zope.hookable Documentation

Release 4.0

Zope Foundation Contributors

January 28, 2015

Contents

Contents:

CHAPTER 1

Hookable Object Support

`zope.hookable` supports the efficient creation of hookable objects, which are callable objects that are meant to be replaced by other callables, at least optionally.

The idea is to create a function that does some default thing and make it hookable. Later, someone can modify what it does by calling its sethook method and changing its implementation. All users of the function, including those that imported it, will see the change.

```
>>> from zope.hookable import hookable
>>> def f41():
...     return 41
>>> f = hookable(f41)
>>> f.implementation is f.original
True
>>> f()
41
```

We can replace the implementation, without replacing `f`: this means that modules which have already imported `f` will see the hooked version.

```
>>> old = f.sethook(lambda: 42)
>>> f.implementation is f.original
False
>>> old is f41
True
>>> f()
42
>>> f.original()
41
>>> f.implementation()
42
```

We can undo the hook by calling `reset`.

```
>>> f.reset()
>>> f()
41
```

CHAPTER 2

zope.hookable API

Hookable object support

class zope.hookable.**hookable**
Callable objects that support being overridden

reset()
Reset the hook to the original value

sethook()
Set the hook implementation for the hookable object

CHAPTER 3

Hacking on `zope.hookable`

3.1 Getting the Code

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

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

You can get a read-only checkout from there:

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

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

```
$ git clone git@github.com/jrandom/zope.hookable.git
```

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

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

You can branch the trunk from there using Bazaar:

```
$ bzr branch lp:zope.hookable
```

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:

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

Next, get this package registered as a "development egg" in the environment:

```
$ /tmp/hack-zope.hookable/bin/python setup.py develop
```

3.2.2 Running the tests

Run the tests using the build-in `setuptools` testrunner:

```
$ /tmp/hack-zope.hookable/bin/python setup.py test -q
running test
...
----------------------------------------------------------------------
Ran 18 tests in 0.000s

OK
```

The `dev` command alias downloads and installs extra tools, like the `nose` testrunner and the `coverage` coverage analyzer:

```
$ /tmp/hack-zope.hookable/bin/python setup.py dev
$ /tmp/hack-zope.hookable/bin/nosetests
running nosetests
..................................... (lots more dots)
----------------------------------------------------------------------
Ran 18 tests in 0.001s

OK
```

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

```
$ /tmp/hack-zope.hookable/bin/nosetests --with coverage
running nosetests
..................................... (lots more dots)
Name                                 Stmts   Miss  Cover   Missing
------------------------------------------------------------------
zope.hookable                           23      0   100%
------------------------------------------------------------------
TOTAL                                   23      0   100%
----------------------------------------------------------------------
Ran 18 tests in 0.001s

OK
```

3.2.3 Building the documentation

`zope.hookable` 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:

The `docs` command alias downloads and installs Sphinx and its dependencies:

```
$ /tmp/hack-zope.hookable/bin/python setup.py docs
...
$ bin/sphinx-build -b html -d docs/_build/doctrees docs docs/_build/html
...
build succeeded.

Build finished. The HTML pages are in docs/_build/html.
```

You can also test the code snippets in the documentation:

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

Document: index
---------------
```

```
1 items passed all tests:
  13 tests in default
13 tests in 1 items.
13 passed and 0 failed.
Test passed.

Doctest summary
===============
   13 tests
    0 failures in tests
    0 failures in setup code
build succeeded.
Testing of doctests in the sources finished, look at the  \
    results in docs/_build/doctest/output.txt.
```

3.3 Using `zc.buildout`

3.3.1 Setting up the buildout

`zope.hookable` 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/BTK/hookable/.'
...
Generated script '.../bin/sphinx-quickstart'.
Generated script '.../bin/sphinx-build'.
```

3.3.2 Running the tests

You can now 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 702 tests with 0 failures and 0 errors in 0.000 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.hookable` configures the following `tox` environments via its `tox.ini` file:

- The `py26`, `py27`, `py33`, `py34`, and `pypy` environments builds a `virtualenv` with `pypy`, installs `zope.hookable` and dependencies, and runs the tests via `python setup.py test -q`.
- The `coverage` environment builds a `virtualenv` with `python2.6`, installs `zope.hookable` and dependencies, installs `nose` and `coverage`, and runs `nosetests` with statement coverage.
- The `docs` environment builds a virtualenv with `python2.6`, installs `zope.hookable` and dependencies, 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.hookable/setup.py
py26 sdist-reinst: .../zope.hookable/.tox/dist/zope.hookable-4.0.2dev.zip
py26 runtests: commands[0]
...
----------------------------------------------------------------------
Ran 18 tests in 0.001s

OK
_____________________________________ summary ______________________________________
py26: commands succeeded
congratulations :)
```

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

```
$ tox
GLOB sdist-make: .../zope.hookable/setup.py
py26 sdist-reinst: .../zope.hookable/.tox/dist/zope.hookable-4.0.2dev.zip
py26 runtests: commands[0]
...
Doctest summary
===============
13 tests
 0 failures in tests
 0 failures in setup code
 0 failures in cleanup code
build succeeded.
_____________________________________ summary ______________________________________
py26: commands succeeded
py27: commands succeeded
py32: commands succeeded
pypy: commands succeeded
coverage: commands succeeded
docs: commands succeeded
congratulations :)
```

3.5 Contributing to `zope.hookable`

3.5.1 Submitting a Bug Report

`zope.hookable` tracks its bugs on Github:

https://github.com/zopefoundation/zope.hookable/issues

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 Githb 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.hookable/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.hookable/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.

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*

Z

H

R

S

Z