Smooks Example - java-to-java

This example illustrates how Smooks can be used to perform Java to Java transforms.

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

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

Overview

In this particular example, Smooks uses the SAX processing model, which means no intermediate object model is constructed for populating the target Java Object Graph. Instead, we go straight from the source Java Object Graph, to a stream of SAX events, which are used to populate the target Java Object Graph.

Source and Target Object Models

The required mappings from the source to target Object models are as follows:

Source Model Event Stream

Using the Html Reporting tool (see Execution code at end of page), we can see that the SAX Event Stream produced by the source Object Model is as follows:

```
<example.srcmodel.Order>
    <header>
        <customerNumber>
        </customerNumber>
        <customerName>
        </customerName>
    </header>
    <orderItems>
        <example.srcmodel.OrderItem>
            <productId>
            </productId>
            <quantity>
            </quantity>
            <price>
            </price>
        </example.srcmodel.OrderItem>
    </orderItems>
</example.srcmodel.Order>
```

So we need to target the Smooks Javabean resources at this event stream. This is shown in the Smooks Configuration.

Smooks Configuration

The Smooks configuration for performing this transform ("smooks-config.xml") is as follows (see the Source Model Event Stream above):

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

    <resource-config selector="global-parameters">
        <param name="stream.filter.type">SAX</param>
    </resource-config>

    <resource-config selector="example.srcmodel.Order">
        <resource>org.milyn.javabean.BeanPopulator</resource>
        <param name="beanId">lineOrder</param>
        <param name="beanClass">example.trgmodel.LineOrder</param>
        <param name="bindings">
            <binding property="customerId" selector="header/customerNumber" />
            <binding property="customerName" selector="header/customerName" />
            <binding property="lineItems" selector="${lineItems}" />
        </param>
    </resource-config>

    <resource-config selector="orderItems">
        <resource>org.milyn.javabean.BeanPopulator</resource>
        <param name="beanId">lineItems</param>
        <param name="beanClass">example.trgmodel.LineItem[]</param>
        <param name="bindings">
            <binding selector="${lineItem}" />
        </param>
    </resource-config>

    <resource-config selector="example.srcmodel.OrderItem">
        <resource>org.milyn.javabean.BeanPopulator</resource>
        <param name="beanId">lineItem</param>
        <param name="beanClass">example.trgmodel.LineItem</param>
        <param name="bindings">
            <binding property="productCode"
selector="example.srcmodel.OrderItem/productId" />
            <binding property="unitQuantity" type="Integer"
selector="example.srcmodel.OrderItem/quantity" />
            <binding property="unitPrice" type="BigDecimal"
selector="example.srcmodel.OrderItem/price" />
        </param>
    </resource-config>

</smooks-resource-list>
```

Smooks Execution

The source object model is provided to Smooks via a **org.milyn.delivery.JavaSource** Object. This object is created by passing the constructor the root object of the source model. The resulting JavaSource object is used in the **Smooks#filter** method. The resulting code could look like as follows:

```
protected LineOrder runSmooksTransform(Order srcOrder) throws IOException,
SAXException {
        Smooks smooks = new Smooks("smooks-config.xml");
        ExecutionContext executionContext = smooks.createExecutionContext();

        // Transform the source Order to the target LineOrder via a
        // JavaSource and JavaResult instance...
        JavaSource source = new JavaSource(srcOrder);
        JavaResult result = new JavaResult();

        // Configure the execution context to generate a report...
        executionContext.setEventListener(new
HtmlReportGenerator("target/report/report.html"));

        smooks.filter(source, result, executionContext);

        return (LineOrder) result.getBean("lineOrder");
    }
```