LetThereBe Documentation

Release 0.0

Mayeul d'Avezac, Ilektra Christidi, Alice Harpole, David Perez Sua

Mar 29, 2017

Contents:

`LetThereBe` automates the creation of a software project for collaborative development across any discipline to support best practice. It lowers the barrier for good practice development by providing an extension to GitHub to generate an immediately usable project structure automatically.

CHAPTER 1

Installation

pypi

The LetThereBe command line tool can be installed from pypi (the **Py**thon **P**ackage **I**ndex). To do this, enter the following command into your terminal:

```
pip install lettherebe
```

From source

Alternatively, it can be installed from source, provided you have the following python packages (listed in REQUIREMENTS) installed on your machine:

```
requests
pygithub
click
colorama
```

You can then install the package by cloning the GitHub repository, navigating to the project directory and executing the python setup script:

```
git clone git@github.com:LetThereBe/lettherebe.git
cd lettherebe
python setup.py install
```

CHAPTER 2

Setting up a project

To get some simple instructions on how to use `LetThereBe`, execute the command without any arguments:

```
lettherebe
```

Before setting up your project, you will need a GitHub account. If you do not currently have one, you will need to register.

To begin setting up a project using `LetThereBe`, navigate to the project directory and enter the command

```
lettherebe quickstart [--options]
```

where `options` are optional options for your project (see below).

This will then take you through the set up of your project with a set of prompts. The first prompt will ask you for the details of your GitHub account - if you have not got one, you will be directed to register for one. Next, you will be asked for the name of the project, then `LetThereBe` will then provide a series of prompts about the specifics of your project. If all the default options are chosen, your project will end up with:

- a GitHub repository
- an **MIT License**
- simple example **tests** to be executed using `py.test`
- **continuous integration** of these tests using Travis CI
- **code coverage** provided by Codecov
- **documentation** with Sphinx which will be automatically built on ReadTheDocs

Advanced Setup

If you choose not to use the default options, you will then be prompted to provide answers to a series of questions which will allow you to customise these settings. Alternatively, you will have the ability to provide a configuration file describing the settings you would like to use, making it easy to reuse settings from other projects. **Insert information about config file here**.

The initial project setup command can also be modified using the following options as an alternative to passing these when prompted by the wizard.

--project=name	name of project
--service=service	version control service to use
--language=language	sets the project language (e.g. `python`, `julia`, `R`...)
--license=license	sets the project license (e.g. MIT, BSD,....
--CI=provider	sets which continuous integration provider to use, e.g. Travis CI, CircleCI,...
--docs=package	sets which package to use to build the documentation, e.g. Sphinx, Doxygen...

`LetThereBe` can also be used to set up each of the above components separately. To do this, first navigate to the project directory, then execute one of the following commands

```
lettherebe docs [-options]
```

If no options are passed, this will set up sphinx documentation which is then automatically built on ReadTheDocs. Alternatively, the following options can be passed:

-sphinx	documentation is set up using Sphinx
-doxygen	documentation is set up using Doxygen
-readthedocs	documentation hosted and built online in ReadTheDocs
-offline	documentation built locally only (not connected to ReadTheDocs)

```
lettherebe tests [-options]
```

If there are no options passed and the project language has been set to `python`, this will set up some simple tests that can be run using `py.test` and will connected to a `travis.ci` project so that they are executed when changes to the code are pushed to the GitHub repository. The following options are possible:

-travis	sets up continuous integration using `travis.ci`
-circle	sets up continuous integration using `Circle CI`

CHAPTER 3

Developing your project

`LetThereBe` will set up your project with some basic tests and documentation and link it to the tools needed to run and build these for you, however as your project grows you will need to develop these further. On this page we provide links to some guides to help you do this.

Testing

- Testing Your Code from *The Hitchiker's Guide to Python*: general guidance on writing tests and `python`-specific examples
- Automated testing from *Learn Python the Hard Way*
- pytest framework documentation for testing `python` code

Documentation

- A beginner's guide to writing documentation from Write The Docs
- ReadTheDocs documentation
- Quick reStructured Text: guide on rst (reStructured Text) syntax. This is the default markup language used by sphinx for documentation.

CHAPTER 4

Writing a plugin

LetThereBe is designed to be extensible and customisable so that it can be **tailored to other services or programming languages, and users can write **plugins. Any code below can be either added to a config.py file in the directory from where you are running the `lettherebe` command, or you can add them to the main package via There are three main components that can be extended:

Repository hosting providers

At the moment we have provided support for GitHub, but other services such as BitBucket or GitLab could be added. To do this, you need to define a function that sets up a repo on the given service. The basic structure of the code to do this is:

```
from lettherebe.registry import repository_host, Repository


class MyRepository(Repository):

    def add_directory_to_repo(local_path, remote_path, message):
        # Code here that knows how to put the directory at local_path into
        # the repository, in a sub-directory given by remote_path. The
        # message argument gives the commit message.

    def instructions_to_get_repo(self):
        # This should return a string with instructions on how to access
        # the repository and how to clone.


@repository_host('name-of-service')
def set_up_repo(argument1, argument2=False):
    """
    Set up an empty repository on a repository host

    Parameters
    ----------
    argument1 : str
        What is the value of argument1?
    argument2 : bool, optional
        What about the value of argument2?
    """
    # create repo here then return class to provide abstract interface
    return MyRepository(...)
```

The questions for the command-line and website are taken from the docstring, which should be in NumpyDoc format. Default values can be specified by using keyword arguments. It is important that this function returns a Repository subclass, but otherwise you can do anything you like in the function. All the rest will be taken care of by the `lettherebe` package!

Package templates for different languages

To add support for a 'best practice template' for a new language, you will need to define a function as follows:

```
from lettherebe.registry import package_language

@package_language('fortran66')
def set_up_fortran66_package(tests=True):
    """
    Set up a basic Python package

    Parameters
    ----------
    tests : bool, optional
        Should tests be included?
    """
    directory = tempfile.mkdtemp()
    # populate directory here
    return directory
```

The use of `tests` as the argument is just an example, and you can put any number of arguments/options. The function should write the generated files to a local temporary directory and return the path to this directory. The rest of the package will then automatically sync those generated files to the remote repository.

CHAPTER 5

Contributing

To contribute to `LetThereBe`, please submit a pull request. We ask that you ensure that the code is covered by unit tests and documentation and that the code style is consistent with that used elsewhere in the project.

Note that any contributed code will be covered by the *project license*.

CHAPTER 6

License