Integration As a Service with Apache Camel

Keith Babo

Rundown

- Integration
- SOA
- Patterns / Techniques
- Camel
- Technologies

Enterprise Integration

- Applications, data, processes, systems
- Mature and evolved
  - Products
  - Patterns
  - Projects
- Value prop is clear and self-evident

SOA Manifesto

Service orientation is a paradigm that frames what you do.
Service-oriented architecture (SOA) is a type of architecture
that results from applying service orientation.

We have been applying service orientation to help organizations
consistently deliver sustainable business value, with increased agility
and cost effectiveness, in line with changing business needs.

Through our work we have come to prioritize:

Business value over technical strategy
Strategic goals over project-specific benefits
Intrinsic interoperability over custom integration
Shared services over specific-purpose implementations
Flexibility over optimization
Evolutionary refinement over pursuit of initial perfection

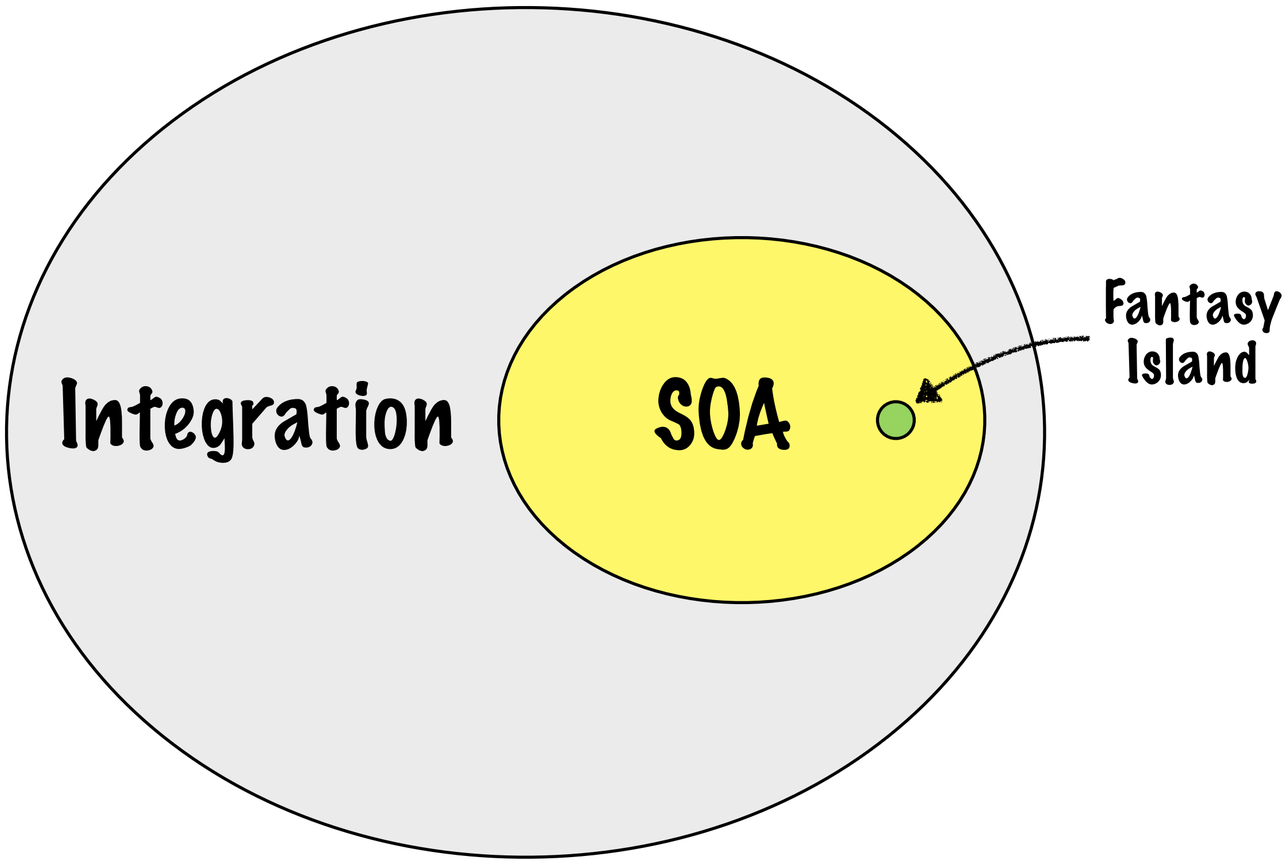
That is, while we value the items on the right, we value the items on the
left more.

http://www.soa-manifesto.org/

SOA Principles

- Service Contract
- Abstraction
- Reusability
- Discoverability
- Loose Coupling
- Autonomy
- Composability

Integration & SOA



Integration Challenge

SOA Challenge

- ESB project in JBoss Community
- Remembering the ‘SO’ in SOA
- Camel is an important ingredient

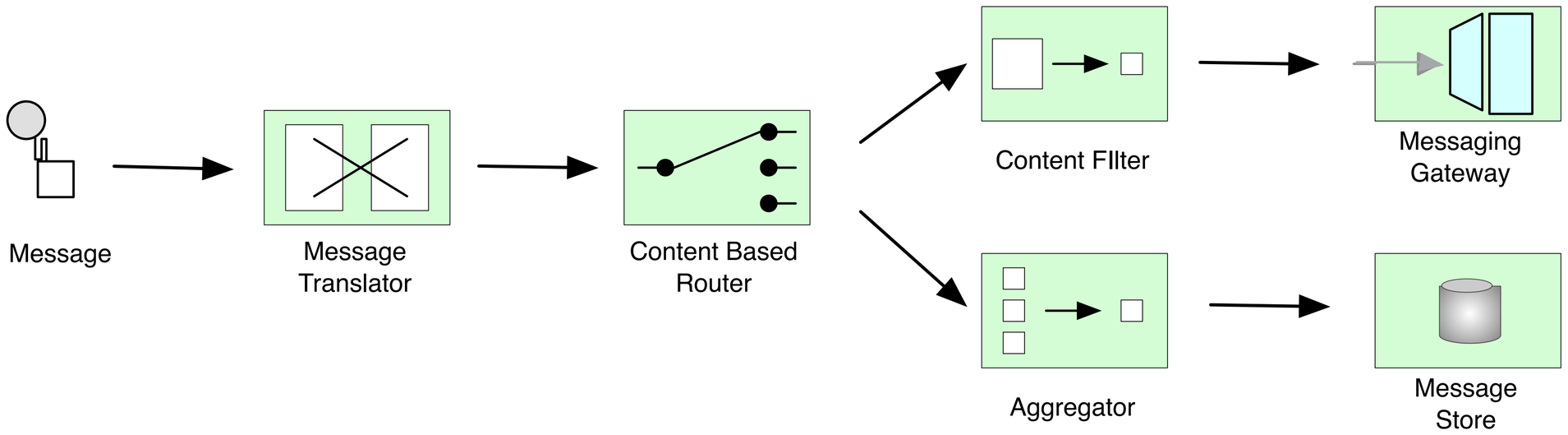
Focal Points

- Application Model
- Service Contracts
- Dependencies
- Binding Abstraction
- Cross Cutting Concerns
- Governance

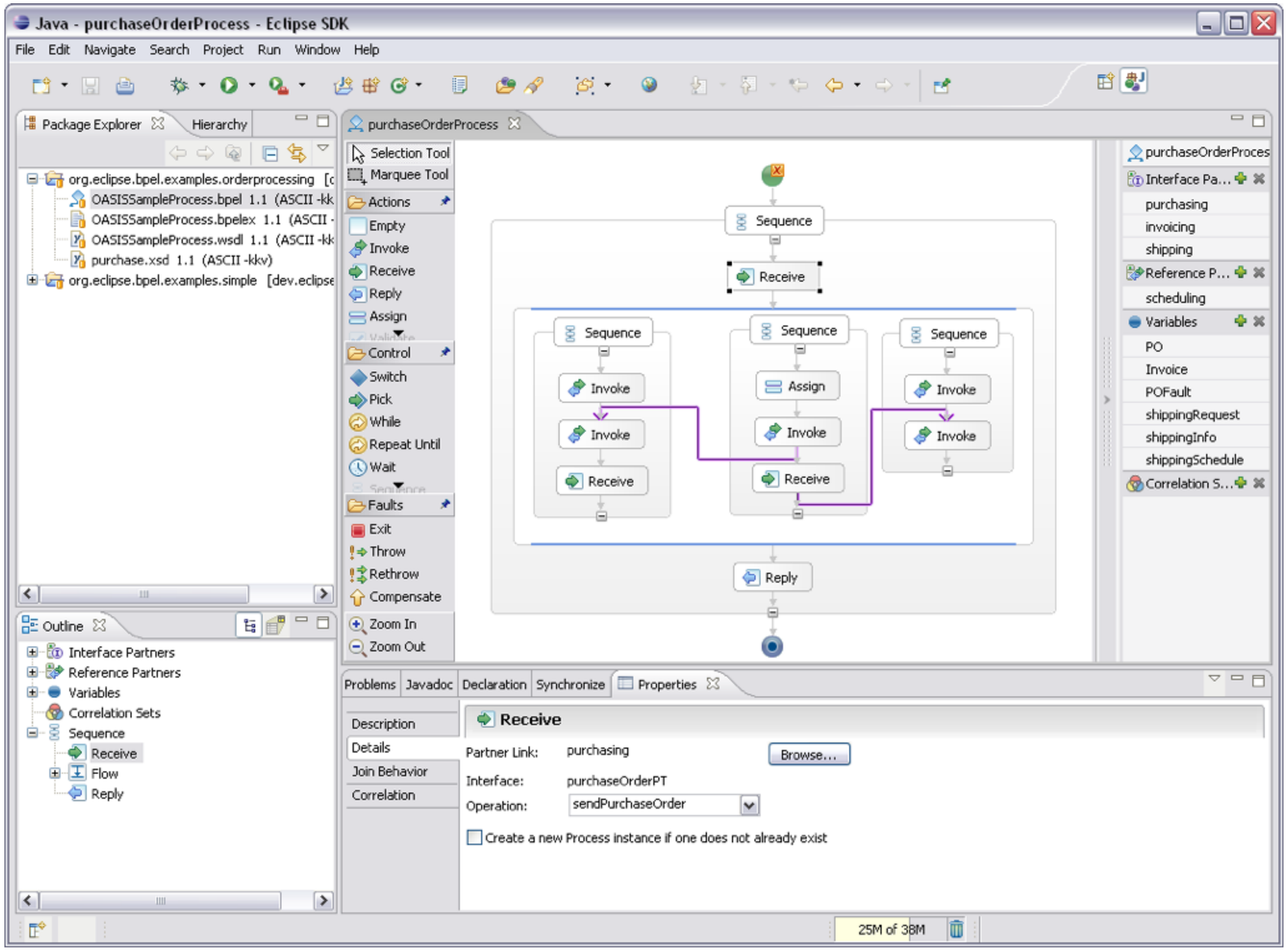
Application Model

- What's in your application?
  - Services provided
  - Services consumed
  - Service contracts
  - Policies

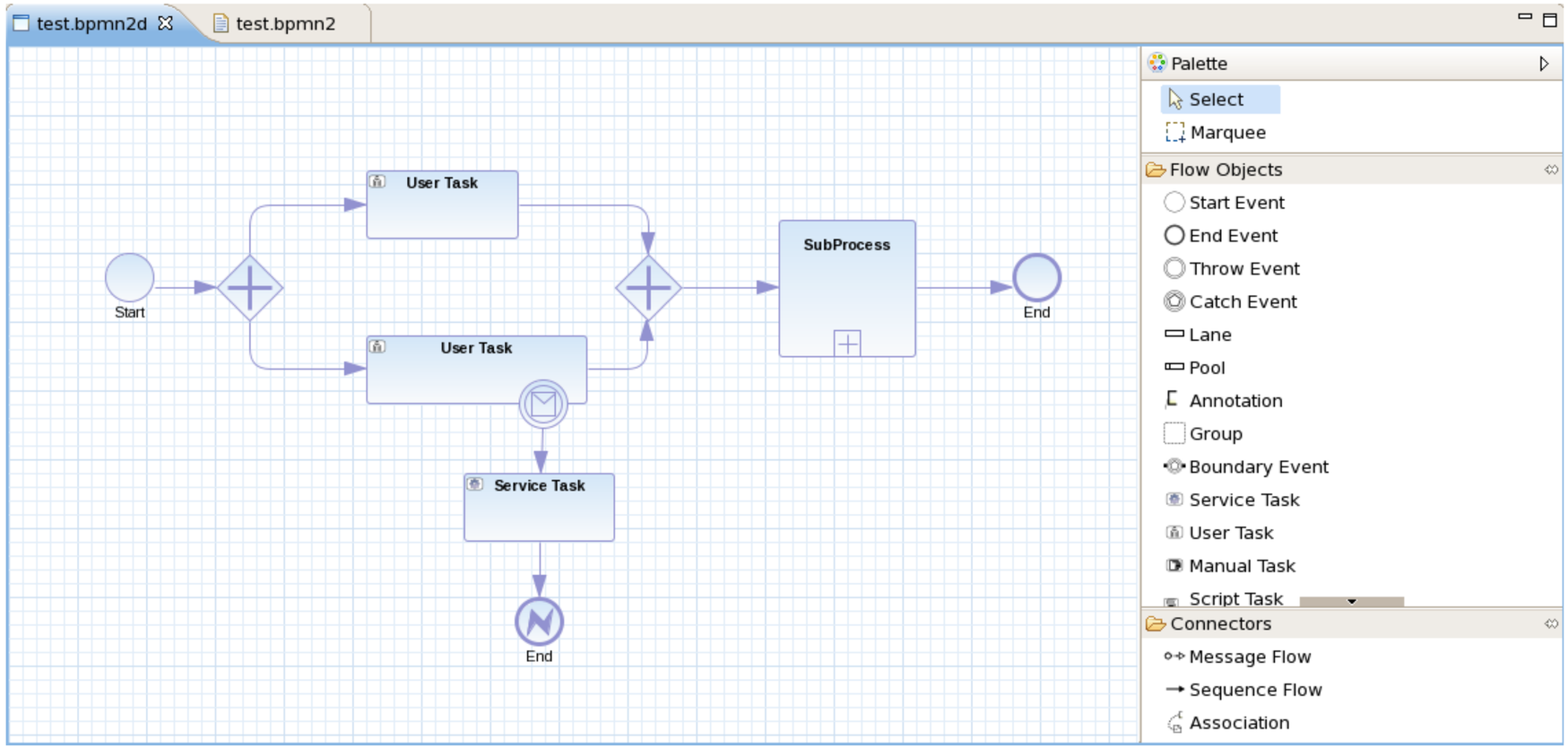
Integration Flow



Orchestration



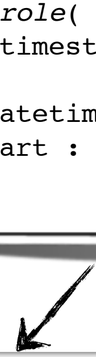
Workflow


test.bpmn2d
test.bpmn2
User Task
SubProcess
Start
End
User Task
Service Task
End
Palette
Select
Marquee
Flow Objects
Start Event
End Event
Throw Event
Catch Event
Lane
Pool
Annotation
Group
Boundary Event
Service Task
User Task
Manual Task
Script Task
Connectors
Message Flow
Sequence Flow
Association

Event Processing

```
declare StockTick
    @role( event )
    @timestamp (datetime)

    datetime : java.util.Date
    symbol   : String
    price    : double
end
```

```
declare MonkeyDart
    @role( event )
    @timestamp (datetime)

    datetime : java.util.Date
    dart : org.dartboard.Position
end
```

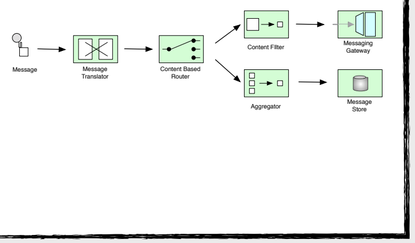


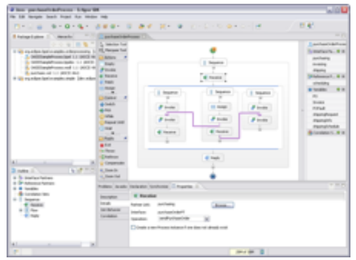
```
rule "Buy Low Sell High"
when
    YearLow( $low : low )
    $st : StockTick( price <= $low )
    $md : MonkeyDart( this after[0,5m] $st )
then
    // buy all the stock you can get!
    OrderService.buy(st.getTicker(), Integer.MAX_VALUE)
end
```

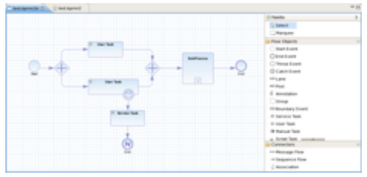
Java

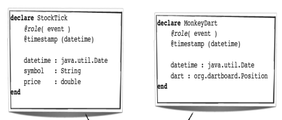
```
public class SendFileRecordsToQueueBean {
    @Produce(uri = "activemq:personnel.records")
    ProducerTemplate producer;

    @Consume(uri = "file:src/data?noop=true")
    public void onFileSendToQueue(String body) {
        producer.sendBody(body);
    }
}
```

Big Picture









```
declare StockTick
    @role( event )
    @timestamp (datetime)

    datetime : java.util.Date
    symbol   : String
    price    : double
end
```

```
declare MonkeyDart
    @role( event )
    @timestamp (datetime)

    datetime : java.util.Date
    dart : org.dartboard.Position
end
```

```
rule "Buy Low Sell High"
when
    YearLow( $low : low )
    $st : StockTick( price <= $low )
    $md : MonkeyDart( this after[0,5m] $st )
then
    // buy all the stock you can get!
    OrdersService.buy($st.getTicker(), Integer.MAX_VALUE)
end
```

```
public class SendFileRecordsToQueueBean {
    @Produce(uri = "activemq:personnel.records")
    ProducerTemplate producer;

    @Consume(uri = "file:src/data?noop=true")
    public void onFileSendToQueue(String body) {
        producer.sendBody(body);
    }
}
```

...

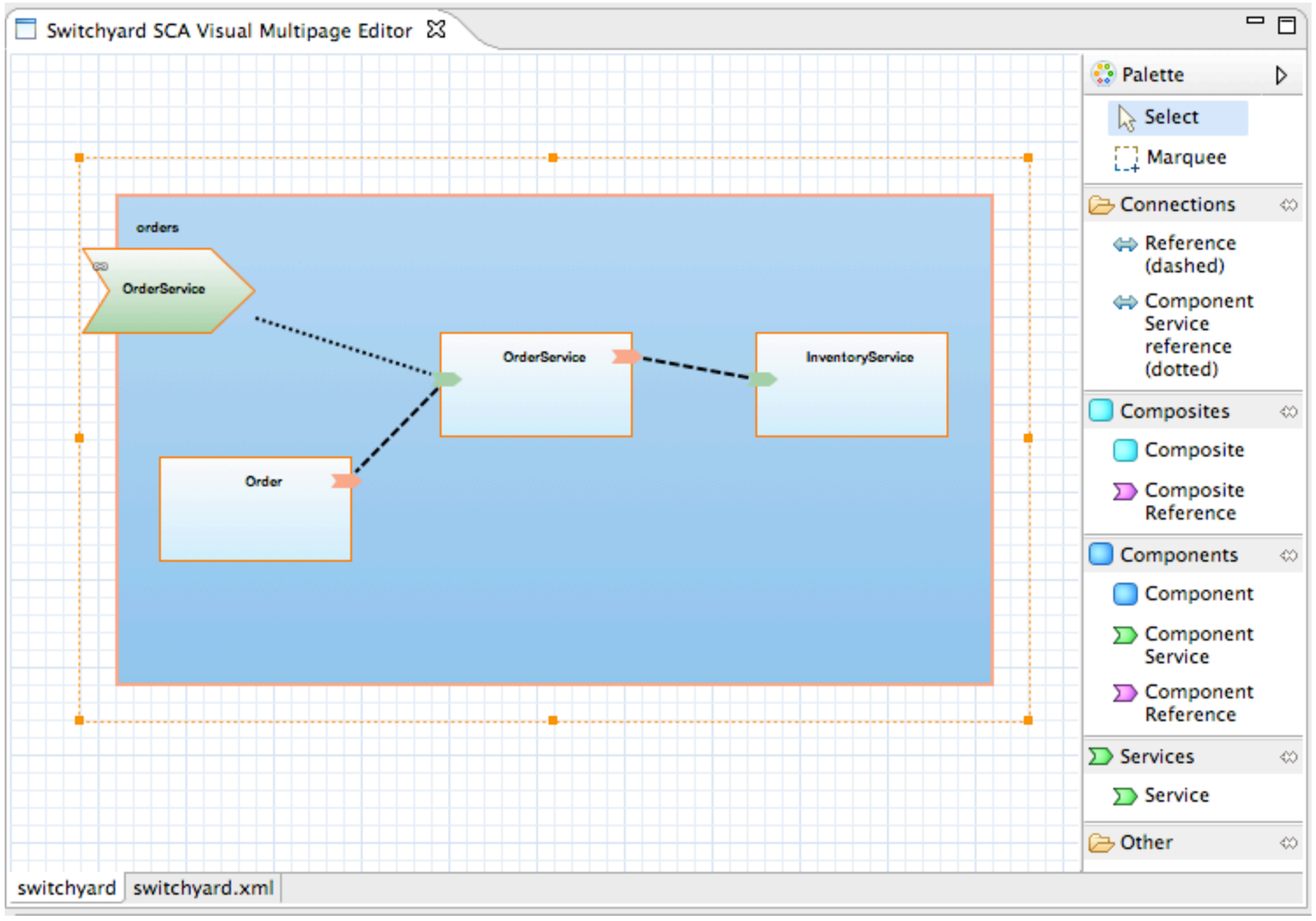
SCA

- Service Component Architecture
- OASIS standard for creating service-oriented applications
  - Describe
  - Assemble
  - Build

Application Descriptor

```
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           xmlns:bean="urn:switchyard-component-bean:config:1.0"
           xmlns:camel="urn:switchyard-component-camel:config:1.0">
           name="orders" targetNamespace="urn:example:orders">
    <service name="OrderService" promote="OrderService">
        <interface.wsdl interface="wsdl/OrderService.wsdl#wsdl.porttype(OrderService)"/>
        <binding.soap xmlns="urn:switchyard-component-soap:config:1.0">
            <wsdl>wsdl/OrderService.wsdl</wsdl>
            <socketAddr>:18001</socketAddr>
            <contextPath>demo-orders</contextPath>
        </binding.soap>
    </service>
    <component name="InventoryService">
        <bean:implementation.bean class="org.example.orders.InventoryServiceBean"/>
        <service name="InventoryService">
            <interface.java interface="org.example.orders.InventoryService"/>
        </service>
    </component>
    <component name="Order">
        <bean:implementation.bean class="org.example.orders.Order"/>
        <reference name="OrderService">
            <interface.java interface="org.example.orders.OrderService"/>
        </reference>
    </component>
    <component name="OrderService">
        <camel:implementation.camel>
           <camel:java class="org.example.orders.OrdersRoute"/>
        </camel:implementation.camel>
        <service name="OrderService">
            <interface.java interface="org.example.orders.OrderService"/>
        </service>
        <reference name="InventoryService">
            <interface.java interface="org.example.orders.InventoryService"/>
        </reference>
    </component>
</composite>
```

Application Model

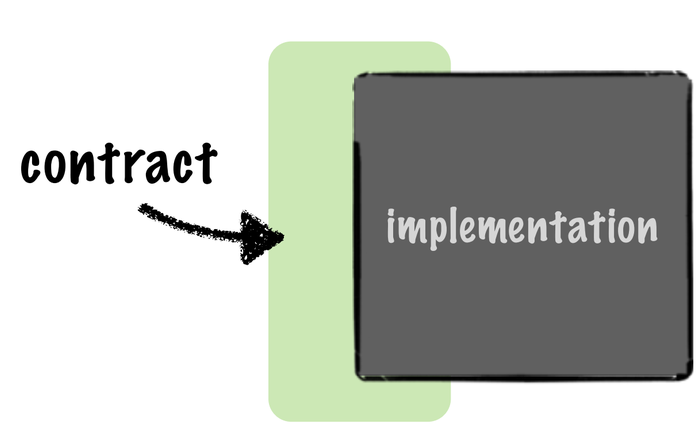


Takeaways

- Need for a uniform application model increases as number of implementation languages increases
- Easter eggs are bad news
- Choose a “standard”
- Visualization is extremely valuable

Service Contracts

- Agreement between consumer and provider
  - Messages
  - Policy
  - SLA
- Contract-based development
  - Service Interface
  - Implementation Hiding



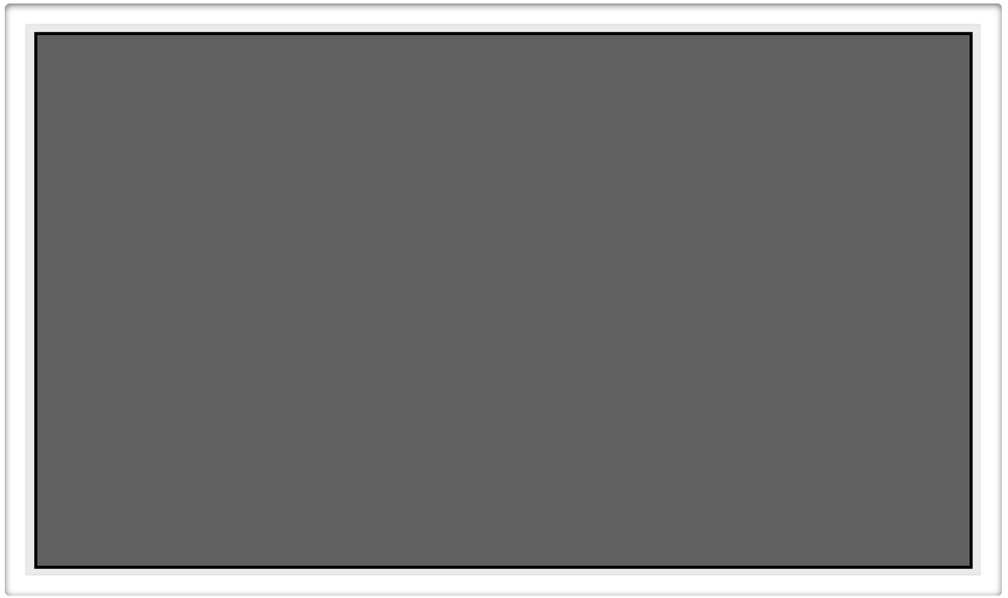
Service Contracts

- What's the contract for this service?

```
from("cxf:bean:OrderWebService")
  .split().tokenizeXML("order")
  .choice()
     .when(xpath("/order/region = 'NA'"))
       .to("direct:DirectSales")
     .when(xpath("/order/region = 'LAM'"))
       .to("bean:LAMOrderProcessor")
     .when(xpath("/order/region = 'EMEA'"))
       .to("vm:eu.orders.inbound")
     .when(xpath("/order/region = 'APAC'"))
       .to("jms:queue:ordersQueue");
```

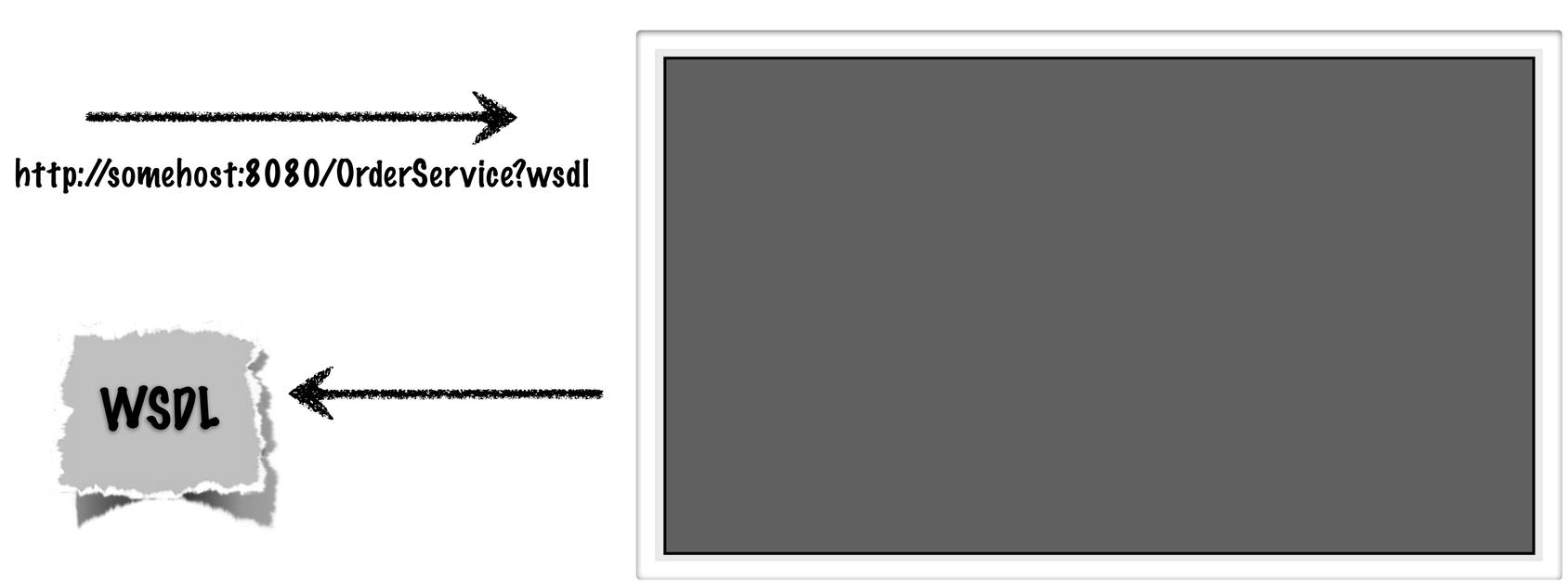
Service Contracts

- What's the contract for this service?



Service Contracts

- What's the contract for this service?



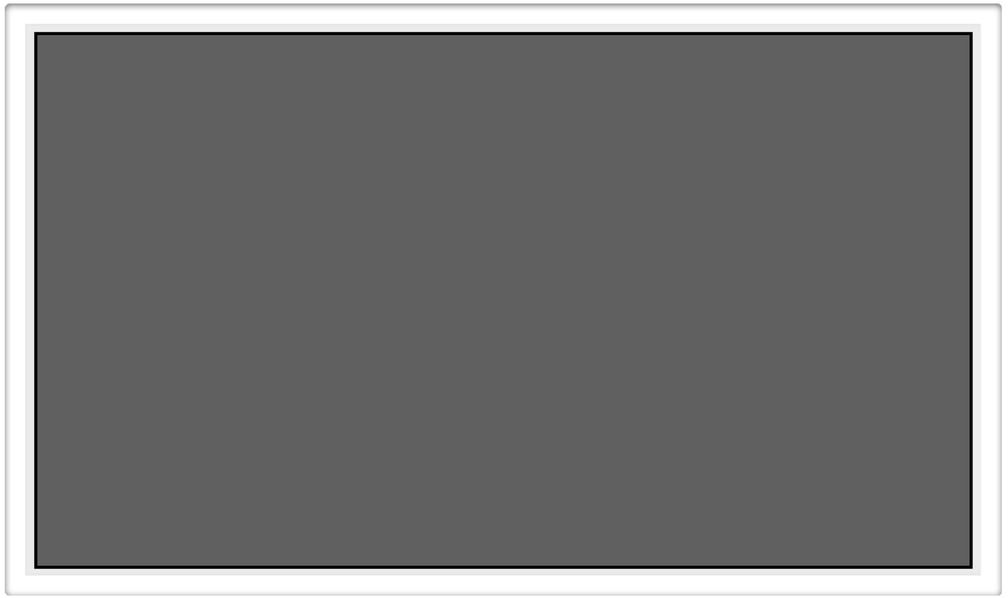
Service Contracts

- How about this one?

```
from("jms:OrderServiceQueue")
  .split().tokenizeXML("order")
  .choice()
     .when(xpath("/order/region = 'NA'"))
       .to("direct:DirectSales")
     .when(xpath("/order/region = 'LAM'"))
       .to("bean:LAMOrderProcessor")
     .when(xpath("/order/region = 'EMEA'"))
       .to("vm:eu.orders.inbound")
     .when(xpath("/order/region = 'APAC'"))
       .to("jms:queue:ordersQueue");
```

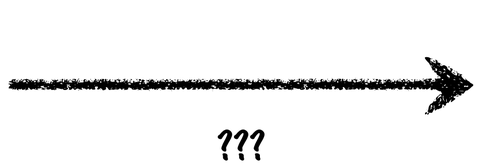
Service Contracts

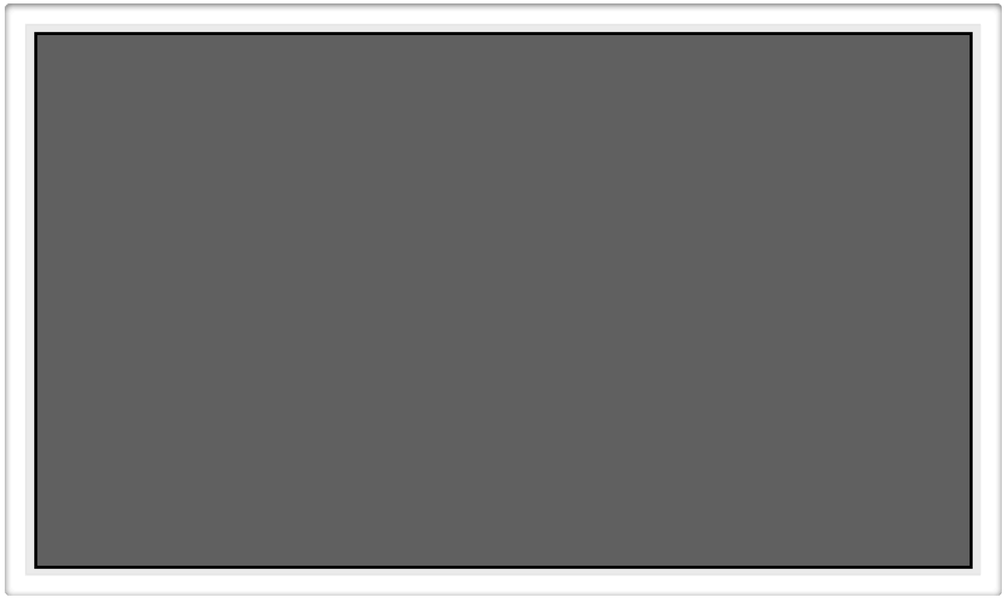
- How about this one?



Service Contracts

- How about this one?





Service Contracts

- Wrap the implementation as a service component

```
<sca:composite
   xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200912"
   xmlns:camel="urn:switchyard-component-camel:config:1.0"
   name="camel-example"
   targetNamespace="urn:examples:1.0">
   <sca:component name="ProxyService">
      <camel:implementation.camel>
         <camel:java class="org.example.orders.OrdersRoute"/>
      </camel:implementation.camel>
      <sca:service name="ProxyService">
         <sca:interface.wsdl
              interface="Inventory.wsdl#wsdl.porttype(InventoryService)"/>
         </sca:service>
    </sca:component>
</sca:composite>
```

Contracts with Camel

```
<xs:complexType name="routeDefinition">
  <xs:complexContent>
    <xs:extension base="tns:processorDefinition">
      <!-- snip -->
      <xs:anyAttribute namespace="##other" processContents="skip"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

```
<route id="inside" xmlns:ext="urn:camel:extensions”
   ext:contract="wsdl:MyService.wsdl(MyServicePortType)">
     <from uri="direct:MyService"/>
     <to uri="mock:somewhereElse"/>
</route>
```

Takeaways

- Contracts can be your change shield
- Keep it abstract
- Embrace impurity
- Make it discoverable
- Contract first vs. contract last

Dependencies

- Concerns outside the application boundary
- Integration app = dependencies
- Key Principles
  - Autonomy
  - Composability

Dependency Analysis is Easy

```
from("switchyard://OrderService")
  .split().tokenizeXML("order")
  .choice()
    .when(xpath("/order/region = 'NA'"))
      .to("direct:DirectSales")
    .when(xpath("/order/region = 'LAM'"))
      .to("bean:LAMOrderProcessor")
    .when(xpath("/order/region = 'EMEA'"))
      .to("vm:eu.orders.inbound")
    .when(xpath("/order/region = 'APAC'"))
      .to("jms:queue:ordersQueue");
```

Dependency Analysis is Easy


```
from("switchyard://OrderService")
  .split().tokenizeXML("order")
  .choice()
    .when(xpath("/order/region = 'NA'"))
     .to("direct:DirectSales")
    .when(xpath("/order/region = 'LAM'"))
     .to("bean:LAMOrderProcessor")
    .when(xpath("/order/region = 'EMEA'"))
      .to("vm:eu.orders.inbound")
    .when(xpath("/order/region = 'APAC'"))
      .to("jms:queue:ordersQueue");
```

Dependency Analysis is Easy

Internal

```
from("switchyard://OrderService")
  .split().tokenizeXML("order")
  .choice()
    .when(xpath("/order/region = 'NA'"))
     .to("direct:DirectSales")
    .when(xpath("/order/region = 'LAM'"))
     .to("bean:LAMOrderProcessor")
    .when(xpath("/order/region = 'EMEA'"))
      .to("vm:eu.orders.inbound")
    .when(xpath("/order/region = 'APAC'"))
      .to("jms:queue:ordersQueue");
```

External

Well, Sort Of ...

```
from("switchyard://OrderService")
  .split().tokenizeXML("order")
  .choice()
    .when(xpath("/order/region = 'NA'"))
      .to("direct:DirectSales")
    .when(xpath("/order/region = 'LAM'"))
      .to("bean:LAMOrderProcessor")
    .when(xpath("/order/region = 'EMEA'"))
      .to("vm:eu.orders.inbound")
    .when(xpath("/order/region = 'APAC'"))
      .to("jms:queue:ordersQueue");
```

Well, Sort Of ...

```
from("direct://DirectSales")
.to("cxf:bean:OrderWebService")
```

```
public class LAMOrderProcessor {
    @Produce(uri = "activemq:lam.orders")
    ProducerTemplate producer;

    public void send(String body) {
        producer.sendBody(body);
    }
}
```

```
from("switchyard://OrderService")
  .split().tokenizeXML("order")
  .choice()
    .when(xpath("/order/region = 'NA'"))
      .to("direct:DirectSales")
    .when(xpath("/order/region = 'LAM'"))
      .to("bean:LAMOrderProcessor")
    .when(xpath("/order/region = 'EMEA'"))
      .to("vm:eu.orders.inbound")
    .when(xpath("/order/region = 'APAC'"))
      .to("jms:queue:ordersQueue");
```

Well, Sort Of ...

```
from("direct://DirectSales")
.to("cxf:bean:OrderWebService")
```

```
public class LAMOrderProcessor {
    @Produce(uri = "activemq:lam.orders")
    ProducerTemplate producer;

    public void send(String body) {
        producer.sendBody(body);
    }
}
```

```
from("switchyard://OrderService")
  .split().tokenizeXML("order")
  .choice()
    .when(xpath("/order/region = 'NA'"))
      .to("direct:DirectSales")
    .when(xpath("/order/region = 'LAM'"))
      .to("bean:LAMOrderProcessor")
    .when(xpath("/order/region = 'EMEA'"))
      .to("vm:eu.orders.inbound")
    .when(xpath("/order/region = 'APAC'"))
      .to("jms:queue:ordersQueue");
```

External

Maybe Not

```
from("switchyard://OrderService")
  .split().tokenizeXML("order")
  .choice()
    .when(xpath("/order/region = 'NA'"))
      .to("direct:DirectSales")
    .when(xpath("/order/region = 'LAM'"))
      .to("bean:LAMOrderProcessor")
    .when(xpath("/order/region = 'EMEA'"))
      .to("vm:eu.orders.inbound")
    .when(xpath("/order/region = 'APAC'"))
      .to("jms:queue:ordersQueue");
```

Maybe Not

```
from("switchyard://OrderService")
  .split().tokenizeXML("order")
  .choice()
    .when(xpath("/order/region = 'NA'"))
      .to("direct:DirectSales")
    .when(xpath("/order/region = 'LAM'"))
      .to("bean:LAMOrderProcessor")
    .when(xpath("/order/region = 'EMEA'"))
      .to("vm:eu.orders.inbound")
    .when(xpath("/order/region = 'APAC'"))
      .to("jms:queue:ordersQueue");
```

```
from("jms://queue:ordersQueue")
.to("bean:APACOrderProcessor")
```

```
from("vm://eu.orders.inbound")
.to("direct:ThisIsRidiculous")
```

Maybe Not

Internal

```
from("switchyard://OrderService")
   .split().tokenizeXML("order")
   .choice()
      .when(xpath("/order/region = 'NA'"))
      .to("direct:DirectSales")
      .when(xpath("/order/region = 'LAM'"))
      .to("bean:LAMOrderProcessor")
      .when(xpath("/order/region = 'EMEA'"))
      .to("vm:eu.orders.inbound")
      .when(xpath("/order/region = 'APAC'"))
      .to("jms:queue:ordersQueue");
```

```
from("jms://queue:ordersQueue")
.to("bean:APACOrderProcessor")
```

```
from("vm://eu.orders.inbound")
.to("direct:ThisIsRidiculous")
```

Defining Dependencies

*abridged

```
<sca:composite name="example" targetNamespace="urn:com.example"
          xmlns:sca="http://docs.oasis-open.org/ns/opencsa/sca/200912"
          xmlns:bean="urn:switchyard-component-bean:config:1.0"
          xmlns:camel="urn:switchyard-component-camel:config:1.0">
   <sca:reference multiplicity="0..1" name="EMEAOrders" promote="OrderIntake/EMEAOrders">
      <!-- binding omitted -->
   </sca:reference>
   <sca:reference multiplicity="0..1" name="APACOrders" promote="OrderIntake/APACOrders"/>
      <!-- binding omitted -->
   </sca:reference>
   <sca:component name="OrderIntake">
       <camel:implementation.camel>
           <camel:java class="org.example.IntakeRoute"/>
       </camel:implementation.camel>
       <sca:service name="OrderService">
           <sca:interface.java interface="org.example.OrderService"/>
       </sca:service>
       <sca:reference name="NAOrders"/>
       <sca:reference name="EMEAOrders"/>
       <sca:reference name="LAMOrders"/>
       <sca:reference name="APACOrders"/>
   </sca:component>
   <sca:component name="DirectSales">
       <bean:implementation.bean class="org.example.OrderServiceTest"/>
       <sca:service name="NAOrders"/>
   </sca:component>
   <sca:component name="LAMOrderProcessor">
       <camel:implementation.camel>
           <camel:java class="MyRoute"/>
       </camel:implementation.camel>
       <sca:service name="LAMOrders"/>
   </sca:component>
</sca:composite>
```

Service Definition

```
<sca:component name="OrderIntake">
    <camel:implementation.camel>
        <camel:java class="org.example.IntakeRoute"/>
    </camel:implementation.camel>
    <sca:service name="OrderService">
        <sca:interface.java interface="org.example.OrderService"/>
    </sca:service>
    <sca:reference name="NAOrders"/>
    <sca:reference name="EMEAOrders"/>
    <sca:reference name="LAMOrders"/>
    <sca:reference name="APACOrders"/>
</sca:component>
```

Application-Local Services

```
<sca:component name="DirectSales">
    <bean:implementation.bean class="org.example.OrderServiceTest"/>
    <sca:service name="NAOrders"/>
</sca:component>
<sca:component name="LAMOrderProcessor">
    <camel:implementation.camel>
        <camel:java class="MyRoute"/>
    </camel:implementation.camel>
    <sca:service name="LAMOrders"/>
</sca:component>
```

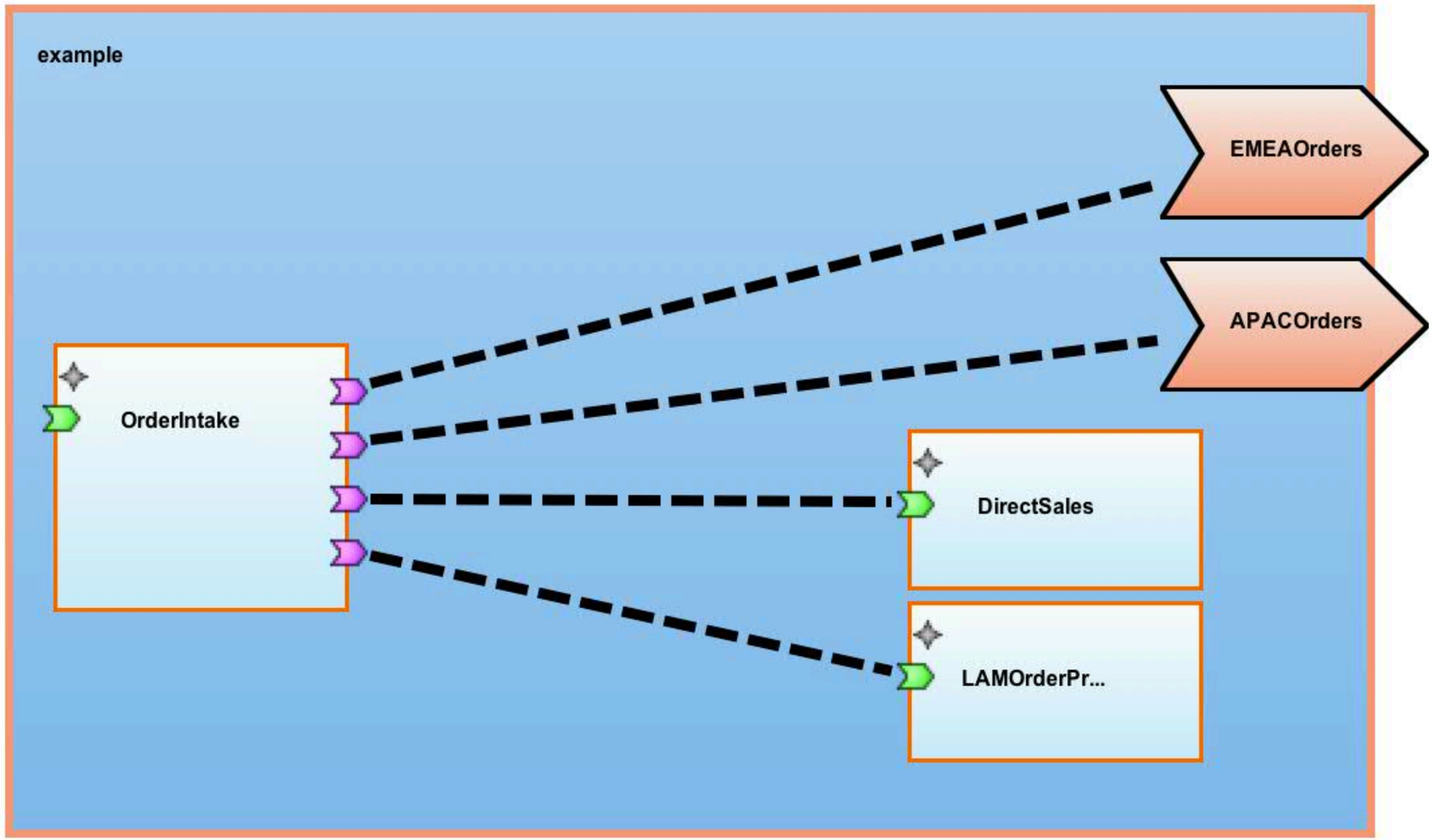
Remote Services

```
<sca:reference multiplicity="0..1"
    name="EMEAOrders" promote="OrderIntake/EMEAOrders">
   <!-- binding omitted -->
</sca:reference>
<sca:reference multiplicity="0..1"
    name="APACOrders" promote="OrderIntake/APACOrders"/>
   <!-- binding omitted -->
</sca:reference>
```

Enough XML

```
from("switchyard://OrderService")
  .split().tokenizeXML("order")
  .choice()
    .when(xpath("/order/region = 'NA'"))
      .to("switchyard:NAOrders")
    .when(xpath("/order/region = 'LAM'"))
      .to("switchyard:LAMOrders")
    .when(xpath("/order/region = 'EMEA'"))
      .to("switchyard:EMEAOrders")
    .when(xpath("/order/region = 'APAC'"))
      .to("switchyard:APACOrders");
```

Enough XML


example
EMEAOrders
APACOrders
OrderIntake
DirectSales
LAMOrderPr...

Takeaways

- Do not ignore this!
- Figure out a way to document dependencies
- Operations and support will thank you
  - Lifecycle management
  - Resource management
  - Change management

Binding Abstraction

- Separating abstract and concrete (again)
  - Service implementation
  - Service endpoint
- Flexibility
- Reuse

Necessary?

```
from("file:///opt/LSYS-HOME/inbox")
  .split().tokenizeXML("order")
  .choice()
    .when(xpath("/order/region = 'NA'"))
      .to("direct:DirectSales")
    .when(xpath("/order/region = 'LAM'"))
      .to("bean:LAMOrderProcessor")
    .when(xpath("/order/region = 'EMEA'"))
      .to("vm:eu.orders.inbound")
    .when(xpath("/order/region = 'APAC'"))
      .to("jms:queue:ordersQueue");
```

Better?

```
from("file:///opt/LSYS-HOME/inbox")
.to("direct:OrderIntake");
```

```
from("direct:OrderIntake")
   .split().tokenizeXML("order")
   .choice()
      .when(xpath("/order/region = 'NA'"))
        .to("direct:DirectSales")
      .when(xpath("/order/region = 'LAM'"))
        .to("bean:LAMOrderProcessor")
      .when(xpath("/order/region = 'EMEA'"))
        .to("vm:eu.orders.inbound")
      .when(xpath("/order/region = 'APAC'"))
        .to("direct:APACOrders");
```

```
from("direct:APACOrders")
.to("jms:queue:ordersQueue");
```

Abstraction in SCA

```
<sca:service name="OrderIntake" promote="OrderIntake">
   <camel:binding.camel configURI="file:///opt/LSYS-HOME/inbox">
</sca:service>
```

```
<sca:reference multiplicity="0..1"
    name="APACOrders" promote="OrderIntake/APACOrders"/>
   <camel:binding.camel
    configURI="jms://ordersQueue?connectionFactory=#ConnectionFactory"/>
</sca:reference>
```

Another Flavor

- Camel Properties component
- Separated from route
- Late binding
- Registry, file, Blueprint, ...

```
orders.endpoint=file://inbox
apac.endpoint=jms:queue:ordersQueue
```

```
from("{{orders.endpoint}}")
  .split().tokenizeXML("order")
  .choice()
    .when(xpath("/order/region = 'NA'"))
      .to("direct:DirectSales")
    .when(xpath("/order/region = 'LAM'"))
      .to("bean:LAMOrderProcessor")
    .when(xpath("/order/region = 'EMEA'"))
      .to("vm:eu.orders.inbound")
    .when(xpath("/order/region = 'APAC'"))
      .to("{{apac.endpoint}}");
```

Takeaways

- Bind endpoints as late as possible
- Consider “gateway” routes
- Gateway routes offer distinct lifecycle
- Offload binding-specific logic to dedicated routes

Cross Cutting Concerns

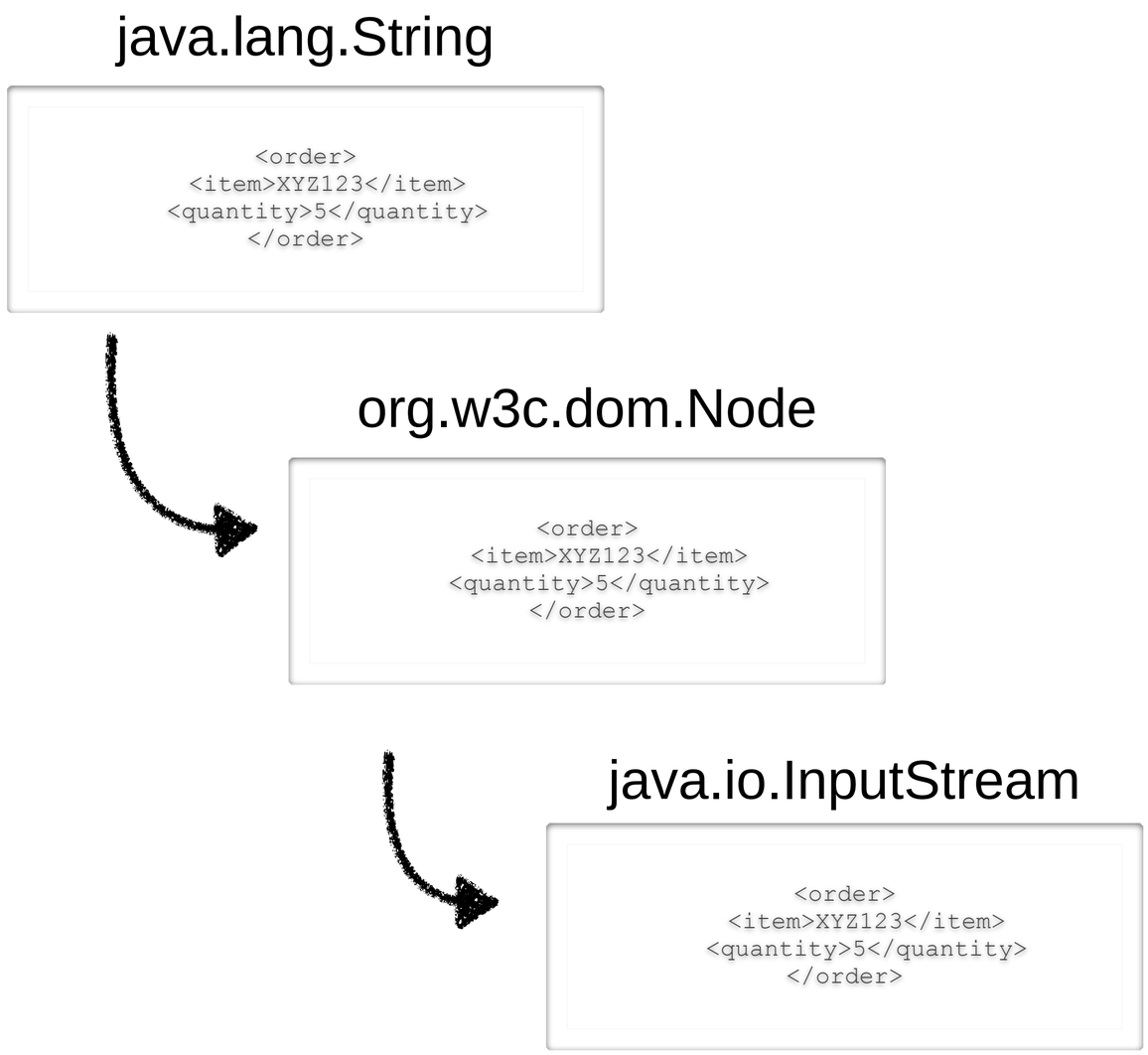
- Isolation from application logic
- Coverage
- Reuse
- Examples : Transformation, Policy

Transformation

- Change in data representation
- Change in data format
- Change in data itself
- Insidious coupling

Change In Representation

- @Converter
- In a bean or processor
  - `message.getBody(MyType.class)`
- In a route
  - `<convertBodyTo type="java.lang.String"/>`
  - `convertBodyTo(String.class)`
- Internal implementation detail



Change In Data

- Driven by business logic
- Enrich, Mine, Merge
- Velocity, XSLT, XQuery, Smooks, etc.
- Compatible with service contract

```
from("direct:SupplyCheck")
      .to("xslt://inventoryRequest.xsl")
      .to("direct:Inventory")
      .to("xslt://supplyResponse.xsl");
```

Change in Format

- What is a format ?
  - Java, XML, CSV, EDI, JSON, ...
- Camel Data Format
- Procedural

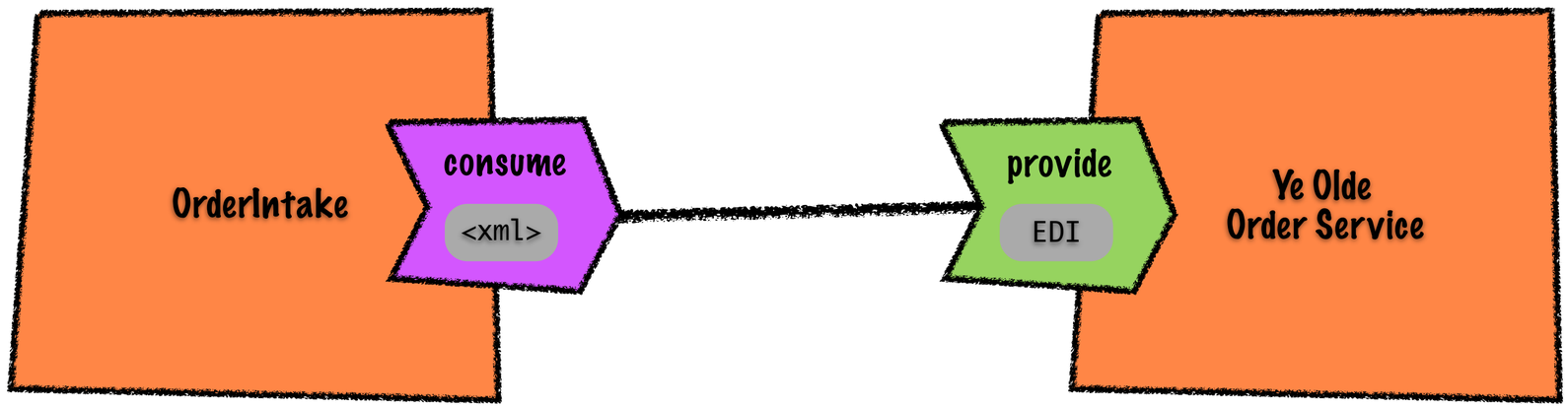
```
<camelContext id="camel"
              xmlns="http://camel.apache.org/schema/spring">
  <dataFormats>
    <jaxb id="myJaxb" contextPath="org.apache.camel.example"/>
  </dataFormats>
  <route>
    <from uri="direct:start"/>
    <marshal ref="myJaxb"/>
    <to uri="direct:marshalled"/>
  </route>
</camelContext>
```

Change in Format

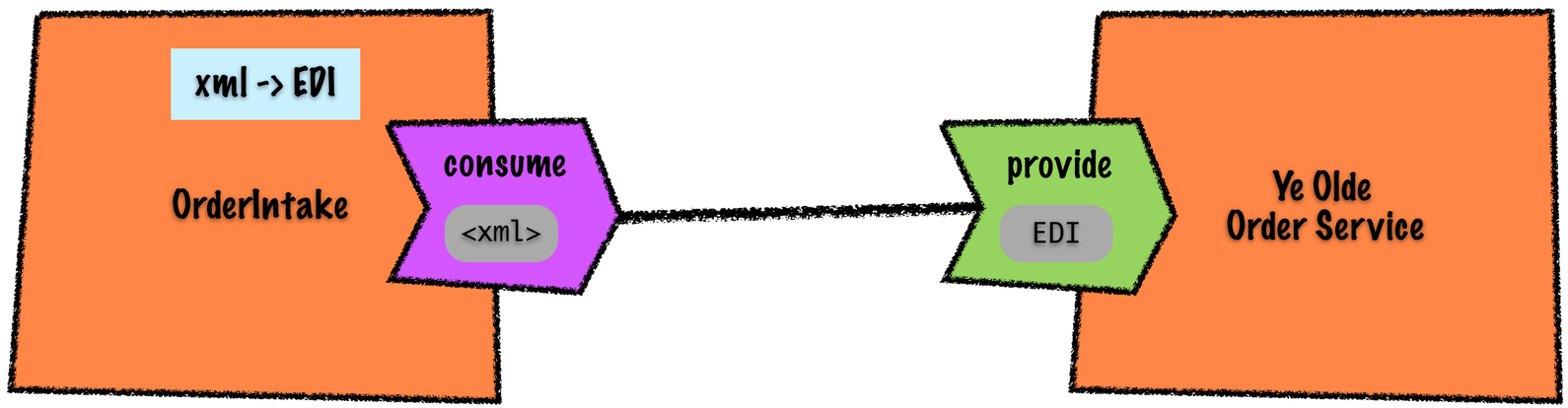
- What is a format ?
  - Java, XML, CSV, EDI, JSON, ...
- Camel Data Format
- Procedural

```
<camelContext id="camel"
              xmlns="http://camel.apache.org/schema/spring">
  <dataFormats>
    <jaxb id="myJaxb" contextPath="org.apache.camel.example"/>
  </dataFormats>
  <route>
    <from uri="direct:start"/>
    <marshal ref="myJaxb"/>
    <to uri="direct:marshalled"/>
  </route>
</camelContext>
```

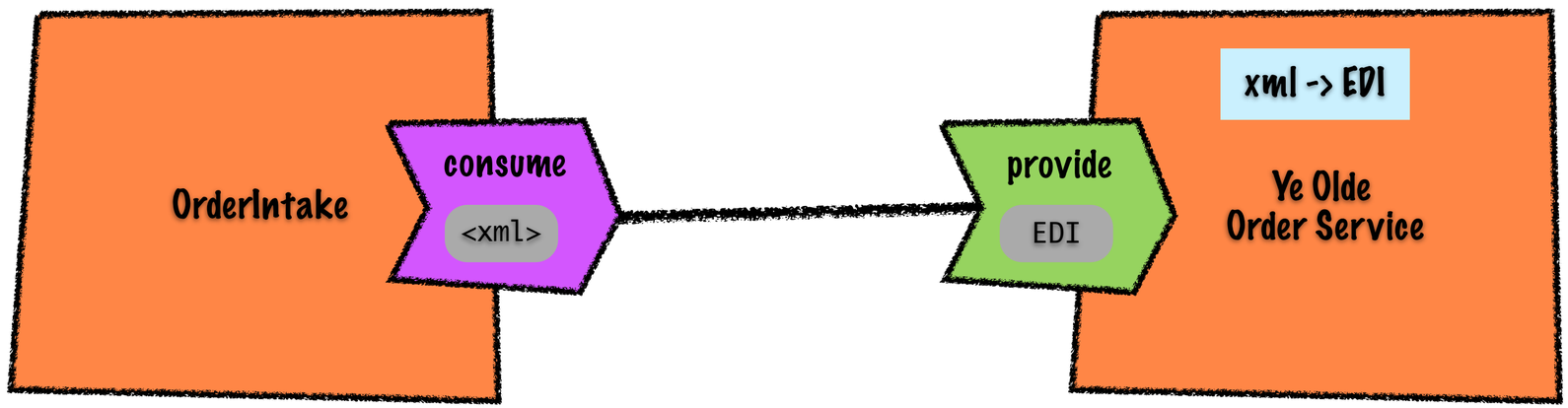
Declarative Transformation



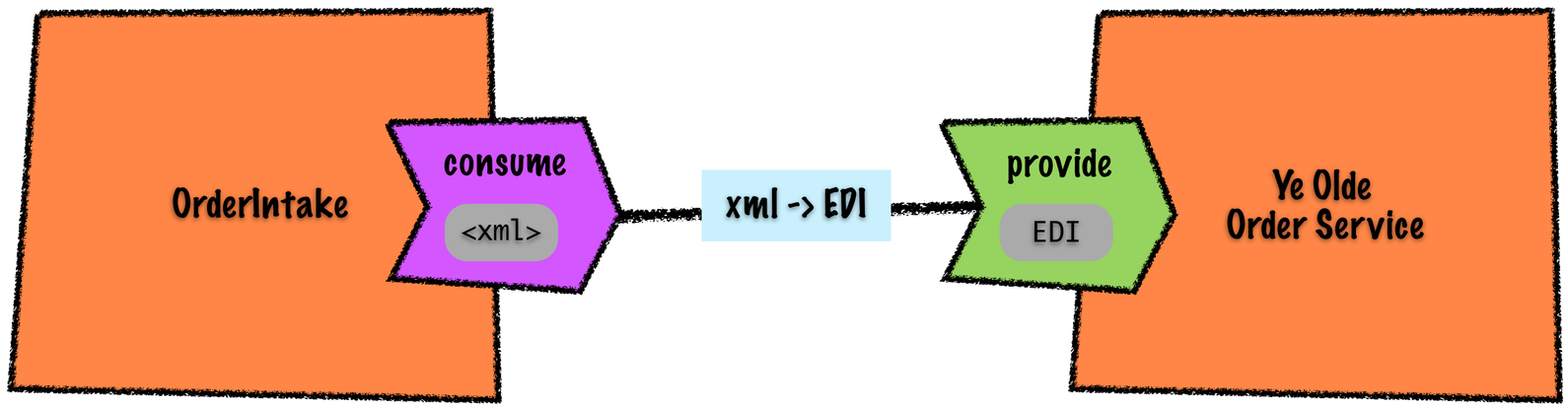
Declarative Transformation



Declarative Transformation



Declarative Transformation



Transformer Definition

```
import org.switchyard.transform.Transformer;

@Transformer(
      from = "{urn:switchyard-example:orders:1.0}submitOrder"
      to   = "edi:x12:850")
public String transform(Element from) {
   // Unimportant
}
```

Camel

- Interceptor
- Message name as metadata
- Would love to see this feature

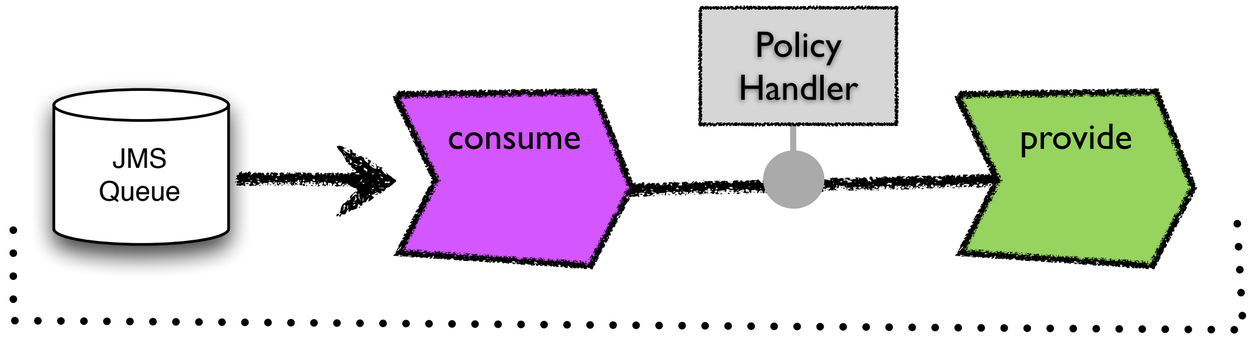
Takeaways

- Using the right tool for the job
- Don’t get coupled by data formats
- Favor declarative options over procedural
- Works for validation as well

Policy

- Declare the ‘what’, defer the ‘how’
- Common service policies
  - Transactional behavior
  - Security
  - Reliability

Policy



desired transaction scope

```
<service name="OrderService requires="propagatesTransaction">
  <camel:implementation.camel>
    ...
  </camel:implementation.camel>
</service>

<camel:binding.camel
  configURI="jms://OrderQueue?
             connectionFactory=JmsXA&amp;
             transactionManager=jtaTM&amp;
             transacted=true"/>
```

Policy In Camel

```
<route routePolicyRef="confidential">
   <from uri="direct:OrderService"/>
   <to uri="direct:SomewhereElse"/>
</route>
```

```
<route>
   <from uri="jetty:https://0.0.0.0/Orders"/>
   <to uri="direct:OrderService"/>
</route>
```

Takeaways

- Power for the developer
- Flexibility for the deployer
- Favor declarative over procedural

Governance

- Design-time
  - Service Repository Integration
- Runtime
  - Service Monitoring

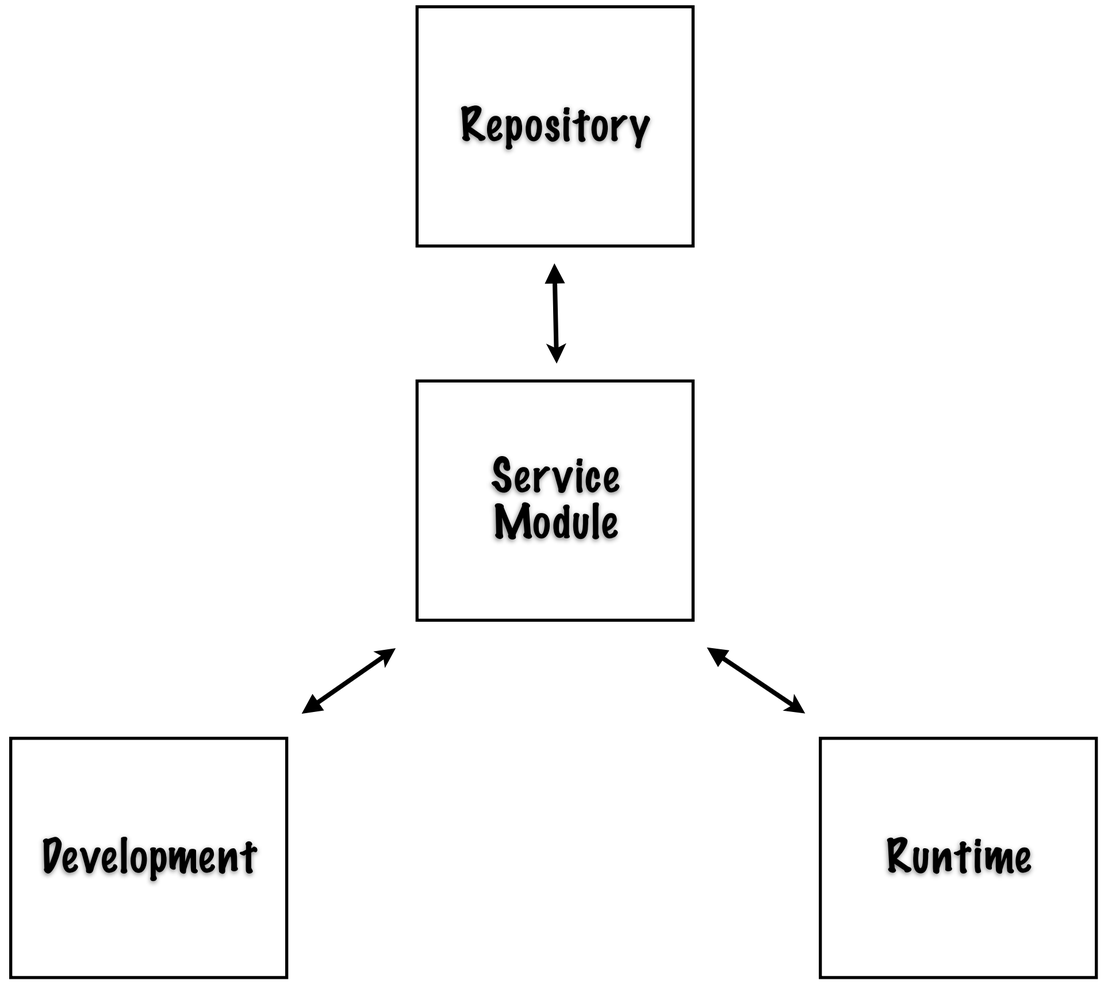
Service Repository

- There are artifacts related to your application services that need to be:
  - Published
  - Discovered
  - Related
  - Managed
  - Versioned

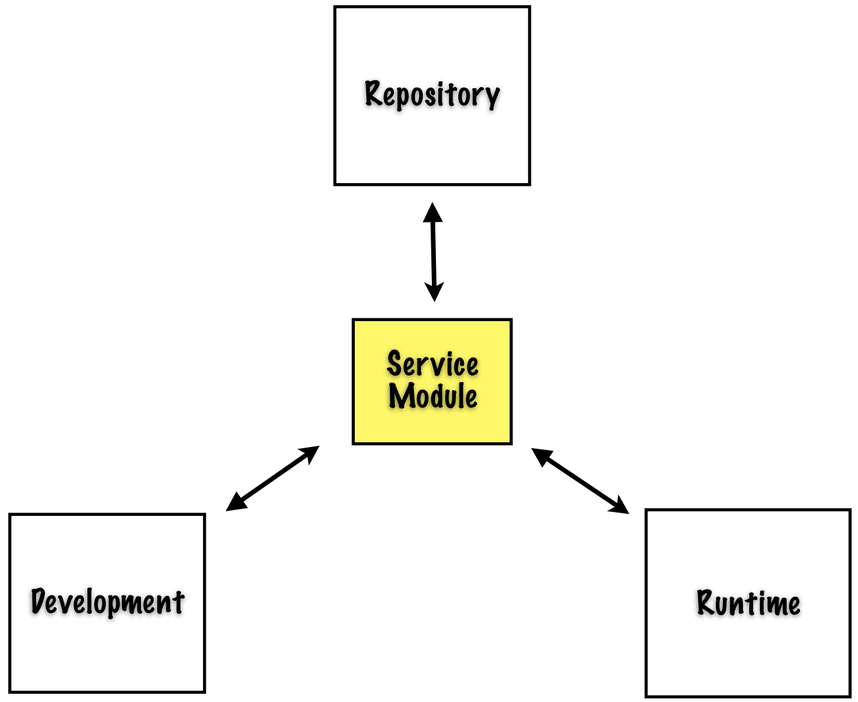
Basic Use Cases

- Publication of service artifacts from an application project
- Import service artifacts into an application project
- Application and artifacts are deployed to runtime

Basic Architecture



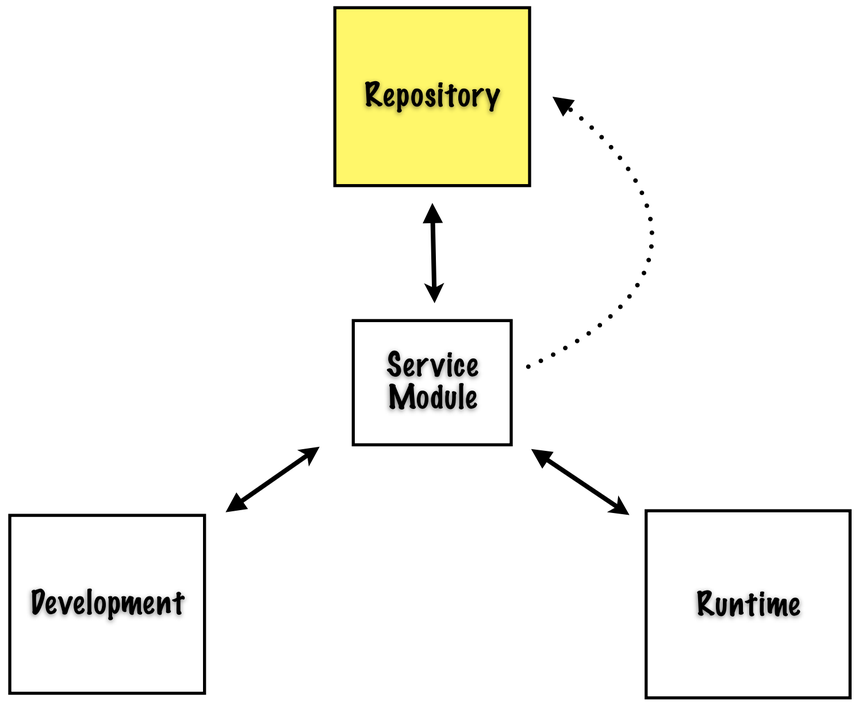
Basic Solution




```
<project xmlns="http://maven.apache.org/POM/4.0.0">
    <modelVersion>4.0.0</modelVersion>
    <groupId>org.example</groupId>
    <artifactId>MyServiceArtifacts</artifactId>
    <version>3.2.0</version>
    <name>My Service Artifacts</name>
</project>
```

```
src/main/java:
   MyService.java

src/main/resources:
   MyService.wsdl
   MyTypes.xsd
```

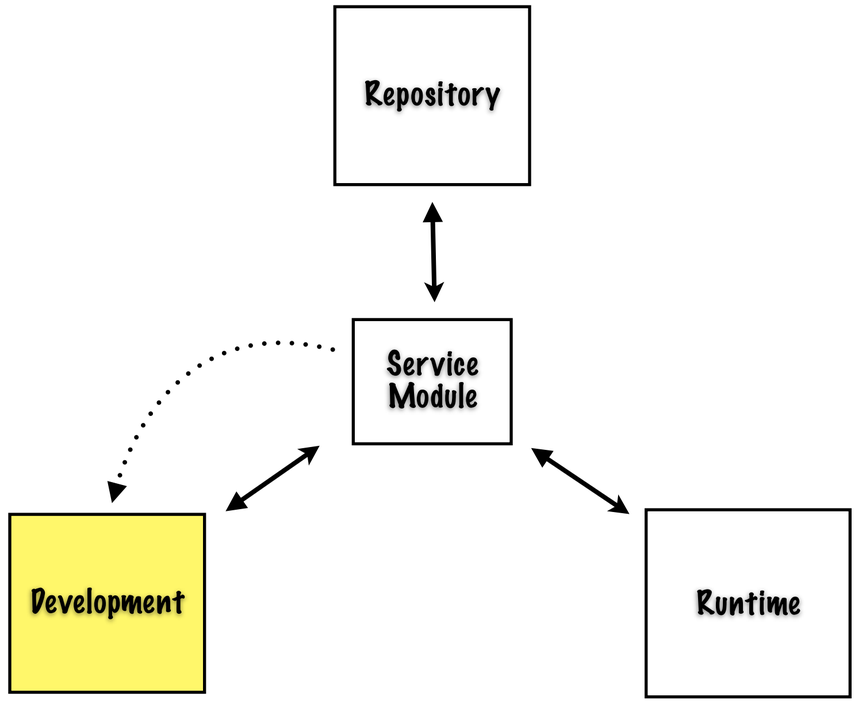
Basic Solution




```
> mvn install
```

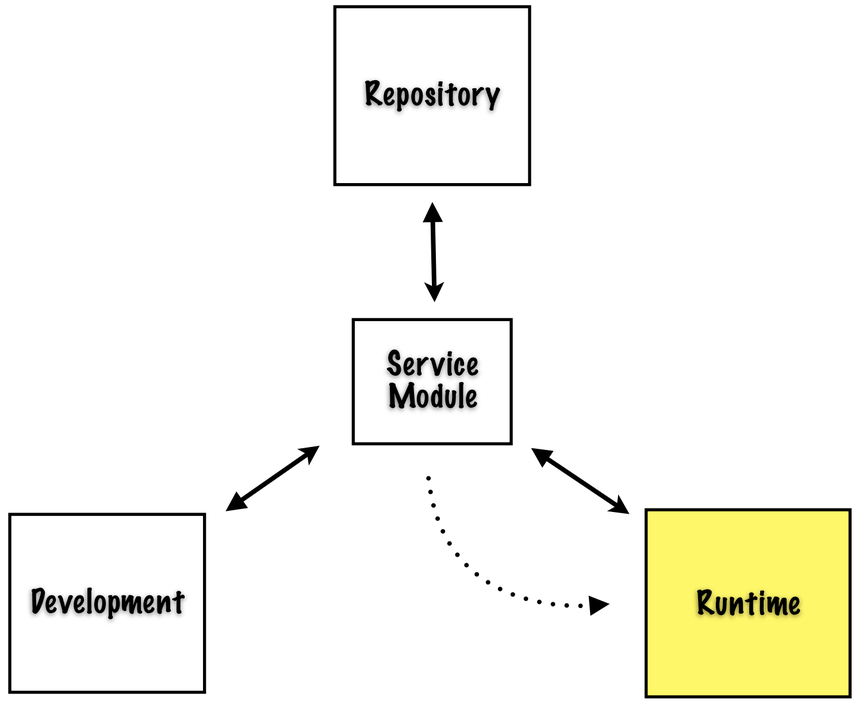
```
> mvn deploy
```

Basic Solution




```
<dependency>
    <groupId>org.example</groupId>
    <artifactId>MyServiceArtifacts</artifactId>
    <version>3.2.0</version>
</dependency>
```

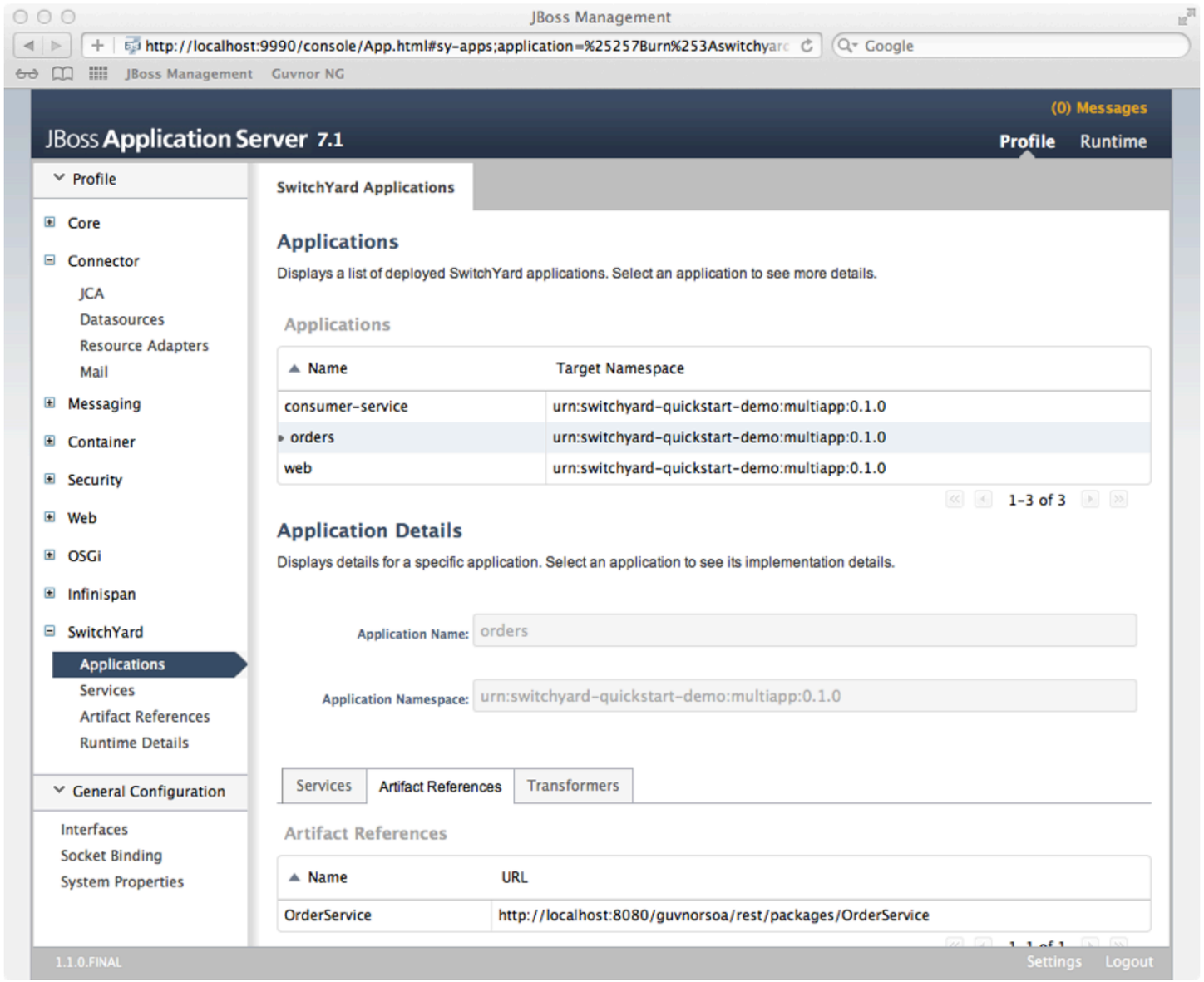
Basic Solution



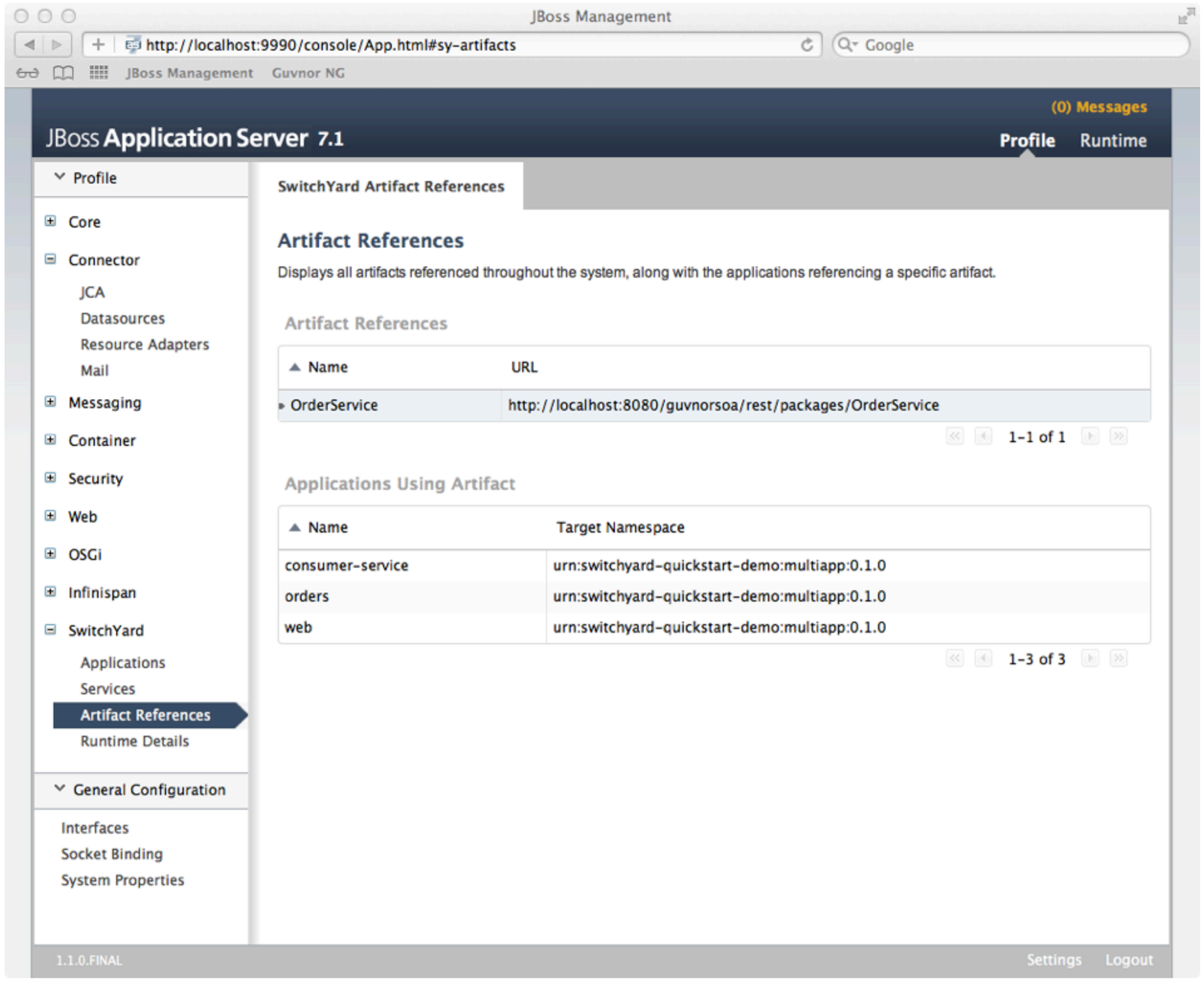

```
<plugin>
   <groupId>org.apache.maven.plugins</groupId>
   <artifactId>maven-jar-plugin</artifactId>
   <configuration>
     <archive>
       <manifestEntries>
         <Dependencies>deployment.MyServiceArtifacts.jar</Dependencies>
       </manifestEntries>
     </archive>
   </configuration>
</plugin>
```

```
<artifacts>
   <artifact
      name="MyServiceArtifacts" url="[repo-url]"/>
</artifacts>
```

Runtime


JBoss Management
http://localhost:9990/console/App.html#sy-apps;application=%25257Burn%253Aswitchyard
Google
JBoss Management
Guvnor NG
(0) Messages
JBoss Application Server 7.1
Profile
Runtime
Profile
Core
Connector
JCA
Datasources
Resource Adapters
Mail
Messaging
Container
Security
Web
OSGi
Infinispan
SwitchYard
Applications
Services
Artifact References
Runtime Details
General Configuration
Interfaces
Socket Binding
System Properties
SwitchYard Applications
Applications
Displays a list of deployed SwitchYard applications. Select an application to see more details.
Applications
Name
Target Namespace
consumer-service
urn:switchyard-quickstart-demo:multiapp:0.1.0
orders
urn:switchyard-quickstart-demo:multiapp:0.1.0
web
urn:switchyard-quickstart-demo:multiapp:0.1.0
1-3 of 3
Application Details
Displays details for a specific application. Select an application to see its implementation details.
Application Name:
orders
Application Namespace:
urn:switchyard-quickstart-demo:multiapp:0.1.0
Services
Artifact References
Transformers
Artifact References
Name
URL
OrderService
http://localhost:8080/guvnorsoa/rest/packages/OrderService
1.1.0.FINAL
Settings
Logout

Runtime

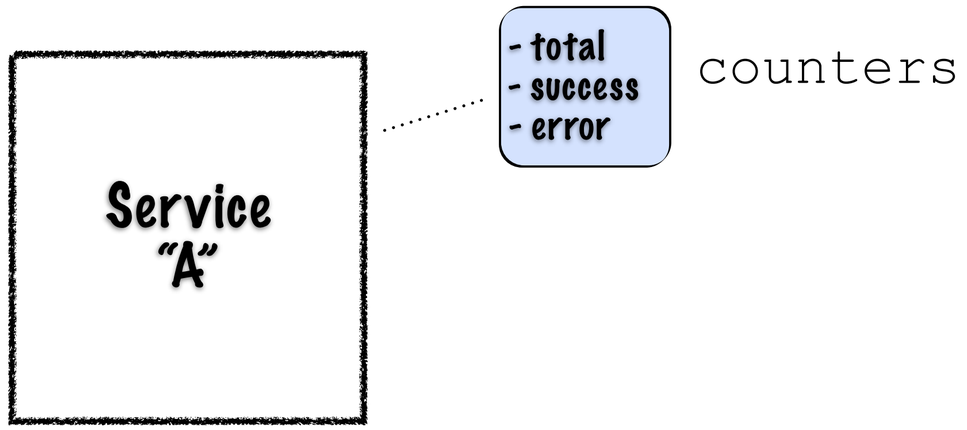

JBoss Management
http://localhost:9990/console/App.html#sy-artifacts
Google
JBoss Management
Guvnor NG
(0) Messages
JBoss Application Server 7.1
Profile
Runtime
Profile
Core
Connector
JCA
Datasources
Resource Adapters
Mail
Messaging
Container
Security
Web
OSGi
Infinispan
SwitchYard
Applications
Services
Artifact References
Runtime Details
General Configuration
Interfaces
Socket Binding
System Properties
SwitchYard Artifact References
Artifact References
Displays all artifacts referenced throughout the system, along with the applications referencing a specific artifact.
Artifact References
Name
URL
OrderService
http://localhost:8080/guvnorsoa/rest/packages/OrderService
1-1 of 1
Applications Using Artifact
Name
Target Namespace
consumer-service
urn:switchyard-quickstart-demo:multiapp:0.1.0
orders
urn:switchyard-quickstart-demo:multiapp:0.1.0
web
urn:switchyard-quickstart-demo:multiapp:0.1.0
1-3 of 3
1.1.0.FINAL
Settings
Logout

Takeaways

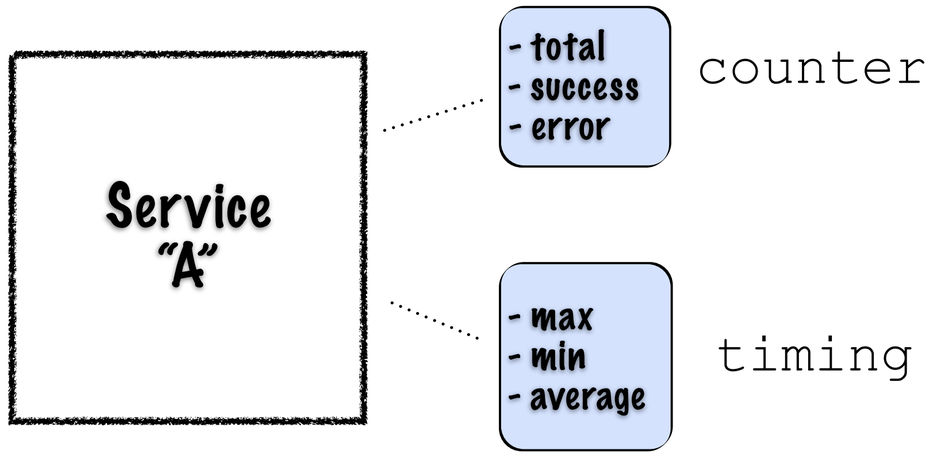
- Maven is a pragmatic DIY option
  - Publish, discover, version, manage
- A little metadata can get you the rest of the way
  - Application -> Artifact module
- Modularity provides sanity

Service Monitoring

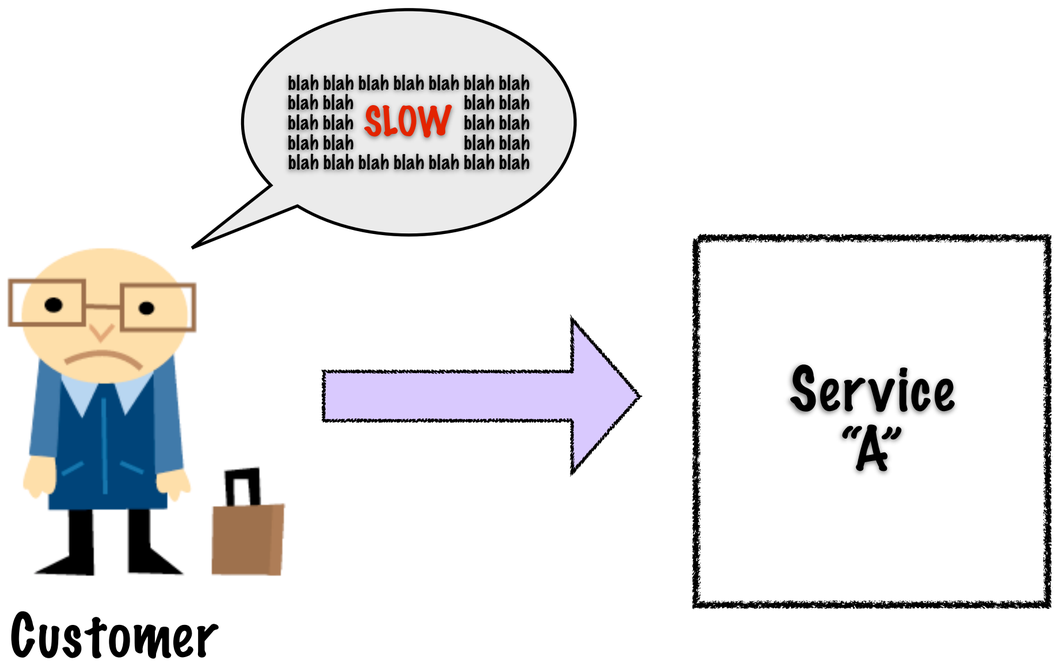
Service Monitoring



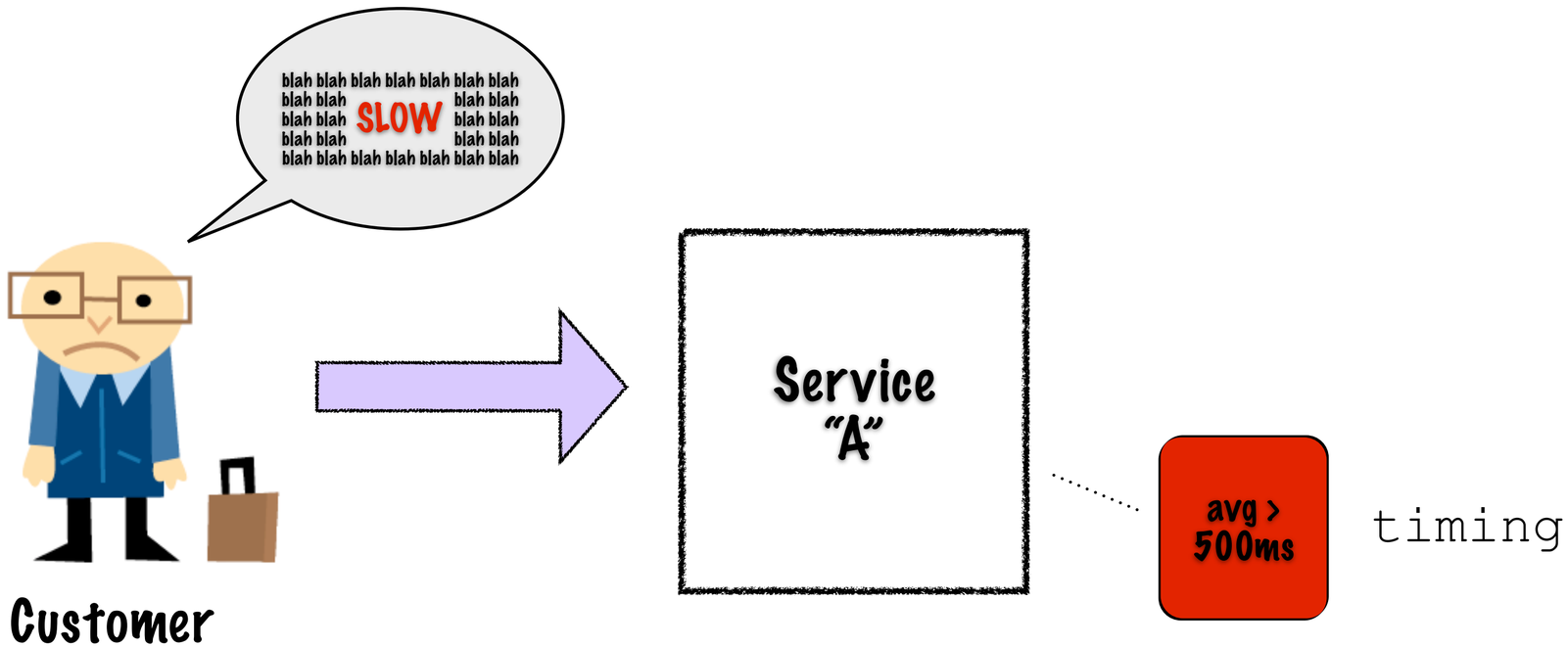
Service Monitoring



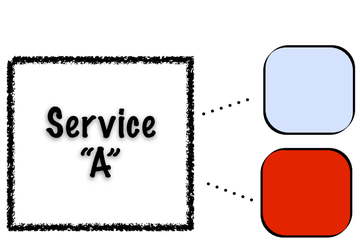
Metrics In Action



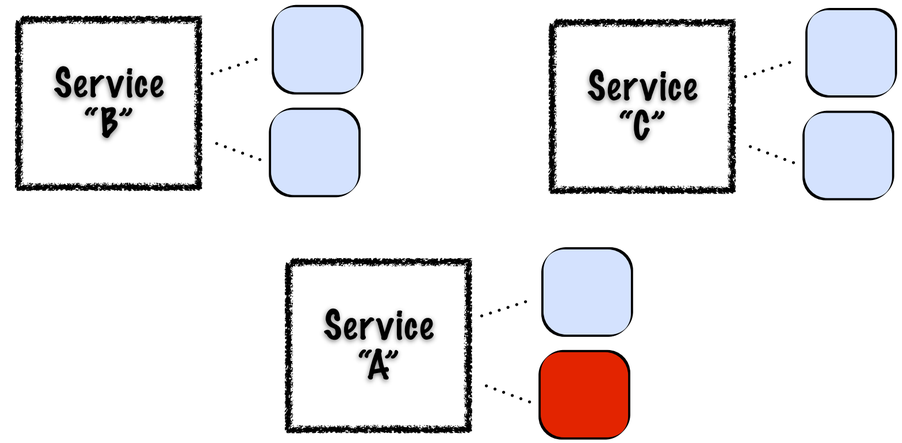
Metrics In Action



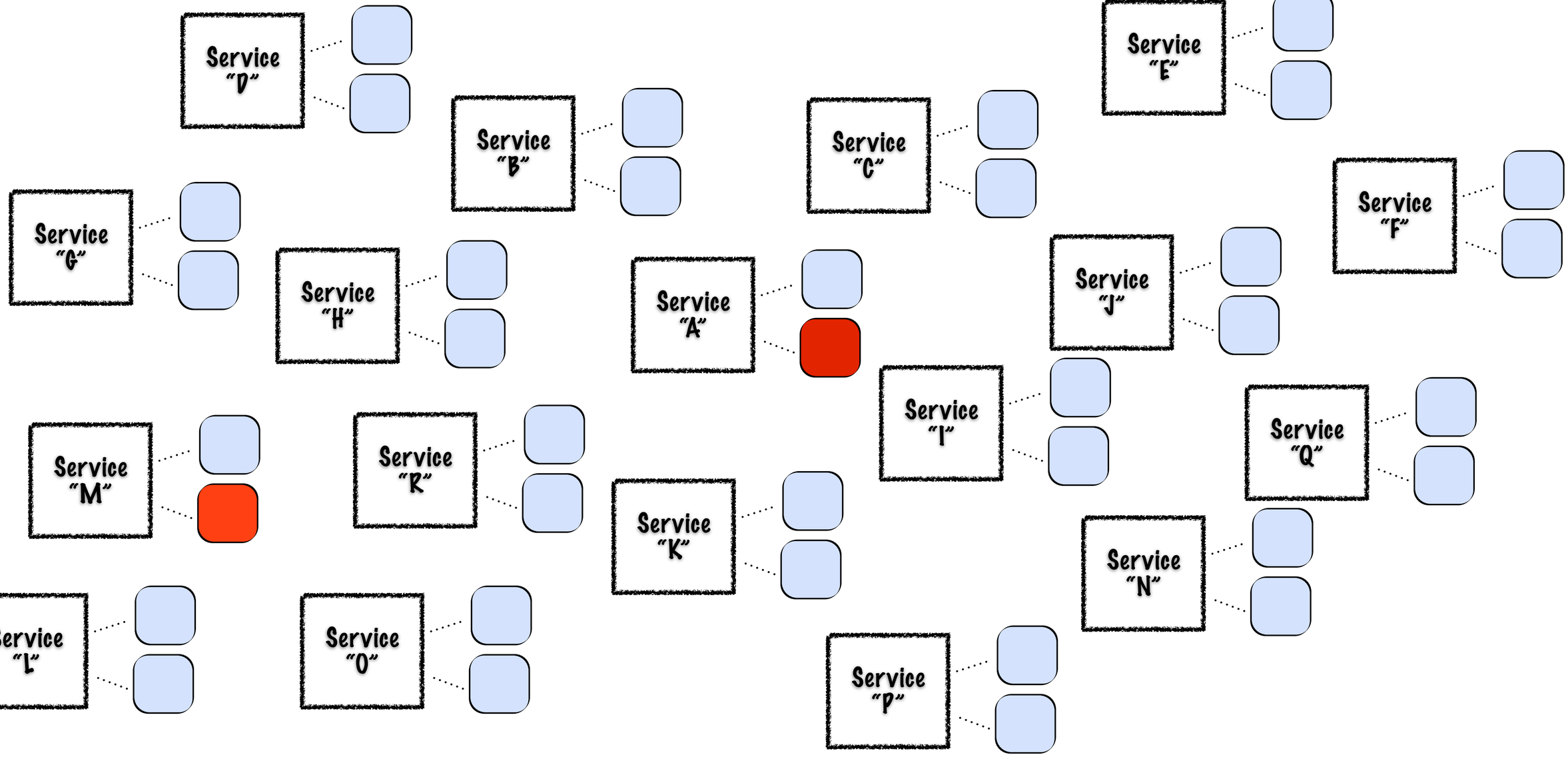
Metric Explosion



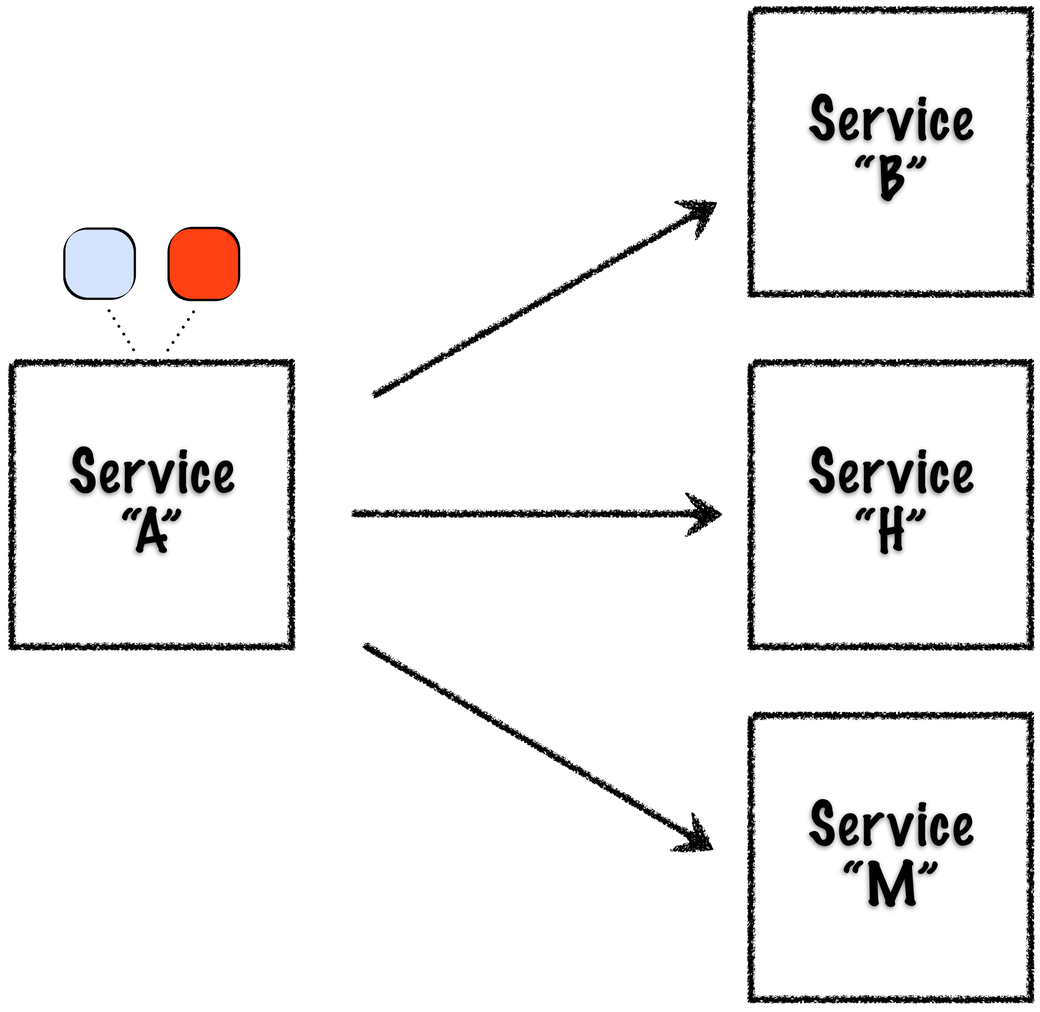
Metric Explosion



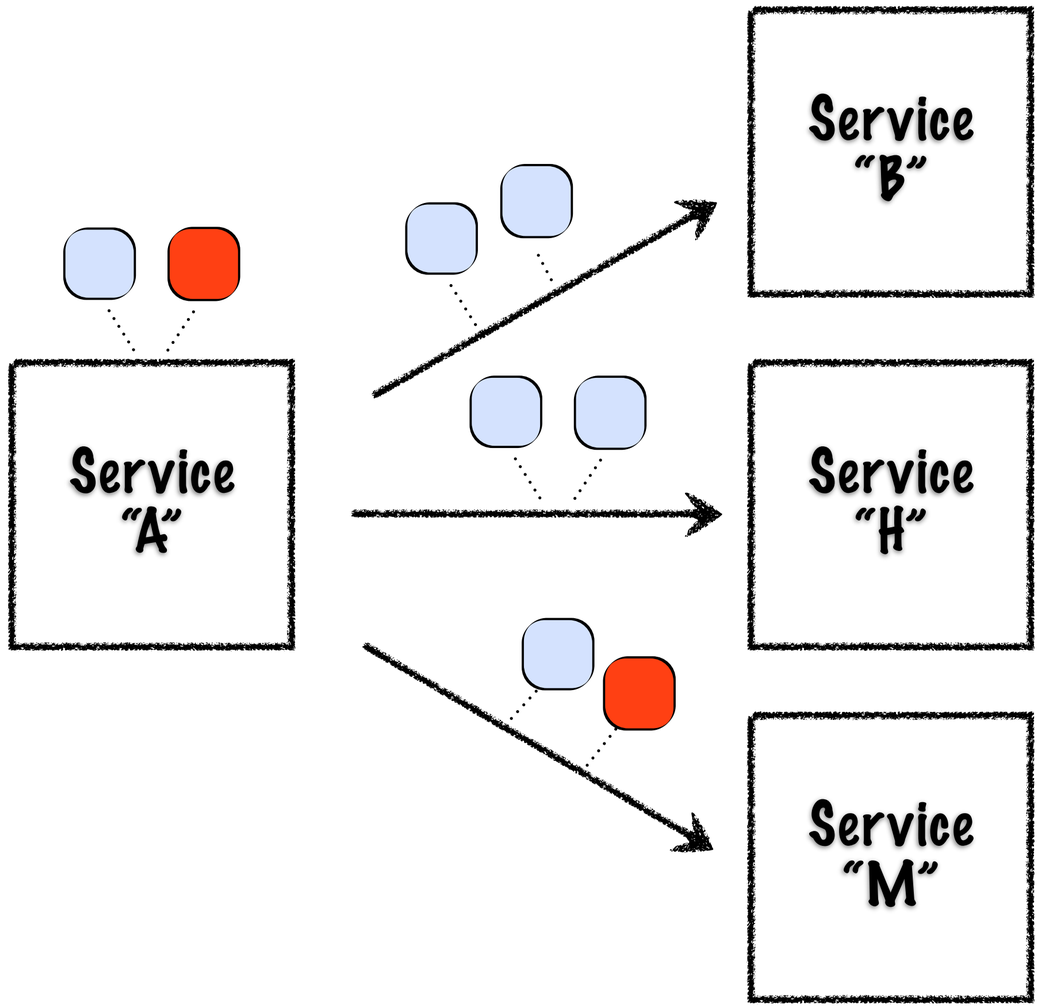
Metric Explosion



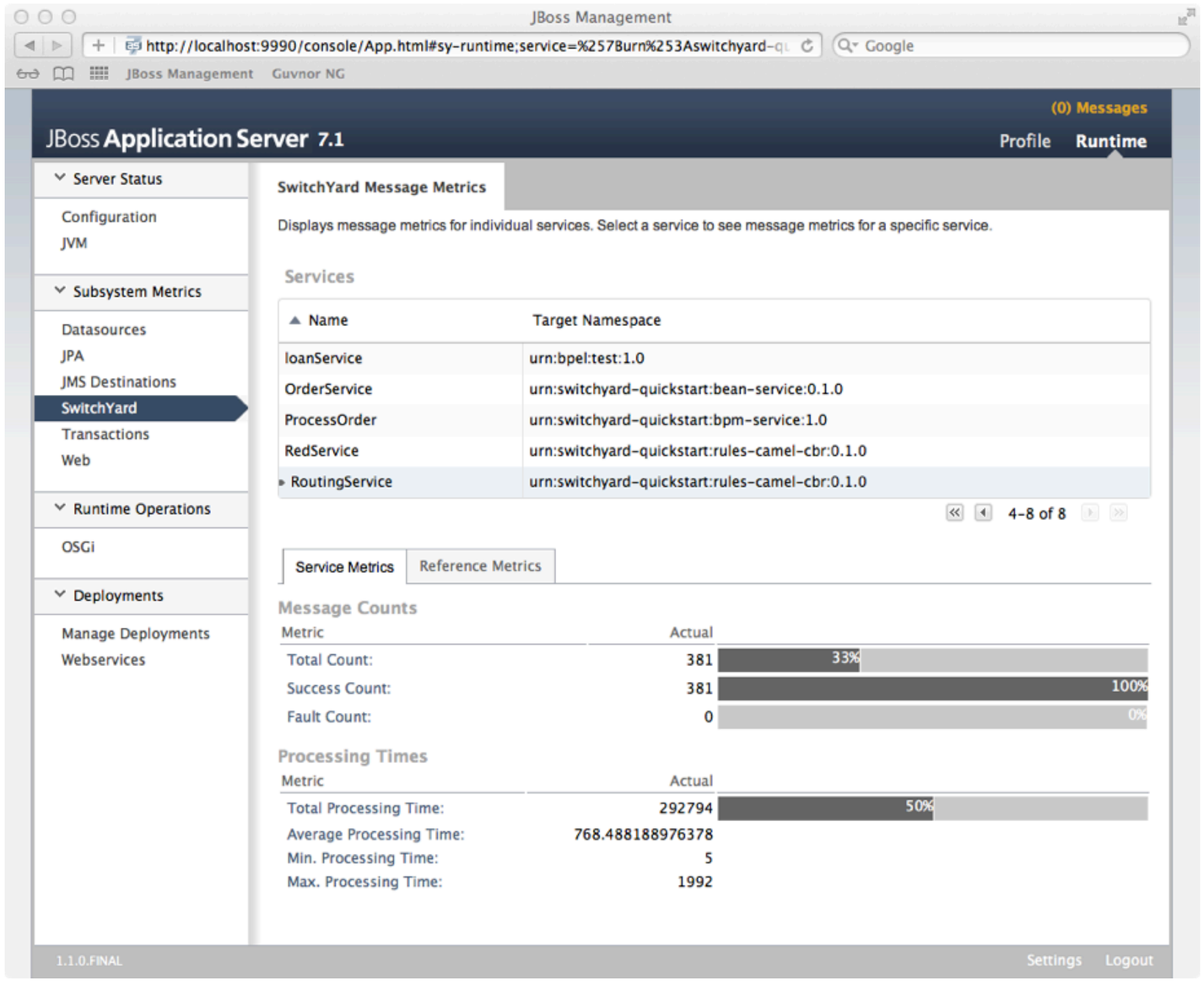
Service References



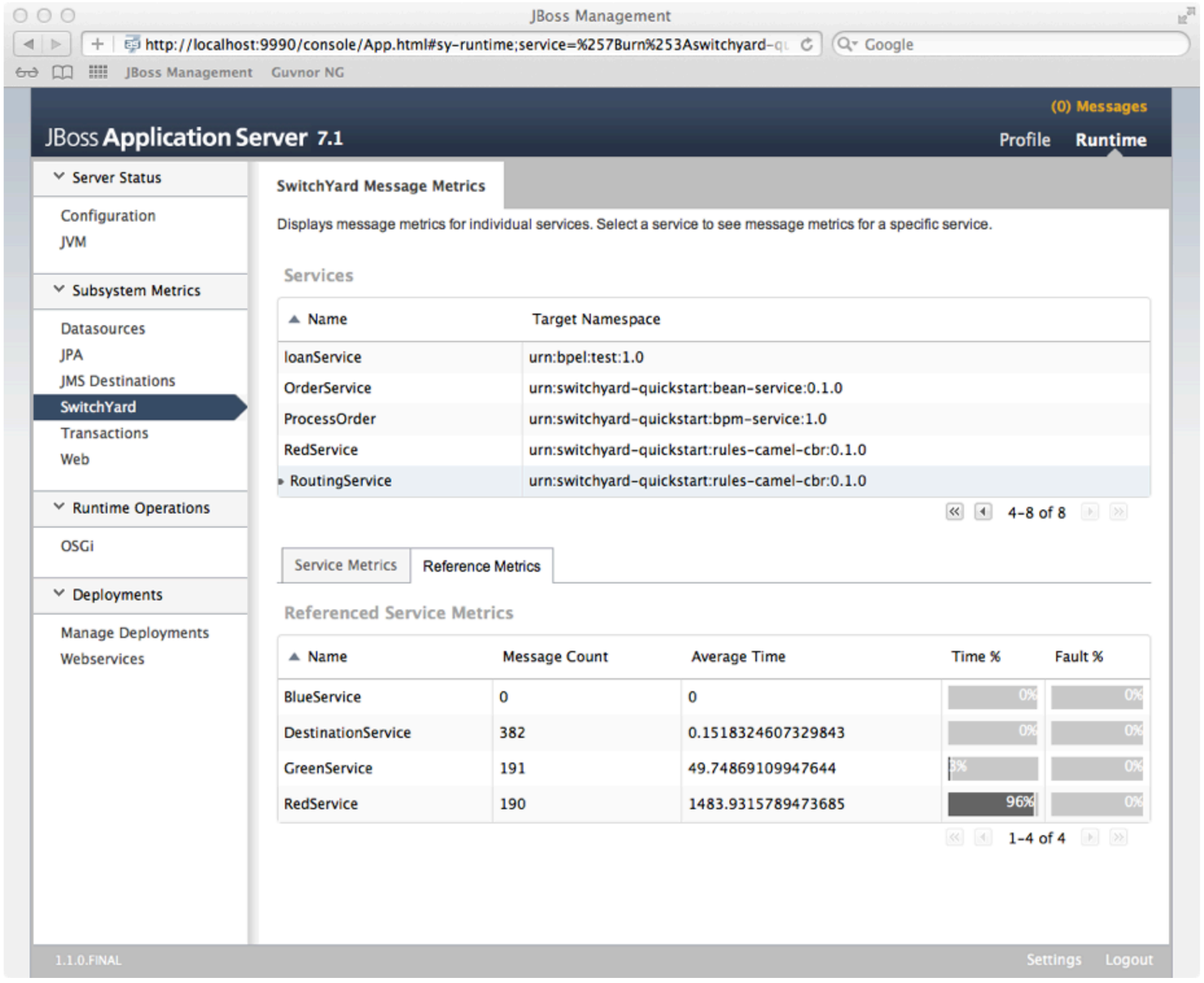
Service References



Service Metrics



Reference Metrics



Takeaways

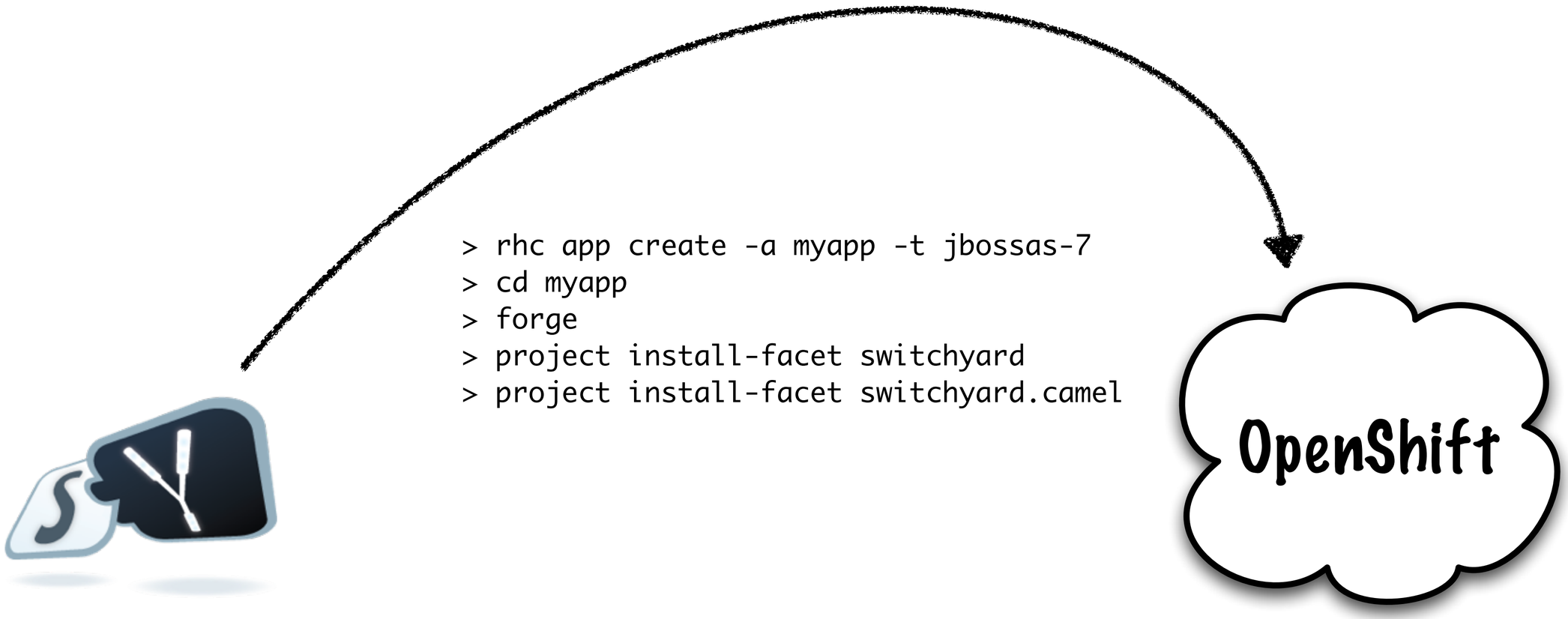
- Focus is critical
- Infrastructure vs. Service
  - JMS queue, web service, ftp directory
  - Order processing for Europe
- Context
  - Relationships
  - Correlation

Cloud

- IaaS vs. PaaS
- Free
  - Speech
  - Beer too!
- Easy

OPENSHIFT

Camel on OpenShift



More Info

- SwitchYard

  http://jboss.org/switchyard

- JBoss

  http://redhat.com/products/jbossenterprisemiddleware

- OpenShift

  http://openshift.redhat.com

- SCA

  http://oasis-open.org/committees/sca-assembly

Q & A