json2xml Documentation

Release 4.1.0

Vinit Kumar

Jan 12, 2024

CONTENTS:

CHAPTER

ONE

JSON2XML

json2xml is a Python library that allows you to convert JSON data into XML format. It's simple, efficient, and easy to use.

Documentation: https://json2xml.readthedocs.io.

The library was initially dependent on the *dict2xml* project, but it has now been integrated into json2xml itself. This has led to cleaner code, the addition of types and tests, and overall improved performance.

1.1 Features

json2xml supports the following features:

- Conversion from a *json* string to XML
- Conversion from a *json* file to XML
- Conversion from an API that emits *json* data to XML

1.2 Usage

You can use the json2xml library in the following ways:

```
from json2xml import json2xml
from json2xml.utils import readfromurl, readfromstring, readfromjson

# Convert JSON data from a URL to XML
data = readfromurl("https://api.publicapis.org/entries")
print(json2xml.Json2xml(data).to_xml())

# Convert a JSON string to XML
data = readfromstring(
    '{"login":"mojombo","id":1,"avatar_url":"https://avatars0.githubusercontent.com/u/1?
↪v=4"}'
)
```

(continues on next page)

(continued from previous page)

```
print(json2xml.Json2xml(data).to_xml())

# Convert a JSON file to XML
data = readfromjson("examples/licht.json")
print(json2xml.Json2xml(data).to_xml())
```

1.3 Custom Wrappers and Indentation

By default, a wrapper *all* and pretty *True* is set. However, you can easily change this in your code like this:

```
from json2xml import json2xml
from json2xml.utils import readfromurl, readfromstring, readfromjson

data = readfromstring(
    '{"login":"mojombo","id":1,"avatar_url":"https://avatars0.githubusercontent.com/u/1?
↪v=4"}'
)
print(json2xml.Json2xml(data, wrapper="all", pretty=True).to_xml())
```

Outputs this:

```
<?xml version="1.0" ?>
<all>
  <login type="str">mojombo</login>
  <id type="int">1</id>
  <avatar_url type="str">https://avatars0.githubusercontent.com/u/1?v=4</avatar_url>
</all>
```

1.4 Omit List item

Assume the following json input

```
{
  "my_items": [
    { "my_item": { "id": 1 } },
    { "my_item": { "id": 2 } }
  ],
  "my_str_items": ["a", "b"]
}
```

By default, items in an array are wrapped in <item></item>.

Default output:

```
<?xml version="1.0" ?>
<all>
  <my_items type="list">
    <item type="dict">
      <my_item type="dict">
```

(continues on next page)

(continued from previous page)

```
        <id type="int">1</id>
      </my_item>
    </item>
    <item type="dict">
      <my_item type="dict">
        <id type="int">2</id>
      </my_item>
    </item>
  </my_items>
  <my_str_items type="list">
    <item type="str">a</item>
    <item type="str">b</item>
  </my_str_items>
  <empty type="list"/>
</all>
```

However, you can change this behavior using the item_wrap property like this:

```
from json2xml import json2xml
from json2xml.utils import readfromurl, readfromstring, readfromjson

data = readfromstring('{"my_items":[{"my_item":{"id":1} },{"my_item":{"id":2} }],"my_str_
↪items":["a","b"]}')
print(json2xml.Json2xml(data, item_wrap=False).to_xml())
```

Outputs this:

```
<?xml version="1.0" ?>
<all>
  <my_items type="list">
    <my_item type="dict">
      <id type="int">1</id>
    </my_item>
    <my_item type="dict">
      <id type="int">2</id>
    </my_item>
  </my_items>
  <my_str_items type="str">a</my_str_items>
  <my_str_items type="str">b</my_str_items>
</all>
```

1.5 Optional Attribute Type Support

You can also specify if the output XML needs to have type specified or not. Here is the usage:

```
from json2xml import json2xml
from json2xml.utils import readfromurl, readfromstring, readfromjson

data = readfromstring(
    '{"login":"mojombo","id":1,"avatar_url":"https://avatars0.
```

(continues on next page)

(continued from previous page)

```
↪githubusercontent.com/u/1?v=4"}'
)
print(json2xml.Json2xml(data, wrapper="all", pretty=True, attr_type=False).to_
↪xml())
```

Outputs this:

```
<?xml version="1.0" ?>
<all>
  <login>mojombo</login>
  <id>1</id>
  <avatar_url>https://avatars0.githubusercontent.com/u/1?v=4</avatar_url>
</all>
```

The methods are simple and easy to use and there are also checks inside of code to exit cleanly in case any of the input(file, string or API URL) returns invalid JSON.

1.6 How to run tests

This is provided by pytest, which is straight forward.

```
virtualenv venv -p $(which python3.9)
pip install -r requirements-dev.txt
python setup.py install
pytest -vv
```

1.7 Help and Support to maintain this project

- You can sponsor my work for this plugin here: https://github.com/sponsors/vinitkumar/

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install json2xml, run this command in your terminal:

```
$ pip install json2xml
```

This is the preferred method to install json2xml, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for json2xml can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/vinitkumar/json2xml
```

Or download the tarball:

```
$ curl  -OL https://github.com/vinitkumar/json2xml/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER

THREE

USAGE

To use json2xml in a project:

```
import json2xml
```

CHAPTER

FOUR

JSON2XML

4.1 json2xml package

4.1.1 Submodules

4.1.2 json2xml.dicttoxml module

`json2xml.dicttoxml.convert`(*obj: str | int | float | bool | Number | Sequence | datetime | date | None | Dict[str, Any], ids: Any, attr_type: bool, item_func: Callable[[str], str], cdata: bool, item_wrap: bool, parent: str = 'root', list_headers: bool = False*) → str

Routes the elements of an object to the right function to convert them based on their data type

`json2xml.dicttoxml.convert_bool`(*key: str, val: bool, attr_type: bool, attr: dict[str, Any] = {}, cdata: bool = False*) → str

Converts a boolean into an XML element

`json2xml.dicttoxml.convert_dict`(*obj: dict[str, Any], ids: list[str], parent: str, attr_type: bool, item_func: Callable[[str], str], cdata: bool, item_wrap: bool, list_headers: bool = False*) → str

Converts a dict into an XML string.

`json2xml.dicttoxml.convert_kv`(*key: str, val: str | Number, attr_type: bool, attr: dict[str, Any] = {}, cdata: bool = False*) → str

Converts a number or string into an XML element

`json2xml.dicttoxml.convert_list`(*items: Sequence[Any], ids: list[str] | None, parent: str, attr_type: bool, item_func: Callable[[str], str], cdata: bool, item_wrap: bool, list_headers: bool = False*) → str

Converts a list into an XML string.

`json2xml.dicttoxml.convert_none`(*key: str, attr_type: bool, attr: dict[str, Any] = {}, cdata: bool = False*) → str

Converts a null value into an XML element

`json2xml.dicttoxml.default_item_func`(*parent: str*) → str

`json2xml.dicttoxml.dict2xml_str`(*attr_type: bool, attr: dict[str, Any], item: dict[str, Any], item_func: Callable[[str], str], cdata: bool, item_name: str, item_wrap: bool, parentIsList: bool, parent: str = '', list_headers: bool = False*) → str

parse dict2xml

`json2xml.dicttoxml.dicttoxml`(*obj: dict[str, ~typing.Any], root: bool = True, custom_root: str = 'root', ids: list[int] | None = None, attr_type: bool = True, item_wrap: bool = True, item_func: ~collections.abc.Callable[[str], str] = <function default_item_func>, cdata: bool = False, xml_namespaces: dict[str, ~typing.Any] = {}, list_headers: bool = False*) → bytes

Converts a python object into XML.

Parameters

- **obj** (*dict*) – dictionary
- **root** (*bool*) – Default is True specifies wheter the output is wrapped in an XML root element
- **custom_root** – Default is 'root' allows you to specify a custom root element.
- **ids** (*bool*) – Default is False specifies whether elements get unique ids.
- **attr_type** (*bool*) – Default is True specifies whether elements get a data type attribute.
- **item_wrap** (*bool*) – Default is True specifies whether to nest items in array in <item/> Example if True

  ..code-block:: python

  data = {'bike': ['blue', 'green']}

  ```
  <bike>
  <item>blue</item>
  <item>green</item>
  </bike>
  ```

  Example if False

  ..code-block:: python

  data = {'bike': ['blue', 'green']}

  ..code-block:: xml

  <bike>blue</bike> <bike>green</bike>'

- **item_func** – items in a list. Default is 'item' specifies what function should generate the element name for
- **cdata** (*bool*) – Default is False specifies whether string values should be wrapped in CDATA sections.
- **xml_namespaces** – is a dictionary where key is xmlns prefix and value the urn, Default is {}. Example:

  ```
  { 'flex': 'http://www.w3.org/flex/flexBase', 'xsl': "http://www.w3.
  ↪org/1999/XSL/Transform"}
  ```

  results in

  ```
  <root xmlns:xsl="http://www.w3.org/1999/XSL/Transform" xmlns:flex=
  ↪"http://www.w3.org/flex/flexBase">
  ```

- **list_headers** (*bool*) – Default is False Repeats the header for every element in a list. Example if True:

```
"Bike": [
{'frame_color': 'red'},
{'frame_color': 'green'}
]}
```

results in

```
<Bike><frame_color>red</frame_color></Bike>
<Bike><frame_color>green</frame_color></Bike>
```

Dictionaries-keys with special char '@' has special meaning: @attrs: This allows custom xml attributes:

```
{'@attr':{'a':'b'}, 'x':'y'}
```

results in

```
<root a="b"><x>y</x></root>
```

@flat: If a key ends with @flat (or dict contains key '@flat'), encapsulating node is omitted. Similar to item_wrap. @val: @attrs requires complex dict type. If primitive type should be used, then @val is used as key. To add custom xml-attributes on a list {'list': [4, 5, 6]}, you do this:

```
{'list': {'@attrs': {'a':'b','c':'d'}, '@val': [4, 5, 6]}
```

which results in

```
<list a="b" c="d"><item>4</item><item>5</item><item>6</item></list>
```

`json2xml.dicttoxml.escape_xml`(*s: str | Number*) → str

Escape a string for use in XML.

Args:
s (str | numbers.Number): The string to escape.

Returns:
str: The escaped string.

`json2xml.dicttoxml.get_unique_id`(*element: str*) → str

Generate a unique ID for a given element.

Args:
element (str): The element to generate an ID for.

Returns:
str: The unique ID.

`json2xml.dicttoxml.get_xml_type`(*val: str | int | float | bool | Number | Sequence | datetime | date | None | Dict[str, Any]*) → str

Get the XML type of a given value.

Args:
val (ELEMENT): The value to get the type of.

Returns:
str: The XML type.

`json2xml.dicttoxml.is_primitive_type`(*val: Any*) → bool

`json2xml.dicttoxml.key_is_valid_xml(key: str) → bool`

Check if a key is a valid XML name.

Args:
key (str): The key to check.

Returns:
bool: True if the key is a valid XML name, False otherwise.

`json2xml.dicttoxml.list2xml_str(attr_type: bool, attr: dict[str, Any], item: Sequence[Any], item_func: Callable[[str], str], cdata: bool, item_name: str, item_wrap: bool, list_headers: bool = False) → str`

`json2xml.dicttoxml.make_attrstring(attr: dict[str, Any]) → str`

Create a string of XML attributes from a dictionary.

Args:
attr (dict[str, Any]): The dictionary of attributes.

Returns:
str: The string of XML attributes.

`json2xml.dicttoxml.make_id(element: str, start: int = 100000, end: int = 999999) → str`

Generate a random ID for a given element.

Args:
element (str): The element to generate an ID for. start (int, optional): The lower bound for the random number. Defaults to 100000. end (int, optional): The upper bound for the random number. Defaults to 999999.

Returns:
str: The generated ID.

`json2xml.dicttoxml.make_valid_xml_name(key: str, attr: dict[str, Any]) → tuple[str, dict[str, Any]]`

Tests an XML name and fixes it if invalid

`json2xml.dicttoxml.wrap_cdata(s: str | Number) → str`

Wraps a string into CDATA sections

4.1.3 json2xml.json2xml module

class `json2xml.json2xml.Json2xml(data: Dict[str, Any] | None = None, wrapper: str = 'all', root: bool = True, pretty: bool = True, attr_type: bool = True, item_wrap: bool = True)`

Bases: `object`

Wrapper class to convert the data to xml

`to_xml() → Any | None`

Convert to xml using dicttoxml.dicttoxml and then pretty print it.

4.1.4 json2xml.utils module

Utility methods for converting XML data to dictionary from various sources.

exception `json2xml.utils.`**`InvalidDataError`**

Bases: `Exception`

Raised when the data is invalid.

exception `json2xml.utils.`**`JSONReadError`**

Bases: `Exception`

Raised when there is an error reading JSON data.

exception `json2xml.utils.`**`StringReadError`**

Bases: `Exception`

Raised when there is an error reading from a string.

exception `json2xml.utils.`**`URLReadError`**

Bases: `Exception`

Raised when there is an error reading from a URL.

`json2xml.utils.`**`readfromjson`**(*filename: str*) → Dict[str, str]

Reads a JSON file and returns a dictionary.

`json2xml.utils.`**`readfromstring`**(*jsondata: str*) → Dict[str, str]

Loads JSON data from a string and returns a dictionary.

`json2xml.utils.`**`readfromurl`**(*url: str*, *params: Dict[str, str] | None = None*) → Dict[str, str]

Loads JSON data from a URL and returns a dictionary.

4.1.5 Module contents

Top-level package for json2xml.

CHAPTER

FIVE

CONTRIBUTING

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/vinitkumar/json2xml/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

5.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

5.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

5.1.4 Write Documentation

json2xml could always use more documentation, whether as part of the official json2xml docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/vinitkumar/json2xml/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

5.2 Get Started!

Ready to contribute? Here's how to set up *json2xml* for local development.

1. Fork the *json2xml* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/json2xml.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv json2xml
$ cd json2xml/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 json2xml tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

5.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for 3.7+, and for PyPy. Make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_json2xml
```

5.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER

SIX

CREDITS

6.1 Development Lead

- Vinit Kumar <mail@vinitkumar.me>

6.2 Contributors

Valueable contributions were made by these contributors.

https://github.com/vinitkumar/json2xml/graphs/contributors

CHAPTER

SEVEN

HISTORY

CHAPTER EIGHT

4.1.0 / 2023-11-07

- chore(deps): bump urllib3 from 1.26.13 to 1.26.18 (#192, #195)

CHAPTER NINE

4.0.1 / 2023-10-13

- feat: get local working for 3.12 and fix some code quality issue
- Fix/make docs better (#186)
- fix: issue with hardcoded year
- feat: upgrade sphinx theme
- fix: issues with ruff linting, use isinstance in place of type
- fix: make readme better
- chore(deps): bump tornado from 6.3.2 to 6.3.3 in /docs (#185)
- chore(deps): bump certifi from 2022.12.7 to 2023.7.22 in /docs (#183)
- chore(deps): bump pygments from 2.10.0 to 2.15.0 in /docs (#182)

CHAPTER

TEN

4.0.0 / 2023-07-19

- Updated to Python 3.12 beta 4
- Added check for pypy 3.10
- Dropped support for Python 3.7
- Updated fixture for attrs with dict and removed comment (#170)

CHAPTER

ELEVEN

3.21.0 / 2023-01-18

- Implemented list item with attributes (#169)
- Added python3.12 alpha4 to the fix (#168)
- Replaced requests with urllib3 (#167)
- Added security reporting guidelines
- Added CodeQL workflow for GitHub code scanning (#160)
- Set default to False for list_headers (#164)
- Fixed ci issue due to mypy update on python3.11 (#166)
- Updated certifi from 2021.10.8 to 2022.12.7 in /docs (#162)
- Fixed file opening issue
- Celebrated release of Python 3.11 (#159)

CHAPTER TWELVE

3.20.0 / 2022-10-16

- Made dependencies more flexible (#158)
- Used SystemRandom for secure random integer generation (#156)

CHAPTER

THIRTEEN

3.19.5 / 2022-09-18

- Fixed issues #138 and #151, added 2 new unit tests (#154)
- Fixed unit tests for #152 (#153)

CHAPTER

FOURTEEN

3.19.4 / 2022-07-24

- Transitioned from unittest to pytest (#149)
- Upgraded python test to python311beta (#148)
- Tested new version of 3.10 and 3.11 (#147)

CHAPTER

FIFTEEN

3.19.3 / 2022-07-01

- Added UTF-8 encoding type to @readfromjson function in utils.py for Korean language support (#145)

CHAPTER

SIXTEEN

3.19.2 / 2022-06-09

- Escaped xml char when having attrs (#144)
- Adjusted pytest config setting for easier logging
- Bumped python 3.10 and 3.11 version (#142)

CHAPTER

SEVENTEEN

3.19.1 / 2022-06-05

- Bumped version of docs building
- Updated waitress from 2.1.1 to 2.1.2 in /docs (#141)
- Updated docs for dicttoxml (#140)

CHAPTER

EIGHTEEN

3.19.0 / 2022-05-20

- Set xsi location (#135)
- Repeated list headers (#138)
- Added support for python3.11 (#139)
- Improved docs for dicttoxml (#134)
- Fixed types working check for ci mypy (#133)
- Added mypy support to ci (#132)
- Generated changelog
- Removed logging by default (#131)
- Merged two dev requirements files (#129)
- Removed old unused config
- Added types (#125)
- Fixed issue with twine check
- Fixed issue with long description
- Refactored: xmltodict is only test dependency now (#124)
- Added correct list of contributors
- Generated changelog
- Improved dicttoxml (#121)
- Added correct badge
- Started using codecov
- Fixed flake8 tests
- Added coverage to the mix
- Fixed lint issues and CI
- Checked new CI stuff like lint and coverage
- Bumped version and generated changelog
- Fixed issue with wrong output for boolean list
- Made pull requests trigger action runs

CHAPTER

NINETEEN

3.18.0 / 2022-04-23

- Bumped version
- Improved dicttoxml (#121)
- Added correct badge
- Started using codecov
- Fixed flake8 tests
- Added coverage to the mix
- Fixed lint issues and CI
- Checked new CI stuff like lint and coverage
- Bumped version and generated changelog
- Fixed issue with wrong output for boolean list
- Made pull requests trigger action runs

CHAPTER TWENTY

3.17.1 / 2022-04-20

- Fixed issue with wrong output for boolean list
- Made pull requests trigger action runs

CHAPTER

TWENTYONE

3.17.0 / 2022-04-18

- Fixed return of correct xml type for bool (#119)
- Added download counter
- Checked latest alpha (#116)
- Checked latest alpha (#115)
- Updated waitress from 2.0.0 to 2.1.1 in /docs (#114)
- Only python3 wheels are created now

CHAPTER

TWENTYTWO

3.15.0 / 2022-02-24

- Merged remote-tracking branch 'origin/master'
- Bumped version and prepared for new release
- Added new python versions to test against (#110)
- Fixed perflint (#109)
- Supported latest version of 3.10 and 3.11 alpha3 (#98)
- Generated changelog
- Removed unused imports
- Bumped version
- Fixed issue with uncaught UnicodeDecodeError
- Cancelled jobs for concurrent builds in same PR
- Stabilized pypi
- Updated tox config

CHAPTER

TWENTYTHREE

V3.14.0 / 2022-02-10

- Removed unused imports
- Bumped version
- Fixed issue with uncaught UnicodeDecodeError
- fix: remove unused imports
- bump version
- fix: issue with uncaught UnicodeDecodeError
- cancel jobs for concurrent builds in same PR
- pypi is stable now
- feat: update tox config

CHAPTER

TWENTYFOUR

V3.11.0 / 2022-01-31

- bump version
- feat: remove comments
- Feat: install pytest separately and run pytests now
- fix tox
- add some documentation on testing
- split testing libs away from release
- fix: update changelog
- bump version to 3.10.0
- fix: we support Python3.7+ now (#101)
- Issue: #99 dicttoxml igores the root param (#100)

CHAPTER

TWENTYFIVE

V3.10.0 / 2022-01-29

- bump version to 3.10.0
- fix: we support Python3.7+ now (#101)
- Issue: #99 dicttoxml igores the root param (#100)
- feat: bump to a rc1 version
- Add support for Python3.11 alpha and upgrade pytest and py (#97)
- Feat: drop 3.11.0 alphas from the test matrix for now
- feat: find the versions that are in the CI
- fix: typo in the name of python 3.11 version
- sunsetting python 3.6 and add support for python3.11 alpha
- chore: prepare for release 3.9.0
- fix email
- fix readme
- - update readme - add tests - refactor
- resolve #93
- chore: run black on readme doc
- fix: more issues
- fix: garbage in history
- feat: generate history

CHAPTER

TWENTYSIX

V3.9.0 / 2021-12-19

- feat: generate history
- feat: item_wrap for str and int (#93)

CHAPTER TWENTYSEVEN

V3.8.4 / 2021-10-24

- bump version
- fix: version bump and readme generator

CHAPTER

TWENTYEIGHT

V3.8.3 / 2021-10-24

- bump version
- feat: reproduce the error in the test (#90)
- Feat/version (#88)
- Feat/docs theme change (#87)
- Feat/docs theme change (#86)
- Feat/docs theme change (#85)
- Feat/docs theme change (#84)
- Feat/docs theme change (#83)
- feat: update the docs theme (#82)

CHAPTER

TWENTYNINE

V3.8.0 / 2021-10-07

- Feat/security improvements (#81)
- **arrow_up**
  feat: python 3.10 released (#79)

CHAPTER THIRTY

V3.7.0 / 2021-09-11

- **bookmark**
  feat: final release for v3.7.0
- **bookmark**
  feat: bump version

CHAPTER

THIRTYONE

V3.7.0BETA2 / 2021-09-10

- Feat/cleanup and deprecation fix (#78)
- item ommision (#76)
- Create FUNDING.yml

CHAPTER

THIRTYTWO

V3.7.0BETA1 / 2021-08-28

- Feat/fork and update dict2xml (#75)
- chore(deps-dev): bump pip from 18.1 to 19.2 (#73)
- Delete .travis.yml
- chore(deps-dev): bump lxml from 4.6.2 to 4.6.3 (#68)
- Bump lxml from 4.1.1 to 4.6.2 (#66)

CHAPTER

THIRTYTHREE

V3.6.0 / 2020-11-12

- Feat/wip exceptions (#65)
- Add .deepsource.toml
- feat: upgrade the actions
- feat: try & support more os and python versions
- Update pythonpackage.yml

CHAPTER

THIRTYFOUR

V3.5.0 / 2020-08-24

- feat: remove six as dependency as we are python3 only, resolves #60 (#61)
- feat: update makefile for the correct command

CHAPTER

THIRTYFIVE

V3.4.1 / 2020-06-10

- fix: issues with pypi release and bump version
- Feat/attr type docs (#58)
- fix: conflicts
- Feat/attr type docs (#57)
- Merge github.com:vinitkumar/json2xml
- Update json2xml.py (#56)
- Merge github.com:vinitkumar/json2xml
- feat: fix typo in the readme

CHAPTER

THIRTYSIX

V3.3.3 / 2020-02-05

- Update README.rst
- fix: issue with pypi uploads
- fix: version
- bump version
- Update pythonpackage.yml
- Refactor/prospector cleanup (#50)
- Update pythonpackage.yml
- Create pythonpackage.yml
- Update README.rst
- fix: typo in readme
- bump version
- Feature/attribute support (#48)
- Feature/attribute support (#47)
- chore: bump version
- fix: remove print statement in json read because it confuses people
- fix typo in readme

CHAPTER

THIRTYSEVEN

V3.0.0 / 2019-02-26

- Fix/coveralls (#43)
- update coverage report (#42)
- Merge pull request #41 from vinitkumar/fix/coveralls
- add python coveralls
- Merge pull request #40 from vinitkumar/refactor/cookiecutter
- update coverage
- add image for coveralls
- coverage and coveralls integrations
- try and trigger coveralls too
- fix code block in readme
- add doc about custom wrapper
- try at reducing the dependencies
- add tests for custom wrappers as well
- add tests for actualy dict2xml conversion
- fix: remove missing import
- fix: code syntax highlight in the readme again
- fix: code syntax highlight in the readme again
- fix: code syntax highlight in the readme
- chore: update readme with code samples
- test: add testcases for the different utils method
- remove unused imports
- check the third method for generating dict from json string too
- run correct test files
- fix tests
- update requirements and setuptools
- refactor the module into more maintainable code
- chore: add boilerplate

- remove all legacy
- Fix/cleanup (#38)
- cleanup: remove unused modules (#37)
- Merge pull request #35 from vinitkumar/improve-structure
- cleanup
- one again try to get the build working
- travis need full version for latest supported python
- do not hardcode version in a series
- update grammar
- fix conflicts
- Update LICENSE
- cleanup readme
- remove cli
- some cleanup and update the tests
- Update readme.md
- Cleanup Readme.md
- Update issue templates
- fix vulnerabilities in requests

CHAPTER

THIRTYEIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

j

INDEX