

## 10.15 分发包

### 问题

你已经编写了一个有用的库，想将它分享给其他人。

### 解决方案

如果你想分发你的代码，第一件事就是给它一个唯一的名字，并且清理它的目录结构。例如，一个典型的函数库包会类似下面这样：

```
projectname/  
  README.txt  
  Doc/  
    documentation.txt  
  projectname/  
    __init__.py  
    foo.py  
    bar.py  
    utils/  
      __init__.py  
      spam.py  
      grok.py  
  examples/  
    helloworld.py  
  ...
```

要让你的包可以发布出去，首先你要编写一个 `setup.py`，类似下面这样：

```
# setup.py  
from distutils.core import setup  
  
setup(name='projectname',  
      version='1.0',  
      author='Your Name',  
      author_email='you@youraddress.com',  
      url='http://www.you.com/projectname',  
      packages=['projectname', 'projectname.utils'],  
)
```

下一步，就是创建一个 `MANIFEST.in` 文件，列出所有在你的包中需要包含进来的非源码文件：

```
# MANIFEST.in  
include *.txt  
recursive-include examples *  
recursive-include Doc *
```

确保 `setup.py` 和 `MANIFEST.in` 文件放在你的包的最顶级目录中。一旦你已经做了这些，你就可以像下面这样执行命令来创建一个源码分发包了：

```
% bash python3 setup.py sdist
```

它会创建一个文件比如“projectname-1.0.zip”或“projectname-1.0.tar.gz”，具体依赖于你的系统平台。如果一切正常，这个文件就可以发送给别人使用或者上传至 [Python Package Index](#)。

### 讨论

对于纯Python代码，编写一个普通的 `setup.py` 文件通常很简单。一个可能的问题是你必须手动列出所有构成包源码的子目录。一个常见错误就是仅仅只列出一个包的最顶级目录，忘记了包含包的子组件。这也是为什么在 `setup.py` 中对于包的说明包含了列表 `packages=['projectname', 'projectname.utils']`

大部分Python程序员都知道，有很多第三方包管理器供选择，包括`setuptools`、`distribute`等等。有些是为了替代标准库中的`distutils`。注意如果你依赖这些包，用户可能不能安装你的软件，除非他们已经事先安装过所需要的包管理器。正因

如此，你更应该时刻记住越简单越好的道理。最好让你的代码使用标准的Python 3安装。如果其他包也需要的话，可以通过一个可选项来支持。

对于涉及到C扩展的代码打包与分发就更复杂点了。第15章对关于C扩展的这方面知识有一些详细讲解，特别是在15.2小节中。