

Monday, February 23. 2009

PHP Mutation Testing With MutateMe


Writing a while ago about the problem of shallow testing (when tests appear to verify code is working on the surface but actually fail when you examine them more closely) dragged up some enthusiasm to revive my attempt to write a more publicly accessible library for Mutation Testing (other than the mish mash I concocted for my own use over a year ago).

Mutation Testing for PHP is currently almost non-existent though there are a few pieces of work buried in subversion repositories here and there. None of them however are capable of supporting ALL testing tools leaving many tools out in the cold. I never did like the constant parade of tightly integrated solutions in PHP, so I've been working on a standalone version that can operate with any testing tool once a suitable adapter is available for it.


I've dubbed it "MutateMe" and you can find the git repository at <http://github.com/padraic/mutateme>.

Update #1: The git repo now contains a preliminary SimpleTest adapter supporting both the TextReporter and HtmlReporter - simply pass "--adapter simpletest" on the command line and ensure --testfile contains the relevant group file, e.g. all_tests.php. If you want to do a manual install from git, the main thing is editing the files in /bin - they use similar conventions to PHPUnit so you can use the phpunit and phpunit.bat files for your system as rough examples of what PHP paths are needed to be edited.

Update #2: Added installation instructions for the required runkit extension. Added a Windows binary download, and some simple Linux compilation instructions. Note - do not install from PECL since the current release does not support PHP 5.2.


Update #3: I've discovered the runkit CVS does not yet support static methods. There is a patch by David Sklar so I may independently release a git fork and windows DLL with the necessary changes until the runkit CVS is updated in the future. In the meantime, be aware static methods will likely result in errors - oh, the fun of alpha releases .

. I'll update git later with some changes contributed by Benjamin Eberlei in his MutateMe fork at <http://github.com/beberlei/mutateme/tree>.

Update #4: Those on Linux can now compile a slightly patched version of runkit which supports static methods using <http://www.github.com/padraic/runkit>. This is strictly a bridging fork until a similar patch finds its way into a new runkit release to PECL (though last release was in 2007, so...depends on Sara ).


Update #5: Git repository now contains full static method support. I'll roll out a new alpha release to reflect the changes during tomorrow. Thanks to everyone who gave feedback and even fixes since the initial 0.1.0 alpha!

What is Mutation Testing?

Mutation Testing is basically testing...for tests. It ensures that your tests are truly capable of detecting errors and problems with the source code. It does this by mutating the source code itself (using ext/runkit) in such a way that an error is created in the code. If your tests detect the error, all is well with the world. If your tests do not detect the error...well, you better add a new test that does .

These source code mutations are applied consistently against specific areas. For example, MutateMe can replace a plus sign with a minus sign, or substitute a variable value for the original (e.g. `$state=0` becomes `$state=123`), or pretty much any other mutation or change you can imagine. These mutations are applied one by one, with the tests run after every change. At the end of the process you receive a report on which mutations did not cause a test to fail (in Mutation Testing, passing tests are a bad thing!) including a pretty diff of the mutation itself so you can replicate the errors that no test detected.


Obviously, this is a resource intensive operation since you run the test suite for every possible mutation. Unlike Unit Testing, Mutation Testing runs your tests for every mutation - and there can be a lot of mutations! It's a form of testing best reserved for occasional use, and preferably on another machine so you can let it run in the background while you're doing something else.

The other facet of Mutation Testing is that it is not definitive all the time. It may raise false positives because changing some code might not actually break anything. It may also raise very obvious errors which aren't worth the time testing (otherwise we'd all have 100% code coverage 

)). Nevertheless, properly utilised it can point to areas where testing needs improvement.

Mutation Testing vs Code Coverage

In PHP, there is a persistent lack of focus on test quality. Code Coverage has become one of the primary and most celebrated measures of ensuring test quality even though it's real purpose is to ensure tests execute all source code - i.e. it ensures a test is written which executes certain lines of code, but doesn't inform you whether the test is poorly written and incapable of detecting the problems it's supposed to.

Mutation Testing is a complementary practice which does detect the existence of shallow or low quality tests since it deliberately introduces errors into the source code to see if tests will catch it. It's a very simple notion 

Without Mutation Testing, the quality of a suite of tests is very difficult to assess without in-depth reading of the tests, a complete and godlike knowledge of the underlying source code, and the ability to see flaws in how the tests operate. In other words - you need a highly skilled human to do the task. Any human involvement is bound to be unreliable - and expensive. Just as with unit testing, automated tests are cheaper, faster and far more methodical when properly designed.

Introducing MutateMe

Rising from the ashes of my initial pass, then called PHPMutagen, MutateMe is the rewritten fully-tested version of my Mutation Testing tool for PHP. It is designed to hook into any testing framework (like PHPUnit or PHPT), mutate the underlying source code within the current PHP process (source code is mutated in memory using ext/runkit), and run the relevant tests against the mutated form to check whether they detect the deliberate error.

To install, download the current development release from <http://dev.phpspec.org/MutateMe-0.1.0alpha.tgz>.

You should install MutateMe using the PEAR installer, so from the location of the download run:

```
pear install MutateMe-0.1.0alpha.tgz
```

This will also download and install Text_Diff 1.1.0, and suggests you might consider installing ext/diff from PECL (optional). MutateMe does however require the ext/runkit extension which is the most problematic part - you must NOT install this from PECL! The current PECL release does not support PHP 5.2.

If you are on Windows, you can try a precompiled version that works with PHP 5.2, grab a copy from

http://dev.phpspec.org/php_runkit.dll, then just copy it to your extensions directory for PHP5 and enable it in php.ini.

Under Linux, you need to compile ext/runkit (takes just a few minutes) from CVS:

1. Using your package manager, install autoconf, automake, libtool, bison, flex and re2c to compile PHP modules.
2. Log into the PHP CVS server anonymously using:

```
cvs -d :pserver:cvsread@cvs.php.net:/repository login
```

The password is "phpfi".

3. Grab the latest runkit HEAD using:

```
cvs -d:pserver:cvsread@cvs.php.net:/repository co pecl/runkit
```

4. Compile the extension and copy it into your default PHP extensions directory using (assuming you are not already in the runkit source directory):

```
cd runkit
phpize
./configure
make
make install
```

5. Edit whichever php.ini config file that governs the PHP CLI to include an enabling line like:

```
extension=runkit.so
```

6. Check the output of "php -i" or phpinfo() to ensure the extension is loaded.

MutateMe is now ready for action - well, the alpha version with it's initial payload of 5 possible mutations and a dodgy PHPUnit adapter is



. Unfortunately you'll have to wait a bit longer for the SimpleTest and PHPT adapters.

The command line tool is accessed by calling "mutateme". It accepts a few command line options:

--basedir: The base directory containing the source code and tests (see conventions later)
--srcdir: The directory where the source code to mutate is located
--testdir: The directory where unit tests are maintained

PHPUnit options (subject to improvement):

--testfile: The name of the test file to run (e.g. AllTests.php)
--test: The test class to run contained in the test file (e.g. My_AllTests - the actual class name in that file can differ from the file name)
--adapter: Select a specific adapter (currently phpunit or simpletest)

If you want to minimise options, MutateMe assumes the current directory is the base directory unless otherwise specified. On that assumption, it will attempt (unless options are passed) that source code is held in one of /src, /lib or /library subdirectories of the base directory. It also assumes tests are stored in either the /tests or /specs subdirectory of the base directory, where there exists an AllTests.php test file containing an AllTests class (assuming you're using the PHPUnit adapter which is the default).

Unless you meet these conventions exactly - use the specific options! For example:

```
cd ~/projects/mathlib
mutateme --srcdir ./libs --testdir ./tests --testfile AllTests.php --test
Mathlib_AllTests
```

You should ensure, that before using MutateMe, your test suite is recording no Errors or Failures (Skips and Incompletes are fine though), since this a precondition before Mutation Testing can be performed. You'll be told by MutateMe if it doesn't anyway.

You can then look forward to the results...

```
MutateMe Alpha: Mutation Testing for PHP
All initial checks successful! The mutagenic slime has been activated.
PHPUnit 3.3.14 by Sebastian Bergmann.
.
Time: 0 seconds
OK (1 test, 1 assertion)
.
1 Mutant born out of the mutagenic slime!
1 Mutant exterminated!
No Mutants survived! Muahahahaha!
```

If you see anything different, it's time to see if the issue warrants an additional test to catch that defect if it appears in the future:

MutateMe Alpha: Mutation Testing for PHP

All initial checks successful! The mutagenic slime has been activated.

PHPUnit 3.3.14 by Sebastian Bergmann.

```
.
Time: 0 seconds
OK (1 test, 1 assertion)
.
1 Mutant born out of the mutagenic slime!
```

```
0 Mutants exterminated!
```

```
1 Mutant escaped; the integrity of your suite may be compromised by the
following Mutants:
```

```
1)
```

```
Index: ./Math.php
```

```
=====
```

```
@@ -1 +1 @@
```

```
-return$op1+$op2;
```

```
+return$op1-$op2;
```

```
PHPUnit 3.3.14 by Sebastian Bergmann.
```

```
.
Time: 0 seconds
```

```
OK (1 test, 1 assertion)
```

```
Happy Hunting! Remember that some Mutants may just be Ghosts (or if you want to
be boring, false positives).
```

The mutations enabled are very limited for now. There are four primary mutations:

1. Replacing "+" with "-"
2. Replacing "-" with "+"
3. Replacing TRUE with FALSE
4. Replacing FALSE with TRUE

Before an actual beta release, I'll be adding adapters for SimpleTest and PHPT, and obviously adding a lot more mutations to the mix. The current release is mainly there for those interested to take a look at.

Posted by Pádraic Brady in PHP General, PHP Security at 18:37

this sounds awesome. i will try it asap



Anonymous on Feb 23 2009, 23:26

Interesting indeed. Odd that I had never heard of the whole concept before =)

Will have to try this sometime, and definitely make a mental bookmark of this thing Anonymous on Feb 24 2009, 01:52

Awesome news, I will give it try. Anonymous on Feb 24 2009, 08:51

Padraic, you always have crazy but very interesting ideas.

How do you make that :-0 ? Anonymous on Feb 24 2009, 10:35

I'm a lunatic - can't you tell



I think Mutation Testing has its place in the world, and it's accepts enough options to be way more specific about what it's mutating (choose only part of your source code, and a single test subdirectory). I've always found it pretty handy at least...

If you want to do more reading on the topic do a search for Java's Jester tool. Anonymous on Feb 24 2009, 11:10

I forked MutateMe on GitHub to provide patches I had to make to get it working locally. Other than that, good job!

Just curious: did you look at http://www.phpunit.de/browser/phpunit/branches/feature/mutation_testing/? That is the result of GSoC project from two years ago. Anonymous on Feb 24 2009, 13:50

I knew the GSOC code existed - it's one of those referred to pieces buring in subversion repos elsewhere - but didn't look into it very much. Not that it wouldn't be great, but I needed something that was more interoperable outside just PHPUnit.

There's still work to be done and I really appreciate the fork



. It will be nice to solve the current twitchy behaviour before it gets any more complex (which it will when multi-process test suites, like phpt, need support).

Hopefully nobody minds when I temporarily fork a new runkit extension to roll up some patches not currently in CVS for stuff like static method support (without any E_ERROR bypasses).

Much appreciated, Sebastian. Anonymous on Feb 24 2009, 15:32

Me too. It looks very interesting. Anonymous on Mar 8 2009, 10:27