
sphinxcontrib-eval

Wu Zhenyu

Feb 19, 2024

RESOURCES

1	Only evaluate shell	3
2	Evaluate shell, python, vim script	5
3	Evaluate cmd	7
4	FAQ	9
4.1	eval-sh does not work for Windows	9
4.2	Why not Makefile	9
4.3	Why this name	9
4.4	What is translate shell	9
4.5	Eval is evil	9
4.6	Difference from other projects	9
5	Install & Uninstall	11
5.1	PYPI	11
5.2	AUR	11
6	Requirements	13
6.1	dev	13
6.2	myst	13
7	Eval	15
7.1	MyST	16
7.2	RST	16
7.3	Utilities	16
8	sphinxcontrib-eval	17
8.1	Install	17
8.2	Usage	17
	Python Module Index	21
	Index	23

You can override the default `eval_funcs` in your `docs/conf.py`:

ONLY EVALUATE SHELL

```
from sphinxcontrib.eval import eval_sh  
eval_funcs = {"sh": eval_sh}
```


EVALUATE SHELL, PYTHON, VIM SCRIPT

```
from sphinxcontrib.eval import eval_sh, eval_python
from subprocess import check_output

def eval_vim(input: str) -> str:
    try:
        output = check_output( # nsec B603
            ["nvim", "--headless", "-c", input, "-c", "q"],
            universal_newlines=True,
        )
    except Exception:
        output = ""
    return output

eval_funcs = {"sh": eval_sh, "python": eval_python, "vim": eval_vim}
```


EVALUATE CMD

```
def eval_cmd(input: str) -> str:
    try:
        output = check_output( # nsec B603
            ["cmd", "/c", input], universal_newlines=True
        )
    except Exception:
        output = ""
    return output

eval_funcs = {"cmd": eval_cmd}
```


4.1 eval-sh does not work for Windows

POSIX shell is not installed in Windows by default. See *evaluate cmd* or [install a POSIX shell](#)

4.2 Why not Makefile

Save your time to write Makefile.

4.3 Why this name

Comes from eval-rst of [MyST](#).

4.4 What is translate shell

[My another project](#). The intention of this project is to serve its document generation.

4.5 Eval is evil

I don't guarantee any consequence like:

```
rm -rf your_import_file
```

4.6 Difference from other projects

- [sphinx-execute-code](#) Display code execute result in a code fence, not inject generated markdown/rst to the source code.
- [sphinxcontrib-programoutput](#) Same as above.

INSTALL & UNINSTALL

5.1 PYPI

```
pip install sphinxcontrib-eval
```

```
pip uninstall sphinxcontrib-eval
```

5.2 AUR

```
yay -S python-sphinxcontrib-eval
```

```
sudo pacman -R python-sphinxcontrib-eval
```


REQUIREMENTS

6.1 dev

For unit test and code coverage rate test.

- `myst-parser`
- `pre-commit`
- `pytest-cov`
- `tomli`

6.2 myst

For markdown. Recommend `pip install 'sphinxcontrib-eval[myst]'`.

- `myst-parser`

EVAL

Provide `setup()` to `sphinx`.

`sphinxcontrib.eval.eval_bash(input: str) → str`

Eval bash.

Parameters

input (*str*) –

Return type

`str`

`sphinxcontrib.eval.eval_python(input: str) → str`

Eval python.

Parameters

input (*str*) –

Return type

`str`

`sphinxcontrib.eval.eval_sh(input: str) → str`

Eval sh.

Parameters

input (*str*) –

Return type

`str`

`sphinxcontrib.eval.setup(app: Sphinx) → dict[str, Any]`

Set up.

Parameters

app (*Sphinx*) –

Return type

`dict[str, Any]`

7.1 MyST

Provide `MystEvalParser`.

7.2 RST

Provide `RSTEvalParser`.

7.3 Utilities

`sphinxcontrib.eval.utils.get_lang_map(template: str, eval_funcs: dict[str, Callable[[str], str]]) → dict[str, tuple[Pattern, Callable[[str], str]]]`

Get lang map.

Parameters

- **template** (*str*) –
- **eval_funcs** (*dict[str, Callable[[str], str]]*) –

Return type

`dict[str, tuple[re.Pattern, Callable[[str], str]]]`

`sphinxcontrib.eval.utils.patch_parser(template: str, parser: type[Parser]) → type[Parser]`

Patch parser.

Parameters

- **template** (*str*) –
- **parser** (*Type[Parser]*) –

Return type

`Type[Parser]`

`sphinxcontrib.eval.utils.replace(inputstring: str, pat: Pattern, eval_func: Callable[[str], str]) → str`

Replace.

Parameters

- **inputstring** (*str*) –
- **pat** (*re.Pattern*) –
- **eval_func** (*Callable[[str], str]*) –

Return type

`str`

SPHINXCONTRIB-EVAL

Evaluate shell command or python code in sphinx and myst.

8.1 Install

See [here](#).

8.2 Usage

8.2.1 Enable

docs/conf.py

```
extensions = [  
    "sphinxcontrib.eval",  
]
```

Or

```
extensions = [  
    "myst_parser",  
    "sphinxcontrib.eval", # must be after myst_parser  
]
```

8.2.2 Demonstration

For myst:

```
``{eval-sh}  
echo My OS is $OSTYPE.  
``
```

For rst:

```
.. eval-sh::  
    echo My OS is $OSTYPE.
```

Then build:

```
sphinx-build docs docs/_build/html
```

Result:

```
My OS is linux-gnu.
```

NOTE: the current working directory depends on you. That is, if you run `cd docs && sphinx-build . _build/html && cd -`, CWD will be docs, which is the default setting of <https://readthedocs.org>. So if your code structure is like

```
$ tree --level 1  
.  
├── docs  
├── scripts  
├── src  
└── tests
```

And you want to run `scripts/*.sh`, you need `cd ..` firstly from docs to `.` else you have to run `../scripts/*.sh`.

8.2.3 Advanced Usages

All of the following examples are myst. The corresponding examples of rst are similar. Click the hyperlinks of the titles and scripts to see the actual examples.

Generate API Document

Note: A more “sphinx” solution is [sphinxcontrib-autofile](#).

Before:

```
# API of Translate Shell  
  
``{eval-rst}  
.. automodule:: translate_shell  
    :members:  
.. automodule:: translate_shell.__main__  
    :members:  
... (More)  
``
```

Now

```
# API of Translate Shell

```{eval-rst}
```{eval-sh}
cd ..
scripts/generate-api.md.pl src/**/*.py
```
```
```

Where `scripts/generate-api.md.pl` replaces all `src/translate_shell/XXX.pys` to

```
.. automodule:: translate_shell.XXX
   :members:
```

Generate TODO Document

Before:

```
# TODO

- <https://github.com/Freed-Wu/tranlate-shell/tree/main/src/translate_shell/translators/
  ↳ stardict/__init__.py#L4>
  more stardicts.
- <https://github.com/Freed-Wu/tranlate-shell/tree/main/src/translate_shell/translators/
  ↳ stardict/__init__.py#L5>
  Create different subclasses for different dict to get phonetic, explains
- <https://github.com/Freed-Wu/tranlate-shell/tree/main/src/translate_shell/ui/repl.py
  ↳ #L33>
  make the last line gray like ptpython
- ...
```

Now: (notice `eval-bash` because `readthedocs` uses `dash` as their default `$SHELL`)

```
# TODO

```{eval-bash}
cd ..
shopt -s globstar
scripts/generate-todo.md.pl src/**/*.py
```
```

Where `scripts/generate-todo.md.pl` searches all TODOs in code then convert them to correct hyperlinks.

Generate Requirements Document

Note: A more “sphinx” solution is `sphinxcontrib-requirements-txt`.

Before:

```
# Requirements

## completion

Generate shell completion scripts.

- [shtab](https://pypi.org/project/shtab)

...
```

Now

```
# Requirements

```{eval-sh}
cd ..
generate-requirements.md.pl
```
```

Where `scripts/generate-requirements.md.pl` searches all `requirements/*.txts` and `requirements/completion.txt` is:

```
#!/usr/bin/env -S pip install -r
# Generate shell completion scripts.

shtab
```

See [document](#) to know more.

PYTHON MODULE INDEX

S

- `sphinxcontrib.eval`, [15](#)
- `sphinxcontrib.eval._version`, [16](#)
- `sphinxcontrib.eval.myst`, [16](#)
- `sphinxcontrib.eval.rst`, [16](#)
- `sphinxcontrib.eval.utils`, [16](#)

INDEX

E

`eval_bash()` (in module *sphinxcontrib.eval*), 15
`eval_python()` (in module *sphinxcontrib.eval*), 15
`eval_sh()` (in module *sphinxcontrib.eval*), 15

G

`get_lang_map()` (in module *sphinxcontrib.eval.utils*),
16

M

module
 sphinxcontrib.eval, 15
 sphinxcontrib.eval._version, 15
 sphinxcontrib.eval.myst, 15
 sphinxcontrib.eval.rst, 16
 sphinxcontrib.eval.utils, 16

P

`patch_parser()` (in module *sphinxcontrib.eval.utils*),
16

R

`replace()` (in module *sphinxcontrib.eval.utils*), 16

S

`setup()` (in module *sphinxcontrib.eval*), 15
sphinxcontrib.eval
 module, 15
sphinxcontrib.eval._version
 module, 15
sphinxcontrib.eval.myst
 module, 15
sphinxcontrib.eval.rst
 module, 16
sphinxcontrib.eval.utils
 module, 16