在Pypi上发布自己的Python包

使用Python编程的都知道，Python的包安装非常的方便，一般都是可以**pip**来安装搞定：

```
sudo pip install <package name>
```

pip的安装请移步：https://pip.pypa.io/en/stable/installing/

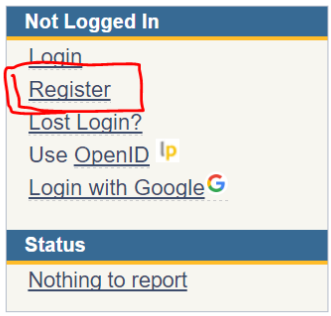
最近因为项目上的需要，发布了一个自己的pypi Python包，这里我大致分享如何发布自己的Pypi包一般过程。

打包工作主要依赖python的一个叫setuptools的包来完成，在进行下面操作前请使用pip安装它：

```
sudo pip install setuptools
```

（这里面我都是假设你已经准备好你的代码，测试代码以及目录结构，加上我今天要分享的，就可以组成一个完整的**python**包）

- **1. 第一步，就是到pypi（https://pypi.python.org/pypi）上注册自己的用户，点击"Register"，填写自己的用户名，密码，邮件地址**



记住自己的用户名和密码，后面上传的时候要输入的

- **2. 准备setup.py/setup.conf文件，它是放在你包的根目录的，这一步至关重要，包括要发布的包名字，版本，license，描述，特性（classifier)等等，下面是我自己包的一个setup.py文件的内容，基本上只需要在这个上面修改就行了，具体如下：**

```
#!/usr/bin/env python
# coding=utf-8

from setuptools import setup, find_packages

setup(
    name='<项目的名称>',
    version=<项目版本>,
    description=(
        '<项目的简单描述>'
    ),
    long_description=open('README.rst').read(),
    author='<你的名字>',
    author_email='<你的邮件地址>',
```

```
    maintainer='<维护人员的名字>',
    maintainer_email='<维护人员的邮件地址',
    license='BSD License',
    packages=find_packages(),
    platforms=["all"],
    url='<项目的网址，我一般都是github的url>',
    classifiers=[
        'Development Status :: 4 - Beta',
        'Operating System :: OS Independent',
        'Intended Audience :: Developers',
        'License :: OSI Approved :: BSD License',
        'Programming Language :: Python',
        'Programming Language :: Python :: Implementation',
        'Programming Language :: Python :: 2',
        'Programming Language :: Python :: 2.7',
        'Programming Language :: Python :: 3',
        'Programming Language :: Python :: 3.4',
        'Programming Language :: Python :: 3.5',
        'Programming Language :: Python :: 3.6',
        'Topic :: Software Development :: Libraries'
    ],
)
```

需要注意的上面的字段：

- **version** - 这个简单，就是包的发布的版本，可以直接写在这，也可以从其他地方引用过来。
- **long_description** - 必须是rst（reStructuredText)格式的，因为这个里面的内容是显示在pypi包首页上，具体rst的语法可以参考：http://rest-sphinx-memo.readthedocs.io/en/latest/ReST.html

  我的long_description是同目录下的README.rst的内容，同时这个README也是我的github项目首页。
- **packages** - 申明你的包里面要包含的目录，比如 ['mypackage', 'mypackage_test'] 可以是这种使用我的示例，让setuptools自动决定要包含哪些包
- **install_requires** - 申明依赖包，安装包时pip会自动安装：格式如下（我上面的setup.py没有这个参数，因为我不依赖第三方包:)）：

```
    install_requires=[
        'Twisted>=13.1.0',
        'w3lib>=1.17.0',
        'queuelib',
        'lxml',
        'pyOpenSSL',
        'cssselect>=0.9',
        'six>=1.5.2',
        'parsel>=1.1',
        'PyDispatcher>=2.0.5',
        'service_identity',
    ]
```

3. 准备requirements.txt 和 test-requirements.txt，这个申明包的依赖包和跑自动化测试的测试依赖包，具体格式示例如下：

```
mock>=2.0.0
flake8>=3.2.1
eventlet>=0.19.0
nose2>=0.6.5
cov_core>=1.15.0
virtualenv>=15.1.0
```

2016年2月 (1)
2016年1月 (1)
2015年9月 (2)
2015年4月 (1)
2015年3月 (1)
2015年1月 (2)
2014年12月 (2)

最新评论

1. Re:OpenStack中的Multipath faulty device的成因及解决(part 2)
d
--u先生
2. Re:用gdb调试python多线程代码-记一次死锁的发现
好文值得学习
--xybaby
3. Re:使用docker部署standalone cinder
支持支持
--cfgx294
4. Re:Windows上Ruby开发环境的配置
@JustYong引用最近也有个RFID的项目需要用到Ruby，将脚本布置到RFID设备中，用于跟踪设备的状态。共勉倒是想问下，RFID不是一般都是嵌入式设备吗？还能跑Ruby？:)...
--孤独的居士
5. Re:Windows上Ruby开发环境的配置
最近也有个RFID的项目需要用到Ruby，将脚本布置到RFID设备中，用于跟踪设备的状态。共勉
--JustYong

阅读排行榜

1. 在Ubuntu Linux下制作Windows 启动安装 USB盘(5376)
2. Floating IP in OpenStack Neutron(5310)
3. Protobuf 在Ubuntu 14上的编译与使用(5031)
4. 基于SSH协议的端口转发(4026)
5. docker X509 证书错误的终极解决办法(3892)

评论排行榜

1. Windows上Ruby开发环境的配置(3)
2. 基于SSH协议的端口转发(2)
3. OpenStack中的Multipath faulty device的成因及解决(part 2)(1)
4. 用gdb调试python多线程代码-记一次死锁的发现(1)
5. 使用docker部署standalone cinder(1)

推荐排行榜

1. OpenvSwitch Port Mirror in OpenStack Neutron(1)
2. Floating IP in OpenStack Neutron(1)
3. 用gdb调试python多线程代码-记一次死锁的发现(1)
4. Windows上Ruby开发环境的配置(1)
5. 在Pypi上发布自己的Python包(1)

以上是我的test-requirements.txt的内容，requirements.txt的格式个上面一样。

准备这个两个文件不是必须的。

但是，有了它们，用户可以自己手动安装依赖包

```
pip install -r requirements.txt
```

有了它们，结合tox等工具，可以非常方便的加入自动化测试。

- **4. 准备一个项目的README.rst文件,前面也提到了它的格式要求，第一次发包，可以直接copy别人的格式，这东西熟能生巧，多写就会了。**

README的截图就不放了，以免广告嫌疑。有兴趣可以到参考http://rest-sphinx-memo.readthedocs.io/en/latest/ReST.html

- **5. 准备好上面的步骤，一个包就基本完整了，剩下的就是打包了(cd到包的根目录)**:

可以使用下面命令打包一个源代码的包:

```
python setup.py sdist build
```

这样在当前目录的dist文件夹下，就会多出一个以tar.gz结尾的包了:

也可以打包一个wheels格式的包，使用下面的命令搞定:

```
python setup.py bdist_wheel --universal
```

这样会在dist文件夹下生成一个whl文件，

- **6. 上传生成的包，可以使用setuptools,或者twine上传,推荐使用twine上次，因为使用setuptools上传时，你的用户名和密码是明文或者未加密传输，安全起见还是使用twine吧**

```
# 上传source 包
python setup.py sdist upload
# 上传pre-compiled包
python setup.py bdist_wheel upload
```

使用twine上传,先安装twine

```
sudo pip install twine

twine upload dist/*
```

上次前都会提示你前面注册的用户名和密码。一切搞定，你的包现在可以通过pip在任何地方安装了。

后续

其实对于一个包，你要长久维护，自动测试肯定要加入，后面有时间再分享下如何使用tox的使用和与第三方CI的集成。

引用链接:

pypi详细教程: https://packaging.python.org/distributing

标签: Python, pip, pypi, setuptools

孤独的居士
关注 - 3
粉丝 - 7
+加关注

1 0

« 上一篇: docker X509 证书错误的终极解决办法
» 下一篇: Windows上Ruby开发环境的配置

posted @ 2017-03-14 22:48 孤独的居士 阅读(3159) 评论(0) 编辑 收藏