import$_{s}tringDocumentation$

Release 0.1.0

Bruno Rocha

Mar 22, 2017

Contents

Contents:

CHAPTER 1

import_string

Imports an object based on a string

- Free software: ISC license
- Documentation: https://import-string.readthedocs.io.

Features

Imports an object based on a string. This is useful if you want to use import paths as endpoints or something similar. An import path can be specified either in dotted notation (`.`) or with a colon as object delimiter (`:`). If *silent* is True the return value will be *None* if the import fails.

Usage

```
import import_string

module = import_string('my_system.my_package.my_module')

function = import_string('my_system.my_module:some_function')

Class = import_string('my_system.my_module:SomeClass', silent=True)
# If path doesn't exist Class = None
```

Live demo

See it in action here: https://repl.it/EGdS/0

Credits

- This package was extracted from *werkzeug.utils* module
- This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

Stable release

To install import_string, run this command in your terminal:

```
$ pip install import_string
```

This is the preferred method to install import_string, 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.

From sources

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

You can either clone the public repository:

```
$ git clone git://github.com/rochacbruno/import_string
```

Or download the tarball:

```
$ curl  -OL https://github.com/rochacbruno/import_string/tarball/master
```

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

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

CHAPTER 3

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/rochacbruno/import_string/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.

Fix Bugs

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

Implement Features

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

Write Documentation

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

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/rochacbruno/import_string/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 :)

Get Started!

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

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

```
$ git clone git@github.com:your_name_here/import_string.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 import_string
$ cd import_string/
$ 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 import_string 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.

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 Python 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/rochacbruno/import_string/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test tests.test_import_string
```

CHAPTER 4

Credits

Development Lead

- Bruno Rocha <rochacbruno@gmail.com>

Contributors

None yet. Why not be the first?

CHAPTER 5

History

0.1.0 (2016-06-30)

- First release on PyPI.

CHAPTER 6

Indices and tables

- genindex
- modindex
- search