My Google Summer of Code Experience

Alexandra Livadas

July 30, 2019

My GSOC Project

Some Links: INCF Project Idea 12 | INCF GSOC Page | My GSOC Project Page

What is GSOC?

- 14,000+ students, 109 countries, 651 organizations
- INCF (International NeuroInformatics Coordinating Facility) has 18 projects this year!
- Other organizations: TensorFlow, Ruby, Python, Open Robotics, Jenkins, git,

My GSOC Project

The Project Idea and My Proposal:

- Improve on existing tests
- Improve codebase Testability
- Incrementation of unit tests and code coverage
- Documentation

Summer Roadmap:

May/Beginning of June: Setting up LORIS and getting to know the code!

June and July: Coding (and Release Testing)!

August: Coding and documentation!

August 23: My Last Day

LORIS' Pre-existing Coverage

PHP libraries:

~10k lines of actual code

9 / 51 libraries had partial unit tests

- Outdated + small % of methods covered

Modules:

~21k lines of actual code

Modules have test plans and integration tests written

- Mostly frontend testing

Writing Tests

My Open PRs:

- #4987, #4988 : Adding two unit tests for the **Visit, VisitController** and **Settings** libraries
- #4979 : Unit tests for the **User** library
- #4936 : Adding to unit tests for the **Candidate** library
- #4916 : Adding to unit tests for the **LorisForm** library
- #4861 : Unit tests for the **Utility** library

My Merged PRs:

- #4840 : Unit tests for the **BreadcrumbTrail** library
- #4769 : Unit tests for the **Breadcrumb** library

Writing Tests

Total # of Tests Written (so far!) : 194

Total # of Methods Covered : 114

PHP Libraries covered: 9

Lines of Code Covered: 1973

Lines of Code Written: 4554 (based on my PRs!)

Documentation and Next Steps

Some Links: Testing Guide | Testing Log

Roadmap for August:

- Get my open PRs merged!
- Write tests for some final libraries
- Finish up the Testing Guide
- Create testing plans and roadmaps for the remaining libraries

Code Smells 💩

Small things:

- Mismatched return types
- Abstraction and modularity in functions
- Functions that have many uses

Code Smells 💩

There are 2 ways of declaring the database object!

Version 2 has to be tested differently and it is harder to mock the database if it is declared this way!

Examples from the Utility class, 11 lines apart :
Version 1 | Version 2

Version 1

```
$factory = NDB_Factory::singleton();
$DB      = $factory->database();
```

Version 2

```
$DB =& \Database::singleton();
```

```
106     static function getConsentList(): array
107     {
108         $factory = NDB_Factory::singleton();
109         $DB      = $factory->database();
110
111         $query = "SELECT ConsentID, Name, Label FROM consent";
112         $key   = "ConsentID";
113
114         $result = $DB->pselectWithIndexKey($query, array(), $key);
115
116         return $result;
117     }
118     /**
119      * Returns a list of sites in the database
120      *
121      * @param bool $study_site If true only return sites that are
122      *                         study sites according to the psc
123      *                         table
124      *
125      * @return array an associative array("center ID" => "site name")
126      */
127     static function getSiteList(bool $study_site = true): array
128     {
129         $DB =& \Database::singleton();
130
```

An example of this code smell (php/libraries/Utility.class.inc)!

Writing Tests in the Future!

Database mocking

Method 1

Example: PR #4861

Pros:

- Good way to test whether the correct query is called
- Easy to set up

Cons:

- If the second way of declaring the database is used, you need to create a “Fake” class!

```
$query  = "SELECT ID, Visit_label FROM session
    WHERE CandID=:Candidate AND Active='Y' ORDER BY ID";
$result = $db->pselect($query, array('Candidate' => $candID));
```

```
//mock pselect from getListOfVisitLabels
$this->_dbMock->expects($this->at(3))
    ->method('pselect')
    ->with(
        $this->stringStartsWith('SELECT ID, Visit_label FROM session'),
        $this->arrayHasKey('Candidate')
    )
    ->willReturn($selectReturns);
```

Database mocking

Method 2

Example: PR #4979

Pros:

- Works for both methods of declaring the database

Cons:

- Less flexible and harder to set up
- Uses a “real” database, so the unit tests will take longer to compile or set-up

```
// get user data from database
$query = "SELECT users.*,
    GROUP_CONCAT(psc.Name ORDER BY psc.Name SEPARATOR ';') AS Sites
    FROM users
```

```
$this->_dbMock->setFakeTableData(
    "users",
    array(0 => $this->_userInfo)
);
```

If you want to change anything in the fake table:

```
$this->_dbMock->run("DROP TEMPORARY TABLE users");
```

Final Tips!

- Test-driven Development...
- Or something a little less heavy too!
  - Short methods usually have short unit tests
- Check out my Testing Guide
  - How to set up your testing environment
  - How-tos for specific method types
  - Tips for specific problems I ran into

Feedback for GSOC in the future!

- Better set up documentation
  - Especially for remote GSOC students
- A guide to common errors or questions
  - Stack Overflow
  - Wiki

A question for you all...

Thank you for listening and for a great summer so far!

Race to 21!