Packtools Documentation

Release 2.4.1

SciELO

Dec 03, 2018

Contents

Packtools is a Python library and set of command line utilities which can be used to handle SciELO Publishing Schema XML files.

CHAPTER 1

User guide

Step-by-step guide to use the features provided by **packtools**.

1.1 Installing Packtools

Packtools works with CPython 2.7, 3.3+ and depends solely on lxml>=3.3.4. Please, read lxml's installation instructions to make sure it is installed correctly.

1.1.1 Pypi (recommended)

```
$ pip install packtools
```

1.1.2 Source code (development version)

```
$ git clone https://github.com/scieloorg/packtools.git
$ cd packtools
$ python setup.py install
```

1.1.3 Installing on Windows

Requirements

- What is the Windows version?
- What is the Python version?
- Is the architecture 32 or 64 bits?
- The packages *lxml* and *packtools* can be downloaded at PyPi.

For example, if you want to install packtools on a *64 bits Windows 10* machine running *Python 2.7.3* you should download and install lxml-3.7.3.win-amd64-py2.7.exe and packtools-2.0.1-py2.py3-none-any.whl. While *lxml* comes with a double-click graphic installer, *packtools* will require the following command at the command prompt:

```
$ pip install path/to/packtools-2.0.1-py2.py3-none-any.whl
```

You can test the installation executing:

```
$ stylechecker -h
```

1.2 Tutorial

1.2.1 XML Catalog configuration

An XML Catalog is a lookup mechanism which can be used to prevent network requests from being performed while loading external DTDs.

For performance and safety, instances of `stylechecker.XMLValidator` do not perform network connections, so we strongly recommend that you set up an XML catalog, which translates public ids to local file URIs.

packtools is shipped with a standard catalog, and can be used basically in 2 ways:

1. Registering packtools' catalog in the super catalog with the appropriate delegates, which can be done by adding the following lines to make the file `/etc/xml/catalog` looks like (this is preferred for production):

```
<?xml version="1.0"?>
<!DOCTYPE catalog PUBLIC "-//OASIS//DTD Entity Resolution XML Catalog V1.0//EN"
↪"http://www.oasis-open.org/committees/entity/release/1.0/catalog.dtd">
<catalog xmlns="urn:oasis:names:tc:entity:xmlns:xml:catalog">
    <delegatePublic publicIdStartString="-//NLM//DTD JATS"
                    catalog="file://<packtools_dir>/packtools/catalogs/scielo-
↪publishing-schema.xml"/>
    <delegatePublic publicIdStartString="-//NLM//DTD Journal"
                    catalog="file://<packtools_dir>/packtools/catalogs/scielo-
↪publishing-schema.xml"/>
    <delegateSystem systemIdStartString="JATS-journalpublishing1.dtd"
                    catalog="file://<packtools_dir>/packtools/catalogs/scielo-
↪publishing-schema.xml"/>
    <delegateSystem systemIdStartString="journalpublishing3.dtd"
                    catalog="file://<packtools_dir>/packtools/catalogs/scielo-
↪publishing-schema.xml"/>
    <delegateSystem systemIdStartString="http://jats.nlm.nih.gov/publishing/"
                    catalog="file://<packtools_dir>/packtools/catalogs/scielo-
↪publishing-schema.xml"/>
</catalog>
```

This shell script can help you with the task.

2. Setting the environment variable *XML_CATALOG_FILES* with the absolute path to `<packtools_dir>/packtools/catalogs/scielo-publishing-schema.xml`. This setup can also be made by the main Python program, so for these cases a constant pointing to the catalog file is also provided.

```
import os
from packtools.catalogs import XML_CATALOG
os.environ['XML_CATALOG_FILES'] = XML_CATALOG
```

In some cases where the system's entry-point is a single function, for instance the `main` function, a special helper decorator can be used, as follows:

```
from packtools.utils import config_xml_catalog
@config_xml_catalog
def main():
    """At this point the XML Catalog is configured"""
```

More information at http://xmlsoft.org/catalog.html#Simple

1.2.2 Settings up the logger handler

It is expected that the application using *packtools* defines a logger for *packtools*, e.g.:

```
import logging
logging.getLogger('packtools').addHandler(logging.StreamHandler())
```

See the official docs for more info.

1.2.3 Validation basics

The validation of an XML document is performed through instances of *packtools.XMLValidator*. The easiest way to get an instance is by running *packtools.XMLValidator.parse()*, which in addition to accepting absolute or relative path to file in the local filesystem, URL, etree objects, or file-objects, it also loads the most appropriate validation schemas to the document according to its version.

```
import packtools
xmlvalidator = packtools.XMLValidator.parse('path/to/file.xml')
```

The validation can be performed in two levels: DTD and SciELO Style. To do this, the *packtools.XMLValidator.validate()* and *packtools.XMLValidator.validate_style()* methods are available, respectively. Full validation can be performed with the *packtools.XMLValidator.validate_all()* method. All these methods return a *tuple* comprising the validation status and the errors list.

```
import packtools
xmlvalidator = packtools.XMLValidator.parse('path/to/file.xml')
is_valid, errors = xmlvalidator.validate_all()
```

1.3 Command-line tools

1.3.1 stylechecker

The stylechecker utility performs structural validations on XML documents according to the SciELO PS specification.

Usage:

```
stylechecker [-h] [--annotated | --raw] [--nonetwork]
             [--assetsdir ASSETSDIR] [--version] [--loglevel LOGLEVEL]
             [--nocolors] [--extrasch EXTRASCH] [--sysinfo]
             [file [file ...]]
```

The stylechecker utility validates the contents of *file* or, by default, its standard input, and prints the validation report, encoded in JSON format, to the standard output.

The options are as follows:

```
-h, --help            show this help message and exit
--annotated           reproduces the XML with notes at elements that have
                      errors
--raw                 each result is encoded as json, without any
                      formatting, and written to stdout in a single line.
--nonetwork           prevents the retrieval of the DTD through the network
--assetsdir ASSETSDIR
                      lookup, at the given directory, for each asset
                      referenced by the XML. current working directory will
                      be used by default.
--version             show program's version number and exit
--loglevel LOGLEVEL
--nocolors            prevents the output from being colorized by ANSI
                      escape sequences
--extrasch EXTRASCH   runs an extra validation using an external schematron
                      schema. built-in schemas are available through the
                      prefix `@`: @scielo-br, @sps-1.1, @sps-1.2, @sps-1.3,
                      @sps-1.4, @sps-1.5.
--sysinfo             show program's installation info and exit.
```

Exit status: The stylechecker utility exits 0 on success, and >0 if an error occurs.

CHAPTER 2

API documentation

If you are looking for information about the library internals, this if for you.

2.1 Main interface

2.1.1 Domain-level classes

These are the classes users will more frequently interact with.

class packtools.**XMLValidator**(*file*, *dtd=None*, *style_validators=None*)

Adapter that performs SPS validations.

SPS validation stages are:

- JATS 1.0 or PMC 3.0 (as bound by the doctype declaration or passed explicitly)
- SciELO Style - ISO Schematron
- SciELO Style - Python based pipeline

Parameters

- **file** – etree._ElementTree instance.
- **sps_version** – the version of the SPS that will be the basis for validation.
- **dtd** – (optional) etree.DTD instance. If not provided, we try the external DTD.
- **style_validators** – (optional) list of packtools.domain.SchematronValidator objects.

annotate_errors(*fail_fast=False*)

Add notes on all elements that have errors.

The errors list is generated as the result of calling *validate_all()*.

assets
Lists all static assets referenced by the XML.

lookup_assets(*base*)
Look for each asset in *base*, and returns a list of tuples with the asset name and its presence status.

Parameters **base** – any container that implements membership tests, i.e. it must support the `in` operator.

meta
Article metadata.

classmethod parse(*file*, *no_doctype=False*, *sps_version=None*, *supported_sps_versions=None*, *extra_sch_schemas=None*, ***kwargs*)
Factory of XMLValidator instances.

If *file* is not an etree instance, it will be parsed using *packtools.utils.XML()*.

If the DOCTYPE is declared, its public id is validated against a white list, declared by `ALLOWED_PUBLIC_IDS` module variable. The system id is ignored. By default, the allowed values are:

- SciELO PS >= 1.2: `- -//NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.0 20120330//EN`
- SciELO PS 1.1: `- -//NLM//DTD JATS (Z39.96) Journal Publishing DTD v1.0 20120330//EN -//NLM//DTD Journal Publishing DTD v3.0 20080202//EN`

Parameters

- **file** – Path to the XML file, URL, etree or file-object.
- **no_doctype** – (optional) if missing DOCTYPE declaration is accepted.
- **sps_version** – (optional) force the style validation against a SPS version.
- **supported_sps_versions** – (optional) list of supported versions. the only way to bypass this restriction is by using the arg *sps_version*.
- **extra_sch_schemas** – (optional) list of extra Schematron schemas.

validate(**args*, ***kwargs*)
Validate the source XML against JATS DTD.

Returns a tuple comprising the validation status and the errors list.

validate_all(*fail_fast=False*)
Runs all validations.

First, the XML is validated against the DTD (calling *validate()*). If no DTD is provided and the argument `fail_fast == True`, a `TypeError` is raised. After that, the XML is validated against the SciELO style (calling *validate_style()*).

Parameters **fail_fast** – (optional) raise `TypeError` if the DTD has not been loaded.

validate_style(**args*, ***kwargs*)
Validate the source XML against SPS-Style Tagging guidelines.

Returns a tuple comprising the validation status and the errors list.

class packtools.**HTMLGenerator**(*file*, *xslt=None*, *css=None*, *print_css=None*, *js=None*, *permlink=None*, *url_article_page=None*, *url_download_ris=None*)
Adapter that generates HTML from SPS XML.

Basic usage:

```
from lxml import etree

xml = etree.parse('valid-sps-file.xml')
generator = HTMLGenerator(xml)

html = generator.generate('pt')
html_string = etree.tostring(html, encoding='unicode', method='html')
```

Parameters

- **file** – etree._ElementTree instance.
- **xslt** – (optional) etree.XSLT instance. If not provided, the default XSLT is used.
- **css** – (optional) URI for a CSS file.

generate(*lang*)
Generates the HTML in the language `lang`.

Parameters **lang** – 2-digit ISO 639-1 text string.

language
The language of the main document.

languages
The language of the main document plus all translations.

classmethod parse(*file*, *valid_only=True*, ***kwargs*)
Factory of HTMLGenerator instances.

If *file* is not an etree instance, it will be parsed using `XML()`.

Parameters

- **file** – Path to the XML file, URL, etree or file-object.
- **valid_only** – (optional) prevents the generation of HTML for invalid XMLs.

2.1.2 Utils

packtools.utils.**XML**(*file*, *no_network=True*, *load_dtd=True*)
Parses *file* to produce an etree instance.

The XML can be retrieved given its filesystem path, an URL or a file-object.

Parameters

- **file** – Path to the XML file, URL or file-object.
- **no_network** – (optional) prevent network access for external DTD.
- **load_dtd** – (optional) load DTD during parse-time.

class packtools.utils.**Xray**(*zip_file*)
Zip-file introspector.

Parameters **zip_file** – instance of `zipfile.ZipFile`.

close()
Close the archive file.

get_file(*member*, *mode='r'*)
Get file object for member.

A complete list of members can be checked calling *show_members()*.

Parameters **member** – a zip member, e.g. 'foo.xml'

show_members()
Shows the package members.

show_sorted_members()
Shows the package members sorted by their file extensions.

packtools.utils.**cachedmethod**(*wrappee*)
Caches method calls within known arguments.

packtools.utils.**config_xml_catalog**(*wrapped*)
Decorator that wraps the execution of a function, setting-up and tearing-down the XML_CATALOG_FILES environment variable for the current process.

```
@config_xml_catalog
def main(xml_filepath):
    xml = XMLValidator(xml_filepath)
    # do some work here
```

packtools.utils.**flatten**(*paths*)
Produces absolute path for each path in paths.

Glob expansions are allowed.

Parameters **paths** – Collection of paths. A path can be relative, absolute or a glob expression.

packtools.utils.**get_schematron_from_buffer**(*buff*, *parser=<XMLParser object>*)
Returns an isoschematron.Schematron for buff.

The default parser doesn't collect ids on a hash table, i.e.: collect_ids=False.

packtools.utils.**get_static_assets**(*xml_et*)
Returns an iterable with all static assets referenced by xml_et.

packtools.utils.**normalize_string**(*unistr*)
Return the NFKC form for the unicode string unistr.

The normal form KD (NFKD) will apply the compatibility decomposition, i.e. replace all compatibility characters with their equivalents, followed by the canonical composition.

packtools.utils.**prettify**(*jsonobj*, *colorize=True*)
Serialize and prettify a Python object as JSON.

On windows, bypass pygments colorization.

Function copied from Circus process manager: https://github.com/circus-tent/circus/blob/master/circus/circusctl.py

packtools.utils.**resolve_schematron_filepath**(*value*)
Determine the filepath for value.

The lookup is run against all known schemas from packtools.catalog.SCH_SCHEMAS. If value is already a filepath, than it is returned as it is.

packtools.utils.**setdefault**(*object*, *attribute*, *producer*)
Like dict().setdefault but for object attributes.

CHAPTER 3

Bug-reports and feedback

- Bugs should be reported at: https://github.com/scieloorg/packtools/issues
- Join us on the IRC: Freenode, channel `#scielo`
- Mailing-list for any feedback or question: https://groups.google.com/forum/#!forum/scielo-dev

Python Module Index

p

Index