包管理

为什么使用包管理

Python的模块或者源文件直接可以复制到目标项目目录中，就可以导入使用了。

但是为了更多项目调用使用，或者共享给别人，就需要打包，或发布到网络，以便供人使用。

目的也是为了复用。

Pypi（Python Package Index），公共的模块存储中心，[https://pypi.python.org/pypi](http://https://pypi.python.org/pypi)

主要工具

distutils

官方库distutils，使用安装脚本setup.py 来构建、安装包。

从1998年就是标准库的一部分，直到2000年停止开发。

setuptools

它是替代distutils的增强版工具集，包含easy\_install工具，使用ez\_setup.py文件。支持egg格式的构建和安装。

提供查询、下载、安装、构建、发布、管理等包管理功能。

setuptools是包管理的核心模块。

后来，setuptools开发缓慢了，出现基于setuptools的distribute来替代setuptools。2013年，这两个项目重新合

并，distribute被废弃，setuptools依然是Python安装打包的标准方式。

pip

pip目前包管理的事实标准。

构建在setuptools之上，替代easy\_install的。同样提供丰富的包管理功能。

Python3.4之前，需要单独安装，从Python3.4开始直接包含在安装文件中。

wheel

wheel格式定义在PEP427中。

wheel文件中不包含.pyc文件。

提供 bdist\_wheel 作为 setuptools 的扩展命令，这个命令可以用来生成新打包格式 wheel。

pip 从1.4版本开始 提供了一个 wheel 子命令来安装 wheel 包。当然，需要先安装 wheel 模块。

它可以让Python库以二进制形式安装，而不需要在本地编译。

使用setup.py打包

setup.py创建一个源代码分发包的例子，参照例子 https://docs.python.org/3.5/distutils/setupscript.html。

可以在帮助文档chm上搜索索引setup，点击最上面的Distributing Python Modules (Legacy version) ，然后选

择  Writing  the  Setup  Script

#  包结构

m

|--  \_\_init\_\_.py

|--  m1.py

|--  m2

|--  \_\_init\_\_.py

|--  m21

|--  \_\_init\_\_.py

|--  m22.py

项目根目录下，构建一个setup.py文件，setup.py如下

from  distutils.core  import  setup

#  导入setup函数并传参

setup(name='m',

version='0.1.0',

description='Python  test  m',

author='wayne',

author\_email='wayne@magedu.com',

#url='https://www.python.org/sigs/distutils-sig/',

#packages=['m',  'm.m1',  'm.m2',  'm.m2.m21'],

packages=['m']

)

#  name名字

#  version  版本

#  packages=[]  打包列表，

#  packages=['m'],指定m，就会把m所有的非目录子模块打包

#  ['m',  'm.m1.m2.m3'],逐级建立目录，但是只把m的所有非目录子模块打包，把m.m1.m2.m3打包

#  ['m',  'm.m1',  'm.m1.m2',  'm.m1.m2.m3']

#  description  描述信息

#  author  作者

#  author\_email  作者邮件

#  url  包的主页，可以不写

查询命令的帮助

$ setup.py --help [cmd1 cmd2 ...]

$ python setup.py --help-commands

$ setup.py cmd --help

build命令，编译

创建一个build目录

$ python setup.py build

以下是packages=['m']配置的结果

running  build

running  build\_py

creating  build

creating  build\lib

creating  build\lib\m

copying  m\m1.py  ->  build\lib\m

copying  m\\_\_init\_\_.py  ->  build\lib\m

在项目目录下多了build目录，有一个lib子目录，lib下就是模块m的目录了。

m目录下的\*.py文件被复制了，但是子目录没有被复制。

以下是packages=['m.m2.m21']配置的结果

running  build

running  build\_py

creating  build

creating  build\lib

creating  build\lib\m

creating  build\lib\m\m2

creating  build\lib\m\m2\m21

copying  m\m2\m21\\_\_init\_\_.py  ->  build\lib\m\m2\m21

可以看出，逐级构建了同样的目录结构，并只拷贝了m21的  \_\_init\_\_.py  文件

以下是packages=['m', 'm.m2.m21']配置的结果

running  build

running  build\_py

creating  build\lib\m

creating  build\lib\m\m2

creating  build\lib\m\m2\m21

copying  m\m2\m21\\_\_init\_\_.py  ->  build\lib\m\m2\m21

copying  m\m1.py  ->  build\lib\m

copying  m\\_\_init\_\_.py  ->  build\lib\m

build得到的文件，直接拷贝到其他项目就可以用

install命令，安装

build后就可以install，直接运行

$  python  setup.py  install

如果没有build，会先build编译，然后安装。

sdist命令，分发

sdist命令

$  python  setup.py  sdist

创建源代码的分发包。



产生一个dist目录，里面生成一个带版本号的压缩包。

在其他地方解压缩这个文件，里面有setup.py，就可以使用   $  python  setup.py  install   安装了， 也可以  $  pip

install  m-0.1.0.zip  直接使用pip安装这个压缩包。

$  python  setup.py  bdist\_wininst  #  制作windows下的分发包

$  python  setup.py  bdist\_rpm  #  打包成rpm

可以把自己写好的模块发布到公共的Pypi上，也可以搭建Pypi私服，供企业内部使用。

Pypi里面的模块没有太好的审核机制，不保证安全，请慎重使用。

wheel包

安装wheel依赖

$  pip  install  wheel

setup.py修改如下

#  from  distutils.core  import  setup  #  可能失败

from  setuptools  import  setup

setup(name='m',

version='0.1',

description='m  module',

author='Wayne',

author\_email='wayne',

url='http://www.magedu.com',

packages=['m',  'm.m2.m21',  'm.m2'],

)

python  setup.py  bdist\_egg

python  setup.py  bdist\_wheel