smother Documentation

Release 0.1.0

Chris Beaumont

Aug 12, 2018

Contents

Smother is a tool to measure "who tests what" in a Python test suite. It uses coverage.py to track coverage separately for each test. This can be used to:

- Lookup which tests visit a particular line of sourcecode.
- Perform regression test selection – given a diff of local modifications, enumerate which tests in a lengthy test suite might have broken.
- Explore the amount of coupling between test and application code.
- Accelerate fancier test techniques like mutation testing.

CHAPTER 1

Quick Tour

Here's a tour of using smother on smother's own test suite

```
# runs test suite, builds a .smother report
> py.test --smother=smother

# which tests ran line 153 of module smother.python?
> smother lookup smother.python:153

smother/tests/test_cli.py::test_lookup[module]
smother/tests/test_cli.py::test_lookup[name]
smother/tests/test_cli.py::test_lookup[range]
smother/tests/test_cli.py::test_lookup[single]
smother/tests/test_cli.py::test_semantic_flatten
...

# which tests visited the function PythonFile.line_count in that module?
> smother lookup smother.python:PythonFile.line_count

smother/tests/test_cli.py::test_lookup[module]
smother/tests/test_interval.py::test_init_module
smother/tests/test_interval.py::test_module_path

# Hmm I've modified something locally
> git diff
@@ -175,6 +175,7 @@ class PythonFile(object):

     @property
     def line_count(self):
+        assert False, "This is a new bug!"
         return len(self.lines)

# What might have broken?
> smother diff
```

(continues on next page)

(continued from previous page)

```
smother/tests/test_cli.py::test_lookup[module]
smother/tests/test_interval.py::test_init_module
smother/tests/test_interval.py::test_module_path

# Run those tests!
smother diff | xargs pytest  # 3 failures

# Dump the report to a csv of <source context, test context>
smother csv report.csv

# Build a vanilla .coverage file
smother to_coverage && coverage html
```

Contents:

1.1 Collecting Smother Data

Smother is a wrapper around coverage.py, and by itself is pretty dumb. It basically runs coverage as normal, but stores the coverage output for each test separately. This doesn't have a huge impact on the runtime of your test suite, although the size of the report will be larger.

Building a smother report currently requires that your test suite uses nose or pytest.

1.1.1 Installation

```
pip install smother
```

This installs the `smother` command line utility, as well as plugins for nose and pytest.

1.1.2 Smother with pytest

You can invoke `py.test` with a `--smother` keyword listing which modules you want to track smother data for.

```
py.test --smother=my_module
```

You can configure coverage-specific options by specifying a coveragerc file (default is `.coveragerc`, but you can override via the `--smother-config` option).

See `py.test --help` for more keywords

1.1.3 Smother with nose

```
nosetests --smother-package=my_module
```

See `nosetests --help` for more keywords.

1.1.4 Smother with Coverage

Smother **cannot** be enabled the same time as coverage.py – neither will record the correct information given how they compete for python's `settrace` functionality. This means that you cannot run something like

```
py.test --smother=foo --cover=foo`` or ``nosetests --smother-package=foo --cover-
↪package=foo``.
```

Instead, you should run smother by itself, and then produce a coverage file from a smother file

```
smother to_coverage
```

This will create a .coverage file that you can then use normally with coverage.py

1.2 Working with Smother Data

1.2.1 Querying

Once you have build a `.smother` file, you can query it in a variety of ways. It's main usage is to lookup which tests ran a particular section of code. This is what `smother lookup` is for. The lookup command takes a description of a code section in a variety of formats:

```
# module name. Matches the entire module
smother lookup smother.python

# module_name:line_number. Matches 1 line
smother lookup smother.python:50

# line range
smother lookup smother.python:50-60

# module:class_or_function
smother lookup smother.python:PythonFile

# module:nested_class_or_function
smother lookup smother.python:PythonFile.line_count
```

Smother determines which line range each section corresponds to, and prints out all of the tests which visited that region.

1.2.2 Semantic vs Literal Mode

By default, the code sections above are converted to a range of line numbers, and smother looks for tests which also visit these line numbers. However smother also takes a `--semantic` keyword. In semantic mode, regions are expanded into the smallest function, class, or module definition that contains the entire region. For example, consider a particular line in smother's source. The following lines all expand to the same section of code in semantic mode:

```
smother --semantic lookup smother.python:34
smother --semantic lookup smother.python:24-38
smother --semantic lookup smother.python:Visitor.__init__
```

This is primarily useful for diff reporting and CSV output (below), but can be enabled manually in any context.

1.2.3 Diff Reporting

A common use case for smother is to determine, given a code change, which tests might have broken. If the code is in a git repository, smother provides a `diff` command to do this:

```
smother diff
smother diff origin/master
```

The behavior of the diff command is as follows:

- Determine which semantic regions were modified (deleted) in the old version of the code.
- Determine which semantic regions were modified (added) in the new version of the code.
- Take the union of these semantic regions – call this `U`.
- Report which tests visited region `U`, assuming that the smother report was generated against the old version of the code.

Note that semantic mode is implied by the `diff` command.

1.2.4 CSV Dumps

Smother provides a `csv` command to dump each `(source,test)` pair to a CSV file. Each source record will be a line number or, if `--semantic` mode is enabled, a semantic region (module, class, or function block). This can be used to more easily analyze the coupling between app and test logic.

1.2.5 Coverage Reports

the `to_coverage` command converts a *.smother* datafile into a `coverage.py` datafile, for use with `coverage`.

CHAPTER 2

Indices and tables

- genindex
- modindex
- search