Systems Programming in Python - Week 6

Distributing Python Applications

Stand-alone executables

Brian Dorsey
brian@dorseys.org

RANDOM

0_urllib2.py # embed raw

```python
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import urllib2

gh_url = 'https://api.github.com'
gh_user= 'user'
gh_pass = 'pass'

req = urllib2.Request(gh_url)

password_manager = urllib2.HTTPPasswordMgrWithDefaultRealm()
password_manager.add_password(None, gh_url, gh_user, gh_pass)

auth_manager = urllib2.HTTPBasicAuthHandler(password_manager)
opener = urllib2.build_opener(auth_manager)

urllib2.install_opener(opener)

handler = urllib2.urlopen(req)

print handler.getcode()
print handler.headers.getheader('content-type')

# ------
# 200
# 'application/json'
```

Requests: HTTP for Humans

Release v0.4.1. (*Installation*)

Requests is an *ISC Licensed* HTTP library, written in Python, for human beings.

Most existing Python modules for sending HTTP requests are extremely verbose and cumbersome. Python's builtin **urllib2** module provides most of the HTTP capabilities you should need, but the api is thoroughly **broken**. It requires an *enormous* amount of work (even method overrides) to perform the simplest of tasks.

Things shouldn't be this way. Not in Python.

```
>>> r = requests.get('https://api.github.com', auth=('user', 'pass'))
>>> r.status_code
200
>>> r.headers['content-type']
'application/json'
```

http://docs.python-requests.org/

THE PROBLEM

-

You've written
the next great application

How are other people going to get it?

web apps
and other servers

no worries

scripts that just use the standard library?

GUIs?
other modules?
binary dependencies?
platform integration?
installers?

we need some way to deploy stand-alone executables

this is one place where being a scripting language is NOT an advantage

a trick if you're making internal tools for an organization:

share out a full installation of Python on the network with your application

STAND-ALONE EXECUTABLES

bundle tools

your script
all python dependencies
all binary dependencies
trickery

- my application
- dependencies
- py2exe
- InnoSetup

demo

I’m going to show what I know.

there are lots of tools to do this, and py2exe is currently neglected

GOTCHAS

trust no one

you'll need to test your application all over again to be sure it works

bonus points for cross platform testing

automation?

run time environment differences

paths

config files

dlls and other binaries

temp files

“one file” modes add yet another layer of trickery

avoid them if you can

extracting to temp files on each launch, LoadLibrary tricks on Windows, etc

missing dependencies

console app

vs

GUI app

TOOLS

cx_Freeze

- Windows, OS X, Linux
- Python 2.3 - 3.2
- Only Win/Linux choice for Python 3.x
- Zipped eggs OK (probably)

http://cx-freeze.sourceforge.net/

pyInstaller

- Windows, OS X, Linux
- Python 2.2 - 2.7
- Zipped eggs OK

http://www.pyinstaller.org/

py2exe

- Windows only
- Python 2.x
- last release in 2008
- stable and full-featured
- many specialized windows features supported: versions, windows services, etc)
- works great for Windows services
- no zipped eggs (easy_install makes these by default, use -Z to unzip)
- http://www.py2exe.org/old/ - better intro docs than you'll find on the wiki

http://www.py2exe.org/

py2app

- OS X only
- Python 2.5 - 3.2
- Zipped eggs OK

http://packages.python.org/py2app/

bbfreeze

- Windows and Linux
- Python 2.4 - 2.7
- Zipped eggs OK
- (started as a fork of cx_freeze)

http://pypi.python.org/pypi/bbfreeze/

gui2exe

- "GUI2Exe is a Graphical User Interface frontend to all the "executable builders" available for the Python programming language. It can be used to build standalone Windows executables, Linux applications and Mac OS application bundles and plugins starting from Python scripts."

http://code.google.com/p/gui2exe/

RECOMMENDATIONS

-

Windows only?
Doing deep Windows integration?

py2exe

otherwise, try this order?

cx_Freeze

pyInstaller

py2exe / py2app

This is a bit of a guess.

Questions?