Wednesday, February 25. 2009

Mutation Testing: MutateMe 0.2.0alpha Released

I love git! After the initial alpha, the git repository was forked on github by Sebastian Bergmann and Benjamin Eberlei. The new release incorporates all changes pulled from their forks, and huge thanks to both for their feedback and fixes. Also, see the revised README (copied in this blog post) with much improved detailed installation and usage instructions included.

You can download the PEAR package at http://dev.phpspec.org/MutateMe-0.2.0alpha.tgz - the source code is maintained in git at http://www.github.com/padraic/mutateme

The other major changes included funtionality to ensure mutated methods retain their class visibility, i.e. whether public, protected or private. In addition, I also implemented optional static method support which is missing from the current Runkit extension CVS. The static support can be implemented in one of two ways.

The preferred option is to compile a new runkit extension using a slightly patched version of the extension from CVS, which I have forked to http://github.com/padraic/runkit. The only change from PECL CVS is to add a patch written by David Sklar a while ago to add static method support to runkit. As suggested on Twitter, I could simply request to maintain runkit but my C skills are rarely exercised so I wouldn't be comfortable doing that. The small fork will do until a new runkit release appears on PECL.

The alternative option, currently applicable mainly to Windows users, is to disable E_STRICT error reporting when using MutateMe. I haven't had time to compile a Windows DLL for the patched version of runkit but I'll get around to it soon. E_STRICT kicks in, since mutated static methods only retain the public/protected/private visibility and don't retain their static nature. Using a method statically, if not flagged as static, results in a deprecation error. This error will not impact anyone using the patched version of runkit.

The final major change was adding support for SimpleTest! It wouldn't be much of a Mutation Testing framework if it only supported PHPUnit, now would it?

Here's the updated README

```
MutateMe is a Mutation Testing framework for PHP5, currently released as an

alpha version for interested developers to assist in offering feedback. It is

not yet fully functional, and stability is a relative term
```

.

```
Requirements:
PHP: 5.2.4, probably less but I haven't checked
PEAR: Text_Diff 1.1.0
Extensions: ext/runkit, and optionally ext/xdiff
Note: Do not install runkit from PECL since the current release does not support
PHP 5.2. Follow the installation instructions below to grab a patched version of
runkit which supports both PHP 5.2 and static methods. Windows users should
download
```

the DLL from the given link - a patched version of the DLL will be made available
soon. If you have installed from PECL delete the module and follow the instructions
below to replace it with a slightly improved patched version.

Installation (with PEAR):

1. Download the latest PEAR packaged release and run the following command:
   pear install MutateMe-0.2.0alpha.tgz
2. Follow the manual instructions below from Step 3 to install the PHP
    runkit extension.

Installation (Manual Installation without PEAR):

1. Copy the contents of /library to a location on the PHP include_path, e.g.
   /usr/share/php.
2. Copy /bin/mutateme and /bin/mutateme.bat to a location on PATH so they are
   accessible from the command line. Edit the contents of each to replace @php_bin@,
   @bin_dir@ with the path to the php binary (e.g. /usr/bin/php) and the path to the
   directory you're putting these scripts respectively.
3. Install the runkit extension but NOT from PECL. Windows users can grab the DLL
   from http://dev.phpspec.org/php_runkit.dll but should disable E_STRICT error
   reporting (I will compile a patched DLL for use with MutateMe soon). Linux users
   should compile runkit from source. I've put up a patched version supporting
   static methods (via a David Sklar patch) and PHP 5.2 (from CVS HEAD in PECL)
   at http://github.com/padraic/runkit
   Use the following commands to compile from source (install git first!):
   git clone git://github.com/padraic/runkit.git runkit
   cd runkit
   phpize
   ./configure
   make
   make install
   Finally, enable the extension in the php.ini file which is loaded for the cli
   environment using:
   extension=runkit.so
   Check the output of "php -i" or phpinfo() to ensure the extension is loaded.

Usage:

MutateMe has just a few necessary command line options:

   --adapter   Adapter to use: phpunit or simpletest. PHPUnit is the default.
   --basedir   Base directory containing source code and tests
   --srcdir   Directory containing source code to mutate
   --testdir   Directory containing the tests to run
   --testfile   Name of the test file to execute, e.g. AllTests.php or all_tests.php
   --test   (PHPUnit only) Name of the test class to run, e.g. MyLibrary_AllTests

If options are omitted, MutateMe will attempt to autodiscover the correct values
using the following conventions:

   1. The base directory is assumed to be the current working directory
   2. The source directory is assumed to be a subdirectory of the base directory
      called either "src", "lib" or "library"

3. The test directory is assumed to be a subdirectory of the base directory
   called either "tests" or "specs"
4. The test file is assumed to be either AllTests.php or all_tests.php
depending
   on the adapter used
5. (PHPUnit only) The test class is assumed to be AllTests

If you cannot use these conventions completely, please use the command line options appropriately.

Example:

```
$ mutateme --adapter phpunit --basedir ./ --srcdir ./src --testdir ./tests \
        --testfile AllTests.php --test MyLib_AllTests
```

In the example above, we could have omitted the --basedir, --srcdir, --testdir and
--testfile options since they meet the requirements of the autodetectable convention as
described above.

Output:

A successful Mutation Test would output something like:

```
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!
```

A failed Mutation Test would indicate a test failed to detect an introduced error in the
source code. This might be a completely spurious failed mutation, since not all introduced errors actually cause problems, but often it will highlight a new condition
that you should a new test for so a future real error will be detected and not completely missed by the test suite:

```
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;
```

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

Posted by PÃ¡draic Brady in PHP General, PHP Security at 13:19

hahaha, I like it. I'll try to test it this week!

but just for fun, or maybe with ZF, later!

regards! Anonymous on Feb 25 2009, 14:43

Just bear in mind it only supports 5 mutations so far (see the last blog entry). I'll add more for a near future beta release which will bump up it's usefulness to new heights  . Any testing is hugely welcome - as an alpha I expect there to be small issues to resolve and the feedback to date has been of great assistance. Anonymous on Feb 25 2009, 15:26