O'REILLY®
OSCON™
Open Source Convention

Untestable Code

Sebastian Bergmann

http://thePHP.cc

July 23rd 2009

Who I am

thePHP.cc

- Sebastian Bergmann
- Involved in the PHP project since 2000
- Creator of PHPUnit
- Co-Founder and Principal Consultant with thePHP.cc

”The secret in testing is

in writing testable code”

- Miško Hevery

What makes code untestable?

thePHP.cc

Most people say

- Make things private
- Use final keyword
- Write long methods
- ...

Real issues

- Mix new with logic
- Look for things
- Work in constructor
- Singletons
- Global state
- Static methods

This slide contains material by Miško Hevery

Constructor Issues

```php
<?php
class Document {
    private $html;

    public function __construct($url) {
        $client = new HttpClient;
        $this->html = $client->get($url);
    }
}
?>
```

Mixing object graph construction with work

Doing work in the constructor

Constructor Issues


```
<?php
class Document {
    private $html;

    public function __construct($url) {
        $client = new HttpClient;
        $this->html = $client->get($url);
    }
}
?>
```

- A `Document` object cannot be isolated from its dependency on an `HttpClient` object as we cannot pass a *Test Double* to the constructor
- Support for *Class Posing* in PHP could help
- **But we should solve the design problem**

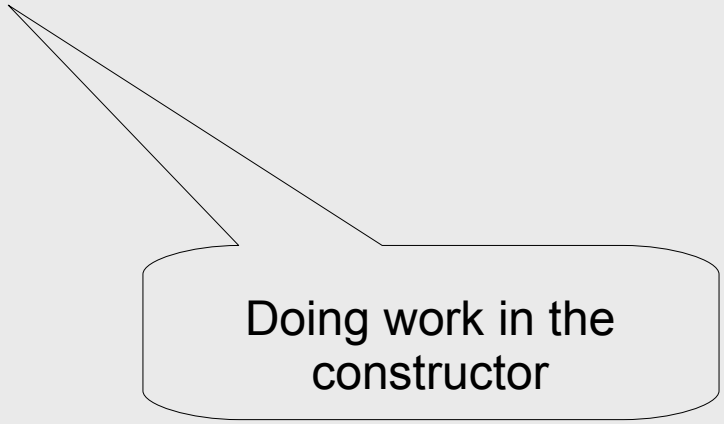
This slide contains material by Miško Hevery

Solving the design problem

Step 1: Do not mix new with logic

```php
<?php
class Document {
    private $html;

    public function __construct(HttpClient $client, $url) {
        $this->html = $client->get($url);
    }
}
?>
```



Interlude

Test Doubles

Replacing a component on which the tested component depends with a "test-specific equivalent".

```php
<?php
class DocumentTest extends PHPUnit_Framework_TestCase
{
    public function testSomething()
    {
        $stub = $this->getMock('HttpClient');
        $stub->expects($this->any())
             ->method('get')
             ->will($this->returnValue('<html>...</html>'));

        $document = new Document($stub, 'url');

        // ...
    }
}
?>
```

Solving the design problem

Step 2: Do not perform work in the constructor

```php
<?php
class Document {
    private $html;

    public function __construct($html) {
        $this->html = $html;
    }
}
```

Solving the design problem

Step 2: Do not perform work in the constructor

```php
<?php
class Document {
    private $html;

    public function __construct($html) {
        $this->html = $html;
    }
}

class DocumentFactory {
    public function __construct(HttpClient $client) {
        $this->client = $client;
    }

    public function build($url) {
        return new Document($this->client->get($url));
    }
}
```

Global State

- Global variables
- Static variables in classes
- Static variables in functions and methods
- Persistent Data
  - Database
  - Filesystem
- Distributed Data
  - memcached

Global State

Global Variables in PHP

- A variable `$foo` declared in the global scope is stored in `$GLOBALS['foo']`.
- `$GLOBALS` is a so-called superglobal.
- Superglobals are built-in variables that are available in all scopes.
- `$GLOBALS['foo']` can be used to access the global variable `$foo` in the scope of a function or method
- `global $foo;` can be used to create a local variable with a reference to the global variable.

Global State

Super-Globals Variables in PHP

```
$GLOBALS = array(
   'GLOBALS'  => &$GLOBALS,
   '_ENV'     => array(),
   '_POST'    => array(),
   '_GET'     => array(),
   '_COOKIE'  => array(),
   '_SERVER'  => array(),
   '_FILES'   => array(),
   '_REQUEST' => array()
);
```

Global State

Global Variables and PHPUnit

```php
<?php
class GlobalsTest extends PHPUnit_Framework_TestCase {
    public function testOne() {
        $GLOBALS['foo'] = 'bar';
        $this->assertEquals('bar', $GLOBALS['foo']);
    }

    public function testTwo() {
        $this->assertFalse(isset($GLOBALS['foo']));
    }
}
?>
```

```
sb@ubuntu ~ % phpunit GlobalsTest
PHPUnit 3.4.0 by Sebastian Bergmann.

..

Time: 0 seconds

OK (2 tests, 2 assertions)
```

Global State

Global Variables and PHPUnit

```php
<?php
/**
 * @backupGlobals disabled
 */
class GlobalsTest extends PHPUnit_Framework_TestCase {
    public function testOne() {
        $GLOBALS['foo'] = 'bar';
        $this->assertEquals('bar', $GLOBALS['foo']);
    }

    public function testTwo() {
        $this->assertTrue(isset($GLOBALS['foo']));
    }
}
?>
```

```
sb@ubuntu ~ % phpunit GlobalsTest
PHPUnit 3.4.0 by Sebastian Bergmann.

..

Time: 0 seconds

OK (2 tests, 2 assertions)
```

Global State

thePHP.cc

Global Variables and PHPUnit

```php
<?php
/**
 * @backupGlobals disabled
 */
class GlobalsTest extends PHPUnit_Framework_TestCase {
    /**
     * @backupGlobals enabled
     */
    public function testOne() {
        $GLOBALS['foo'] = 'bar';
        $this->assertEquals('bar', $GLOBALS['foo']);
    }

    public function testTwo() {
        $this->assertFalse(isset($GLOBALS['foo']));
    }
}
?>
```

```
sb@ubuntu ~ % phpunit GlobalsTest
PHPUnit 3.4.0 by Sebastian Bergmann.

..

Time: 0 seconds

OK (2 tests, 2 assertions)
```

Global State

Global Variables and PHPUnit

```php
<?php
class GlobalsTest extends PHPUnit_Framework_TestCase {
    protected $backupGlobalsBlacklist = array('foo');

    public function testOne() {
        $GLOBALS['foo'] = 'bar';
        $this->assertEquals('bar', $GLOBALS['foo']);
    }

    public function testTwo() {
        $this->assertTrue(isset($GLOBALS['foo']));
    }
}
?>
```

```
sb@ubuntu ~ % phpunit GlobalsTest
PHPUnit 3.4.0 by Sebastian Bergmann.

..

Time: 0 seconds

OK (2 tests, 2 assertions)
```

Global State

Global Variables and PHPUnit

- Default Behaviour
  - Backup `$GLOBALS` before the execution of a test
  - Run the test
  - Restore `$GLOBALS` after the execution of a test
- `$backupGlobals = FALSE`
  - Do not backup and restore `$GLOBALS`
- `$backupGlobalsBlacklist = array()`
  - Backup and restore `$GLOBALS` but not the variables listed in the array

Global State

Static Attributes and PHPUnit

```php
<?php
class Singleton {
    private static $uniqueInstance = NULL;

    protected function __construct() {}
    private final function __clone() {}

    public static function getInstance() {
        if (self::$uniqueInstance === NULL) {
            self::$uniqueInstance = new Singleton;
        }

        return self::$uniqueInstance;
    }
}
?>
```

Global State

Static Attributes and PHPUnit

```php
<?php
class SingletonTest extends PHPUnit_Framework_TestCase {
    public function testFirstCallToGetInstance() {
        return Singleton::getInstance();
    }

    /**
     * @depends testFirstCallToGetInstance
     */
    public function testSecondCallToGetInstance($singleton) {
        $this->assertNotSame(
          $singleton, Singleton::getInstance()
        );
    }
}
?>
```

```
sb@ubuntu ~ % phpunit SingletonTest
PHPUnit 3.4.0 by Sebastian Bergmann.

..

Time: 0 seconds

OK (2 tests, 2 assertions)
```

Global State

Static Attributes and PHPUnit

```php
<?php
/**
 * @backupStaticAttributes disabled
 */
class SingletonTest extends PHPUnit_Framework_TestCase {
    public function testFirstCallToGetInstance() {
        return Singleton::getInstance();
    }

    /**
     * @depends testFirstCallToGetInstance
     */
    public function testSecondCallToGetInstance($singleton) {
        $this->assertSame(
          $singleton, Singleton::getInstance()
        );
    }
}
?>
```

```
sb@ubuntu ~ % phpunit SingletonTest
PHPUnit 3.4.0 by Sebastian Bergmann.

..

Time: 0 seconds

OK (2 tests, 2 assertions)
```

Global State

Static Attributes and PHPUnit

```php
<?php
class SingletonTest extends PHPUnit_Framework_TestCase {
    /**
     * @backupStaticAttributes disabled
     */
    public function testFirstCallToGetInstance() {
        return Singleton::getInstance();
    }

    /**
     * @depends testFirstCallToGetInstance
     */
    public function testSecondCallToGetInstance($singleton) {
        $this->assertSame(
          $singleton, Singleton::getInstance()
        );
    }
}
?>
```

```
sb@ubuntu ~ % phpunit SingletonTest
PHPUnit 3.4.0 by Sebastian Bergmann.

..

Time: 0 seconds

OK (2 tests, 2 assertions)
```

Global State

Static Attributes and PHPUnit

```php
<?php
class SingletonTest extends PHPUnit_Framework_TestCase {
    protected $backupStaticAttributesBlacklist = array(
      'Singleton' => array('uniqueInstance')
    );

    public function testFirstCallToGetInstance() {
        return Singleton::getInstance();
    }

    /**
     * @depends testFirstCallToGetInstance
     */
    public function testSecondCallToGetInstance($singleton) {
        $this->assertSame(
          $singleton, Singleton::getInstance()
        );
    }
}
?>
```

```
sb@ubuntu ~ % phpunit SingletonTest
PHPUnit 3.4.0 by Sebastian Bergmann.

..

Time: 0 seconds

OK (2 tests, 2 assertions)
```

Global State

Static Attributes and PHPUnit

- Default Behaviour
  - Backup static attributes of user-defined classes before the execution of a test
  - Run the test
  - Restore static attributes of user-defined classes after the execution of a test

- `$backupStaticAttributes = FALSE`
  - Do not backup and restore static attributes of user-defined classes

- `$backupStaticAttributesBlacklist = array()`
  - Backup and restore static attributes of user-defined classes but not the variables listed in the array

Global State

Filesystem

```php
<?php
class Example {
    protected $id;
    protected $directory;

    public function __construct($id) {
        $this->id = $id;
    }

    public function setDirectory($directory) {
        $this->directory = $directory . DIRECTORY_SEPARATOR . $this->id;

        if (!file_exists($this->directory)) {
            mkdir($this->directory, 0700, TRUE);
        }
    }
}
?>
```

Global State

Filesystem

```php
<?php
require_once 'Example.php';

class ExampleTest extends PHPUnit_Framework_TestCase {
    protected function setUp() {
        if (file_exists(__DIR__ . '/id')) {
            rmdir(__DIR__ . '/id');
        }
    }

    public function testDirectoryIsCreated() {
        $example = new Example('id');
        $this->assertFalse(file_exists(__DIR__ . '/id'));

        $example->setDirectory(__DIR__);
        $this->assertTrue(file_exists(__DIR__ . '/id'));
    }

    protected function tearDown() {
        if (file_exists(__DIR__ . '/id')) {
            rmdir(__DIR__ . '/id');
        }
    }
}
?>
```

Global State

Mocking the Filesystem with vfsStream

```php
<?php
require_once 'vfsStream/vfsStream.php';
require_once 'Example.php';

class ExampleTest extends PHPUnit_Framework_TestCase {
    public function setUp() {
        vfsStreamWrapper::register();
        vfsStreamWrapper::setRoot(new vfsStreamDirectory('exampleDir'));
    }

    public function testDirectoryIsCreated() {
        $example = new Example('id');
        $this->assertFalse(vfsStreamWrapper::getRoot()->hasChild('id'));

        $example->setDirectory(vfsStream::url('exampleDir'));
        $this->assertTrue(vfsStreamWrapper::getRoot()->hasChild('id'));
    }
}
?>
```

Global State

Hidden Global State in PHP

- Loaded source files
- Loaded classes and functions
- Sent HTTP headers

- `new DateTime()`
- `time()`
- `mktime()`
- `rand()`

The End

Thank you for your interest!

These slides will be posted on

http://slideshare.net/sebastian_bergmann

License