zope.minmax Documentation

Release 2.1

Zope Foundation and Contributors

September 01, 2016

Contents

Contents:

CHAPTER 1

Conflict Resolution using Maximum or Minimum Values

The `zope.minmax.AbstractValue` class provides a super class which can be subclassed to store arbitrary *homogeneous* values in a persistent storage and apply different conflict resolution policies.

The subclasses defined here are resolving the conflicts using always either the maximum or the minimum of the conflicting values.

1.1 Maximum

The `zope.minmax.Maximum` class always resolves conflicts favoring the maximum value. Let's instantiate one object and verify that it satisfies the interface.

```
>>> import zope.minmax
>>> import zope.interface.verify
>>> max_favored = zope.minmax.Maximum()
>>> zope.interface.verify.verifyObject(
...     zope.minmax.interfaces.IAbstractValue, max_favored)
True
```

We can confirm that the initial value is zero.

```
>>> bool(max_favored)
False
>>> print(max_favored.value)
None
```

Now, we can store a new value in the object.

```
>>> max_favored.value = 11
>>> print(max_favored.value)
11
>>> bool(max_favored)
True
```

Or we can use the methods.

```
>>> max_favored.__setstate__(4532)
>>> max_favored.__getstate__()
4532
>>> print(max_favored.value)
4532
>>> bool(max_favored)
True
```

Do notice that using a direct assignment to the value attribute is a more natural use.

1.2 Minimum

The `zope.minmax.Minimum` class always resolves conflicts favoring the minimum value. Again, we instantiate an object and verify that it satisfies the interface.

```
>>> min_favored = zope.minmax.Minimum()
>>> zope.interface.verify.verifyObject(
...     zope.minmax.interfaces.IAbstractValue, min_favored)
True
```

We need a confirmation that the initial value is zero.

```
>>> bool(min_favored)
False
>>> print(min_favored.value)
None
```

Let's populate this one too.

```
>>> min_favored.value = 22
>>> print(min_favored.value)
22
>>> bool(min_favored)
True
```

Or we can use the methods, again.

```
>>> min_favored.__setstate__(8796)
>>> min_favored.__getstate__()
8796
>>> print(min_favored.value)
8796
>>> bool(min_favored)
True
```

Please, notice, again, that using a direct assignment to the value attribute is a more natural use.

1.3 Conflict Resolution

Now, we need to exercise the conflict resolution interface. First for the `zope.minmax.Maximum`:

Let's try differing values larger than the old value.

```
>>> max_favored._p_resolveConflict(max_favored.value, 4536, 4535)
4536
>>> max_favored._p_resolveConflict(max_favored.value, 4573, 4574)
4574
```

What happens when all the values are equal, including the old.

```
>>> max_favored._p_resolveConflict(max_favored.value, 4532, 4532)
4532
```

Notice that when the old value is larger than both the committed and new, it is still disregarded.

```
>>> max_favored._p_resolveConflict(max_favored.value, 4531, 4530)
4531
```

Now, the `zope.minmax.Minimum`:

Let's try differing values smaller than the old value.

```
>>> min_favored._p_resolveConflict(min_favored.value, 8792, 8791)
8791
>>> min_favored._p_resolveConflict(min_favored.value, 8785, 8786)
8785
```

What happens when all the values are equal, including the old.

```
>>> min_favored._p_resolveConflict(min_favored.value, 8796, 8796)
8796
```

Notice that when the old value is smaller than both the committed and new, it is still disregarded.

```
>>> min_favored._p_resolveConflict(min_favored.value, 8798, 8799)
8798
```

How about an example that is not numerical?

```
>>> max_word = zope.minmax.Maximum('joy')
>>> print(max_word.value)
joy
>>> bool(max_word)
True
>>> max_word._p_resolveConflict(max_word.value, 'happiness', 'exuberance')
'happiness'
>>> max_word._p_resolveConflict(max_word.value, 'exuberance', 'happiness')
'happiness'
>>> min_word = zope.minmax.Minimum(max_word.value)
>>> print(min_word.value)
joy
>>> bool(min_word)
True
>>> min_word._p_resolveConflict(min_word.value, 'happiness', 'exuberance')
'exuberance'
>>> min_word._p_resolveConflict(min_word.value, 'exuberance', 'happiness')
'exuberance'
```

As indicated, we don't need to have numbers, just *homegeneous* items. The homogeneous values are not really inherently required. However, it makes no sense to apply min() or max() on, say, one number and one string. Simply, the ordering relations do not work at all on heterogeneous values.

CHAPTER 2

zope.minmax API

2.1 Interfaces

CHAPTER 3

Hacking on `zope.minmax`

3.1 Getting the Code

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

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

You can get a read-only checkout from there:

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

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

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

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

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

You can branch the trunk from there using Bazaar:

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

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.minmax
```

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

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

3.2.2 Running the tests

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

```
$ /tmp/hack-zope.minmax/bin/python setup.py test
running test
.........
----------------------------------------------------------------------
Ran 9 tests in 0.000s

OK
```

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

```
$ /tmp/hack-zope.minmax/bin/easy_install nose
...
$ /tmp/hack-zope.minmax/bin/nosetests
.........
----------------------------------------------------------------------
Ran 18 tests in 0.000s

OK
```

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

```
$ /tmp/hack-zope.minmax/bin/easy_install nose coverage
...
$ /tmp/hack-zope.minmax/bin/nosetests --with coverage
running nosetests
..................
Name                        Stmts   Miss Branch BrPart  Cover   Missing
-----------------------------------------------------------------------
zope/minmax.py                  1      0      0      0   100%
zope/minmax/_minmax.py         22      0      2      0   100%
zope/minmax/interfaces.py       2      0      0      0   100%
-----------------------------------------------------------------------
TOTAL                          25      0      2      0   100%
----------------------------------------------------------------------
Ran 18 tests in 0.027s

OK
```

3.2.3 Building the documentation

`zope.minmax` 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.minmax/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
===============
   42 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.minmax` ships with its own `buildout.cfg` file and `bootstrap.py` for setting up a development buildout:

```
$ /path/to/python2.7 bootstrap.py
...
Generated script '.../bin/buildout'
$ bin/buildout
Develop: '/home/jrandom/projects/Zope/zope.minmax/.'
...
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.minmax` configures the following `tox` environments via its `tox.ini` file:

- The `py26`, `py27`, `py33`, `py34`, `pypy`, and ``pypy3` environments build a `virtualenv` with the appropriate interpreter, installs `zope.minmax` and dependencies, and runs the tests via `python setup.py test -q`.
- The `coverage` environment builds a `virtualenv` with `python2.6`, installs `zope.minmax`, installs `nose` and `coverage`, and runs `nosetests` with statement coverage.
- The `docs` environment builds a virtualenv with `python2.6`, installs `zope.minmax`, 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 9 tests in 0.152s

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.interface/setup.py
py26 sdist-reinst: .../zope.interface/.tox/dist/zope.interface-4.0.2dev.zip
py26 runtests: commands[0]
...
Doctest summary
===============
  42 tests
   0 failures in tests
   0 failures in setup code
   0 failures in cleanup code
build succeeded.
_____________________________________ summary ______________________________________
py26: commands succeeded
py27: commands succeeded
py33: commands succeeded
py34: commands succeeded
pypy: commands succeeded
coverage: commands succeeded
docs: commands succeeded
congratulations :)
```

3.5 Contributing to `zope.minmax`

3.5.1 Submitting a Bug Report

`zope.minmax` tracks its bugs on Github:

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