Germ Documentation

Release 0.1.0

Jesse Roberts

February 18, 2017

Contents

Contents:

CHAPTER 1

Germ

About

Germ provides a lightweight framework for building Python applications. By abstracting the main thread and configuration logic, Germ allows you to simply focus on building your domain logic. Integrating with Germ is as simple as creating a package that organizes your code into some logical sub packages and declaring them in your application config.

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

Examples

Simple Chat Bot

Here's all you'd need to write for a generic chat bot.

- Threads - Listener, Messenger
- Chat adapter (facilitate communication with chat system & bot threads)

Your application configuration would specify instances of the Listener/Messenger threads as well as connection parameters for your chat adapter. To facilitate communication between threads, standard queues can be used. This is another simple config entry to create the Queue instance which your threads will expect.

Media Storage Server

A more complex example, this application can store media (ie ebooks, music, etc), update metadata, and serve the files to end users.

- Threads - Metadata Parser,
- Storage - File/Database
- Webservice - API for accessing media

Features (^planned)

- ^On the fly plugin updates

- ^Live configuration changes
- ^Load plugins from PyPi, GitHub, disk
- ^CLI for quick setup/testing

Todo

- Documentation
- Everything

CHAPTER 2

Germ Design

Rationale

My goal with Germ is simple: provide a reusable framework for building Python applications. I've built several distributed Python applications and have ended up largely copying the base framework between them. This has led to a couple frustrations:

1. I hate copy-pasting code. As I make improvements in one codebase I have to backport

those changes to the other codebase... or just let the old code rot.

2. I waste time tweaking the base code that I could be spending on my problem domain.

I started using Flask for a few projects and fell in love. I began moving my Bottle & Django applications over and realized that the Flask philosophy could work for me.

Why not build a framework that handles the very basics and allows(encourages) building extra functionality on top as plugins?

Design

While Flask is a huge inspiration, I don't intend to simply copy Flasks design. I am slightly more opinionated about the structure of plugins and how they integrate into Germ.

With that in mind, you won't write an application and import Germ. Instead you would write your functionality as a plugin and tell Germ to run it.

CHAPTER 3

Installation

Stable release

To install Germ, run this command in your terminal:

```
$ pip install germ
```

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

You can either clone the public repository:

```
$ git clone git://github.com/jesseops/germ
```

Or download the tarball:

```
$ curl  -OL https://github.com/jesseops/germ/tarball/master
```

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

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

CHAPTER 4

Usage

To use Germ in a project:

```
import germ
```

CHAPTER 5

Plugin Development

Overview

A Germ plugin

CHAPTER 6

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/jesseops/germ/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

Germ could always use more documentation, whether as part of the official Germ 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/jesseops/germ/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 *germ* for local development.

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

```
$ git clone git@github.com:your_name_here/germ.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 germ
$ cd germ/
$ 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 germ 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/jesseops/germ/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest tests.test_germ
```

CHAPTER 7

Indices and tables

- genindex
- modindex
- search