web2project
unit testing strategy

Trevor Morse
trevor.morse@gmail.com

web2project

- Forked from dotproject in 2007
- 4 core developers
- ~ 153k lines of code - http://www.ohloh.net/p/web2project
- Runs on LAMP stack, working on WIMP
- 1 year ago had no unit tests, currently have 255

Why Unit Tests?

- Aging code base that needs re-factoring
- Make re-factoring safe
- Make current code base more reliable
- Possibly set up future where bug fixes require unit test upon submission, letting us know the bug is fixed and stays fixed
- TDD in the future
- I wanted to get involved in open source, had experience with web2project and unit testing.

Unit Testing Challenges

- Lots of database interactivity
- Older code base relies on use of global variables, which makes it hard to test
- All code already written, without unit tests in mind

Unit Testing Solutions

- Lots of database interactivity
  - Use PHPUnit's database extensions
- Older code base relies on use of global variables, which makes it hard to test
  - Configure PHPUnit to use global variables. Not ideal but works for now
- All code already written, without unit tests in mind
  - Put head down and just write them, learn as I go

Use PHPUnit DB Extension

Upsides

- Allows easy interaction with the db use xml or csv files to define table structure
- Can have the database in same state at beginning of every test
- Easy comparison of data for testing without having to write sql

Downsides

- Destructive, completely wipes out tables when loading data for tests
- Can be slow, depends on your database
- How to test fields that are date/time stamped with static xml file?

Code to test

PHP Code

```
<?php
class Project
{

    public function create($program_name)
    {
        $this->project_id = $this->db->execute('INSERT INTO
           project(project_name) VALUES($program_name)');

        return $this->project_id;
    }
}
```

Database Structure

```
| Field        | Type              |
+--------------+-------------------+
| project_id   | int(10) unsigned |
| project_name| varchar(255)      |
```

Use PHPUnit DB Extension

Update Your Test Suite and add functions

```
require_once 'PHPUnit/Framework.php';
require_once 'PHPUnit/Extensions/Database/TestCase.php';

class Projects_Test extends PHPUnit_Extensions_Database_TestCase
{

    protected function getConnection()
    {
        $pdo = new PDO('mysli:host=localhost;dbname=dbname,'dbuser',
           'dbpass'));
        return $this->createDefaultDBConnection($pdo, 'dbname');
    }

    protected function getDataSet()
    {
        return $this->createXMLDataSet('projectsSeed.xml');
    }
}
```

Use PHPUnit DB Extension

Use XML Dataset to compare with whats in db

```
// ...
public function testCreateProject()
{
    $project = new Project();

    $project_id = $project->create('Test Project');

    $this->assertEquals(2, $project_id);

    $xml_dataset = $this->createXMLDataSet('testCreate.xml');
    $this->assertTablesEqual(
        $xml_dataset->getTable('projects'),
        $this->getConnection()->createDataSet()->getTable
       ('projects')
    );
}
// ...
```

Use PHPUnit DB Extension

Seed file gets loaded before test suite runs

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- projectsSeed.xml - Seed data for projects test. -->
<dataset>
    <table name="project">
        <column>project_id</column>
        <column>project_name</column>
        <row>
            <value>1</value>
            <value>Project 1</value>
        </row>
    </table>
</dataset>
```

Use PHPUnit DB Extension

File to test after calling createProject

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- testCreate.xml – Data to test after project created. -->
<dataset>
    <table name="project">
        <column>project_id</column>
        <column>project_name</column>
        <row>
            <value>1</value>
            <value>Project 1</value>
        </row>
        <row>
            <value>2</value>
            <value>Test Project</value>
        </row>
    </table>
</dataset>
```

Use PHPUnit DB Extension

Filter fields you do not want to test

```
// ...
public function testCreateProject()
{
    $project = new Project();

    $project_id = $project->create('Test Project');

    $this->assertEquals(2, $project_id);

    $xml_dataset = $this->createXMLDataSet('testCreate.xml');
    $xml_filtered_dataset = new
       PHPUnit_Extensions_Database_DataSet_DataSetFilter(
            $xml_file_dataset,
            array('projects' => array('project_created',
               'project_updated')
            )
        );
    // ...
```

Use PHPUnit DB Extension

```
    // ...
    $xml_db_dataset = $this->getConnection()->createDataSet();
    $xml_db_filtered_dataset = new
       PHPUnit_Extensions_Database_DataSet_DataSetFilter(
            $xml_db_dataset,
            array('projects' =>
                array('project_created', 'project_updated')
            )
        );

    $this->assertTablesEqual($xml_file_filtered_dataset->getTable
        ('projects'), $xml_db_filtered_dataset->getTable
        ('projects')
    );
}
// ...
```

Use PHPUnit DB Extension

Protect users from ruining there database

```
<?xml version="1.0"?>
<project name="web2project" basedir="." default="warning">
  <target name="warning">
    <echo message="Running Unit Tests is a destructive process and
will drop/restore your database multiple times." />
    <echo message="Please only run Unit Tests against your development
or test databases." />
    <echo message="To run tests, use the command: phing run-tests" />
  </target>
  <target name="run-tests">
    <!-- Do actual tests here -->
  </target>
</project>
```

Configure PHPUnit To Use Globals

Update Test Code to not handle globals and declare them

```
global $global_var;

$global_var = true;

require_once 'PHPUnit/Extensions/Database/TestCase.php';
require_once 'PHPUnit/Extensions/Database/DataSet/DataSetFilter.php';

class Projects_Test extends PHPUnit_Extensions_Database_TestCase
{

    protected $backupGlobals = FALSE;

    // ...
}
```

Configure PHPUnit To Use Globals

Make sure you reset globals for following tests

```
// ...

    public function testSomething()
    {
        $old_global_var = $global_var;
        $global_var     = false;

        $something = $this->project->doSomething();

        $this->assertEquals(10, $something);

        $global_var = $old_global_var;
    }
// ...
```

What The Future Holds

- Test Driven Development
- Removing global objects
- Mock objects for DB related stuff
- Selenium for user experience testing

Questions?

Links

- http://www.web2project.net
- http://www.phpunit.de
- http://github.com/trevormorse/web2project
- http://seleniumhq.org