Xul Documentation

Release 2.4.2

Peter Adrichem

Apr 18, 2022

Contents

XML[1] utilities written in Python.

Current version: 2.4.2

[1] Extensible Markup Language (XML) 1.0

CHAPTER 1

Xul scripts

1.1 ppx – Pretty Print XML

Use `ppx` to pretty print an *XML source* in human readable form.

```
ppx file.xml
```

1.1.1 White Space

For greater readability `ppx` removes and adds *white space*.

Warning: White space can be significant in an XML document[1]. So be careful when using `ppx` to rewrite XML files.

1.1.2 Options

`ppx` can be used with the following command-line options:

```
$ ppx --help

usage: ppx [-h] [-V] [-n] [-o] [xml_source [xml_source ...]]

Pretty Print XML source in human readable form.

positional arguments:
  xml_source            XML source (file, <stdin>, http://...)

optional arguments:
```

(continues on next page)

[1] Extensible Markup Language §2.10 White Space Handling

(continued from previous page)

```
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit
  -n, --no-syntax       no syntax highlighting
  -o, --omit-declaration
                        omit the XML declaration
```

1.1.3 Syntax Highlighting

`ppx` will syntax highlight the XML source if you have Pygments installed.

Pretty print the XML Schema 1.0 schema document:

```
ppx http://www.w3.org/2001/XMLSchema.xsd
```

`-n, --no-syntax`

You can disable syntax highlighting with the `--no-syntax` option.

1.1.4 XML declaration

XML documents should begin with an XML declaration which specifies the version of XML being used[2].

By default `ppx` will print an (UTF-8) XML declaration.

`-o, --omit-declaration`

Omit the XML declaration with the `--omit-declaration` option.

```
ppx --omit-declaration file.xml
```

1.1.5 Examples

Pretty print any local XML file:

```
ppx data_dump.xml
```

RSS feed:

```
ppx http://feeds.feedburner.com/PythonInsider
```

Page XML file with less:

```
ppx xml/large.xml | less -RX
```

Redirect output (pipe) to `ppx`:

```
curl -s https://www.python.org/dev/peps/peps.rss/ | ppx
```

Rewrite XML:

```
ppx -n data_dump.xml > pp_data_dump.xml
```

[2] Extensible Markup Language §2.8 Prolog and Document Type Declaration

1.2 xp – Select nodes with XPath

1.2.1 XPath expression

Select nodes in an *XML source* with an XPath[1] expression.

List all attributes of an XML file:

```
xp "//@*" file.xml
```

List the latest Python PEPs:

```
curl -s https://www.python.org/dev/peps/peps.rss/ | \
xp "//item/title/text()"
```

List the latest Python PEPs with their link:

```
curl -s https://www.python.org/dev/peps/peps.rss/ | \
xp "//item/*[name()='title' or name()='link']/text()"
```

1.2.2 Options

xp can be used with the following command-line options:

```
$ xp --help

usage: xp [-h] [-V] [-e] [-d DEFAULT_NS_PREFIX] [-r] [-p] [-m] [-f | -F] [-q]
          xpath_expr [xml_source [xml_source ...]]

Select nodes in an XML source with an XPath expression.

positional arguments:
  xpath_expr            XPath expression
  xml_source            XML source (file, <stdin>, http://...)

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit
  -e, --exslt           add EXSLT XML namespace prefixes
  -d DEFAULT_NS_PREFIX, --default-prefix DEFAULT_NS_PREFIX
                        set the prefix for the default namespace in XPath
                        [default: 'd']
  -r, --result-xpath    print the XPath expression of the result element (or
                        its parent)
  -p, --pretty-element  pretty print the result element
  -m, --method          use ElementTree.xpath method instead of XPath class
  -f, -l, --files-with-hits
                        only the names of files with a non-false and non-NaN
                        result are written to standard output
  -F, -L, --files-without-hits
                        only the names of files with a false or NaN result, or
                        without any results are written to standard output
  -q, --quiet           don't print the XML namespace list
```

[1] XML Path Language (XPath) 1.0

1.2.3 Print result's XPath

-r, --result-xpath

Print the XPath expression of each result element with the `--result-xpath` option. Each XPath expression will have an absolute location path.

```
xp --result-xpath "//title" file.xml
```

If an XPath result is a text or attribute node `xp` prints the parent element's XPath expression.

List the XPath expressions of all elements with attributes:

```
xp -r "//@*" file.xml
```

1.2.4 Namespaces in XML

List all the XML namespaces[2] (prefix, URI) of the document element:

```
xp 'namespace::*' file.xml
```

Print the default namespace of the document element, if it has one:

```
xp 'namespace::*[name()=""]' file.xml
```

The default XML namespace in an XML document has no prefix (*None*). To select nodes in an XML namespace XPath needs prefixed names (qualified names). Therefore `xp` uses `d` as the prefix for the default XML namespace.

List the five most recent Python Insider posts:

```
xp "descendant::d:entry[position()<=5]/d:title/text()" \
http://feeds.feedburner.com/PythonInsider
```

-d <prefix>, **--default-prefix** <prefix>

You can change the prefix for the default namespace with the `--default-prefix` option:

```
xp -d p "descendant::p:entry[position()<=5]/p:title/text()" \
http://feeds.feedburner.com/PythonInsider
```

1.2.5 Extensions to XSLT

-e, --exslt

lxml supports the EXSLT[3] extensions through libxslt (requires libxslt 1.1.25 or higher). Add EXSLT namespaces with the `--exslt` command-line option.

Find Python Insider posts published in or after 2015 with EXSLT (`date` prefix):

```
xp -e "//d:entry[date:year(d:published) >= '2015']/d:title/text()" \
http://feeds.feedburner.com/PythonInsider
```

Python Insider posts updated in December:

[2] Namespaces in XML 1.0
[3] Extensions to XSLT (EXSLT)

```
xp -e "//d:entry[date:month-name(d:updated) = 'December']/d:title/text()" \
http://feeds.feedburner.com/PythonInsider
```

-q, --quiet

The `--quiet` command-line option will not print the list with XML namespaces.

Use the power of regular expression (`re` prefix). Find Python PEPs with "remove" or "specification" in the title (case-insensitive):

```
curl -s https://www.python.org/dev/peps/peps.rss/ | \
xp -e '//item/title[re:match(text(), "(remove|specification)", "i")]' -q
```

1.2.6 Pretty print element result

-p, --pretty-element

A result element node can be pretty printed with the `--pretty-element` command-line option.

> **Warning:** The `--pretty-element` option removes all white space text nodes *before* applying the XPath expression. Therefore there will be no white space text nodes in the results.

Pretty print the latest Python PEP:

```
curl -s https://www.python.org/dev/peps/peps.rss/ | xp "//item[1]" -p
```

1.2.7 Print file names

-f, -l, --files-with-hits

The `--files-with-hits` command-line option only prints the names of files with an XPath result that is not false and not NaN (not a number).

Find XML files with HTTP URL's:

```
xp "//mpeg7:MediaUri[starts-with(., 'http://')]" *.xml -f
```

XML files where all the book prices are below € 25,-.

```
xp -el "math:max(//book/price[@currency='€'])<25" *.xml
```

-F, -L, --files-without-hits

The `--files-without-hits` command-line option only prints the names of files without any XPath results, or with a false or NaN result.

XML files without a person with the family name 'Bauwens':

```
xp "//mpeg7:FamilyName[text()='Bauwens']" *.xml -F
```

1.2.8 xpath method

-m, --method

xp uses the *lxml.etree.XPath* class by default. You can choose the *lxml.etree.ElementTree.xpath* method with the --method command-line option. The results should be the same but error reporting can be different.

1.3 validate – Validate XML

The validate script can check if an *XML source* conforms to an XML schema. It supports the following XML schema languages.

1.3.1 XSD

-x <xml_schema>, **--xsd** <xml_schema>

Use the --xsd option to validate an XML source with an XSD[1] file:

```
validate -x schema.xsd source.xml
```

1.3.2 DTD

-d <dtd_schema>, **--dtd** <dtd_schema>

Validate an XML source with a DTD[2] file with the --dtd option:

```
validate -d doctype.dtd source.xml
```

1.3.3 RELAX NG

-r <relax_ng_schema>, **--relaxng** <relax_ng_schema>

The --relaxng option validates an XML source with a RELAX NG[3] file:

```
validate -r relaxng.rng source.xml
```

1.3.4 Options

validate can be used with the following command-line options:

```
$ validate --help

usage: validate [-h] [-V] (-x XSD_SOURCE | -d DTD_SOURCE | -r RELAXNG_SOURCE)
                [-f | -F]
                [xml_source [xml_source ...]]

Validate XML source with XSD, DTD or RELAX NG.
```

(continues on next page)

[1] XML Schema 1.0 and 1.1
[2] XML Document Type Definition
[3] RELAX NG Specification

(continued from previous page)

```
positional arguments:
  xml_source            XML source (file, <stdin>, http://...)

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit
  -x XSD_SOURCE, --xsd XSD_SOURCE
                        XML Schema Definition (XSD) source
  -d DTD_SOURCE, --dtd DTD_SOURCE
                        Document Type Definition (DTD) source
  -r RELAXNG_SOURCE, --relaxng RELAXNG_SOURCE
                        RELAX NG source
  -f, -l, --validated-files
                        only the names of validated XML files are written to
                        standard output
  -F, -L, --invalidated-files
                        only the names of invalidated XML files are written to
                        standard output
```

1.3.5 XML Validation

Validate XHTML with the `XHTML 1.0 strict DTD`:

```
curl -s https://www.webstandards.org/learn/reference/templates/xhtml10s/ | validate -
↪d examples/dtd/xhtml1-strict.dtd
```

Validate XHTML with the `XHTML 1.0 strict XSD`:

```
curl -s https://www.webstandards.org/learn/reference/templates/xhtml10s/ | validate -
↪x examples/xsd/xhtml1-strict.xsd
```

1.3.6 Validation Errors

If an *XML source* doesn't validate the `validate` script will show the reason with some additional information:

```
validate -x TV-Anytime.xsd NED120200816E.xml

XML source 'NED120200816E.xml' does not validate
line 92, column 0: Element '{urn:tva:metadata:2019}Broadcaster': This element is not␣
↪expected. Expected is one of ( {urn:tva:metadata:2019}FirstShowing,
↪{urn:tva:metadata:2019}LastShowing, {urn:tva:metadata:2019}Free ).
```

1.3.7 XSD Validation

Validate an XSD file with the `XML Schema schema document`:

```
validate -x examples/xsd/XMLSchema.xsd schema_file.xsd
```

Validate the XML Schema 1.1 XSD with the (identical) XML Schema schema document:

```
validate -x examples/xsd/XMLSchema.xsd http://www.w3.org/2009/XMLSchema/XMLSchema.xsd
```

And vice versa:

```
validate -x http://www.w3.org/2009/XMLSchema/XMLSchema.xsd examples/xsd/XMLSchema.xsd
```

Validate the XML Schema XSD with the `DTD for XML Schema`:

```
validate -d examples/dtd/XMLSchema.dtd examples/xsd/XMLSchema.xsd
```

1.3.8 Print file names

`-f, -l, --validated-files`

The `-f, -l, --validated-files` command-line option only prints the names of validated XML files.

Find XML files that validate:

```
validate -x schema.xsd *.xml -l
```

`-F, -L, --invalidated-files`

The `-F, -L, --invalidated-files` command-line option only prints the names of XML files that don't validate.

Remove XML files that fail to validate:

```
validate -x schema.xsd *.xml -L | xargs rm
```

1.4 transform – Transform XML

`transform` is a simple command-line script to apply XSLT[1] stylesheets to an *XML source*. If you need a command-line XSLT processor with more options have a look at xsltproc

Transform an XML file:

```
transform stylesheet.xsl file.xml
```

Transform an XML file and *pretty print* the result:

```
transform --xsl-output stylesheet.xsl file.xml | ppx
```

1.4.1 Options

`transform` can be used with the following command-line options:

```
$ transform --help

usage: transform [-h] [-V] [-x | -o] [-f FILE] xslt_source xml_source

Transform XML source with XSLT.
```

(continues on next page)

[1] XSL Transformations (XSLT) 1.0

(continued from previous page)

```
positional arguments:
  xslt_source           XSLT source (file, http://...)
  xml_source            XML source (file, <stdin>, http://...)

optional arguments:
  -h, --help            show this help message and exit
  -V, --version         show program's version number and exit
  -x, --xsl-output      honor xsl:output
  -o, --omit-declaration
                        omit the XML declaration
  -f FILE, --file FILE  save result to file
```

1.4.2 XSL output

-x, --xsl-output

You can honor the `xsl:output` element[2] with the `--xsl-output` option.

```
transform --xsl-output stylesheet.xsl file.xml
```

1.4.3 Save transformation result to file

-f FILE, **--file** FILE

Example stylesheet that converts an XML document to UTF-16 encoding:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  version="1.0" id="utf16"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="xml" version="1.0" encoding="UTF-16" indent="yes" />

  <xsl:template match="/">
   <xsl:copy-of select="." />
  </xsl:template>

</xsl:stylesheet>
```

Save the transformation result to a little-endian UTF-16 Unicode text file.

```
transform --xsl-output to_utf16.xsl utf8.xml --file utf16.xml
```

When saving to file use the `--xsl-output` option to preserve the character encoding of the transformation.

1.4.4 XML declaration

XML documents should begin with an XML declaration which specifies the version of XML being used[3].

-o, --omit-declaration

[2] XSL Transformations: 16 Output

[3] Extensible Markup Language §2.8 Prolog and Document Type Declaration

You can omit the XML declaration with the `--omit-declaration` option.

```
transform --omit-declaration stylesheet.xsl file.xml
```

CHAPTER 2

Other

2.1 XML source

The Xul scripts require an XML source to operate on. An XML source can be a local file, an URL or a pipe.

2.1.1 File

With `xp` you can select nodes in a local XML file with an XPath expression:

```
xp 'node()' entity.xml
```

2.1.2 Pipe

Redirect output (pipe) to a Xul script:

```
curl -s https://developer.apple.com/news/rss/news.rss | ppx
```

2.1.3 URL

libxml2 also supports loading XML through HTTP (and FTP). For example, to pretty print an RSS feed:

```
ppx http://feeds.launchpad.net/pytz/announcements.atom
```

Loading XML through HTTPS is not supported and will result in an *failed to load external entity* error.

XHTML

XHTML[1] is part of the family of XML markup languages. It's obsolete.

[1] XHTML™ 1.0 The Extensible HyperText Markup Language

2.1.4 Examples

Pretty print an XHTML document:

```
curl -s https://www.webstandards.org/learn/reference/templates/xhtml11/ | ppx
```

Validate an XHTML document with the `XHTML 1.0 strict DTD`:

```
curl -s https://www.webstandards.org/learn/reference/templates/xhtml10t/ | validate -
↪d examples/dtd/xhtml1-transitional.dtd
```

Print the link destinations in an XHTML document:

```
xp -d html "//html:link/@href" http://www.w3.org/1999/xhtml/
```

More XSDs and DTDs examples can be found in the Xul Bitbucket repository.

See also:

Xul scripts: *ppx*, *xp*, *validate*, *transform*

2.2 Changelog

This document records all notable changes to Xul.

2.2.1 2.4.1 (2022-02-14)

- Fixed Changelog URL.

2.2.2 2.4.0 (2022-02-14)

- Beter handling of encodings other than UTF-8 (e.g. ISO-8859, UTF-16, UCS-2, UCS-4).
- Added `--file FILE` option to *transform*: save result to file.
- *transform*: now only transforms a single file.
- Added `--xsl-output` option to *transform*: honor `xsl:output`.
- Removed xul.dom module (legacy).

2.2.3 2.3.0 (2021-01-28)

- Added `--invalidated-files` option to *validate*: only print names of invalidated files.
- Added `--validated-files` option to *validate*: only print names of validated XML files.
- *xp*: `--files-with-hits` and `--files-without-hits` options are mutually exclusive.
- Consistent broken pipes `errno.EPIPE` exit status (Python 2).

2.2.4 2.2.1 (2021-01-14)

- *xp* `--pretty-element` fix: output multiple results to a pipe (Python 2).

2.2.5 2.2.0 (2020-10-07)

- *xp*: handle *NaN*[1] result as a false result (`--files-with|without-hits`).
- Renamed *xp* `--files-without-results` option to `--files-without-hits`: only print names of files with a false or *NaN*[1] result, or without any results.
- Renamed *xp* `--files-with-results` option to `--files-with-hits`: only print names of files with a non-false and non-*NaN*[1] result.
- Added `--relaxng` option to *validate*: validate an XML source with RELAX NG.
- Refactored *validate* script.
- README: documentation is on Read The Docs.

2.2.6 2.1.0 (2020-09-09)

- Added `--quiet` option to *xp*: don't print the XML namespace list.
- Added `--files-without-results` option to *xp*: only print names of files with a false result or without any results.
- Added `--files-with-results` option to *xp*: only print names of files with XPath matches.

2.2.7 2.0.3 (2020-06-10)

- Fix output encoding when piping output to a pager like less (Python 2).

2.2.8 2.0.2 (2020-05-31)

- Fix: removed encoding from Pygments formatter so highlight returns Unicode strings.

2.2.9 2.0.1 (2020-03-08)

- Added install extra "syntax" (Pygments): `pip install Xul[syntax]`

2.2.10 2.0.0 (2020-03-07)

Open sourced Xul.

[1] NaN stands for "Not a Number".

CHAPTER 3

Installing

The Xul command-line scripts can be installed with pip:

```
pip install Xul
```

Install Xul with Pygments for XML syntax highlighting.

```
pip install Xul[syntax]
```

CHAPTER 4

Dependencies

Xul uses the excellent lxml XML toolkit, a Pythonic binding for the C libraries libxml2 and libxslt.

CHAPTER 5

Changelog

Xul *Changelog*.

CHAPTER 6

Source

The source can be found on Bitbucket.

CHAPTER 7

Indices and search

- genindex
- search

Index

Symbols

D

E

N

P

R

S

T