Smooks Example - xml-to-xml

This tutorial illustrates how Smooks (in conjunction with FreeMarker) can be used to perform an XML to XML transformation on a huge message (GBs).

It should be noted however that this tutorial can be used as the basis for an character based transformation e.g. EDI to XML, CSV to XML, XML to EDI etc. It's just a mater of defining the stream parser for the Source (e.g. the CSV Parser), and modifying the templates for the output.

SVN - **Download** - **Other Tutorials**

To Build: "mvn clean install"
To Run: "mvn exec:java"

See Mixing DOM and SAX Models with Smooks

Overview

FreeMarker is an extremely powerful templating engine. One really useful feature is the ability to create and use a **NodeModel** as the domain model in a templating operation. Smooks adds the ability to perform fragment based templating transforms, as well as the ability to use this model on huge messages (GBs).

Source and Target Message Formats

Source:

```
<order id='332'>
    <header>
        <customer number="123">Joe</customer>
    </header>
    <order-items>
        <order-item id='1'>
            <product>1</product>
            <quantity>2</quantity>
            <price>8.80</price>
        </order-item>

        <!-- etc etc -->

    </order-items>
</order>
```

Target:

```
<salesorder>
    <details>
        <orderid>332</orderid>
        <customer>
            <id>123</id>
            <name>Joe</name>
        </customer>
    <details>
    <itemList>
        <item>
            <id>1</id>
            <productId>1</productId>
            <quantity>2</quantity>
            <price>8.80</price>
        <item>

        <!-- etc etc -->

    </itemList>
</salesorder>
```

So the basic structure is the same. We just need to rearrange things a little.

Smooks Configuraton

```
<?xml version="1.0"?>
<smooks-resource-list xmlns="http://www.milyn.org/xsd/smooks-1.1.xsd"

xmlns:ftl="http://www.milyn.org/xsd/smooks/freemarker-1.1.xsd">

    <!--
    Filter the message using the SAX Filter (i.e. not DOM, so no
    intermediate DOM for the "complete" message - there are "mini" DOMs
    for the NodeModels below)....
    -->
    <params>
        <param name="stream.filter.type">SAX</param>
        <param name="default.serialization.on">false</param>
    </params>

    <!--
    Create 2 NodeModels.  One high level model for the "order"
    (header etc) and then one per "order-item".

    These models are used in the FreeMarker templating resources
    defined below.  You need to make sure you set the selector such
    that the total memory footprint is as low as possible.  In this
    example, the "order" model will contain everything accept the
    <order-item> data (the main bulk of data in the message).  The
```

```
    "order-item" model only contains the current <order-item> data
    (i.e. there's max 1 order-item in memory at any one time).
    -->
    <resource-config selector="order,order-item">
        <resource>org.milyn.delivery.DomModelCreator</resource>
    </resource-config>

    <!--
    Apply the first part of the template when we reach the start
    of the <order-items> element.  Apply the second part when we
    reach the end.

    Note the <?TEMPLATE-SPLIT-PI?> Processing Instruction in the
    template.  This tells Smooks where to split the template,
    resulting in the order-items being inserted at this point.
    -->
    <ftl:freemarker applyOnElement="order-items">
        <ftl:template><!--<salesorder>
    <details>
        <orderid>${order.@id}</orderid>
        <customer>
            <id>${order.header.customer.@number}</id>
            <name>${order.header.customer}</name>
        </customer>
    </details>
    <itemList>
        <?TEMPLATE-SPLIT-PI?>
    </itemList>
</salesorder>--></ftl:template>
    </ftl:freemarker>

    <!--
    Output the <order-items> elements.  This will appear in the
    output message where the <?TEMPLATE-SPLIT-PI?> token appears in the
    order-items template.
    -->
    <ftl:freemarker applyOnElement="order-item">
        <ftl:template><!--        <item>
            <id>${.vars["order-item"].@id}</id>
            <productId>${.vars["order-item"].product}</productId>
            <quantity>${.vars["order-item"].quantity}</quantity>
            <price>${.vars["order-item"].price}</price>
        </item>
        --></ftl:template>
    </ftl:freemarker>
```

```
</smooks-resource-list>
```

Executing the Transform

```
Smooks smooks = new Smooks("smooks-config.xml");
try {
    smooks.filter(new StreamSource(new
FileInputStream("input-message.xml")), new StreamResult(System.out));
} finally {
    smooks.close();
}
```