tangle Documentation

Release 0.1.0

Fifman Feng

Sep 26, 2017

Contents

Contents:

CHAPTER 1

1 tangle

A python IoC and AOP framework.

- Free software: MIT license
- Documentation: https://tangle.readthedocs.io.

Contents

1.1 Features

- Decorator (annotation) based configuration support for AOP (aspect oriented programming) and bean (instance of python class) memebers (`Field`).
- Programmatic configuration support for bean construction and application context (i.e. IoC container) configuration.
- Supports application context inheritance, that is, a parent context can be specified to an application context. Note: multi-parents are not supported.
- Supports autowire
- supports the `scope` feature, which determines the lifecycle of the beans managed by the context, including the `Singleton` and `Prototype` scope.
- Context and bean lifecycle hooks support

1.2 Installation

See Installation.

1.3 Usage

See Usage.

1.4 Credits

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

Inspired by Spring Framework.

CHAPTER 2

Installation

Stable release

To install tangle, run this command in your terminal:

```
$ pip install tangle
```

This is the preferred method to install tangle, 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 tangle can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/fifman/tangle
```

Or download the tarball:

```
$ curl  -OL https://github.com/fifman/tangle/tarball/master
```

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

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

CHAPTER 3

Usage

To use tangle in a project:

```
import tangle
```

Basic Concept

If you are familiar with the Spring Framework, you may find the idea and mechanism of this framework quite easy to understand. Let's start with `application context` and `bean`:

- `application context` (a.k.a. `context`): The IoC container which creates and manages the lifecycle of `bean`.
- `bean`: A python object which is an instance of some class.
- `configuration source`: A class instance which defines a list of bean definitions.

Note that when we refer to `bean`, we focus on class instances, although python objects can be of any types (e.g. function, number, string, etc.). What about a class object as we know that it is also an instance of class (i.e. metaclass)? Well theoretically it should be OK to register class objects as beans in a context, but it is not fully tested. Therefore, it is *NOT* recommended to use `tangle` to manage class objects (maybe in the future).

When you apply `tangle` to your python program, the program may contain one or several `context`, and many `bean`. If you want to use the beans, use the `get(bean_id)` method of the context to obtain a bean instance. The `bean_id` is provided by your `configuration source`.

Context

context

Bean

bean

Bean management

Dependency

Scope

Autowire

Event

event

Aspect

aspect

CHAPTER 4

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/fifman/tangle/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

tangle could always use more documentation, whether as part of the official tangle 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/fifman/tangle/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 *tangle* for local development.

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

```
$ git clone git@github.com:your_name_here/tangle.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 tangle
$ cd tangle/
$ 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 tangle 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/fifman/tangle/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_tangle
```

CHAPTER 5

Credits

Development Lead

- Fifman Feng <fifman@163.com>

Contributors

None yet. Why not be the first?

CHAPTER 6

History

0.1.0 (2017-08-14)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search