

Simplistix

Python Package Management Sucks

Chris Withers

Who am I?

- Chris Withers
- Independent Zope and Python Consultant
- Using Zope and Python since 1999

- 80% of what I do is web-based

- I talk about the elephant in the room...

Excuse the rough edges...

...this preparation time for this talk was cut down by problems with python package management!

<rant>

...and it's early, and I'm probably hungover

</rant>

In The Beginning

- I want to write some code
- I want to share some code
- I want to use someone else's code

Python Imports

- The Python Path
 - PYTHONPATH

```
set PYTHONPATH=C:\somefolder
export PYTHONPATH=/somefolder
```

- sys.path

```
import sys
sys.path.append(os.path.join('', 'somefolder'))
```

- .pth files

- c:\Python25\Lib\site-packages\something.pth

```
C:\somefolder
```

- Anything else?

Modules

mymodule.py

```
def sayHello():  
    print 'Hello'
```

- How do we use someone else's module?
 - just make sure they're on the python path

```
from mymodule import sayHello  
  
sayHello()
```

- How do we share a module?

Packages

- An example:

```
./mypackage
./mypackage/__init__.py
./mypackage/mymodule.py
./mypackage/myothermodule.py

...

./mypackage/readme.txt
./mypackage/tests

...
```

Packages

- How do we use other people's packages?
 - just make sure they're on the python path

```
from mypackage.mymodule import sayHello  
  
sayHello()
```

<rant>

Don't do things on import, especially in `__init__.py`

</rant>

- How do we share packages?
 - maybe .zip or .tgz them up to send to people

Why isn't this enough?

- How do you distribute a package?
 - .zip
 - .tgz
 - .tar
- How do you release new versions?
- What if your package has C extensions?
- Where do you distribute a package?

<rant>

It often is enough!

</rant>

Distutils

- Lets fiddle with our package

```
./mydistro
./mydistro/setup.py
./mydistro/doc
./mydistro/mypackage/__init__.py
./mydistro/mypackage/mymodule.py
./mydistro/mypackage/myothermodule.py
```

```
./mydistro/setup.py
```

```
from distutils.core import setup

setup(
    name='mypackage',
    version='1.0',
    packages=['mypackage']
)
```

Distutils

- How do we use other people's packages?

```
$ tar xzf mypackage-1.0.tar.gz  
$ cd mypackage-1.0  
$ python setup.py install
```

- How do we share our package?

```
$ cd mydistro  
$ python setup.py sdist
```

- But what do we do with the distribution then?

Distutils – Using others' code

<rant>

Building extensions on windows

</rant>

```
prefix/lib/pythonver/distutils/distutils.cfg  
$HOME/.pydistutils.cfg  
setup.cfg
```

```
prefix\Lib\distutils\distutils.cfg  
%HOME%\pydistutils.cfg  
setup.cfg
```

Distutils – Using others' code

<rant>

- Why do I need two copies of everything?
- What version is installed?
- How do I uninstall a package?
- Where are the package's docs?
- What if I have more than one package?
 - incompatible versions

</rant>

Distutils – Using others' code

<rant>

Where does this package come from?

</rant>

```
from mypackage.mymodule import sayHello  
  
sayHello()
```

- depends on various paths
 - PYTHONPATH
 - .pth files
 - sys.path manipulation

Distutils – Sharing code

<rant>

I don't like this layout!

</rant>

```
./mydistro  
./mydistro/setup.py  
./mydistro/doc  
./mydistro/mypackage/__init__.py  
./mydistro/mypackage/mymodule.py  
./mydistro/mypackage/myothermodule.py
```

Distutils – Sharing code

<rant>

How do I package the rest of my files?

</rant>

- How do I distribute my package?
- How do I release new versions?
- How do I specify dependencies?

Your Pain?

- What have you suffered with Distutils that I missed?

EasyInstall – using others' code

- Installing EasyInstall

```
# wget http://peak.telecommunity.com/dist/ez_setup.py  
# python ez_setup.py
```

- Installing a package

```
# easy_install mypackage
```

- wow, that was easy!

EasyInstall – using others' code

```
# easy_install mypackage
```

- Where does this look?
 - following links in html is silly!
- Why does it need to be run as root?
- How can I specify a version?
 - What if the version I want isn't available as a Win32 egg?
- Why would I want an alpha or beta?

EasyInstall – using others' code

```
# easy_install mypackage
```

- What happens with dependencies?
 - what happens when you install packages separately?
- What if I have different projects that need different versions of a package?
 - especially if they're hidden in dependencies

EasyInstall – using others' code

```
# easy_install mypackage
```

- Where does stuff end up?
 - site packages
- How does that end up working?

```
site-packages/easy-install.pth
```

```
import sys; sys.__plen = len(sys.path)
./mypackage-1.0-py2.5.egg
...
import sys; new=sys.path[sys.__plen:]; del sys.path[sys.__plen:];
p=getattr(sys, '__egginsert', 0); sys.path[p:p]=new; sys.__egginsert =
p+len(new)
```

EasyInstall – using others' code

<rant>

Why the .pth nastiness?

</rant>

- Silly format of .egg dir
- Silly support for multiple versions of eggs

EasyInstall – using others' code

- Ever tried debugging through an easy_installed package?

```
File '.../tmpT7sHL7/Paste-1.7.1-py2.4.egg/  
    paste/exceptions/errormiddleware.py',  
line 144 in __call__
```

- What does easy_install do?
 - automatic compilation
 - copy to final location
- How do I fix?
 - blow away all .pyc in final location

EasyInstall – using others' code

- What is a namespace package?

```
./zope
./zope/__init__.py
./zope/component/
./zope/component/__init__.py
..
./zope/interface/
./zope/interface/__init__.py
```

- What does `easy_install` do?

```
site-packages/zope.component-3.5.1-py25.egg/zope/component/
site-packages/zope.component-3.5.1-py25.egg/zope/component/__init__.py
..
site-packages/zope.interface-3.4.1-py25.egg/zope/interface/
site-packages/zope.interface-3.4.1-py25.egg/zope/interface/__init__.py
```

EasyInstall – using others' code

```
# easy_install mypackage
```

- Looks easy, huh?

<rant>

All the pain of distutils
...with added mysticism

</rant>

SetupTools – Sharing Code

- How do I distribute my package?
- How do I release new versions?
- How do I specify dependencies?

SetupTools – Sharing Code

- Lets fiddle with our package

```
./mydistro/setup.py
```

```
from setuptools import setup, find_packages
setup(
    name='mypackage',
    version='1.0',
    author='Chris Withers',
    author_email='chris@simplistix.co.uk',
    license='MIT',
    description="A package",
    long_description='No really, a *package*',
    url='http://example.com/mypackage',
    classifiers=[ ... ],
    packages=find_packages(),
    zip_safe=False,
    include_package_data=True,
    install_requires=[
        'elementtree>=1.0',
        'zope.interface<=5.0',
    ],
)
```

SetupTools – Sharing Code

- How do I distribute my package?
- How do I release new versions?

```
$ cd mydistro/tags/1.0  
$ python setup.py sdist bdist_egg register upload
```

- Too easy?

SetupTools – Other Stuff

- **scripts**

```
from setuptools import setup, find_packages

setup(
    ...
    entry_points = {
        'console_scripts': [
            'foo = mypackage.mymodule:sayHello',
        ],
        'gui_scripts': [
            'bar = mypackage.mymodule:sayHello',
        ]
    }
)
```

- generates a script in global `Scripts`
- runs `sayHello()` in `mypackage.mymodule`

SetupTools – Other Stuff

- declaring optional features

```
./mydistro/setup.py
```

```
from setuptools import setup, find_packages

setup(
    ...
    extras_require = {
        'PDF': ["ReportLab"],
    })
```

```
./mydistro2/setup.py
```

```
from setuptools import setup, find_packages

setup(
    ...
    install_requires = [
        'mypackage [PDF]',
    ])
```

SetupTools – Other Stuff

- where does easy_install look?
 - PyPI
 - `easy_install --index-url`
 - `easy_install --find-links`
- **setuptools lets developers be evil**

```
from setuptools import setup, find_packages

setup(
    ...
    dependency_links = [
        "http://peak.telecommunity.com/snapshots/"
    ]
})
```

Where do I find packages?

- Python Package Index
 - PyPI, **not** PyPy!
- <http://pypi.python.org/pypi>
- Designed for human use
 - both uploading and downloading
- Also used by `setuptools` & `easy_install`

PyPI Problems

- no dependency information
- no ratings
- no date of activity shown
- weird auto-hiding stuff
- not reliable

- not designed to be used by automated tools
 - “simple” index is an afterthought
- painful to get any changes to software made

Your Pain?

- What have you suffered with that I missed?

Why is this a Python Problem?

- Surely we can just use the OS package management stuff?
- All Package Managers are not created equal
- Everything needs to be re-packaged for each one
- Packages are “Python's Plugins”
 - so it is a Python Problem!

What can we do about this?

- insulate each project
- make sure dependencies are met
- make sure a projects can easily reproduce their setup

VirtualEnv

- Create a new environment

```
$python virtualenv.py --no-site-packages myenv
```

- Enter the new environment

```
$cd myenv  
$source bin/activate  
  
C:\myenv> Scripts\activate
```

VirtualEnv

- Using your new environment

```
(myenv) C:\myenv>python  
(myenv) C:\myenv>easy_install
```

- Leaving the environment

```
(myenv) C:\myenv>deactivate
```

VirtualEnv Problems

- still just a python setup
 - same problems with versions
- bloat on disk due to isolation
- copies existing Libs

Buildout

- getting started with buildout

```
$wget \  
http://svn.zope.org/*checkout*/zc.buildout/trunk/bootstrap/bootstrap.py
```

```
buildout.cfg
```

```
[buildout]  
parts =
```

- bootstrap your buildout

```
$cd myproject  
$python bootstrap.py  
$bin/buildout
```

Buildout

- what you get

```
./myproject
./bootstrap.py
./buildout.cfg

./bin
./bin/buildout

./develop-eggs
./eggs
./parts
```

Buildout

- eggs example

buildout.cfg

```
[buildout]
parts = myeggs
prefer-final = true

[myeggs]
recipe = zc.recipe.egg
interpreter = mypy
eggs =
    BeautifulSoup
    elementtree
```

```
$bin/buildout
...
$bin/mypy
```

Buildout

- generating scripts

```
./mydistro/setup.py
```

```
from setuptools import setup, find_packages

setup(
    name='mypackage',
    version='1.1',
    packages=find_packages(),
    entry_points={
        'console_scripts': ['hello = mypackage.mymodule:sayHello']
    }
)
```

```
$bin/buildout
...
$bin/hello
```

Buildout

- generating scripts

buildout.cfg

```
[buildout]
parts = myeggs
prefer-final = true

[myeggs]
recipe = zc.recipe.egg
interpreter = mpy
eggs = mypackage
entry-points = hello2=mypackage.mymodule:sayHello
```

```
$bin/buildout
...
$bin/hello2
```

Buildout

- controlling where eggs come from
- set a different index

buildout.cfg

```
[buildout]
parts = myeggs
prefer-final = true
index = C:\myindex

find-links =
  http://www.example.com/myeggs
  C:\more_eggs
  C:\somefolder\someegg-1.0-py2.5.egg

...
```

Buildout Problems

- still can't tell what's installed
 - lock down with versions section

buildout.cfg

```
[buildout]
parts = myeggs
prefer-final = true
versions = versions

[versions]
mypackage = 1.0
zope.interface = 3.4.1

...
```

Buildout Problems

- storage bloat

```
$home/.buildout/default.cfg
```

```
[buildout]  
eggs-directory = ~/my-egg-cache
```

- make sure you create the directory first!

Buildout Problems

- starts with your base python
 - make sure it's sterile!
- millions of import paths

```
./bin/hello
```

```
#!/"python"

import sys
sys.path[0:0] = [
    'eggs/mypackage-1.0-py2.5.egg',
    'eggs/zope.interface-3.3.0.egg',
    'eggs/elementtree-1.2.6_20050316-py2.5.egg',
    'eggs/setuptools-0.6c8-py2.5.egg',
]

import mypackage.mymodule

if __name__ == '__main__':
    mypackage.mymodule.sayHello()
```

Buildout Problems

- documentation
- built on setuptools
- not limited to python packages
- open to recipe abuse
- finding the right recipe
 - zope instances
- a mixture of eggs
 - buildout controls some
 - doesn't know about others

Buildout Questions?



Common Problem Themes

- try to do too much
- build on existing problems
- too many special cases

- What about future unexpected cases?
 - packaging for Google App Engine

What can we do?

- join catalog-sig and campaign for:
 - version dependency metadata
 - ratings

<http://www.python.org/community/sigs/current/catalog-sig>

- join distutils-sig:
 - complain about setuptools
 - complain about buildout
 - offer patches with tests!

<http://www.python.org/community/sigs/current/distutils-sig/>

What can we do?

- join python-dev:
 - complain about .pth files!

<http://mail.python.org/mailman/listinfo/python-dev>

What's still missing?

- What version do I have installed?
- Where is it coming from?
- How do I uninstall?
- Why do I have to learn so much?

Wouldn't it be nice if?

- `pacman install package [version]`
- `pacman uninstall package`
- `pacman help [command]`
- `pacman inspect [path] [package]`

Questions?



Thankyou!

- Chris Withers
- chris@simplistix.co.uk
- <http://www.simplistix.co.uk/software>
- <http://www.simplistix.co.uk/presentations>