untangle Documentation

Release 1.1.1

Christian Stefanescu

Jul 01, 2022

Contents

untangle is a tiny Python library which converts an XML document to a Python object. It is available under the MIT license.

Contents

CHAPTER 1

Usage

untangle has a very simple API. You just need to call the parse function to get back a Python object. The parameter can be:

- a string
- a filename
- a URL

`untangle.`**`parse`**(*filename*, ***parser_features*)

Interprets the given string as a filename, URL or XML data string, parses it and returns a Python object which represents the given document.

Extra arguments to this function are treated as feature values to pass to `parser.setFeature()`. For example, `feature_external_ges=False` will set `xml.sax.handler.feature_external_ges` to False, disabling the parser's inclusion of external general (text) entities such as DTDs.

Raises `ValueError` if the first argument is None / empty string.

Raises `AttributeError` if a requested xml.sax feature is not found in `xml.sax.handler`.

Raises `xml.sax.SAXParseException` if something goes wrong during parsing.

If you are looking for information on a specific function, class or method, this part of the documentation is for you.

The object you get back represents the complete XML document. Child elements can be accessed with `parent.child`, attributes with `element['attribute']`. Siblings with similar names are grouped into a list.

CHAPTER 2

Example

Considering this XML document:

```
<?xml version="1.0"?>
<root>
    <child name="child1"/>
</root>
```

and assuming it's available in a variable called *xml*, we could use untangle like this:

```
doc = untangle.parse(xml)
child_name = doc.root.child['name'] # 'child1'
```

For text/data inbetween tags, this is described as cdata. After specifying the relevant element as explained above, the data/cdata can be accessed by adding ".cdata" (without the quotes) to the end of your dictionary call.

For more examples, have a look at (and launch) examples.py.

CHAPTER 3

Installation

It is recommended to use pip, which will always download the latest stable release:

```
pip install untangle
```

untangle works with Python versions 2.6, 2.7, 3.3, 3.4, 3.5, 3.6 and pypy

CHAPTER 4

Motivation

untangle is available for that use case, where you have a 20-line XML file you got back from an API and you just need to extract some values out of it. You might not want to use regular expressions, but just as well you might not want to install a complex libxml2-based solution (and look up its terse API).

Performance and memory usage might be bad, but these tradeoffs were made in order to allow a simple API and no external dependencies. See also: *Limitations*.

CHAPTER 5

Limitations

untangle trades features for a simple API, which is why untangle substitutes `-`, `.` and `:` with `_`:

- `<foobar><foo-bar/></foobar>` can be accessed with `foobar.foo_bar`
- `<foo.bar.baz/>` can be accessed with `foo_bar_baz`
- `<foo:bar><foo:baz/></foo:bar>` can be accessed with `foo_bar.foo_baz`

CHAPTER 6

Encoding

Be aware that with certain characters or maybe also depending on the python version you might get an error on accessing specific attributes, such as `UnicodeEncodeError: 'ascii' codec can't encode character u'\xfc' in position 385: ordinal not in range(128)` In most cases it should be enough to import the sys module, and set utf-8 as encoding, with:

```
import sys
reload(sys) # just to be sure
sys.setdefaultencoding('utf-8')
```

CHAPTER 7

SAX features

It is possible to pass specific SAX features to the handler used by untangle, for instance:

```
untangle.parse(my_xml, feature_external_ges=False)
```

This will toggle the SAX handler feature described here.

CHAPTER 8

Changelog

see https://github.com/stchris/untangle/blob/master/CHANGELOG.md

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

u

Index

P

U