Fast Docbook Rendering: Unleash The Power of tDOM, mod_websh and apache2

Simon Hefti
Netcetera AG
Zypressenstrasse 71
8040 Zurich, Switzerland
simon.hefti@netcetera.ch

ABSTRACT

Docbook, an XML application, is more and more used for technical documentation or books. While Docbook is very powerful, the Docbook conversion process is still rather slow - motivation enough to search for faster ways for XML translation. This talk demonstrates how we combined a set of Tcl extensions to build a fast Docbook converter based on tDOM, mod_websh and apache2 as a web service. One of the central features of mod_websh is the re-use of Tcl interpreters for many requests. This includes pre-compiled application code and other data structures like the Docbook XSL style sheet. The re-use of the parsed style sheets allows very fast translations of Docbook documents into various output formats such as HTML or PDF.

1. INTRODUCTION

Docbook is a XML description useful for technical documentation. While it is somewhat clumsy to write, it's main advantage is multi-channel publishing, i.e. the possibility to translate it to various different formats, say plain text or HTML. One interesting way of doing Docbook translations is by applying XSL style sheets to the document. Since both Docbook and XSL are standardized, the translation process can be formulated platform- and even language independent. Because of it's rich feature set, this makes Docbook an interesting candidate to study XSLT capabilities and performance.

Often, programmers choose Docbook for documentation because it works somewhat like the program-compile-link cycle they are so familiar with, where they can produce output for, say, different platforms from a single source. Docbook allows documentations along these lines: you specify the content in a standardized XML format, and derive different output formats from it, like HTML pages or Windows Help files. One problem is the rather slow "compile" step. Processing a reasonably large technical document containing chapters, sections, subsections, tables, images, examples, notes, and cross-references can easily take 30 seconds and more, depending on your infrastructure. This makes it tedious to edit documents when you need to verify the changes frequently.

Reason enough, we think, to see if Tcl can offer a solution to the problem. The idea is to keep a Tcl interpreter running within an apache web server, ready to use whenever we need it, and to implement the document transform as a web service. Since the style sheet itself does not change as frequently as the actual document does, we also considered to cache the parsed DOM of the XSL document (more about that in a minute).

During XSL transforms, three processes need to be performed. One, the document itself needs to be parsed in order to make its structure and content known to the transformer. Two, the XSL (also in XML format) needs to be parsed, to configure the transformer. And last, the XSL instructions need to be applied to the document, i.e. the transformation has to be performed.

Since some time, Tcl has, while being helpful, started to be considered a tool coming to a certain age. Therefore, we would not have been astonished to see the XML train pass the Tcl community. It turned out different. Tcl has, in our opinion, one of the best XML parsers around, and the fastest, too. tDOM, an open source project lead by Jochen Loewer, has reached a stability and feature set that makes it worth to be considered for all kinds of XML-related work. tDOM comes with the Mozilla Public License. In this paper, we use tDOM as XML parser and XSL transformer, and compare it's performance against other processors around.

One other way to increase performance is to ensure everything needed for the task is already ready to use. Java programmers would say that it takes time to start the virtual machine. Similarly, it takes some time to start a Tcl interpreter and to configure it for the XSLT work (by loading tDOM.so, the additional helper code and the transformer itself). We use mod_websh for this purpose. It basically provides reusable Tcl interpreters within the apache web server, enriching the interpreter with the "web::" namespace which provides commands useful for web application programming, such as command dispatching, session management and logging. In this paper, we just use mod_websh to provide us with a pool of pre configured reusable Tcl interpreters.

For this paper, we used Tcl 8.4, tDOM 0.7.4, Websh 3.5, Apache 2.0.40, and Docbook XML V4.1.2. For the performance comparison, we used a a technical documentation containing various chapters and sections as well as tables, examples, and images. On the client side, we use wget to retrieve a processed document.

2. MEASUREMENTS

We have performed measurements with Xalan-C, Saxon, Gnome XSLT, and tDOM, where we have taken all but the tDOM from the most recent Debian packages without any specific configuration (see table 1 for details). All processes were run under exactly the same conditions on an otherwise unloaded machine.

With performance measurements, tree points are of importance:

- the end-to-end processing time, i.e. the time elapsed from the moment when the user issues the command until she gets the result back,
- the correctness of the result, and
- memory usage.

Here, we do not care about memory usage. It is interesting to note, however, that xsltproc, xalan, and mod_websh/tDOM (within httpd) all use about 16 MB, while the Java processes are somewhat fatter. We have verified the correctness of the transformed documents for each XSL processors.

For the performance measurements itself, we redirected all output to /dev/null to avoid any file system dependency of the results. The tests were performed with two examples: a small one for testing purposes (short.xml), and a real-life technical documentation (t1.xml). The results are compiled in table 2

Clearly, Java based parses cannot compete for command line based usage, simply because the start-up time of the virtual machine is too long. It would be interesting to measure the performance of Java based XSLT processors in a web environment similar to the one described here for tDOM. We have not considered such a set-up, however.

That leaves us with xsltproc, xalan, tDOM and mod_websh/tDOM. The general picture is clear: tDOM outperforms xsltproc, which outperforms xalan. We find that for the examples considered, tDOM is the best-performing implementation. The difference between tDOM and xsltproc is larger for small documents. This fits nicely with the typical usage in a dynamic web application, say a portal. There, smaller XML fragments need to be transformed one at a time, depending on the user settings. For such a situation, tDOM is the XSLT processor of choice. It is interesting to note that the mod_websh/tDOM is faster than xsltproc despite the fact that communication over HTTP is involved.

3. EXAMPLE CODE

3.1 Compiling the components

Building tDOM is straight forward: configure and make, as usual. When building httpd, you have to be sure to enable mod_so in order to load mod_websh into apache. Building mod_websh is straight forward as well. For mod_websh.so, you need to specify where the httpd include files can be found.

```
# building httpd

cd src/httpd-2.0.40/
./configure --prefix=/tDOM-xslt/httpd-2.0.40 --enable-so
make; make install
# test if it works
/tDOM-xslt/httpd-2.0.40/bin/httpd -l

# buildiung Tcl

./configure --prefix=/tDOM-xslt/tcl8.4b2
make; make install

# building websh

cd websh-3.5.0/src/unix
./configure --prefix=/tDOM-xslt/websh-3.5.0 \
  --with-tcl=/tDOM-xslt/tcl8.4b2/lib/ \
  --with-tclinclude=/tDOM-xslt/tcl8.4b2/include/ \
  --with-httpdinclude=/tDOM-xslt/httpd-2.0.40/include/
make; make mod\_websh.so

# building tDOM
cd tDOM-0.7.4/unix
./configure --prefix=/tDOM-xslt/tDOM-0.7.4 \
  --with-tcl=/tDOM-xslt/tcl8.4b2/lib \
```

Name	package name	version	command line used
Xalan-C	xalan	1.2-2.2	xalan -Q -XSL $xslfn -IN $xmlfn
Xalan-J			
Saxon	saxon-catalog	20000203-5	saxoncat $xmlfn $xslfn
Gnome XSLT	xsltproc	1.0.18-0.1	xsltproc $xslfn $xmlfn
tDOM		0.7.4	websh3.5 mwxslt.ws3
tDOM/mod_websh		3.5	wget -q http://localhost:5000/mwxslt.ws3

Table 1: XSLT parsers and versions

Parser	time	
	t1.xml [ms]	short.xml [ms]
Gnome XSLT	1800 ± 400	1000 ± 30
tDOM	1300 ± 100	390 ± 20
mod_websh/tDOM	1550 ± 80	370 ± 20
Xalan-C	5200 ± 60	2300 ± 30
Saxon	n/a	29000
Xalan-J	n/a	n/a

Table 2: XSLT Performance Comparison

```
  --with-tclinclude=/tDOM-xslt/tcl8.4b2/include/
make; make install
```

3.2 mod_websh configuration

In order to load mod_websh into httpd, you need the following configuration lines in `httpd.conf`:

```
LoadModule websh_module /path/to/mod_websh3.5.0.so
AddHandler websh .ws3
```

This tells httpd to use mod_websh whenever a file with the extension `.ws3` is requested by the client. Remember to start `httpd` with the `LD_LIBRARY_PATH` set properly. Otherwise, Tcl will not be found.

3.3 XSLT with tDOM

Here is the pseudo code for a tDOM based xsl processor:

```
# load tDOM

load   /tDOM-xslt/src/tDOM-0.7.4/unix/tDOM0.7.so
source /tDOM-xslt/src/tDOM-0.7.4/lib/tDOM.tcl

# parse xml

set xmldoc [dom parse \
  -baseurl $baseurl \
  -keepEmpties \
  -externalentitycommand ::fxslt::extRefHandler \
  $xmldata]

# and same for xsl ...

# access the root node of the xml document
set xmlroot [$xmldoc documentElement]

# call the method ''xslt'' on the root node,
# giving the xslt dom parsed before

$xmlroot xslt $xsldoc resultDoc

# access root node of the resulting document
set root [$resultDoc documentElement]

# and show it as HTML
set res [$root asHTML]

# there might be more data:
set nextRoot [$root nextSibling]
while {$nextRoot != ""} {
    append res [$nextRoot asHTML]
    set nextRoot [$nextRoot nextSibling]
}
```

4. SUMMARY

In this paper, we have compared the performance of different XSLT processors based on a real-life example of technical documentation. We find that the tDOM parser for Tcl is a very fast, faster in fact than the other C-based processor, `xsltproc`, despite that fact that it is called from a scripted environment. We find that the tDOM performance can be made available as a web service through the usage of mod_websh, which pools pre-configured Tcl interpreters within a Apache web server, ready to use whenever a request comes in.

5. RESOURCES

- tDOM: http://loewerj.freeshell.org/tDOM.cgi
- websh: http://tcl.apache.org/websh/
- httpd: http://httpd.apache.org/
- Tcl: http://www.tcl.tk/
- xalan: http://xml.apache.org/xalan-c
- Saxon: http://saxon.sourceforge.net/
- xsltproc: http://xmlsoft.org/XSLT/

6. ACKNOWLEDGMENTS

Many other persons at Netcetera supported various aspects of this work, especially Corsin Decurtins and Ronnie Brunner.