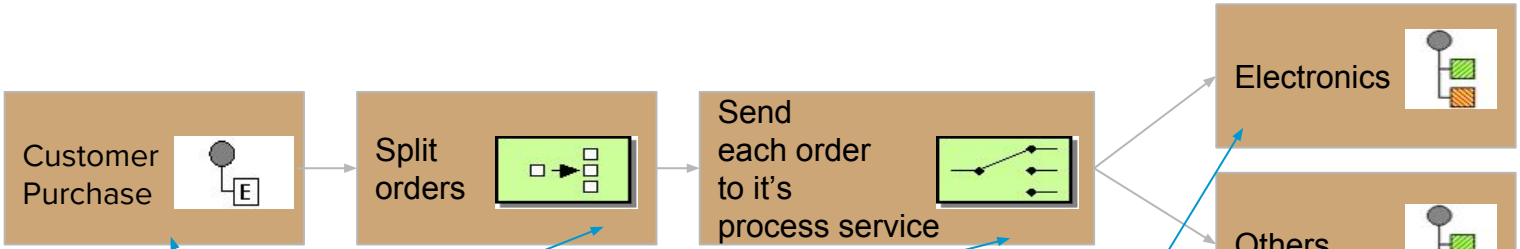
Apache Camel Context

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



Apache Camel Route

- Integration pipeline between an Consumer and Producers

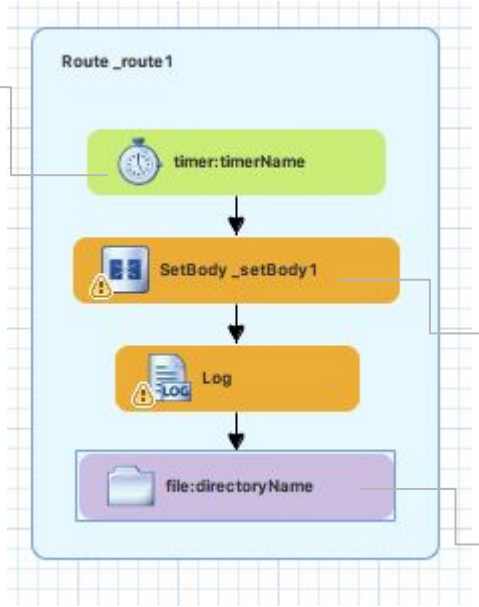



```
from("file:work/cbr/input")
  .split(xpath("//orders"))
    .choice()
      .when(xpath("/order:order/order:type = 'E'"))
        .to("amqp:queue:electronic/us")
      .otherwise()
        .recipientList(simple("http4://otherservice"));
```

Apache Camel Route

Consumer

- Consume requests
- Start of a route
- Dispatching outgoing replies



Processor

- Intermediate node in the pipeline
- standard processors or customized ones

Producer

- Produce requests
- End of route
- Dispatching outgoing requests

Apache Camel Route - XML vs JavaDSL

```xml
<route id="cbr-route">
 <from id="_from1" uri="file:work/cbr/input" />
 <split id="_split1">
   <xpath>//orders</xpath>
   <choice id="_choice1">
     <when id="_when1">
       <xpath>/order:order/order:type='E'</xpath>
       <to id="_to1" uri="amqp:queue:electronic"/>
     <otherwise id="_otherwise1">
       <recipientList id="_recipientList1">
         <simple>http4://otherservice</simple>
       </recipientList>
   </choice>
 </split>
</route>
```

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

Which Camel DSL to Choose and Why?

Apache Camel Endpoint

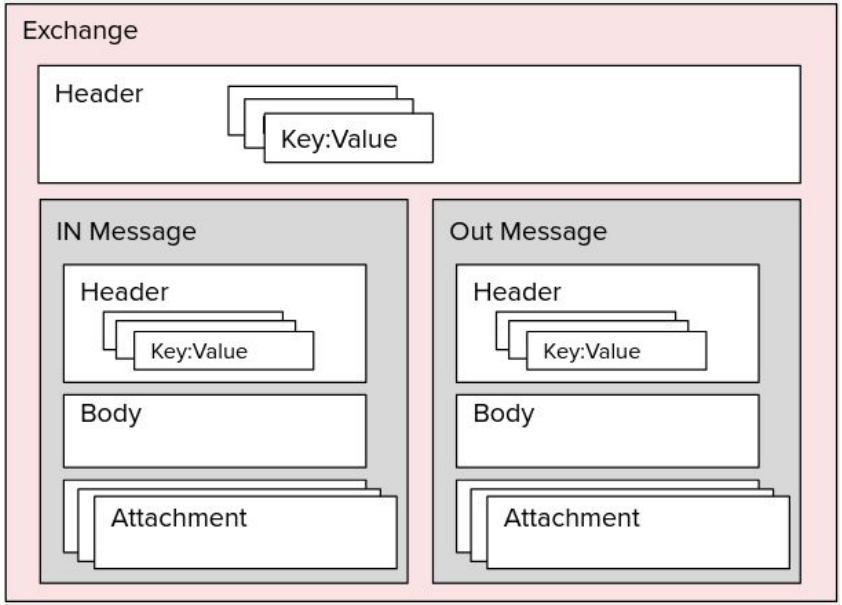
- Represents endpoint which is capable of sending and receiving (producing and consuming) messages e.g. FTP server, a Web Service or a JMS broker
- Described by URIs:
  - schema:context/path?options
    - schema = identifies component
    - context/path = identifies location of a resource or destination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

Apache Camel Endpoint Roles

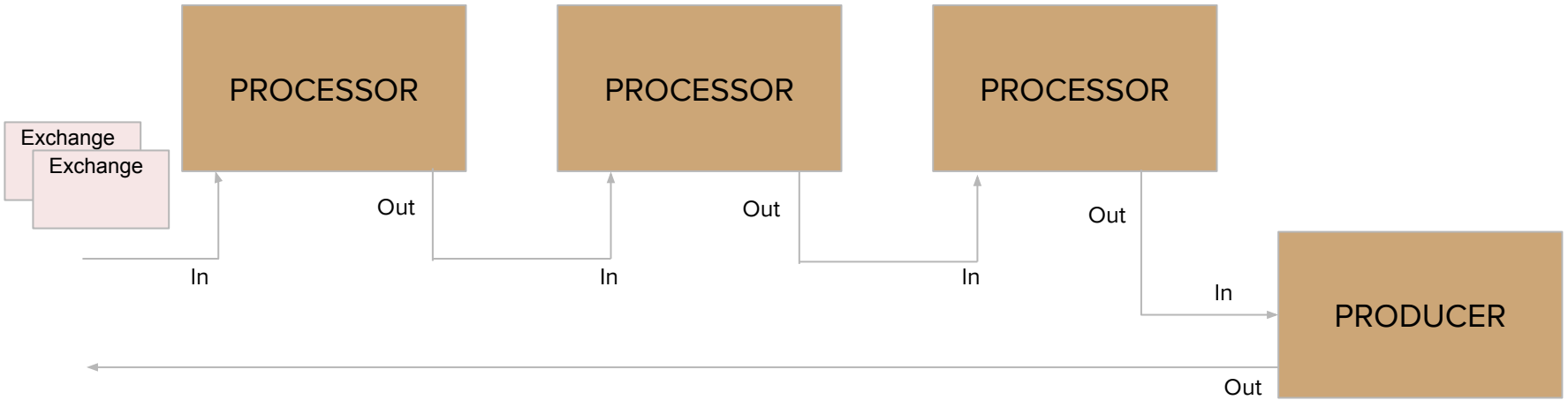
- **Consumer (from)**
  - receives messages from an external source and creates a message exchange object, which the routing rule processes
  - event-driven consumer - waits until message arrives e.g. JMS, HTTP, tcp, udp
  - polling consumer - actively checks for new messages e.g. FTP, file, email
-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

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 Exchange in Route

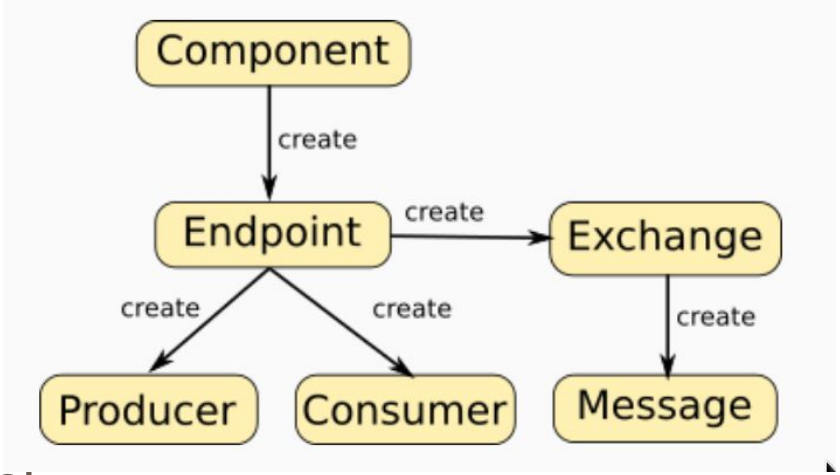


Apache Camel Processor

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

Apache Camel Component

- Main extension point in Camel
- Contains configurations for Endpoints
- Factory for Endpoint instances
- +200 Endpoint Components



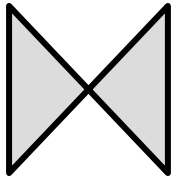
activemq	cxf	kubernetes	jasypt
activemq-journal	cxfrs	freemarker	javaspace
amqp	dataset	ftp/ftps/sftp	jbi
atom	db4o	gae	jcr
bean	direct	hdfs	jdbc
bean validation	ejb	hibernate	jetty
browse	esper	hl7	jms
cache	event	http	jmx
cometd	exec	ibatis	jpa
crypto	file	irc	jt/400

Data Transformation

- Data format transformation
  - the data format of message body is transformed from e.g. CSV to formatted XML
- Data type transformation
  - the data type of the message body is transformed
    - java.lang.String -> javax.jms.TextMessage
  - automatic type converter mechanism
- Different Data Formats
  - bindy, csv, json, xml, jaxb, hl7, zip, ...

Converting between Data Format

- Marshal
  - Java Bean → Textual format
- Unmarshal
  - Textual, Binary format → Java Bean
- Dozer
  - Fine-grained integration
  - mapping literal values
  - Use expressions



Sample Data Transformation

Input XML File:

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

Camel Route:

```
...

from("file:///xmlsourcedir")
  .unmarshal().jaxb()
  .process(...)
  .marshal().json()
  .to("file:///jsondestdir");

...
```

Output JSON File:

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

Error Handling

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

```
// Generic error handler
errorHandler(deadLetterChannel("errors").maximumRedeliveries(1));

// Special error handling for validation errors
onException(ValidationException.class)
      .to("amqp:queue:validation");

from("file:work/cbr/input")
 .onException(ShipOrderException.class)
   .handled(true)
   .bean(ShipService.class, "shipFailed")
   .end()
 .split(xpath("//orders"))
   …;
```

Security

- Route Security
  - Authentication and Authorization services to proceed on a route or route segment
- Configuration Security
  - Camel allows to crypt/decrypt configuration files containing sensitive information
- Endpoint Security
  - Security offered by components through uri endpointUri associated with the component
- Payload Security
  - Data Formats that offer encryption/decryption services at the payload level

INTEGRATING APPLICATIONS

API and REST DSL

Verb defining http method

```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <rest path="/say">
    <get uri="/hello">
      <to uri="direct:hello"/>
    </get>
    <get uri="/bye" consumes="application/json">
      <to uri="direct:bye"/>
    </get>
    <post uri="/bye">
      <to uri="mock:update"/>
    </post>
  </rest>
  <route>
    <from uri="direct:hello"/> …
  </route>
  <route>
    <from uri="direct:bye"> …
  </route>
</camelContext>
```

Basepath
The service path

Consumes
Accept data format setting

Uri template
The service method and parameters

REST DSL

Message Body	Direction	Binding Mode	Message Body
XML	Incoming	auto, xml, json_xml	POJO
POJO	Outgoing	auto, xml, json_xml	XML
JSON	Incoming	auto, xml, json_xml	POJO
POJO	Outgoing	auto, xml, json_xml	JSON

```
<restConfiguration bindingMode="auto" component="servlet" port="8080"/>
```

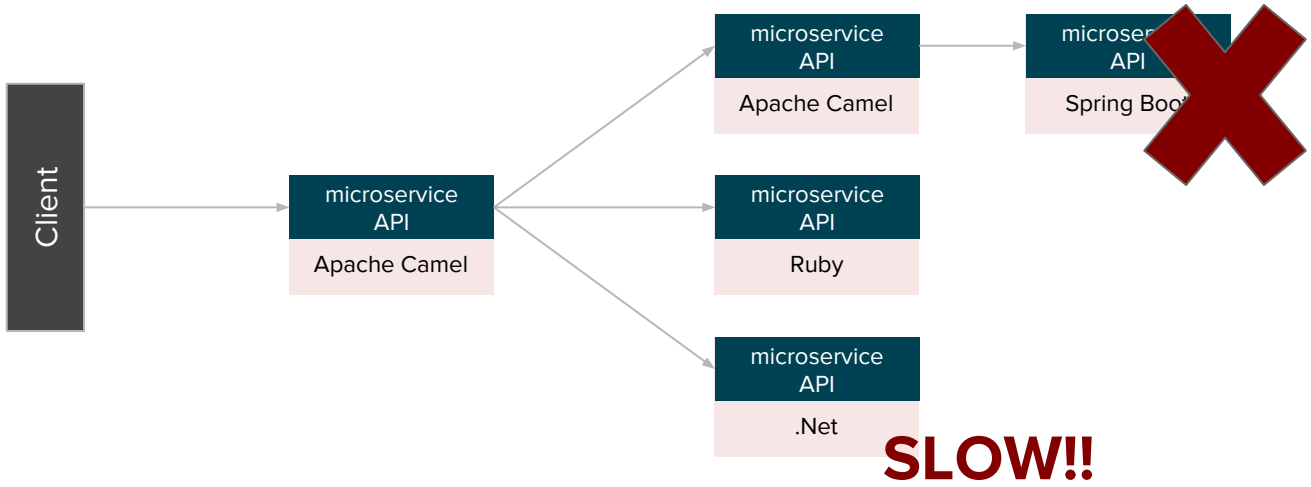
- camel-netty4-http
- camel-jetty
- camel-servlet
- camel-undertow

Swagger

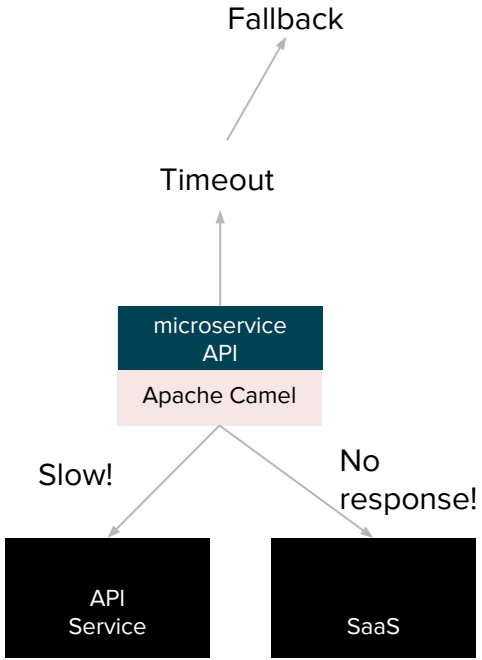
```
<restConfiguration apiContextPath="api-docs" bindingMode="json" component="servlet"
     contextPath="/demos">
          <apiProperty key="cors" value="true"/>
          <apiProperty key="api.title" value="API for demo"/>
          <apiProperty key="api.version" value="1.0.0"/>
</restConfiguration>
```

```
<get uri="/{id}" outType="com.redhat.User">
  <description>Find user by id</description>
  <param name="id" type="path" description="The id of the user to get" dataType="int"/>
  <to uri="bean:userService?method=getUser(${header.id})"/>
</get>
```

Service Resilience



Circuit Breaker




```
<camelContext xmlns="http://camel.apache.org/schema/spring">
  <route>
    <from uri="direct:start"/>
    <hystrix>
      <to uri="http://fooservice.com/slow"/>
      <onFallback>
        <transform>
          <constant>Fallback message</constant>
        </transform>
      </onFallback>
    </hystrix>
    <to uri="mock:result"/>
  </route>
</camelContext>
```

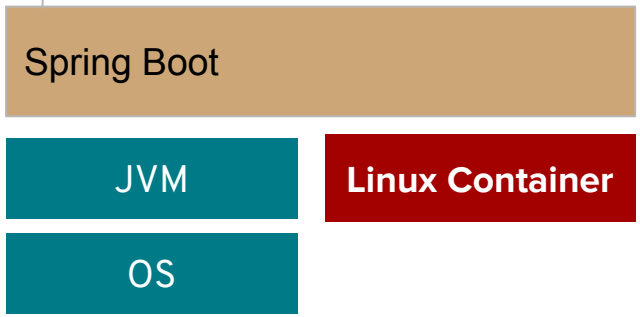
RUNNING APACHE CAMEL

Deploying Apache Camel

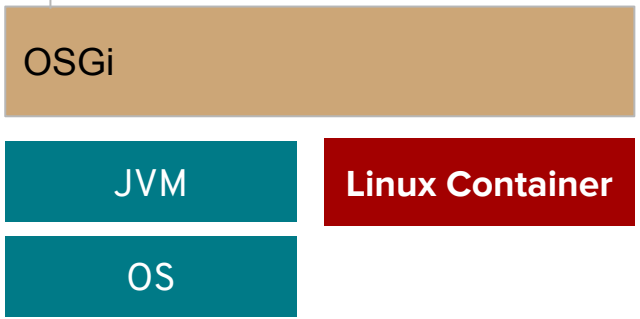
- Standalone JAR
- WAR - Servlet Container, e.g. Apache Tomcat, Jetty
- Spring - Spring Boot
- Java EE - e.g. Wildfly, Glassfish, WebLogic, WebSphere
- OSGi Container - e.g. Apache Karaf, ServiceMix
- Cloud - e.g. OpenShift, Kubernetes, Google Compute Engine, Amazon EC2

Spring Boot vs OSGi

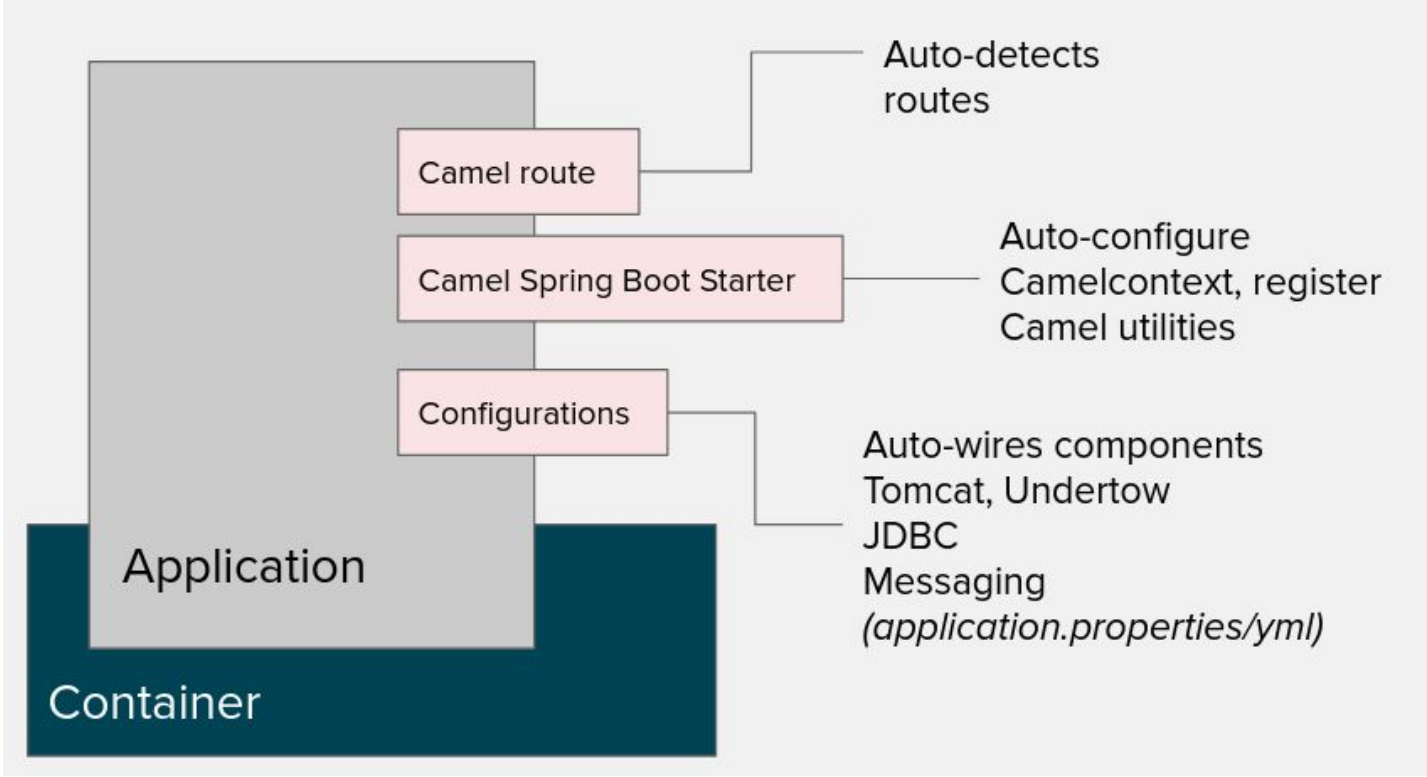
Spring DSL/Java
Spring Boot Starter module
Fat JARs
Stand-alone App
Embedded dependency
Pre-configured, pre-sugared
Small and lightweight



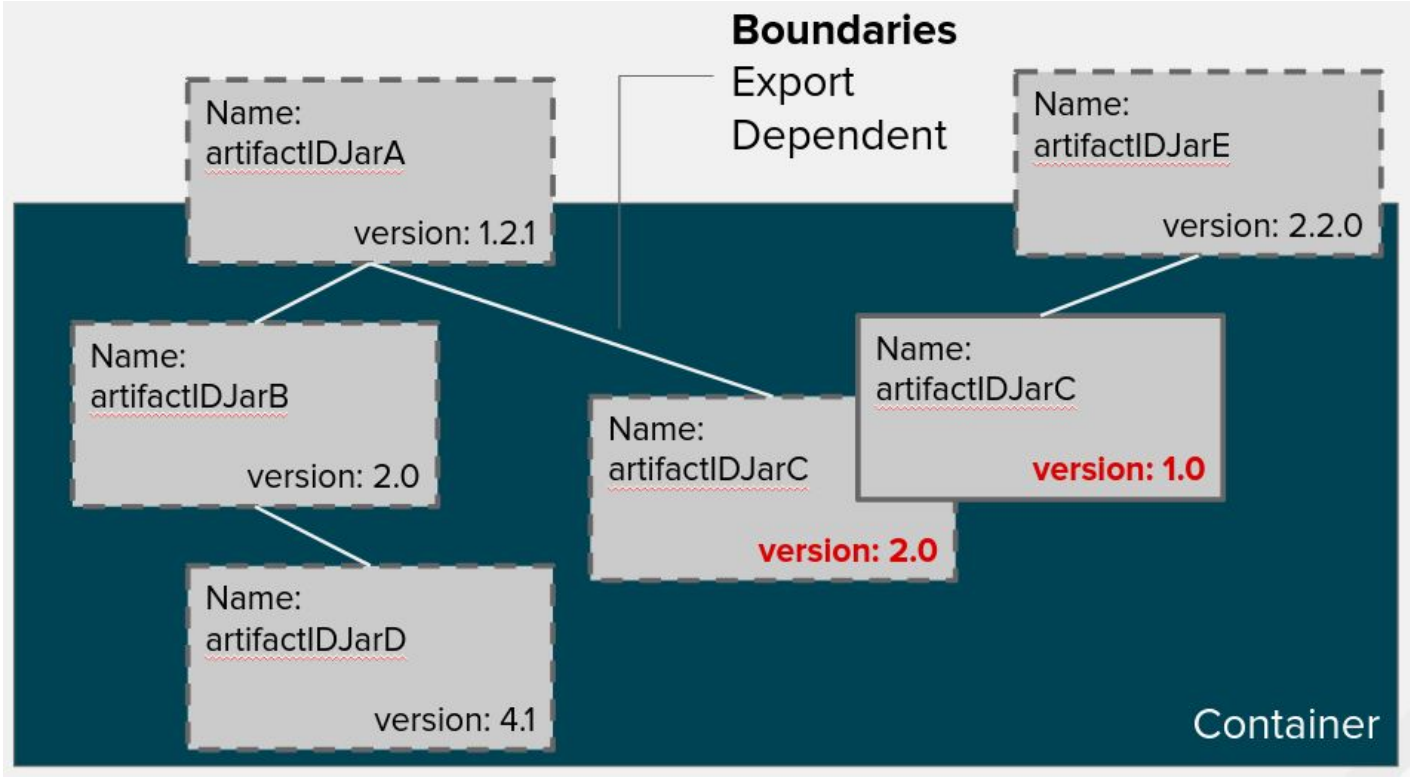
Blueprint DSL/Java
Blueprint module
Bundles
Modularized
Explicit dependency
Versioned
Hot redeploy



Spring Boot

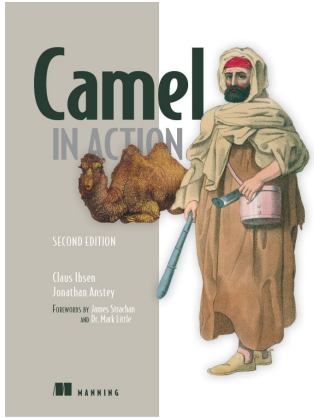

Auto-detects
routes
Camel route
Camel Spring Boot Starter
Auto-configure
Camelcontext, register
Camel utilities
Configurations
Auto-wires components
Tomcat, Undertow
JDBC
Messaging
(application.properties/yml)
Application
Container

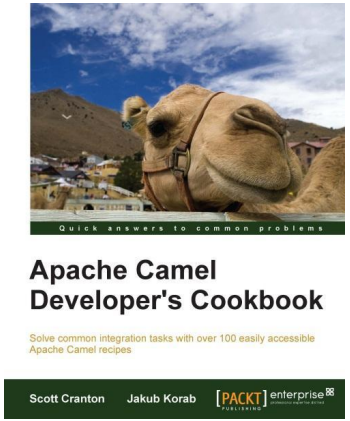
OSGi

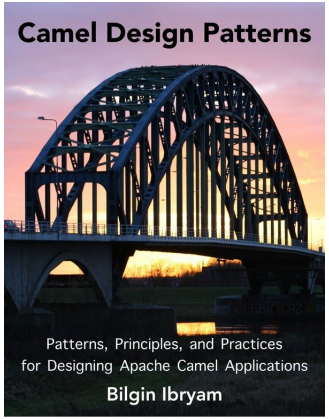

Boundaries
Export
Dependent
Name:
artifactIDJarA
version: 1.2.1
Name:
artifactIDJarE
version: 2.2.0
Name:
artifactIDJarB
version: 2.0
Name:
artifactIDJarC
version: 1.0
Name:
artifactIDJarC
version: 2.0
Name:
artifactIDJarD
version: 4.1
Container

More Information

- Camel in Action
- Apache Camel Developer's Cookbook
- Camel Design Patterns
- Community website
  - http://camel.apache.org/
  - https://github.com/apache/camel







- Spring Boot Sample: https://github.com/rmarting/fis-workshop

open south code

Questions?

APACHE Camel

open south code

Thank you!

APACHE Camel