JBoss Community

Today's Plan

- Session 1 : Introduction & Overview
- Lab 1
- Session 2 : Service-Oriented Integration
- Lunch
- Lab 2
- Session 3 : Enterprise Service Development
- Lab 3
- Session 4 : Futures, Feedback, and Q&A

Introduction and Overview

JBoss Community

Who Am I ?

- Core Developer at JBoss
- Project Lead for SwitchYard
- Last gig was at Sun
  - ESB/EAI/B2B product and standards development

Agenda

- Project Background
- Service-Oriented Applications
- Feature Overview
- Q & A

Introducing SwitchYard

- New JBoss community project with the goal of creating our next generation Enterprise Service Bus
- What happened to JBoss ESB?
  - Same team
  - Active development continues in support of SOA-P
- Taking the next evolutionary step
  - Focus on consistent, intuitive user experience
  - Refactor core to eliminate known pain points
  - Leverage standards and complimentary technologies

Introducing SwitchYard

- Why a separate project?
  - Isolate disruptive changes
  - Focus community
  - Implement faster, get feedback sooner
- Project goals aligned
  - End deliverable is a better SOA Platform

Activity

- SwitchYard
  - Milestone 1 - February, 2011
  - 0.1 release - June, 2011
  - 0.2 release - August, 2011
  - 0.*n* releases every 8-10 weeks
- JBoss ESB
  - SOA Platform 5.1 - March, 2011
  - JBoss ESB 4.10 - August 2011
  - SOA Platform 5.2 - November, 2011

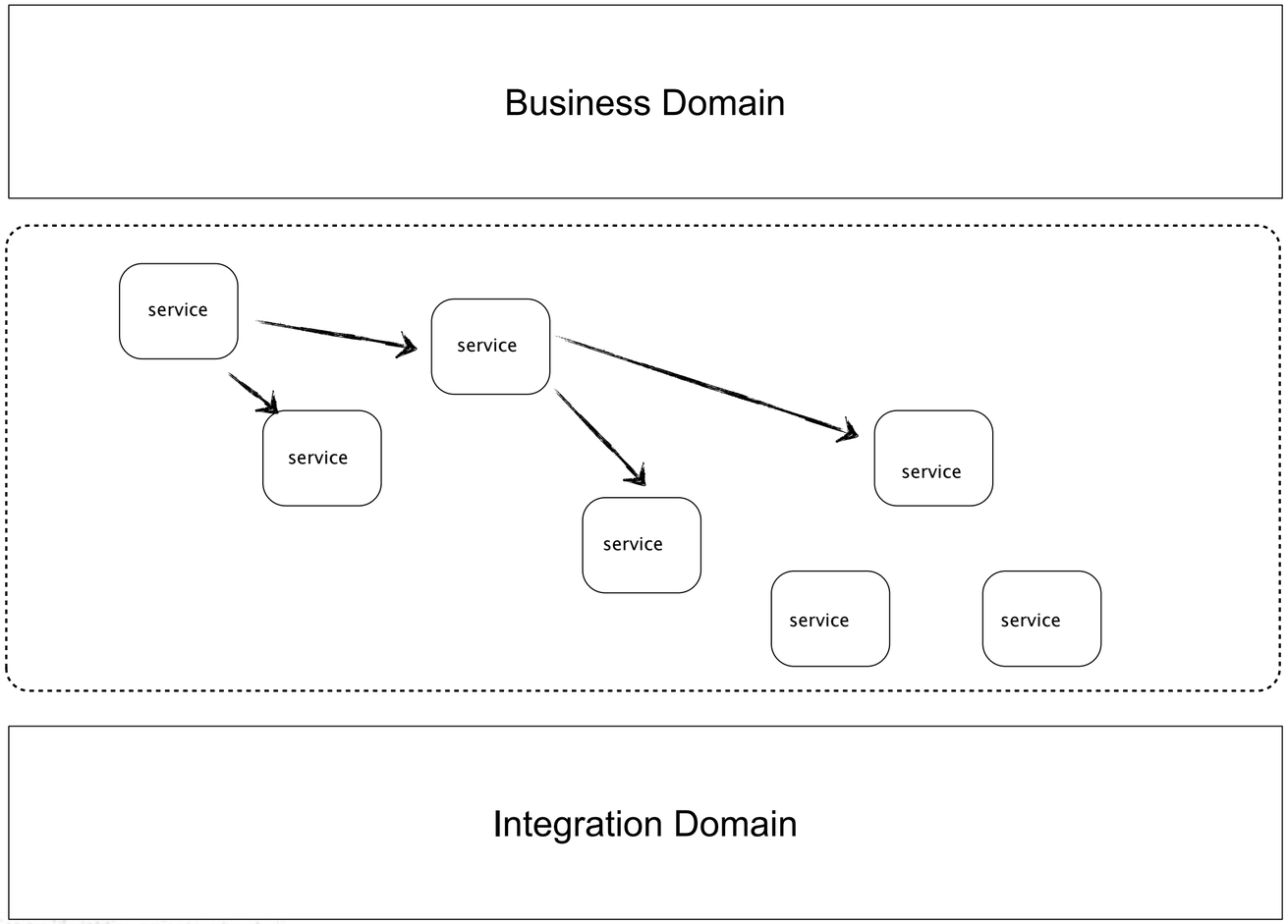
What's an ESB ?

Business Domain

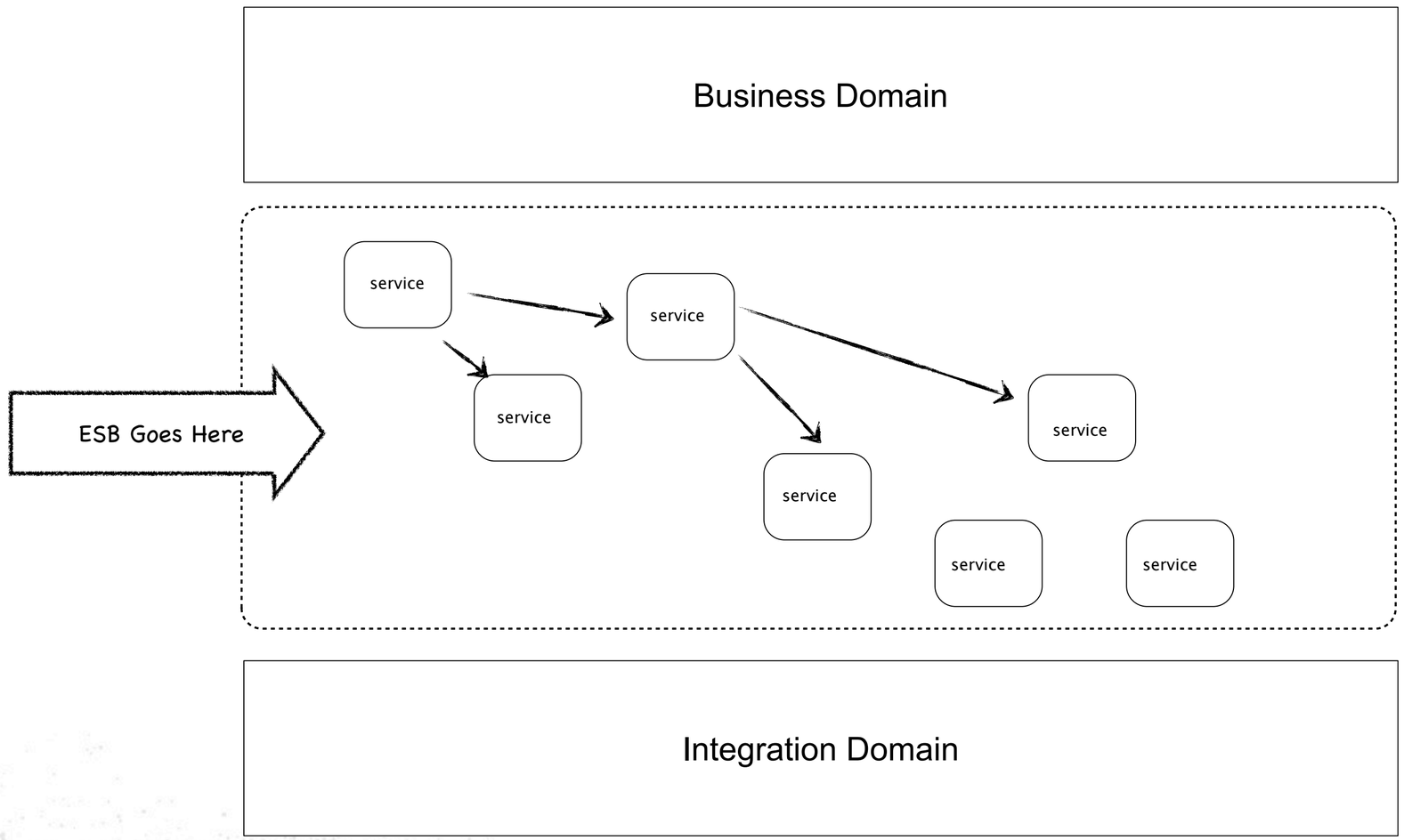
Integration Domain

JBoss Community

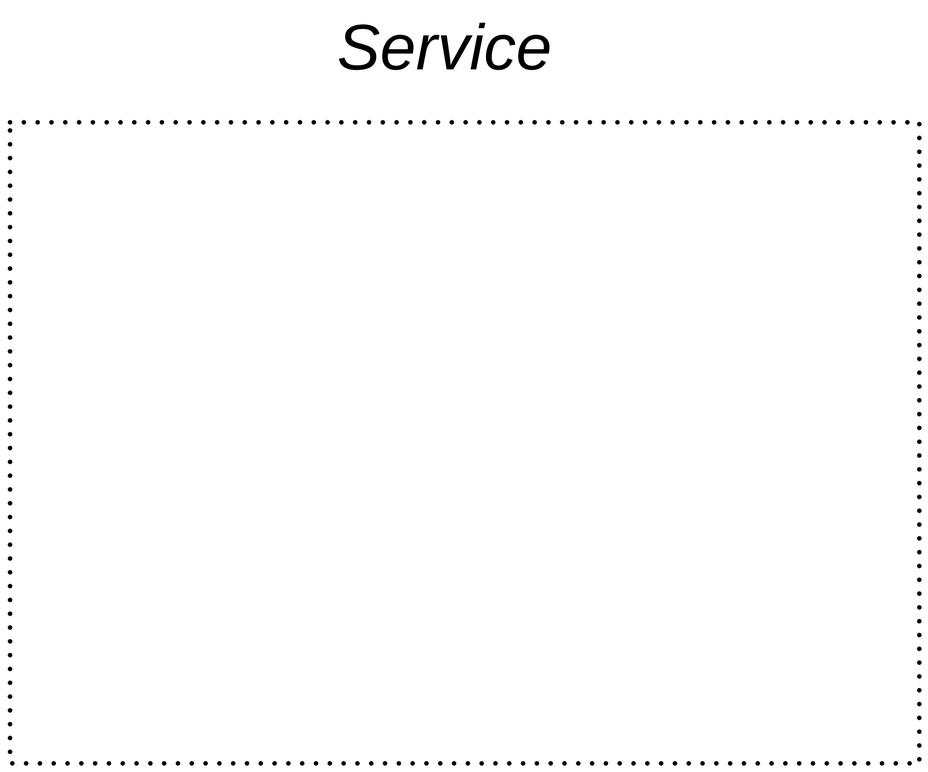
What's an ESB ?


Business Domain
service
service
service
service
service
service
service
service
Integration Domain

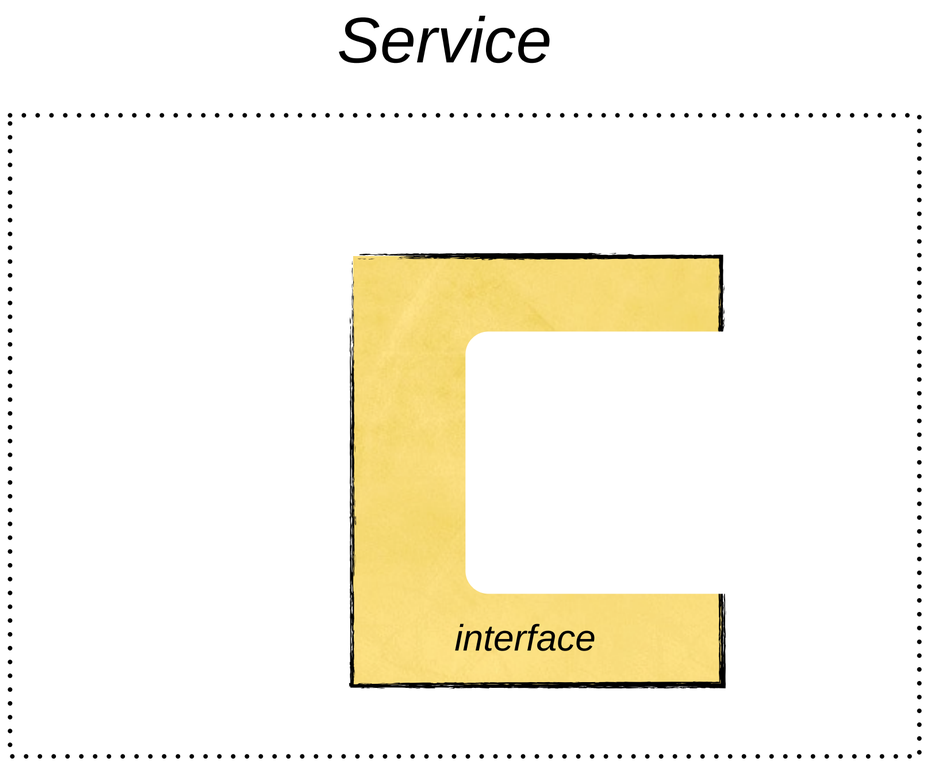
What's an ESB ?


Business Domain
service
service
service
service
service
service
service
service
ESB Goes Here
Integration Domain

A Superficial Guide to Service-Oriented Principles

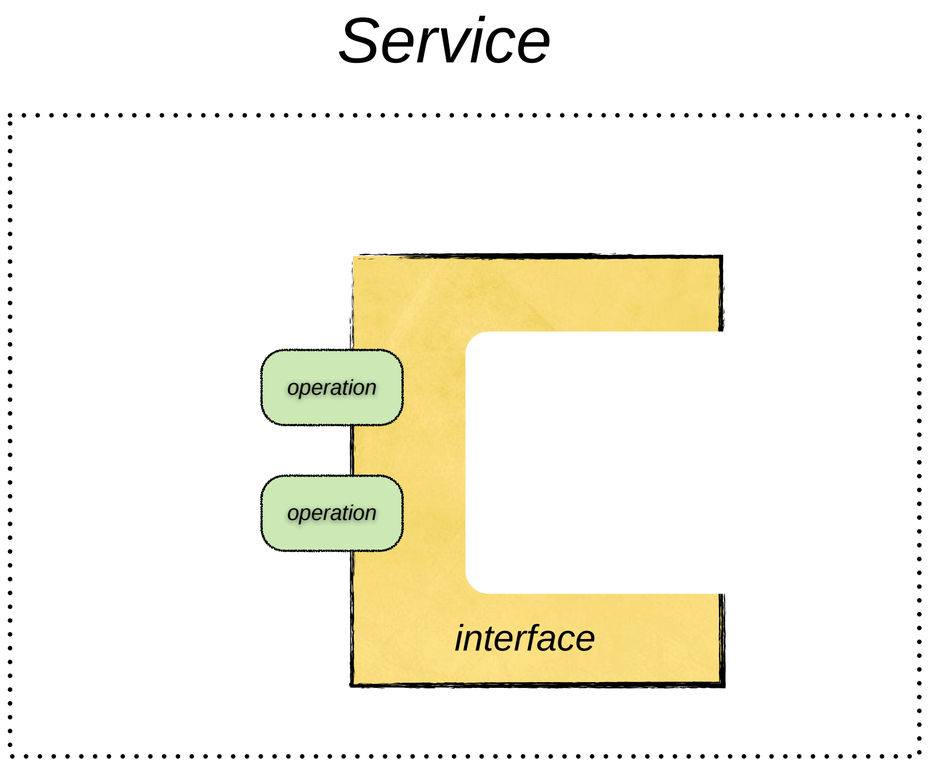


A Superficial Guide to Service-Oriented Principles

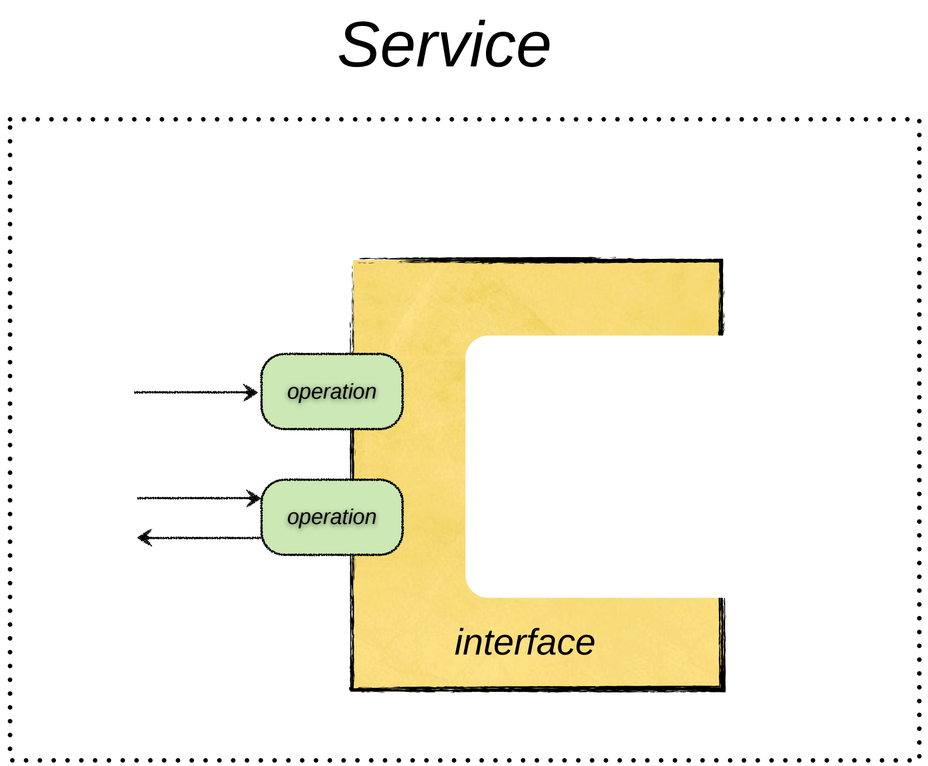


JBoss Community

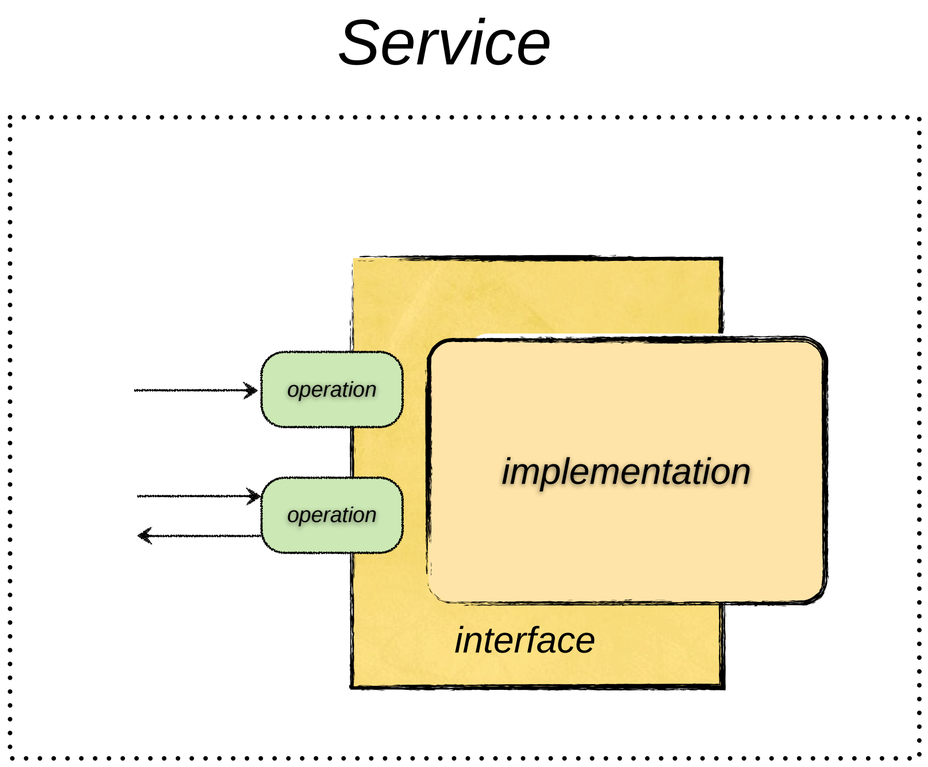
A Superficial Guide to Service-Oriented Principles



A Superficial Guide to Service-Oriented Principles

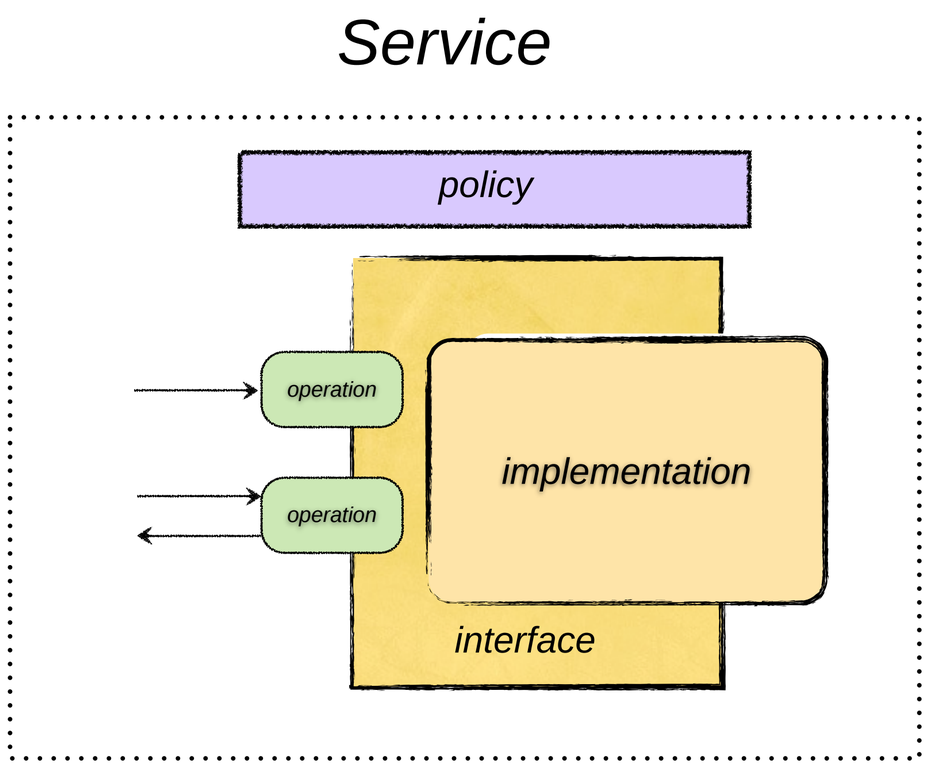


A Superficial Guide to Service-Oriented Principles



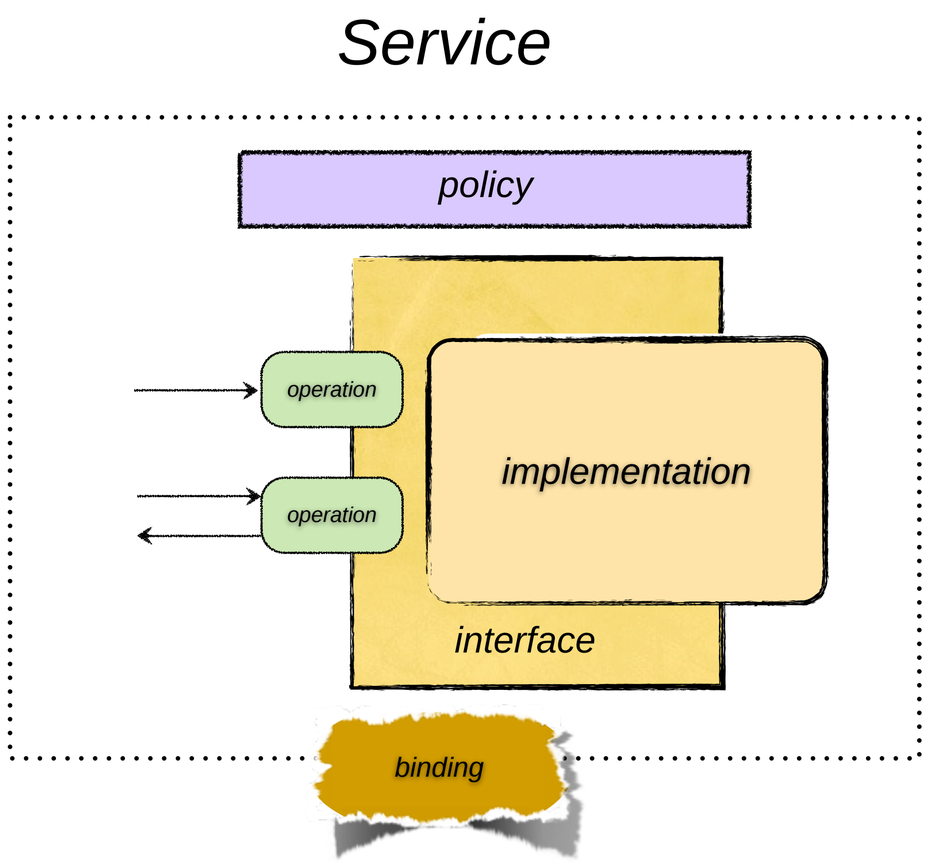
JBoss Community

A Superficial Guide to Service-Oriented Principles



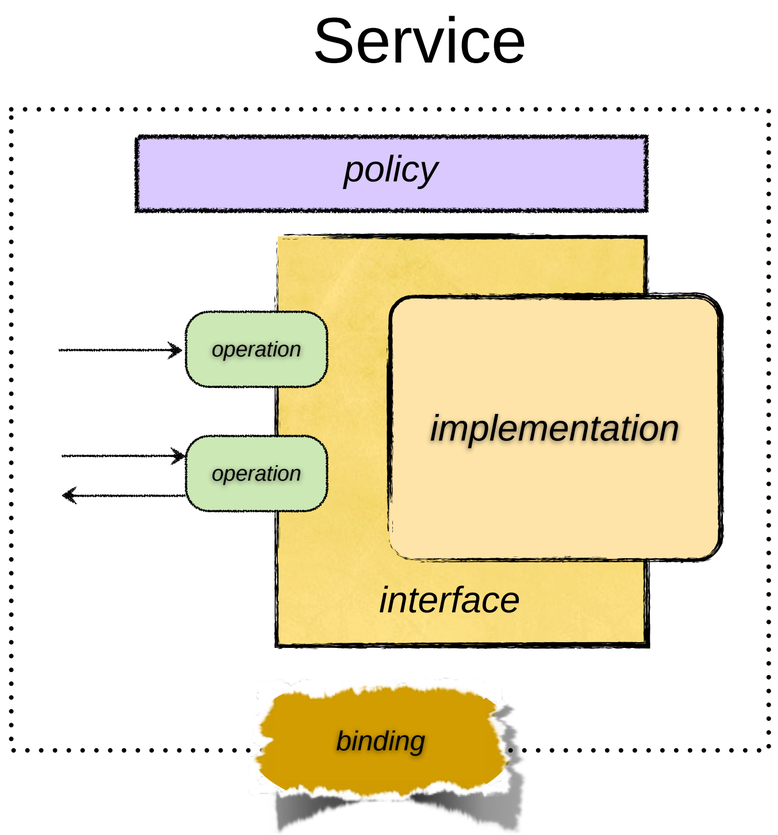
JBoss Community

A Superficial Guide to Service-Oriented Principles

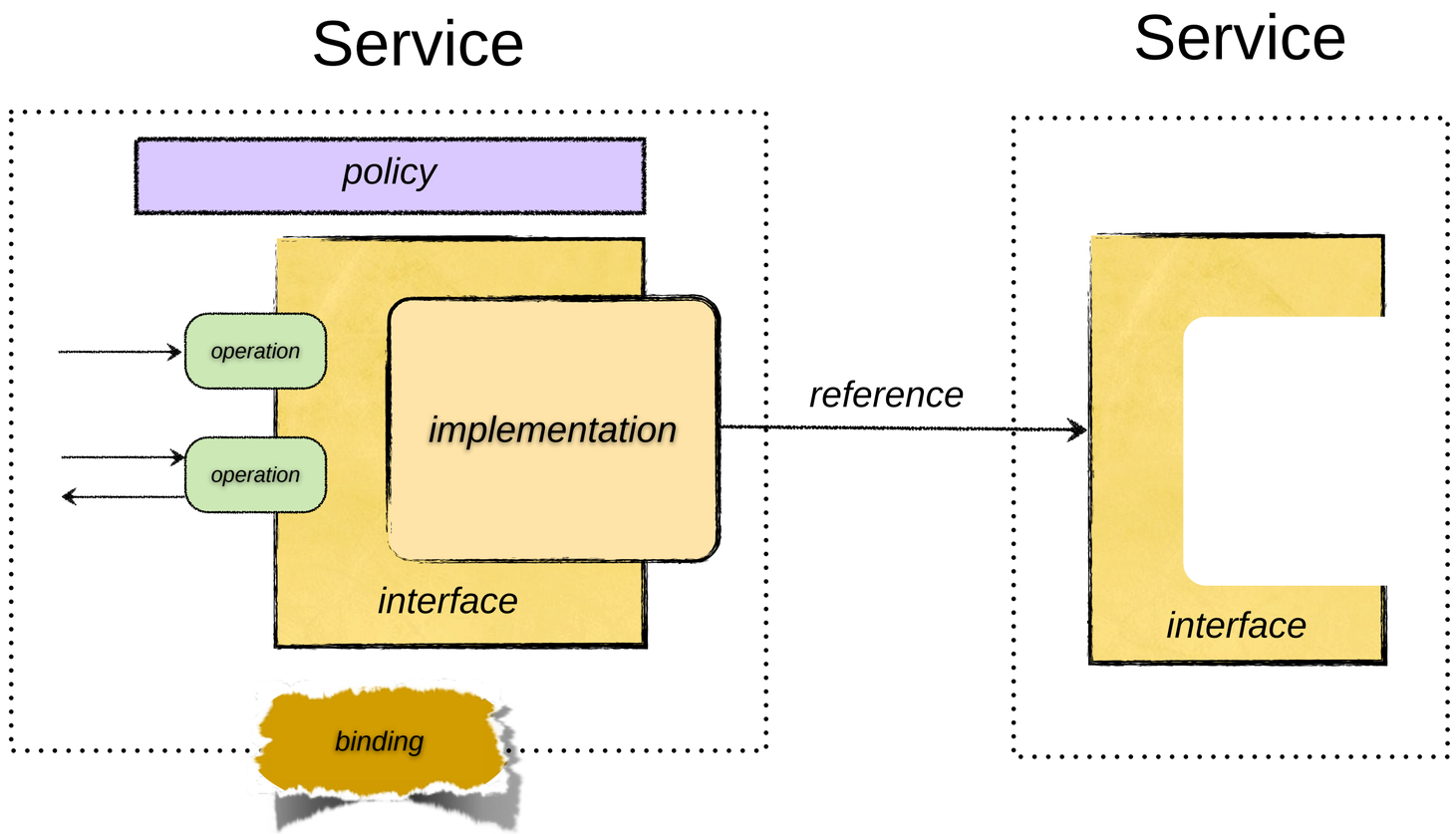


JBoss Community

A Superficial Guide to Service-Oriented Principles

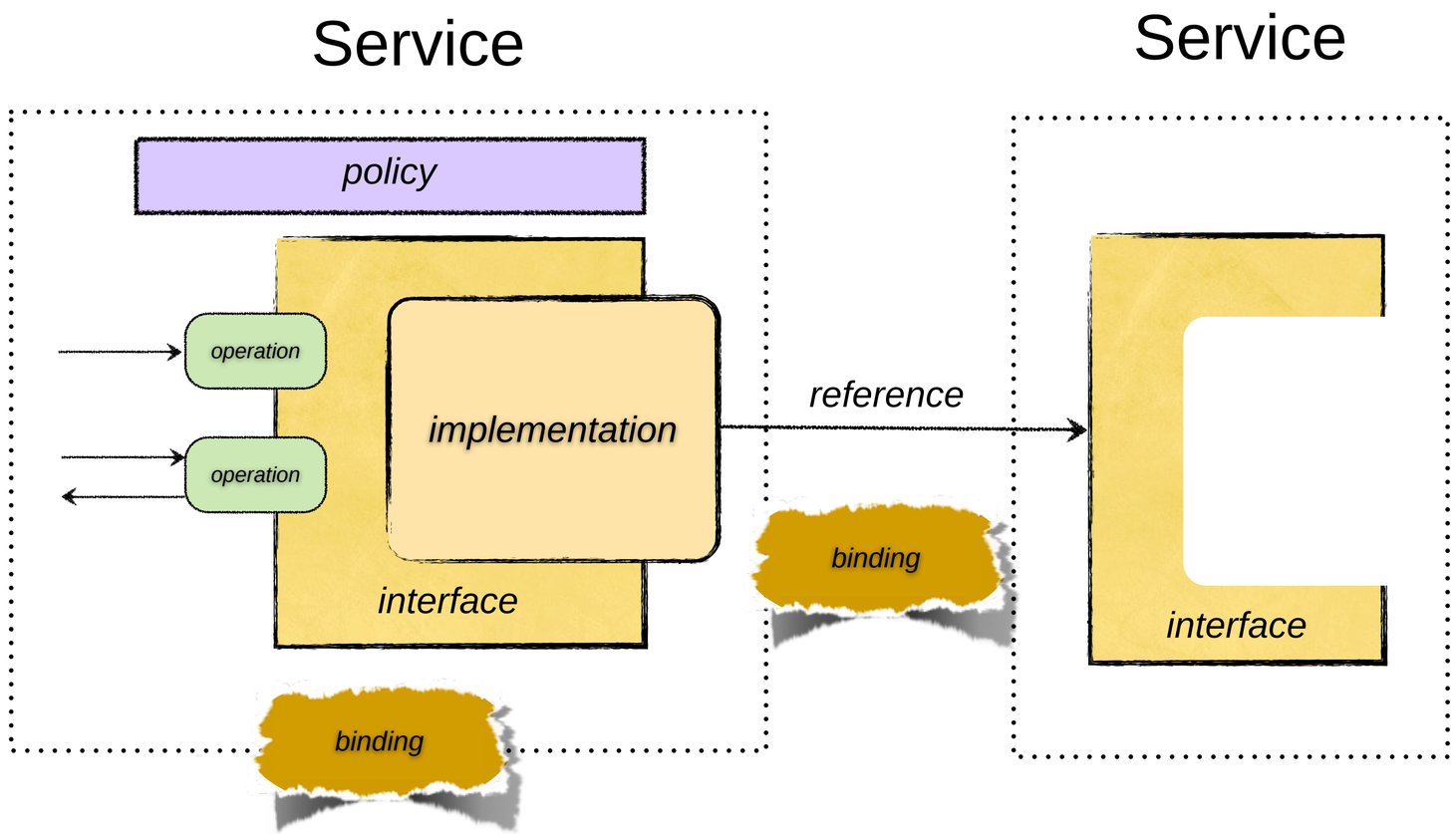


A Superficial Guide to Service-Oriented Principles



JBoss Community

A Superficial Guide to Service-Oriented Principles



JBoss Community


SHOW
ME THE
MONEY!

Service Interface

```
public interface OrderService {
    OrderAck submitOrder(Order order);
}
```

Service Implementation

```
public class OrderServiceBean implements OrderService {
    @Override
    public OrderAck submitOrder(Order order) {
        // Create an order ack
        return new OrderAck()
            .setOrderId(order.getOrderId())
            .setAccepted(true)
            .setStatus("Processing Order");
    }
```

Service Implementation

```
@Service(OrderService.class)
public class OrderServiceBean implements OrderService {
    @Override
    public OrderAck submitOrder(Order order) {
        // Create an order ack
        return new OrderAck()
            .setOrderId(order.getOrderId())
            .setAccepted(true)
            .setStatus("Processing Order");
    }
```

Service Reference

```
@Service(OrderService.class)
public class OrderServiceBean implements OrderService {

    @Inject @Reference
    private InventoryService _inventory;

    @Override
    public OrderAck submitOrder(Order order) {
        // Check the inventory
        Item orderItem = _inventory.lookupItem(order.getItemId());

        // Create an order ack
        ...
    }
}
```

Service Binding

```
$ soap-binding bind-service
    --serviceName OrderService
    --wsdl wsdl/FeedbackService.wsdl
```

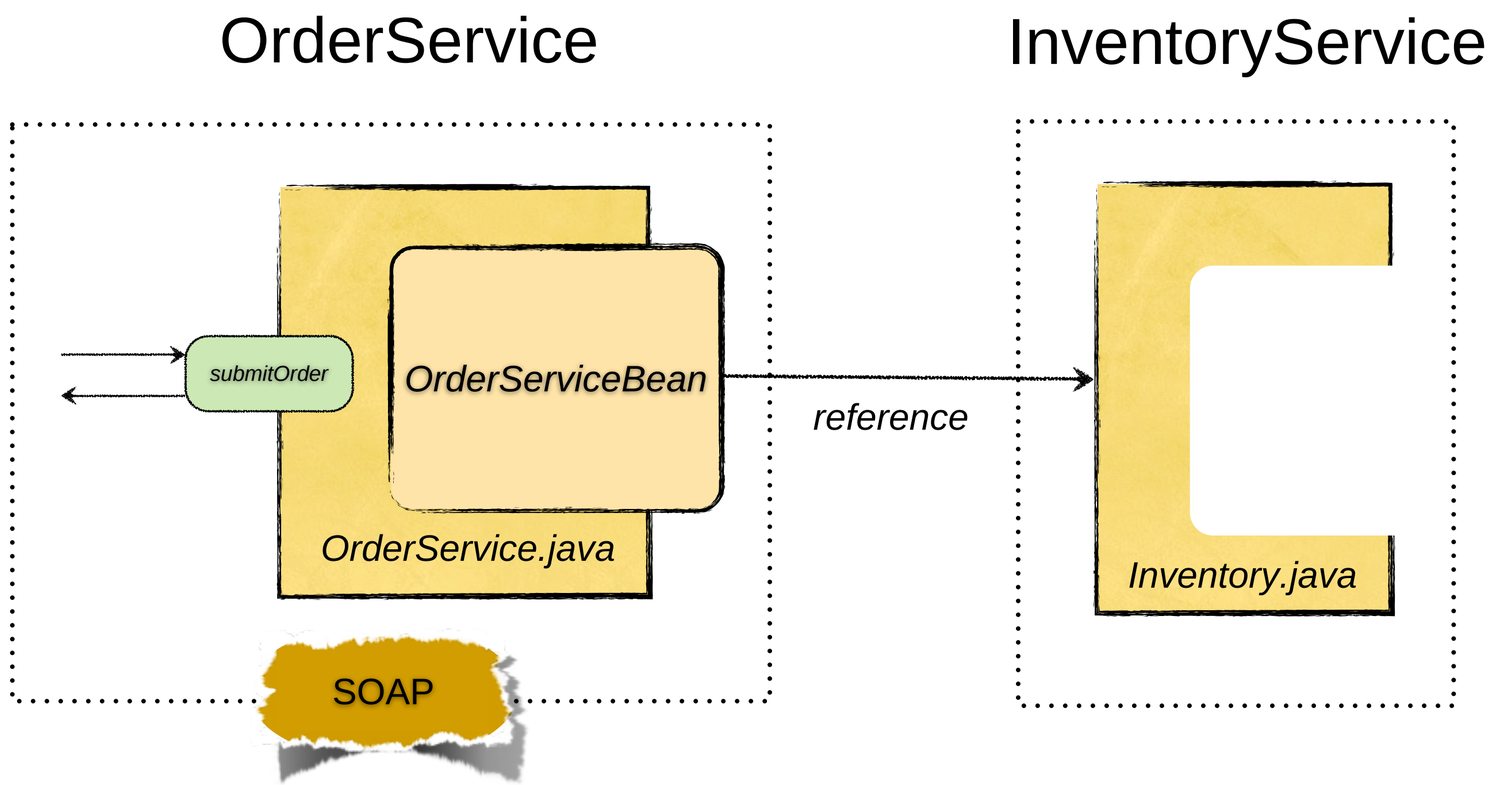
JBoss Community

Message Transformation

```
public class OrderAckTransform {

    @Transform(to = "{urn:examples:order:1.0}submitOrderResponse"}”)
    public Element transform(OrderAck orderAck) {
        // Transformation code goes here
    }
}
```

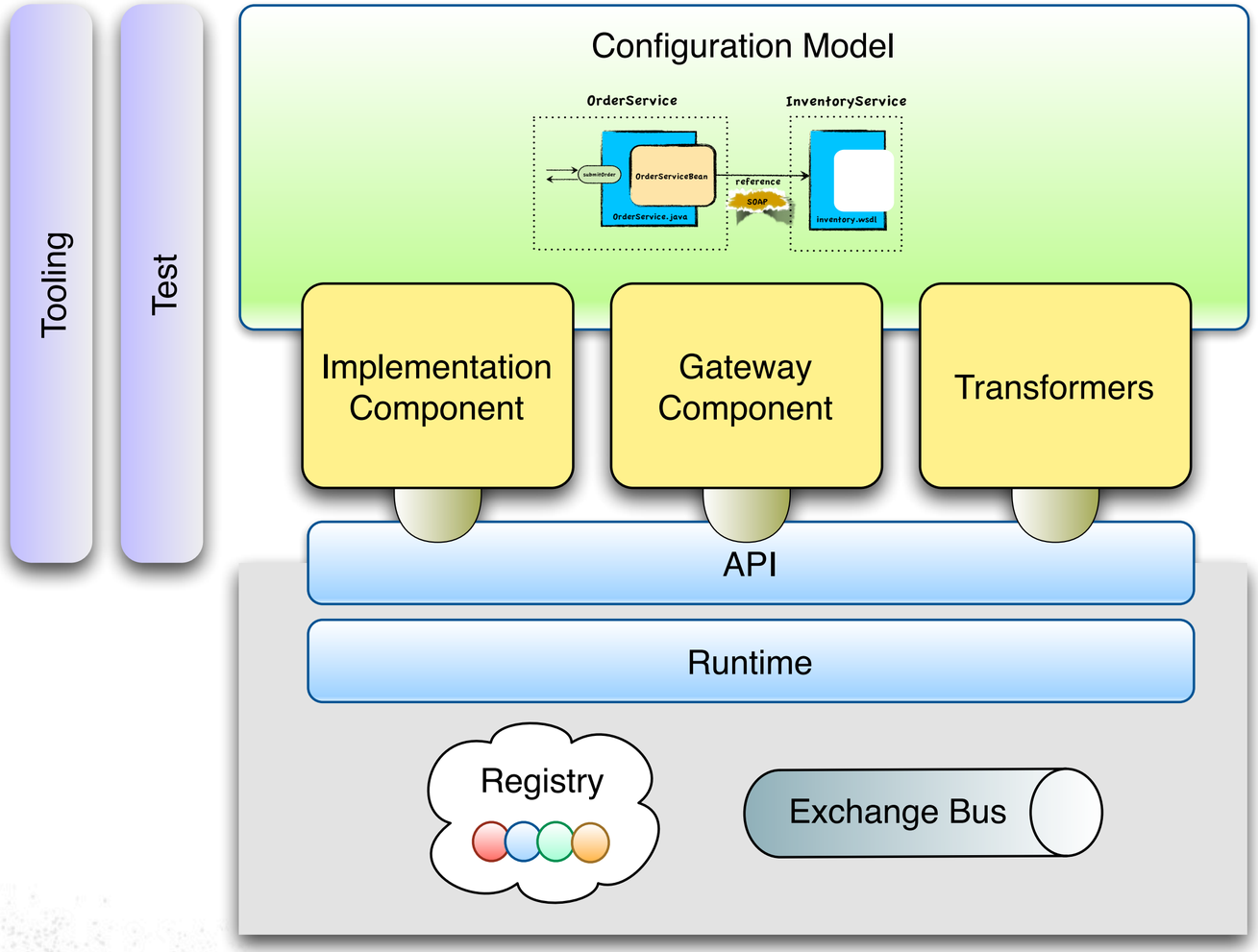
Voilà!


OrderService
InventoryService
submitOrder
OrderServiceBean
reference
OrderService.java
Inventory.java
SOAP

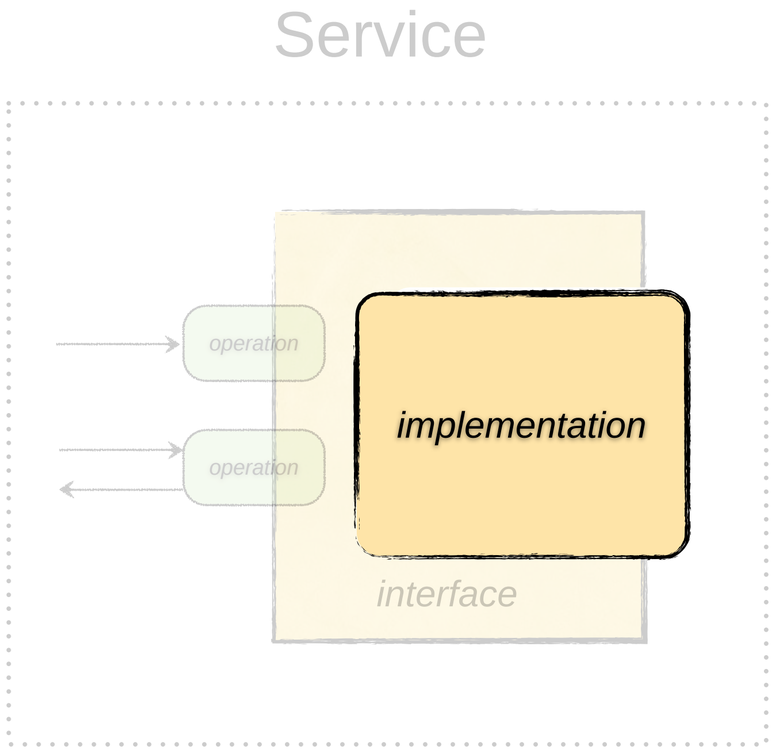
Application Description

```
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           name="orders"
           targetNamespace="urn:switchyard-quickstart:bean-service:0.1.0">
        <service name="OrderService" promote="OrderService">
            <binding.soap xmlns="urn:switchyard-component-soap:config:1.0">
                <wsdl>wsdl/OrderService.wsdl</wsdl>
                <serverPort>18001</serverPort>
            </binding.soap>
        </service>
        <component name="InventoryService">
            <implementation.bean xmlns="urn:switchyard-component-bean:config:1.0"
                                 class="org.switchyard.quickstarts.bean.service.InventoryServiceBean"/>
            <service name="InventoryService">
                <interface.java interface="org.switchyard.quickstarts.bean.service.InventoryService"/>
            </service>
        </component>
        <component name="OrderService">
            <implementation.bean xmlns="urn:switchyard-component-bean:config:1.0"
                                 class="org.switchyard.quickstarts.bean.service.OrderServiceBean"/>
            <service name="OrderService">
                <interface.java interface="org.switchyard.quickstarts.bean.service.OrderService"/>
            </service>
            <reference name="InventoryService">
                <interface.java interface="org.switchyard.quickstarts.bean.service.InventoryService"/>
            </reference>
        </component>
    </composite>
```

What's In SwitchYard



Implementation Components



Bean Component

- POJO = Service ... ‘nuff said
- Easy to use
  - Annotation-based
  - Config auto-generated
  - Service auto-registered
- Consistent with core principles
  - Services declare a service interface
  - References injected based on service interface
  - Dependencies are explicit

Bean Service

```
public class MyServiceBean implements MyService {

    public void processApplication(Application app) {
        // process the application
    }
}
```

Bean Service

```
@Service(MyService.class)
public class MyServiceBean implements MyService {

    public void processApplication(Application app) {
        // process the application
    }
}
```

Camel Component

- Integrates Apache Camel with SwitchYard
- Camel provides
  - Routing engine and language(s)
  - Loads of EIP
  - Cornucopia of components
- Camel as a service
  - Routes provide pipeline orchestration
  - Service interface
  - Service references resolved independent of binding

Camel Service

```
public class OrderServiceBuilder extends RouteBuilder {

    public void configure() {
        from("file://orders/input")
            .log("Order Received : ${body}")
            .to("bean:prioritize")
            .filter().xpath("/order[@priority='high']""))
            .to("file://shipping/input");
    }
}
```

Camel Service

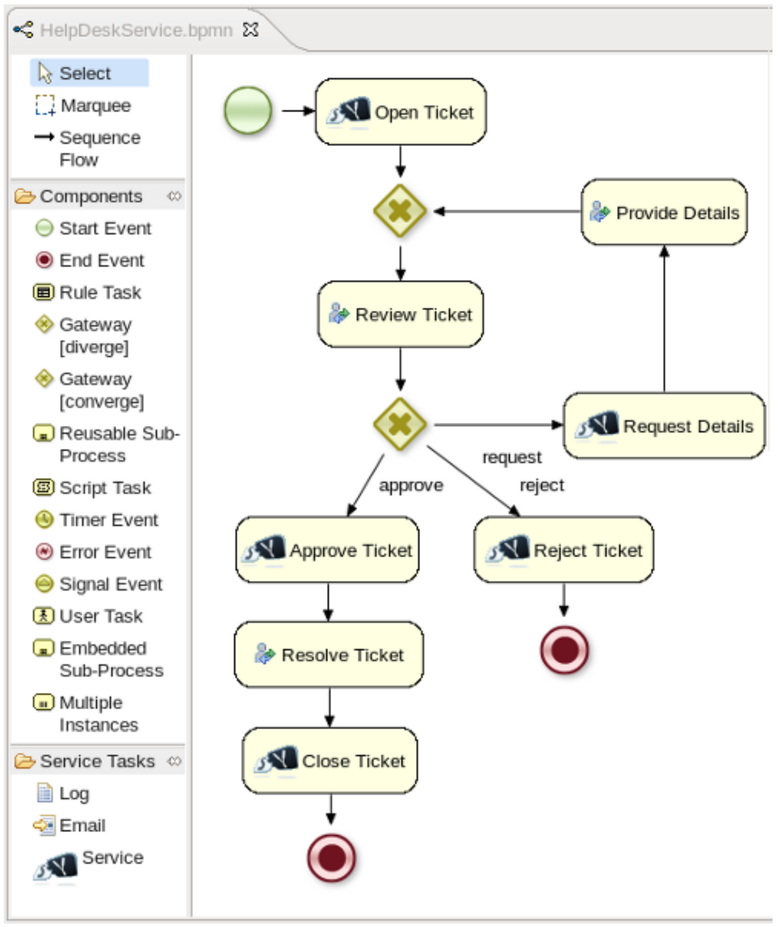
```
@Route(OrderService.class)
public class OrderServiceBuilder extends RouteBuilder {

    public void configure() {
        from("switchyard://OrderService")
            .log("Order Received : ${body}")
            .to("bean:prioritize")
            .filter().xpath("/order[@priority='high']""))
            .to("switchyard://ShippingService");
    }
}
```

BPM Component

- Provides workflow execution support in SwitchYard
- Based on jBPM 5
- BPMN 2.0
- Service Orchestration
- Human Workflow

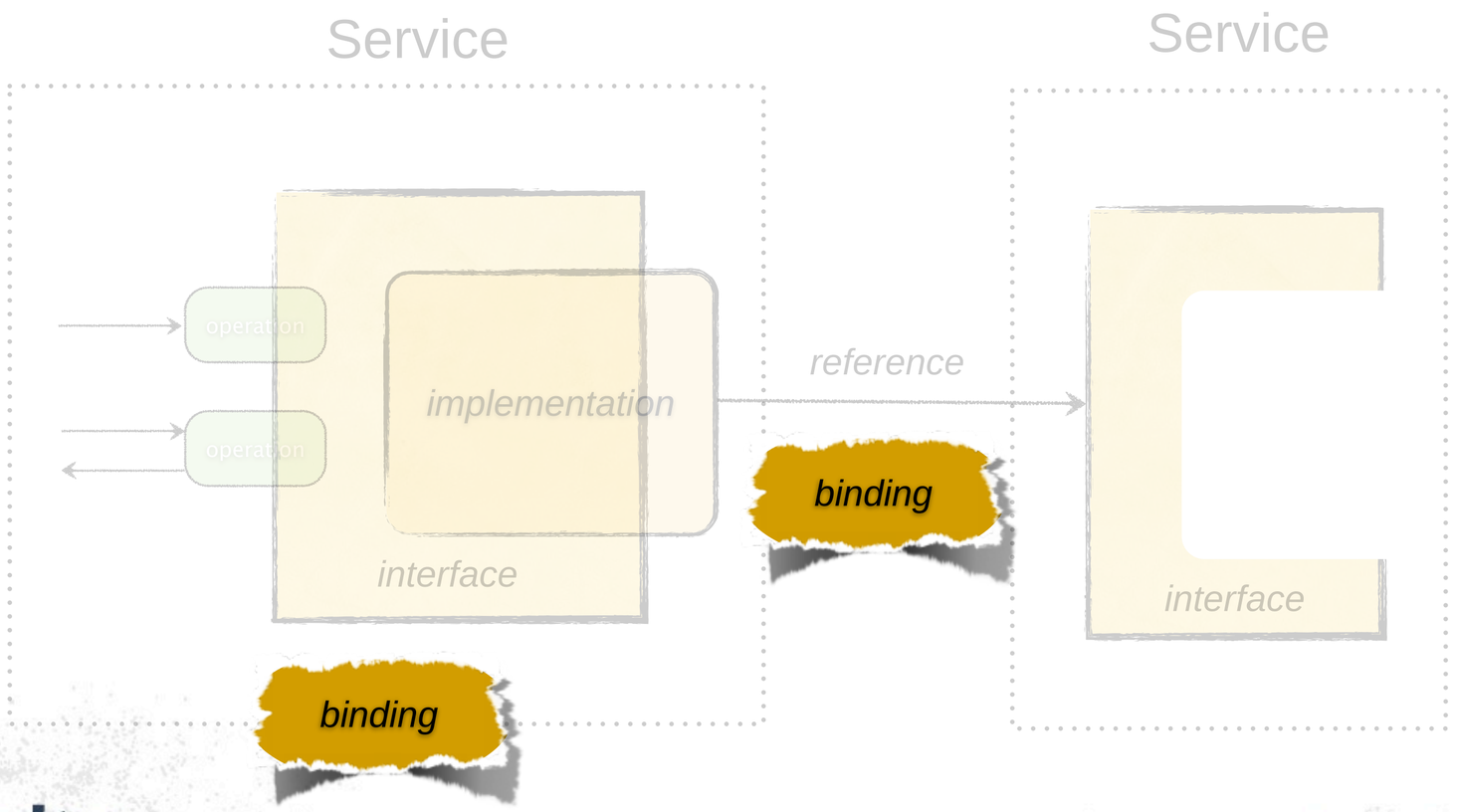
BPM Service



JBoss Community

Gateway Components

- Provide protocol binding for services and references
- Gateway components are bi-directional



Gateway Components

- Two fundamental rules of gateways
  - Never have enough
  - The ones you have don’t do enough
- Our approach
  - Focus on key gateways for platform
    - SOAP, HornetQ, etc.
  - Incorporate adapters from other communities
    - Camel components
  - Straightforward pluggability for rolling your own
    - Tooling, configuration, deployment

Transformation

- Ubiquitous challenge in application integration and SOA
- Three flavors
  - Change in data representation
  - Change in data format
  - Change in data itself
- Typically addressed in a procedural manner

Conversion

- Change in representation
- Representation = Java type
- Transformation is simply a type conversion
- No semantic knowledge required

java.lang.String

```
<order>
<item>XYZ123</item>
<quantity>5</quantity>
</order>
```

=

org.w3c.dom.Node

```
<order>
<item>XYZ123</item>
<quantity>5</quantity>
</order>
```

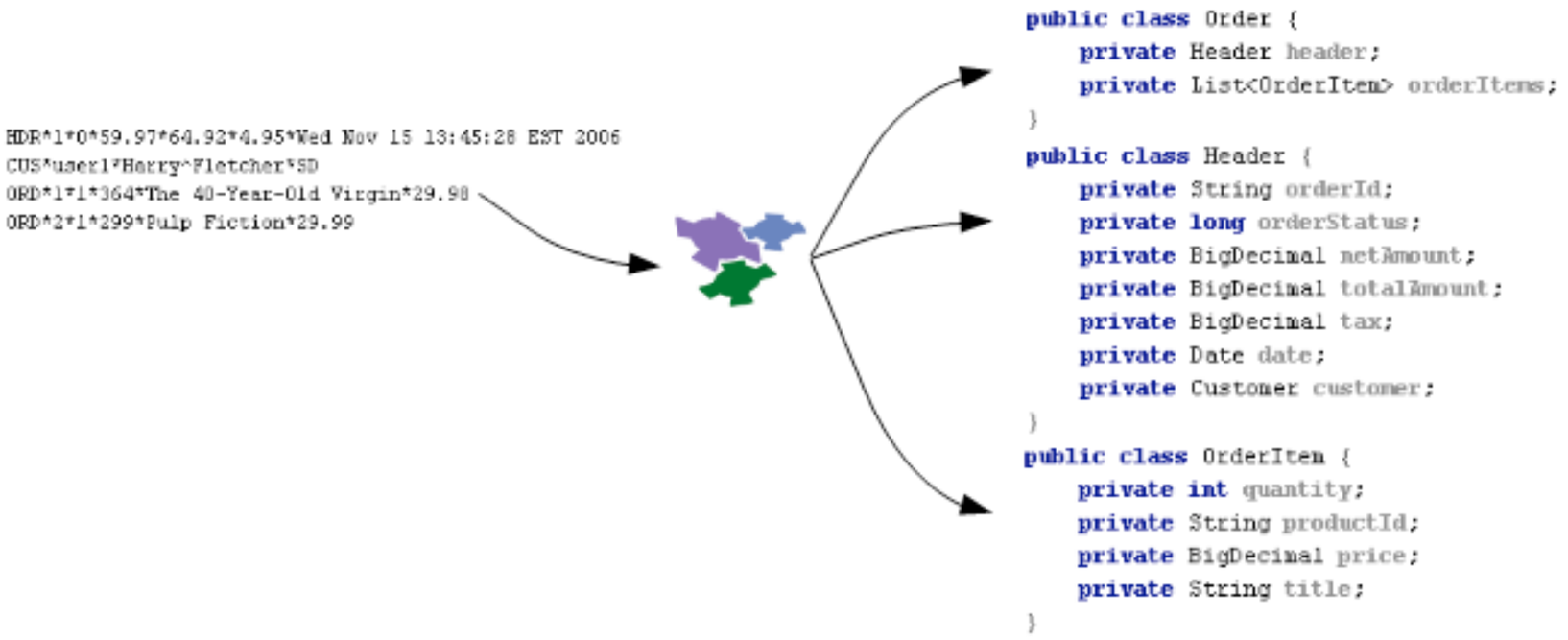
=

java.io.InputStream

```
<order>
<item>XYZ123</item>
<quantity>5</quantity>
</order>
```

Translation

- Requires semantic understanding of data types
- Machines cannot do this on their own ...

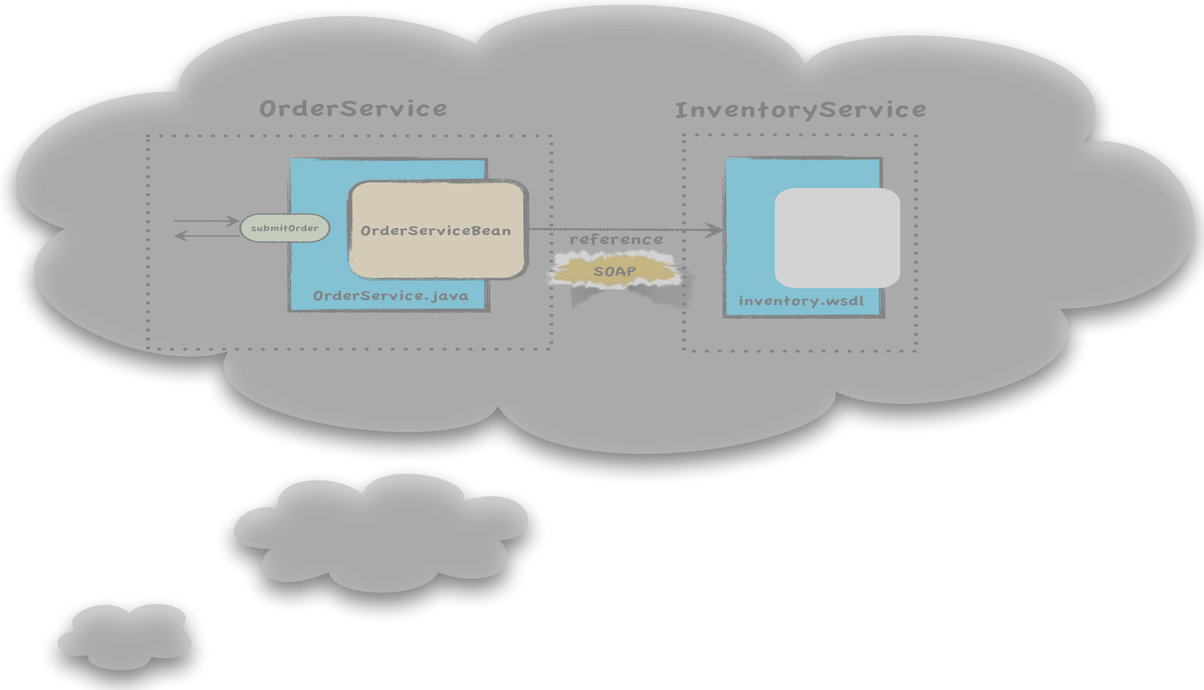


Transformers

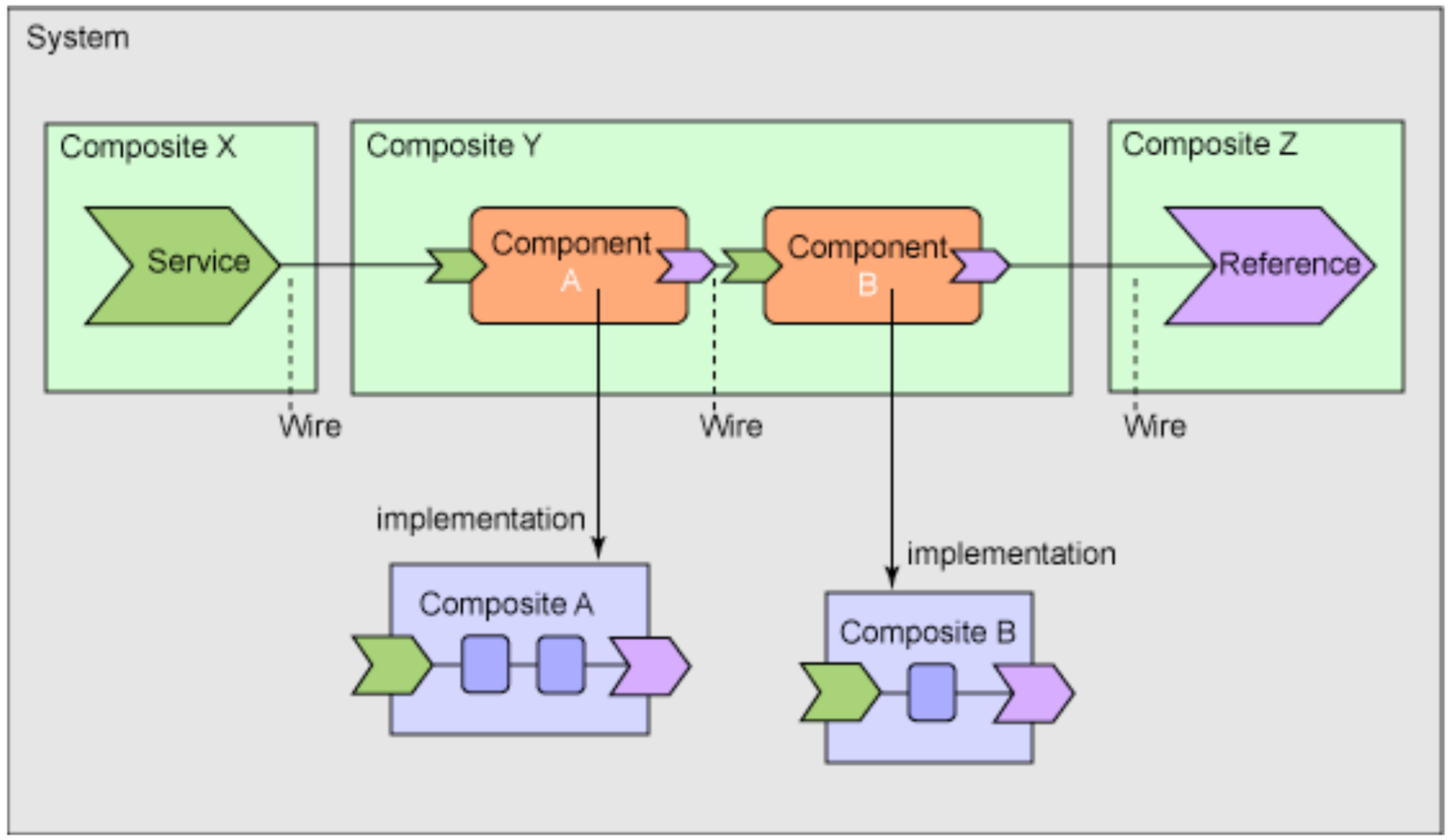
- Transformation is wired into SwitchYard core
  - Types declared via service contract
  - Transformer resolved dynamically at runtime
- Declarative, not procedural
- Bring on the canonical data models
- Current Transformers
  - Java, Smooks, XSLT, JAXB, JSON

Configuration

- We need a way to represent this ...



Configuration



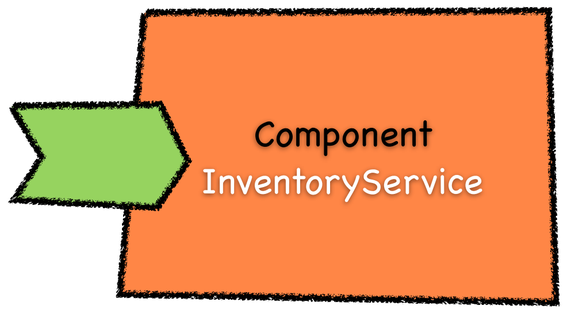
SCA - Service Component Architecture

- Set of specifications for building applications in a manner consistent with SOA principles
- Assembly spec is none too shabby
  - Service definition language
  - Encapsulation model
- Better than defining our own configuration format?
  - Skills portability is nice
  - Runtime portability much less certain

Application Description

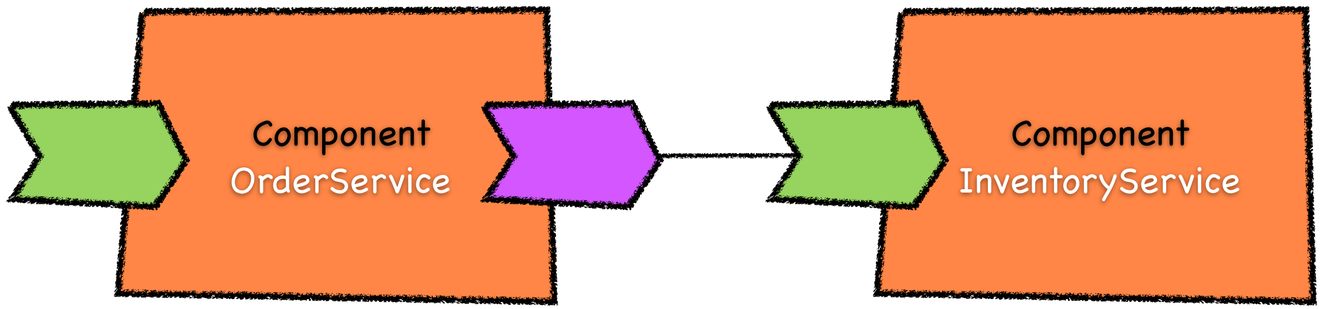
```
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           name="orders"
           targetNamespace="urn:switchyard-quickstart:bean-service:0.1.0">
        <service name="OrderService" promote="OrderService">
            <binding.soap xmlns="urn:switchyard-component-soap:config:1.0">
                <wsdl>wsdl/OrderService.wsdl</wsdl>
                <serverPort>18001</serverPort>
            </binding.soap>
        </service>
        <component name="InventoryService">
            <implementation.bean xmlns="urn:switchyard-component-bean:config:1.0"
                                 class="org.switchyard.quickstarts.bean.service.InventoryServiceBean"/>
            <service name="InventoryService">
                <interface.java interface="org.switchyard.quickstarts.bean.service.InventoryService"/>
            </service>
        </component>
        <component name="OrderService">
            <implementation.bean xmlns="urn:switchyard-component-bean:config:1.0"
                                 class="org.switchyard.quickstarts.bean.service.OrderServiceBean"/>
            <service name="OrderService">
                <interface.java interface="org.switchyard.quickstarts.bean.service.OrderService"/>
            </service>
            <reference name="InventoryService">
                <interface.java interface="org.switchyard.quickstarts.bean.service.InventoryService"/>
            </reference>
        </component>
    </composite>
```

Service Component



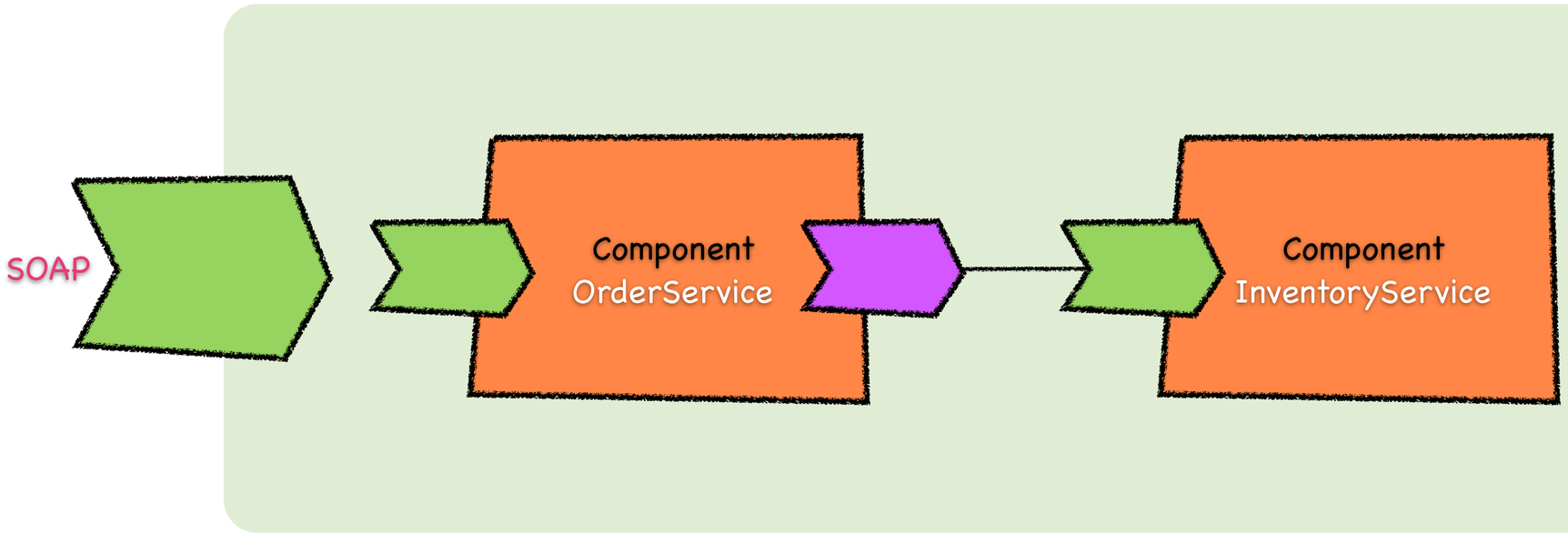

```
<component name="InventoryService">
   <bean:implementation.bean
      class="org.switchyard.quickstarts.bean.service.InventoryServiceBean"/>
   <service name="InventoryService">
      <interface.java
         interface="org.switchyard.quickstarts.bean.service.InventoryService"/>
   </service>
</component>
```

Service Reference




```
<component name="OrderService">
   <bean:implementation.bean
      class="org.switchyard.quickstarts.bean.service.OrderServiceBean"/>
   <service name="OrderService">
      <interface.java interface="org.switchyard.quickstarts.bean.service.OrderService"/>
   </service>
   <reference name="InventoryService">
      <interface.java interface="org.switchyard.quickstarts.bean.service.InventoryService"/>
   </reference>
</component>
```

Service Reference




```
<composite xmlns="http://docs.oasis-open.org/ns/opencsa/sca/200912"
           name="orders"
           targetNamespace="urn:switchyard-quickstart:bean-service:0.1.0">
   <service name="OrderService" promote="OrderService">
      <soap:binding.soap>
         <wsdl>wsdl/OrderService.wsdl</wsdl>
         <serverPort>18001</serverPort>
      </soap:binding.soap>
   </service>
</composite>
```

Runtime Options

- JBoss AS6
- JBoss AS7
- WAR Deployment
- Embedded Unit Test
- OSGi coming soon ....

```
20:59:45,785 INFO  [org.jboss.as] (Controller Boot Thread) JBoss AS 7.0.0.Final "Lightning" started in 2079ms -
Started 94 of 149 services (55 services are passive or on-demand)
```

!!!!!!

```
started in 2079ms
```

Testing

- Big Bang testing of SOA applications must stop!
- Develop and test your services iteratively
  - Service, transformation, binding, etc.
- SwitchYardRunner
  - Bootstraps runtime, components, and application
- MixIns
  - Enriches test case via composition vs. extension
  - CDI, HTTP, Smooks, BPM, HornetQ
- Arquillian

Tooling

- Focus on intuitive user experience with quick onramp
- Seam Forge
  - Rapid application development tool
  - Ease of a wizard, power of a shell
  - More coming in the demo ...
- Admin Console
- IDEs
  - Maven support provides baseline functionality across IDEs
  - Specific tooling features for Eclipse

Questions?

JBoss Community

JBoss Community