Testable Code

PHP Usergroup Düsseldorf

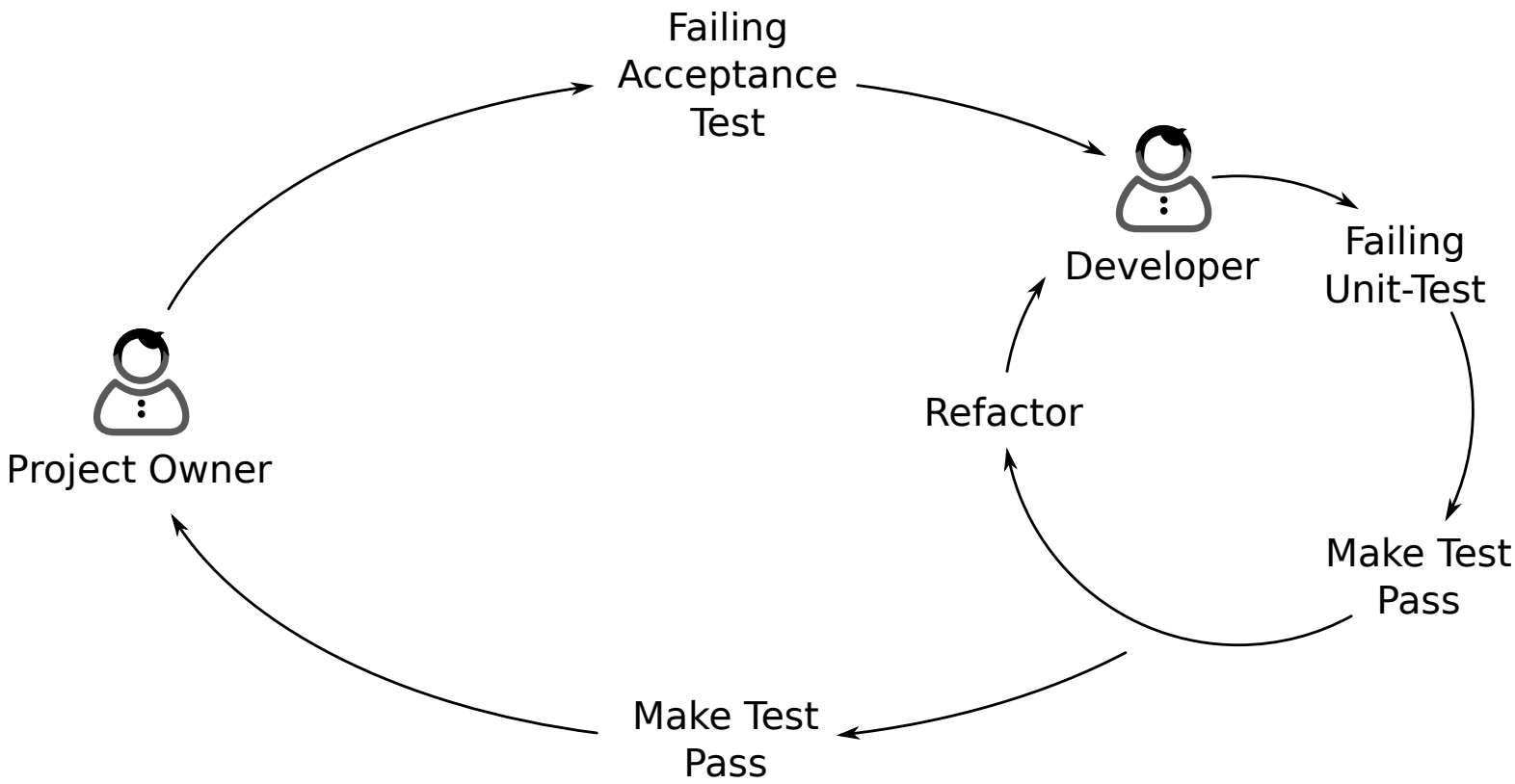
Tobias Schlitt (@tobySen)

2013-10-10

About me

- Tobias Schlitt
  - Twitter: tobySen
  - Email: toby@qafoo.com
- Trainer / Consultant at Qafoo (`http://qafoo.com`)
  - Object Oriented Design
  - Automated Testing

Multi Feedback Test Cycle



Unit Testing?

Do you do unit testing?

Outline

Testing issues

Conclusion

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;
    }
    /* ... */
}
```

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

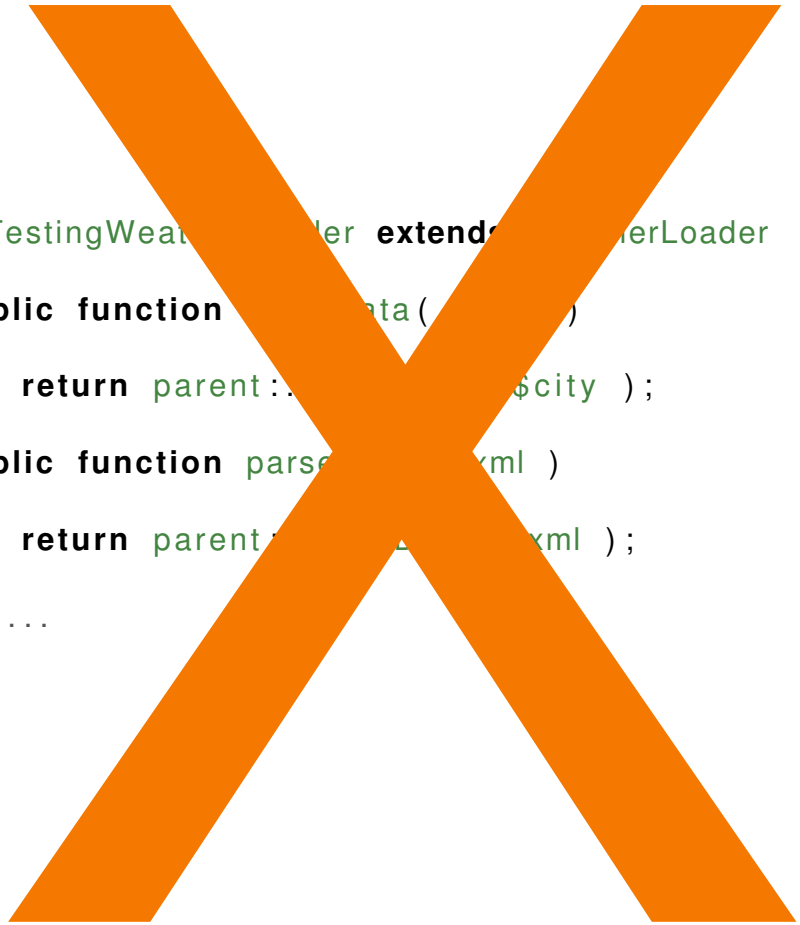
```
1  <?php
2
3  class Weathe
4  {
5      public func      tWeatherForLo      Location $location )
6      {
7          $xml = $th      chData(      on->city );
8          Logger::logL      'Fetch      ', $xml );
9          return $this-      Dat      );
10     }
11     public function fetc      ity )
12     {
13         $url = sprintf(      .?city=%s', $city );
14         return $this->      ( $url );
15     }
16     public function      ata(
17     {
18         $weather      Weather();
19         $weathe      itions = $thi      eConditions( $xml );
20         $weath      dSpeed = $this-      oKilometers(
21             $      parseWindSpeed( $x
22         );
23         re      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

```
1   <?php
2
3   class TestingWeat      der extend      erLoader
4   {
5       public function      ata(      )
6       {
7           return parent::      $city );
8       }
9       public function parse      xml )
10      {
11          return parent:      xml );
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

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

Issue #2

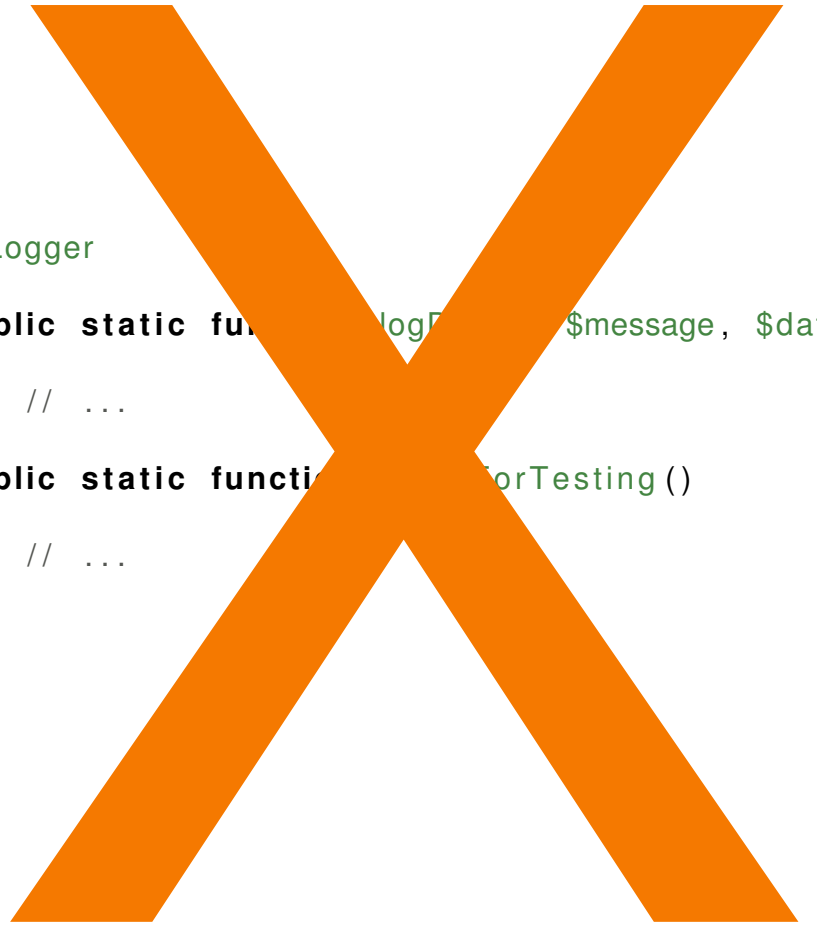
```php
<?php

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

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

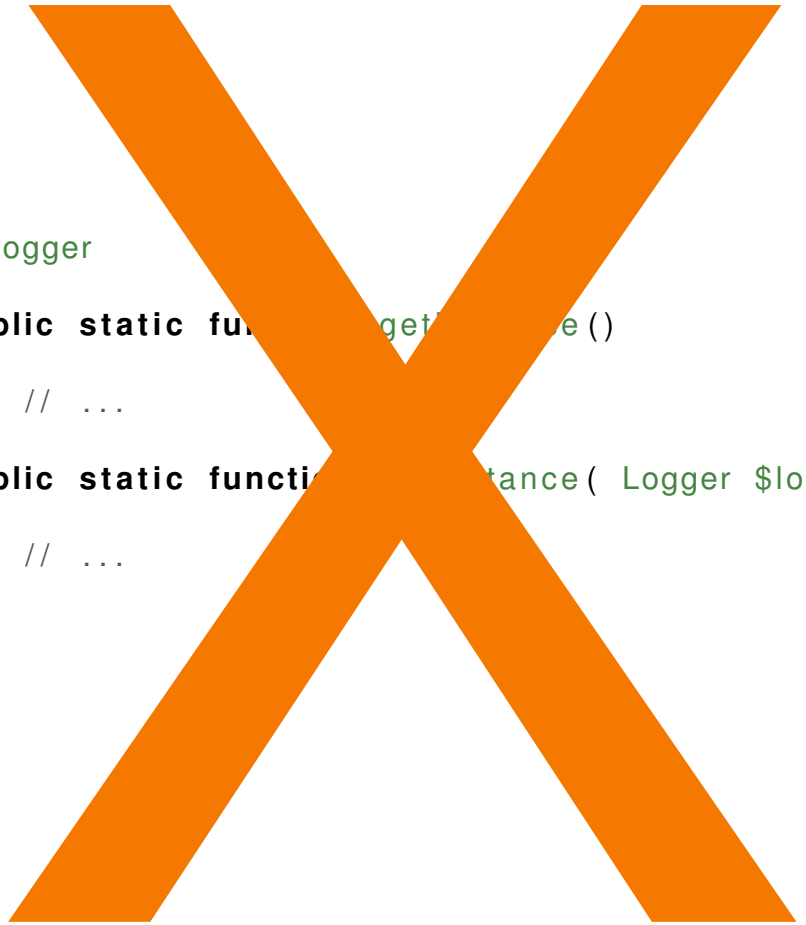


Test Code in Production - continued

```php
<?php

class Logger
{
    public static function getInstance()
    {
        // ...
    }
    public static function setInstance( Logger $logger )
    {
        // ...
    }
}
```

Test Code in Production - continued

```php
<?php

class Logger
{
    public static fu    get    e()
    {
        // ...
    }
    public static functi    tance( 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)

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', $location->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

```
1  <?php
2
3  class Weathe      eTest extends PHPU      ework_TestCase
4  {
5      public func      stGetWeather()
6      {
7          $httpClien      = $this->      k( 'HttpClient' );
8          $httpClientM      xpect      s->once() )
9              ->method(      )
10             /* ... */;
11
12         $appRegistryMock      ->getMock( 'AppRegistry' );
13         $appRegistryMock      ( $this->once() )
14             ->method(
15             /* ... */
16
17         $service =      eatherSu      $appRegistryMock );
18         $this->a      uals(
19             '..
20             $s      >getWeather( ne      ion() )
21         );
22     }
23 }
```

Using Productive Code in Tests

```
1  <?php
2
3  class WeatherServiceTest extends PHPUnit_Framework_TestCase
4  {
5      public function testGetWeather()
6      {
7          $httpClientMock = $this->getMock( 'HttpClient' );
8          $httpClientMock->expects( $this->once() )
9              ->method( 'get' )
10             /* ... */;
11
12         $appRegistry = new AppRegistry();
13         $appRegistry->set( 'http_client', $httpClientMock );
14
15         $service = new WeatherService( $appRegistry );
16         $this->assertEquals(
17             '...',
18             $service->getWeather( new Location() )
19         );
20     }
21 }
```

Using Productive Code in Tests

```
1  <?php
2
3  class Weathe      Test extends PHP       mework_TestCase
4  {
5      public funct      tGetWeather
6      {
7          $httpClient       $this-      ck( 'HttpClient' );
8          $httpClientM      xpec      is->once() )
9              ->method(      )
10             /* ... */;
11
12         $appRegistry = n      gistry();
13         $appRegistry->s      lient', $httpClientMock );
14
15         $service = n      ther      ( $appRegistry );
16         $this->ass      is(
17             '...'
18             $se      getWeather(      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

Issue #4

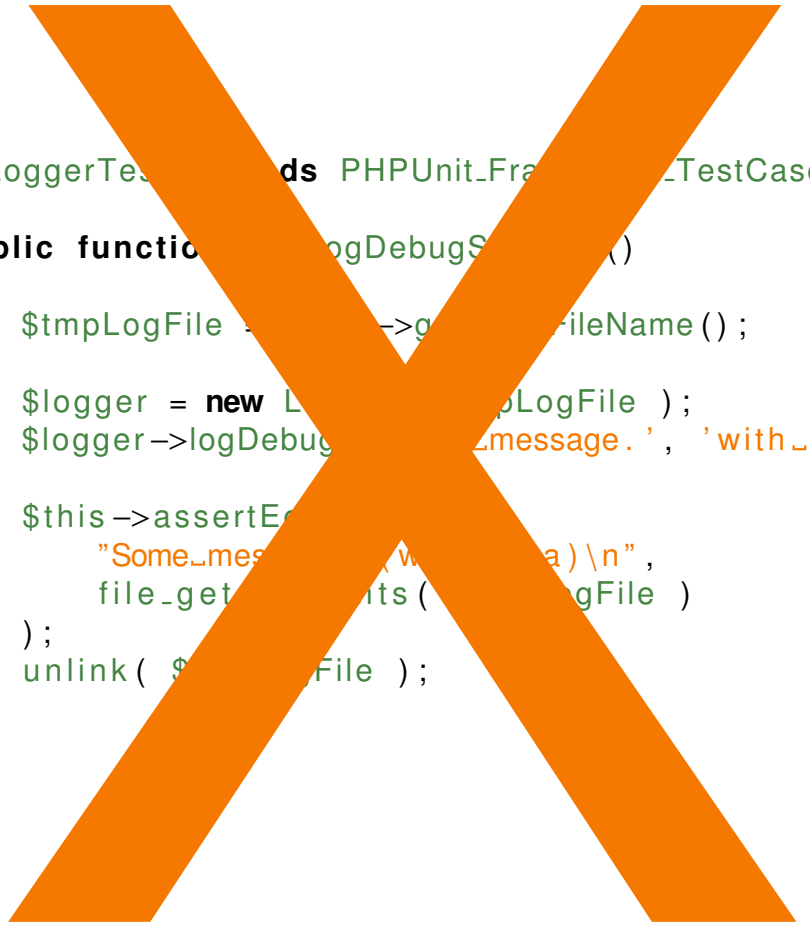
```
1  <?php
2
3  class Logger
4  {
5      public function __construct( $fileName )
6      {
7          // ... error checks ...
8          $this->fileHandle = fopen( $fileName, 'a' );
9      }
10     public function logDebug( $message, $data )
11     {
12         fwrite(
13             $this->fileHandle,
14             sprintf(
15                 "%s (%s)\n",
16                 $message,
17                 $data
18             )
19         );
20     }
21 }
```

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 LoggerTe      ds PHPUnit_Fra      TestCase
4  {
5      public functio      ogDebugS          ()
6      {
7          $tmpLogFile =       ->g       ileName();
8
9          $logger = new L         pLogFile );
10         $logger->logDebug       message.', 'with data' );
11
12         $this->assertE
13             "Some mes       a)\n",
14             file_get        its(       gFile )
15         );
16         unlink( $       File );
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

```
1  <?php
2
3  class LoggerT      ends PHPUnit_Fram      estCase
4  {
5      public funct      tLogDebugSu
6      {
7          vfsStream::       'test'
8          $logFile = v      m::       est' ) . '/message.log';
9
10         $logger = new L          File );
11         $logger->logDebu          message.', 'with data' );
12
13         $this->assertTr
14             vfsStreamW          ot()->hasChild( 'message.log' )
15         );
16         $this->ass          ls(
17             "Some          e.(with          ",
18             file          ntents( $lo
19         );
20     }
21 }
```

The Virtual File System

- Works, but . . .

The Real Issue

E_HARD_SYSTEM_DEPENDENCY

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

Issue #5

```
1   <?php
2
3   class WeatherWidget
4   {
5       protected $loader;
6       protected $userRepository;
7       protected $geolocator;
8       protected $cache;
9       protected $logger;
10
11      public function __construct(
12          WeatherLoader $weatherLoader,
13          UserRepository $userRepository,
14          Geolocator $geolactor,
15          Cache $cache,
16          Logger $logger
17      )
18      {
19      }
20  }
```

Using the DI Container

```php
<?php

class WeatherWidgetTest extends WebTestCase
{
    protected $widget;

    public function setUp()
    {
        $this->widget = $this->getContainer()
            ->get('weather_widget');
    }
}
```

Using the DI Container

```
1   <?php
2
3   class WeatherWidge      xtends      Case
4   {
5       protected $widget
6
7       public function setU
8       {
9           $this->widget =          tContainer()
10              ->get('we                ;
11      }
12  }
```

Using Mocks

```php
<?php

class WeatherWidgetTest extends PHPUnit_Framework_TestCase
{
    protected $widget;
    protected $loader;
    protected $userRepository;
    protected $geolocator;
    protected $cache;
    protected $logger;

    public function setUp()
    {
        $this->loader = $this->getMock('WeatherLoader');
        $this->userRepository = $this->getMock('UserRepository');
        $this->geolocator = $this->getMock('Geolocator');
        $this->cache = $this->getMock('Cache');
        $this->logger = $this->getMock('Logger');

        $this->widget = new WeatherWidget(
            $this->loader,
            $this->userRepository,
            $this->geolocator,
            $this->cache,
            $this->logger
        );
    }
}
```

Using Mocks

```
1   <?php
2
3   class Weathe      Test extends PHPUn      ework_TestCase
4   {
5       protected $
6       protected $lo
7       protected $use      itory;
8       protected $geo
9       protected $cache,
10      protected $logger;
11
12      public function setUp
13      {
14          $this->loader       Mock('WeatherLoader');
15          $this->userR      y       ->getMock('UserRepository');
16          $this->geol       = $th      Mock('Geolocator');
17          $this->cac      his->ge      Cache');
18          $this->lo       $this->ge      Logger');
19
20          $this-      = new WeatherW
21              oader,
22              >userRepository,
23              ->geolocator,
24              s->cache,
25              $this->logger
26          );
27      }
28  }
```

The Real Issue

E_TOO_MANY_DEPENDENCIES

The Fix

```php
<?php

interface WeatherWidget
{
    public function getUserCurrentWeather($username);
}

class WeatherCacheWidget implements WeatherWidget
{
    public function __construct(WeatherWidget $widget, Cache $cache)
    {
    }
}

class WeatherLoggerWidget implements WeatherWidget
{
    public function __construct(WeatherWidget $widget, Logger $logger)
    {
    }
}
```

The Fix

```php
<?php

class UserLocationService
{
    public function __construct(UserRepository $repository, Gelocator $geolocator)
    {
    }
}

class WeatherLoaderWidget implements WeatherWidget
{
    public function __construct(WeatherLoader $loader, UserLocationService $service)
    {
    }
}
```

The Fix

- Combine services using...
- .. aggregation
- .. decoration

Issue #6

```php
<?php

class WeatherWidgetTest extends PHPUnit_Framework_TestCase
{
    public function testUserWeather()
    {
        $user = new User();
        $location = new Location();

        $this->repository
            ->expects($this->once())
            ->method('findByUsername')
            ->with('beberlei')
            ->will($this->returnValue($user));

        $this->geolocator
            ->expects($this->once())
            ->method('getCurrentLocation')
            ->with($this->isSame($user))
            ->will($this->returnValue($location));

        $this->loader
            ->expects($this->once())
            ->method('getWeatherForLocation')
            ->with($this->isSame($location))
            ->will($this->returnValue(new LocatedWeather()));

        // ...
```

The Fix

- Same as before..
- .. combine services
- Minimize service interaction in procedual code

Testing Issues

- Issue #1: Too many responsibilities
- Issue #2: Static dependencies
- Issue #3: Reaching through objects
- Issue #4: System Dependencies
- Issue #5: Too many dependencies
- Issue #6: Rebuilding Logic

Outline

Testing issues

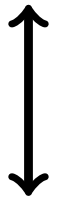
Conclusion

What have we seen?

- Single Responsibility Principle (Issue #1, #5, #6)
- Open Close Principle (Issue #2)
- Law of Demeter (Issue #3)
- Dependency Inversion Principle (Issue #4)

Conclusion

Testable Code



Good OOD

Questions?

Time for Q/A and discussions

THANK YOU

Rent a quality expert
qafoo.com