JBoss R&D – April 8th, 2008

Smooks

(http://milyn.codehaus.org/smooks)
(LGPL)
(v1.0)

(Tom Fennelly - tom.fennelly@jboss.com)

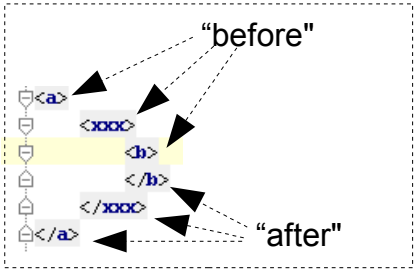
Agenda

- What is Smooks?
  - Typical Use-cases
- Feature Drill Down
- Smooks in JBossESB
- Smooks in a Web Environment
- Smooks and other JBoss Projects/Products?
- Future Directions
- Q & A

What is Smooks?

High Level Description

- "*Structured Data Event Stream Processor*"
  - Source --> Event Stream --> Result
    - Source
      - XML, EDI, CSV, Java, Etc
    - Event Stream
      - SAX, DOM
      - "Element Visitors" Processing Events to the Result (Event Handlers)
    - Result
      - XML, EDI, CSV, Java, Etc



Typical Use-cases

- What does the Smooks “Event Stream Processor” Enable?
  - Transformation
    - Fragment based Transforms
      - XSLT, FreeMarker, StringTemplate, Groovy, Java
    - Multiple Sources and Results Supported
      - XML, EDI, CSV, Java etc...
    - Java to Java Supported
  - Java Binding
    - Source (XML, EDI, Java etc) model and Java model don't need to “line up”
    - Supports Bindings to 1+ Object Models concurrently
    - Supports “Virtual Object Models” i.e. Maps of Maps of Maps etc.
      - Not obliged to define a Physical Object model
      - Values are Typed – not just Strings

continued....

...continued

- ...continued
  - Message Splitting (1 to *n* Messages)
    - Huge Message (Gbs)
    - Complex Splits
      - Not course grained “Single Fragment” Splits
      - Merging Data from multiple sub-hierarchies
      - Uses Java Binding functionality
    - Multiple Concurrent Splits
  - Message Fragment Routing
    - Splitting + Templating = “Split Message”
    - Route “Split Messages” to File, JMS, Database, *ESB Service* etc
    - Multiple Concurrent Routings
  - Message Fragment Enrichment
    - From Database DataSource or other

Feature Drill Down

Smooks Basics

- Source --> Event Stream --> Result
  - Source
    - javax.xml.transform.Source
      - StreamSource / DOMSource: *XML, EDI, CSV etc*
      - JavaSource: *Java Object Model*
  - Event Stream
    - Processed via org.milyn.delivery.sax.SAXElementVisitor impls
      - Defined in the Smooks Configuration
  - Result
    - javax.xml.transform.Result
      - StreamResult / DOMResult: *XML, EDI, CSV etc*
      - JavaResult: *Java Object Model*

examples....

Executing Smooks - Examples

Smooks Configuration

```
<?xml version="1.0"?>
<smooks-resource-list xmlns="http://www.milyn.org/xsd/smooks-1.0.xsd">

    <!--
    Configure the EDI Parser to parse the message stream into a stream of SAX events.
    -->
    <resource-config selector="org.xml.sax.driver">
        <resource>org.milyn.smooks.edi.SmooksEDIParser</resource>
        <param name="mapping-model">/example/edi-to-xml-order-mapping.xml</param>
    </resource-config>

</smooks-resource-list>
```

Smooks Execution

```
public void ediToXML(Reader ediSource, Writer xmlResult) throws IOException, SAXException {
    Smooks smooks = new Smooks("/configs/edi-to-xml.xml");
    Source source = new StreamSource(ediSource);
    Result result = new StreamResult(xmlResult);

    smooks.filter(source, result);
}
```

Input Message

```
HDR*1*0*59.97*64.92*4.95*Wed Nov 15 13:45:28 EST 2006
CUS*user1*Harry^Fletcher*SD
ORD*1*1*364*The 40-Year-Old Virgin*29.98
ORD*2*1*299*Pulp Fiction*29.99
```

Output Message

```
<Order>
        <header>
                <order-id>1</order-id>
                <status-code>0</status-code>
                <net-amount>59.97</net-amount>
                <total-amount>64.92</total-amount>
                <tax>4.95</tax>
                <date>Wed Nov 15 13:45:28 EST 2006</date>
        </header>
        <customer-details>
                <username>user1</username>
                <name>
                        <firstname>Harry</firstname>
                        <lastname>Fletcher</lastname>
                </name>
                <state>SD</state>
        </customer-details>
        <order-item>
                <position>1</position>
                <quantity>1</quantity>
                <product-id>364</product-id>
                <title>The 40-Year-Old Virgin</title>
                <price>29.98</price>
        </order-item>
        <order-item>
                <position>2</position>
                <quantity>1</quantity>
                <product-id>299</product-id>
                <title>Pulp Fiction</title>
                <price>29.99</price>
        </order-item>
</Order>
```

continued....

... continued

- XML to XML

```
public void xmlToXML(Reader xmlSource, Writer xmlResult) throws IOException, SAXException {
    Smooks smooks = new Smooks("/configs/xml-to-xml.xml");
    Source source = new StreamSource(xmlSource);
    Result result = new StreamResult(xmlResult);

    smooks.filter(source, result);
}
```

Configure Smooks Instance (Cache it!)

StreamSource and StreamResult

Filter Source to Result using **Smooks** Instance

- EDI to Java

```
public Order ediToOrder(Reader ediSource) throws IOException, SAXException {
    Smooks smooks = new Smooks("/configs/edi-to-java.xml");
    Source source = new StreamSource(ediSource);
    JavaResult result = new JavaResult();

    smooks.filter(source, result);

    return (Order) result.getBean("order");
}
```

StreamSource and **JavaResult**

Get the Populated **Order** bean from the JavaResult

- Java to Java

```
public SalesforceOrder orderToOrder(SAPOrder sapOrder) throws IOException, SAXException {
    Smooks smooks = new Smooks("/configs/sap-to-sf.xml");
    Source source = new JavaSource(sapOrder);
    JavaResult result = new JavaResult();

    smooks.filter(source, result);

    return (SalesforceOrder) result.getBean("order");
}
```

JavaSource and **JavaResult**

Get the Populated **Salesforce Order** bean from the JavaResult

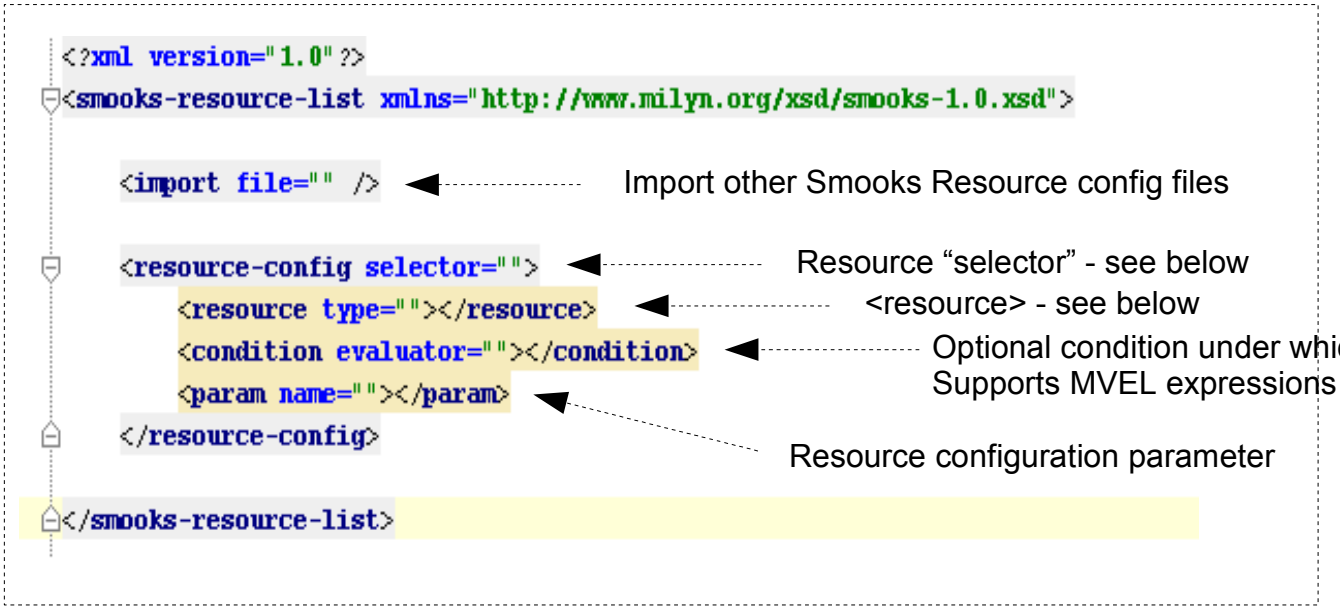
Configuring Smooks

- Everything is configured as a “Resource”
  - “Event Stream Filtering Resources”
    - Element Visitor Logic (SAXElementVisitor Impls)
      - Fragment Transforms
      - Java bean Populators
        - DataDecoders – DateDecoder, EnumDecoder, MappingDecoder etc
    - Routing Resources
      - Output Stream Resources
      - Database DataSource
    - Message Enrichment Resources
      - Database DataSource
      - Static Context Properties
    - Filtering Parameters
    - Anything!!

config format....

... Configuring Smooks

- ... config format




```
<?xml version="1.0"?>
<smooks-resource-list xmlns="http://www.milyn.org/xsd/smooks-1.0.xsd">

    <import file="" />

    <resource-config selector="">
        <resource type=""></resource>
        <condition evaluator=""></condition>
        <param name=""></param>
    </resource-config>

</smooks-resource-list>
```

  - *“selector”*: Meaning depends on the <resource> type. For a SAX Event Stream, it's usually the name of the XML element at which the Resource is “targeted” e.g. for a FreeMarker template <resource> for transforming that fragment
  - *<resource>*: The resource associated with the selector e.g. SAXElementVisitor impl, XSL Template, FreeMarker Template, StringTemplate Template, Groovy script etc
    - “type” attribute usually only required when the resource is defined “inline”.

SAXElementVisitor

- The Basic "***Building Block***"
- Stateless:
  - State can be maintained in the **ExecutionContext**
  - One **ExecutionContext** created per *Smooks.filter(Source, Result)* operation
- SAXElementVisitor = SAXVisitBefore + SAXVisitChildren + SAXVisitAfter
- Simple Example:

Reflective Config Injection

Visitor Impl

```
public class EchoMessageVisitor implements SAXVisitBefore, SAXVisitAfter {

    @ConfigParam
    private String message;

    public void visitBefore(SAXElement element, ExecutionContext executionContext) throws SmooksException, IOException {
        System.out.println("visitBefore message: " + message);
    }

    public void visitAfter(SAXElement element, ExecutionContext executionContext) throws SmooksException, IOException {
        System.out.println("visitAfter message: " + message);
    }
}
```

Smooks Config

```
<?xml version="1.0" ?>
<smooks-resource-list xmlns="http://www.milyn.org/xsd/smooks-1.0.xsd">

    <resource-config selector="xxx">
        <resource>example.EchoMessageVisitor</resource>
        <param name="message">Hi there!!</param>
    </resource-config>

</smooks-resource-list>
```

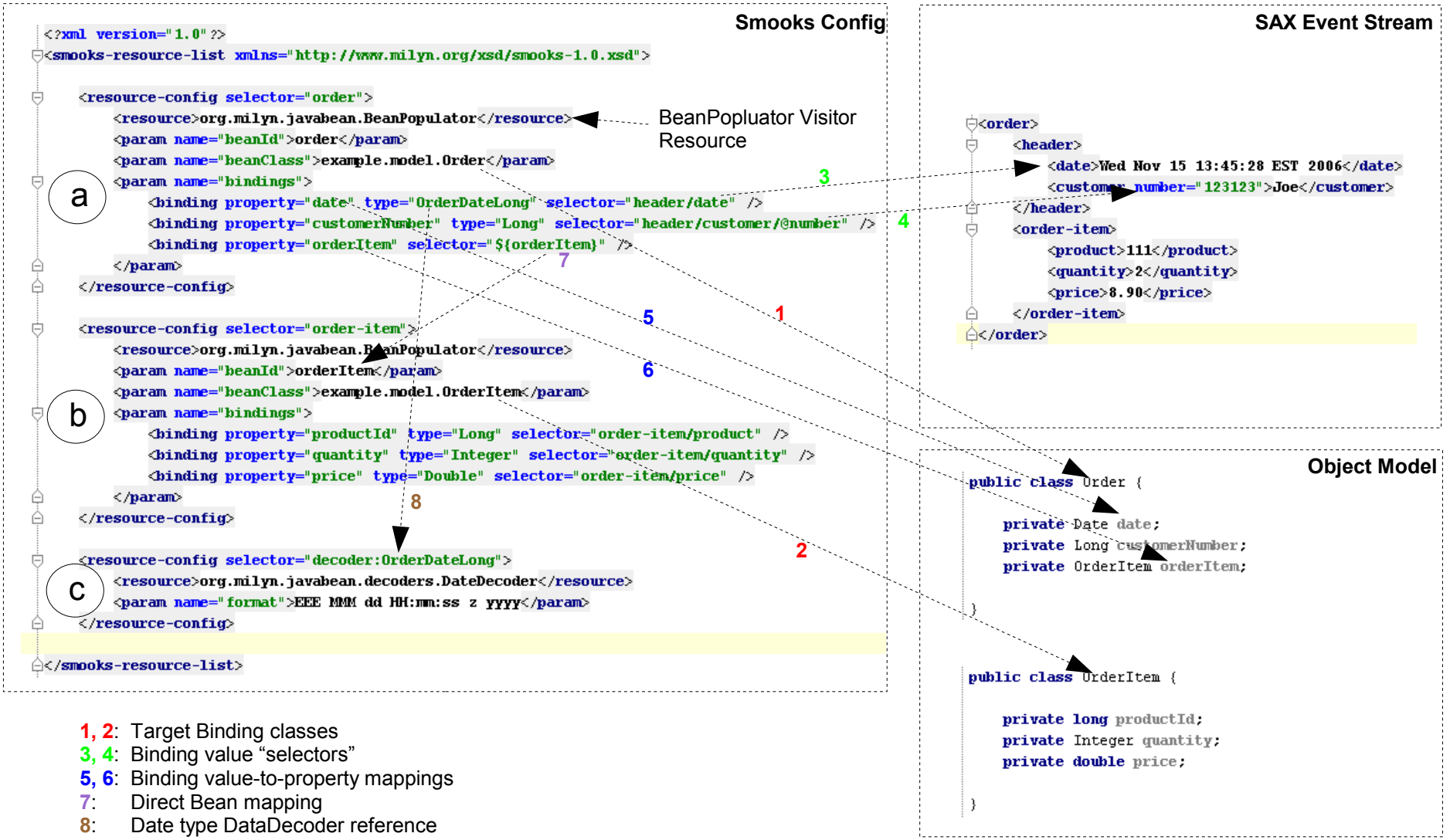
SAX Event Stream

```
<a>
    <xxx>
        <b>
        </b>
    </xxx>
</a>
```

Smooks Cartridges

- Bundling “Visitor” Impls by Functionality
  - Javabean
    - Java binding
  - Templating
    - Fragment based Templating Transforms
    - XSLT, FreeMarker, StringTemplate
  - Routing
    - Routing Message Fragments (Splitting)
    - File, JMS, Database
  - Stream Readers
    - Event Stream Generators (SAX)
    - CSV, EDI
  - Miscellaneous/Experimental
    - Servlet, CSS

Javabean Cartridge



1, 2: Target Binding classes
3, 4: Binding value "selectors"
5, 6: Binding value-to-property mappings
7: Direct Bean mapping
8: Date type DataDecoder reference

continued....

... continued

```
public Order xmlToOrder(Reader xmlSource) throws IOException, SAXException {
    Smooks smooks = new Smooks("/configs/xml-to-java.xml");
    Source source = new StreamSource(xmlSource);
    JavaResult result = new JavaResult();

    smooks.filter(source, result);

    return (Order) result.getBean("order");
}
```

StreamSource and **JavaResult**

Get the Populated **Order** bean from the JavaResult

continued....

... continued

- Javabean Cartridge used by other Cartridges
  - Templating
    - Populated models available to XSLT, FreeMarker, StringTemplate
      - “Model Driven Transforms”
  - Splitting & Routing:
    1. Generate “Split Messages” by populating an Object Model
       - Split Messages can contain data from multiple sub-hierarchies within the message i.e. not course grained Splitting
    2. Apply a Template (e.g. FreeMarker) to Object Model
    3. Route result
    4. Repeat 1 to 3 for each Message fragment (e.g. each <order-item>)
  - Persistence
    - “Routing” populated Object Model to a DataSource

Templating Cartridge

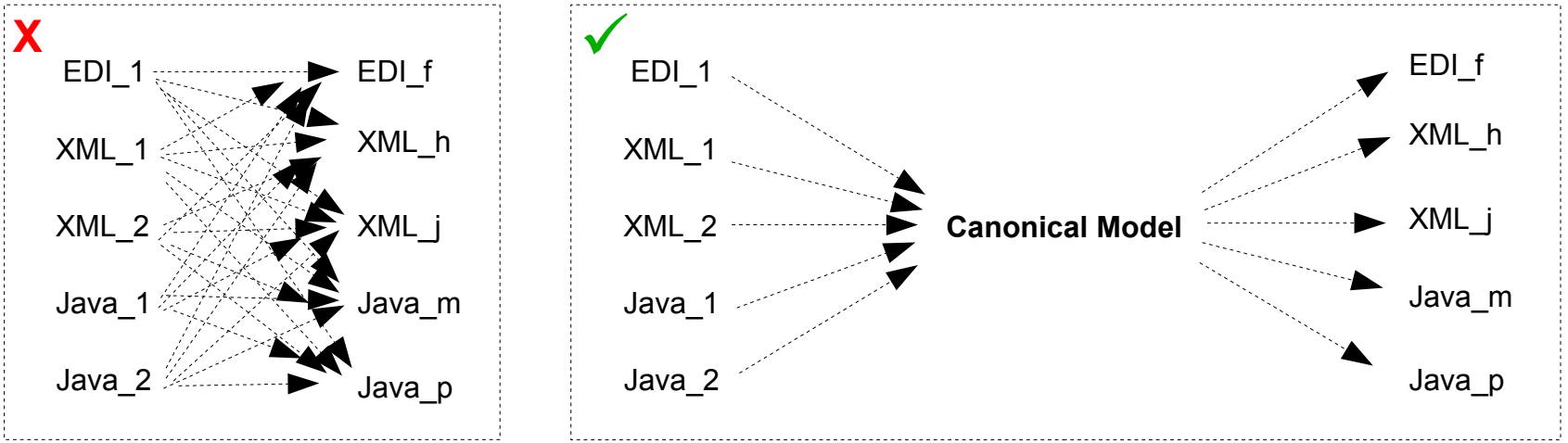
- "Model Driven Transforms"
  - Apply a populated object model to Template
    - Object model populated by **Javabean Cartridge**
    - Object model can be "Virtual" i.e. Maps of Maps etc
  - Apply Templating Result to selected message Fragment
    - replace, addto, insertbefore etc
  - Support for FreeMarker, XSLT, StringTemplate
    - Others possible – Velocity, MVEL etc
  - FreeMarker Example:

```
<resource-config selector="c">
    <resource>/org/milyn/templating/freemarker/test-template.ftl</resource>
    <param name="action">insertbefore</param>
</resource-config>
```

continued....

… continued

- Object Model as a “Canonical Form”
  - Using a populated Object Model to represent a message's “**Canonical Form**”
  - *n* inputs -> Canonical Form -> *n* outputs
    - Avoid 1:1 mappings/transforms



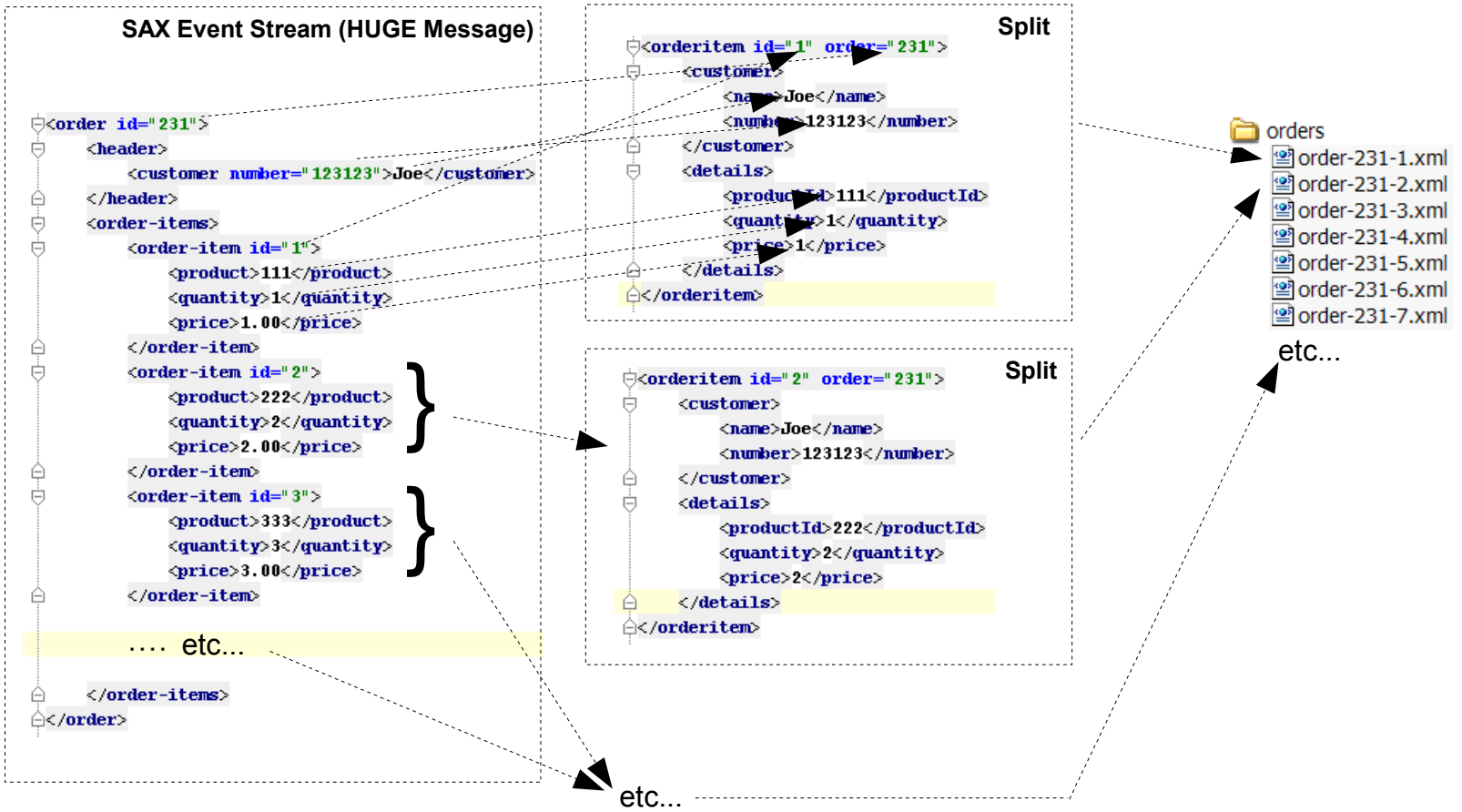
Routing Cartridge

- Splitting & Routing
  - Using Smooks to process **HUGE** messages (GBs)
    - Using SAX Event Stream Filter
  - Javabean Cartridge (Again!)
    - Repeatedly Populate fragment model(s)
      - Repeatedly Route populated models (e.g. Java/JMS), and/or
      - Repeatedly Apply template(s) to model(s) and Route XML, EDI, etc etc
  - Destination Types Supported:
    - *OutputStreamResource*
      - File
    - JMS Destination
      - Queue, Topic
    - Database DataSource
    - JBossESB
      - Transport Agnostic “ESBRouter” in JBossESB 4.3

continued....

...Routing to File

- Example – **HUGE** “Order” message **split** to many “OrderItem” messages, **routed** to file...



continued....

...Routing to File... config...

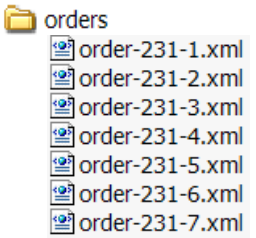
Smooks Configuration

```
<?xml version="1.0"?>
<smooks-resource-list xmlns="http://www.milyn.org/xsd/smooks-1.0.xsd">

    <resource-config selector="order">
        <resource>org.milyn.javabean.BeanPpopulator</resource>
        <param name="beanId">order</param>
        <param name="beanClass">java.util.Hashtable</param>
        <param name="bindings">
            <binding property="orderId" type="Integer" selector="order/@id" />
            <binding property="customerNumber" type="Long" selector="header/customer/@number" />
            <binding property="customerName" selector="header/customer" />
            <binding property="orderItem" selector="${orderItem}" />
        </param>
    </resource-config>

    <resource-config selector="order-item">
        <resource>org.milyn.javabean.BeanPopulator</resource>
        <param name="beanId">orderItem</param>
        <param name="beanClass">java.util.Hashtable</param>
        <param name="bindings">
            <binding property="itemId" type="Integer" selector="order-item/@id" />
            <binding property="productId" type="Long" selector="order-item/product" />
            <binding property="quantity" type="Integer" selector="order-item/quantity" />
            <binding property="price" type="Double" selector="order-item/price" />
        </param>
    </resource-config>

    <resource-config selector="order-item">
        <resource>org.milyn.routing.file.FileOutputStreamResource</resource>
        <param name="resourceName">orderItemSplitStream</param>
        <param name="fileNamePattern">order-${order.orderId}-${order.orderItem.itemId}.xml</param>
        <param name="listFileNamePattern">order-${order.orderId}.lst</param>
        <param name="destinationDirectory">orders</param>
        <param name="highWaterMark">10</param>
    </resource-config>

    <resource-config selector="order-item">
        <resource>/org/milyn/routing/file/orderitem-split.ftl</resource>
        <param name="outputStreamResource">orderItemSplitStream</param>
    </resource-config>

</smooks-resource-list>
```

1
2
3
4

Throttle split file generation to "destinationDirectory"

1, 2: Model Binding Configurations (Virtual Model)
3: File OutputStream Resource Configuration
4: FreeMarker Template Resource Configuration

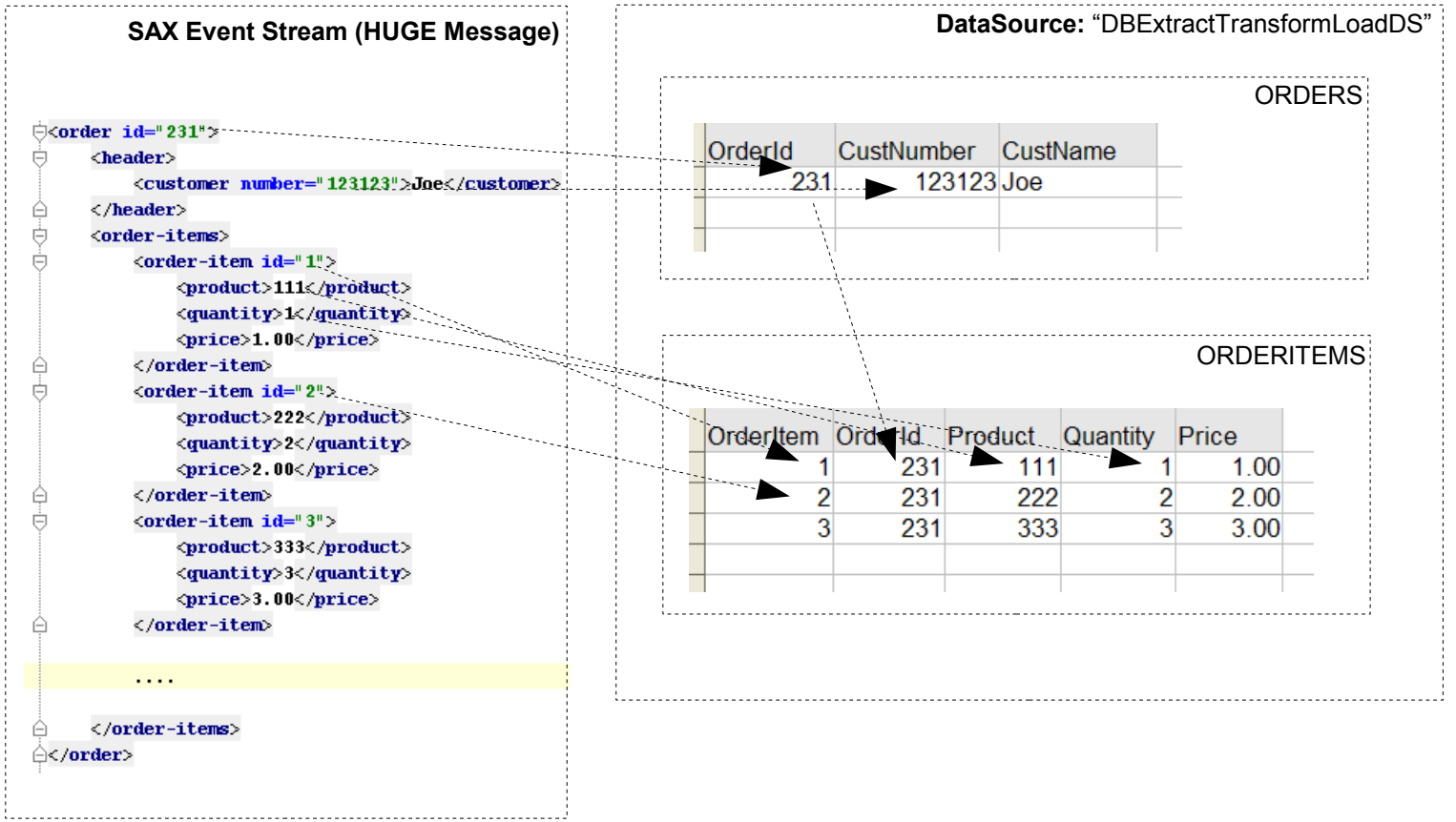


etc...

```
<orderitem id="${order.orderItem.itemId}" order="${order.orderId}">
    <customer>
        <name>${order.customerName}</name>
        <number>${order.customerNumber?c}</number>
    </customer>
    <details>
        <productId>${order.orderItem.productId}</productId>
        <quantity>${order.orderItem.quantity}</quantity>
        <price>${order.orderItem.price}</price>
    </details>
</orderitem>
```

FreeMarker Template

Routing to a DataSource



continued....

...Routing to a DataSource

```
<resource-config selector="$document">
    <resource>org.milyn.db.DirectDataSource</resource>
    <param name="datasource">DBExtractTransformLoadDS</param>
    <param name="driver">org.hsqldb.jdbcDriver</param>
    <param name="url">jdbc:hsqldb:hsql://localhost:9201/milyn-hsql-9201</param>
    <param name="username">sa</param>
    <param name="password"></param>
    <param name="autoCommit">false</param>
</resource-config>
```

1: DataSource Resource Configuration
2, 3, 4: SQLExecutor Resource Configuration

```
<!-- Assert whether it's an insert or update.  Need to do this just before we do the
     insert/update, which is triggered to happen just before the order-items are processed... -->
<resource-config selector="order-items">
    <resource>org.milyn.routing.db.SQLExecutor</resource>
    <param name="executeBefore">true</param>
    <param name="datasource">DBExtractTransformLoadDS</param>
    <param name="statement">select orderId from ORDERS where orderId = ${order.orderId}</param>
    <param name="resultSetName">orderExistsRS</param>
</resource-config>

<!-- If it's an insert (orderExistsRS.isEmpty()), insert the order just before the order-items are processed... -->
<resource-config selector="order-items">
    <resource>org.milyn.routing.db.SQLExecutor</resource>
    <condition evaluator="org.milyn.javabean.expression.BeanMapExpressionEvaluator">orderExistsRS.isEmpty()</condition>
    <param name="executeBefore">true</param>
    <param name="datasource">DBExtractTransformLoadDS</param>
    <param name="statement">INSERT INTO ORDERS VALUES(${order.orderId}, ${order.customerNumber}, ${order.customerName})</param>
</resource-config>

<!-- And insert each order-item... -->
<resource-config selector="order-item">
    <resource>org.milyn.routing.db.SQLExecutor</resource>
    <condition evaluator="org.milyn.javabean.expression.BeanMapExpressionEvaluator">orderExistsRS.isEmpty()</condition>
    <param name="executeBefore">false</param>
    <param name="datasource">DBExtractTransformLoadDS</param>
    <param name="statement">INSERT INTO ORDERITEMS VALUES (${order.orderItem.itemId}, ${order.orderId}, .....</param>
</resource-config>
```

"conditional" resources

Smooks in JBossESB

JBossESB

- 4.0.x, 4.2.x
  - Smooks v0.9
    - No ability to process HUGE Messages
    - No Splitting Routing
    - No Java to Java
    - ESB Action only
      - Not used within the “core” of the ESB
- 4.3
  - Smooks v1.0
    - HUGE Messages
    - Splitting Routing
    - Java to Java
    - Etc
- 5.0+
  - ?

Smooks in a Web Environment

Smooks Servlet Filter

- Sister Project “Tinak” - Browser/Device **Recognition** & **Profiling**

```
<?xml version="1.0"?>
<!DOCTYPE device-ident PUBLIC "-//MILYN//DTD TINAK 1.0//EN" "http://www.milyn.org/dtd/device-ident-1.0.dtd">
<device-ident>
    <device name="Firefox">
        <http-req-header name="User-Agent" value=".*Firefox.*" />
    </device>
    <device name="MSIE">
        <http-req-header name="User-Agent" value=".*MSIE.*" />
    </device>
    <device name="Unknown-HTML4">
        <http-req-header name="User-Agent" value="Mozilla\/4.*" />
    </device>
</device-ident>
```

- Using Profiles in Smooks
- CSS Support

```
<?xml version="1.0"?>
<smooks-resource-list xmlns="http://www.milyn.org/xsd/smooks-1.0.xsd">

    <profiles>
        <profile base-profile="MSIE" sub-profiles="windows,large" />
        <profile base-profile="Firefox" sub-profiles="windows,large" />
    </profiles>

    <!-- Will be applied to MS Internet Explorer only... -->
    <resource-config selector="b" target-profile="MSIE">
        <resource>org.milyn.cdres.trans.SetAttributeTU</resource>
        <param name="attributeName">style</param>
        <param name="attributeValue">color: red</param>
    </resource-config>

    <!-- Will be applied to all "large" useragents (browsers) i.e.
         both MSIE and Firefox... -->
    <resource-config selector="b" target-profile="large">
        <resource>org.milyn.cdres.trans.RenameElementTU</resource>
        <param name="replacementElement">i</param>
    </resource-config>

</smooks-resource-list>
```

Smooks and other JBoss Projects/Products?

Other JBoss Projects??

- MetaMatrix?
- Hibernate?
- Seam?
- ??

Future Directions

Todos...

- Tooling
  - Smooks Configuration ... Grrrrr!!!!
    - Verbose, Everything is a Resource etc
  - Debug Tooling
  - Design Time Tooling
- Additional Cartridges & Enhancements
  - Crypto, JPA, XQuery, Javabean Cartridge + cglib etc
- Grow the Community
  - We've been too quiet!

Thank You!!

Q & A??