Montague

Release 0.2.0

June 14, 2015

Contents

Contents:

CHAPTER 1

Overview

1.1 Montague-PasteDeploy

Montague-PasteDeploy is a loader for Montague with full PasteDeploy backwards compatibility.

- Free software: MIT license

1.1.1 Status

Montague-PasteDeploy is a pre-1.0 project using Semantic Versioning.

1.1.2 Installation

```
pip install montague_pastedeploy
```

1.1.3 Documentation

https://montague-pastedeploy.readthedocs.org/

1.1.4 Development

To run the all tests run:

```
tox
```

CHAPTER 2

Installation

At the command line:

```
pip install montague
```

CHAPTER 3

Usage

To use Montague in a project:

```
import montague
```

CHAPTER 4

Writing your own config loader

Do you want to store your WSGI app config in Redis? In a TOML file? In ZooKeeper? You can do that.

You will need to decide two things:

1. A filename extension for your configuration format: for actual files, that's easy. A JSON loader should work with `foo.json` files. However, if you are loading your configuration from a service, you still need to pick a filename extension; Montague uses that to dispatch to config loaders. Perhaps you will store the Redis connection info in `myfile.redis`, or perhaps the file doesn't even exist and you extract the connection info from the path name.
2. Whether you need to support individual `app_config`, `server_config`, etc methods: the default INI loader supports these because the PasteDeploy INI format allows individual app configs to override global variables for that specific app.

Config loaders must provide the methods listed in the `montague.interfaces.IConfigLoader` interface; it's recommended, though not required, that they actually implement the interface using `zope.interface`.

4.1 Montauge Standard Format

The actual layout of your configuration information will obviously vary from format to format. Because of this, Montague has a standard layout you should use when implementing the `montague.interfaces.IConfigLoader.config()` method. You should return a dict that looks like this:

```
{
    "globals": {},
    "application": {},
    "composite": {},
    "filter": {},
    "server": {},
}
```

Of course, the dict can contain other keys as well, but those are the ones Montague cares about.

CHAPTER 5

Reference

5.1 montague_pastedeploy

CHAPTER 6

Contributing

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

6.1 Bug reports

When 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.

6.2 Documentation improvements

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

6.3 Feature requests and feedback

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

6.4 Development

To set up *montague* for local development:

1. Fork montague on GitHub.
2. Clone your fork locally:

```
git clone git@github.com:your_name_here/montague.git
```

3. Create a branch for local development:

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

   Now you can make your changes locally.

4. When you're done making changes, run all the checks, doc builder and spell checker with tox one command:

```
tox
```

5. 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
```

6. Submit a pull request through the GitHub website.

6.4.1 Pull Request Guidelines

If you need some code review or feedback while you're developing the code just make the pull request.

For merging, you should:

1. Include passing tests (run `tox`) [1].
2. Update documentation when there's new API, functionality etc.
3. Add a note to `CHANGELOG.rst` about the changes.
4. Add yourself to `AUTHORS.rst`.

6.4.2 Tips

To run a subset of tests:

```
tox -e envname -- py.test -k test_myfeature
```

To run all the test environments in *parallel* (you need to `pip install detox`):

```
detox
```

[1] If you don't have all the necessary python versions available locally you can rely on Travis - it will run the tests for each change you add in the pull request.
It will be slower though ...

CHAPTER 7

Authors

- Jon Rosebaugh - http://inklesspen.com

Based upon PasteDeploy 1.5.2 by Ian Bicking.

CHAPTER 8

Changelog

8.1 0.2.0 (2015-06-14)

- changed DEFAULT behavior in test config loader to correspond to Montague 0.2.0. This is a breaking change.
- added support for logging_config
- Removed bundled fakeapp egg in favor of montague_testapps.
- Removed zope interfaces to correspond to the next version of Montague.

8.2 0.1.0 (2014-11-12)

- First release on PyPI, corresponding to PasteDeploy 1.5.2.
- Backwards incompatibility: ConfigMiddleware no longer offers a threadlocal `CONFIG` importable. (This removes the dependency on Paste.)

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

m

M