JooReports 2.0 Manual (draft)

03/07/2005 - Mirko Nasato

Table of Contents

Introduction

JooReports is a Java solution for generating office documents, based on *OpenOffice.org**. You may have guessed that the *Joo* prefix stands for *Java* and *OpenOffice.org*.

JooReports is organised into 3 different modules:

- ***Templates***: generates OpenDocument documents from templates and data. Templates are regular OpenDocument files enriched with *FreeMarker** expressions. Data can be passed as Java objects or in XML format.

- ***Converter***: converts documents into different formats. It is used to convert an OpenDocument generated using *Templates* into PDF, RTF, Word, Excel or many other formats. It can also be used as a standalone tool to convert between many different document formats, e.g. Word to PDF or PowerPoint to Flash.

- ***Web Integration***: provides support infrastructure for generating documents in a web environment. This is based on the *Spring framework**, including its MVC web layer. The included web application can be used as a staring point for cusomizations, or directly for generating documents from data passed from other web applications. For the latter it is sufficient to add the templates and configure the XML mappings, with no additional code needed. (So ideally you could use it even if you don't know Java.)

Templates

A template is based on a regular OpenDocument file, edited to include *FreeMarker** expressions and directives.

Simple Templates

Let's start with the classic “Hello” example. Using OpenOffice.org Writer, create a Text document with the following content

```
Hey ${name}!
```

where *${name}* is a FreeMarker expression that will be replaced when this template is processed. Now save this document as *hello-template.odt,* and you have just created your first *JooReports* template!

To generate a document based on that template, here's the Java code you would use

```
Map model = new HashMap();
model.put("name", "Joe");  // variable to be replaced in the template
DocumentTemplate template =
      new ZippedDocumentTemplate(new FileInputStream("hello-template.odt"));
template.createDocument(model, new FileOutputStream("output.odt");
```

That's it. This will create an OpenDocument Text file *output.odt* containing

```
Hey Joe!
```

You may notice that we used a *HashMap* for passing the variables to be replaced; this is the easiest way for simple cases but you can have more complex data models, basically any Java object or an XML stream.

Advanced Templates

In the previous example we used as template a document saved directly by *OpenOffice.org*. This works if your template only contains simple variables, such as *${name}*. If you need more power, like repeating a table row over a list of elements, you need to modify your template by hand inserting *FreeMarker* directives.

An OpenDocument is basically a ZIP file containing different entries, most of them are in XML format. We are interested in the document content, so what we need to modify is the *content.xml* entry. So unzip the template to a directory, and edit *content.xml* adding whatever *FreeMarker* instructions you like. You can then zip everything back into a single file and use a *ZippedDocumentTemplate* as in the first example, or leave it in an expanded directory passed to *UnzippedDocumentTemplate*.

Please see the *FreeMarker* documentation for a complete reference of the template language. (Only section "3. The Template" of chapter "I. Designer's Guide" and chapter "IV. Reference" are of interest for our purposes.)

Please see the Order sample for an example of advanced template.

Data Models

JooReports uses *FreeMarker* for all templating features.

A data model can be a regularJava object, as a simple Map or using JavaBean properties.

Additionally you can use an XML file using *FreeMarker*'s XML support.

Converter

To perform conversions between different document formats *JooReports Converter* needs to connect to a running *OpenOffice.org 2.0* instance.

So before using *JooReports Converter* you need to start *OpenOffice.org* in listening mode, on the same machine and under the same user privileges.

Please see the *OpenOffice.org* Developer's Guide on how to do this. A quick way is to type on the command line

```
soffice -headless -accept="socket,port=8100;urp"
```

The *JooReports* distribution include a command line tool to perform conversion. For example to convert an OpenDocument Text to PDF type the following

```
java -classpath ... net.sf.JooReports.tools.Convert document.odt document.pdf
```

Web Integration

JooReports provides support for generating office documents on a web environment, based on the the *Spring Framework* and its Web MVC layer.

The web application can be used directly for generating documents from data passed from other web applications (not necessarily Java applications, possibly even static HTML). Alternatively it can be used as a starting point for custom development.

For direct use, you need just a few steps:

- be sure to always have OpenOffice.org listening on port 8100 before starting the webapp
- do an ANT build to prepare the webapp, then copy the *"web"* directory into your application server
- put your templates into "*WEB-INF/templates"*. A template for generating a document "*invoice"* must be named "*invoice-template.odt"* or unzipped into a directory "*invoice-template"*
- configure mappings in *web.xml* and *DynamicDocument-servlet.xml* in *WEB-INF*

A few examples are included in the webapp:

- *letter*: simple template (can be edited directly in OpenOffice.org Writer), uses a predefined controller with HTTP parameters as the data model
- *order*: advanced template (hand-modified *content.xml*), uses a predefined controller that accepts a data model in XML format