Mutant Testing
En PHP
Por Miguel G. Flores

@miguelgflores
www.miguelg.com

Hola!

Soy Miguel González Flores

Twitter: **@miguelgflores** Web: **miguelg.com**

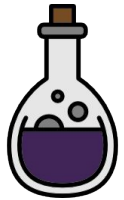
Actualmente CTO en **@finizens** y mantengo otros proyectos como **Empire-Strike.com**

Índice

¿Qué son las pruebas?

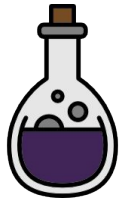
- Comprueban que el sistema hace lo que debe de hacer
- Existen distintos tipos de pruebas
- Automáticas

Pruebas Unitarias



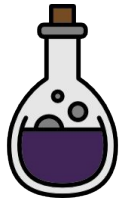
- Aportan valor específicamente en el desarrollo
- Dan un feedback rápido a los cambios
- Ayudan a mejorar el diseño del software
- Comprueban regresión de forma instantánea

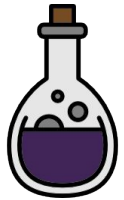
Pruebas Unitarias



```php
class WarIndexCalculator
{
    public function calculate(float $warIndex, float $empireValue, bool $isBeginner): float
    {
        if ($empireValue < 500 || true === $isBeginner) {
            return $warIndex / 2;
        }

        return $warIndex * 1.5;
    }
}
```

```php
class WarIndexCalculatorTest extends TestCase
{
    private $warIndexCalculator;

    protected function setUp(): void
    {
        $this->warIndexCalculator = new WarIndexCalculator();
    }

    public function testCalculateLowEmpireValue(): void
    {
        $result = $this->warIndexCalculator->calculate(100, 300, true);

        $this->assertEquals(50, $result);
    }

    public function testCalculateLowEmpireValue2(): void
    {
        $result = $this->warIndexCalculator->calculate(100, 600, false);

        $this->assertEquals(150, $result);
    }
}
```





```
➜  phpmad-mutant-testing git:(step-1) ✗  phpunit
PHPUnit 6.4.4 by Sebastian Bergmann and contributors.

..                                                                  2 / 2 (100%)

Time: 139 ms, Memory: 6.00MB

OK (2 tests, 2 assertions)
```

Cobertura de código

MutantTesting\WarIndexCalculator		100.00%	1 / 1

```
 9      public function calculate(float $warIndex, float $empireValue, bool $isBeginner): float
10      {
11          if ($empireValue < 500 || true === $isBeginner) {
12              return $warIndex / 2;
13          }
14
15          return $warIndex * 1.5;
16      }
17  }
```

Cobertura de código

- Valor porcentual de cantidad de líneas de código por donde pasan los tests respecto al total.
- Válidas para encontrar caminos no probados.

- Inválidas para determinar la calidad de las pruebas.
- Falsa sensación de confianza

Pruebas de mutación

- Determinan la calidad de los tests

Mutación

```php
if ($empireValue < 500 || true === $isBeginner) {
    return $warIndex / 2;
}
```

```
--- Original
+++ New
@@ @@
-        if ($empireValue < 500 || true === $isBeginner) {
+        if ($empireValue <= 500 || true === $isBeginner) {
```

```
--- Original
+++ New
@@ @@
-        if ($empireValue < 500 || true === $isBeginner) {
+        if ($empireValue < 500 || false === $isBeginner) {
```

```
--- Original
+++ New
@@ @@
-        if ($empireValue < 500 || true === $isBeginner) {
+        if ($empireValue < 500 && true === $isBeginner) {
```

Mutaciones

- Operadores aritméticos
- Intercambio booleanos
- Límites de condicionales
- Negación de condicionales
- Valores de retorno
- Operadores de orden
- Cambio de literales
- Excepciones

Mutante

```php
if ($empireValue < 500 || true === $isBeginner) {
    return $warIndex / 2;
}
```

```
--- Original
+++ New
@@ @@
-        if ($empireValue < 500 || true === $isBeginner) {
+        if ($empireValue <= 500 || true === $isBeginner) {
```

```
--- Original
+++ New
@@ @@
-        if ($empireValue < 500 || true === $isBeginner) {
+        if ($empireValue < 500 || false === $isBeginner) {
```

```
--- Original
+++ New
@@ @@
-        if ($empireValue < 500 || true === $isBeginner) {
+        if ($empireValue < 500 && true === $isBeginner) {
```

Pruebas de mutación

- Aplican **mutaciones** sobre la base de código, generando **mutantes**
- Pasan los tests unitarios (con cierta inteligencia)
- Comprueban el resultado

Posibles resultados

Mutante Muerto

- El test falla después de la mutación
- Otras causas de muerte: timeout, fatal error, ...

Posibles resultados

Mutante Vivo

- El test continúa pasando en verde después de la mutación

Posibles resultados

Mutante No Cubierto

- Ningún test pasa por la línea de código donde está la mutación

Frameworks

Mutant Testing en PHP

- Infection PHP

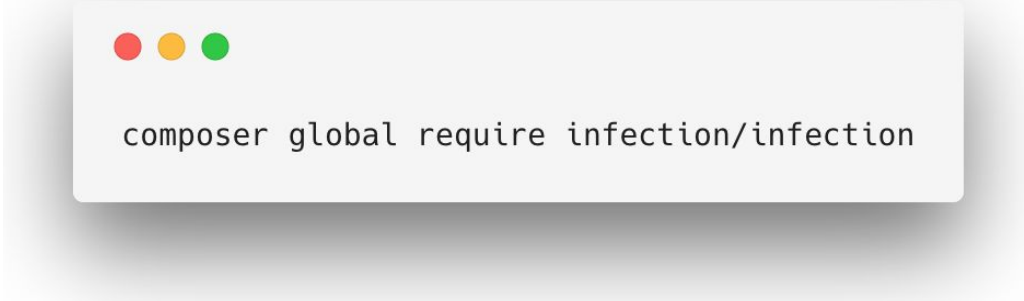
  https://github.com/infection/infection

Infection PHP

- PHP 7.1+
- Usa AST (Abstract Syntax Tree) para realizar las mutaciones
- Funciona con PHPUnit, PHPSpec (y Codeception proximamente)

Infection PHP

Instalación



Infection PHP

Configuración

```
➜  infection

We did not find a configuration file. The following questions will help us to generate it for you.

Which source directories do you want to include (comma separated)? [src]:
  [0] .
  [1] src
  [2] tests
  [3] vendor
 > 1

Single test suite timeout in seconds [10]:

Where do you want to store the text log file? [infection-log.txt]:

Configuration file "infection.json.dist" was created.
```

Ejecución

```
.: killed, M: escaped, S: uncovered, E: fatal error, T: timed out

MMM...M.....                                                (12 / 12)

12 mutations were generated:
       8 mutants were killed
       0 mutants were not covered by tests
       4 covered mutants were not detected
       0 errors were encountered
       0 time outs were encountered

Metrics:
         Mutation Score Indicator (MSI): 67%
         Mutation Code Coverage: 100%
         Covered Code MSI: 67%
```

Infection PHP

Mutation Score Indicator

MSI = (MutantesMuertos / MutantesTotales) * 100

MSI = (**8** / **12**) * 100 = **67%**

Mutation Code Coverage

MCC = **100%**

Infection PHP

Covered Code MSI

CCMSI = **67%**

Mutantes generados

```php
class WarIndexCalculator
{
    public function calculate(float $warIndex, float $empireValue, bool $isBeginner): float
    {
        if ($empireValue < 500 || true === $isBeginner) {
            return $warIndex / 2;
        }

        return $warIndex * 1.5;
    }
}
```

Mutantes generados

```
➜ infection --log-verbosity=2 --show-mutations
```

```
-       if ($empireValue < 500 || true === $isBeginner) {
+       if ($empireValue < 499 || true === $isBeginner) {
```

```
-       if ($empireValue < 500 || true === $isBeginner) {
+       if ($empireValue < 501 || true === $isBeginner) {
```

```
-       if ($empireValue < 500 || true === $isBeginner) {
+       if ($empireValue <= 500 || true === $isBeginner) {
```

```
-       if ($empireValue < 500 || true === $isBeginner) {
+       if ($empireValue < 500 && true === $isBeginner) {
```

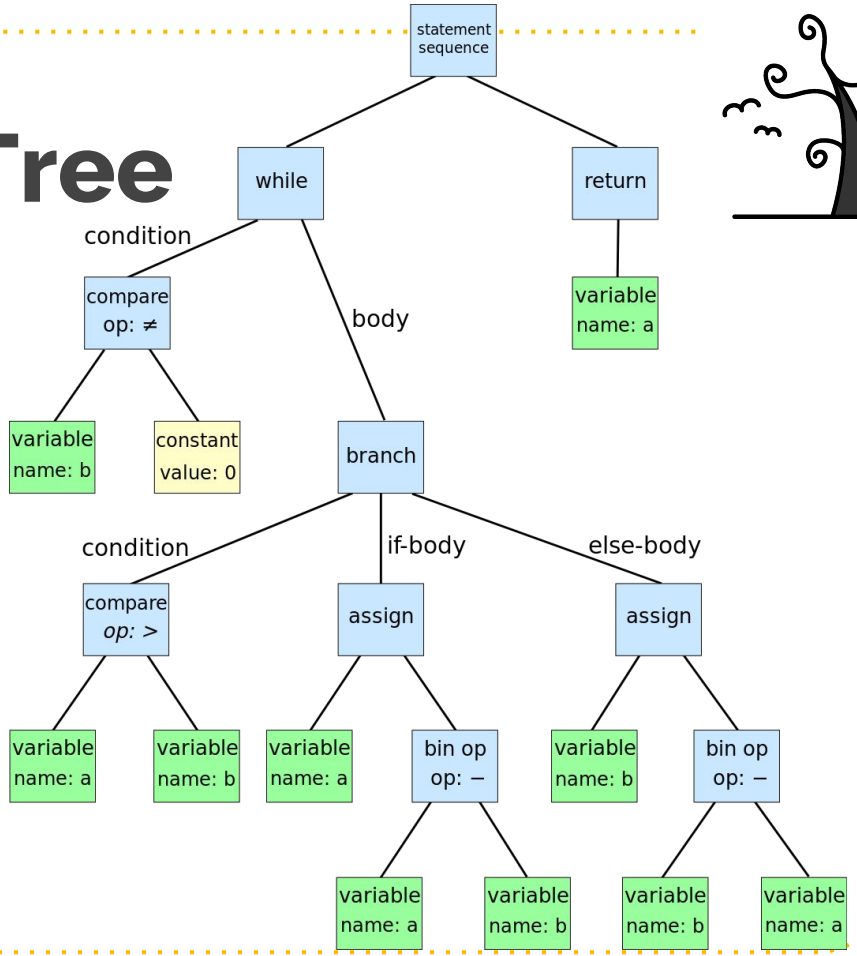
¿Cómo genera esto?

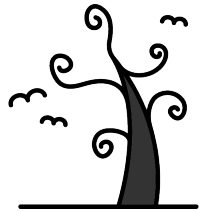
- Usa Abstract Syntax Tree para realizar las mutaciones

Infection PHP

Abstract Syntax Tree

```
while ($b !== 0) {
    if ($a > $b) {
        $a = $a - $b;
    } else {
        $b = $b - $a;
    }
}

return $a;
```





```php
namespace Infection\Mutator\ConditionalBoundary;

final class LessThan extends Mutator
{
    /**
     * Replaces "<" with "<="
     *
     * @param Node $node
     *
     * @return Node\Expr\BinaryOp\SmallerOrEqual
     */
    public function mutate(Node $node)
    {
        return new Node\Expr\BinaryOp\SmallerOrEqual($node->left, $node->right, $node->getAttributes());
    }

    protected function mutatesNode(Node $node): bool
    {
        return $node instanceof Node\Expr\BinaryOp\Smaller;
    }
}
```

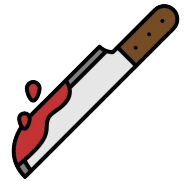
¿Cómo genera esto?

- Usa Abstract Syntax Tree para realizar las mutaciones
- Genera un archivo phpunit.xml por cada mutación
- Intercepta el sistema de archivos para incluir el fichero con la mutación en lugar del original

¿Cómo genera esto?

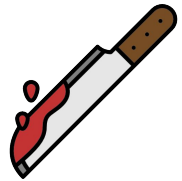
```php
public static function enable(): void
{
    if (!isset(self::$intercept) || !isset(self::$replacement)) {
        throw new \RuntimeException(
            'Set a file to intercept and its replacement before enabling wrapper'
        );
    }
    stream_wrapper_unregister('file');
    stream_wrapper_register('file', __CLASS__);
}

public static function disable(): void
{
    stream_wrapper_restore('file');
}
```

Eliminando mutantes



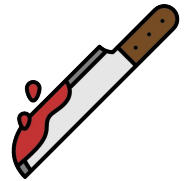
```php
class WarIndexCalculator
{
    public function calculate(float $warIndex, float $empireValue, bool $isBeginner): float
    {
        if ($empireValue < 500 || true === $isBeginner) {
            return $warIndex / 2;
        }

        return $warIndex * 1.5;
    }
}
```

Eliminando mutantes



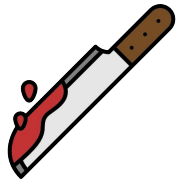
```php
/**
 * @dataProvider calculateWithEmpireValueProvider
 */
public function testCalculateWithEmpireValue(float $expected, float $warIndex, float $empireValue): void
{
    $result = $this->warIndexCalculator->calculate($warIndex, $empireValue, false);

    $this->assertEquals($expected, $result);
}

public function calculateWithEmpireValueProvider(): array
{
    return [
        [50, 100, 499],
        [150, 100, 500],
        [150, 100, 501],
    ];
}
```

Eliminando mutantes



```php
public function testCalculateWithBeginners(): void
{
    $result = $this->warIndexCalculator->calculate(100, 600, true);

    $this->assertEquals(50, $result);
}
```

Eliminando mutantes



```
.: killed, M: escaped, S: uncovered, E: fatal error, T: timed out

............                                                (12 / 12)

12 mutations were generated:
      12 mutants were killed
       0 mutants were not covered by tests
       0 covered mutants were not detected
       0 errors were encountered
       0 time outs were encountered

Metrics:
         Mutation Score Indicator (MSI): 100%
         Mutation Code Coverage: 100%
         Covered Code MSI: 100%
```

A tener en cuenta...

Falsos positivos

- Algunos mutantes son imposibles de matar

```php
public function exists(string $needle): bool
{
    return \in_array($needle, ['a', 'b', 'c'], true);
}
```

```
-- --
-         return \in_array($needle, ['a', 'b', 'c'], true);
+         return \in_array($needle, ['a', 'b', 'c'], false);
```

Valores en constantes

- https://github.com/infection/infection/issues/315

```php
class Calculator
{
    private const LIMIT = 500;

    public function calculate(float $a): bool
    {
        return $a < self::LIMIT;
    }
}
```

Velocidad

```
→ infection --threads=4 --filter=src/Domain --only-covered
```

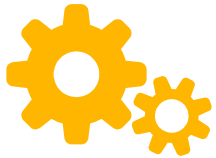
Velocidad

finizens

- Clases: **~3.300**
- LOC: **~170.000**
- Tests: **~4.400**
- Asserts: **~10.250**
- Code coverage: **86%**
- MT log result: **22MB**

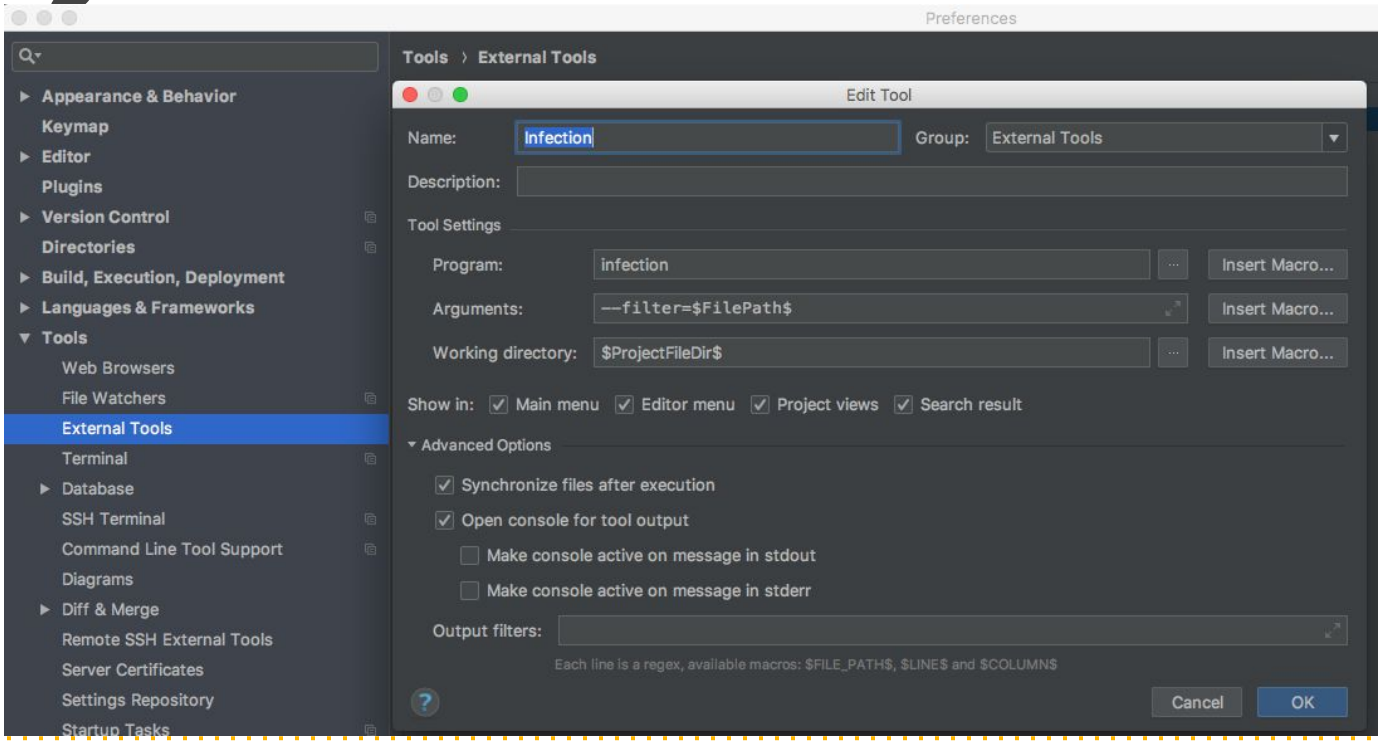
```
18131 mutations were generated:
    8779 mutants were killed
    4617 mutants were not covered by tests
    4066 covered mutants were not detected
     533 errors were encountered
     136 time outs were encountered

Metrics:
        Mutation Score Indicator (MSI): 52%
        Mutation Code Coverage: 75%
        Covered Code MSI: 70%
```

Integración continua

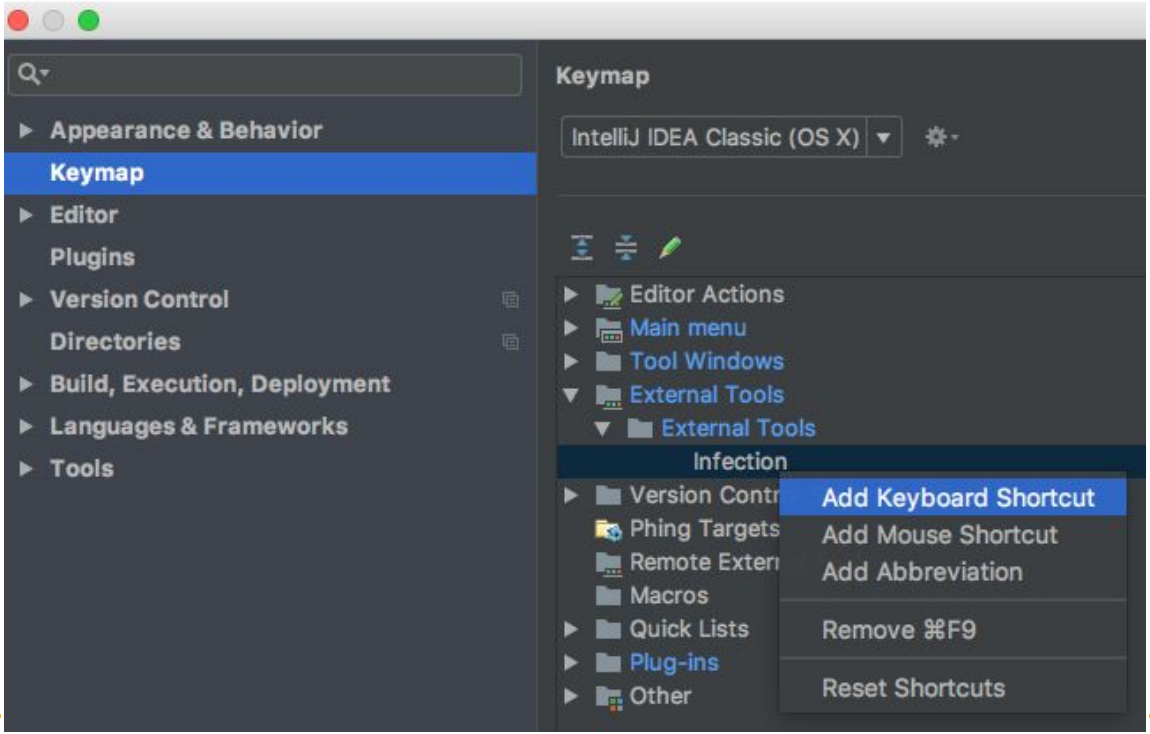


```
before_script:
    - wget https://github.com/infection/infection/releases/download/0.8.1/infection.phar
    - wget https://github.com/infection/infection/releases/download/0.8.1/infection.phar.pubkey
    - chmod +x infection.phar
script:
    - ./infection.phar --min-covered-msi=90 --threads=4
```

Flujo desarollo



Flujo desarollo



Conclusiones

- Los test de mutación determinan la calidad de los tests, no del código
- La cobertura de código como métrica única no es fiable
- Puede ser usado como un filtro más en el sistema de Integración Continua

Referencias

- Infection PHP

  https://github.com/infection/infection
- AST

  https://wiki.php.net/rfc/abstract_syntax_tree
- Pruebas de mutación en PHP (Blog Finizens Engineering)

  https://medium.com/finizens-engineering/pruebas-de-mutaci%C3%B3n-en-php-fad6ba181628

¡Muchas Gracias!

¿Preguntas?

@miguelgflores
www.miguelg.com