tpl

Release 0.9.1

Mar 29, 2021

Contents

You want to fill data into a template file?

```
tpl --yaml data.yaml template.file > rendered.file
```

You have everything already set up in your environment and now you just want to POST it somewhere?

```
tpl structure.json \
  | curl \
      -X POST \
      -H "Content-Type: application/json" \
      -d@- \
      httpbin.org/anything
```

You want to fill in a template in your CD pipeline and have access to docker?

```
echo "My go-to editor is {{VISUAL}} on {{OS}}" \
  | docker run --rm -i -e "VISUAL" -e "OS=$(uname)" m3t0r/tpl -
```

CHAPTER 1

Installation

`pip install tpl`, `docker pull M3t0r/tpl` or `make install`

CHAPTER 2

Input sources

***tpl* supports multiple sources:**

- YAML files (`--yaml <file>`)
- JSON files (`--json <file>`)
- environment variables (`--environment`)

You can specify multiple sources at once, but if a key is present in more than one then it's value will be taken from the latter source. This can be useful if you have default values that you want to always be present:

```
tpl \
  --yaml defaults.yaml \
  --json <(curl -H "Content-Type: application/json" now.httpbin.org) \
  template.jinja2 > results.html
```

CHAPTER 3

Usage

```
Usage:
  tpl [options] <template_file>
  tpl --help
  tpl --version


tpl uses the Jinja2 templating engine to render it's output. You can find the
documentation for template designers at:
    http://jinja.pocoo.org/docs/latest/templates/

If you provide multiple data sources they will be merged together. If a key is
present in more than one source the value of the source that was specified
last will be used. Nested objects will be merged with the same algorithm.

Options:
  -e, --environment    Use all environment variables as data
  --json=<file>        Load JSON data from a file or STDIN
  --yaml=<file>        Load YAML data from a file or STDIN
```

CHAPTER 4

Documentation contents

4.1 `tpl`: render templates with data from various sources

4.1.1 Synopsis

tpl [*options*] <*template_file*> [*output_file*]
tpl *-h* | *--help*
tpl *-v* | *--version*

4.1.2 Description

tpl renders a Jinja2 template file with data aggregated from one or more data sources specified via options and writes the result to *output_file*. It is meant to be easily composable with other unix tools like **xargs**, **curl**, and **jq**.

template_file
: The template file that will be rendered. A *template_file* of "-" stands for STDIN.

output_file
: The file that the rendered template will be written to. If an error occurs during templating the output might end up with incomplete and broken data. When a file with the same name already exists it will be overwritten without notice. If ommitted this argument defaults to "-" which stands for STDOUT.

4.1.3 Options

The order of data source options is important. See the *Data Merging* section for more information.

-h, --help
: Print a help message to STDERR and exit successfully.

-v, --version
: Write the version number to STDOUT and exit successfully.

-e, --environment
Load environmant variables as key-value pairs into the context. This allows you to access, for example, *$PATH* with `{{ PATH }}`.

If no other data source option was specified this option is used by default. Templates can only access the environment if no other data sources were specified or this flag is used. This is to prohibit leaking of secrets from the environment.

--json <file>
Load data from a JSON file into the context. Unlike **jq**, **tpl** does not support multiple JSON objects separated by whitespaces. Internally this uses Python's `json.load()`.

--yaml <file>
Load data from a YAML file into the context. The YAML file can only contain one document. If the parser encounters a second document **tpl** will abort with an error. This data source uses the PyYAML library.

4.1.4 Data Merging

If you provide multiple data sources they will be merged together to provide a context for the Jinja2 engine. If a key is present in more than one source the value of the source that was specified last will be used. Nested objects will be merged with the same algorithm.

Special treatment is given to root objects of every data source when merging: If the root object is a list, it's elements will be added to the end of `_array_data`. If the root object is a a scalar value, like a string, boolean, or number, it's value will be stored in `_scalar_data`. When one of these special behaviours is triggered the already assembled context is not cleared of previously defined key-value pairs:

```
$ tpl \
>     --json <(echo '"the answer"') \
>     --json <(echo '{"foo":"bar"}') \
>     --json <(echo "42") \
>     <(echo 'scalar: {{ _scalar_data }}, foo: {{ foo }}')
scalar: 42, foo: bar
# and not scalar: 42, foo:
```

Although **tpl** is primarily developed as a CLI program there is some documentation of it's internals in the *Python API* section.

4.2 Python API

tpl.**merge_data**(*old: dict*, *new*, *array_key='_array_data'*, *scalar_key='_scalar_data'*)
Merge the data from the different sources.

If the new value is a list it's elements will get appended to the list in _array_data.

If the new value is a scalar (anything not a list or dict) it will replace the value in _scalar_data.

if the new value is a dict it's elements will get merged with the elements already present. This also means that sub dicts in both values will get merged.

4.3 License

tpl is licensed under the MIT license. The license text from the `LICENSE` file is repeated below.

4.3.1 MIT license text

```
MIT License

Copyright (c) 2018 Simon Lutz Brüggen

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

4.4 Changelog

4.4.1 Unreleased

No changes yet.

4.4.2 v0.9

Released on 2019-02-21

- Enabled Loop Controls and Expression Statement extensions.
- Added documentation with ReadTheDocs integration and a manpage. This includes a fix for #8.
- Added changelog with old releases.
- Include auxilliary files in the source distribution generated by setuptools.
- Reformatted README.

4.4.3 v0.8

Released on 2018-10-27

- Added support for Python 3.5.
- Added link to Jinja2 template designer documentation in `--help` output thanks to @loudambiance.
- Changed a decent amount of building code with help from @mre.
- Added CI with travis-ci.

- Use Pipfile instead of requirements.txt for the development environment.
- Fixed `-v` option in Docker image.

4.4.4 v0.7

Released on 2018-07-17

- Instead of simply updating only the top level dict we now merge dictionaries on all levels.
- Added support for lists and scalar values a root level of data sources.
- Added automated builds for Dockerhub

4.4.5 v0.6.2

Released on 2018-07-16

- First release to PyPI.

4.4.6 v0.6.1

Released on 2018-07-16

- Fix release target in Makefile.

4.4.7 v0.6

Released on 2018-07-16

- Generated version numbers now adhere to PEP 440.

4.4.8 v0.5

Released on 2018-07-12

- Added new CLI end-to-end test suit.
- Improved Docker image.
- Printing usage and help texts to `STDERR` instead of `STDOUT`.

4.4.9 v0.4

Released on 2018-07-01

- Improved `--help` message with option explanations.
- Added usage text to README.

4.4.10 v0.3

Released on 2018-07-01

- Use a multistage Dockerfile.

4.4.11 v0.2

Released on 2018-06-29

- Add Dockerfile together with Docker hub integration.
- Forcefully end every render in a newline. Previously newlines at the end were dropped by Jinja.
- Fixed installation not working.

4.4.12 v0.1

Released on 2018-06-29

Initial commit & release.

Python Module Index

t

Index

Symbols

M

O

T