写代码的小孩

苦逼op 快乐地写代码

个人资料

xluren

访问： 299040次
积分： 4002
等级： BLOG 5
排名： 第4890名

原创： 146篇
转载： 82篇
译文： 1篇
评论： 30条

文章分类

网络编程 (13)
系统编程 (8)
笔试面试 (1)
工具配置 (16)
语言误区 (6)
hadoop (1)
模式设计 (3)
个人总结 (67)
数据库相关 (23)
op写前端 (8)
django (2)
linux命令 (11)
监控相关 (7)
python 周边 (46)
mysql (5)
网络相关 (4)
运维工具 (6)
日志处理相关 (3)
个人转载 (46)
系统相关 (9)
flask (2)

关于python中的setup.py

标签： python

2014-11-14 10:41 13642人阅读 评论(1) 收藏 举报

分类： python 周边（45） 个人转载（45）

目录(?) [+]

本文是转载的，源地址：http://lingxiankong.github.io/blog/2013/12/23/Python-setup/

感谢作者，解决了我一个大问题

前言

其实对于setup.py和setup.cfg的关注是从OpenStack的源码包中开始的，OpenStack每个组件的发布时都是一个tar.gz包，同样，我们直接从github上clone代码后也会发现两个文件的存在。当阅读Nova或Ceilometer（其他组件可能也会涉及）的代码时，发现setup.cfg中内容对于代码的理解有很大的影响。那么，到底setup.py和setup.cfg是干什么的？

setup.py

我们从例子开始。假设你要分发一个叫foo的模块，文件名foo.py，那么setup.py内容如下：

```
1. from distutils.core import setup
2. setup(name='foo',
3.       version='1.0',
4.       py_modules=['foo'],
5.       )
```

然后，运行`python setup.py sdist`为模块创建一个源码包

```
1. root@network:/kong/setup# python setup.py sdist
2. running sdist
3. running check
4. warning: check: missing required meta-data: url
5. warning: check: missing meta-data: either (author and author_email) or
   (maintainer and maintainer_email) must be supplied
6. warning: sdist: manifest template 'MANIFEST.in' does not exist (using default
   file list)
7. warning: sdist: standard file not found: should have one of README, README.txt
8. writing manifest file 'MANIFEST'
9. creating foo-1.0
10. making hard links in foo-1.0...
```

web开发 (1)
系统资源统计 (3)
lua (3)
golang (2)
ruby相关 (1)
机器学习 (1)

文章存档
2015年10月 (3)
2015年09月 (2)
2015年08月 (2)
2015年07月 (10)
2015年06月 (3)
展开

阅读排行
mysql索引总结----mysql (43123)
常用的时间同步服务器地 (21441)
解决nginx负载均衡的ses (21388)
关于python中的setup.py (13621)
mysql 字段类型总结---da (8431)
django+ajax (7215)
python的cls，self，clas (6350)
shell getopts 用法 (5860)
wireshark 抓包分析 TCP (5450)
Bringing up interface eth (5072)

评论排行
2014年第一篇哦，nagio (6)
zabbix mysql 迁移总结 (5)
django+ajax (4)
flask cache with memca (3)
linux线程私有数据 (2)
php 操作mysql是否成功 (2)
mysql 字段类型总结---da (2)
私有github搭建使用 (2)
mysql索引总结----mysql (1)
mysql 字段类型总结----d (1)

最新评论
mysql 字段类型总结---date date
倪文杰: tongshang
关于python中的setup.py
jzhx107: 讲得很好，谢谢！
zabbix mysql 迁移总结
freefrankx: 你好请问能否发你的总结给我呢。我的邮箱是2521573576@qq.com，谢谢
mysql索引总结----mysql 索引类
k522045458: --修改表结构的方式添加索引ALTER TABLE table_name ADD INDEX inde...
私有github搭建使用
xluren: @ycdyx:棒棒的，不过你可以试试gitlab。
私有github搭建使用
ycdyx: 写的太简单了，对于小白用户还是很难理解，so，自己整理了一篇：http://www.webyang....
nagios安装配置
xiaoone_csdn: 如果觉得麻烦，可以试试OneAlert一键集成nagios，轻松实现微信、电话、

```
11. hard linking foo.py -> foo-1.0
12. hard linking setup.py -> foo-1.0
13. creating dist
14. Creating tar archive
15. removing 'foo-1.0' (and everything under it)
```

在当前目录下，会创建`dist`目录，里面有个文件名为`foo-1.0.tar.gz`，这个就是可以分发的包。使用者拿到这个包后，解压，到foo-1.0目录下执行：`python setup.py install`，那么，foo.py就会被拷贝到python类路径下，可以被导入使用。

```
1. root@network:/kong/setup/dist/foo-1.0# python setup.py install
2. running install
3. running build
4. running build_py
5. creating build
6. creating build/lib.linux-x86_64-2.7
7. copying foo.py -> build/lib.linux-x86_64-2.7
8. running install_lib
9. copying build/lib.linux-x86_64-2.7/foo.py -> /usr/local/lib/python2.7/dist-packages
10. byte-compiling /usr/local/lib/python2.7/dist-packages/foo.py to foo.pyc
11. running install_egg_info
12. Removing /usr/local/lib/python2.7/dist-packages/foo-1.0.egg-info
13. Writing /usr/local/lib/python2.7/dist-packages/foo-1.0.egg-info
14. root@network:/kong/setup/dist/foo-1.0# ll /usr/local/lib/python2.7/dist-packages/foo
15. foo-1.0.egg-info  foo.py            foo.pyc
```

对于Windows，可以执行`python setup.py bdist_wininst`生成一个exe文件；若要生成RPM包，执行`python setup.py bdist_rpm`，但系统必须有rpm命令的支持。可以运行下面的命令查看所有格式的支持：

```
1. root@network:/kong/setup# python setup.py bdist --help-formats
2. List of available distribution formats:
3.   --formats=rpm      RPM distribution
4.   --formats=gztar    gzip'ed tar file
5.   --formats=bztar    bzip2'ed tar file
6.   --formats=ztar     compressed tar file
7.   --formats=tar      tar file
8.   --formats=wininst  Windows executable installer
9.   --formats=zip      ZIP file
10.  --formats=msi      Microsoft Installer
```

setup函数还有一些参数：

1、`packages`
告诉Distutils需要处理那些包（包含`__init__.py`的文件夹）
2、`package_dir`
告诉Distutils哪些目录下的文件被映射到哪个源码包。一个例子：`package_dir = {'': 'lib'}`，表示“root package”中的模块都在lib目录中。
3、`ext_modules`
是一个包含Extension实例的列表，Extension的定义也有一些参数。
4、`ext_package`
定义extension的相对路径
5、`requires`
定义依赖哪些模块

6、`provides`

定义可以为哪些模块提供依赖

7、`scripts`

指定python源码文件，可以从命令行执行。在安装时指定`--install-script`

8、`package_data`

通常包含与包实现相关的一些数据文件或类似于readme的文件。如果没有提供模板，会被添加到MANIFEST文件中。

9、`data_files`

指定其他的一些文件（如配置文件）

```
setup(...,
      data_files=[('bitmaps', ['bm/b1.gif', 'bm/b2.gif']),
                  ('config', ['cfg/data.cfg']),
                  ('/etc/init.d', ['init-script'])]
      )
```

规定了哪些文件被安装到哪些目录中。如果目录名是相对路径，则是相对于`sys.prefix`或`sys.exec_prefix`的路径。如果没有提供模板，会被添加到MANIFEST文件中。

执行sdist命令时，默认会打包哪些东西呢？

- 所有由`py_modules`或`packages`指定的源码文件
- 所有由`ext_modules`或`libraries`指定的C源码文件
- 由`scripts`指定的脚本文件
- 类似于test/test*.py的文件
- README.txt或README，setup.py，setup.cfg
- 所有`package_data`或`data_files`指定的文件

还有一种方式是写一个manifest template，名为`MANIFEST.in`，定义如何生成MANIFEST文件，内容就是需要包含在分发包中的文件。一个MANIFEST.in文件如下：

```
include *.txt
recursive-include examples *.txt *.py
prune examples/sample?/build
```

setup.cfg

setup.cfg提供一种方式，可以让包的开发者提供命令的默认选项，同时为用户提供修改的机会。对setup.cfg的解析，是在setup.py之后，在命令行执行前。

setup.cfg文件的形式类似于

```
[command]
option=value
...
```

其中，`command`是Distutils的命令参数，`option`是参数选项，可以通过`python setup.py --help build_ext`方式获取。

需要注意的是，比如一个选项是--foo-bar，在setup.cfg中必须改成foo_bar的格式

符合Distutils2的setup.cfg有些不同。包含一些sections：

1、`global`

定义Distutils2的全局选项，可能包含commands，compilers，setup_hook（定义脚本，在setup.cfg被读取后执行，可以修改setup.cfg的配置）

2、`metadata`

3、`files`

- packages_root：根目录
- packages
- modules
- scripts
- extra_files

4、command sections

Setuptools

上面的setup.py和setup.cfg都是遵循python标准库中的Distutils，而setuptools工具针对Python官方的distutils做了很多针对性的功能增强，比如依赖检查，动态扩展等。很多高级功能我就不详述了，自己也没有用过，等用的时候再作补充。

一个典型的遵循setuptools的脚本：

```
from setuptools import setup, find_packages
setup(
    name = "HelloWorld",
    version = "0.1",
    packages = find_packages(),
    scripts = ['say_hello.py'],

    # Project uses reStructuredText, so ensure that the docutils get
    # installed or upgraded on the target machine
    install_requires = ['docutils>=0.3'],

    package_data = {
        # If any package contains *.txt or *.rst files, include them:
        '': ['*.txt', '*.rst'],
        # And include any *.msg files found in the 'hello' package, too:
        'hello': ['*.msg'],
    },

    # metadata for upload to PyPI
    author = "Me",
    author_email = "me@example.com",
    description = "This is an Example Package",
    license = "PSF",
    keywords = "hello world example examples",
    url = "http://example.com/HelloWorld/",   # project home page, if any

    # could also include long_description, download_url, classifiers, etc.
)
```

如何让一个egg可被执行？

```
setup(
    # other arguments here...
    entry_points = {
        'setuptools.installation': [
            'eggsecutable = my_package.some_module:main_func',
        ]
    }
)
```

如何定义一个可选特性？

```
setup(
    name="Project-A",
    ...
    extras_require = {
        'PDF':  ["ReportLab>=1.2", "RXP"],
        'reST': ["docutils>=0.3"],
    }
)
```

特性如何使用呢？需要与entry points结合使用：

```
setup(
    name="Project-A",
    ...
    entry_points = {
        'console_scripts': [
            'rst2pdf = project_a.tools.pdfgen [PDF]',
            'rst2html = project_a.tools.htmlgen',
            # more script entry points ...
        ],
    }
)
```

或者被其他project依赖：install_requires = ["Project-A[PDF]"]

插件式开发

我想大家最熟悉的就是这个特性了吧。比如一个博客系统想用不同的插件支持不同的语言输出格式，那么就可以定义一个“entry point group”，不同的插件就可以注册“entry point”，插件注册的示例：

```
setup(
    # ...
    entry_points = {'blogtool.parsers': ['.rst = some_module:a_func']}
)
# 或者
setup(
    # ...
    entry_points = """
        [blogtool.parsers]
        .rst = some.nested.module:SomeClass.some_classmethod [reST]
    """,
    extras_require = dict(reST = "Docutils>=0.3.5")
)
```

Setuptools中的dependency_links

Setuptools有一个功能叫做 dependency_links

from setuptools import setup

```
setup(
    # ...
    dependency_links = [
        "http://packages.example.com/snapshots/",
        "http://example2.com/p/bar-1.0.tar.gz",
    ],
)
```

这一功能除去了依赖的抽象特性，直接把依赖的获取url标在了setup.py里。就像在Go语言中修改依赖包一样，我们只需要修改依赖链中每个包的 dependency_links 。

管理依赖

我们写依赖声明的时候需要在 setup.py 中写好抽象依赖（install_requires），在 requirements.txt 中写好具体的依赖，但是我们并不想维护两份依赖文件，这样会让我们很难做好同步。requirements.txt 可以更好地处理这种情况，我们可以在有 setup.py 的目录里写下一个这样的 requirements.txt

```
--index https://pypi.python.org/simple/

-e .
```

这样 pip install -r requirements.txt 可以照常工作，它会先安装该文件路径下的包，然后继续开始解析抽象依赖，结合 --index 选项后转换为具体依赖然后再安装他们。

这个办法可以让我们解决一种类似这样的情形：比如你有两个或两个以上的包在一起开发但是是分开发行的，或者说你有一个尚未发布的包并把它分成了几个部分。如果你的顶层的包 依然仅仅按照“名字”来依赖的话，我们依然可以使用 requirements.txt 来安装开发版本的依赖包:

```
--index https://pypi.python.org/simple/

-e https://github.com/foo/bar.git#egg=bar
-e .
```

这会首先从 https://github.com/foo/bar.Git 来安装包 bar ， 然后进行到第二行 -e . ，开始安装 setup 中的抽象依赖，但是包 bar 已经安装过了， 所以 pip 会跳过安装。

Differences between distribute, distutils, setuptools and distutils2

Distutils is the standard tool used for packaging. It works rather well for simple needs, but is limited and not trivial to extend.

Setuptools is a project born from the desire to fill missing distutils functionality and explore new directions. In some subcommunities, it' s a de facto standard. It uses monkey-patching and magic that is frowned upon by Python core developers.

Distribute is a fork of Setuptools that was started by developers feeling that its development pace was too slow and that it was not possible to evolve it. Its development was considerably slowed when distutils2 was started by the same group. 2013-August update: distribute is merged back into setuptools and discontinued.

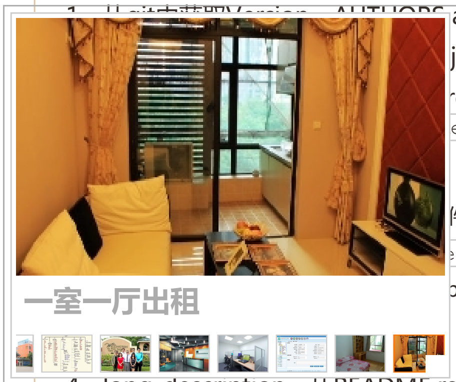
Distutils2 is a new distutils library, started as a fork of the distutils codebase, with good ideas taken from setup tools (of which some were thoroughly discussed in PEPs), and a basic installer inspired by pip. The actual name you use to import Distutils2 is packaging in the Python 3.3+ standard library, or distutils2 in 2.4+ and 3.1–3.2. (A backport will be available soon.) Distutils2 did not make the Python 3.3 release, and it was put on hold.

PBR

pbr 是setuptools的辅助工具，最初是为OpenStack开发(https://launchpad.net/pbr)，基于 d2to1 。

> A library for mana ... ols packaging needs in a consistent manner.

pbr会读取和过滤setup.cfg中的数据，然后将解析后的数据提供给setup.py作为参数。包含如下功能：



1、... and ChangeLog信息
... ject，找到所有模块，生成stub files
... ements.txt，生成setup函数需要的
... ependency_links

... 件的头部可以使用： --index
... / ，这一行把一个抽象的依赖声明如 requests==1.2.0 转变为一个具
... i.python.org/simple/

4、long_description。从README.rst, README.txt or README file中生成 `long_description` 参数

使用pbr很简单：

```
from setuptools import setup

setup(
    setup_requires=['pbr'],
    pbr=True,
)
```

使用pbr时，setup.cfg中有一些配置。在[files]中，有三个key：
`packages` :指定需要包含的包，行为类似于setuptools.find_packages
`namespace_packages` :指定namespace packages
`data_files` : 指定目的目录和源文件路径，一个示例：

```
[files]
data_files =
    etc/pbr = etc/pbr/*
    etc/neutron =
        etc/api-paste.ini
        etc/dhcp-agent.ini
    etc/init.d = neutron.init
```

[entry_points]段跟setuptools的方式相同。

Babel

A collection of tools for internationalizing Python applications

Babel 是 Python 的一个国际化工具包，提供了对distutils或setuptools的支持，包含一些命令。

1、compile_catalog
类似于msgfmt工具，takes a message catalog from a PO file and compiles it to a binary MO file.

```
$ ./setup.py compile_catalog --directory foobar/locale --locale pt_BR
running compile_catalog
compiling catalog to foobar/locale/pt_BR/LC_MESSAGES/messages.mo
```

2、extract_messages
类似于xgettext，it can extract localizable messages from a variety of difference source files, and generate a PO (portable object) template file from the collected messages.

```
$ ./setup.py extract_messages --output-file foobar/locale/messages.pot
running extract_messages
extracting messages from foobar/__init__.py
extracting messages from foobar/core.py
...
writing PO template file to foobar/locale/messages.pot
```

3、update_catalog
类似于msgmerge，it updates an existing translations catalog based on a PO template file (POT).

setup.py和pip

表面上，`python setup.py install`和`pip install`都是用来安装python包的，实际上，`pip`提供了更多的特性，更易于使用。体现在以下几个方面：

- pip会自动下载依赖，而如果使用setup.py，则需要手动搜索和下载；
- pip会自动管理包的信息，使卸载/更新更加方便和容易，使用`pip uninstall`即可。而使用setup.py，必须手动删除，有时容易出错。
- pip提供了对`virtualenv`更好的整合。

结语

OK，讲了这么多琐碎的东西，现在去看看Nova或Ceilometer的setup脚本，是不是一下清晰了很多？！但说实话，setup.py的使用，我还不能讲的特别清楚，需要在后续的实战中学习。

参考文档：

http://docs.python.org/2/distutils/introduction.html

http://pythonhosted.org/setuptools/

查看评论

1楼 jzhx107 2016-04-13 13:25发表

讲得很好，谢谢！

您还没有登录,请[登录]或[注册]