Thomas McLennan

# Black Box Testing

We have tested each of our classes during implementation.

We started writing tests in the first week of implementation.

We can run our tests online (please don’t, it puts a heavy load on the server).

PHPUnit

PHP was first released by Rasmus Ledorf in 1995 (21year ago). Initially, PHP was not a fully fleshed-out language, and did not have *full* object-oriented capabilities until version 5 (13 July, 2004).

PHPUnit was also released in 2004.

PHPUnit is a command line tool for unit-testing php code, it allows both black-box and white-box testing.

Testing stats

We have **5083** lines of code in our project.

We have **754** lines of test code.

Our ratio of lines-of-test-code:implementation-code is **0.15**.

An industrial project would have a 1:1 ratio or higher.

We run 414 assertions on our tests.

Black-box Test Case Examples

Things we tested:

**The MySQL Database Password**: it is manually set on each of our computers and the server to that it doesn’t get posted on GitHub for the public to see. We tested it for :

|  |  |
| --- | --- |
| Password is string | Passed |
| Password’s first character is alpha-numeric | Passed |
| Password is longer than 1 character | Passed |

**Every field** in the database was tested, an example of our **Member table’s number index**:

|  |  |
| --- | --- |
| member\_name is created | Passed |
| member\_name can be "m" | Passed |
| member\_name can be "1" | Passed |
| member\_name can be "xxxx xxxx xxxx xxxx xxxx" | Passed |
| member\_name cannot be "xxxx xxxx xxxx xxxx xxxx xxxx" | Passed |
| member\_name cannot be "" | Acceptable |

We found that instead of throwing an error, in some cases it was better to attempt to fix user-input, for example the **province name field** in our *member* and *provider* database tables.

|  |  |
| --- | --- |
| member\_province is created | True |
| member\_province can be "BC" | True |
| member\_province cannot be "British Columbia" | Acceptable (BC) |
| member\_province cannot be "CB" | true |
| member\_province cannot be "B.C." | Acceptable |
| member\_province cannot be "**M**anito**b**a" | Acceptable (MB) |
| member\_province cannot be "Que" | Acceptable |

If someone enters “**M**anito**b**a”, “MB” is inserted into the database instead of throwing an error. This is because Manitoba is the only province with first letter “M” that has a “b” in its name.

if ($this->getLength($province) > self::PROVINCE\_LENGTH) //p<2

{

$province = strtoupper($province);

if ($province[0] === "A") $province = "AB";

if ($province[0] === "B") $province = "BC";

if ($province[0] === "C") $province = "BC";

if ($province[0] === "M") $province = "MB";

if ($province[0] === "N" && strpos($province, 'B' !== false)) $province = "NB";

if ($province[0] === "N" && strpos($province, 'L' !== false)) $province = "NL";

if ($province[0] === "N" && strpos($province, 'U') !== false) $province = "NU";

if ($province[0] === "N" && $this->getLength($province) > self::PROVINCE\_LENGTH) $province = "NS";

if ($province[0] === "N" && strpos($province, 'T' !== false)) $province = "NT";

if ($province[0] === "O") $province = "ON";

if ($province[0] === "C") $province = "QC";

if ($province[0] === "S") $province = "SK";

if ($province[0] === "Y") $province = "YT";

}

We tested every field in our database. (member\_status pictured)

|  |  |
| --- | --- |
| **Database.member\_status** | **Result** |
| member\_status is created | true |
| member\_status can be "A" | true |
| member\_status can be "S" | true |
| member\_status cannot be "Suspended" | acceptable |
| member\_status cannot be "Active" | acceptable |
| member\_status cannot be "a" | acceptable |
| member\_status cannot be "s" | acceptable |
| member\_status cannot be "0" | true |
| member\_status cannot be "null" | true |

Other **general tests** we ran.

| **Test** | **Result** |
| --- | --- |
| Asserts SERVER[REMOTE\_ADDR] environmental variable is set (it tells the server to actually run in *production* mode, and not output any source code in error messages). | true |
| Test PHP's MySQLi plugin loads (We use it to connect to the MySQL database). | true |
| Check that the Mysqli Plugin exists (but potentially hasn't been started) | true |
| Test that the calculation in one of a getLength function returns the right length. (Private function Person()->getLength() ). | true |