XML Spy

Author: Mihai Rotaru

Date: 16 Dec 2011

XML Spy is described by the company developing it as a an advanced XML editor for modeling, editing, transforming, and debugging XML-related technologies." From this description, it is obvious that use case scenarios for this program are numerous and it would be impossible to provide an in-depth review without spending a lot of time using it, while having at least some degree of familiarity with the technologies it works with.

This review will focus on XML Spy's capacity of aiding XML document authoring. Lately, I accumulated quite a few XML files, due to the fact that I'm writing my blog entries for this module as XML files, which I then use an XSLT template to transform into HTML. Then I post the HTML code to my blog, and paste it into the Word document which is my report. These files will provide sufficient material to compare XML Spy with my current setup - the GVim editor, with a number of plugins to aid with XML editing, and the XML Star command-line utitlies which I use from within GVim to perform tasks such as validating and automatic escaping of characters like < and &.

XML Spy is not free software; to evaluate the program, one is required to give his email address to receive a key used to actvate the program. After receiving the confirming email, I proceded do download the software. The first thing I found out about it during this process is that it is not lightweight; the installation package weights in at a hefty 55MB; installed, it takes up 250MB and during installation requires at least double that amount, for storing temporary files.

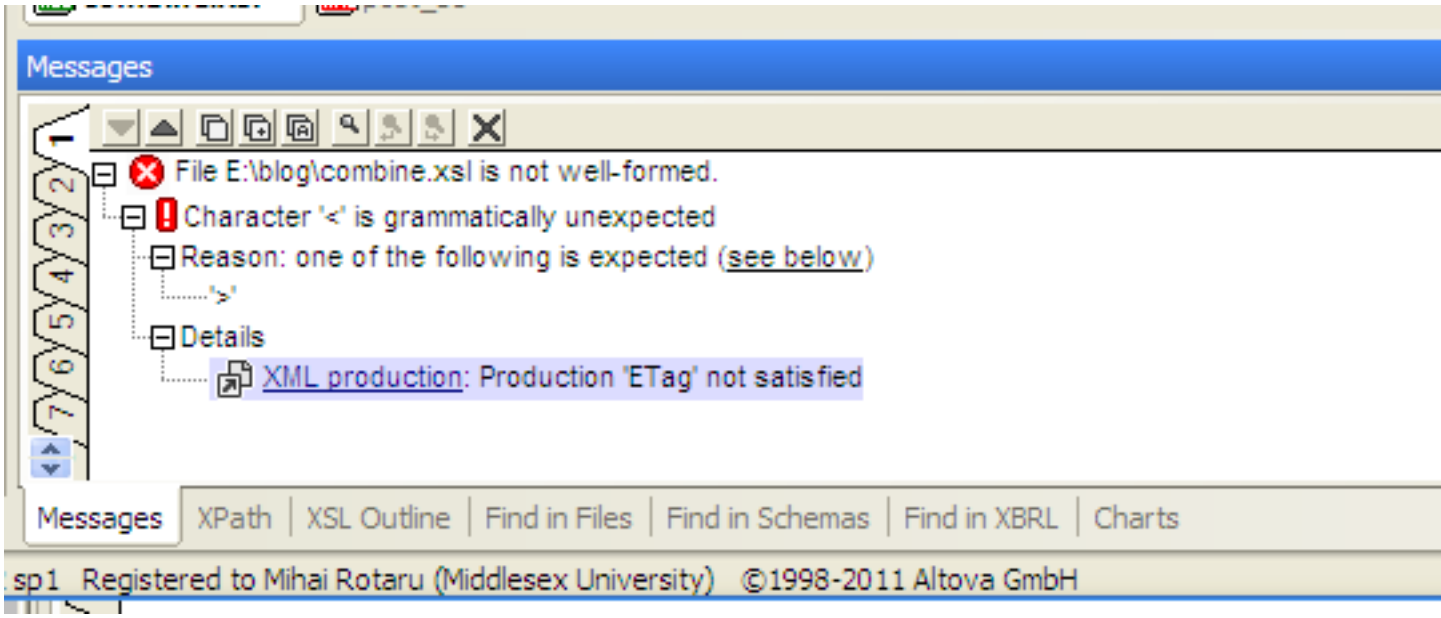
Ather going through the throes of installing this beast, I fired it up, eager to put it to the test. The first impression I got was that the interface was bloated and outdated. Many of the buttons are too small, and the editor has a lot of windows opened by default. Fortunatelly, they can be closed - but the size of the buttons cannot be increased, which makes them a little bit harder to click. These are small greivances, but they do contribute signifficantly to the overall impression the program makes.

The first thing I did after taking a close look at the user interface and menu options, was opening a couple of my XML documents. I opened the combine.xsl and one of the posts for the blog, post_35.xml. After browsing the contents of the files and making small changes for a while, to get an impression of how the editor handles such tasks, I was under the impression that the editor is too wastefull with screen space, and has too many little details which make it harder to concentrate on the code.

I was curious to see how does XML Spy treat illegal characters in XML files - in the editor I use, they are simply hilighted in red. But when I tried to type illegal characters in XML Spy, it simply ignored the keystrokes. I think that is a bad way of handling such issues - even though the character is illegal, the editor should not make decisions for the user, without asking for permission, or even notifying the user. I tried typing such characters later, and the editor accepted the keystrokes, I'm not sure why. When typing an amersand, the editor will show up a pop-up menu, whith the available entities - such as & - but if the user keeps typing, the menu will not update itselft to reflect the new keystrokes.

XML Spy does offer convenient ways of checking for validity and well-formedness, from the XML menu; it also has keyboard shortcuts for these probably quite common operations - to check for well-formedness, all the user needs to do is press F7. For validity testing, the keyboard shortcut is F8.

Error messages XML Spy produces when incorrect syntax is encountered can be confusing. For example, after removing the > character which closed a tag, I pressed F7. XML Spy came up with the followin error:

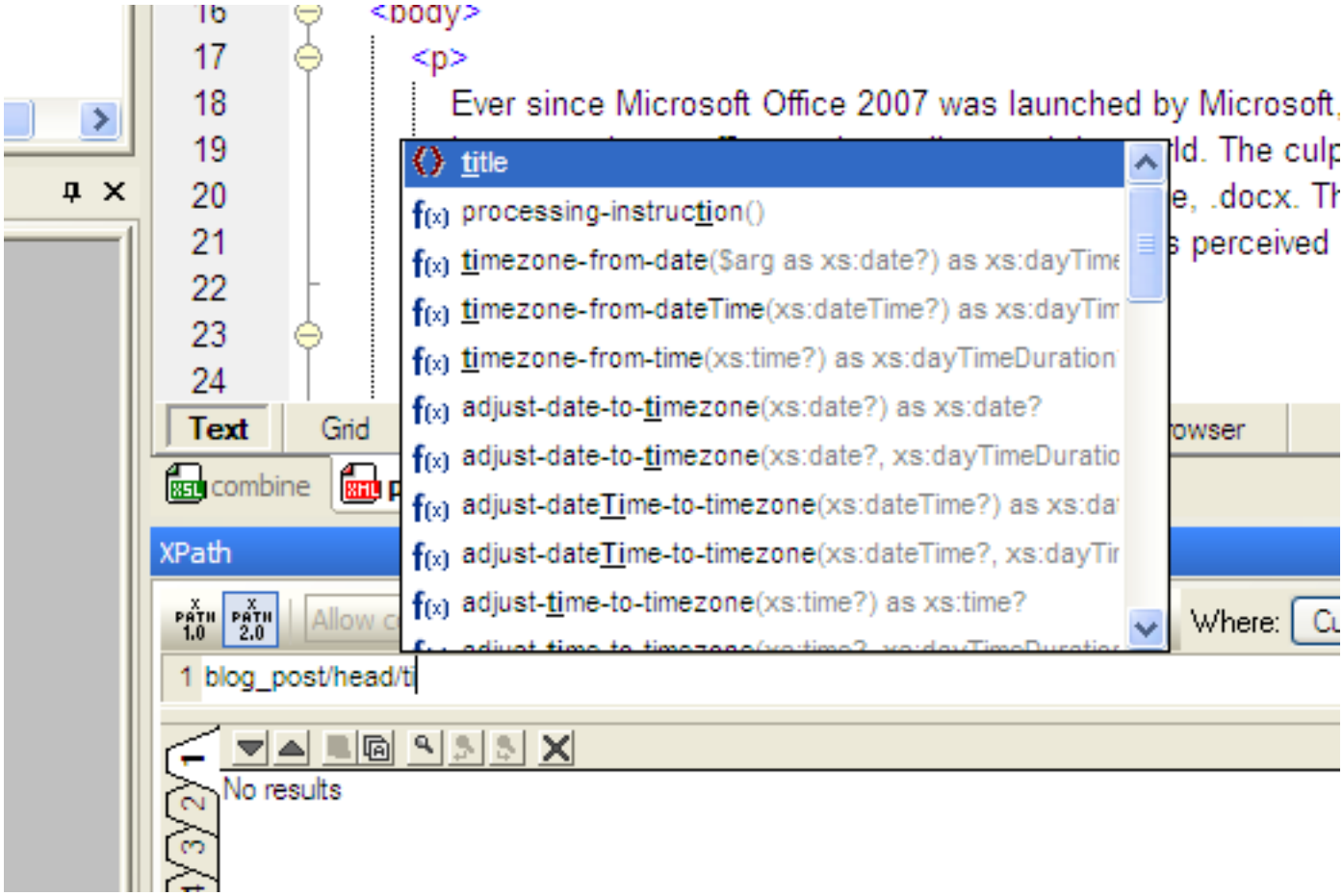


Although the message is correct, it could be more succint; for example, here is the output generated when validating the same document with the xmlstar command line utility:

```
MR917@PC-00104078 /
$ xml val -e /e/blog/combine.xsl
e:/blog/combine.xsl:13.1: expected '>'
                </html>
                ^
e:/blog/combine.xsl - invalid
```

What I did like about XML Spy, however, is that it automatically checked the xsl file for validity after I fixed the error, and saved it. It wasn't expecting that, but it was helpfull since I actually forgot to check for validity/well-formedness before saving. However, for my other open document - the post_35.xml - when I made some small changes and saved it, it did not perform the validity test; XML Spy does not check the validity of documents which do not declare a DTD/Schema. But it could have performed a well-formedness test. However, it did not do it automatically.

XML Spy has very user-friendly XPath tools. The "XML -> Evaluate XPaht..." menu option brings up the XPath window, shown below. XPath expressions can be written in the textbox, and the results will be printed in the window below. This window is very convenient, and also features auto-completion of both elements in the document, as well as XPath functions. For example, the value of the'title' element can be accessed like this:



Pressing "Enter" will result in the value of the title being printed in the "Output" window - in this case, ".doc versus .docx". Clicking the result will result in the window moving to the corresponding element, and hilighting it. Another powerfull feature is that XML Spy allows users to select the version of XPath used, and supports both XPath 1.0 and XPath 2.0.

Although XML Spy is quite heavy, not portable (only available on Windows) and not free, it is also provides a very capable XML editing environment. However, students such as myself are probably not the target audience of this package; it's costly, and complex. Freely available tools do a good enough job for basic XML authoring scenarios, while IDE's such as XML Spy can provide the editing power when it is really needed.