## Contents

1 Dotted Name Resolution 3

2 zope.dottedname API 5

3 Hacking on zope.dottedname 7
   3.1 Getting the Code ........................................... 7
   3.2 Working in a virtualenv ...................................... 7
   3.3 Using zc.buildout ........................................... 9
   3.4 Using tox .................................................. 9
   3.5 Contributing to zope.dottedname ............................ 10

4 Indices and tables 13
zope.dottedname provides one function, resolve(), that resolves strings containing dotted names into the appropriate Python object.

Dotted names are resolved by importing modules and by getting attributes from imported modules. Names may be relative, provided the module they are relative to is supplied.

Here are some examples of importing absolute names:

```python
>>> from zope.dottedname.resolve import resolve

>>> resolve('unittest')
<module 'unittest' from '...'>

>>> resolve('datetime.datetime')(2015, 2, 2, 18, 59, 27)
datetime.datetime(2015, 2, 2, 18, 59, 27)

>>> resolve('datetime.datetime.now')
<built-in method now of type object at ...>

>>> resolve('non existent module')
Traceback (most recent call last):
  ...
ImportError: No module named non existent module

>>> resolve('__doc__')
Traceback (most recent call last):
  ...
ImportError: No module named __doc__

>>> resolve('logging.foo')
Traceback (most recent call last):
  ...
ImportError: No module named ...foo
```
Here are some examples of importing relative names:

```python
>>> resolve('.split', 'os.path')
<function split at ...>

>>> resolve('..system', 'os.path')
<built-in function system>

>>> resolve('...datetime', 'os.path')
<module 'datetime' ...>
```

NB: When relative names are imported, a module the name is relative to must be supplied:

```python
>>> resolve('.split').__name__
Traceback (most recent call last):
...
ValueError: relative name without base module
```
CHAPTER 2

zope.dottedname API
Hacking on `zope.dottedname`

### 3.1 Getting the Code

The main repository for `zope.dottedname` is in the Zope Foundation Github repository:

https://github.com/zopefoundation/zope.dottedname

You can get a read-only checkout from there:

```bash
$ git clone https://github.com/zopefoundation/zope.dottedname.git
```

or fork it and get a writeable checkout of your fork:

```bash
$ git clone git@github.com:jrandom/zope.dottedname.git
```

The project also mirrors the trunk from the Github repository as a Bazaar branch on Launchpad:

https://code.launchpad.net/zope.dottedname

You can branch the trunk from there using Bazaar:

```bash
$ bzr branch lp:zope.dottedname
```

### 3.2 Working in a virtualenv

#### 3.2.1 Installing

If you use the `virtualenv` package to create lightweight Python development environments, you can run the tests using nothing more than the `python` binary in a virtualenv. First, create a scratch environment:

```bash
$ /path/to/virtualenv --no-site-packages /tmp/hack-zope.dottedname
```

Next, get this package registered as a “development egg” in the environment:
3.2.2 Running the tests

Run the tests using the build-in setuptools testrunner:

```
$ /tmp/hack-zope.dottedname/bin/python setup.py test
running test
.........
Ran 10 tests in 0.000s
OK
```

If you have the nose package installed in the virtualenv, you can use its testrunner too:

```
$ /tmp/hack-zope.dottedname/bin/easy_install nose
...
$ /tmp/hack-zope.dottedname/bin/nosetests
.........
Ran 10 tests in 0.000s
OK
```

If you have the coverage package installed in the virtualenv, you can see how well the tests cover the code:

```
$ /tmp/hack-zope.dottedname/bin/easy_install nose coverage
...
$ /tmp/hack-zope.dottedname/bin/nosetests --with coverage
running nosetests
.........
Name Stmts Miss Branch BrPart Cover Missing
------- ---------- --------- --------- -------- ----- ----------
zope/dottedname.py 0 0 0 0 0 100%
zope/dottedname/example.py 0 0 0 0 0 100%
zope/dottedname/resolve.py 22 0 8 0 100%

TOTAL 22 0 8 0 100%

Ran 11 tests in 0.000s
OK
```

3.2.3 Building the documentation

```
zope.dottedname uses the nifty Sphinx documentation system for building its docs. Using the same virtualenv you set up to run the tests, you can build the docs:
```
```
$ /tmp/hack-zope.dottedname/bin/easy_install Sphinx
...
$ bin/sphinx-build -b html -d docs/_build/doctrees docs docs/_build/html
...
build succeeded.
```
You can also test the code snippets in the documentation:

```
$ bin/sphinx-build -b doctest -d docs/_build/doctrees docs docs/_build/doctest ...
```

Doctest summary
===============

12 tests
0 failures in tests
0 failures in setup code
build succeeded.

Testing of doctests in the sources finished, look at the results in _build/doctest/output.txt.

### 3.3 Using `zc.buildout`

#### 3.3.1 Setting up the buildout

`zope.dottedname` ships with its own `buildout.cfg` file and `bootstrap.py` for setting up a development buildout:

```
$ /path/to/python2.6 bootstrap.py
...
Generated script '.../bin/buildout'
$ bin/buildout
Develop: '/home/jrandom/projects/Zope/zope.dottedname/.'
...
Generated script '.../bin/sphinx-quickstart'.
Generated script '.../bin/sphinx-build'.
```

#### 3.3.2 Running the tests

Run the tests:

```
$ bin/test --all
Running zope.testing.testrunner.layer.UnitTests tests:
  Set up zope.testing.testrunner.layer.UnitTests in 0.000 seconds.
  Ran 400 tests with 0 failures and 0 errors in 0.366 seconds.
Tearing down left over layers:
  Tear down zope.testing.testrunner.layer.UnitTests in 0.000 seconds.
```

### 3.4 Using `tox`

#### 3.4.1 Running Tests on Multiple Python Versions

`tox` is a Python-based test automation tool designed to run tests against multiple Python versions. It creates a virtualenv for each configured version, installs the current package and configured dependencies into each virtualenv, and then runs the configured commands.

`zope.dottedname` configures the following `tox` environments via its `tox.ini` file:
• The py26, py27, py33, py34, and pypy environments builds a virtualenv with the appropriate interpreter, installs zope.dottedname and dependencies, and runs the tests via python setup.py test -q.

• The coverage environment builds a virtualenv with python2.6, installs zope.dottedname, installs nose and coverage, and runs nosetests with statement coverage.

• The docs environment builds a virtualenv with python2.6, installs zope.dottedname, installs Sphinx and dependencies, and then builds the docs and exercises the doctest snippets.

This example requires that you have a working python2.6 on your path, as well as installing tox:

```
$ tox -e py26
GLOB sdist-make: .../zope.interface/setup.py
py26 sdist-reinst: .../zope.interface/.tox/dist/zope.interface-4.0.2dev.zip
py26 runtests: commands[0]
...........
Ran 10 tests in 0.152s
OK
py26: commands succeeded
```

Running tox with no arguments runs all the configured environments, including building the docs and testing their snippets:

```
$ tox
GLOB sdist-make: .../zope.interface/setup.py
py26 sdist-reinst: .../zope.interface/.tox/dist/zope.interface-4.0.2dev.zip
py26 runtests: commands[0]
...
Doctest summary
12 tests
0 failures in tests
0 failures in setup code
0 failures in cleanup code
build succeeded.
py26: commands succeeded
py27: commands succeeded
py33: commands succeeded
py34: commands succeeded
pypy: commands succeeded
coverage: commands succeeded
docs: commands succeeded
congratulations :)```
Please submit bug reports and feature requests there.

### 3.5.2 Sharing Your Changes

**Note:** Please ensure that all tests are passing before you submit your code. If possible, your submission should include new tests for new features or bug fixes, although it is possible that you may have tested your new code by updating existing tests.

If have made a change you would like to share, the best route is to fork the Github repository, check out your fork, make your changes on a branch in your fork, and push it. You can then submit a pull request from your branch:

```
https://github.com/zopefoundation/zope.dottedname/pulls
```

If you branched the code from Launchpad using Bazaar, you have another option: you can “push” your branch to Launchpad:

```
$ bzr push lp:~jrandom/zope.dottedname/cool_feature
```

After pushing your branch, you can link it to a bug report on Launchpad, or request that the maintainers merge your branch using the Launchpad “merge request” feature.
Indices and tables

- genindex
- modindex
- search