



Riding a Camel through the JEEHara

Markus Eisele, @myfear
Developer Advocate
markus@jboss.org
October, 2015



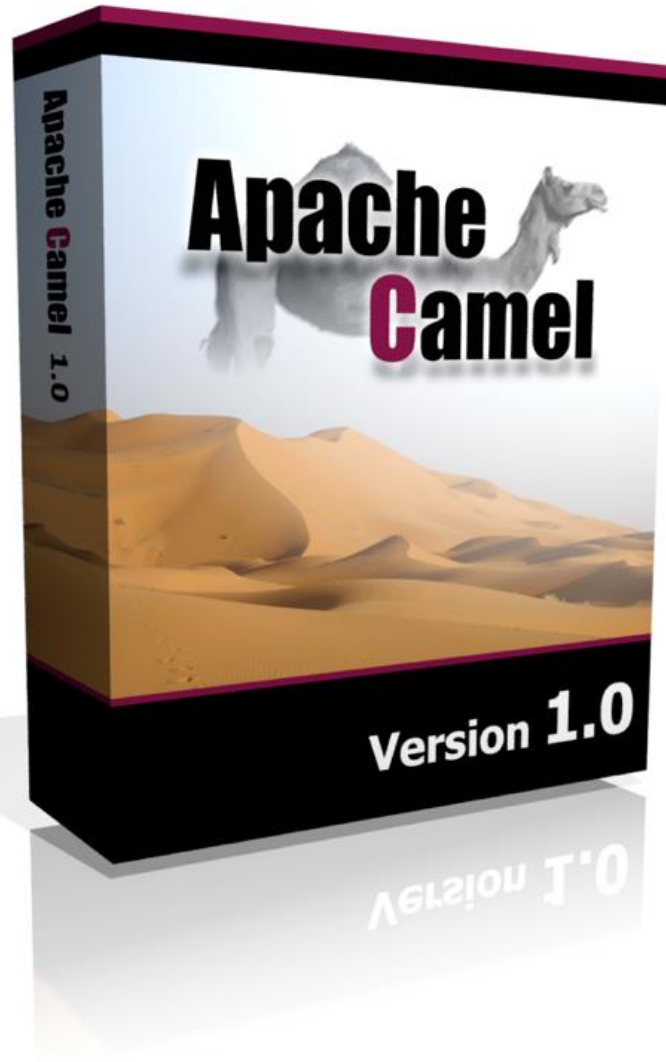
WildFly 

WildFly 

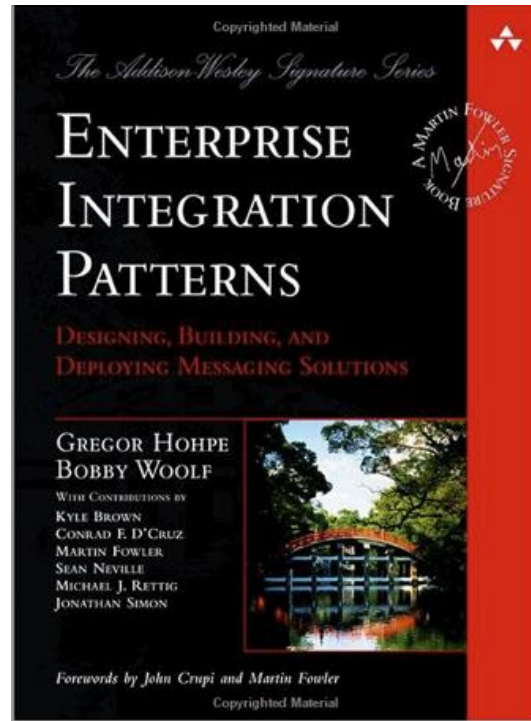
WildFly 

WildFly 

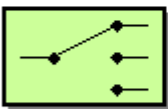
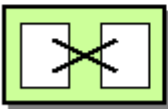
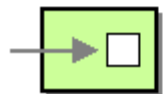
WildFly 

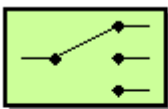
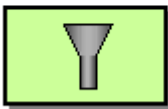
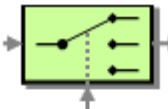
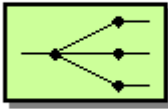
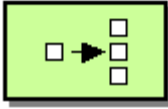
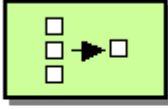
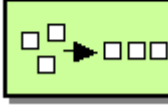
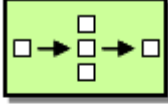


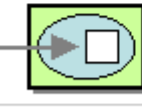


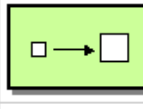
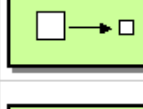
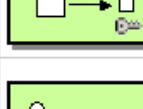


Gregor Hohpe (@ghohpe)
and
Bobby Wolf

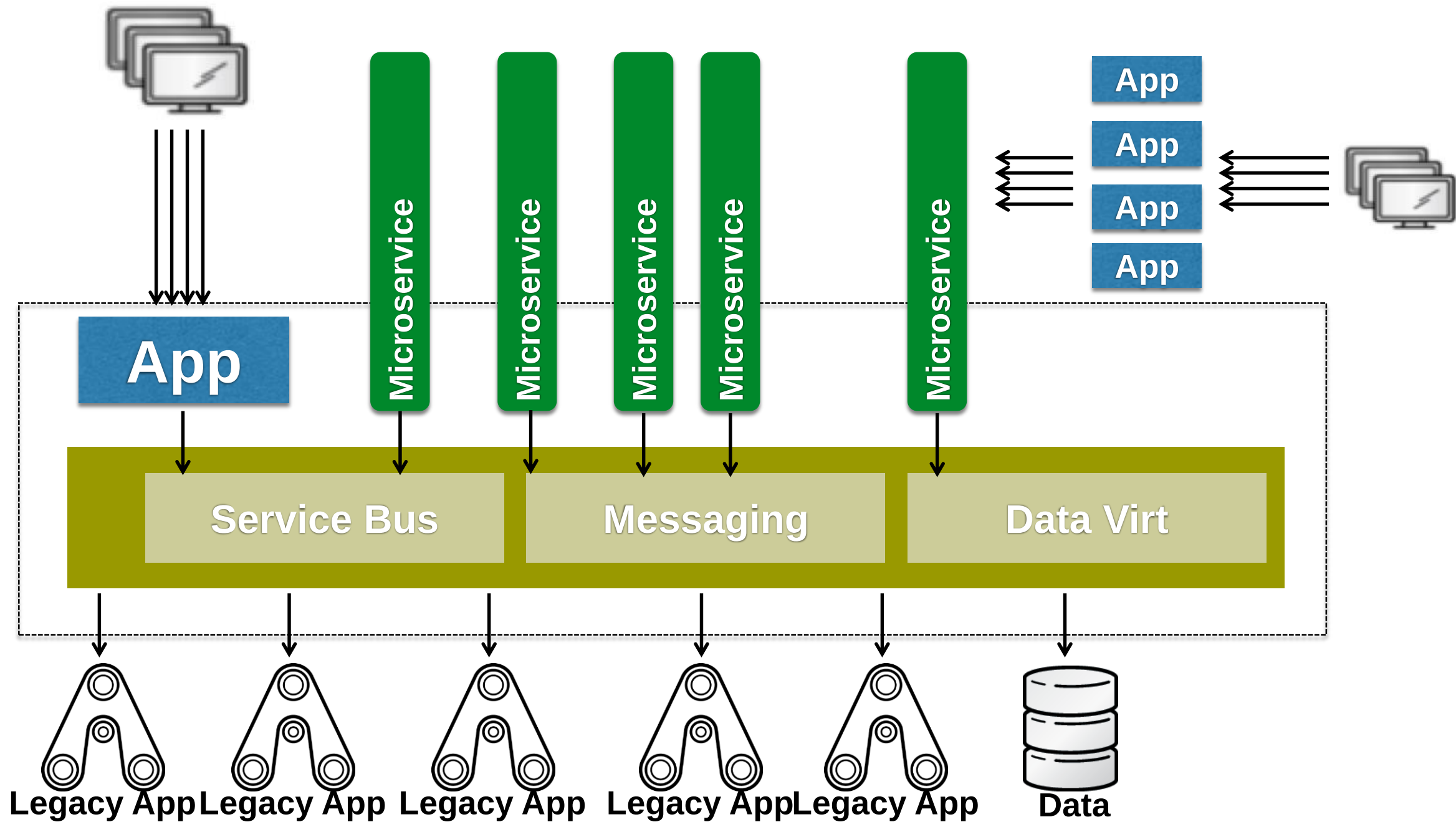
	Message Channel
	Message
	Pipes and Filters
	Message Router
	Message Translator
	Message Endpoint

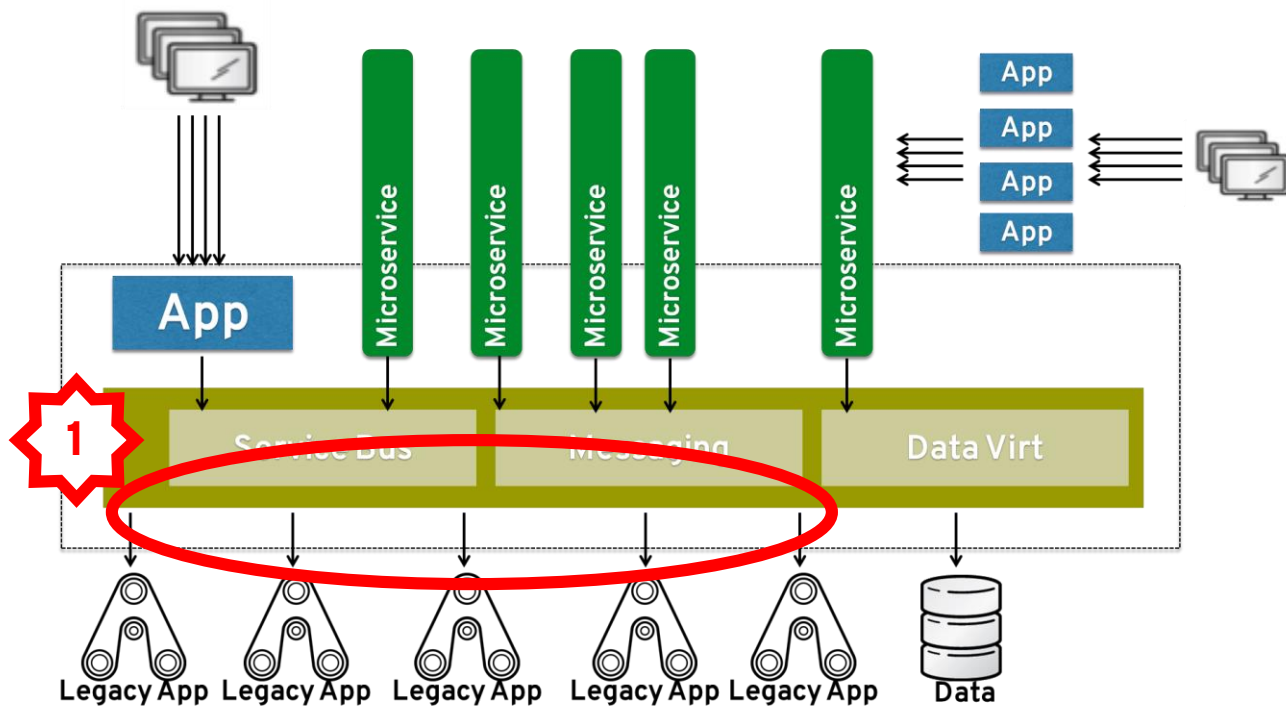
	Content Based Router
	Message Filter
	Dynamic Router
	Recipient List
	Splitter
	Aggregator
	Resequencer
	Composed Message Processor
	Scatter-Gather
	Routing Slip

	Messaging Mapper
	Event Driven Consumer
	Polling Consumer
	Competing Consumers
	Message Dispatcher
	Selective Consumer
	Durable Subscriber
	Idempotent Consumer
	Transactional Client
	Messaging Gateway
	Service Activator

	Content Enricher
	Content Filter
	Claim Check
	Normalizer
	Sort
	Validate

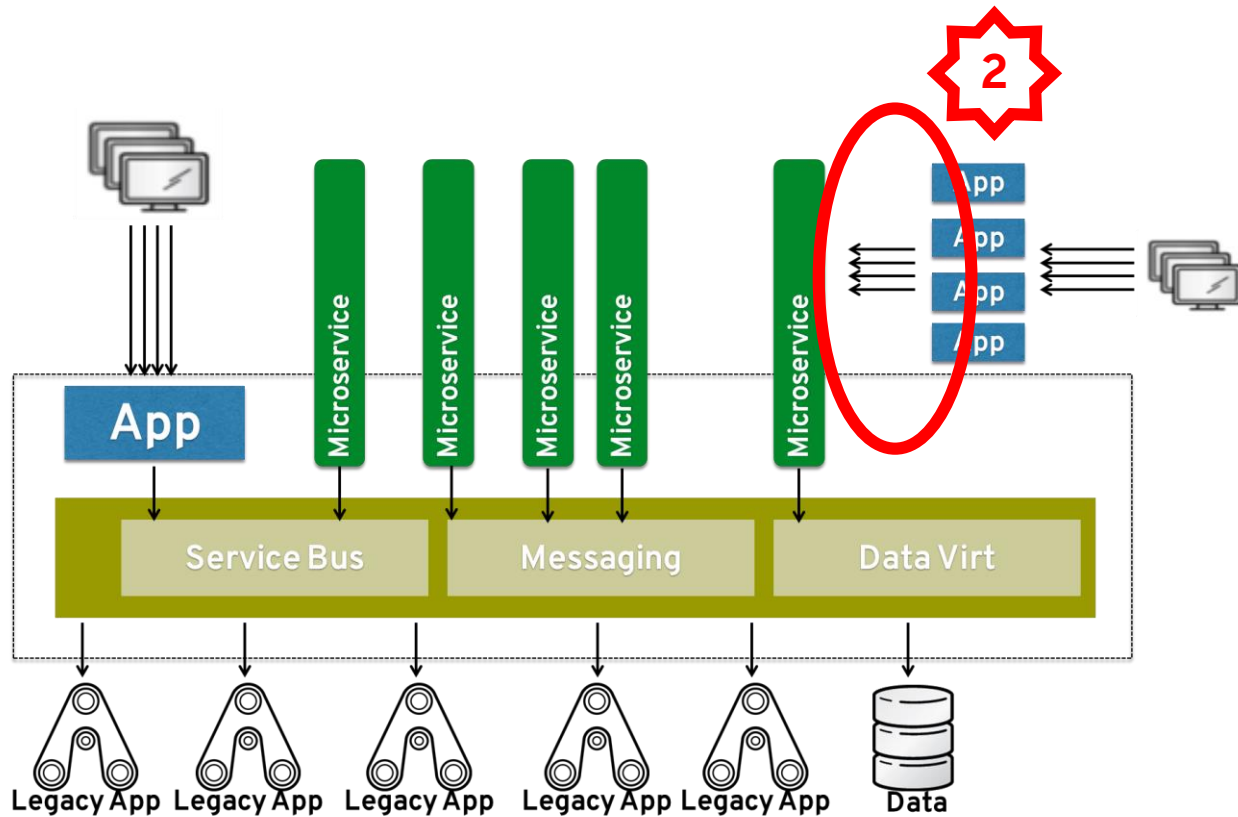
Why Camel?





1) More Services Produced

- Legacy Apps
 - Newly Designed
 - Infrastructure Level
-
- Different Protocols
 - Custom and Standard Apps



2) More Services Consumed

- More Apps
 - Decentralized
 - Chained Services
-
- Lightweight Connectors
 - Standardized Protocols
 - EAI Problems move to apps
 - Monitoring
 - Distributed Responsibility

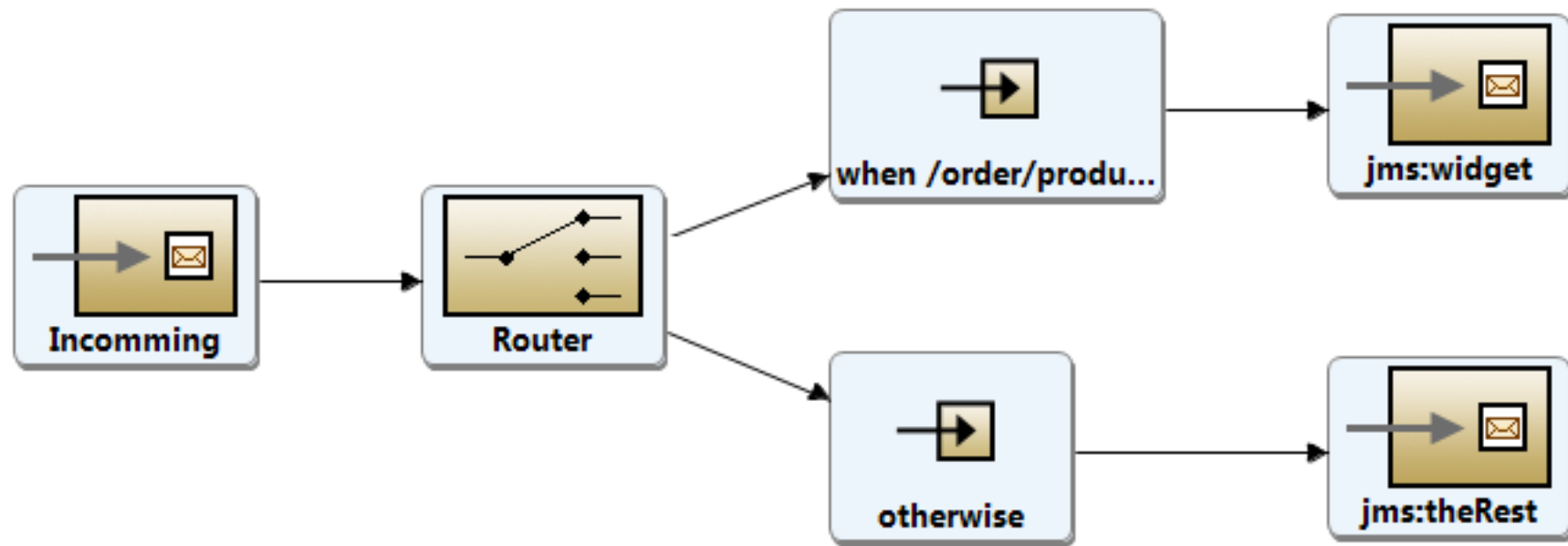
Applications are done by Developers. So, pleeeeaassssee ...

Easy & Lightweight

Therefore Camel!

- **Open source integration library**
- **Simple lightweight, use in any JVM or application server**
- **Can be programmed and not sketched**
- **Connects to anything**
 - **100s of components connecting to databases, file systems, messaging systems, big data, SaaS providers, social media, ...**
- **Tooling**

How? Show me!



```

package org.javaee.why;

import javax.annotation.Resource;
import javax.ejb.MessageDriven;
import javax.jms.Connection;
import javax.jms.ConnectionFactory;
import javax.jms.DeliveryMode;
import javax.jms.JMSException;
import javax.jms.Message;
import javax.jms.MessageListener;
import javax.jms.MessageProducer;
import javax.jms.Queue;
import javax.jms.Session;
import javax.jms.TextMessage;

@MessageDriven
public class SortOrdersBean implements MessageListener {

    @Resource
    private ConnectionFactory connectionFactory;

    @Resource(name = "WidgetQueue")
    private Queue widget;

    @Resource(name = "TheRestQueue")
    private Queue theRest

    public void onMessage(Message message) {
        try {

            final TextMessage order = (TextMessage) message;
            final String type = order.getText();

            if ("widget".equals(type)) {

                respond("widget", message);
            } else {
                respond("rest", message);
            }
        } catch (JMSException e) {
            throw new IllegalStateException(e);
        }
    }

    private void respond(String type, Message message) throws JMSException {

        Connection connection = null;
        Session session = null;

        try {
            connection = connectionFactory.createConnection();
            connection.start();

            // Create a Session
            session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);

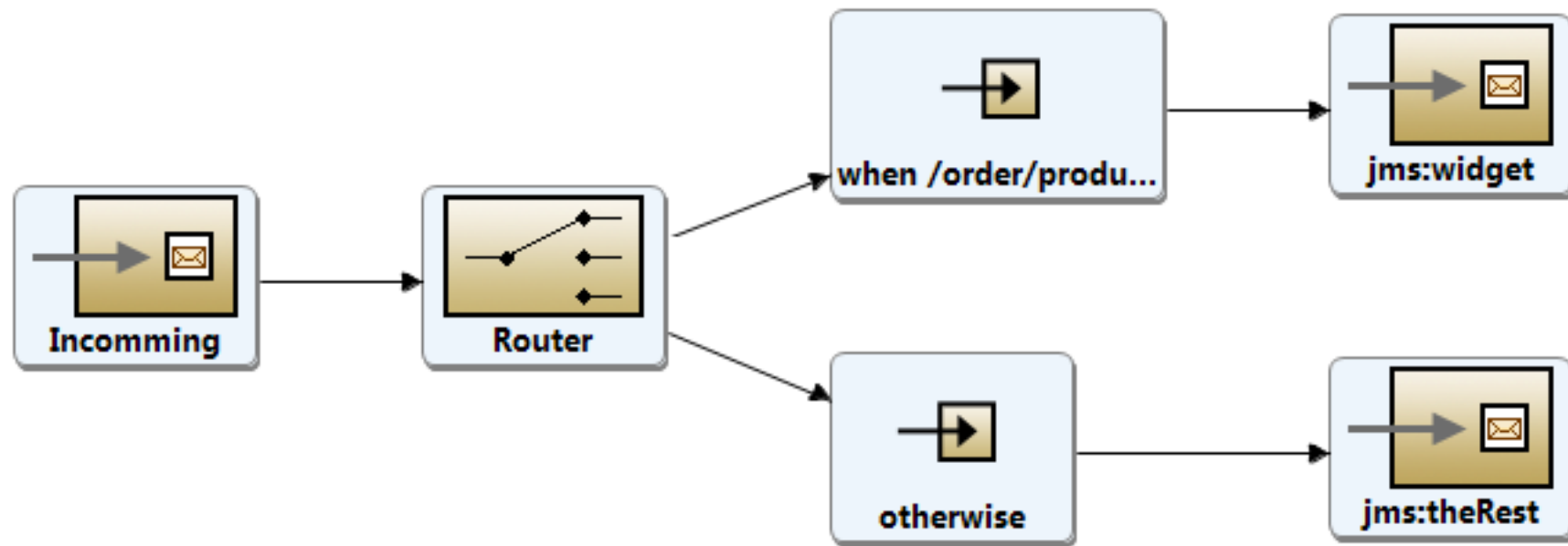
            // Create a MessageProducer from the Session to the Topic or Queue
            MessageProducer producer = session.createProducer(widget);
            producer.setDeliveryMode(DeliveryMode.NON_PERSISTENT);

            // Create a message
            TextMessage message = session.createTextMessage(text);

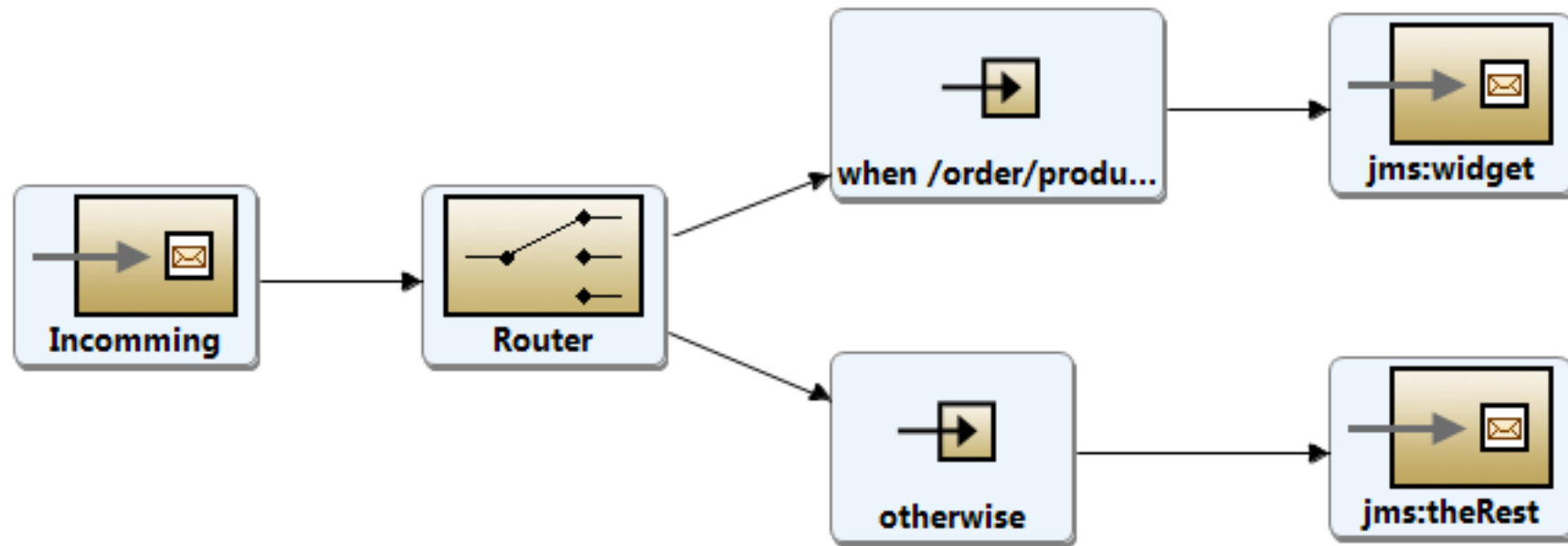
            // Tell the producer to send the message
            producer.send(message);
        } finally {
            // Clean up
            if (session != null) session.close();
            if (connection != null) connection.close();
        }
    }
}

```

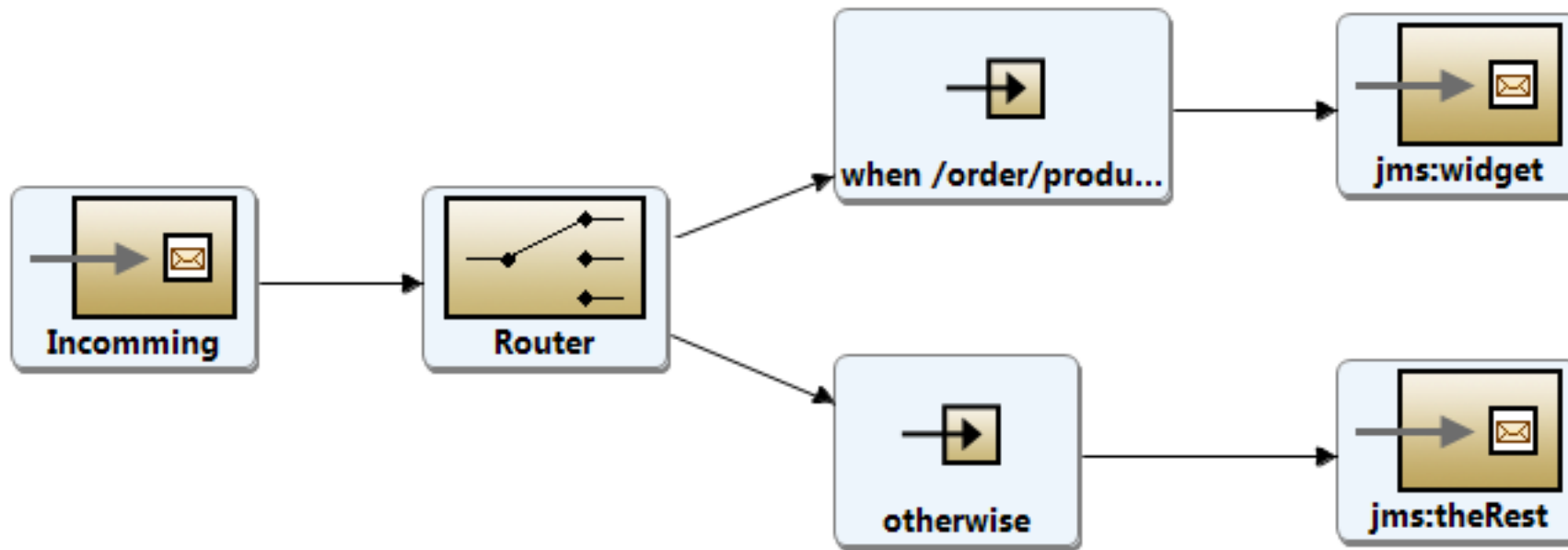
- **It's Java EE 6 (7 is simpler)**
- **Where's my routing logic?**
- **Introspecting messages**
- .
- .
- .
- **Sigh.**



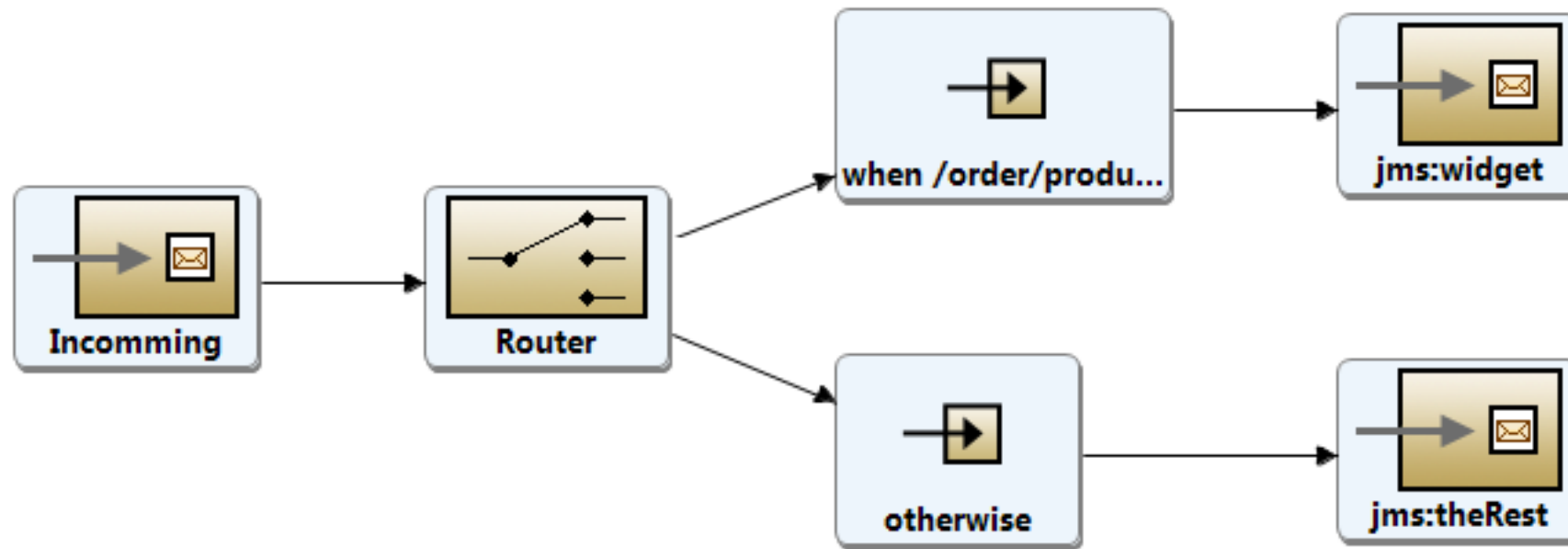
from `newOrder`
choice



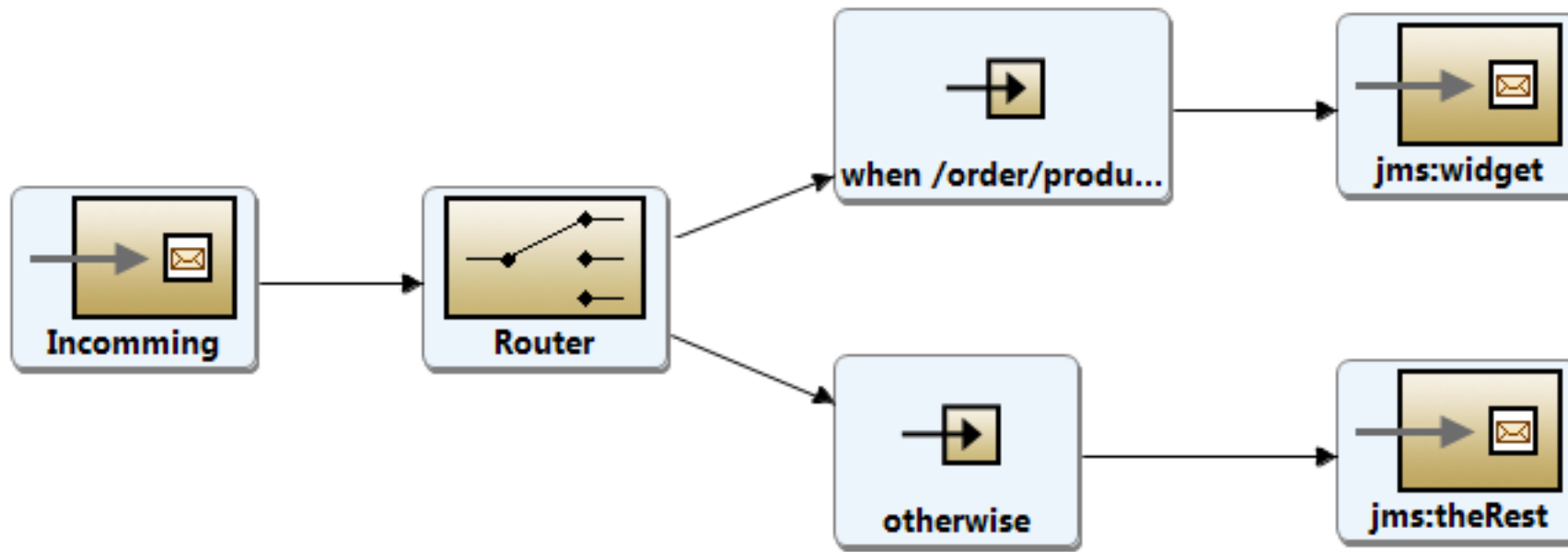
from `newOrder`
choice
when `isWidget` to `widget`



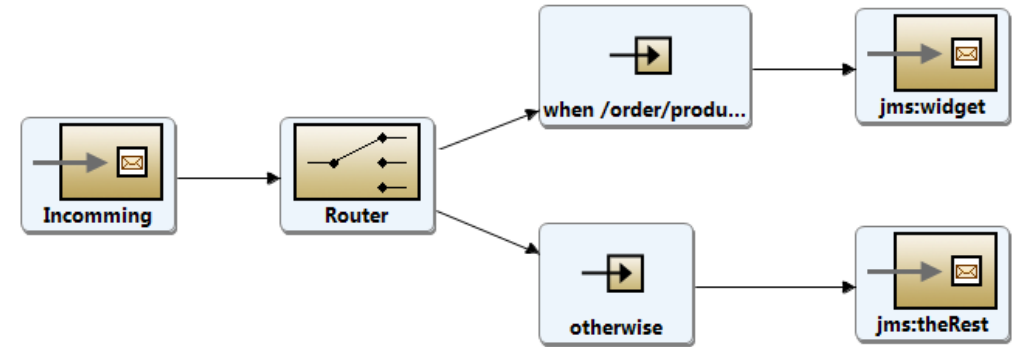
from `newOrder`
choice
 when `isWidget` to `widget`
 otherwise to `theRest`



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



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



```
Endpoint newOrder = endpoint("jms:incomming");  
Predicate isWidget = xpath("/order/product/type = 'widget'");  
Endpoint widget = endpoint("jms:widget");  
Endpoint gadget = endpoint("jms:theRest");
```

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

```
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("jms:incomming");
        Predicate isWidget = xpath("/order/product/type = 'widget'");
        Endpoint widget = endpoint("jms:widget");
        Endpoint gadget = endpoint("jms:theRest");

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

Cool!

Java DSL

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

public class MyRoute extends RouteBuilder {

    public void configure() throws Exception {

        from("jms:incomming")
            .choice()
                .when(xpath("/order/product/type = 'widget'"))
                    .to("jms:widget")
                .otherwise().to("jms:theRest")
            .end();
    }
}
```

XML DSL

```
<route>
  <from uri="jms:newOrder" />
  <choice>
    <when>
      <xpath>/order/product/type='widget'</xpath>
      <to uri="jms:widget"/>
    </when>
    <otherwise>
      <to uri="jms:theRest"/>
    </otherwise>
  </choice>
</route>
```

Groovy DSL

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

public class MyRoute extends RouteBuilder {

    public void configure() throws Exception {

        from("jms:incomming")
            .choice()
                .when({ it.in.header('zipcode') ==~ /\d+/ })
                    .to("jms:widget")
                .otherwise().to("jms:theRest")
            }
    }
}
```

Scala DSL

```
class FilterRoute {  
  def createMyFilterRoute = new RouteBuilder {  
    from("direct:start")  
      .filter(_.in("gold") == "true")  
      .to("mock:gold")  
  }  
}
```

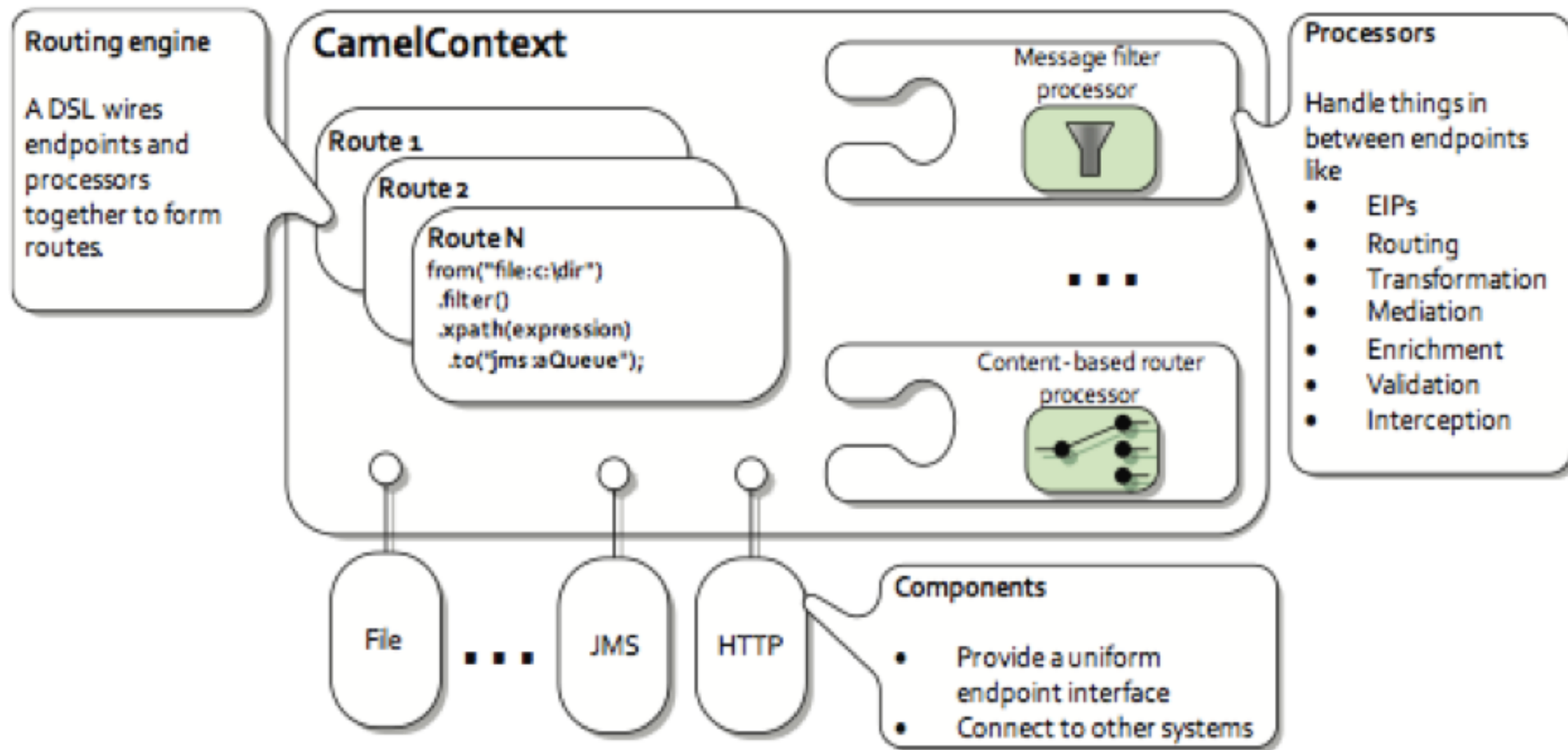
Configure Endpoints as URIs

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

public class MyRoute extends RouteBuilder {

    public void configure() throws Exception {

        from("file:inbox/orders?delete=true")
            .choice()
                .when(xpath("/order/product/type = 'widget'"))
                    .to("jms:widget")
                .otherwise().to("jms:theRest")
            .end();
    }
}
```



150+ Components

<http://camel.apache.org/component.html>

<http://fineartamerica.com/featured/apache-camel-components-poster-gliesian-llc.html>

```
rest("/user").description("User rest service")
    .consumes("application/json").produces("application/json")

    .get("/{id}").description("Find user by id").outType(User.class)
        .to("bean:userService?method=getUser(${header.id})")
    .put().description("Updates or create a user").type(User.class)
        .to("bean:userService?method=updateUser")
    .get("/findAll").description("Find all users").outTypeList(User.class)
        .to("bean:userService?method=listUsers");
```

User Services

Camel Rest Example with Swagger that provides an User REST service

user : User rest service

[Show/Hide](#) | [List Operations](#) | [Expand Operations](#) | [Raw](#)

PUT /user Updates or create a user

GET /user/{id} Find user by id

Response Class

[Model](#) | [Model Schema](#)

```
{
  "id": 0,
  "name": ""
}
```

Response Content Type

Parameters

Parameter	Value	Description	Parameter Type	Data Type
id	(required)		path	string

[Try it out!](#)

GET /user/findAll Find all users

[BASE URL: http://localhost:8080/camel/api-docs , API VERSION: 1.2.3]

19 Data Formats

<http://camel.apache.org/data-format.html>

```
// JSON Data Format
// lets turn Object messages into json (Xstream) then send to MQSeries
from("activemq:My.Queue")
    .marshal().json()
    .to("mqseries:Another.Queue");
```

```
/** Alternative Libraries
    marshal().json(JsonLibrary.Jackson)
    marshal().json(JsonLibrary.Gson).
**/
```

```
// Camel RSS Data Format
// only entries with Camel in the title will get through the filter
from("rss:file:src/test/data/rss20.xml?splitEntries=true&consumer.delay=100")
    .marshal()
    .rss()
    .filter().xpath("//item/title[contains(., 'Camel')]").to("mock:result");
```

15 Expression Languages

<http://camel.apache.org/languages.html>

```
// lets route if a line item is over $100
from("queue:foo")
    .filter(groovy("request.lineItems.any { i -> i.value > 100 }"))
    .to("queue:bar")
```

```
// route exchanges from admin users to a special queue.
from("direct:start")
    .choice()
        .when()
            .javascript("request.headers.get('user') == 'admin'")
            .to("seda:adminQueue")
        .otherwise()
            .to("seda:regularQueue");
```

Wow!

- Integration Framework
- Enterprise Integration Patterns (EIP)
- Routing DSLs
- Endpoint as URIs
- Just Java or XML code (or even more)
- No Container Dependency
- A hell lot of ready made components
- Expressions and Date-Formats

FTW!

Deploying Camel

camel-core.jar = 2.8 MB!

- **Deployment Options**

- Standalone
- Spring
- Java EE
- OSGi

- **Containers**
 - Apache ActiveMQ
 - Apache Karaf
 - Apache ServiceMix
 - Apache Tomcat
 - Fabric8
 - JBoss AS / WildFly
 - JBoss Fuse
 - JBoss Fuse Service Works
 - Jetty
 - WebLogic
 - Glassfish
 - WebSphere

Camel & Java EE

#1 Camel CDI

```
@Singleton
@Startup
public class Bootstrap {
    @Inject
    CdiCamelContext context;

    @PostConstruct
    public void init() {

        // create routes

        // Start Camel Context
        context.start();
    }

    @PreDestroy
    public void shutdown() {
        // Graceful Shutdown Camel Context
        context.stop();
    }
}
```

Keep in Mind!

- Not all components are working as expected
- Deployment gets really large
- Easy to run into classloading issues

#2 Custom Module

- Create a Spring Module
- Create a Camel Module

Keep in Mind!

- The module dependencies have to be correct.
- Need to reference modules from `jboss-deployment-structure.xml`
- No pre-build list / no support

#3 Camel Subsystem

- Simply apply the wildfly-camel-patch to a compatible wildfly version
- wildflyext/wildfly-camel docker distribution

```
@Startup
@ApplicationScoped
@ContextName("jms-camel-context")
public class JmsRouteBuilder extends RouteBuilder {

    @Resource(mappedName = "java:jboss/DefaultJMSConnectionFactory")
    private ConnectionFactory connectionFactory;

    @Override
    public void configure() throws Exception {
        JmsComponent component = new JmsComponent();
        component.setConnectionFactory(connectionFactory);

        getContext().addComponent("jms", component);

        from("timer://sendJMSMessage?fixedRate=true&period=10000")
            .transform(constant("<?xml version='1.0><message><greeting>hello
world</greeting></message>"))
            .to("jms:queue:WildFlyCamelQueue")
            .log("JMS Message sent");
    }
}
```

Keep in Mind!

- Not all components contained so far
- Still under development
- Growing component base

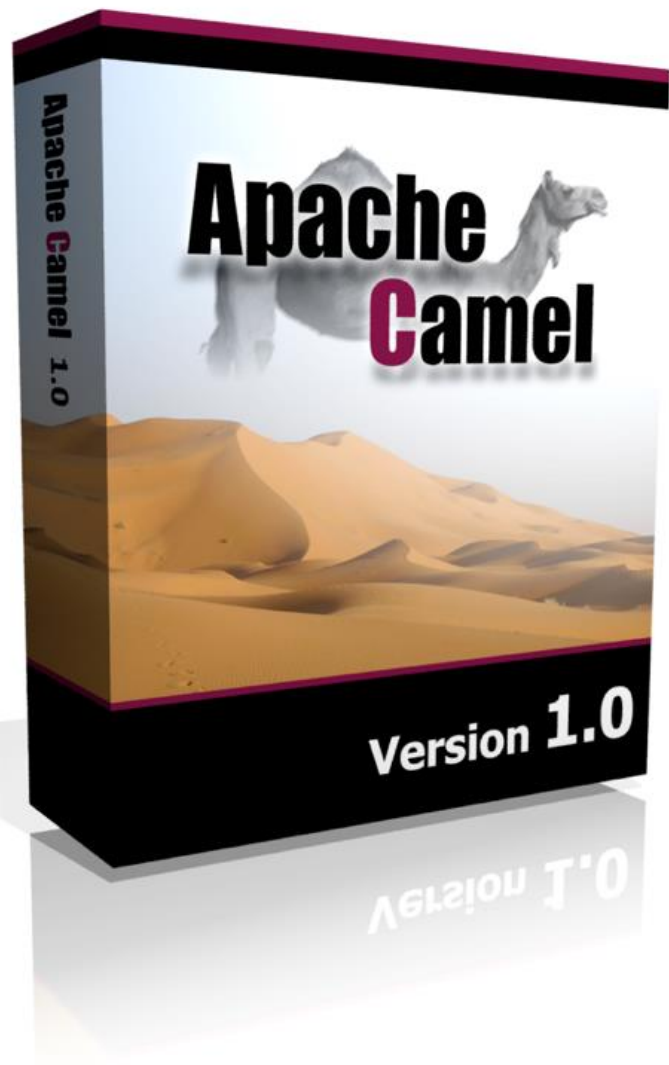
Camel and
JavaEE
are awesome!



What users are telling us

- **Camel rocks in development**
- **very easy to get started and create small projects**
- **can I have more tooling?**
- **need help scaling to 1000s of services and**
- **need help moving integration solutions through
Deployment pipeline**

Something is missing



<http://fabric8.io/>

- **an open source integration platform**
- **designed to make it easy to deploy, manage and scale integration solutions**
- **helps move integration solutions from dev -> test -> prod**

What does it do?

- **provision, configure and manage services**
- **perform incremental and rolling upgrades of changes**
- **discovery and load balancing of services**
- **elastic scaling of services**

How does that look like?

Fabric8 V1.x

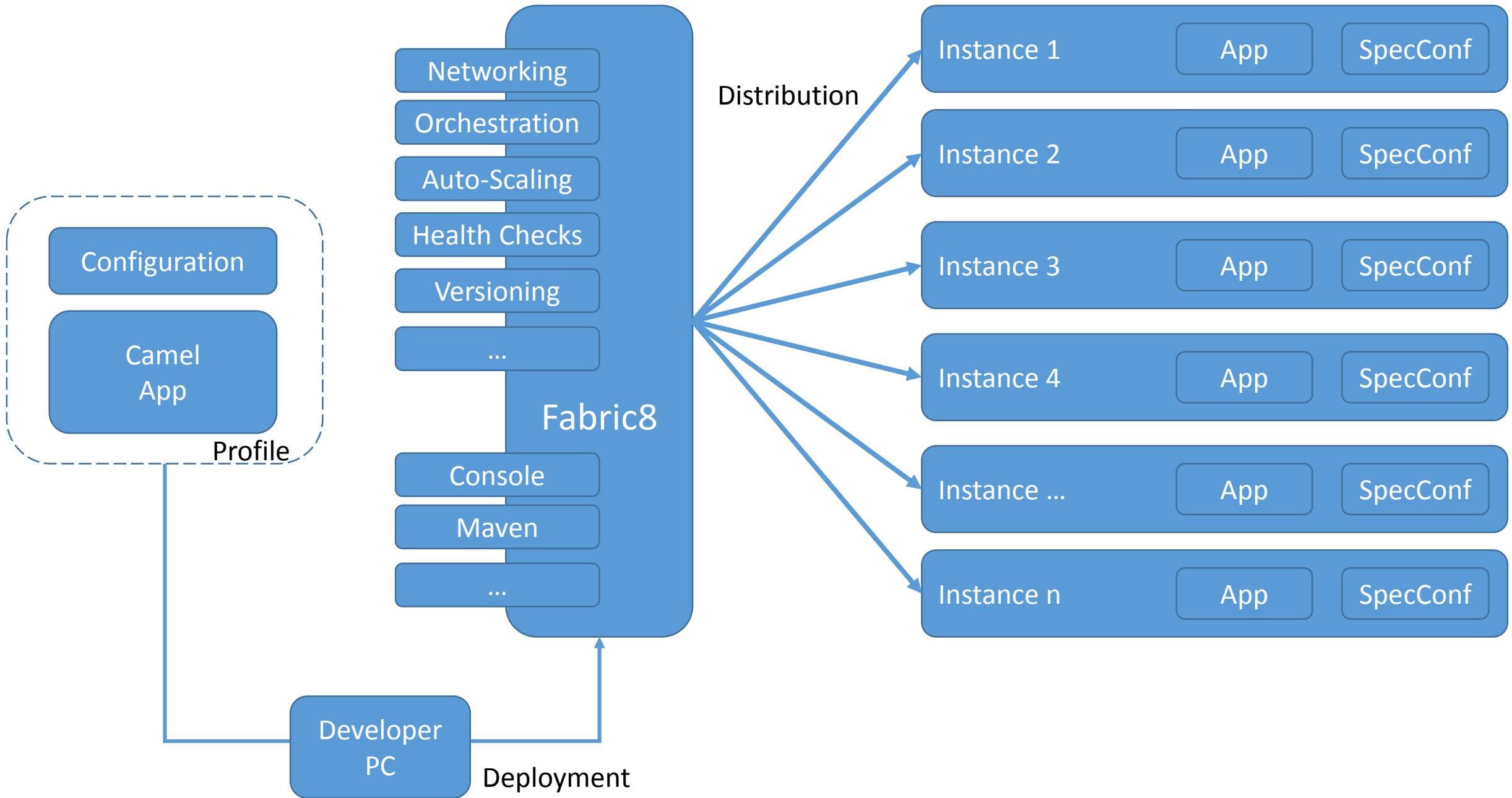


- **Implemented with Zookeeper and git**
- **Intended to be used with a dynamic JVM/Apache Karaf**
- **Profiles store configuration, metadata, end-state deployments**
- **Networking, JVM isolation, orchestration, auto-scaling, health checks, cloud deployments: all up to you**

Fabric8 V1.x



- **Registry**
 - **Where configs are stored, everything centrally managed**
- **Profiles**
 - **Description of end-state deployment of a server**
- **Agent**
 - **Listens on the server that can respond to registry changes/profile updates**



Fabric8 V2



- **Implemented with Docker and Kubernetes**
- **Use any JVM (or any technology)**
- **Docker images, encourage static, well-defined, well-tested deployments**
- **Provides networking, JVM isolation, orchestration, auto-scaling, health checks, cloud deployments**
- **Still in community!**
- **Will support OpenShift v3**

- **Profiles**
- **Runtime Registry**
- **Configuration Repository**
- **Tooling**
- **Profiles**

V1 vs. V2

- **Docker**
- **Kubernetes**
 - **PODs**
 - **Labels**
 - **Services**
 - **Apps**
- **App Zips**
 - **Builds**

Download and install

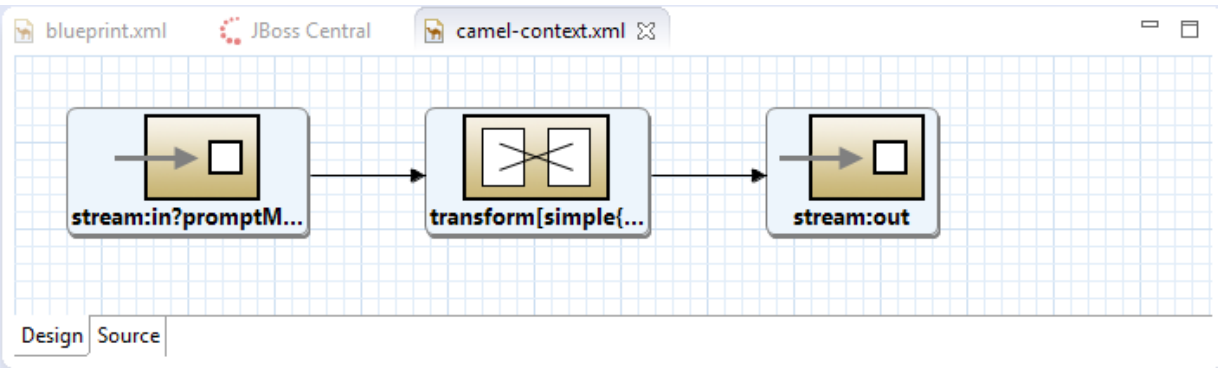
<http://camel.apache.org/download.html>

<http://fabric8.io/v2/>

<http://www.jboss.org/products/devstudio/overview/>

Projec... Packa...

- camel-example-console
 - src/main/java
 - src/main/resources
 - META-INF
 - spring
 - camel-context.xml
 - LICENSE.txt
 - NOTICE.txt
 - log4j.properties
 - JRE System Library [JavaSE-1.6]
 - Maven Dependencies
 - src
 - target
 - pom.xml
 - README.txt



Outline Palette

Select

- Endpoints
 - Endpoint
- Routing
 - Aggregate
 - Choice
- Control Flow

Problems Properties Servers Console OpenShift Explorer

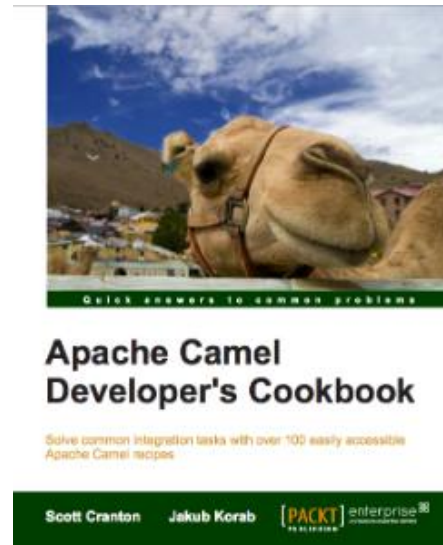
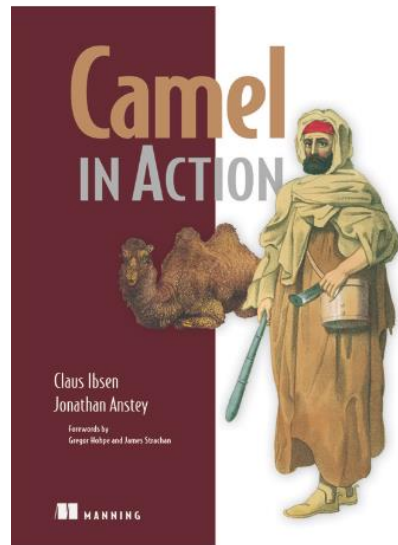
camel-example-console (1) [Maven Build] D:\jdk1.7.0_65\bin\javaw.exe (Jan 20, 2015, 9:53:10 AM)

```

Downloaded: http://repo.maven.apache.org/maven2/org/springframework/spring-test/3.2.11.RELEASE/spring-test-3.2.11.RELEASE.jar (448 KB at 895.1 KB/sec)
Downloaded: http://repo.maven.apache.org/maven2/org/apache/aries/blueprint/org.apache.aries.blueprint/1.1.0/org.apache.aries.blueprint-1.1.0.jar (442 KB)
[INFO] Using org.apache.camel.spring.Main to initiate a CamelContext
[INFO] Starting Camel ...
[pache.camel.spring.Main.main()] MainSupport INFO Apache Camel 2.14.1 starting
[pache.camel.spring.Main.main()] ClassPathXmlApplicationContext INFO Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@704
[pache.camel.spring.Main.main()] XmlBeanDefinitionReader INFO Loading XML bean definitions from file [D:\temp\apache-camel-2.14.1\apache-camel-
[pache.camel.spring.Main.main()] DefaultListableBeanFactory INFO Pre-instantiating singletons in org.springframework.beans.factory.support.Default
[pache.camel.spring.Main.main()] SpringCamelContext INFO Apache Camel 2.14.1 (CamelContext: camel-1) is starting
[pache.camel.spring.Main.main()] ManagedManagementStrategy INFO JMX is enabled
[pache.camel.spring.Main.main()] DefaultTypeConverter INFO Loaded 179 type converters
[pache.camel.spring.Main.main()] SpringCamelContext INFO AllowUseOriginalMessage is enabled. If access to the original message is not need
[pache.camel.spring.Main.main()] SpringCamelContext INFO StreamCaching is not in use. If using streams then its recommended to enable stre
[pache.camel.spring.Main.main()] SpringCamelContext INFO Route: route1 started and consuming from: Endpoint[stream://in?promptMessage=Ente
[pache.camel.spring.Main.main()] SpringCamelContext INFO Total 1 routes, of which 1 is started.
[pache.camel.spring.Main.main()] SpringCamelContext INFO Apache Camel 2.14.1 (CamelContext: camel-1) started in 0.640 seconds
Enter something: asdasd
ksnASn
  
```

Wanna know more ?

<http://camel.apache.org/examples>



<http://camel.apache.org/mailling-lists.html>

<http://stackoverflow.com/questions/tagged/apache-camel>

<http://camel.apache.org/irc-room.html>

<http://fabric8.io/v2/>

What else ?



**RED HAT® JBOSS®
FUSE**

**RED HAT® JBOSS®
FUSE SERVICE WORKS**

<http://camel.apache.org/commercial-camel-offerings.html>





<http://bit.ly/virtualJBUG>
@vJBUG

O'REILLY®

Modern Java EE Design Patterns

Building Scalable Architecture for
Sustainable Enterprise Development



Markus Eisele



<http://developers.redhat.com/promotions/distributed-javaee-architecture>

