Testable Code

Zend Webinar

Tobias Schlitt
@tobySen, toby@qafoo.com

2012-03-15

About me

- Degree in computer sience

About me

- Degree in computer sience
- Professional PHP since 2000

About me

- Degree in computer sience
- Professional PHP since 2000
- Open source enthusiasts
- Various FLOSS projects

Co-founder of

contact@qafoo.com
http://talks.qafoo.com/

Co-founder of

Helping people to create high quality web applications.

`http://qafoo.com`

Co-founder of

Helping people to create high quality web applications.

Expert Consulting

Individual Training

http://qafoo.com

Co-founder of

Helping people to create high quality web applications.

Expert Consulting

REST

Audits

Software Architecture

Discussion

Individual Training

http://qafoo.com

Co-founder of

Helping people to create high quality web applications.

Expert Consulting

REST Audits

Software Architecture

Discussion

Individual Training

Testing Automation

Quality Assurance

CI

`http://qafoo.com`

Co-founder of

Helping people to create high quality web applications.

Expert Consulting

REST Audits

Software Architecture

Discussion

Individual Training

Testing Automation SOLID Review

Quality Assurance Software Design

CI Specification

http://qafoo.com

Co-founder of

Helping people to create high quality web applications.

Expert Consulting **Individual Training**

REST Audits Testing Automation SOLID Review

Software Architecture Quality Assurance Software Design

Discussion CI Specification

Now also JavaScript!

http://qafoo.com

Outline

Ways of testing

- Automatic vs. manual
- Developer vs. tester
- Internal vs. external
- Back end vs. front end
- Code vs. appearance
- Functional vs. non-functional
- Dynamic vs. static

Ways of testing

- Automatic vs. manual
- Developer vs. tester
- Internal vs. external
- Back end vs. front end
- Code vs. appearance
- Functional vs. non-functional
- Dynamic vs. static

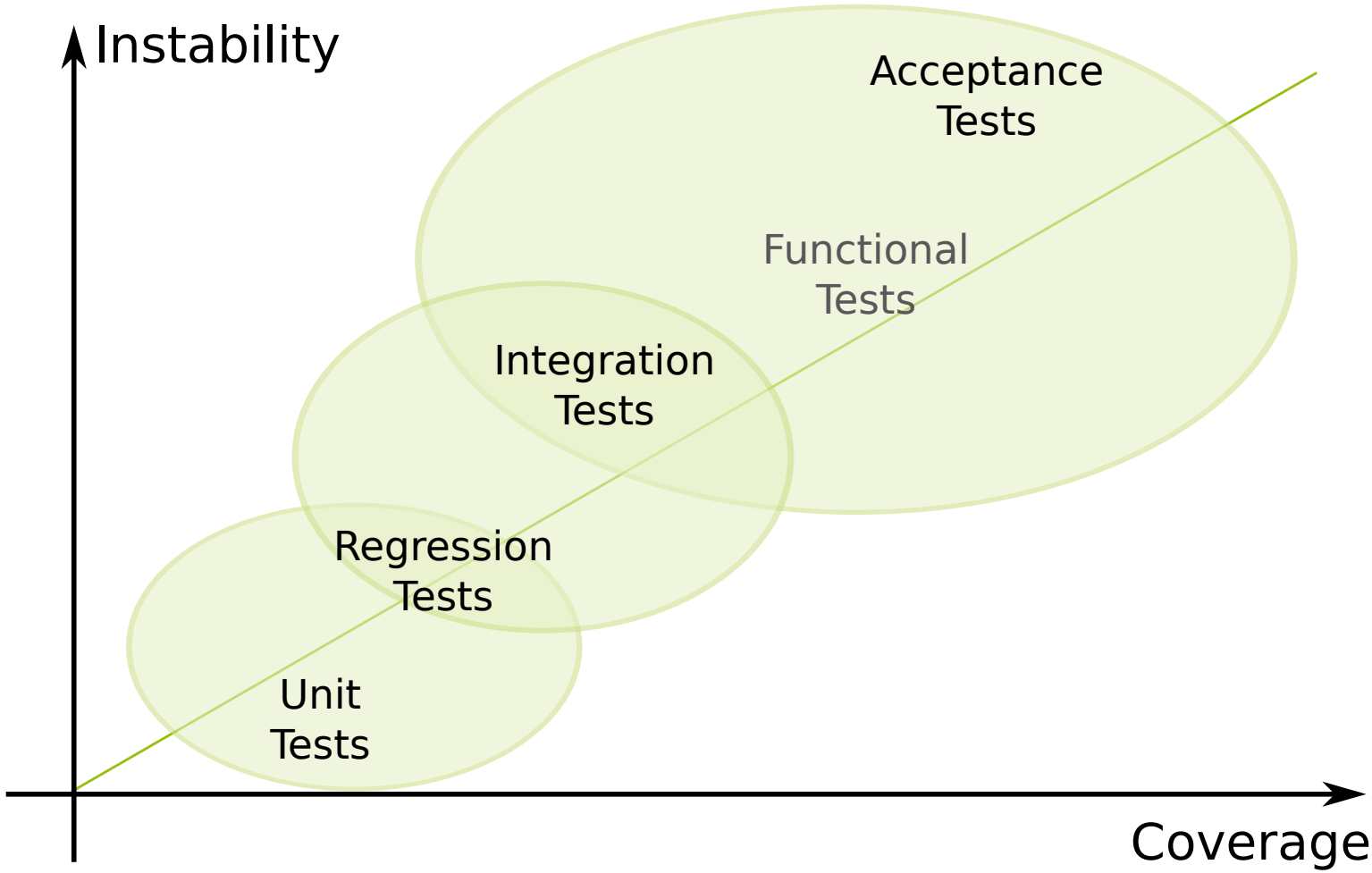
Common test methods

- Unit tests
- Integration tests
- System tests
- Regression tests
- Acceptance tests

Requirements for Tests

- Easy to read
- Reliable
- Extensive
- Stable

Instability vs. Coverage



Outline

The Example

```php
<?php

class WeatherLoader
{
    public function getWeatherForLocation( Location $location )
    {
        $xml = $this->fetchData( $location->city );
        Logger::logDebug( 'Fetched XML', $xml );
        return $this->parseData( $xml );
    }
    protected function fetchData( $city )
    {
        $url = sprintf( 'http://...?city=%s', $city );
        return $this->fetchFromUrl( $url );
    }
    protected function parseData( $xml )
    {
        $weather = new Weather();
        $weather->conditions = $this->parseConditions( $xml );
        $weather->windSpeed = $this->milesToKilometers(
            $this->parseWindSpeed( $xml )
        );
        return $weather;
    }
    /* ... */
}
```

The Example

```php
<?php

class WeatherLoader
{
    public function getWeatherForLocation( Location $location )
    {
        $xml = $this->fetchData( $location->city );
        Logger::logDebug( 'Fetched XML', $xml );
        return $this->parseData( $xml );
    }
    protected function fetchData( $city )
    {
        $url = sprintf( 'http://...?city=%s', $city );
        return $this->fetchFromUrl( $url );
    }
    protected function parseData( $xml )
    {
        $weather = new Weather();
        $weather->conditions = $this->parseConditions( $xml );
        $weather->windSpeed = $this->milesToKilometers(
            $this->parseWindSpeed( $xml )
        );
        return $weather;
    }
    /* ... */
}
```

Outline

Testing issues
- Issue #1
- Issue #2
- Issue #3
- Issue #4

Issue #1

```php
<?php

class WeatherLoader
{
    public function getWeatherForLocation( Location $location )
    {
        $xml = $this->fetchData( $location->city );
        Logger::logDebug( 'Fetched XML', $xml );
        return $this->parseData( $xml );
    }
    protected function fetchData( $city )
    {
        $url = sprintf( 'http://...?city=%s', $city );
        return $this->fetchFromUrl( $url );
    }
    protected function parseData( $xml )
    {
        $weather = new Weather();
        $weather->conditions = $this->parseConditions( $xml );
        $weather->windSpeed = $this->milesToKilometers(
            $this->parseWindSpeed( $xml )
        );
        return $weather;
    }
    /* ... */
}
```

Issue #1

```php
<?php

class WeatherLoader
{
    public function getWeatherForLocation( Location $location )
    {
        $xml = $this->fetchData( $location->city );
        Logger::logDebug( 'Fetched XML', $xml );
        return $this->parseData( $xml );
    }
    protected function fetchData( $city )
    {
        $url = sprintf( 'http://...?city=%s', $city );
        return $this->fetchFromUrl( $url );
    }
    protected function parseData( $xml )
    {
        $weather = new Weather();
        $weather->conditions = $this->parseConditions( $xml );
        $weather->windSpeed = $this->milesToKilometers(
            $this->parseWindSpeed( $xml )
        );
        return $weather;
    }
    /* ... */
}
```

Protected to Public

```php
<?php

class WeatherLoader
{
    public function getWeatherForLocation( Location $location )
    {
        $xml = $this->fetchData( $location->city );
        Logger::logDebug( 'Fetched XML', $xml );
        return $this->parseData( $xml );
    }
    public function fetchData( $city )
    {
        $url = sprintf( 'http://...?city=%s', $city );
        return $this->fetchFromUrl( $url );
    }
    public function parseData( $xml )
    {
        $weather = new Weather();
        $weather->conditions = $this->parseConditions( $xml );
        $weather->windSpeed = $this->milesToKilometers(
            $this->parseWindSpeed( $xml )
        );
        return $weather;
    }
    /* ... */
}
```

Protected to Public

```
1  <?php
2
3  class Weather
4  {
5      public funct        WeatherForL        ( Location $location )
6      {
7          $xml = $thi        hData(        tion->city );
8          Logger::logD        Fet        L', $xml );
9          return $this->        a        ml );
10     }
11     public function fetch        city )
12     {
13         $url = sprintf(        ?city=%s', $city );
14         return $this->        $url );
15     }
16     public function        Data( $
17     {
18         $weather        Weather();
19         $weath        ditions = $this        Conditions( $xml );
20         $weat        ndSpeed = $this->        Kilometers(
21                 parseWindSpeed( $xm
22         );
23         r        weather;
24     }
25     /* ... */
26 }
```

Mocking the Subject

```php
<?php

class TestingWeatherLoader extends WeatherLoader
{
    public function fetchData( $city )
    {
        return parent::fetchData( $city );
    }
    public function parseData( $xml )
    {
        return parent::parseData( $xml );
    }
    // ...
}
```

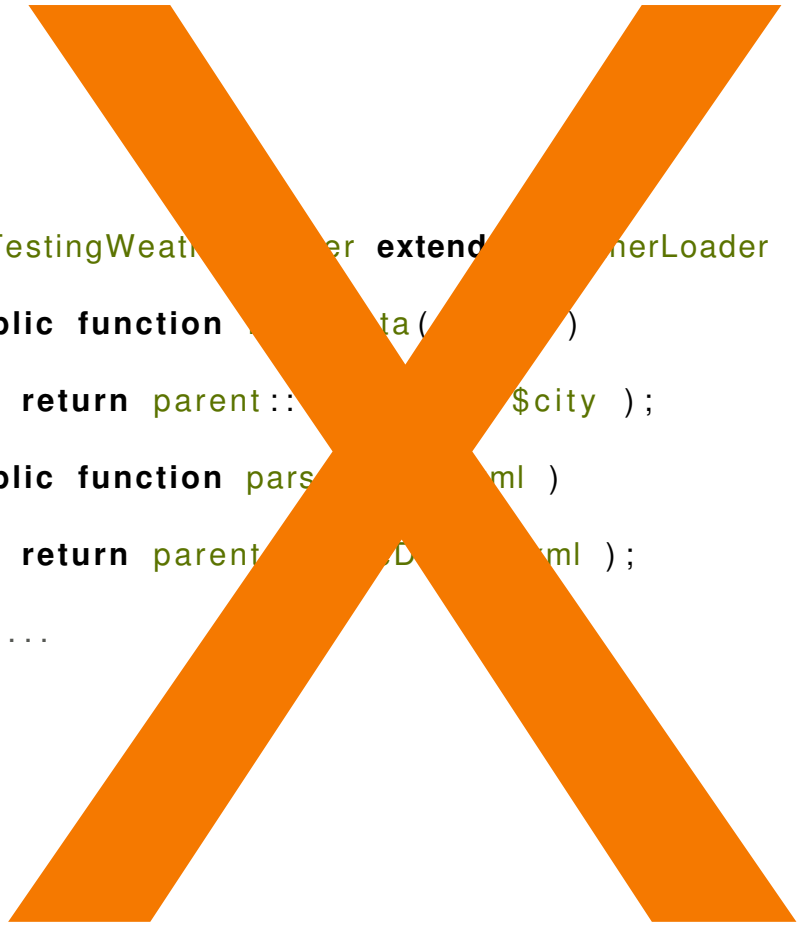
Mocking the Subject

```php
<?php

class TestingWeatherLoader extends WeatherLoader
{
    public function fetchData( $city )
    {
        return parent::fetchData( $city );
    }
    public function parseData( $xml )
    {
        return parent::parseData( $xml );
    }
    // ...
}
```

Mocking the Subject

```
1   <?php
2
3   class TestingWeat      er extend      herLoader
4   {
5       public function      ta(      )
6       {
7           return parent::          $city );
8       }
9       public function pars          ml )
10      {
11          return parent      D      ml );
12      }
13      // ...
14  }
```



Protected to Public

- Exposed functionality will be used
- Creates public API that is hard to change
- Internal dependencies might break

The Real Issue

E_TOO_MANY_RESPONSIBILITIES

The Fix

```php
<?php

class WeatherLoader
{
    public function __construct( WeatherService $service, WeatherParser $parser )
    {
        // ...
    }
    public function getWeatherForLocation( Location $location )
    {
        $data = $this->service->getWeather( $location );
        Logger::logDebug( 'Fetched data', $data );
        return $this->parser->parseData( $data );
    }
}
```

The Fix

```php
<?php

class WeatherLoader
{
    public function __construct( WeatherService $service, WeatherParser $parser )
    {
        // ...
    }
    public function getWeatherForLocation( Location $location )
    {
        $data = $this->service->getWeather( $location );
        Logger::logDebug( 'Fetched data', $data );
        return $this->parser->parseData( $data );
    }
}
```

The Fix

- Never test private/protected explicitely
- Test them implicitely . . .
- . . . or change the code

Outline

Testing issues

- Issue #1
- Issue #2
- Issue #3
- Issue #4

Issue #2

```php
<?php

class WeatherLoader
{
    public function __construct( WeatherService $service, WeatherParser $parser )
    {
        // ...
    }
    public function getWeatherForLocation( Location $location )
    {
        $data = $this->service->getWeather( $location );
        Logger::logDebug( 'Fetched data', $data );
        return $this->parser->parseData( $data );
    }
}
```

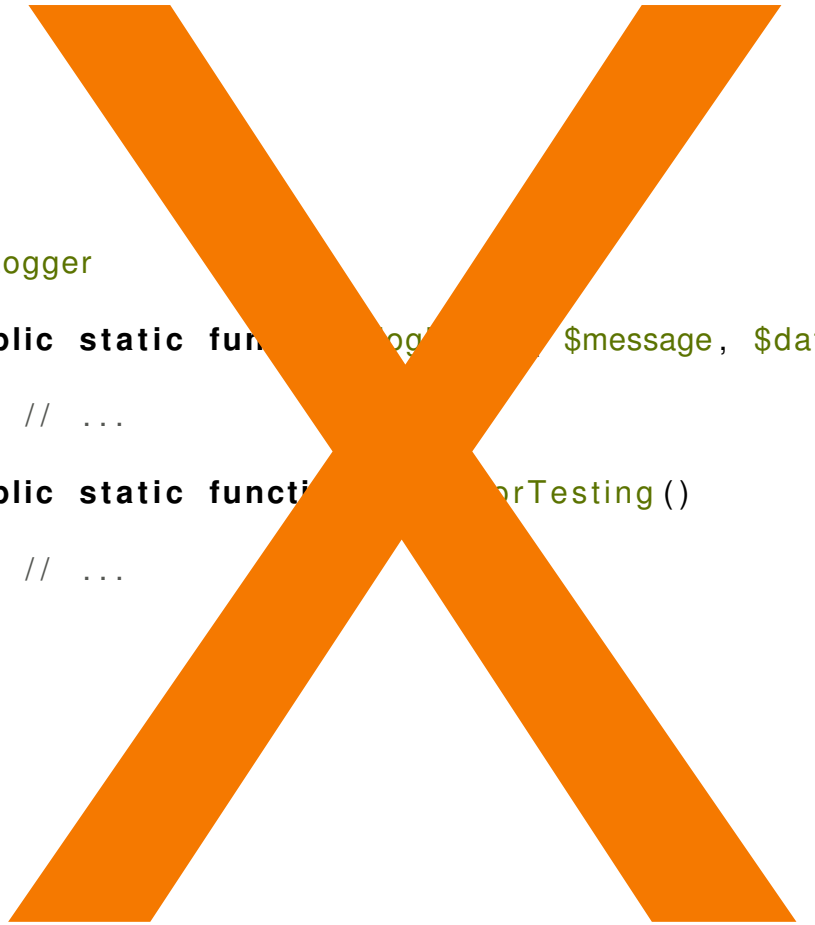
Test Code in Production

```
1   <?php
2
3   class Logger
4   {
5       public static function logDebug( $message, $data )
6       {
7           // ...
8       }
9       public static function resetForTesting()
10      {
11          // ...
12      }
13  }
```

Test Code in Production

```php
<?php

class Logger
{
    public static fun      og      $message, $data )
    {
        // ...
    }
    public static functi      orTesting()
    {
        // ...
    }
}
```



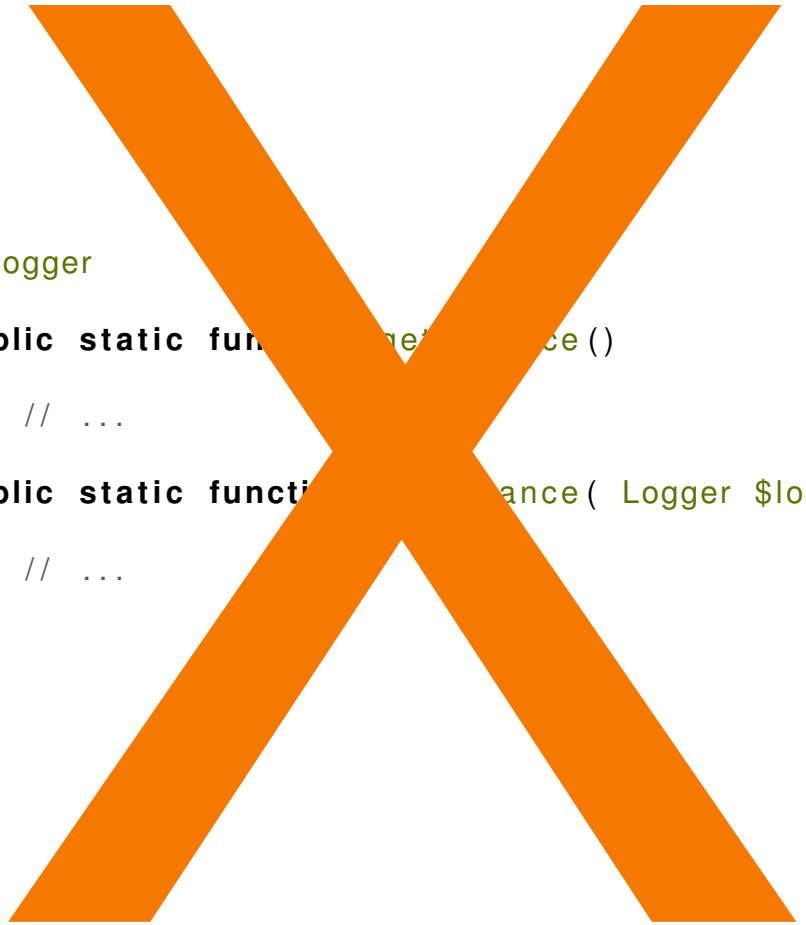
Test Code in Production - continued

```
1   <?php
2
3   class Logger
4   {
5       public static function getInstance()
6       {
7           // ...
8       }
9       public static function setInstance( Logger $logger )
10      {
11          // ...
12      }
13  }
```

Test Code in Production - continued

```php
<?php

class Logger
{
    public static fun      ge      ce()
    {
        // ...
    }
    public static functi      ance( Logger $logger )
    {
        // ...
    }
}
```



The Real Issue

E_STATIC_DEPENDENCY

The Fix

```php
<?php

class WeatherLoader
{
    public function __construct(
        WeatherService $service,
        WeatherParser $parser
        Logger $logger )
    {
        // ...
    }
    public function getWeatherForLocation( Location $location )
    {
        $data = $this->service->getWeather( $location );
        $this->logger->logDebug( 'Fetched data', $data );
        return $this->parser->parseData( $data );
    }
}
```

The Fix

- Never use `static` access
- Always inject dependencies
- Maybe use a dependency injection container (DIC)

Outline

Testing issues

- Issue #1
- Issue #2
- Issue #3
- Issue #4

Issue #3

```php
<?php

class WeatherService
{
    public function __construct( AppRegistry $registry )
    {
        // ...
    }
    public function getWeather( Location $location )
    {
        $httpClient = $this->appRegistry->get( 'http_client' );
        $url = sprintf( 'http://...?city=%s', $city );
        return $httpClient->get( $url );
    }
}
```

Issue #3

```php
<?php

class WeatherService
{
    public function __construct( AppRegistry $registry )
    {
        // ...
    }
    public function getWeather( Location $location )
    {
        $httpClient = $this->appRegistry->get( 'http_client' );
        $url = sprintf( 'http://...?city=%s', $city );
        return $httpClient->get( $url );
    }
}
```

Mocking to Mock

```php
<?php

class WeatherServiceTest extends PHPUnit_Framework_TestCase
{
    public function testGetWeather()
    {
        $httpClientMock = $this->getMock( 'HttpClient' );
        $httpClientMock->expects( $this->once() )
            ->method( 'get' )
            /* ... */;

        $appRegistryMock = $this->getMock( 'AppRegistry' );
        $appRegistryMock->expects( $this->once() )
            ->method( 'get' )
            /* ... */;

        $service = new WeatherService( $appRegistryMock );
        $this->assertEquals(
            '...',
            $service->getWeather( new Location() )
        );
    }
}
```

Mocking to Mock

```php
<?php

class WeatherServiceTest extends PHPUnit_Framework_TestCase
{
    public function testGetWeather()
    {
        $httpClientMock = $this->getMock( 'HttpClient' );
        $httpClientMock->expects( $this->once() )
            ->method( 'get' )
            /* ... */;

        $appRegistryMock = $this->getMock( 'AppRegistry' );
        $appRegistryMock->expects( $this->once() )
            ->method( 'get' )
            /* ... */;

        $service = new WeatherService( $appRegistryMock );
        $this->assertEquals(
            '...',
            $service->getWeather( new Location() )
        );
    }
}
```

Mocking to Mock

```php
<?php

class WeatherServiceTest extends PHPUnit_Framework_TestCase
{
    public function testGetWeather()
    {
        $httpClientMock = $this->getMock( 'HttpClient' );
        $httpClientMock->expects( $this->once() )
            ->method( 'get' )
            /* ... */;

        $appRegistryMock = $this->getMock( 'AppRegistry' );
        $appRegistryMock->expects( $this->once() )
            ->method( 'get' )
            /* ... */;

        $service = new WeatherService( $appRegistryMock );
        $this->assertEquals(
            '...',
            $service->getWeather( new Location() )
        );
    }
}
```

Mocking to Mock

```
1  <?php
2
3  class Weathe      Test extends PHP      mework_TestCase
4  {
5      public funct      tGetWeather
6      {
7          $httpClient       $this-      ck( 'HttpClient' );
8          $httpClientM      pec      s->once() )
9              ->method(      )
10             /* ... */;
11
12         $appRegistryMock      >getMock( 'AppRegistry' );
13         $appRegistryMoc          $this->once() )
14             ->method(      )
15             /* ... *
16
17         $service      eatherSe      appRegistryMock );
18         $this->a      quals(
19             '
20             $      ->getWeather( new      on() )
21         );
22     }
23 }
```

Using Productive Code in Tests

```php
<?php

class WeatherServiceTest extends PHPUnit_Framework_TestCase
{
    public function testGetWeather()
    {
        $httpClientMock = $this->getMock( 'HttpClient' );
        $httpClientMock->expects( $this->once() )
            ->method( 'get' )
            /* ... */;

        $appRegistry = new AppRegistry();
        $appRegistry->set( 'http_client', $httpClientMock );

        $service = new WeatherService( $appRegistry );
        $this->assertEquals(
            '...',
            $service->getWeather( new Location() )
        );
    }
}
```

Using Productive Code in Tests

```
1   <?php
2
3   class Weather      Test extends PHP      mework_TestCase
4   {
5       public funct      GetWeather
6       {
7           $httpClientM      $this      ck( 'HttpClient' );
8           $httpClientMo      pec      his->once() )
9               ->method(
10              /* ... */;
11
12          $appRegistry = r      istry();
13          $appRegistry->      lient', $httpClientMock );
14
15          $service = r      ather      ( $appRegistry );
16          $this->ass      als(
17              '...'
18              $se      getWeather( r      ation() )
19          );
20      }
21  }
```

The Real Issue

E_REACHING_THROUGH_OBJECTS

The Fix

```php
<?php

class WeatherService
{
    public function __construct( HttpClient $httpClient )
    {
        // ...
    }
    public function getWeather( Location $location )
    {
        $url = sprintf( 'http://...?city=%s', $city );
        return $this->httpClient->get( $url );
    }
}
```

The Fix

- Do not pull dependencies . . .
- . . . push them
- Do not reach through objects

Outline

Testing issues

- Issue #1
- Issue #2
- Issue #3
- Issue #4

Issue #4

```php
<?php

class Logger
{
    public function __construct( $fileName )
    {
        // ... error checks ...
        $this->fileHandle = fopen( $fileName, 'a' );
    }
    public function logDebug( $message, $data )
    {
        fwrite(
            $this->fileHandle,
            sprintf(
                "%s (%s)\n",
                $message,
                $data
            )
        );
    }
}
```

Issue #4

```php
<?php

class Logger
{
    public function __construct( $fileName )
    {
        // ... error checks ...
        $this->fileHandle = fopen( $fileName, 'a' );
    }
    public function logDebug( $message, $data )
    {
        fwrite(
            $this->fileHandle,
            sprintf(
                "%s (%s)\n",
                $message,
                $data
            )
        );
    }
}
```

Issue #4

```php
<?php

class Logger
{
    public function __construct( $fileName )
    {
        // ... error checks ...
        $this->fileHandle = fopen( $fileName, 'a' );
    }
    public function logDebug( $message, $data )
    {
        fwrite(
            $this->fileHandle,
            sprintf(
                "%s (%s)\n",
                $message,
                $data
            )
        );
    }
}
```

Accessing File System in Tests

```php
<?php

class LoggerTest extends PHPUnit_Framework_TestCase
{
    public function testLogDebugSuccess()
    {
        $tmpLogFile = $this->getTempFileName();

        $logger = new Logger( $tmpLogFile );
        $logger->logDebug( 'Some message.', 'with data' );

        $this->assertEquals(
            "Some message. (with data)\n",
            file_get_contents( $tmpLogFile )
        );
        unlink( $tmpLogFile );
    }
}
```

Accessing File System in Tests

```php
<?php

class LoggerTest extends PHPUnit_Framework_TestCase
{
    public function testLogDebugSuccess()
    {
        $tmpLogFile = $this->getTempFileName();

        $logger = new Logger( $tmpLogFile );
        $logger->logDebug( 'Some message.', 'with data' );

        $this->assertEquals(
            "Some message. (with data)\n",
            file_get_contents( $tmpLogFile )
        );
        unlink( $tmpLogFile );
    }
}
```

Accessing File System in Tests

```
1   <?php
2
3   class LoggerTest extends PHPUnit_Framework_TestCase
4   {
5       public function testLogDebugSuccess()
6       {
7           $tmpLogFile = $this->getTempFileName();
8
9           $logger = new Logger( $tmpLogFile );
10          $logger->logDebug( 'Some message.', 'with data' );
11
12          $this->assertEquals(
13              "Some message. (with data)\n",
14              file_get_contents( $tmpLogFile )
15          );
16          unlink( $tmpLogFile );
17      }
18  }
```

Accessing File System in Tests

```
1  <?php
2
3  class LoggerTest extends PHPUnit_Framework_TestCase
4  {
5      public function testLogDebugSuccess()
6      {
7          $tmpLogFile = $this->getTempFileName();
8
9          $logger = new Logger( $tmpLogFile );
10         $logger->logDebug( 'Some message.', 'with data' );
11
12         $this->assertEquals(
13             "Some message. (with data)\n",
14             file_get_contents( $tmpLogFile )
15         );
16         unlink( $tmpLogFile );
17     }
18 }
```

Accessing File System in Tests

```php
<?php

class LoggerTest extends PHPUnit_Framework_TestCase
{
    public function testLogDebugSuccess()
    {
        $tmpLogFile = $this->getTempFileName();

        $logger = new Logger( $tmpLogFile );
        $logger->logDebug( 'Some message.', 'with data' );

        $this->assertEquals(
            "Some message. (with data)\n",
            file_get_contents( $tmpLogFile )
        );
        unlink( $tmpLogFile );
    }
}
```

Accessing File System in Tests

```
1   <?php
2
3   class LoggerTes      ds PHPUnit_Fr      _TestCase
4   {
5       public functio     gDebug      ()
6       {
7           $tmpLogFile =      >      FileName();
8
9           $logger = new Lo          pLogFile );
10          $logger->logDebu          message.', 'with data' );
11
12          $this->assertE
13              "Some me          (w          a)\n",
14              file_get          nts(          gFile )
15          );
16          unlink(          gFile );
17      }
18  }
```

Accessing File System in Tests

- No file access in unit tests (slow!)
- Maintaining temporary files sucks
  - Creating
  - Cleanup
  - System differences

The Virtual File System

```php
<?php

class LoggerTest extends PHPUnit_Framework_TestCase
{
    public function testLogDebugSuccess()
    {
        vfsStream::setup( 'test' );
        $logFile = vfsStream::url( 'test' ) . '/message.log';

        $logger = new Logger( $logFile );
        $logger->logDebug( 'Some message.', 'with data' );

        $this->assertTrue(
            vfsStreamWrapper::getRoot()->hasChild( 'message.log' )
        );
        $this->assertEquals(
            "Some message. (with data)\n",
            file_get_contents( $logFile )
        );
    }
}
```

The Virtual File System

```php
<?php

class LoggerTest extends PHPUnit_Framework_TestCase
{
    public function testLogDebugSuccess()
    {
        vfsStream::setup( 'test' );
        $logFile = vfsStream::url( 'test' ) . '/message.log';

        $logger = new Logger( $logFile );
        $logger->logDebug( 'Some message.', 'with data' );

        $this->assertTrue(
            vfsStreamWrapper::getRoot()->hasChild( 'message.log' )
        );
        $this->assertEquals(
            "Some message. (with data)\n",
            file_get_contents( $logFile )
        );
    }
}
```

The Virtual File System

```php
<?php

class LoggerTest extends PHPUnit_Framework_TestCase
{
    public function testLogDebugSuccess()
    {
        vfsStream::setup( 'test' );
        $logFile = vfsStream::url( 'test' ) . '/message.log';

        $logger = new Logger( $logFile );
        $logger->logDebug( 'Some message.', 'with data' );

        $this->assertTrue(
            vfsStreamWrapper::getRoot()->hasChild( 'message.log' )
        );
        $this->assertEquals(
            "Some message. (with data)\n",
            file_get_contents( $logFile )
        );
    }
}
```

The Virtual File System

```
1  <?php
2
3  class LoggerT      nds PHPUnit_Fram      estCase
4  {
5      public functi      LogDebugSu      )
6      {
7          vfsStream::      'test
8          $logFile = vf      ::      test' ) . '/message.log';
9
10         $logger = new L      gFile );
11         $logger->logDebug      message.', 'with data' );
12
13         $this->assertT
14             vfsStream      r::      t()->hasChild( 'message.log' )
15         );
16         $this->ass      als(
17             "Som      ge. (with      ",
18             file      ntents( $log
19         );
20     }
21 }
```

The Virtual File System

- Works, but . . .

The Real Issue

E_HARD_SYSTEM_DEPENDENCY

The Fix

```
1  <?php
2
3  class Logger
4  {
5      public function __construct( FileHandler $fileHandler )
6      {
7          $this->fileHandler = $fileHandler;
8      }
9      public function logDebug( $message, $data )
10     {
11         $this->fileHandler->write(
12             sprintf(
13                 "%s (%s)\n",
14                 $message,
15                 $data
16             )
17         );
18     }
19 }
```

The Fix

```php
<?php

class Logger
{
    public function __construct( FileHandler $fileHandler )
    {
        $this->fileHandler = $fileHandler;
    }
    public function logDebug( $message, $data )
    {
        $this->fileHandler->write(
            sprintf(
                "%s (%s)\n",
                $message,
                $data
            )
        );
    }
}
```

The Fix

- Abstract system dependencies . . .
- . . . as low as possible

Outline

What have we seen?

- Single Responsibility Principle

What have we seen?

- Single Responsibility Principle
- Open Close Principle

What have we seen?

- Single Responsibility Principle
- Open Close Principle
- Law of Demeter

What have we seen?

- Single Responsibility Principle
- Open Close Principle
- Law of Demeter
- Dependency Inversion Principle

Conclusion

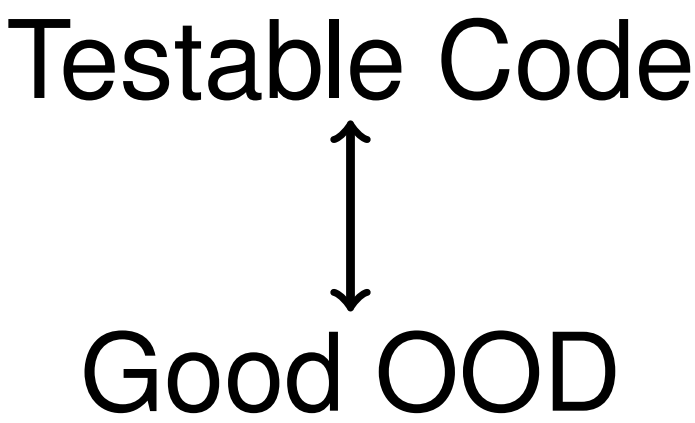
Testable Code

Conclusion

Testable Code

Good OOD

Conclusion



Q/A

Questions? Comments? Feedback?

Thanks for Listening

Stay in touch

- Tobias Schlitt
- toby@qafoo.com
- @tobySen / @qafoo

Slides: `http://talks.qafoo.com`

Rent a PHP quality expert:
`http://qafoo.com`