链滴

Python包管理工具setuptools详解

作者：zhuangyan

原文链接：https://ld246.com/article/1423985716890

来源网站：链滴

<h3 id="jdsit">0.什么是setuptools</h3>
setuptools是Python distutils增强版的集合，它可以帮助我们更简单的创建和分发Python包，其是拥有依赖关系的。用户在使用setuptools创建的包时，并不需要已安装setuptools，只要一个启动模块即可。
功能亮点：

利用EasyInstall自动查找下载、安装、升级依赖包
创建Python Eggs
包含包目录内的数据文件
自动包含包目录内的所有的包，而不用在setup.py中列举
自动包含包内和发布有关的所有相关文件，而不用创建一个MANIFEST.in文件
自动生成经过包装的脚本或Windows执行文件
支持Pyrex，即在可以setup.py中列出.pyx文件，而最终用户无须安装Pyrex
支持上传到PyPI
可以部署开发模式，使项目在sys.path中
用新命令或setup()参数扩展distutils，为多个项目发布/重用扩展
在项目setup()中简单声明entry points，创建可以自动发现扩展的应用和框架

总之，setuptools就是比distutils好用的多，基本满足大型项目的安装和发布
<h3 id="ycbdu">1.安装setuptools</h3>
1) 最简单安装，假定在ubuntu下
<pre><code>sudo apt-get install python-setuptools
</code></pre>
2) 启动脚本安装
<pre><code>wget http://peak.telecommunity.com/dist/ez_setup.py
sudo python ez_setup.py
</code></pre>
<h3 id="nouyx">2.创建一个简单的包</h3>
有了setuptools后，创建一个包基本上是无脑操作
<pre><code>cd /tmp
mkdir demo
cd demo
</code></pre>
在demo中创建一个<code>setup.py</code>文件，写入
<pre><code>from setuptools import setup, find_packages
setup(
    name = "demo",
    version = "0.1",
    packages = find_packages(),
)
</code></pre>
执行<code>python setup.py bdist_egg</code>即可打包一个test的包了。
<pre><code>demo
|-- build
| `-- bdist.linux-x86_64
|-- demo.egg-info
| |-- dependency_links.txt
| |-- PKG-INFO
| |-- SOURCES.txt
| `-- top_level.txt
</code></pre>

|-- dist
| `-- demo-0.1-py2.7.egg
`-- setup.py
</code></pre>
在dist中生成的是egg包
<pre><code>file dist/demo-0.1-py2.7.egg
dist/demo-0.1-py2.7.egg: Zip archive data, at least v2.0 to extract
</code></pre>
看一下生成的.egg文件，是个zip包，解开看看先
<pre><code>upzip -l dist/demo-0.1-py2.7.egg
Archive: dist/demo-0.1-py2.7.egg
Length   Date   Time   Name
<hr>
<pre><code class="highlight-chroma"><span class="highlight
cl">   1 2013-06-07 22:03   EGG-INFO/dependency_links.txt
    1 2013-06-07
2:03   EGG-INFO/zip-safe
  120 2013-06-07
22:03   EGG-INFO/SOURCES.txt
    1 2013-06-07
2:03   EGG-INFO/top_level.txt
  176 2013-06-07
22:03   EGG-INFO/PKG-INFO
</code></pre>
<hr>
<pre><code class="highlight-chroma"><span class="highlight
cl"> 299   5 files
</code></pre>
</code> <code></code> </pre>
我们可以看到，里面是一系列自动生成的文件。现在可以介绍一下刚刚setup()中的参数了

name 包名
version 版本号
packages 所包含的其他包

要想发布到PyPI中，需要增加别的参数，这个可以参考<a href="https://ld246.com/forward?goto=http%3A%2F%2Fpythonhosted.org%2Fsetuptools%2Fsetuptools.html%23basic-use" targe
="_blank" rel="nofollow ugc">官方文档中的例子了。
<h3 id="mflhy">3.给包增加内容</h3>
上面生成的egg中没有实质的内容，显然谁也用不了，现在我们稍微调色一下，增加一点内容。<
p>
在demo中执行<code>mkdir demo</code>，再创建一个目录，在这个demo目录中创建一个
code>__init__.py</code>的文件，表示这个目录是一个包，然后写入：
<pre><code>#!/usr/bin/env python
#-*- coding:utf-8 -*-
def test():
print "hello world!"
</code> <code>if name == 'main':
test()
</code> </pre>
现在的主目录结构为下：
<pre><code>demo
|-- demo
| `-- __init__.py

```
`-- setup.py
```

再次执行`python setup.py bdist_egg`后，再看egg包

```
Archive:  dist/demo-0.1-py2.7.egg
  Length      Date    Time    Name
---------  ---------- -----   ----
        1  2013-06-07 22:23   EGG-INFO/dependency_links.txt
        1  2013-06-07 22:23   EGG-INFO/zip-safe
      137  2013-06-07 22:23   EGG-INFO/SOURCES.txt
        5  2013-06-07 22:23   EGG-INFO/top_level.txt
      176  2013-06-07 22:23   EGG-INFO/PKG-INFO
       95  2013-06-07 22:21   demo/__init__.py
      338  2013-06-07 22:23   demo/__init__.pyc
---------                     -------
      753                     7 files
```

这回包内多了demo目录，显然已经有了我们自己的东西了，安装体验一下。

```
python setup.py install
```

这个命令会讲我们创建的egg安装到python的dist-packages目录下，我这里的位置在

```
tree /usr/local/lib/python2.7/dist-packages/demo-0.1-py2.7.egg
```

查看一下它的结构：

```
/usr/local/lib/python2.7/dist-packages/demo-0.1-py2.7.egg
|-- demo
|   |-- __init__.py
|   `-- __init__.pyc
`-- EGG-INFO
    |-- dependency_links.txt
    |-- PKG-INFO
    |-- SOURCES.txt
    |-- top_level.txt
    `-- zip-safe
```

打开python终端或者ipython都行，直接导入我们的包

```
&gt;&gt;&gt; import demo
&gt;&gt;&gt; demo.test()
hello world!
&gt;&gt;&gt;
```

好了，执行成功！

4.setuptools进阶

在上例中，在前两例中，我们基本都使用setup()的默认参数，这只能写一些简单的egg。一旦我的project逐渐变大以后，维护起来就有点复杂了，下面是setup()的其他参数，我们可以学习一下

使用find_packages()

对于简单工程来说，手动增加packages参数很容易，刚刚我们用到了这个函数，它默认在和setu.py同一目录下搜索各个含有`__init__.py`的包。其实我们可以将包统一放在一个src录中，另外，这个包内可能还有aaa.txt文件和data数据文件夹。

```
demo
├── setup.py
└── src
    └── demo
        ├── __init__.py
```

```
        ├── aaa.txt
        └── data
            ├── abc.dat
            └── abcd.dat
</code></pre>
<p>如果不加控制，则setuptools只会将<code>__init__.py</code>加入到egg中，想要将这些文都添加，需要修改<code>setup.py</code></p>
<pre><code>from setuptools import setup, find_packages
setup(
    packages = find_packages('src'),  # 包含所有src中的包
    package_dir = {'':'src'},   # 告诉distutils包都在src下
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight cl">package_data = {
</span></span><span class="highlight-line"><span class="highlight-cl">    # 任何包中含有.t t文件，都包含它
</span></span><span class="highlight-line"><span class="highlight-cl">    '': ['*.txt'],
</span></span><span class="highlight-line"><span class="highlight-cl">    # 包含demo包d ta文件夹中的 *.dat文件
</span></span><span class="highlight-line"><span class="highlight-cl">    'demo': ['data/* dat'],
</span></span><span class="highlight-line"><span class="highlight-cl">}
</span></span></code></pre>
</code><p><code>)<br>
</code></p></pre><p></p>
<p>这样，在生成的egg中就包含了所需文件了。看看：</p>
<pre><code>Archive:  dist/demo-0.0.1-py2.7.egg
  Length     Date   Time    Name
 --------    ----   ----    ----
       88  06-07-13 23:40   demo/__init__.py
      347  06-07-13 23:52   demo/__init__.pyc
        0  06-07-13 23:45   demo/aaa.txt
        0  06-07-13 23:46   demo/data/abc.dat
        0  06-07-13 23:46   demo/data/abcd.dat
        1  06-07-13 23:52   EGG-INFO/dependency_links.txt
      178  06-07-13 23:52   EGG-INFO/PKG-INFO
      157  06-07-13 23:52   EGG-INFO/SOURCES.txt
        5  06-07-13 23:52   EGG-INFO/top_level.txt
        1  06-07-13 23:52   EGG-INFO/zip-safe
 --------                   -------
      777                   10 files
</code></pre>
<p>另外，也可以排除一些特定的包，如果在src中再增加一个tests包，可以通过exclude来排除它,<p>
<pre><code>find_packages(exclude=["*.tests", "*.tests.*", "tests.*", "tests"])
</code></pre>
<h4 id="entrypoints">使用entry_points</h4>
<p>一个字典，从entry point组名映射道一个表示entry point的字符串或字符串列表。Entry point是用来支持动态发现服务和插件的，也用来支持自动生成脚本。这个还是看例子比较好理解：</p>
<pre><code>setup(
    entry_points = {
        'console_scripts': [
            'foo = demo:test',
            'bar = demo:test',
        ],
```

```
        'gui_scripts': [
            'baz = demo:test',
        ]
    }
)
```

修改`setup.py`增加以上内容以后，再次安装这个egg，可以发现在安装信息里多了两行代码（Linux下）：

```
Installing foo script to /usr/local/bin
Installing bar script to /usr/local/bin
```

查看`/usr/local/bin/foo`内容

```
#!/usr/bin/python
# EASY-INSTALL-ENTRY-SCRIPT: 'demo==0.1','console_scripts','foo'
__requires__ = 'demo==0.1'
import sys
from pkg_resources import load_entry_point
if name == 'main':
sys.exit(
load_entry_point('demo==0.1', 'console_scripts', 'foo')()
)
```

这个内容其实显示的意思是，foo将执行console_scripts中定义的foo所代表的函数。执行foo，现打出了`hello world!`，和预期结果一样。

使用Eggsecutable Scripts

从字面上来理解这个词，Eggsecutable是Eggs和executable合成词，翻译过来就是另eggs可执。也就是说定义好一个参数以后，可以另你生成的.egg文件可以被直接执行，貌似Java的.jar也有这制？不很清楚，下面是使用方法：

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

这么写意味着在执行`python *.egg`时，会执行我的test()函数，在文档中说需将.egg放到PATH路径中。

包含数据文件

在3中我们已经列举了如何包含数据文件，其实setuptools提供的不只这么一种方法，下面是另两种

1）包含所有包内文件

这种方法中包内所有文件指的是受版本控制（CVS/SVN/GIT等）的文件，或者通过MANIFEST.i声明的

```
from setuptools import setup, find_packages
setup(
    ...
    include_package_data = True
)
```

2）包含一部分，排除一部分

```
from setuptools import setup, find_packages
setup(
```

```
    ...
    packages = find_packages('src'),
    package_dir = {'':'src'},
<pre><code class="highlight-chroma"><span class="highlight-line"><span class="highlight
cl">include_package_data = True,
</span></span><span class="highlight-line"><span class="highlight-cl">
</span></span><span class="highlight-line"><span class="highlight-cl"># 排除所有 READ
E.txt
</span></span><span class="highlight-line"><span class="highlight-cl">exclude_package_
ata = { '': ['README.txt'] },
</span></span></code></pre>
</code><p><code>)<br>
</code></p></pre><p></p>
<p>如果没有使用版本控制的话，可以还是使用3中提到的包含方法</p>
<h4 id="section-3">可扩展的框架和应用</h4>
<p>setuptools可以帮助你将应用变成插件模式，供别的应用使用。官网举例是一个帮助博客更改输
类型的插件，一个博客可能想要输出不同类型的文章，但是总自己写输出格式化代码太繁琐，可以借
一个已经写好的应用，在编写博客程序的时候动态调用其中的代码。</p>
<p>通过entry_points可以定义一系列接口，供别的应用或者自己调用，例如：</p>
<pre><code>setup(
    entry_points = {'blogtool.parsers': '.rst = some_module:SomeClass'}
)
<p>setup(<br>
entry_points = {'blogtool.parsers': ['.rst = some_module:a_func']}<br>
)</p>
</code><p><code>setup(<br>
entry_points = """<br>
[blogtool.parsers]<br>
.rst = some.nested.module:SomeClass.some_classmethod [reST]<br>
""",<br>
extras_require = dict(reST = "Docutils&gt;=0.3.5")<br>
)<br>
</code></p></pre><p></p>
<p>上面列举了三中定义方式，即我们将我们some_module中的函数，以名字为blogtool.parsers
借口共享给别的应用。</p>
<p>别的应用使用的方法是通过<code>pkg_resources.require()</code>来导入这些模块。</p>
<p>另外，一个名叫<a href="https://ld246.com/forward?goto=http%3A%2F%2Fstevedore.rea
thedocs.org%2Fen%2Flatest%2Findex.html" target="_blank" rel="nofollow ugc">stevedore</
>的库将这个方式做了封装，更加方便进行应用的扩展。</p>
<p>&amp;nbsp;</p>
```