package Template Documentation

Release 1.0.0

Mirko Mälicke

May 16, 2018

Contents:

This is an example documentation for the package-template GitHub project. This is meant to be overwritten by the documentation for your code. Go to the *Howto Guide* to learn how to copy this template. navigate this documentation to see some example pages.

CHAPTER 1

Installation

1.1 Installation Guide

Explain how your package can be installed. As we use the PyPI approach here, you will most likely want to show the pip installation. It is also recommended to show how your package can be install from source, as some users prefer downloading code from github.

1.1.1 PyPI

The lastest stable version can be found on PyPI and installed via pip.

```
pip install package_template
```

1.1.2 GitHub

A more recent version might be available on GitHub.

```
git clone https://github.com/KIT-HYD/package-template.git
cd package-template
pip install -r requirements.txt
pip install -e .
```

1.2 Using this Template

Important: The above section gave an example, how a installation guide for this package would look like. In case you are planning to use this package as a template for your own code, please refer to the *Howto Guide*.

CHAPTER 2

Use the Template

2.1 How to use this template

Important: Although this template is meant to be used for publishing your code on the Python Package index (PyPI), you cannot install this template using pip. Please follow this description

2.2 Download the template

Download and unzip the template from http://github.com/KIT-HYD/package-template either by downloading it directly or using the command line:

```
wget https://gitub.com/KIT-HYD/package-template/archive/master.zip
unzip master.zip
```

Important: Do not clone the repository as it will download the hidden `.git` folder as well. This will prevent from initializing your own git project at a later step.

2.3 Initialize your Project

Before you continue, make sure to signup at GitHub, GitLab, BitBucket or any other repository that supports git. As far as I know, there is no way to create a new project locally and just push it to the repository. Therefore initializing your project need some more steps:

Create a new repository on e.g. GitHub. I will use GitHub for this guide. You will have to use a unique name for your project on account level. However, if you want to add your package also to the Python Package Index, you will need a global unique name. So check the PyPI before you choose a name.

Once your repository is created, clone it to your computer, wherever you want your files to live. On the command line this can be done by:

```
cd to/your/path
git clone https://github.com/username/reponame
```

Of course you need to change username and reponame to your project information. In case you use Windows, you can also use the Desktop application for this step: https://desktop.github.com. In case you want to use PyCharm (https://www.jetbrains.com/pycarm), you can create a new Project directly from the IDE. This is my preferred option.

2.4 Rename the template

After unzipping the package-template you will find a folder called `package-template-master` in that directory. Copy the content of this folder into your newly created github project directory. Depending on the repository you chose, this project might already contain a `README.md`, `LICENSE` and definitely a `.gitignore` file. Make sure to replace these files with the versions from `package-template-master`.

2.5 Commit changes

Finally you need to add the new files to the GitHub repository. Only added files will be versioned. At some point you can then *commit* changes. A commit can be thought of a local snapshot of your project, which can also be reverted. Once you created a commit, this commit can then be *pushed* to the remote repository on GitHub. Then all changes are also available on the online version.

In case you are using the Desktop application, you can add, commit and push from within the software.

PyCharm offers a VCS menu, where you can choose to commit changes. The dialog will also show you all unversioned (not added) files. Instead of just commit your changes, you can also choose the 'commit and push' option.

From the command line these steps can be achieved like:

Inside the GitHub repository, all files can be added by:

```
git add .
```

Then commit the changes

```
git commit -m "initial commit"
```

Where the -m flag will set a commit message. It is highly recommended to always add a very short commit message, to keep track of the changes.

Finally, push the commit to the master branch.

```
git push
```

2.6 Integrations

This far, you jst set up a GitHub (or gitlab, bitbucket) repository and copied the template files into your project. The main purpose of the template is to make the integration of your project into helpful developer tools easier. But you will usually have to sign up for thrid party services. Luckily you can use your GitHub account for most services, with the exception of PyPI.

See also:

Refer to the following pages to get started with the different developer tools:

- *Python Package Index*

CHAPTER 3

Integration

3.1 Python Package Index

3.1.1 PyPI

Setup

Note: Before you can publish any package to the Python Package Index, you'll have to register here: https://pypi.org/account/register/

In a second step, you will have to configure the `setuptools` to be able to pack your project file. The easiest approach is to add a new hidden file called `.pypirc` to your home folder. This file should look like:

Listing 1: .pypirc

```
[disutils]
index-servers =
    pypi

[pypi]
repository: https://www.python.org/pypi
username: yourname
password: yourpassword
```

Replace `yourname` and `yourpassword` with the username and password you chose when registering on PyPI.

Warning: Your password is saved as plain text in this file, so make sure that nobody else has access to your home folder. Otherwise refer to the PyPI documentation for upload procedures without saved password.

Prepare your project

Todo: Write this section

Upload your project

Todo: Write this section

CHAPTER 4

API Reference

4.1 core package

4.1.1 mymodel