Downloading and installing software for the Saxon-JS workshop at Declarative Amsterdam 2020

Post-conference note: the temporary Saxon-EE licence, included with this software, has expired. Refer to the section “About licences” below. You may want to replace it with your own licence, which you may have to buy at Saxonica. Maybe they will also supply a trial licence.

You will need to replace the file `saxon-license.lic` in the folder `home/config`.

Alternatively, you may compile stylesheets manually by using node.js or Oxygen. In that case, you will need to tweak the XSLWeb configuration. Contact me (`pieter at masereeuw dot nl`) if you would like me to help you.

Installing the XSLWeb software and the workshop files

The XSLWeb distribution that we are going to run is the ready-to-run version that you can download from Github. However, we are using a version that has been extended with files for this workshop.

The following is copied (and slightly adapted) from the original XSLWeb manual:

Installation of the ready-to-run distribution

Download

The ready-to-run distribution can be downloaded from:

- https://www.masereeuw.nl/declarative-amsterdam/xslweb-v4.0.0-RC1-linux.tar.gz (for Linux 64-bits machines)
- http://www.masereeuw.nl/declarative-amsterdam/xslweb-v4.0.0-RC1-osx.tar.gz (for macOS 64-bits machines)
- http://www.masereeuw.nl/declarative-amsterdam/xslweb-v4.0.0-RC1-windows.zip (for Windows 64-bits machines)

Apart from the .war file, each distribution file contains a platform specific custom OpenJDK version 13 JRE, the Jetty Java application server and several platform specific batch files or shell scripts.

Unpack

After downloading, the distribution file may be unzipped/untarred in a directory of your choice, under Windows and macOS by double clicking on the file and under Linux and macOS by running the following command in a shell:

Unpacking the zipped tar file on Linux

```
tar -xzvf xslweb-ready-to-run-v4.0.0-RC1-linux-x64.tar.gz
```

Unpacking the zipped tar file on macOS

```
tar -xzvf xslweb-ready-to-run-v4.0.0-RC1-osx-x64.tar.gz
```

In case the browser already zipped the file during the download process, the commands are:

Unpacking the tar file on Linux

```
tar -xvf xslweb-ready-to-run-v4.0.0-RC1-linux-x64.tar
```

Unpacking the tar file on macOS

```
tar -xvf xslweb-ready-to-run-v4.0.0-RC1-osx-x64.tar
```

Renaming the installation directory

If you like, you may give the installation directory a new and/or shorter name, but you don't need to: its name is irrelevant, but if you do change the name, it may be wise not to use spaces or special characters in its path.

Run and install

XSLWeb can be started by running the batch file `<install-dir>\bin\run-service.bat` (Windows) or `<install-dir>/bin/run-service.sh` (*nix). After the service is started, the correct installation can be tested by opening a browser and going to the url http://localhost:8152. XSLWeb can be stopped by running the corresponding batch file `stop-service.bat` or shell script `stop-service.sh` from another command prompt or shell.

> To increase or decrease the Java heap memory size that is available to the XSLWeb process please alter the variable `MAXMEM` in the file `<install-dir>\bin\service\config.cmd` (Windows) or `<install-dir>/bin/run-service.sh` (Linux and macOS). The default is 256m (megabytes).

Why XSLWeb?

Because Saxon-JS is written in Javascript, and because running Javascript from a local file system can cause issues, the demo files of the Saxon-JS workshop will need to be accessed using a web server.

As a web server, the choice was made for XSLWeb. XSLWeb is a webserver that is entirely configured using XML and XSLT. The main reason for selecting XSLWeb for this workshop, is that it can automatically compile a Saxon-JS stylesheet when it is requested by a browser. Another reason is that XSLWeb makes it possible to use XSLT to create the boilerplate code of the exercises: the HTML `<head>`-sections and the footers with navigation links. The nice thing about XSLWeb is that it allows you to use only XSLT in the back end – and in the front end.

Although this workshop is not about XSLWeb, feel free to have a look at it. It is nice.

Compiling a Saxon-JS stylesheet

Compiled Saxon-JS stylesheet files end in the extension `.sef.json`. XSLWeb will automatically compile the corresponding xslt-source in order to send the sef.json-file to the browser. In development mode (which is the case for this workshop), it will not cache the result. Instead, it will recompile on every request.

About licences

In order to compile a Saxon-JS-stylesheet with Saxon's Java version, you need a licence file. Saxonica has kindly provided us with a temporary licence, which is included in the distribution. This licence is valid until October 21, 2020. If you want to use Saxon-JS after that, consult the Saxonica website for your options – there is a free compiler (written in Saxon-JS!) that does not require a licence. The Java version is better, but you need a Saxon EE (enterprise) licence for that, which costs (at this moment) £360.00. Consider supporting Saxonica. It is a great company!

Troubleshooting stylesheets

If something appears to go wrong during compilation, check the terminal window where XSLWeb puts its log messages (alternatively, you can look for its logfile, `xslweb.log`).

If the stylesheet does not run as expected, open the browser window (often using function key F12) and check the browser console. It may also help to send debug messages to the console using `<xsl:message>`.

Fighting the browser cache

Before starting the workshop, it may help if you find out beforehand how you can clear the cache of the browser you are going to use. Firefox has a nice add-on that lets you press F9 to clear the cache. It is called, not surprisingly, *Clear Cache*. XSLWeb can send "do not cache" headers to the browser, but you never know.

Editing XML and XSLT files

While it is possible to use a simple editor like Notepad for editing XML files, it is nicer to use an editor that helps you by means of syntax highlighting, syntax checking and code suggestions. As we cannot all afford a commercial product, it may well be worth the effort to find a free tool that suits your need. One of the tools that I use when I find myself lost somewhere without Oxygen, is Notepad++. It has an *XML Tools* plug-in that does syntax highlighting, can check well-formedness and can do some pretty printing. If you know of other or better tools, also for the non-Windows platform, I'd like to hear from you.

Pieter Masereeuw, `pieter at masereeuw dot nl`.