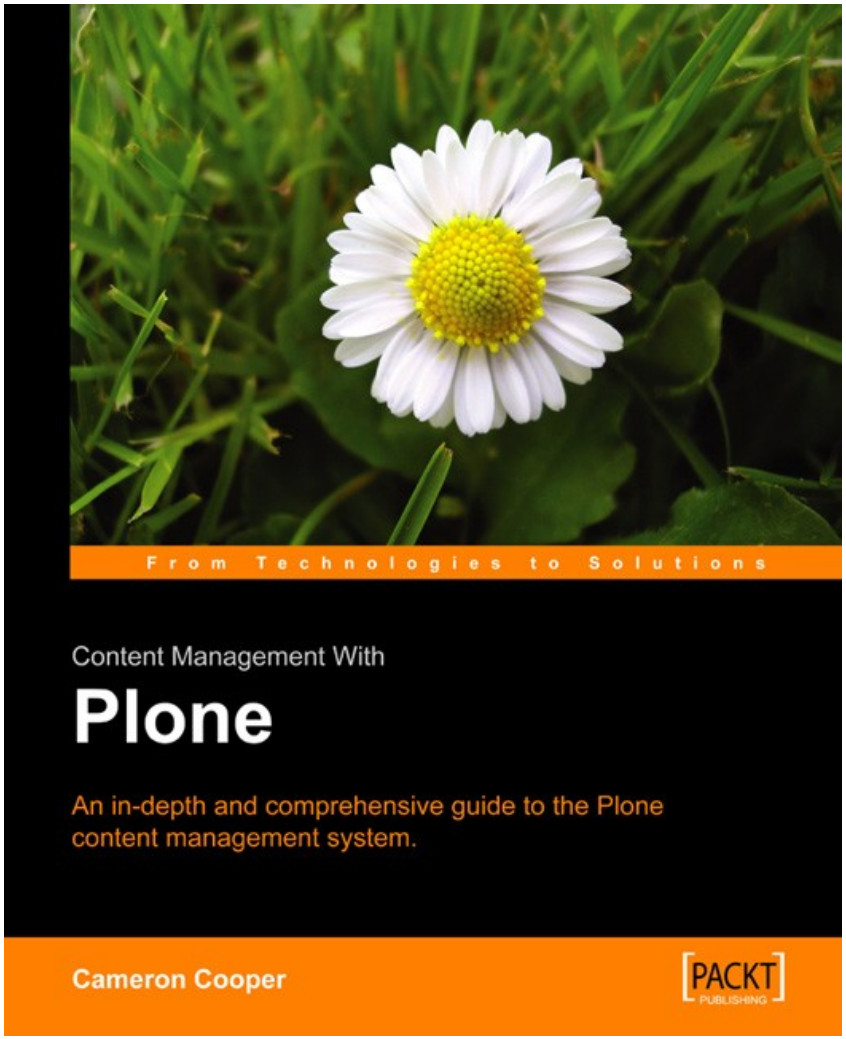
Debugging Zope
NOLA 2006

J. Cameron Cooper

enfoldsystems

Who's this guy?



enfoldsystems

CONNEXIONS

enfoldsystems

Debugging ain't simple

- It's also not just one thing
- We'll look at a few strategies
  - lots of stuff for a few minutes
- This is just an introduction
  - you can see code online
- Lots of other resources out there

enfoldsystems

Tracebacks

- What you get when something fails
- In error_log or event log
- Absolutely essential for debugging
  - Always include the traceback!
  - Always!
  - Seriously.
- Tells you the code path
  - But not the values

Reading Tracebacks

- Bottom up
  - In Zope, the top few lines are chaff
- Example:

```
Traceback (innermost last):
  Module ZPublisher.Publish, line 101, in publish
  Module ZPublisher.mapply, line 88, in mapply
  Module ZPublisher.Publish, line 39, in call_object
  Module Products.CMFCore.FSPythonScript, line 108, in __call__
  Module Shared.DC.Scripts.Bindings, line 306, in __call__
  Module Shared.DC.Scripts.Bindings, line 343, in _bindAndExec
  Module Products.CMFCore.FSPythonScript, line 163, in _exec
  Module None, line 4, in zipupload
   - <FSPythonScript at /Plone/zipupload>
   - Line 4
TypeError: doTransform() takes at least 3 arguments (2 given)
```

Verbose Security

- That "not found" traceback isn't very helpful
- Turn on `verbose-security` in zope.conf
  - Or install VerboseSecurity in < 2.8.3
- Example:

Unauthorized: Your user account does not have the required permission. Access to 'addHelpCenterTutorial' of (__FactoryDispatcher__ instance at 023E57E0) denied. Your user account, tester, exists at /mysite/acl_users. Access requires Add_portal_content_Permission, granted to the following roles: ['Manager', 'Owner']. Your roles in this context are ['Authenticated', 'Member'].

enfoldsystems

When Tracebacks Fail

- You don't always get a traceback
  - Bad behavior, for example
- How did it get there?
  - You don't see the values
- Previously run code is the problem
  - It's already out of the stack, but its side effects remain

enfoldsystems

What's that code doing?

- Think real hard
- Logging/printing
- Parallel execution
- Watch execution live

enfoldsystems

Think real hard

- Mental algorithm prover
  - Use when coding == less debugging
- Not a bad approach
  - until you have to carry too much state
- Can't teach it here
  - May or may not be teachable at all

enfoldsystems

Logging/Printing

- Old school
  - but still effective
- Easiest way to watch lots of iterations
- Two ways:
  - Insert print
    - Watch foreground console
  - Insert log
    - tail -f is good
    - Sometimes can be left in for later use

enfoldsystems

Parallel Execution

- Python has a console: use it!
  - useful for creating algorithms without intervening layers
  - or the restart/run cycle
- Zope "debug" console
  - Same as above, but a live Zope available
- Turns into doctests

enfoldsystems

Watch Execution Live

- pdb!
- Trust me, it's not scary
- We get to step through the code
  - What is it doing?
  - What are the values?
- Can't change course, though

enfoldsystems

Enough Theory Already!

- Let's see how...

enfoldsystems

Logging

- zLOG (deprecated)
  - from zLOG import zLOG, DEBUG
  - zLOG("Module name", DEBUG, "message")
- Python logging
  - from logging import getLogger
  - logger = getLogger(name)
  - logger.debug("message")
- plone_log
  - context.plone_log("message")

enfoldsystems

Debug Console

- You know how to launch a Python console
  - what if you need to interact with Zope?
- Attach a Python console to your ZODB
  - ask it questions
  - poke it
- ZEO makes this easier
  - leave regular instance running
  - but not entirely necessary

enfoldsystems

Unixish Systems

- Just say `zopectl debug`
- It's really that simple
  - No ZEO? Shut down Zope first
  - But you should be running ZEO anyway
    - That's another talk
- Zope root bound to `app`

enfoldsystems

Windows

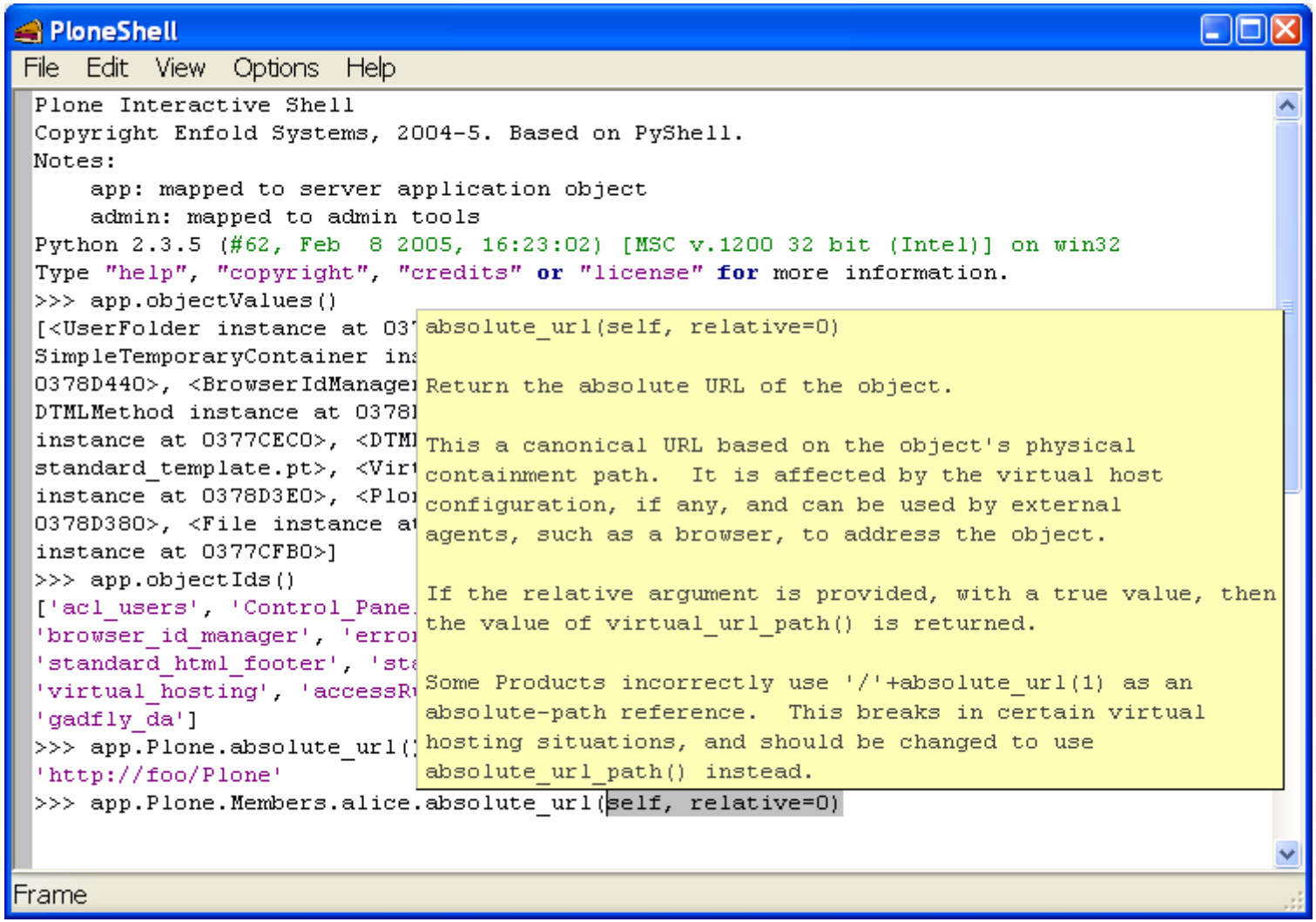
- zopectl unavailable on Windows
- Plone Shell
  - enfoldsystems.com/Products/OpenPloneShell
  - same idea, with other nifty features
- Install wxPython in Zope Python path
  - site-packages
- Install PloneShell similarly
- `C:\Program Files\Plone 2\Python>python.exe Lib\site-packages\Shell\start.py ..\Data`

enfoldsystems

What do we have here?

- Python prompt
- Handle on Zope root
  - same data as though the web
  - call methods, get or set attributes
- We're in a transaction
  - nothing we do is permanent unless we commit
  - changes to other instances don't come here unless we ask

Example



enfoldsystems

Plone Shell Goodies

- Highlighting
- Call tips
  - show what parameters a method takes
- Auto completion
  - there are ways to get this for general console
- Based on PyShell (part of PyCrust in wxPython)

enfoldsystems

What can I do with this?

- Extract values unavailable TTW
- Call methods unavailable TTW
- Administrative tasks
- Shadow existing code to see what it does
  - or to test changes
- Develop algorithms
- Write doctests
- Anything!

Dealing with the Transaction

- It's a sandbox
- Aborted on close; changes lost
- To commit, say
  - import transaction; transaction.commit()
  - *deprecated version:* get_transaction().commit()
  - *force abort:* transaction.abort()
- To update:
  - app._p_jar.sync()

enfoldsystems

Security

- You're not any particular user
- Only explicit security checks will notice
- To become someone, say:
  - from AccessControl.SecurityManagement import newSecurityManager
  - alice = app.acl_users.getUser('alice')
  - newSecurityManager(alice)

enfoldsystems

Up Close and Personal

- We can also watch execution in progress
- No simulation! Live data! Introspection!
- Watching only; no manipulation.
- Attach a debugger at a certain point

enfoldsystems

pdb

- Standard Python debugger
  - some IDEs will do the same thing
- See Python docs for `pdb` package
- Starts working only at specific "break points"
  - no need to look at all the overhead code
- Many ways to set a break point
- Almost always will use
  - import pdb; pdb.set_trace()

enfoldsystems

But where?

- Before whatever code you want to watch
  - tracebacks will often provide a clue
- Sometimes in other people's code
  - break the "never touch external code" rule
  - it's okay, it's not permanent
- NOT in Python Scripts; use zdb
- Don't forget to remove it!
  - "Hey, why's Zope hanging?"

enfoldsystems

Trigger the code

- Reload the web browser
- In debug console:
  - import Zope
  - Zope.debug(url, u=user)
    - simulates a request
- Postmortem mode launches with exception
  - Zope.debug(url, u=user, pm=1)
  - don't have to insert trace point
  - can't step through code, but can see values

enfoldsystems

Using pdb

- pdb starts when it hits a break point
- You get a prompt like:
  - (Pdb)
- Then you can type various commands
- Say ? or see Python lib docs for list of commands
  - most commands have long form and short form (one letter)

enfoldsystems

Usually...

- r(eturn) to get out of the set_trace method
- w(here) to get a non-error stack trace
- l(ist) to show the surrounding code
- n(ext) or s(tep) to go to the next line
  - step goes inside functions, next doesn't
- c(ontinue) to run uninterrupted until the next break point
- Name of any variable to see its value

enfoldsystems

Example

- Simple application ZipDealer
- Calculates prefix of zip file contents
- But when we get a certain file, it fails
- We put in a pdb break point to watch the code

enfoldsystems

So what happens?

- As we watch, we see prefix is not calculated with a 1 element file.
  - A simple oversight.
- We can add a new case to fix this.

Advanced Topics

- Hope we have time...

enfoldsystems

Preemptive Strike

- Automated tests!
  - They will save your life.
  - And make it easier.
  - Seriously.
- You can even attach pdb in a failing test
- Keep tabs on a third party resource with daily runs
- Writing tests? That's another talk.

enfoldsystems

Debugging Spinning Process

- Also, long-running process
- DeadlockDebugger product
  - threadframe Python module
- Visit special URL (manage_debug_threads)
- Shows current stack of each thread
- Run several times
  - same thing? Suspicious!
- Turn off when you're done!

enfoldsystems

Transaction Contents

- What have we modified in the transaction
- During transaction
  - t = get_transaction()
  - t._objects
- After commit
  - fsdump
    - provides hex oids like 0000000000000001
  - map to object with
    - o = app._p_jar['\x00\x00\x00\x00\x00\x00\x01']

enfoldsystems

Broken objects

- POSKeyError?
  - locate with fsrefs.py, and get object like previously with oid
  - walk tree until something throws POSKey
- "Load state" error also provides uid
  - To turn hex number (0x0a4b) into string:
    - from ZODB.utils import p64
    - p64(0x45aa) == '\x00\x00\x00\x00\x00\x00E\xaa'
- container.manage_delObjects(id)

enfoldsystems

That's it

- Any questions?