PHPUnit

latest

Sebastian Bergmann

2020-12-17

Contents

PHPUnit latest 2020-12-17

Sebastian Bergmann

CHAPTER 1

PHPUnit

1.1

PHPUnit latest PHP 7.3 PHP

PHPUnit dom json

PHPUnit pcre reflection spl PHP C

Xdebug 2.7.0 tokenizer XML xmlwriter

1.2 PHP PHAR

PHPUnit PHPUnit PHP PHAR PHPUnit

PHP PHAR phar

Suhosin `php.ini` PHAR

```
suhosin.executor.include.whitelist = phar
```

PHPUnit PHAR

```
$ wget https://phar.phpunit.de/phpunit-latest.phar
$ php phpunit-latest.phar --version
PHPUnit x.y.z by Sebastian Bergmann and contributors.
```

PHAR

```
$ wget https://phar.phpunit.de/phpunit-latest.phar
$ chmod +x phpunit-latest.phar
$ ./phpunit-latest.phar --version
PHPUnit x.y.z by Sebastian Bergmann and contributors.
```

1.2.1 PHPUnit PHAR

PHPUnit phar.phpunit.de PGP SHA256

`phpunit.phar` PGP `phpunit.phar.asc`

```
$ wget https://phar.phpunit.de/phpunit-latest.phar
$ wget https://phar.phpunit.de/phpunit-latest.phar.asc
```

phpunit-x.y.phar PHPUnit PHP phpunit-x.y.phar.asc

```
$ gpg phpunit-latest.phar.asc
gpg: Signature made Sat 19 Jul 2014 01:28:02 PM CEST using RSA key ID 6372C20A
gpg: Can't check signature: public key not found
```

6372C20A pgp.uni-mainz.de

```
$ curl --silent https://sebastian-bergmann.de/gpg.asc | gpg --import
gpg: key 4AA394086372C20A: 452 signatures not checked due to missing keys
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 4AA394086372C20A: public key "Sebastian Bergmann <sb@sebastian-bergmann.de>" imported
gpg: Total number processed: 1
gpg:               imported: 1
gpg: no ultimately trusted keys found
```

"Sebastian Bergmann <sb@sebastian-bergmann.de>" Sebastian Bergmann HTML

```
$ gpg phpunit-latest.phar.asc
gpg: Signature made Sat 19 Jul 2014 01:28:02 PM CEST using RSA key ID 6372C20A
gpg: Good signature from "Sebastian Bergmann <sb@sebastian-bergmann.de>"
gpg:                 aka "Sebastian Bergmann <sebastian@php.net>"
gpg:                 aka "Sebastian Bergmann <sebastian@thephp.cc>"
gpg:                 aka "Sebastian Bergmann <sebastian@phpunit.de>"
gpg:                 aka "Sebastian Bergmann <sebastian.bergmann@thephp.cc>"
gpg:                 aka "[jpeg image of size 40635]"
gpg: WARNING: This key is not certified with a trusted signature!
gpg:          There is no indication that the signature belongs to the owner.
Primary key fingerprint: D840 6D0D 8294 7747 2937  7831 4AA3 9408 6372 C20A
```

6372C20A Sebastian Bergmann

" "

GPG PHPUnit PHAR PHAR PHIVE PHIVE

1.3 Composer

Composer composer.json phpunit/phpunit

```
composer require --dev phpunit/phpunit ^latest
```

1.4

PHPUnit /usr/bin/phpunit /usr/local/bin/phpunit

PHPUnit

PHPUnit PHAR tools PHIVE Composer composer.json PHPUnit

1.5 Web

PHPUnit PHPUnit Web

PHPUnit Web `vendor` **Web**

PHPUnit Web “ ”

CHAPTER 2

PHPUnit

2.1 PHPUnit PHP PHPUnit

1. `Class` `ClassTest`
2. `ClassTest` `PHPUnit\Framework\TestCase`
3. `test*`

   docblock `@test`
4. `assertSame()`

2.1: PHPUnit

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StackTest extends TestCase
{
    public function testPushAndPop(): void
    {
        $stack = [];
        $this->assertSame(0, count($stack));

        array_push($stack, 'foo');
        $this->assertSame('foo', $stack[count($stack)-1]);
        $this->assertSame(1, count($stack));

        $this->assertSame('foo', array_pop($stack));
        $this->assertSame(0, count($stack));
    }
}
```

Martin Fowler

`print`

2.1

Adrian Kuhn et. al.

bug

PHPUnit producer fixture consumer

- producer
- consumer

2.2 @depends

2.2: @depends

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StackTest extends TestCase
{
    public function testEmpty(): array
    {
        $stack = [];
        $this->assertEmpty($stack);

        return $stack;
    }

    /**
     * @depends testEmpty
     */
    public function testPush(array $stack): array
    {
        array_push($stack, 'foo');
        $this->assertSame('foo', $stack[count($stack)-1]);
        $this->assertNotEmpty($stack);

        return $stack;
    }

    /**
     * @depends testPush
     */
    public function testPop(array $stack): void
    {
        $this->assertSame('foo', array_pop($stack));
        $this->assertEmpty($stack);
    }
}
```

testEmpty() testPush() testEmpty() testPop() testPush()

“ ” a @depends clone b @depends shallowClone PHP clone

PHPUnit 2.3

2.3:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class DependencyFailureTest extends TestCase
{
    public function testOne(): void
    {
        $this->assertTrue(false);
    }

    /**
     * @depends testOne
     */
    public function testTwo(): void
    {
    }
}
```

```
$ phpunit --verbose DependencyFailureTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

FS

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) DependencyFailureTest::testOne
Failed asserting that false is true.

/home/sb/DependencyFailureTest.php:6

There was 1 skipped test:

1) DependencyFailureTest::testTwo
This test depends on "DependencyFailureTest::testOne" to pass.

FAILURES!
Tests: 1, Assertions: 1, Failures: 1, Skipped: 1.
```

`@depends` PHPUnit

`@depends` 2.4

2.4:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class MultipleDependenciesTest extends TestCase
{
    public function testProducerFirst(): string
    {
        $this->assertTrue(true);

        return 'first';
    }

    public function testProducerSecond(): string
    {
```

```
        $this->assertTrue(true);

        return 'second';
    }

    /**
     * @depends testProducerFirst
     * @depends testProducerSecond
     */
    public function testConsumer(string $a, string $b): void
    {
        $this->assertSame('first', $a);
        $this->assertSame('second', $b);
    }
}
```

```
$ phpunit --verbose MultipleDependenciesTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

...

Time: 0 seconds, Memory: 3.25Mb

OK (3 tests, 4 assertions)
```

2.2

2.5 additionProvider() @dataProvider

public Iterator

2.5:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class DataTest extends TestCase
{
    /**
     * @dataProvider additionProvider
     */
    public function testAdd(int $a, int $b, int $expected): void
    {
        $this->assertSame($expected, $a + $b);
    }

    public function additionProvider(): array
    {
        return [
            [0, 0, 0],
            [0, 1, 1],
            [1, 0, 1],
            [1, 1, 3]
        ];
    }
}
```

```
$ phpunit DataTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.
```

```
...F

Time: 0 seconds, Memory: 5.75Mb

There was 1 failure:

1) DataTest::testAdd with data set #3 (1, 1, 3)
Failed asserting that 2 is identical to 3.

/home/sb/DataTest.php:9

FAILURES!
Tests: 4, Assertions: 4, Failures: 1.
```

2.6:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class DataTest extends TestCase
{
    /**
     * @dataProvider additionProvider
     */
    public function testAdd(int $a, int $b, int $expected): void
    {
        $this->assertSame($expected, $a + $b);
    }

    public function additionProvider(): array
    {
        return [
            'adding zeros'  => [0, 0, 0],
            'zero plus one' => [0, 1, 1],
            'one plus zero' => [1, 0, 1],
            'one plus one'  => [1, 1, 3]
        ];
    }
}
```

```
$ phpunit DataTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

...F

Time: 0 seconds, Memory: 5.75Mb

There was 1 failure:

1) DataTest::testAdd with data set "one plus one" (1, 1, 3)
Failed asserting that 2 is identical to 3.

/home/sb/DataTest.php:9

FAILURES!
Tests: 4, Assertions: 4, Failures: 1.
```

2.7: Iterator

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class DataTest extends TestCase
{
    /**
     * @dataProvider additionProvider
     */
    public function testAdd(int $a, int $b, int $expected): void
    {
        $this->assertSame($expected, $a + $b);
    }

    public function additionProvider(): CsvFileIterator
    {
        return new CsvFileIterator('data.csv');
    }
}
```

```
$ phpunit DataTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

...F

Time: 0 seconds, Memory: 5.75Mb

There was 1 failure:

1) DataTest::testAdd with data set #3 ('1', '1', '3')
Failed asserting that 2 is identical to 3.

/home/sb/DataTest.php:11

FAILURES!
Tests: 4, Assertions: 4, Failures: 1.
```

2.8: CsvFileIterator

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class CsvFileIterator implements Iterator
{
    private $file;
    private $key = 0;
    private $current;

    public function __construct(string $file)
    {
        $this->file = fopen($file, 'r');
    }

    public function __destruct()
    {
        fclose($this->file);
    }

    public function rewind(): void
    {
```

```
        rewind($this->file);

        $this->current = fgetcsv($this->file);
        $this->key     = 0;
    }

    public function valid(): bool
    {
        return !feof($this->file);
    }

    public function key(): int
    {
        return $this->key;
    }

    public function current(): array
    {
        return $this->current;
    }

    public function next(): void
    {
        $this->current = fgetcsv($this->file);

        $this->key++;
    }
}
```

`@dataProvider` `@depends` 2.9

2.9: `@depends` `@dataProvider`

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class DependencyAndDataProviderComboTest extends TestCase
{
    public function provider(): array
    {
        return [['provider1'], ['provider2']];
    }

    public function testProducerFirst(): void
    {
        $this->assertTrue(true);

        return 'first';
    }

    public function testProducerSecond(): void
    {
        $this->assertTrue(true);

        return 'second';
    }

    /**
     * @depends testProducerFirst
     * @depends testProducerSecond
     * @dataProvider provider
     */
    public function testConsumer(): void
```

```
    {
        $this->assertSame(
            ['provider1', 'first', 'second'],
            func_get_args()
        );
    }
}
```

```
$ phpunit --verbose DependencyAndDataProviderComboTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

...F

Time: 0 seconds, Memory: 3.50Mb

There was 1 failure:

1) DependencyAndDataProviderComboTest::testConsumer with data set #1 ('provider2')
Failed asserting that two arrays are identical.
--- Expected
+++ Actual
@@ @@
Array &0 (
-    0 => 'provider1'
+    0 => 'provider2'
     1 => 'first'
     2 => 'second'
)
/home/sb/DependencyAndDataProviderComboTest.php:32

FAILURES!
Tests: 4, Assertions: 4, Failures: 1.
```

2.10:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class DataTest extends TestCase
{
    /**
     * @dataProvider additionWithNonNegativeNumbersProvider
     * @dataProvider additionWithNegativeNumbersProvider
     */
    public function testAdd(int $a, int $b, int $expected): void
    {
        $this->assertSame($expected, $a + $b);
    }

    public function additionWithNonNegativeNumbersProvider(): void
    {
        return [
            [0, 1, 1],
            [1, 0, 1],
            [1, 1, 3]
        ];
    }

    public function additionWithNegativeNumbersProvider(): array
    {
        return [
```

```
            [-1, 1, 0],
            [-1, -1, -2],
            [1, -1, 0]
        ];
    }
}
```

```
$ phpunit DataTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

..F...                                                              6 / 6 (100%)

Time: 0 seconds, Memory: 5.75Mb

There was 1 failure:

1) DataTest::testAdd with data set #3 (1, 1, 3)
Failed asserting that 2 is identical to 3.

/home/sb/DataTest.php:12

FAILURES!
Tests: 6, Assertions: 6, Failures: 1.
```

---

setUpBeforeClass() setUp() PHPUnit

2.3

2.11 @expectException

2.11: expectException()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ExceptionTest extends TestCase
{
    public function testException(): void
    {
        $this->expectException(InvalidArgumentException::class);
    }
}
```

```
$ phpunit ExceptionTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb
```

```
There was 1 failure:

1) ExceptionTest::testException
Failed asserting that exception of type "InvalidArgumentException" is thrown.

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

expectException() expectExceptionCode() expectExceptionMessage() expectExceptionMessageMatches()

---

expectExceptionMessage() $actual $expected

2.4 PHP

PHPUnit PHP PHP 2.12

PHP error_reporting PHPUnit PHP

2.12: PHP

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ErrorTest extends TestCase
{
    public function testDeprecationCanBeExpected(): void
    {
        $this->expectDeprecation();

        //
        $this->expectDeprecationMessage('foo');

        //
        $this->expectDeprecationMessageMatches('/foo/');

        \trigger_error('foo', \E_USER_DEPRECATED);
    }

    public function testNoticeCanBeExpected(): void
    {
        $this->expectNotice();

        //
        $this->expectNoticeMessage('foo');

        //
        $this->expectNoticeMessageMatches('/foo/');

        \trigger_error('foo', \E_USER_NOTICE);
    }

    public function testWarningCanBeExpected(): void
    {
```

```
        $this->expectWarning();

        //
        $this->expectWarningMessage('foo');

        //
        $this->expectWarningMessageMatches('/foo/');

        \trigger_error('foo', \E_USER_WARNING);
    }

    public function testErrorCanBeExpected(): void
    {
        $this->expectError();

        //
        $this->expectErrorMessage('foo');

        //
        $this->expectErrorMessageMatches('/foo/');

        \trigger_error('foo', \E_USER_ERROR);
    }
}
```

PHP fopen PHPUnit

2.13: PHP

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ErrorSuppressionTest extends TestCase
{
    public function testFileWriting(): void
    {
        $writer = new FileWriter;

        $this->assertFalse(@$writer->write('/is-not-writeable/file', 'stuff'));
    }
}

final class FileWriter
{
    public function write($file, $content)
    {
        $file = fopen($file, 'w');

        if ($file === false) {
            return false;
        }

        // ...
    }
}
```

```
$ phpunit ErrorSuppressionTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

.

Time: 1 seconds, Memory: 5.25Mb
```

```
OK (1 test, 1 assertion)
```

```
                fopen(/is-not-writeable/file): failed to open stream: No such file or
directory
```

2.5

echo print PHPUnit\Framework\TestCase PHP

2.14 expectOutputString()

2.14:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class OutputTest extends TestCase
{
    public function testExpectFooActualFoo(): void
    {
        $this->expectOutputString('foo');

        print 'foo';
    }

    public function testExpectBarActualBaz(): void
    {
        $this->expectOutputString('bar');

        print 'baz';
    }
}
```

```
$ phpunit OutputTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

.F

Time: 0 seconds, Memory: 5.75Mb

There was 1 failure:

1) OutputTest::testExpectBarActualBaz
Failed asserting that two strings are equal.
--- Expected
+++ Actual
@@ @@
-'bar'
+'baz'

FAILURES!
Tests: 2, Assertions: 2, Failures: 1.
```

2.1

2.1:

void expectOutputRegex(string $regularExpression)	$regularExpression
void expectOutputString(string $expectedString)	$expectedString
bool setOutputCallback(callable $callback)	
string getActualOutput()	

2.6

PHPUnit

2.15:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ArrayDiffTest extends TestCase
{
    public function testEquality(): void
    {
        $this->assertSame(
            [1, 2,  3, 4, 5, 6],
            [1, 2, 33, 4, 5, 6]
        );
    }
}
```

```
$ phpunit ArrayDiffTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) ArrayDiffTest::testEquality
Failed asserting that two arrays are identical.
--- Expected
+++ Actual
@@ @@
 Array (
     0 => 1
     1 => 2
-    2 => 3
+    2 => 33
     3 => 4
     4 => 5
     5 => 6
 )
```

```
/home/sb/ArrayDiffTest.php:7

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

PHPUnit

2.16:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class LongArrayDiffTest extends TestCase
{
    public function testEquality(): void
    {
        $this->assertSame(
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2,  3, 4, 5, 6],
            [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 33, 4, 5, 6]
        );
    }
}
```

```
$ phpunit LongArrayDiffTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) LongArrayDiffTest::testEquality
Failed asserting that two arrays are identical.
--- Expected
+++ Actual
@@ @@
     11 => 0
     12 => 1
     13 => 2
-    14 => 3
+    14 => 33
     15 => 4
     16 => 5
     17 => 6
 )

/home/sb/LongArrayDiffTest.php:7

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

2.6.1

PHPUnit

assertEquals() “ ”

2.17:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ArrayWeakComparisonTest extends TestCase
{
    public function testEquality(): void
    {
        $this->assertEquals(
            [1, 2, 3, 4, 5, 6],
            ['1', 2, 33, 4, 5, 6]
        );
    }
}
```

```
$ phpunit ArrayWeakComparisonTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) ArrayWeakComparisonTest::testEquality
Failed asserting that two arrays are equal.
--- Expected
+++ Actual
@@ @@
 Array (
-    0 => 1
+    0 => '1'
     1 => 2
-    2 => 3
+    2 => 33
     3 => 4
     4 => 5
     5 => 6
 )

/home/sb/ArrayWeakComparisonTest.php:7

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

```
         1  '1'        assertEquals()
```

CHAPTER 3

PHPUnit `phpunit` PHPUnit

```
$ phpunit ArrayTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

..

Time: 0 seconds

OK (2 tests, 2 assertions)
```

PHPUnit `ArrayTest.php` `ArrayTest`

PHPUnit

```
.

F

E

R

S

I
```

PHPUnit *failure* *error* PHPUnit `assertSame()` exception PHP

3.1

```
$ phpunit --help
PHPUnit latest.0 by Sebastian Bergmann and contributors.

Usage:
  phpunit [options] UnitTest.php
  phpunit [options] <directory>

Code Coverage Options:
  --coverage-clover <file>    Generate code coverage report in Clover XML format
  --coverage-crap4j <file>    Generate code coverage report in Crap4J XML format
  --coverage-html <dir>       Generate code coverage report in HTML format
  --coverage-php <file>       Export PHP_CodeCoverage object to file
  --coverage-text <file>      Generate code coverage report in text format [default:␣
↪standard output]
  --coverage-xml <dir>        Generate code coverage report in PHPUnit XML format
  --coverage-cache <dir>      Cache static analysis results
  --warm-coverage-cache       Warm static analysis cache
  --coverage-filter <dir>     Include <dir> in code coverage analysis
  --path-coverage             Perform path coverage analysis
  --disable-coverage-ignore   Disable annotations for ignoring code coverage
  --no-coverage               Ignore code coverage configuration

Logging Options:
  --log-junit <file>          Log test execution in JUnit XML format to file
  --log-teamcity <file>       Log test execution in TeamCity format to file
  --testdox-html <file>       Write agile documentation in HTML format to file
  --testdox-text <file>       Write agile documentation in Text format to file
  --testdox-xml <file>        Write agile documentation in XML format to file
  --reverse-list              Print defects in reverse order
  --no-logging                Ignore logging configuration

Test Selection Options:
  --filter <pattern>          Filter which tests to run
  --testsuite <name>          Filter which testsuite to run
  --group <name>              Only runs tests from the specified group(s)
  --exclude-group <name>      Exclude tests from the specified group(s)
  --list-groups               List available test groups
  --list-suites               List available test suites
  --list-tests                List available tests
  --list-tests-xml <file>     List available tests in XML format
  --test-suffix <suffixes>    Only search for test in files with specified suffix(es).
↪ Default: Test.php,.phpt

Test Execution Options:
  --dont-report-useless-tests Do not report tests that do not test anything
  --strict-coverage           Be strict about @covers annotation usage
  --strict-global-state       Be strict about changes to global state
  --disallow-test-output      Be strict about output during tests
  --disallow-resource-usage   Be strict about resource usage during small tests
  --enforce-time-limit        Enforce time limit based on test size
  --default-time-limit <sec>  Timeout in seconds for tests without @small, @medium or␣
↪@large
  --disallow-todo-tests       Disallow @todo-annotated tests
```

```
  --process-isolation         Run each test in a separate PHP process
  --globals-backup            Backup and restore $GLOBALS for each test
  --static-backup             Backup and restore static attributes for each test

  --colors <flag>             Use colors in output ("never", "auto" or "always")
  --columns <n>               Number of columns to use for progress output
  --columns max               Use maximum number of columns for progress output
  --stderr                    Write to STDERR instead of STDOUT
  --stop-on-defect            Stop execution upon first not-passed test
  --stop-on-error             Stop execution upon first error
  --stop-on-failure           Stop execution upon first error or failure
  --stop-on-warning           Stop execution upon first warning
  --stop-on-risky             Stop execution upon first risky test
  --stop-on-skipped           Stop execution upon first skipped test
  --stop-on-incomplete        Stop execution upon first incomplete test
  --fail-on-incomplete        Treat incomplete tests as failures
  --fail-on-risky             Treat risky tests as failures
  --fail-on-skipped           Treat skipped tests as failures
  --fail-on-warning           Treat tests with warnings as failures
  -v|--verbose                Output more verbose information
  --debug                     Display debugging information

  --repeat <times>            Runs the test(s) repeatedly
  --teamcity                  Report test execution progress in TeamCity format
  --testdox                   Report test execution progress in TestDox format
  --testdox-group             Only include tests from the specified group(s)
  --testdox-exclude-group     Exclude tests from the specified group(s)
  --no-interaction            Disable TestDox progress animation
  --printer <printer>         TestListener implementation to use

  --order-by <order>          Run tests in order: default|defects|duration|no-
↪depends|random|reverse|size
  --random-order-seed <N>     Use a specific random seed <N> for random order
  --cache-result              Write test results to cache file
  --do-not-cache-result       Do not write test results to cache file

Configuration Options:
  --prepend <file>            A PHP script that is included as early as possible
  --bootstrap <file>          A PHP script that is included before the tests run
  -c|--configuration <file>   Read configuration from XML file
  --no-configuration          Ignore default configuration file (phpunit.xml)
  --extensions <extensions>   A comma separated list of PHPUnit extensions to load
  --no-extensions             Do not load PHPUnit extensions
  --include-path <path(s)>    Prepend PHP's include_path with given path(s)
  -d <key[=value]>            Sets a php.ini value
  --cache-result-file <file>  Specify result cache path and filename
  --generate-configuration    Generate configuration file with suggested settings
  --migrate-configuration     Migrate configuration file to current format

Miscellaneous Options:
  -h|--help                   Prints this usage information
  --version                   Prints the version and exits
  --atleast-version <min>     Checks that version is greater than min and exits
  --check-version             Check whether PHPUnit is the latest version
```

```
phpunit UnitTest
```

```
       UnitTest        UnitTest.php

    UnitTest                PHPUnit\Framework\TestCase     public static suite()
```

PHPUnit\Framework\Test PHPUnit\Framework\TestSuite

phpunit UnitTest UnitTest.php

UnitTest

--coverage-clover

XML

--coverage-crap4j

Crap4j

--coverage-html

HTML

--coverage-php

PHP_CodeCoverage

--coverage-text

--log-junit

JUnit XML

--testdox-html --testdox-text

HTML *TestDox*

--filter

PHPUnit /

TestNamespace\TestCaseClass::testMethod

__METHOD__

TestNamespace\TestCaseClass::testMethod with data set #0

TestNamespace\TestCaseClass::testMethod with data set "my named data"

3.1

3.1:

```php
<?php
use PHPUnit\Framework\TestCase;

namespace TestNamespace;

class TestCaseClass extends TestCase
{
    /**
     * @dataProvider provider
     */
    public function testMethod($data)
    {
        $this->assertTrue($data);
    }

    public function provider()
    {
        return [
            'my named data' => [true],
```

```
            'my data'       => [true]
        ];
    }
}
```

/path/to/my/test.phpt

PHPT

3.2

3.2:

```
--filter 'TestNamespace\\TestCaseClass::testMethod'
--filter 'TestNamespace\\TestCaseClass'
--filter TestNamespace
--filter TestCaseClase
--filter testMethod
--filter '/::testMethod .*"my named data"/'
--filter '/::testMethod .*#5$/'
--filter '/::testMethod .*#(5|6|7)$/'
```

3.3

3.3:

```
--filter 'testMethod#2'
--filter 'testMethod#2-4'
--filter '#2'
--filter '#2-4'
--filter 'testMethod@my named data'
--filter 'testMethod@my.*data'
--filter '@my named data'
--filter '@my.*data'
```

--testsuite

--group

@group

@author @ticket @group ID

--exclude-group

@group

--list-groups

--test-suffix

--dont-report-useless-tests

--strict-coverage

--strict-global-state

--disallow-test-output

--disallow-todo-tests
@todo
--enforce-time-limit

--process-isolation
PHP
--no-globals-backup
$GLOBALS
--static-backup

--colors
Windows ANSICON ConEmu

- never --colors
- auto
- always

--colors auto
--columns
max
--stderr
STDERR STDOUT
--stop-on-error

--stop-on-failure

--stop-on-risky

--stop-on-skipped

--stop-on-incomplete

--verbose

--debug

--loader

PHPUnit\Runner\TestSuiteLoader

PHP include_path Project_Package_Class Project/Package/Class.php

--repeat

--testdox

TestDox *TestDox*

--printer

printer PHPUnit\Util\Printer PHPUnit\Framework\TestListener

--bootstrap

"bootstrap" PHP

--configuration -c

XML *XML*

phpunit.xml phpunit.xml.dist --configuration

phpunit.xml phpunit.xml.dist

--no-configuration

phpunit.xml phpunit.xml.dist

--include-path

PHP *include_path*

-d

PHP

3.2 TestDox

PHPUnit TestDox camel case snake_case PHP testBalanceIsInitiallyZero() test_balance_is_initially_zero() "Balance is initially zero" testBalanceCannotBecomeNegative() testBalanceCannotBecomeNegative2() "Balance cannot become negative"

BankAccount

```
$ phpunit --testdox BankAccountTest.php
PHPUnit latest.0 by Sebastian Bergmann and contributors.

BankAccount
  Balance is initially zero
  Balance cannot become negative
```

HTML --testdox-html --testdox-text

CHAPTER 4

fixture

fixture

PHPUnit $stack

PHPUnit setUp() setUp() tearDown() tearDown()

@depends - 4.1 StackTest $stack array setUp() assertSame() $this->stack $stack

4.1: setUp()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StackTest extends TestCase
{
    private $stack;

    protected function setUp(): void
    {
        $this->stack = [];
    }

    public function testEmpty(): void
    {
        $this->assertTrue(empty($this->stack));
    }

    public function testPush(): void
    {
        array_push($this->stack, 'foo');

        $this->assertSame('foo', $this->stack[count($this->stack)-1]);
        $this->assertFalse(empty($this->stack));
    }

    public function testPop(): void
    {
```

```
        array_push($this->stack, 'foo');

        $this->assertSame('foo', array_pop($this->stack));
        $this->assertTrue(empty($this->stack));
    }
}
```

setUp() tearDown()

setUpBeforeClass() tearDownAfterClass()

4.2:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class TemplateMethodsTest extends TestCase
{
    public static function setUpBeforeClass(): void
    {
        fwrite(STDOUT, __METHOD__ . "\n");
    }

    protected function setUp(): void
    {
        fwrite(STDOUT, __METHOD__ . "\n");
    }

    protected function assertPreConditions(): void
    {
        fwrite(STDOUT, __METHOD__ . "\n");
    }

    public function testOne(): void
    {
        fwrite(STDOUT, __METHOD__ . "\n");
        $this->assertTrue(true);
    }

    public function testTwo(): void
    {
        fwrite(STDOUT, __METHOD__ . "\n");
        $this->assertTrue(false);
    }

    protected function assertPostConditions(): void
    {
        fwrite(STDOUT, __METHOD__ . "\n");
    }

    protected function tearDown(): void
    {
        fwrite(STDOUT, __METHOD__ . "\n");
    }

    public static function tearDownAfterClass(): void
    {
        fwrite(STDOUT, __METHOD__ . "\n");
    }

    protected function onNotSuccessfulTest(Throwable $t): void
    {
```

```
        fwrite(STDOUT, __METHOD__ . "\n");
        throw $t;
    }
}
```

```
$ phpunit TemplateMethodsTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

TemplateMethodsTest::setUpBeforeClass
TemplateMethodsTest::setUp
TemplateMethodsTest::assertPreConditions
TemplateMethodsTest::testOne
TemplateMethodsTest::assertPostConditions
TemplateMethodsTest::tearDown
.TemplateMethodsTest::setUp
TemplateMethodsTest::assertPreConditions
TemplateMethodsTest::testTwo
TemplateMethodsTest::tearDown
TemplateMethodsTest::onNotSuccessfulTest
FTemplateMethodsTest::tearDownAfterClass

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) TemplateMethodsTest::testTwo
Failed asserting that <boolean:false> is true.
/home/sb/TemplateMethodsTest.php:30

FAILURES!
Tests: 2, Assertions: 2, Failures: 1.
```

4.1 setUp() tearDown()

setUp() tearDown() setUp() tearDown() setUp() PHP tearDown() setUp() tearDown() unset()

4.2

- setUp() setUp()
- setUp()

4.3

4.3 setUpBeforeClass() tearDownAfterClass()

4.3:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class DatabaseTest extends TestCase
{
    private static $dbh;

    public static function setUpBeforeClass(): void
    {
        self::$dbh = new PDO('sqlite::memory:');
    }

    public static function tearDownAfterClass(): void
    {
        self::$dbh = null;
    }
}
```

stub

4.4

singleton

PHP

- $foo = 'bar'; $GLOBALS['foo'] = 'bar';
- $GLOBALS
-
- $foo $GLOBALS['foo'] global $foo;

6 PHPUnit ($GLOBALS $_ENV $_POST $_GET $_COOKIE $_SERVER $_FILES $_REQUEST)

6 PHPUnit --globals-backup XML backupGlobals="true"

--static-backup XML backupStaticAttributes="true"

---

serialize() unserialize()

PDO $GLOBALS

@backupGlobals @backupGlobals

```php
final class MyTest extends TestCase
{
    protected $backupGlobalsExcludeList = ['globalVariable'];

    // ...
}
```

---

setUp() $backupGlobalsBlacklist

@backupStaticAttributes @backupStaticAttributes

@backupStaticAttributes @backupStaticAttributes
—— PHP

/

setUp() tearDown()

```
final class MyTest extends TestCase
{
    protected $backupStaticAttributesExcludeList = [
        'className' => ['attributeName']
    ];

    // ...
}
```

setUp() $backupStaticAttributesExcludeList

CHAPTER 5

PHPUnit

PHPUnit

5.1

PHPUnit

sebastianbergmann/money `tests` `src` SUT System Under Test package class

```
src                                 tests
`-- Currency.php                    `-- CurrencyTest.php
`-- IntlFormatter.php               `-- IntlFormatterTest.php
`-- Money.php                       `-- MoneyTest.php
`-- autoload.php
```

PHPUnit

```
$ phpunit --bootstrap src/autoload.php tests
PHPUnit latest.0 by Sebastian Bergmann and contributors.

.............................

Time: 636 ms, Memory: 3.50Mb

OK (33 tests, 52 assertions)
```

PHPUnit `*Test.php`

`CurrencyTest` `tests/CurrencyTest.php`

```
$ phpunit --bootstrap src/autoload.php tests/CurrencyTest.php
PHPUnit latest.0 by Sebastian Bergmann and contributors.
```

```
........

Time: 280 ms, Memory: 2.75Mb

OK (8 tests, 8 assertions)
```

--filter

```
$ phpunit --bootstrap src/autoload.php --filter testObjectCanBeConstructedForValidConstructorArgumen
↪tests
PHPUnit latest.0 by Sebastian Bergmann and contributors.

..

Time: 167 ms, Memory: 3.00Mb

OK (2 test, 2 assertions)
```

XML

5.2 XML

PHPUnit XML *XML* 5.1 phpunit.xml tests *Test.php *Test

5.1: XML

```
<phpunit bootstrap="src/autoload.php">
  <testsuites>
    <testsuite name="money">
      <directory>tests</directory>
    </testsuite>
  </testsuites>
</phpunit>
```

--testsuite

```
$ phpunit --bootstrap src/autoload.php --testsuite money
PHPUnit latest.0 by Sebastian Bergmann and contributors.

..

Time: 167 ms, Memory: 3.00Mb

OK (2 test, 2 assertions)
```

phpunit.xml phpunit.xml.dist --configuration

5.2: XML

```
<phpunit bootstrap="src/autoload.php">
  <testsuites>
    <testsuite name="money">
      <file>tests/IntlFormatterTest.php</file>
      <file>tests/MoneyTest.php</file>
      <file>tests/CurrencyTest.php</file>
```

```
    </testsuite>
  </testsuites>
</phpunit>
```

CHAPTER 6

PHPUnit

6.1

PHPUnit `--dont-report-useless-tests` PHPUnit `beStrictAboutTestsThatDoNotTestAnything="false"`

6.2

PHPUnit `--strict-coverage` PHPUnit `beStrictAboutCoversAnnotation="true"`

@covers *@covers* *@uses*

PHPUnit `forceCoversAnnotation="true"` *@covers*

6.3

PHPUnit `--disallow-test-output` PHPUnit `beStrictAboutOutputDuringTests="true"`

print

6.4

`PHP_Invoker` `pcntl` `--enforce-time-limit` PHPUnit `enforceTimeLimit="true"`

`@large` 60 `timeoutForLargeTests`

`@medium` 10 `timeoutForMediumTests`

`@small`	1	`timeoutForSmallTests`

`@small` `@medium` `@large`

6.5

PHPUnit `--strict-global-state` PHPUnit `beStrictAboutChangesToGlobalState="true"`

CHAPTER 7

7.1

```
public function testSomething(): void
{
}
```

PHPUnit —— $this->fail()

PHPUnit\Framework\IncompleteTest PHPUnit\Framework\In

7.1 SampleTest testSomething() markTestIncomplete()``
``PHPUnit\Framework\IncompleteTestError

7.1:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class SampleTest extends TestCase
{
    public function testSomething(): void
    {
        //
        $this->assertTrue(true, 'This should already work.');

        //
        $this->markTestIncomplete(
          'This test has not been implemented yet.'
        );
    }
}
```

PHPUnit I

```
$ phpunit --verbose SampleTest
```

```
PHPUnit latest.0 by Sebastian Bergmann and contributors.

I

Time: 0 seconds, Memory: 3.95Mb

There was 1 incomplete test:

1) SampleTest::testSomething
This test has not been implemented yet.

/home/sb/SampleTest.php:12
OK, but incomplete or skipped tests!
Tests: 1, Assertions: 1, Incomplete: 1.
```

7.1 API

7.1: API

void markTestIncomplete()	
void markTestIncomplete(string $message)	$message

7.2

MySQL MySQL

7.2 DatabaseTest testConnection() setUp() MySQLi markTestSkipped()

7.2:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class DatabaseTest extends TestCase
{
    protected function setUp(): void
    {
        if (!extension_loaded('mysqli')) {
            $this->markTestSkipped(
              'The MySQLi extension is not available.'
            );
        }
    }

    public function testConnection(): void
    {
        // ...
    }
}
```

PHPUnit S

```
$ phpunit --verbose DatabaseTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

S

Time: 0 seconds, Memory: 3.95Mb
```

```
There was 1 skipped test:

1) DatabaseTest::testConnection
The MySQLi extension is not available.

/home/sb/DatabaseTest.php:9
OK, but incomplete or skipped tests!
Tests: 1, Assertions: 0, Skipped: 1.
```

7.2 API

7.2: API

`void markTestSkipped()`	
`void markTestSkipped(string $message)`	`$message`

7.3 @requires

`@requires`

7.3: @requires

`PHP`	PHP	@requires PHP 7.1.20	@requires PHP >= 7.2
`PHPUnit`	PHPUnit	@requires PHPUnit 7.3.1	@requires PHPUnit < 8
`OS`	PHP_OS	@requires OS Linux	@requires OS WIN32|WINNT
`OSFAMILY`	OS family	@requires OSFAMILY Solaris	@requires OSFAMILY Windows
`function`	function_exists	@requires function imap_open	@requires function ReflectionMethod::setAccessible
`extension`		@requires extension mysqli	@requires extension redis >= 2.2.0

PHP PHPUnit `< <= > >= = == != <>`

PHP version_compare `=` `==` `X.Y.Z` `X.Y`

7.3: @requires

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

/**
 * @requires extension mysqli
 */
final class DatabaseTest extends TestCase
{
    /**
     * @requires PHP >= 5.3
     */
    public function testConnection(): void
    {
        //    mysqli      PHP >= 5.3
    }

    // ...    mysqli
}
```

PHP *<testsuites>* XML

CHAPTER 8

Gerard Meszaros *Meszaros2007*

Gerard Meszaros

SUT

DOC API

PHPUnit `createStub($type)` `createMock($type)` `getMockBuilder($type)`

`createStub($type)` `createMock($type)` `__construct()` `__clone()`
`getMockBuilder($type)`

`null` `will($this->returnValue())`

final private static

`final` `private` `static` Stub Mock PHPUnit `static`
`\PHPUnit\Framework\MockObject\BadMethodCallException`

8.1 Stubs

stubbing *Stub* “ ”

8.2 `PHPUnit\Framework\TestCase` `createStub()` `SomeClass`
8.1 PHPUnit “ ”

8.1:

```
<?php declare(strict_types=1);
class SomeClass
{
    public function doSomething()
    {
        //
    }
}
```

8.2:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StubTest extends TestCase
{
    public function testStub(): void
    {
        //  SomeClass
        $stub = $this->createStub(SomeClass::class);

        //
        $stub->method('doSomething')
             ->willReturn('foo');

        //    $stub->doSomething()    'foo'
        $this->assertSame('foo', $stub->doSomething());
    }
```

“method”

“method”

“method” `$stub->expects($this->any())->method('doSomething')->willReturn('foo');`

“ ” `createStub()` PHPUnit PHP

`createStub()`

8.3:

```
<?php declare(strict_types=1);
class C
{
    public function m(): D
    {
        //
    }
}
```

`C::m()` D C `willReturn()` `m()` PHPUnit `m()` D

`m` 0 `int` 0.0 `float` `[]` `array`

8.4 `createStub()`

8.4: API

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StubTest extends TestCase
{
    public function testStub(): void
    {
        //  SomeClass
        $stub = $this->getMockBuilder(SomeClass::class)
                     ->disableOriginalConstructor()
                     ->disableOriginalClone()
                     ->disableArgumentCloning()
```

```
                    ->disallowMockingUnknownTypes()
                    ->getMock();

        //
        $stub->method('doSomething')
             ->willReturn('foo');

        //    $stub->doSomething()    'foo'
        $this->assertSame('foo', $stub->doSomething());
    }
```

willReturn($value) will($this->returnValue($value))

8.5 returnArgument() returnValue()

8.5:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StubTest extends TestCase
{
    public function testReturnArgumentStub(): void
    {
        //  SomeClass
        $stub = $this->createStub(SomeClass::class);

        //
        $stub->method('doSomething')
             ->will($this->returnArgument(0));

        // $stub->doSomething('foo')    'foo'
        $this->assertSame('foo', $stub->doSomething('foo'));

        // $stub->doSomething('bar')    'bar'
        $this->assertSame('bar', $stub->doSomething('bar'));
    }
}
```

8.6 returnSelf()

8.6:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StubTest extends TestCase
{
    public function testReturnSelf(): void
    {
        //  SomeClass
        $stub = $this->createStub(SomeClass::class);

        //
        $stub->method('doSomething')
             ->will($this->returnSelf());

        // $stub->doSomething()    $stub
        $this->assertSame($stub, $stub->doSomething());
    }
}
```

returnValueMap() 8.7

8.7:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StubTest extends TestCase
{
    public function testReturnValueMapStub(): void
    {
        //  SomeClass
        $stub = $this->createStub(SomeClass::class);

        // Create a map of arguments to return values.
        $map = [
            ['a', 'b', 'c', 'd'],
            ['e', 'f', 'g', 'h']
        ];

        //
        $stub->method('doSomething')
             ->will($this->returnValueMap($map));

        // $stub->doSomething()
        $this->assertSame('d', $stub->doSomething('a', 'b', 'c'));
        $this->assertSame('h', $stub->doSomething('e', 'f', 'g'));
    }
}
```

returnValue() returnArgument() returnCallback()
8.8

8.8:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StubTest extends TestCase
{
    public function testReturnCallbackStub(): void
    {
        //  SomeClass
        $stub = $this->createStub(SomeClass::class);

        //
        $stub->method('doSomething')
             ->will($this->returnCallback('str_rot13'));

        // $stub->doSomething($argument)   str_rot13($argument)
        $this->assertSame('fbzrguvat', $stub->doSomething('something'));
    }
}
```

onConsecutiveCalls() 8.9

8.9:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StubTest extends TestCase
{
    public function testOnConsecutiveCallsStub(): void
    {
```

```
        //  SomeClass
        $stub = $this->createStub(SomeClass::class);

        //
        $stub->method('doSomething')
             ->will($this->onConsecutiveCalls(2, 3, 5, 7));

        // $stub->doSomething()
        $this->assertSame(2, $stub->doSomething());
        $this->assertSame(3, $stub->doSomething());
        $this->assertSame(5, $stub->doSomething());
    }
}
```

8.10 `throwException()`

8.10:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StubTest extends TestCase
{
    public function testThrowExceptionStub(): void
    {
        //  SomeClass
        $stub = $this->createStub(SomeClass::class);

        //
        $stub->method('doSomething')
             ->will($this->throwException(new Exception));

        // $stub->doSomething()
        $stub->doSomething();
    }
}
```

facade `Database` `IDatabase` `IDatabase`

8.2 Mock Object

mocking

mock object “ ” Gerard Meszaros

PHPUnit `@depends` PHPUnit

`update()` 8.11 SUT `Subject` `Observer`

8.11: SUT Subject Observer

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;
```

```
class Subject
{
    protected $observers = [];
    protected $name;

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

    public function getName()
    {
        return $this->name;
    }

    public function attach(Observer $observer)
    {
        $this->observers[] = $observer;
    }

    public function doSomething()
    {
        //
        // ...

        //
        $this->notify('something');
    }

    public function doSomethingBad()
    {
        foreach ($this->observers as $observer) {
            $observer->reportError(42, 'Something bad happened', $this);
        }
    }

    protected function notify($argument)
    {
        foreach ($this->observers as $observer) {
            $observer->update($argument);
        }
    }

    //
}

class Observer
{
    public function update($argument)
    {
        //
    }

    public function reportError($errorCode, $errorMessage, Subject $subject)
    {
        //
    }

    //
}
```

8.12 Subject Observer

PHPUnit\Framework\TestCase createMock() Observer

expects() with()

8.12:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class SubjectTest extends TestCase
{
    public function testObserversAreUpdated(): void
    {
        //  Observer
        //   update()
        $observer = $this->createMock(Observer::class);

        //  update()
        //     'something'
        $observer->expects($this->once())
                 ->method('update')
                 ->with($this->equalTo('something'));

        //   Subject      Observer
        $subject = new Subject('My subject');
        $subject->attach($observer);

        //  $subject    doSomething()
        //       'something'     Observer
        //   update()
        $subject->doSomething();
    }
}
```

with()

8.13:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class SubjectTest extends TestCase
{
    public function testErrorReported(): void
    {
        //  Observer      reportError()
        $observer = $this->createMock(Observer::class);

        $observer->expects($this->once())
                 ->method('reportError')
                 ->with(
                       $this->greaterThan(0),
                       $this->stringContains('Something'),
                       $this->anything()
                   );

        $subject = new Subject('My subject');
        $subject->attach($observer);

        // doSomethingBad()
        // reportError()    observer
        $subject->doSomethingBad();
    }
}
```

withConsecutive() with()

8.14:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class FooTest extends TestCase
{
    public function testFunctionCalledTwoTimesWithSpecificArguments(): void
    {
        $mock = $this->getMockBuilder(stdClass::class)
                     ->setMethods(['set'])
                     ->getMock();

        $mock->expects($this->exactly(2))
             ->method('set')
             ->withConsecutive(
                 [$this->equalTo('foo'), $this->greaterThan(0)],
                 [$this->equalTo('bar'), $this->greaterThan(0)]
             );

        $mock->set('foo', 21);
        $mock->set('bar', 48);
    }
}
```

callback() PHP callback PHP true false

8.15:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class SubjectTest extends TestCase
{
    public function testErrorReported(): void
    {
        //  Observer      reportError()
        $observer = $this->createMock(Observer::class);

        $observer->expects($this->once())
                 ->method('reportError')
                 ->with(
                     $this->greaterThan(0),
                     $this->stringContains('Something'),
                     $this->callback(function($subject)
                     {
                         return is_callable([$subject, 'getName']) &&
                                $subject->getName() == 'My subject';
                     }
                 ));

        $subject = new Subject('My subject');
        $subject->attach($observer);

        // doSomethingBad()
        // reportError()    observer
        $subject->doSomethingBad();
    }
}
```

8.16:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class FooTest extends TestCase
{
    public function testIdenticalObjectPassed(): void
    {
        $expectedObject = new stdClass;

        $mock = $this->getMockBuilder(stdClass::class)
                     ->setMethods(['foo'])
                     ->getMock();

        $mock->expects($this->once())
             ->method('foo')
             ->with($this->identicalTo($expectedObject));

        $mock->foo($expectedObject);
    }
}
```

8.17:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class FooTest extends TestCase
{
    public function testIdenticalObjectPassed(): void
    {
        $cloneArguments = true;

        $mock = $this->getMockBuilder(stdClass::class)
                     ->enableArgumentCloning()
                     ->getMock();

        //        identicalTo
    }
}
```

8.1

8.1:

PHPUnit\Framework\MockObject\Matcher\AnyInvokedCount any()	0
PHPUnit\Framework\MockObject\Matcher\InvokedCount never()	
PHPUnit\Framework\MockObject\Matcher\InvokedAtLeastOnce atLeastOnce()	
PHPUnit\Framework\MockObject\Matcher\InvokedCount once()	
PHPUnit\Framework\MockObject\Matcher\InvokedCount exactly(int $count)	$count
PHPUnit\Framework\MockObject\Matcher\InvokedAtIndex at(int $index)	$index

at() $index

createStub() createMock() getMockBuilder($type)

- setMethods(array $methods) setMethods(null)
- setMethodsExcept(array $methods) public setMethods()

- setConstructorArgs(array $args)
- setMockClassName($name)
- disableOriginalConstructor()
- disableOriginalClone()
- disableAutoload() __autoload()

8.3 Trait

getMockForTrait() trait

8.18:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

trait AbstractTrait
{
    public function concreteMethod()
    {
        return $this->abstractMethod();
    }

    public abstract function abstractMethod();
}

final class TraitClassTest extends TestCase
{
    public function testConcreteMethod(): void
    {
        $mock = $this->getMockForTrait(AbstractTrait::class);

        $mock->expects($this->any())
             ->method('abstractMethod')
             ->will($this->returnValue(true));

        $this->assertTrue($mock->concreteMethod());
    }
}
```

getMockForAbstractClass()

8.19:

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

abstract class AbstractClass
{
    public function concreteMethod()
    {
        return $this->abstractMethod();
    }
```

```
    public abstract function abstractMethod();
}

final class AbstractClassTest extends TestCase
{
    public function testConcreteMethod(): void
    {
        $stub = $this->getMockForAbstractClass(AbstractClass::class);

        $stub->expects($this->any())
             ->method('abstractMethod')
             ->will($this->returnValue(true));

        $this->assertTrue($stub->concreteMethod());
    }
}
```

8.4 Web Web Services

web web web getMock() getMockFromWsdl() getMockFromWsdl() WSDL web getMock() PHP

8.20 getMockFromWsdl() GoogleSearch.wsdl web

8.20: web

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class GoogleTest extends TestCase
{
    public function testSearch(): void
    {
        $googleSearch = $this->getMockFromWsdl(
          'GoogleSearch.wsdl', 'GoogleSearch'
        );

        $directoryCategory = new stdClass;
        $directoryCategory->fullViewableName = '';
        $directoryCategory->specialEncoding = '';

        $element = new stdClass;
        $element->summary = '';
        $element->URL = 'https://phpunit.de/';
        $element->snippet = '...';
        $element->title = '<b>PHPUnit</b>';
        $element->cachedSize = '11k';
        $element->relatedInformationPresent = true;
        $element->hostName = 'phpunit.de';
        $element->directoryCategory = $directoryCategory;
        $element->directoryTitle = '';

        $result = new stdClass;
        $result->documentFiltering = false;
        $result->searchComments = '';
        $result->estimatedTotalResultsCount = 3.9000;
        $result->estimateIsExact = false;
        $result->resultElements = [$element];
        $result->searchQuery = 'PHPUnit';
        $result->startIndex = 1;
```

```
        $result->endIndex = 1;
        $result->searchTips = '';
        $result->directoryCategories = [];
        $result->searchTime = 0.248822;

        $googleSearch->expects($this->any())
                     ->method('doGoogleSearch')
                     ->will($this->returnValue($result));

        /**
         * $googleSearch->doGoogleSearch()
         *     web    doGoogleSearch()
         */
        $this->assertEquals(
          $result,
          $googleSearch->doGoogleSearch(
            '00000000000000000000000000000000',
            'PHPUnit',
            0,
            1,
            false,
            '',
            false,
            '',
            '',
            ''
          )
        );
    }
}
```

CHAPTER 9

Wikipedia

bug

PHPUnit php-code-coverage PHP Xdebug PCOV PHPDBG

PHP `php` Xdebug PCOV

PHPUnit HTML Clover Crap4J PHPUnit XML STDOUT PHP

<logging>

9.1

Line Coverage

Line Coverage

Branch Coverage

Branch Coverage `true` `false`

Path Coverage

Path Coverage

Function and Method Coverage

Function and Method Coverage php-code-coverage

Class and Trait Coverage

Class and Trait Coverage php-code-coverage

CRAP *Change Risk Anti-Patterns (CRAP) Index*

CRAP *Change Risk Anti-Patterns (CRAP) Index* cyclomatic complexity CRAP CRAP

9.2

PHPUnit --coverage-filter *The <include> Element*

includeUncoveredFilesInCodeCoverageReport processUncoveredFilesForCodeCoverageReport

- includeUncoveredFilesInCodeCoverageReport="false"
- includeUncoveredFilesInCodeCoverageReport="true"
- processUncoveredFilesForCodeCoverageReport="false" includeUncoveredFilesInCodeCoverageReport="true" PHPUnit
- processUncoveredFilesForCodeCoverageReport="true" PHPUnit

---

processUncoveredFilesForCodeCoverageReport="true"

9.3

PHPUnit @codeCoverageIgnore @codeCoverageIgnoreStart @codeCoverageIgnoreEnd 9.1

9.1: @codeCoverageIgnore @codeCoverageIgnoreStart @codeCoverageIgnoreEnd

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

/**
 * @codeCoverageIgnore
 */
final class Foo
{
    public function bar(): void
    {
    }
}

final class Bar
{
    /**
     * @codeCoverageIgnore
     */
    public function foo(): void
    {
    }
}

if (false) {
    // @codeCoverageIgnoreStart
    print '*';
    // @codeCoverageIgnoreEnd
}
```

```
exit; // @codeCoverageIgnore
```

9.4

@covers *annotation documentation* 9.2

@covers @covers

9.2:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

/**
 * @covers \Invoice
 * @uses \Money
 */
final class InvoiceTest extends TestCase
{
    private $invoice;

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

    public function testAmountInitiallyIsEmpty(): void
    {
        $this->assertEquals(new Money, $this->invoice->getAmount());
    }
}
```

9.3:

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class BankAccountTest extends TestCase
{
    private $ba;

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

    /**
     * @covers \BankAccount::getBalance
     */
    public function testBalanceIsInitiallyZero(): void
    {
        $this->assertSame(0, $this->ba->getBalance());
    }
```

```
    /**
     * @covers \BankAccount::withdrawMoney
     */
    public function testBalanceCannotBecomeNegative(): void
    {
        try {
            $this->ba->withdrawMoney(1);
        }

        catch (BankAccountException $e) {
            $this->assertSame(0, $this->ba->getBalance());

            return;
        }

        $this->fail();
    }

    /**
     * @covers \BankAccount::depositMoney
     */
    public function testBalanceCannotBecomeNegative2(): void
    {
        try {
            $this->ba->depositMoney(-1);
        }

        catch (BankAccountException $e) {
            $this->assertSame(0, $this->ba->getBalance());

            return;
        }

        $this->fail();
    }

    /**
     * @covers \BankAccount::getBalance
     * @covers \BankAccount::depositMoney
     * @covers \BankAccount::withdrawMoney
     */
    public function testDepositWithdrawMoney(): void
    {
        $this->assertSame(0, $this->ba->getBalance());
        $this->ba->depositMoney(1);
        $this->assertSame(1, $this->ba->getBalance());
        $this->ba->withdrawMoney(1);
        $this->assertSame(0, $this->ba->getBalance());
    }
}
```

`@coversNothing` *@coversNothing*

9.4:

```
<?php declare(strict_types=1);
use PHPUnit\DbUnit\TestCase

final class GuestbookIntegrationTest extends TestCase
{
    /**
     * @coversNothing
     */
```

```
    public function testAddEntry(): void
    {
        $guestbook = new Guestbook();
        $guestbook->addEntry("suzy", "Hello world!");

        $queryTable = $this->getConnection()->createQueryTable(
            'guestbook', 'SELECT * FROM guestbook'
        );

        $expectedTable = $this->createFlatXmlDataSet("expectedBook.xml")
                              ->getTable("guestbook");

        $this->assertTablesEqual($expectedTable, $queryTable);
    }
}
```

9.5

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

//   " "
//
if (false) this_function_call_shows_up_as_covered();

//
//
if (false)
    //
    //      if
    will_also_show_up_as_covered();

//
if (false) {
    this_call_will_never_show_up_as_covered();
}
```

CHAPTER 10

PHPUnit

PHPUnit PHPUnit

10.1 PHPUnit\Framework\TestCase

`PHPUnit\Framework\TestCase` PHPUnit

10.2

PHPUnit 10.1 `assertTrue()` `isTrue()` `assertThat()` `isTrue()` `assertThat()`

10.1: PHPUnit\Framework\Assert assertTrue() isTrue()

```php
<?php declare(strict_types=1);
namespace PHPUnit\Framework;

use PHPUnit\Framework\Constraint\IsTrue;

abstract class Assert
{
    // ...

    public static function assertTrue($condition, string $message = ''): void
    {
        static::assertThat($condition, static::isTrue(), $message);
    }

    // ...

    public static function isTrue(): IsTrue
    {
        return new IsTrue;
    }
```

```
    // ...
}
```

10.2 PHPUnit\Framework\Constraint\IsTrue PHPUnit\Framework\Constraint

10.2: PHPUnit\FrameworkConstraint\IsTrue

```
<?php declare(strict_types=1);
namespace PHPUnit\Framework\Constraint;

use PHPUnit\Framework\Constraint;

final class IsTrue extends Constraint
{
    public function toString(): string
    {
        return 'is true';
    }

    protected function matches($other): bool
    {
        return $other === true;
    }
}
```

assertTrue() isTrue() PHPUnit\Framework\Constraint\IsTrue assertThat()
isTrue()

10.3

PHPUnit

- AfterIncompleteTestHook
- AfterLastTestHook
- AfterRiskyTestHook
- AfterSkippedTestHook
- AfterSuccessfulTestHook
- AfterTestErrorHook
- AfterTestFailureHook
- AfterTestWarningHook
- AfterTestHook
- BeforeFirstTestHook
- BeforeTestHook

“hook”

PHPUnit XML *<extensions>*

10.3 BeforeFirstTestHook AfterLastTestHook

10.3:

```
<?php declare(strict_types=1);
namespace Vendor;

use PHPUnit\Runner\BeforeFirstTestHook;
```

```php
use PHPUnit\Runner\AfterLastTestHook;

final class MyExtension implements BeforeFirstTestHook, AfterLastTestHook
{
    public function executeBeforeFirstTest(): void
    {
        //
    }

    public function executeAfterLastTest(): void
    {
        //
    }
}
```

10.3.1

PHPUnit

10.4 __constructor()

10.4:

```php
<?php declare(strict_types=1);
namespace Vendor;

use PHPUnit\Runner\BeforeFirstTestHook;
use PHPUnit\Runner\AfterLastTestHook;

final class MyConfigurableExtension implements BeforeFirstTestHook, AfterLastTestHook
{
    protected $config_value_1 = '';

    protected $config_value_2 = 0;

    public function __construct(string $value1 = '', int $value2 = 0)
    {
        $this->config_value_1 = $config_1;
        $this->config_value_2 = $config_2;
    }

    public function executeBeforeFirstTest(): void
    {
        if (strlen($this->config_value_1) {
            echo 'Testing with configuration value: ' . $this->config_value_1;
        }
    }

    public function executeAfterLastTest(): void
    {
        if ($this->config_value_2 > 10) {
            echo 'Second config value is OK!';
        }
    }
}
```

XML XML extensions 10.5

10.5:

```
<extensions>
    <extension class="Vendor\MyUnconfigurableExtension" />
    <extension class="Vendor\MyConfigurableExtension">
        <arguments>
            <string>Hello world!</string>
            <int>15</int>
        </arguments>
    </extension>
</extensions>
```

arguments *<arguments>*

CHAPTER 11

11.1 vs.

PHPUnit PHPUnit\Framework\Assert PHPUnit\Framework\TestCase PHPUnit\Framework\Assert

static PHPUnit\Framework\Assert::assertTrue() $this->assertTrue() self::assertTrue() PHPUnit\Framework\TestCase assertTrue()

—— PHPUnit —— $this->assertTrue() self::assertTrue() “ ”

$this->assertTrue() “ ” static assertTrue() $this->assertTrue() self::assertTrue()

11.2 assertArrayHasKey()

assertArrayHasKey(mixed $key, array $array[, string $message = ''])

$array $key $message

assertArrayNotHasKey()

11.1: assertArrayHasKey()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ArrayHasKeyTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertArrayHasKey('foo', ['bar' => 'baz']);
    }
}
```

```
$ phpunit ArrayHasKeyTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ArrayHasKeyTest::testFailure
Failed asserting that an array has the key 'foo'.

/home/sb/ArrayHasKeyTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.3 assertClassHasAttribute()

assertClassHasAttribute(string $attributeName, string $className[, string $message = ''])

$className::attributeName $message

assertClassNotHasAttribute()

11.2: assertClassHasAttribute()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ClassHasAttributeTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertClassHasAttribute('foo', stdClass::class);
    }
}
```

```
$ phpunit ClassHasAttributeTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) ClassHasAttributeTest::testFailure
Failed asserting that class "stdClass" has attribute "foo".

/home/sb/ClassHasAttributeTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.4 assertClassHasStaticAttribute()

```
assertClassHasStaticAttribute(string $attributeName, string $className[, string
$message = ''])
```

 $className::attributeName $message

assertClassNotHasStaticAttribute()

11.3: assertClassHasStaticAttribute()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ClassHasStaticAttributeTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertClassHasStaticAttribute('foo', stdClass::class);
    }
}
```

```
$ phpunit ClassHasStaticAttributeTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) ClassHasStaticAttributeTest::testFailure
Failed asserting that class "stdClass" has static attribute "foo".

/home/sb/ClassHasStaticAttributeTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.5 assertContains()

```
assertContains(mixed $needle, iterable $haystack[, string $message = ''])
```

 $needle $haystack $message

assertNotContains()

11.4: assertContains()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ContainsTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertContains(4, [1, 2, 3]);
    }
}
```

```
$ phpunit ContainsTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ContainsTest::testFailure
Failed asserting that an array contains 4.

/home/sb/ContainsTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.6 assertStringContainsString()

```
assertStringContainsString(string $needle, string $haystack[, string $message = ''])
```

 $needle $haystack $message

assertStringNotContainsString()

11.5: assertStringContainsString()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StringContainsStringTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertStringContainsString('foo', 'bar');
    }
}
```

```
$ phpunit StringContainsStringTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F                                                                   1 / 1 (100%)

Time: 37 ms, Memory: 6.00 MB

There was 1 failure:

1) StringContainsStringTest::testFailure
Failed asserting that 'bar' contains "foo".

/home/sb/StringContainsStringTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.7 assertStringContainsStringIgnoringCase()

```
assertStringContainsStringIgnoringCase(string $needle, string $haystack[, string
$message = ''])
```

$needle $haystack $message

$haystack $needle

assertStringNotContainsStringIgnoringCase()

11.6: assertStringContainsStringIgnoringCase()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StringContainsStringIgnoringCaseTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertStringContainsStringIgnoringCase('foo', 'bar');
    }
}
```

```
$ phpunit StringContainsStringIgnoringCaseTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F                                                                   1 / 1 (100%)

Time: 40 ms, Memory: 6.00 MB

There was 1 failure:

1) StringContainsStringTest::testFailure
Failed asserting that 'bar' contains "foo".

/home/sb/StringContainsStringIgnoringCaseTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.8 assertContainsOnly()

```
assertContainsOnly(string $type, iterable $haystack[, boolean $isNativeType = null,
string $message = ''])
```

$haystack $type $message

$isNativeType $type PHP

assertNotContainsOnly()

11.7: assertContainsOnly()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ContainsOnlyTest extends TestCase
{
    public function testFailure(): void
    {
```

```
        $this->assertContainsOnly('string', ['1', '2', 3]);
    }
}
```

```
$ phpunit ContainsOnlyTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ContainsOnlyTest::testFailure
Failed asserting that Array (
    0 => '1'
    1 => '2'
    2 => 3
) contains only values of type "string".

/home/sb/ContainsOnlyTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.9 assertContainsOnlyInstancesOf()

```
assertContainsOnlyInstancesOf(string $classname, Traversable|array $haystack[, string
$message = ''])
```

$haystack	$classname	$message

11.8: assertContainsOnlyInstancesOf()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ContainsOnlyInstancesOfTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertContainsOnlyInstancesOf(
            Foo::class,
            [new Foo, new Bar, new Foo]
        );
    }
}
```

```
$ phpunit ContainsOnlyInstancesOfTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ContainsOnlyInstancesOfTest::testFailure
```

```
Failed asserting that Array ([0]=> Bar Object(...)) is an instance of class "Foo".

/home/sb/ContainsOnlyInstancesOfTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.10 assertCount()

```
assertCount($expectedCount, $haystack[, string $message = ''])
```

 $haystack $expectedCount $message

assertNotCount()

11.9: assertCount()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class CountTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertCount(0, ['foo']);
    }
}
```

```
$ phpunit CountTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) CountTest::testFailure
Failed asserting that actual size 1 matches expected size 0.

/home/sb/CountTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.11 assertDirectoryExists()

```
assertDirectoryExists(string $directory[, string $message = ''])
```

 $directory $message

assertDirectoryDoesNotExist()

11.10: assertDirectoryExists()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class DirectoryExistsTest extends TestCase
```

```
{
    public function testFailure(): void
    {
        $this->assertDirectoryExists('/path/to/directory');
    }
}
```

```
$ phpunit DirectoryExistsTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) DirectoryExistsTest::testFailure
Failed asserting that directory "/path/to/directory" exists.

/home/sb/DirectoryExistsTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.12 assertDirectoryIsReadable()

```
assertDirectoryIsReadable(string $directory[, string $message = ''])
```

 $directory $message

```
assertDirectoryIsNotReadable()
```

11.11: assertDirectoryIsReadable()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class DirectoryIsReadableTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertDirectoryIsReadable('/path/to/directory');
    }
}
```

```
$ phpunit DirectoryIsReadableTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) DirectoryIsReadableTest::testFailure
Failed asserting that "/path/to/directory" is readable.

/home/sb/DirectoryIsReadableTest.php:6
```

```
FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.13 assertDirectoryIsWritable()

```
assertDirectoryIsWritable(string $directory[, string $message = ''])
```

 $directory $message

```
assertDirectoryIsNotWritable()
```

11.12: assertDirectoryIsWritable()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class DirectoryIsWritableTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertDirectoryIsWritable('/path/to/directory');
    }
}
```

```
$ phpunit DirectoryIsWritableTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) DirectoryIsWritableTest::testFailure
Failed asserting that "/path/to/directory" is writable.

/home/sb/DirectoryIsWritableTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.14 assertEmpty()

```
assertEmpty(mixed $actual[, string $message = ''])
```

 $actual $message

```
assertNotEmpty()
```

11.13: assertEmpty()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class EmptyTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertEmpty(['foo']);
```

```
    }
}
```

```
$ phpunit EmptyTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) EmptyTest::testFailure
Failed asserting that an array is empty.

/home/sb/EmptyTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.15 assertEquals()

```
assertEquals(mixed $expected, mixed $actual[, string $message = ''])
```

`$expected` `$actual` `$message`

`assertNotEquals()`

11.14: assertEquals()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class EqualsTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertEquals(1, 0);
    }

    public function testFailure2(): void
    {
        $this->assertEquals('bar', 'baz');
    }

    public function testFailure3(): void
    {
        $this->assertEquals("foo\nbar\nbaz\n", "foo\nbah\nbaz\n");
    }
}
```

```
$ phpunit EqualsTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

FFF

Time: 0 seconds, Memory: 5.25Mb

There were 3 failures:
```

```
1) EqualsTest::testFailure
Failed asserting that 0 matches expected 1.

/home/sb/EqualsTest.php:6

2) EqualsTest::testFailure2
Failed asserting that two strings are equal.
--- Expected
+++ Actual
@@ @@
-'bar'
+'baz'

/home/sb/EqualsTest.php:11

3) EqualsTest::testFailure3
Failed asserting that two strings are equal.
--- Expected
+++ Actual
@@ @@
 'foo
-bar
+bah
 baz
 '

/home/sb/EqualsTest.php:16

FAILURES!
Tests: 3, Assertions: 3, Failures: 3.
```

 $expected $actual

```
assertEquals(DOMDocument $expected, DOMDocument $actual[, string $message = ''])
```

 $expected $actual DOMDocument XML $message

11.15: DOMDocument assertEquals()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class EqualsTest extends TestCase
{
    public function testFailure(): void
    {
        $expected = new DOMDocument;
        $expected->loadXML('<foo><bar/></foo>');

        $actual = new DOMDocument;
        $actual->loadXML('<bar><foo/></bar>');

        $this->assertEquals($expected, $actual);
    }
}
```

```
$ phpunit EqualsTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F
```

```
Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) EqualsTest::testFailure
Failed asserting that two DOM documents are equal.
--- Expected
+++ Actual
@@ @@
 <?xml version="1.0"?>
-<foo>
-  <bar/>
-</foo>
+<bar>
+  <foo/>
+</bar>

/home/sb/EqualsTest.php:12

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

```
assertEquals(object $expected, object $actual[, string $message = ''])
```

$expected	$actual	$message

11.16: assertEquals()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class EqualsTest extends TestCase
{
    public function testFailure(): void
    {
        $expected = new stdClass;
        $expected->foo = 'foo';
        $expected->bar = 'bar';

        $actual = new stdClass;
        $actual->foo = 'bar';
        $actual->baz = 'bar';

        $this->assertEquals($expected, $actual);
    }
}
```

```
$ phpunit EqualsTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) EqualsTest::testFailure
Failed asserting that two objects are equal.
--- Expected
+++ Actual
```

```
@@ @@
 stdClass Object (
-    'foo' => 'foo'
-    'bar' => 'bar'
+    'foo' => 'bar'
+    'baz' => 'bar'
 )

/home/sb/EqualsTest.php:14

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

```
assertEquals(array $expected, array $actual[, string $message = ''])
```

$expected	$actual	$message

11.17: assertEquals()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class EqualsTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertEquals(['a', 'b', 'c'], ['a', 'c', 'd']);
    }
}
```

```
$ phpunit EqualsTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) EqualsTest::testFailure
Failed asserting that two arrays are equal.
--- Expected
+++ Actual
@@ @@
 Array (
     0 => 'a'
-    1 => 'b'
-    2 => 'c'
+    1 => 'c'
+    2 => 'd'
 )

/home/sb/EqualsTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.16 assertEqualsCanonicalizing()

```
assertEqualsCanonicalizing(mixed $expected, mixed $actual[, string $message = ''])
```

$expected $actual $message

$expected $actual $expected $actual $expected $actual

assertNotEqualsCanonicalizing()

11.18: assertEqualsCanonicalizing()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class EqualsCanonicalizingTest extends TestCase
{
    public function testFailure()
    {
        $this->assertEqualsCanonicalizing([3, 2, 1], [2, 3, 0, 1]);
    }
}
```

```
$ phpunit EqualsCanonicalizingTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F                                                                   1 / 1 (100%)

Time: 42 ms, Memory: 6.00 MB

There was 1 failure:

1) EqualsCanonicalizingTest::testFailure
Failed asserting that two arrays are equal.
--- Expected
+++ Actual
@@ @@
 Array (
-    0 => 1
-    1 => 2
-    2 => 3
+    0 => 0
+    1 => 1
+    2 => 2
+    3 => 3
 )

/home/sb/EqualsCanonicalizingTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.17 assertEqualsIgnoringCase()

```
assertEqualsIgnoringCase(mixed $expected, mixed $actual[, string $message = ''])
```

$expected $actual $message

$expected $actual

assertNotEqualsIgnoringCase()

11.19: assertEqualsIgnoringCase()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class EqualsIgnoringCaseTest extends TestCase
{
    public function testFailure()
    {
        $this->assertEqualsIgnoringCase('foo', 'BAR');
    }
}
```

```
$ phpunit EqualsIgnoringCaseTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F                                                                   1 / 1 (100%)

Time: 51 ms, Memory: 6.00 MB

There was 1 failure:

1) EqualsIgnoringCaseTest::testFailure
Failed asserting that two strings are equal.
--- Expected
+++ Actual
@@ @@
-'foo'
+'BAR'

/home/sb/EqualsIgnoringCaseTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.18 assertEqualsWithDelta()

assertEqualsWithDelta(mixed $expected, mixed $actual, float $delta[, string $message = ''])

 $expected $actual $delta $message

What Every Computer Scientist Should Know About Floating-Point Arithmetic $delta

assertNotEqualsWithDelta()

11.20: assertEqualsWithDelta()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class EqualsWithDeltaTest extends TestCase
{
    public function testFailure()
    {
        $this->assertEqualsWithDelta(1.0, 1.5, 0.1);
    }
}
```

```
$ phpunit EqualsWithDeltaTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F                                                                   1 / 1 (100%)

Time: 41 ms, Memory: 6.00 MB

There was 1 failure:

1) EqualsWithDeltaTest::testFailure
Failed asserting that 1.5 matches expected 1.0.

/home/sb/EqualsWithDeltaTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.19 assertObjectEquals()

assertObjectEquals(object $expected, object $actual, string $method = 'equals', string $message = ''])

$actual->$method($expected) $actual $expected $message

assertEquals() assertNotEquals()

equals(self $other): bool assertObjectEquals()

11.21: assertObjectEquals()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class SomethingThatUsesEmailTest extends TestCase
{
    public function testSomething(): void
    {
        $a = new Email('user@example.org');
        $b = new Email('user@example.org');
        $c = new Email('user@example.com');

        // This passes
        $this->assertObjectEquals($a, $b);

        // This fails
        $this->assertObjectEquals($a, $c);
    }
}
```

11.22: equals() Email

```php
<?php declare(strict_types=1);
final class Email
{
    private string $email;

    public function __construct(string $email)
    {
        $this->ensureIsValidEmail($email);
```

```
        $this->email = $email;
    }

    public function asString(): string
    {
        return $this->email;
    }

    public function equals(self $other): bool
    {
        return $this->asString() === $other->asString();
    }

    private function ensureIsValidEmail(string $email): void
    {
        // ...
    }
}
```

```
$ phpunit EqualsTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F                                                                   1 / 1 (100%)

Time: 00:00.017, Memory: 4.00 MB

There was 1 failure:

1) SomethingThatUsesEmailTest::testSomething
Failed asserting that two objects are equal.
The objects are not equal according to Email::equals().

/home/sb/SomethingThatUsesEmailTest.php:16

FAILURES!
Tests: 1, Assertions: 2, Failures: 1.
```

- $actual $method
-
-
- $expected
- bool

  $actual->$method($expected) false

11.20 assertFalse()

```
assertFalse(bool $condition[, string $message = ''])
```

$condition true $message

assertNotFalse()

11.23: assertFalse()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class FalseTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertFalse(true);
    }
}
```

```
$ phpunit FalseTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) FalseTest::testFailure
Failed asserting that true is false.

/home/sb/FalseTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.21 assertFileEquals()

```
assertFileEquals(string $expected, string $actual[, string $message = ''])
```

 $expected $actual $message

assertFileNotEquals()

11.24: assertFileEquals()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class FileEqualsTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertFileEquals('/home/sb/expected', '/home/sb/actual');
    }
}
```

```
$ phpunit FileEqualsTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:
```

```
1) FileEqualsTest::testFailure
Failed asserting that two strings are equal.
--- Expected
+++ Actual
@@ @@
-'expected
+'actual
 '

/home/sb/FileEqualsTest.php:6

FAILURES!
Tests: 1, Assertions: 3, Failures: 1.
```

11.22 assertFileExists()

```
assertFileExists(string $filename[, string $message = ''])
```

$filename	$message

assertFileDoesNotExist()

11.25: assertFileExists()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class FileExistsTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertFileExists('/path/to/file');
    }
}
```

```
$ phpunit FileExistsTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) FileExistsTest::testFailure
Failed asserting that file "/path/to/file" exists.

/home/sb/FileExistsTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.23 assertFileIsReadable()

```
assertFileIsReadable(string $filename[, string $message = ''])
```

$filename	$message

assertFileIsNotReadable()

11.26: assertFileIsReadable()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class FileIsReadableTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertFileIsReadable('/path/to/file');
    }
}
```

```
$ phpunit FileIsReadableTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) FileIsReadableTest::testFailure
Failed asserting that "/path/to/file" is readable.

/home/sb/FileIsReadableTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.24 assertFileIsWritable()

assertFileIsWritable(string $filename[, string $message = ''])

$filename	$message

assertFileIsNotWritable()

11.27: assertFileIsWritable()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class FileIsWritableTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertFileIsWritable('/path/to/file');
    }
}
```

```
$ phpunit FileIsWritableTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb
```

```
There was 1 failure:

1) FileIsWritableTest::testFailure
Failed asserting that "/path/to/file" is writable.

/home/sb/FileIsWritableTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.25 assertGreaterThan()

```
assertGreaterThan(mixed $expected, mixed $actual[, string $message = ''])
```

$actual	$expected	$message

11.28: assertGreaterThan()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class GreaterThanTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertGreaterThan(2, 1);
    }
}
```

```
$ phpunit GreaterThanTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) GreaterThanTest::testFailure
Failed asserting that 1 is greater than 2.

/home/sb/GreaterThanTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.26 assertGreaterThanOrEqual()

```
assertGreaterThanOrEqual(mixed $expected, mixed $actual[, string $message = ''])
```

$actual	$expected	$message

11.29: assertGreaterThanOrEqual()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class GreatThanOrEqualTest extends TestCase
```

```
{
    public function testFailure(): void
    {
        $this->assertGreaterThanOrEqual(2, 1);
    }
}
```

```
$ phpunit GreaterThanOrEqualTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) GreatThanOrEqualTest::testFailure
Failed asserting that 1 is equal to 2 or is greater than 2.

/home/sb/GreaterThanOrEqualTest.php:6

FAILURES!
Tests: 1, Assertions: 2, Failures: 1.
```

11.27 assertInfinite()

```
assertInfinite(mixed $variable[, string $message = ''])
```

$actual	INF	$message

assertFinite()

11.30: assertInfinite()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class InfiniteTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertInfinite(1);
    }
}
```

```
$ phpunit InfiniteTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) InfiniteTest::testFailure
Failed asserting that 1 is infinite.

/home/sb/InfiniteTest.php:6
```

```
FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.28 assertInstanceOf()

```
assertInstanceOf($expected, $actual[, $message = ''])
```

 $actual $expected $message

```
assertNotInstanceOf()
```

11.31: assertInstanceOf()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class InstanceOfTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertInstanceOf(RuntimeException::class, new Exception);
    }
}
```

```
$ phpunit InstanceOfTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) InstanceOfTest::testFailure
Failed asserting that Exception Object (...) is an instance of class␣
↪"RuntimeException".

/home/sb/InstanceOfTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.29 assertIsArray()

```
assertIsArray($actual[, $message = ''])
```

 $actual array $message

```
assertIsNotArray()
```

11.32: assertIsArray()

```php
<?php
use PHPUnit\Framework\TestCase;

class ArrayTest extends TestCase
{
    public function testFailure()
    {
```

```
        $this->assertIsArray(null);
    }
}
```

```
$ phpunit ArrayTest
PHPUnit |version|.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ArrayTest::testFailure
Failed asserting that null is of type "array".

/home/sb/ArrayTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.30 assertIsBool()

```
assertIsBool($actual[, $message = ''])
```

$actual	bool	$message

assertIsNotBool()

11.33: assertIsBool()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class BoolTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertIsBool(null);
    }
}
```

```
$ phpunit BoolTest
PHPUnit |version|.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) BoolTest::testFailure
Failed asserting that null is of type "bool".

/home/sb/BoolTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.31 assertIsCallable()

```
assertIsCallable($actual[, $message = ''])
```

 $actual callable $message

```
assertIsNotCallable()
```

11.34: assertIsCallable()

```
<?php
use PHPUnit\Framework\TestCase;

class CallableTest extends TestCase
{
    public function testFailure()
    {
        $this->assertIsCallable(null);
    }
}
```

```
$ phpunit CallableTest
PHPUnit |version|.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) CallableTest::testFailure
Failed asserting that null is of type "callable".

/home/sb/CallableTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.32 assertIsFloat()

```
assertIsFloat($actual[, $message = ''])
```

 $actual float $message

```
assertIsNotFloat()
```

11.35: assertIsFloat()

```
<?php
use PHPUnit\Framework\TestCase;

class FloatTest extends TestCase
{
    public function testFailure()
    {
        $this->assertIsFloat(null);
    }
}
```

```
$ phpunit FloatTest
PHPUnit |version|.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) FloatTest::testFailure
Failed asserting that null is of type "float".

/home/sb/FloatTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.33 assertIsInt()

```
assertIsInt($actual[, $message = ''])
```

$actual	int	$message

```
assertIsNotInt()
```

11.36: assertIsInt()

```
<?php
use PHPUnit\Framework\TestCase;

class IntTest extends TestCase
{
    public function testFailure()
    {
        $this->assertIsInt(null);
    }
}
```

```
$ phpunit IntTest
PHPUnit |version|.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) IntTest::testFailure
Failed asserting that null is of type "int".

/home/sb/IntTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.34 assertIsIterable()

```
assertIsIterable($actual[, $message = ''])
```

 $actual iterable $message

assertIsNotIterable()

11.37: assertIsIterable()

```
<?php
use PHPUnit\Framework\TestCase;

class IterableTest extends TestCase
{
    public function testFailure()
    {
        $this->assertIsIterable(null);
    }
}
```

```
$ phpunit IterableTest
PHPUnit |version|.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) IterableTest::testFailure
Failed asserting that null is of type "iterable".

/home/sb/IterableTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.35 assertIsNumeric()

assertIsNumeric($actual[, $message = ''])

 $actual numeric $message

assertIsNotNumeric()

11.38: assertIsNumeric()

```
<?php
use PHPUnit\Framework\TestCase;

class NumericTest extends TestCase
{
    public function testFailure()
    {
        $this->assertIsNumeric(null);
    }
}
```

```
$ phpunit NumericTest
PHPUnit |version|.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb
```

```
There was 1 failure:

1) NumericTest::testFailure
Failed asserting that null is of type "numeric".

/home/sb/NumericTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.36 assertIsObject()

```
assertIsObject($actual[, $message = ''])
```

$actual	object	$message

assertIsNotObject()

11.39: assertIsObject()

```
<?php
use PHPUnit\Framework\TestCase;

class ObjectTest extends TestCase
{
    public function testFailure()
    {
        $this->assertIsObject(null);
    }
}
```

```
$ phpunit ObjectTest
PHPUnit |version|.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ObjectTest::testFailure
Failed asserting that null is of type "object".

/home/sb/ObjectTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.37 assertIsResource()

```
assertIsResource($actual[, $message = ''])
```

$actual	resource	$message

assertIsNotResource()

11.40: assertIsResource()

```
<?php
use PHPUnit\Framework\TestCase;

class ResourceTest extends TestCase
{
    public function testFailure()
    {
        $this->assertIsResource(null);
    }
}
```

```
$ phpunit ResourceTest
PHPUnit |version|.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) ResourceTest::testFailure
Failed asserting that null is of type "resource".

/home/sb/ResourceTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.38 assertIsScalar()

```
assertIsScalar($actual[, $message = ''])
```

$actual	scalar	$message

assertIsNotScalar()

11.41: assertIsScalar()

```
<?php
use PHPUnit\Framework\TestCase;

class ScalarTest extends TestCase
{
    public function testFailure()
    {
        $this->assertIsScalar(null);
    }
}
```

```
$ phpunit ScalarTest
PHPUnit |version|.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:
```

```
1) ScalarTest::testFailure
Failed asserting that null is of type "scalar".

/home/sb/ScalarTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.39 assertIsString()

```
assertIsString($actual[, $message = ''])
```

$actual	string	$message

assertIsNotString()

11.42: assertIsString()

```
<?php
use PHPUnit\Framework\TestCase;

class StringTest extends TestCase
{
    public function testFailure()
    {
        $this->assertIsString(null);
    }
}
```

```
$ phpunit StringTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) StringTest::testFailure
Failed asserting that null is of type "string".

/home/sb/StringTest.php:8

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.40 assertIsReadable()

```
assertIsReadable(string $filename[, string $message = ''])
```

$filename	$message

assertIsNotReadable()

11.43: assertIsReadable()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;
```

```php
final class IsReadableTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertIsReadable('/path/to/unreadable');
    }
}
```

```
$ phpunit IsReadableTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) IsReadableTest::testFailure
Failed asserting that "/path/to/unreadable" is readable.

/home/sb/IsReadableTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.41 assertIsWritable()

```
assertIsWritable(string $filename[, string $message = ''])
```

 $filename $message

assertIsNotWritable()

11.44: assertIsWritable()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class IsWritableTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertIsWritable('/path/to/unwritable');
    }
}
```

```
$ phpunit IsWritableTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) IsWritableTest::testFailure
Failed asserting that "/path/to/unwritable" is writable.
```

```
/home/sb/IsWritableTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.42 assertJsonFileEqualsJsonFile()

```
assertJsonFileEqualsJsonFile(mixed $expectedFile, mixed $actualFile[, string $message
= ''])
```

$actualFile	$expectedFile	$message

11.45: assertJsonFileEqualsJsonFile()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class JsonFileEqualsJsonFileTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertJsonFileEqualsJsonFile(
          'path/to/fixture/file', 'path/to/actual/file');
    }
}
```

```
$ phpunit JsonFileEqualsJsonFileTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) JsonFileEqualsJsonFile::testFailure
Failed asserting that '{"Mascot":"Tux"}' matches JSON string "["Mascott", "Tux", "OS",
↪ "Linux"]".

/home/sb/JsonFileEqualsJsonFileTest.php:5

FAILURES!
Tests: 1, Assertions: 3, Failures: 1.
```

11.43 assertJsonStringEqualsJsonFile()

```
assertJsonStringEqualsJsonFile(mixed $expectedFile, mixed $actualJson[, string
$message = ''])
```

$actualJson	$expectedFile	$message

11.46: assertJsonStringEqualsJsonFile()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class JsonStringEqualsJsonFileTest extends TestCase
{
```

```
    public function testFailure(): void
    {
        $this->assertJsonStringEqualsJsonFile(
            'path/to/fixture/file', json_encode(['Mascot' => 'ux'])
        );
    }
}
```

```
$ phpunit JsonStringEqualsJsonFileTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) JsonStringEqualsJsonFile::testFailure
Failed asserting that '{"Mascot":"ux"}' matches JSON string "{"Mascott":"Tux"}".

/home/sb/JsonStringEqualsJsonFileTest.php:5

FAILURES!
Tests: 1, Assertions: 3, Failures: 1.
```

11.44 assertJsonStringEqualsJsonString()

```
assertJsonStringEqualsJsonString(mixed $expectedJson, mixed $actualJson[, string
$message = ''])
```

$actualJson	$expectedJson	$message

11.47: assertJsonStringEqualsJsonString()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class JsonStringEqualsJsonStringTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertJsonStringEqualsJsonString(
            json_encode(['Mascot' => 'Tux']),
            json_encode(['Mascot' => 'ux'])
        );
    }
}
```

```
$ phpunit JsonStringEqualsJsonStringTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) JsonStringEqualsJsonStringTest::testFailure
```

```
Failed asserting that two objects are equal.
--- Expected
+++ Actual
@@ @@
 stdClass Object (
 -    'Mascot' => 'Tux'
 +    'Mascot' => 'ux'
)

/home/sb/JsonStringEqualsJsonStringTest.php:5

FAILURES!
Tests: 1, Assertions: 3, Failures: 1.
```

11.45 assertLessThan()

```
assertLessThan(mixed $expected, mixed $actual[, string $message = ''])
```

$actual	$expected	$message

11.48: assertLessThan()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class LessThanTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertLessThan(1, 2);
    }
}
```

```
$ phpunit LessThanTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) LessThanTest::testFailure
Failed asserting that 2 is less than 1.

/home/sb/LessThanTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.46 assertLessThanOrEqual()

```
assertLessThanOrEqual(mixed $expected, mixed $actual[, string $message = ''])
```

$actual	$expected	$message

11.49: assertLessThanOrEqual()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class LessThanOrEqualTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertLessThanOrEqual(1, 2);
    }
}
```

```
$ phpunit LessThanOrEqualTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) LessThanOrEqualTest::testFailure
Failed asserting that 2 is equal to 1 or is less than 1.

/home/sb/LessThanOrEqualTest.php:6

FAILURES!
Tests: 1, Assertions: 2, Failures: 1.
```

11.47 assertNan()

```
assertNan(mixed $variable[, string $message = ''])
```

 $variable NAN $message

11.50: assertNan()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class NanTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertNan(1);
    }
}
```

```
$ phpunit NanTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:
```

```
1) NanTest::testFailure
Failed asserting that 1 is nan.

/home/sb/NanTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.48 assertNull()

```
assertNull(mixed $variable[, string $message = ''])
```

 $actual null $message

```
assertNotNull()
```

11.51: assertNull()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class NullTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertNull('foo');
    }
}
```

```
$ phpunit NotNullTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) NullTest::testFailure
Failed asserting that 'foo' is null.

/home/sb/NotNullTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.49 assertObjectHasAttribute()

```
assertObjectHasAttribute(string $attributeName, object $object[, string $message =
'']])
```

 $object->attributeName $message

```
assertObjectNotHasAttribute()
```

11.52: assertObjectHasAttribute()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class ObjectHasAttributeTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertObjectHasAttribute('foo', new stdClass);
    }
}
```

```
$ phpunit ObjectHasAttributeTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) ObjectHasAttributeTest::testFailure
Failed asserting that object of class "stdClass" has attribute "foo".

/home/sb/ObjectHasAttributeTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.50 assertMatchesRegularExpression()

```
assertMatchesRegularExpression(string $pattern, string $string[, string $message =
''])
```

$string $pattern $message

assertDoesNotMatchRegularExpression()

11.53: assertMatchesRegularExpression()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class RegExpTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertMatchesRegularExpression('/foo/', 'bar');
    }
}
```

```
$ phpunit RegExpTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb
```

```
There was 1 failure:

1) RegExpTest::testFailure
Failed asserting that 'bar' matches PCRE pattern "/foo/".

/home/sb/RegExpTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.51 assertStringMatchesFormat()

```
assertStringMatchesFormat(string $format, string $string[, string $message = ''])
```

 $string $format $message

```
assertStringNotMatchesFormat()
```

11.54: assertStringMatchesFormat()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StringMatchesFormatTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertStringMatchesFormat('%i', 'foo');
    }
}
```

```
$ phpunit StringMatchesFormatTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) StringMatchesFormatTest::testFailure
Failed asserting that 'foo' matches PCRE pattern "/^[+-]?d+$/s".

/home/sb/StringMatchesFormatTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

- %e Linux /
- %s
- %S
- %a
- %A
- %w
- %i +3142 -3142

- %d 123456
- %x 0-9 a-f A-F
- %f 3.142 -3.142 3.142E-10 3.142e+10
- %c
- %% %

11.52 assertStringMatchesFormatFile()

assertStringMatchesFormatFile(string $formatFile, string $string[, string $message = ''])

$string $formatFile $message

assertStringNotMatchesFormatFile()

11.55: assertStringMatchesFormatFile()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StringMatchesFormatFileTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertStringMatchesFormatFile('/path/to/expected.txt', 'foo');
    }
}
```

```
$ phpunit StringMatchesFormatFileTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) StringMatchesFormatFileTest::testFailure
Failed asserting that 'foo' matches PCRE pattern "/^[+-]?d+
$/s".

/home/sb/StringMatchesFormatFileTest.php:6

FAILURES!
Tests: 1, Assertions: 2, Failures: 1.
```

11.53 assertSame()

assertSame(mixed $expected, mixed $actual[, string $message = ''])

$expected $actual $message

assertNotSame()

11.56: assertSame()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class SameTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertSame('2204', 2204);
    }
}
```

```
$ phpunit SameTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) SameTest::testFailure
Failed asserting that 2204 is identical to '2204'.

/home/sb/SameTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

```
assertSame(object $expected, object $actual[, string $message = ''])
```

$expected $actual $message

11.57: assertSame()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class SameTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertSame(new stdClass, new stdClass);
    }
}
```

```
$ phpunit SameTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 4.75Mb

There was 1 failure:

1) SameTest::testFailure
Failed asserting that two variables reference the same object.

/home/sb/SameTest.php:6
```

```
FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.54 assertStringEndsWith()

```
assertStringEndsWith(string $suffix, string $string[, string $message = ''])
```

 $string $suffix $message

```
assertStringEndsNotWith()
```

11.58: assertStringEndsWith()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StringEndsWithTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertStringEndsWith('suffix', 'foo');
    }
}
```

```
$ phpunit StringEndsWithTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 1 second, Memory: 5.00Mb

There was 1 failure:

1) StringEndsWithTest::testFailure
Failed asserting that 'foo' ends with "suffix".

/home/sb/StringEndsWithTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.55 assertStringEqualsFile()

```
assertStringEqualsFile(string $expectedFile, string $actualString[, string $message =
'')])
```

 $expectedFile $actualString $message

```
assertStringNotEqualsFile()
```

11.59: assertStringEqualsFile()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StringEqualsFileTest extends TestCase
{
    public function testFailure(): void
    {
```

```
        $this->assertStringEqualsFile('/home/sb/expected', 'actual');
    }
}
```

```
$ phpunit StringEqualsFileTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) StringEqualsFileTest::testFailure
Failed asserting that two strings are equal.
--- Expected
+++ Actual
@@ @@
-'expected
-'
+'actual'

/home/sb/StringEqualsFileTest.php:6

FAILURES!
Tests: 1, Assertions: 2, Failures: 1.
```

11.56 assertStringStartsWith()

```
assertStringStartsWith(string $prefix, string $string[, string $message = ''])
```

 $string $prefix $message

```
assertStringStartsNotWith()
```

11.60: assertStringStartsWith()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class StringStartsWithTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertStringStartsWith('prefix', 'foo');
    }
}
```

```
$ phpunit StringStartsWithTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) StringStartsWithTest::testFailure
```

```
Failed asserting that 'foo' starts with "prefix".

/home/sb/StringStartsWithTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.57 assertThat()

PHPUnit\Framework\Constraint assertThat() 11.61 logicalNot() equalTo() assertNotEquals()

```
assertThat(mixed $value, PHPUnit\Framework\Constraint $constraint[, $message = ''])
```

$value $constraint $message

11.61: assertThat()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class BiscuitTest extends TestCase
{
    public function testEquals(): void
    {
        $theBiscuit = new Biscuit('Ginger');
        $myBiscuit  = new Biscuit('Ginger');

        $this->assertThat(
          $theBiscuit,
          $this->logicalNot(
            $this->equalTo($myBiscuit)
          )
        );
    }
}
```

11.1 PHPUnit\Framework\Constraint

11.1:

PHPUnit\Framework\Constraint\IsAnything anything()
PHPUnit\Framework\Constraint\ArrayHasKey arrayHasKey(mixed $key)
PHPUnit\Framework\Constraint\TraversableContains contains(mixed $value)
PHPUnit\Framework\Constraint\TraversableContainsOnly containsOnly(string $type)
PHPUnit\Framework\Constraint\TraversableContainsOnly containsOnlyInstancesOf(string $classname)
PHPUnit\Framework\Constraint\IsEqual equalTo($value, $delta = 0, $maxDepth = 10)
PHPUnit\Framework\Constraint\DirectoryExists directoryExists()
PHPUnit\Framework\Constraint\FileExists fileExists()
PHPUnit\Framework\Constraint\IsReadable isReadable()
PHPUnit\Framework\Constraint\IsWritable isWritable()
PHPUnit\Framework\Constraint\GreaterThan greaterThan(mixed $value)
PHPUnit\Framework\Constraint\LogicalOr greaterThanOrEqual(mixed $value)
PHPUnit\Framework\Constraint\ClassHasAttribute classHasAttribute(string $attributeName)
PHPUnit\Framework\Constraint\ClassHasStaticAttribute classHasStaticAttribute(string $attributeName)
PHPUnit\Framework\Constraint\ObjectHasAttribute objectHasAttribute(string $attributeName)
PHPUnit\Framework\Constraint\IsIdentical identicalTo(mixed $value)

11.1 – continued from previous page

PHPUnit\Framework\Constraint\IsFalse isFalse()
PHPUnit\Framework\Constraint\IsInstanceOf isInstanceOf(string $className)
PHPUnit\Framework\Constraint\IsNull isNull()
PHPUnit\Framework\Constraint\IsTrue isTrue()
PHPUnit\Framework\Constraint\IsType isType(string $type)
PHPUnit\Framework\Constraint\LessThan lessThan(mixed $value)
PHPUnit\Framework\Constraint\LogicalOr lessThanOrEqual(mixed $value)
logicalAnd()
logicalNot(PHPUnit\Framework\Constraint $constraint)
logicalOr()
logicalXor()
PHPUnit\Framework\Constraint\PCREMatch matchesRegularExpression(string $pattern)
PHPUnit\Framework\Constraint\StringContains stringContains(string $string, bool $case)
PHPUnit\Framework\Constraint\StringEndsWith stringEndsWith(string $suffix)
PHPUnit\Framework\Constraint\StringStartsWith stringStartsWith(string $prefix)

11.58 assertTrue()

```
assertTrue(bool $condition[, string $message = ''])
```

 $condition false $message

assertNotTrue()

11.62: assertTrue()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class TrueTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertTrue(false);
    }
}
```

```
$ phpunit TrueTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) TrueTest::testFailure
Failed asserting that false is true.

/home/sb/TrueTest.php:6

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

11.59 assertXmlFileEqualsXmlFile()

```
assertXmlFileEqualsXmlFile(string $expectedFile, string $actualFile[, string $message
= ''])
```

$actualFile XML $expectedFile XML $message

assertXmlFileNotEqualsXmlFile()

11.63: assertXmlFileEqualsXmlFile()

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class XmlFileEqualsXmlFileTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertXmlFileEqualsXmlFile(
          '/home/sb/expected.xml', '/home/sb/actual.xml');
    }
}
```

```
$ phpunit XmlFileEqualsXmlFileTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) XmlFileEqualsXmlFileTest::testFailure
Failed asserting that two DOM documents are equal.
--- Expected
+++ Actual
@@ @@
 <?xml version="1.0"?>
 <foo>
-  <bar/>
+  <baz/>
 </foo>

/home/sb/XmlFileEqualsXmlFileTest.php:7

FAILURES!
Tests: 1, Assertions: 3, Failures: 1.
```

11.60 assertXmlStringEqualsXmlFile()

```
assertXmlStringEqualsXmlFile(string $expectedFile, string $actualXml[, string $message
= ''])
```

$actualXml XML $expectedFile XML $message

assertXmlStringNotEqualsXmlFile()

11.64: assertXmlStringEqualsXmlFile()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class XmlStringEqualsXmlFileTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertXmlStringEqualsXmlFile(
          '/home/sb/expected.xml', '<foo><baz/></foo>');
    }
}
```

```
$ phpunit XmlStringEqualsXmlFileTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.25Mb

There was 1 failure:

1) XmlStringEqualsXmlFileTest::testFailure
Failed asserting that two DOM documents are equal.
--- Expected
+++ Actual
@@ @@
 <?xml version="1.0"?>
 <foo>
-  <bar/>
+  <baz/>
 </foo>

/home/sb/XmlStringEqualsXmlFileTest.php:7

FAILURES!
Tests: 1, Assertions: 2, Failures: 1.
```

11.61 assertXmlStringEqualsXmlString()

```
assertXmlStringEqualsXmlString(string $expectedXml, string $actualXml[, string
$message = ''])
```

 $actualXml XML $expectedXml XML $message

assertXmlStringNotEqualsXmlString()

11.65: assertXmlStringEqualsXmlString()

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class XmlStringEqualsXmlStringTest extends TestCase
{
    public function testFailure(): void
    {
        $this->assertXmlStringEqualsXmlString(
          '<foo><bar/></foo>', '<foo><baz/></foo>');
```

```
    }
}
```

```
$ phpunit XmlStringEqualsXmlStringTest
PHPUnit latest.0 by Sebastian Bergmann and contributors.

F

Time: 0 seconds, Memory: 5.00Mb

There was 1 failure:

1) XmlStringEqualsXmlStringTest::testFailure
Failed asserting that two DOM documents are equal.
--- Expected
+++ Actual
@@ @@
 <?xml version="1.0"?>
 <foo>
-  <bar/>
+  <baz/>
 </foo>

/home/sb/XmlStringEqualsXmlStringTest.php:7

FAILURES!
Tests: 1, Assertions: 1, Failures: 1.
```

CHAPTER 12

PHP PHP @annotation arguments
PHP API getDocComment() PHPUnit

PHP /** */

PHPUnit

12.1 @author

@author @group *@group*

12.2 @after

@after

```php
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class MyTest extends TestCase
{
    /**
     * @after
     */
    public function tearDownSomeFixtures(): void
    {
        // ...
    }

    /**
     * @after
     */
    public function tearDownSomeOtherFixtures(): void
```

```
    {
        // ...
    }
}
```

12.3 @afterClass

@afterClass

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class MyTest extends TestCase
{
    /**
     * @afterClass
     */
    public static function tearDownSomeSharedFixtures(): void
    {
        // ...
    }

    /**
     * @afterClass
     */
    public static function tearDownSomeOtherSharedFixtures(): void
    {
        // ...
    }
}
```

12.4 @backupGlobals

PHPUnit

@backupGlobals enabled

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

/**
 * @backupGlobals enabled
 */
final class MyTest extends TestCase
{
    // ...
}
```

@backupGlobals

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

/**
 * @backupGlobals enabled
 */
final class MyTest extends TestCase
{
```

```
    public function testThatInteractsWithGlobalVariables()
    {
        // ...
    }

    /**
     * @backupGlobals disabled
     */
    public function testThatDoesNotInteractWithGlobalVariables(): void
    {
        // ...
    }
}
```

12.5 @backupStaticAttributes

PHPUnit

@backupStaticAttributes enabled

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

/**
 * @backupStaticAttributes enabled
 */
final class MyTest extends TestCase
{
    // ...
}
```

@backupStaticAttributes

```
use PHPUnit\Framework\TestCase;

/**
 * @backupStaticAttributes enabled
 */
class MyTest extends TestCase
{
    public function testThatInteractsWithStaticAttributes(): void
    {
        // ...
    }

    /**
     * @backupStaticAttributes disabled
     */
    public function testThatDoesNotInteractWithStaticAttributes(): void
    {
        // ...
    }
}
```

PHP @backupStaticAttributes

12.6 @before

@before

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class MyTest extends TestCase
{
    /**
     * @before
     */
    public function setupSomeFixtures(): void
    {
        // ...
    }

    /**
     * @before
     */
    public function setupSomeOtherFixtures(): void
    {
        // ...
    }
}
```

12.7 @beforeClass

@beforeClass

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class MyTest extends TestCase
{
    /**
     * @beforeClass
     */
    public static function setUpSomeSharedFixtures(): void
    {
        // ...
    }

    /**
     * @beforeClass
     */
    public static function setUpSomeOtherSharedFixtures(): void
    {
        // ...
    }
}
```

12.8 @codeCoverageIgnore*

@codeCoverageIgnore @codeCoverageIgnoreStart @codeCoverageIgnoreEnd

12.9 @covers

`@covers`

```
/**
 * @covers \BankAccount
 */
public function testBalanceIsInitiallyZero(): void
{
    $this->assertSame(0, $this->ba->getBalance());
}
```

`forceCoversAnnotation` `true` `@covers`

12.1 `@covers`

FQCN fully-qualified class name

12.1:

`@covers ClassName::methodName`	
`@covers ClassName`	
`@covers ClassName<extended>`	
`@covers ClassName::<public>`	public
`@covers ClassName::<protected>`	protected
`@covers ClassName::<private>`	private
`@covers ClassName::<!public>`	public
`@covers ClassName::<!protected>`	protected
`@covers ClassName::<!private>`	private
`@covers ::functionName`	

12.10 @coversDefaultClass

`@coversDefaultClass` `@covers` 12.1

FQCN fully-qualified class name

12.1: @coversDefaultClass

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

/**
 * @coversDefaultClass \Foo\CoveredClass
 */
final class CoversDefaultClassTest extends TestCase
{
    /**
     * @covers ::publicMethod
     */
    public function testSomething(): void
    {
        $o = new Foo\CoveredClass;
        $o->publicMethod();
    }
}
```

12.11 @coversNothing

`@coversNothing`

`@covers`

12.12 @dataProvider

`provider()` `@dataProvider`

12.13 @depends

PHPUnit producer fixture consumer *@depends*
`@depends`

12.14 @doesNotPerformAssertions

12.15 @group

`@group`

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class MyTest extends TestCase
{
    /**
     * @group specification
     */
    public function testSomething(): void
    {
    }

    /**
     * @group regression
     * @group bug2204
     */
    public function testSomethingElse(): void
    {
    }
}
```

`@group` “ ”

`--group` `--exclude-group` XML

12.16 @large

`@large` `@group large`

`PHP_Invoker` 60 large XML `timeoutForLargeTests`

12.17 @medium

`@medium` `@group medium` medium `@large`

`PHP_Invoker` 10 medium XML `timeoutForMediumTests`

12.18 @preserveGlobalState

PHPUnit `@preserveGlobalState` PHPUnit

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class MyTest extends TestCase
{
    /**
     * @runInSeparateProcess
     * @preserveGlobalState disabled
     */
    public function testInSeparateProcess(): void
    {
        // ...
    }
}
```

12.19 @requires

`@requires` PHP

@requires

12.20 @runTestsInSeparateProcesses

PHP

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

/**
 * @runTestsInSeparateProcesses
 */
final class MyTest extends TestCase
{
    // ...
}
```

PHPUnit *@preserveGlobalState*

12.21 @runInSeparateProcess

PHP

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

final class MyTest extends TestCase
{
    /**
     * @runInSeparateProcess
     */
    public function testInSeparateProcess(): void
    {
        // ...
    }
}
```

PHPUnit *@preserveGlobalState*

12.22 @small

`@small` `@group small` small `@medium` `@large`

PHP_Invoker 1 small XML `timeoutForSmallTests`

`@small` `@medium` `@large`

12.23 @test

test `@test`

```
/**
 * @test
 */
public function initialBalanceShouldBe0(): void
{
    $this->assertSame(0, $this->ba->getBalance());
}
```

12.24 @testdox

`@testdox`

```
<?php declare(strict_types=1);
use PHPUnit\Framework\TestCase;

/**
 * @testdox A bank account
 */
final class BankAccountTest extends TestCase
```

```
{
    /**
     * @testdox has an initial balance of zero
     */
    public function balanceIsInitiallyZero(): void
    {
        $this->assertSame(0, $this->ba->getBalance());
    }
}
```

PHPUnit 7.0 bug @testdox @test

@testdox @dataProvider

```
/**
 * @dataProvider additionProvider
 * @testdox Adding $a to $b results in $expected
 */
public function testAdd($a, $b, $expected)
{
    $this->assertSame($expected, $a + $b);
}

public function additionProvider()
{
    return [
        [0, 0, 0],
        [0, 1, 1],
        [1, 0, 1],
        [1, 1, 3]
    ];
}
```

12.25 @testWith

@dataProvider @testWith

JSON

```
/**
 * @testWith ["test", 4]
 *           ["longer-string", 13]
 */
public function testStringLength(string $input, int $expectedLength): void
{
    $this->assertSame($expectedLength, strlen($input));
}
```

JSON

```
/**
 * @testWith [{"day": "monday", "conditions": "sunny"}, ["day", "conditions"]]
 */
public function testArrayKeys(array $array, array $keys): void
{
```

```
    $this->assertSame($keys, array_keys($array));
}
```

12.26 @ticket

`@ticket` `@group` *@group* ID

12.27 @uses

`@uses`

```
/**
 * @covers \BankAccount
 * @uses   \Money
 */
public function testMoneyCanBeDepositedInAccount(): void
{
    // ...
}
```

9.2

FQCN fully-qualified class name

CHAPTER 13

XML

13.1 <phpunit>

13.1.1 backupGlobals

`true` `false` `false`

PHPUnit

`@backupGlobals`

13.1.2 backupStaticAttributes

`true` `false` `false`

PHPUnit

`@backupStaticAttributes`

13.1.3 bootstrap

13.1.4 cacheResult

`true` `false` `true`

13.1.5 cacheResultFile

13.1.6 colors

true false false

PHPUnit

true --colors=auto

false --colors=never

13.1.7 columns

max 80

max

13.1.8 convertDeprecationsToExceptions

true false true

E_DEPRECATED E_USER_DEPRECATED

13.1.9 convertErrorsToExceptions

true false true

E_ERROR E_USER_ERROR

13.1.10 convertNoticesToExceptions

true false true

E_STRICT E_NOTICE E_USER_NOTICE

13.1.11 convertWarningsToExceptions

true false true

E_WARNING E_USER_WARNING

13.1.12 forceCoversAnnotation

true false false

@covers

13.1.13 printerClass

PHPUnit\TextUI\ResultPrinter

PHPUnit\TextUI\ResultPrinter PHPUnit\TextUI\ResultPrinter

13.1.14 printerFile

printerClass

13.1.15 processIsolation

true false false

PHP

13.1.16 stopOnError

true false false

“ error ”

13.1.17 stopOnFailure

true false false

“ failure ”

13.1.18 stopOnIncomplete

true false false

“ incomplete ”

13.1.19 stopOnRisky

true false false

“ risky ”

13.1.20 stopOnSkipped

true false false

“ skipped ”

13.1.21 stopOnWarning

true false false

“ warning ”

13.1.22 stopOnDefect

true false false

“ error ” “ failure ” “ risky ” “ warning ”

13.1.23 failOnRisky

true false false

risky PHPUnit sell

13.1.24 failOnWarning

`true` `false` `false`

warning PHPUnit sell

13.1.25 beStrictAboutChangesToGlobalState

`true` `false` `false`

PHPUnit risky

13.1.26 beStrictAboutOutputDuringTests

`true` `false` `false`

PHPUnit risky

13.1.27 beStrictAboutResourceUsageDuringSmallTests

`true` `false` `false`

`@small` `resource` PHP PHPUnit risky

13.1.28 beStrictAboutTestsThatDoNotTestAnything

`true` `false` `true`

PHPUnit risky

13.1.29 beStrictAboutTodoAnnotatedTests

`true` `false` `false`

`@todo` PHPUnit

13.1.30 beStrictAboutCoversAnnotation

`true` `false` `false`

`@covers` `@uses` PHPUnit

13.1.31 enforceTimeLimit

`true` `false` `false`

13.1.32 defaultTimeLimit

0

13.1.33 timeoutForSmallTests

1

@small

13.1.34 timeoutForMediumTests

10

@medium

13.1.35 timeoutForLargeTests

60

@large

13.1.36 testSuiteLoaderClass

PHPUnit\Runner\StandardTestSuiteLoader

PHPUnit\Runner\TestSuiteLoader

13.1.37 testSuiteLoaderFile

testSuiteLoaderClass

13.1.38 defaultTestSuite

13.1.39 verbose

true false false

13.1.40 stderr

true false false

PHPUnit stderr stdout

13.1.41 reverseDefectList

true false false

13.1.42 registerMockObjectsFromTestArgumentsRecursively

true false false

@depends

13.1.43 extensionsDirectory

phpunit.phar * .phar PHPUnit

13.1.44 executionOrder

default defects depends no-depends duration random reverse size

,

13.1.45 resolveDependencies

true false true

@depends

13.1.46 testdox

true false false

TestDox

13.1.47 noInteraction

true false false

TestDox

13.2 <testsuites>

<phpunit>

<testsuite>

13.2.1 <testsuite>

<testsuites>

<testsuite> name <directory> <file>

```
<testsuites>
  <testsuite name="unit">
    <directory>tests/unit</directory>
  </testsuite>

  <testsuite name="integration">
    <directory>tests/integration</directory>
  </testsuite>

  <testsuite name="edge-to-edge">
    <directory>tests/edge-to-edge</directory>
  </testsuite>
</testsuites>
```

phpVersion phpVersionOperator PHP

```
<testsuites>
  <testsuite name="unit">
    <directory phpVersion="8.0.0" phpVersionOperator=">=">tests/unit</directory>
  </testsuite>
</testsuites>
```

PHP 8.0.0 tests/unit phpVersionOperator >=

13.3 <coverage>

<phpunit>

<coverage>

```
<coverage cacheDirectory="/path/to/directory"
          includeUncoveredFiles="true"
          processUncoveredFiles="true"
          pathCoverage="false"
          ignoreDeprecatedCodeUnits="true"
          disableCodeCoverageIgnore="true">
    <!-- ... -->
</coverage>
```

13.3.1 cacheDirectory

cacheDirectory

13.3.2 includeUncoveredFiles

true false true

true

13.3.3 processUncoveredFiles

true false false

true

13.3.4 ignoreDeprecatedCodeUnits

true false false

@deprecated

13.3.5 pathCoverage

true false false

false

true Xdebug

13.3.6 disableCodeCoverageIgnore

true false false

@codeCoverageIgnore*

13.3.7 The <include> Element

<coverage>

```
<include>
    <directory suffix=".php">src</directory>
</include>
```

PHPUnit src .php

13.3.8 <exclude>

<coverage>

```
<include>
    <directory suffix=".php">src</directory>
</include>

<exclude>
    <directory suffix=".php">src/generated</directory>
    <file>src/autoload.php</file>
</exclude>
```

PHPUnit src .php src/generated .php src/autoload.php

13.3.9 <directory>

<include> <exclude>

prefix

suffix

string '.php'

phpVersion

PHPUnit PHP

phpVersionOperator

'<' 'lt' '<=' 'le' '>' 'gt' '>=' 'ge' '==' '=' 'eq', '!=', '<>', 'ne' '>='

PHPUnit PHP version_compare()

13.3.10 <file>

<include> <exclude>

13.3.11 <report>

<coverage>

```
<report>
    <clover outputFile="clover.xml"/>
    <crap4j outputFile="crap4j.xml" threshold="50"/>
    <html outputDirectory="html-coverage" lowUpperBound="50" highLowerBound="90"/>
    <php outputFile="coverage.php"/>
    <text outputFile="coverage.txt" showUncoveredFiles="false" showOnlySummary="true"/>
    <xml outputDirectory="xml-coverage"/>
</report>
```

<clover>

<report>

Clover XML

outputFile

Clover XML

<crap4j>

<report>

Crap4J XML

outputFile

Crap4J XML

threshold

integer 50

<html>

<report>
HTML

outputDirectory

HTML

lowUpperBound

integer 50
“ ”

highLowerBound

90
“ ”

<php>

<report>
PHP

outputFile

PHP

<text>

<report>

outputFile

showUncoveredFiles

true false false

showOnlySummary

true false false

<xml>

<report>

PHPUnit XML

outputDirectory

PHPUnit XML

13.4 <logging>

<phpunit>

<logging>

```
<logging>
    <junit outputFile="junit.xml"/>
    <teamcity outputFile="teamcity.txt"/>
    <testdoxHtml outputFile="testdox.html"/>
    <testdoxText outputFile="testdox.txt"/>
    <testdoxXml outputFile="testdox.xml"/>
    <text outputFile="logfile.txt"/>
</logging>
```

13.4.1 <junit>

<logging>

JUnit XML

outputFile

JUnit XML

13.4.2 <teamcity>

<logging>

TeamCity

outputFile

TeamCity

13.4.3 <testdoxHtml>

<logging>

TestDox HTML

outputFile

TestDox HTML

13.4.4 <testdoxText>

<logging>

TestDox

outputFile

TestDox

13.4.5 <testdoxXml>

<logging>

TestDox XML

outputFile

TestDox XML

13.4.6 <text>

<logging>

outputFile

13.5 <groups>

<phpunit>

<groups> <include> <exclude> <group> @group *@group*

```
<groups>
  <include>
    <group>name</group>
  </include>
  <exclude>
    <group>name</group>
  </exclude>
</groups>
```

--group name --exclude-group name PHPUnit

13.6 <testdoxGroups>

<phpunit>

… < > …

13.7 <listeners>

<phpunit>

<listeners> <listener>

13.7.1 <listener>

<listeners>

```
<listeners>
  <listener class="MyListener" file="/optional/path/to/MyListener.php">
    <arguments>
      <array>
        <element key="0">
          <string>Sebastian</string>
        </element>
      </array>
      <integer>22</integer>
      <string>April</string>
      <double>19.78</double>
      <null/>
      <object class="stdClass"/>
    </arguments>
  </listener>
</listeners>
```

XML $listener

```
$listener = new MyListener(
    ['Sebastian'],
    22,
    'April',
    19.78,
    null,
    new stdClass
);
```

PHPUnit\Framework\TestListener

13.8 <extensions>

<phpunit>

<extensions> <extension>

13.8.1 <extension>

<extensions>

```
<extensions>
    <extension class="Vendor\MyExtension"/>
</extensions>
```

<arguments>

<extension>

<arguments> <extension>

__constructor

- <boolean>
- <integer>
- <string>
- <double>
- <array>
- <object>

```
<extension class="Vendor\MyExtension">
    <arguments>
        <integer>1</integer>
        <integer>2</integer>
        <integer>3</integer>
        <string>hello world</string>
        <boolean>true</boolean>
        <double>1.23</double>
        <array>
            <element index="0">
                <string>value1</string>
            </element>
            <element index="1">
                <string>value2</string>
            </element>
        </array>
        <object class="Vendor\MyPhpClass">
            <string>constructor arg 1</string>
            <string>constructor arg 2</string>
        </object>
    </arguments>
</extension>
```

13.9 <php>

<phpunit>

<php> PHP include_path

13.9.1 <includePath>

<php>

include_path

13.9.2 <ini>

<php>

PHP

```
<php>
  <ini name="foo" value="bar"/>
</php>
```

XML PHP

```
ini_set('foo', 'bar');
```

13.9.3 <const>

<php>

```
<php>
  <const name="foo" value="bar"/>
</php>
```

XML PHP

```
define('foo', 'bar');
```

13.9.4 *<var>*

<php>

```
<php>
  <var name="foo" value="bar"/>
</php>
```

XML PHP

```
$GLOBALS['foo'] = 'bar';
```

13.9.5 <env>

<php>

$_ENV

```
<php>
  <env name="foo" value="bar"/>
</php>
```

XML PHP

```
$_ENV['foo'] = 'bar';
```

force

```
<php>
  <env name="foo" value="bar" force="true"/>
</php>
```

13.9.6 <get>

<php>

$_GET

```
<php>
  <get name="foo" value="bar"/>
</php>
```

XML PHP

```
$_GET['foo'] = 'bar';
```

13.9.7 <post>

<php>

$_POST

```
<php>
  <post name="foo" value="bar"/>
</php>
```

XML PHP

```
$_POST['foo'] = 'bar';
```

13.9.8 <cookie>

<php>

$_COOKIE

```
<php>
  <cookie name="foo" value="bar"/>
</php>
```

XML PHP

```
$_COOKIE['foo'] = 'bar';
```

13.9.9 <server>

<php>

$_SERVER

```
<php>
  <server name="foo" value="bar"/>
</php>
```

XML PHP

```
$_SERVER['foo'] = 'bar';
```

13.9.10 <files>

<php>

$_FILES

```
<php>
  <files name="foo" value="bar"/>
</php>
```

XML PHP

```
$_FILES['foo'] = 'bar';
```

13.9.11 <request>

<php>

$_REQUEST

```
<php>
  <request name="foo" value="bar"/>
</php>
```

XML PHP

```
$_REQUEST['foo'] = 'bar';
```

CHAPTER 14

[Meszaros2007] Gerard Meszaros xUnit Test Patterns: Refactoring Test Code

CHAPTER 15

```
   © 2005-2020 Sebastian Bergmann

    Creative Commons Attribution 3.0 Unported


--------------------------------------------------------------------



    *  --
    *  --




    *

    *

    *




======================================================================

Creative Commons Legal Code
Attribution 3.0 Unported

CREATIVE COMMONS CORPORATION IS NOT A LAW FIRM AND DOES NOT PROVIDE
LEGAL SERVICES. DISTRIBUTION OF THIS LICENSE DOES NOT CREATE AN
ATTORNEY-CLIENT RELATIONSHIP. CREATIVE COMMONS PROVIDES THIS
INFORMATION ON AN "AS-IS" BASIS. CREATIVE COMMONS MAKES NO
WARRANTIES REGARDING THE INFORMATION PROVIDED, AND DISCLAIMS
LIABILITY FOR DAMAGES RESULTING FROM ITS USE.
```

License

THE WORK (AS DEFINED BELOW) IS PROVIDED UNDER THE TERMS OF THIS
CREATIVE COMMONS PUBLIC LICENSE ("CCPL" OR "LICENSE"). THE WORK IS
PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE
WORK OTHER THAN AS AUTHORIZED UNDER THIS LICENSE OR COPYRIGHT LAW
IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND
AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS
LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU
THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF
SUCH TERMS AND CONDITIONS.

1. Definitions

   a. "Adaptation" means a work based upon the Work, or upon the
      Work and other pre-existing works, such as a translation,
      adaptation, derivative work, arrangement of music or other
      alterations of a literary or artistic work, or phonogram or
      performance and includes cinematographic adaptations or any
      other form in which the Work may be recast, transformed, or
      adapted including in any form recognizably derived from the
      original, except that a work that constitutes a Collection
      will not be considered an Adaptation for the purpose of this
      License. For the avoidance of doubt, where the Work is a
      musical work, performance or phonogram, the synchronization of
      the Work in timed-relation with a moving image ("synching")
      will be considered an Adaptation for the purpose of this
      License.

   b. "Collection" means a collection of literary or artistic works,
      such as encyclopedias and anthologies, or performances,
      phonograms or broadcasts, or other works or subject matter
      other than works listed in Section 1(f) below, which, by
      reason of the selection and arrangement of their contents,
      constitute intellectual creations, in which the Work is
      included in its entirety in unmodified form along with one or
      more other contributions, each constituting separate and
      independent works in themselves, which together are assembled
      into a collective whole. A work that constitutes a Collection
      will not be considered an Adaptation (as defined above) for
      the purposes of this License.

   c. "Distribute" means to make available to the public the
      original and copies of the Work or Adaptation, as appropriate,
      through sale or other transfer of ownership.

   d. "Licensor" means the individual, individuals, entity or
      entities that offer(s) the Work under the terms of this License.

   e. "Original Author" means, in the case of a literary or artistic
      work, the individual, individuals, entity or entities who
      created the Work or if no individual or entity can be
      identified, the publisher; and in addition (i) in the case of
      a performance the actors, singers, musicians, dancers, and
      other persons who act, sing, deliver, declaim, play in,
      interpret or otherwise perform literary or artistic works or
      expressions of folklore; (ii) in the case of a phonogram the
      producer being the person or legal entity who first fixes the
      sounds of a performance or other sounds; and, (iii) in the
      case of broadcasts, the organization that transmits the
      broadcast.

```
   f. "Work" means the literary and/or artistic work offered under
      the terms of this License including without limitation any
      production in the literary, scientific and artistic domain,
      whatever may be the mode or form of its expression including
      digital form, such as a book, pamphlet and other writing; a
      lecture, address, sermon or other work of the same nature; a
      dramatic or dramatico-musical work; a choreographic work or
      entertainment in dumb show; a musical composition with or
      without words; a cinematographic work to which are assimilated
      works expressed by a process analogous to cinematography; a
      work of drawing, painting, architecture, sculpture, engraving
      or lithography; a photographic work to which are assimilated
      works expressed by a process analogous to photography; a work
      of applied art; an illustration, map, plan, sketch or three-
      dimensional work relative to geography, topography,
      architecture or science; a performance; a broadcast; a
      phonogram; a compilation of data to the extent it is protected
      as a copyrightable work; or a work performed by a variety or
      circus performer to the extent it is not otherwise considered
      a literary or artistic work.

   g. "You" means an individual or entity exercising rights under
      this License who has not previously violated the terms of
      this License with respect to the Work, or who has received
      express permission from the Licensor to exercise rights under
      this License despite a previous violation.

   h. "Publicly Perform" means to perform public recitations of the
      Work and to communicate to the public those public
      recitations, by any means or process, including by wire or
      wireless means or public digital performances; to make
      available to the public Works in such a way that members of
      the public may access these Works from a place and at a place
      individually chosen by them; to perform the Work to the public
      by any means or process and the communication to the public of
      the performances of the Work, including by public digital
      performance; to broadcast and rebroadcast the Work by any
      means including signs, sounds or images.

   i. "Reproduce" means to make copies of the Work by any means
      including without limitation by sound or visual recordings and
      the right of fixation and reproducing fixations of the Work,
      including storage of a protected performance or phonogram in
      digital form or other electronic medium.

2. Fair Dealing Rights. Nothing in this License is intended to
   reduce, limit, or restrict any uses free from copyright or rights
   arising from limitations or exceptions that are provided for in
   connection with the copyright protection under copyright law or
   other applicable laws.

3. License Grant. Subject to the terms and conditions of this
   License, Licensor hereby grants You a worldwide, royalty-free,
   non-exclusive, perpetual (for the duration of the applicable
   copyright) license to exercise the rights in the Work as stated
   below:

   a. to Reproduce the Work, to incorporate the Work into one or
      more Collections, and to Reproduce the Work as incorporated
      in the Collections;

   b. to create and Reproduce Adaptations provided that any such
```

```
      Adaptation, including any translation in any medium, takes
      reasonable steps to clearly label, demarcate or otherwise
      identify that changes were made to the original Work. For
      example, a translation could be marked "The original work was
      translated from English to Spanish," or a modification could
      indicate "The original work has been modified.";

   c. to Distribute and Publicly Perform the Work including as
      incorporated in Collections; and,

   d. to Distribute and Publicly Perform Adaptations.

   e. For the avoidance of doubt:

      i. Non-waivable Compulsory License Schemes. In those
         jurisdictions in which the right to collect royalties
         through any statutory or compulsory licensing scheme cannot
         be waived, the Licensor reserves the exclusive right to
         collect such royalties for any exercise by You of the
         rights granted under this License;

      ii. Waivable Compulsory License Schemes. In those
          jurisdictions in which the right to collect royalties
          through any statutory or compulsory licensing scheme can
          be waived, the Licensor waives the exclusive right to
          collect such royalties for any exercise by You of the
          rights granted under this License; and,

      iii. Voluntary License Schemes. The Licensor waives the right
           to collect royalties, whether individually or, in the
           event that the Licensor is a member of a collecting
           society that administers voluntary licensing schemes, via
           that society, from any exercise by You of the rights
           granted under this License.

The above rights may be exercised in all media and formats whether
now known or hereafter devised. The above rights include the right
to make such modifications as are technically necessary to exercise
the rights in other media and formats. Subject to Section 8(f), all
rights not expressly granted by Licensor are hereby reserved.

4. Restrictions. The license granted in Section 3 above is expressly
   made subject to and limited by the following restrictions:

   a. You may Distribute or Publicly Perform the Work only under the
      terms of this License. You must include a copy of, or the
      Uniform Resource Identifier (URI) for, this License with every
      copy of the Work You Distribute or Publicly Perform. You may
      not offer or impose any terms on the Work that restrict the
      terms of this License or the ability of the recipient of the
      Work to exercise the rights granted to that recipient under
      the terms of the License. You may not sublicense the Work. You
      must keep intact all notices that refer to this License and to
      the disclaimer of warranties with every copy of the Work You
      Distribute or Publicly Perform. When You Distribute or
      Publicly Perform the Work, You may not impose any effective
      technological measures on the Work that restrict the ability
      of a recipient of the Work from You to exercise the rights
      granted to that recipient under the terms of the License. This
      Section 4(a) applies to the Work as incorporated in a
      Collection, but this does not require the Collection apart
      from the Work itself to be made subject to the terms of this
      License. If You create a Collection, upon notice from any
```

```
     Licensor You must, to the extent practicable, remove from the
     Collection any credit as required by Section 4(b), as
     requested. If You create an Adaptation, upon notice from any
     Licensor You must, to the extent practicable, remove from the
     Adaptation any credit as required by Section 4(b), as requested.

  b. If You Distribute, or Publicly Perform the Work or any
     Adaptations or Collections, You must, unless a request has
     been made pursuant to Section 4(a), keep intact all copyright
     notices for the Work and provide, reasonable to the medium or
     means You are utilizing: (i) the name of the Original Author
     (or pseudonym, if applicable) if supplied, and/or if the
     Original Author and/or Licensor designate another party or
     parties (e.g., a sponsor institute, publishing entity,
     journal) for attribution ("Attribution Parties") in Licensor's
     copyright notice, terms of service or by other reasonable
     means, the name of such party or parties; (ii) the title of
     the Work if supplied; (iii) to the extent reasonably
     practicable, the URI, if any, that Licensor specifies to be
     associated with the Work, unless such URI does not refer to
     the copyright notice or licensing information for the Work;
     and (iv), consistent with Section 3(b), in the case of an
     Adaptation, a credit identifying the use of the Work in the
     Adaptation (e.g., "French translation of the Work by Original
     Author," or "Screenplay based on original Work by Original
     Author"). The credit required by this Section 4 (b) may be
     implemented in any reasonable manner; provided, however, that
     in the case of a Adaptation or Collection, at a minimum such
     credit will appear, if a credit for all contributing authors
     of the Adaptation or Collection appears, then as part of these
     credits and in a manner at least as prominent as the credits
     for the other contributing authors. For the avoidance of
     doubt, You may only use the credit required by this Section
     for the purpose of attribution in the manner set out above
     and, by exercising Your rights under this License, You may not
     implicitly or explicitly assert or imply any connection with,
     sponsorship or endorsement by the Original Author, Licensor
     and/or Attribution Parties, as appropriate, of You or Your use
     of the Work, without the separate, express prior written
     permission of the Original Author, Licensor and/or
     Attribution Parties.

  c. Except as otherwise agreed in writing by the Licensor or as
     may be otherwise permitted by applicable law, if You
     Reproduce, Distribute or Publicly Perform the Work either by
     itself or as part of any Adaptations or Collections, You must
     not distort, mutilate, modify or take other derogatory action
     in relation to the Work which would be prejudicial to the
     Original Author's honor or reputation. Licensor agrees that in
     those jurisdictions (e.g. Japan), in which any exercise of the
     right granted in Section 3(b) of this License (the right to
     make Adaptations) would be deemed to be a distortion,
     mutilation, modification or other derogatory action
     prejudicial to the Original Author's honor and reputation, the
     Licensor will waive or not assert, as appropriate, this
     Section, to the fullest extent permitted by the applicable
     national law, to enable You to reasonably exercise Your right
     under Section 3(b) of this License (right to make Adaptations)
     but not otherwise.

5. Representations, Warranties and Disclaimer

UNLESS OTHERWISE MUTUALLY AGREED TO BY THE PARTIES IN WRITING,
```

```
LICENSOR OFFERS THE WORK AS-IS AND MAKES NO REPRESENTATIONS OR
WARRANTIES OF ANY KIND CONCERNING THE WORK, EXPRESS, IMPLIED,
STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, WARRANTIES OF
TITLE, MERCHANTIBILITY, FITNESS FOR A PARTICULAR PURPOSE,
NONINFRINGEMENT, OR THE ABSENCE OF LATENT OR OTHER DEFECTS,
ACCURACY, OR THE PRESENCE OF ABSENCE OF ERRORS, WHETHER OR NOT
DISCOVERABLE. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF
IMPLIED WARRANTIES, SO SUCH EXCLUSION MAY NOT APPLY TO YOU.

6. Limitation on Liability. EXCEPT TO THE EXTENT REQUIRED BY
   APPLICABLE LAW, IN NO EVENT WILL LICENSOR BE LIABLE TO YOU ON ANY
   LEGAL THEORY FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, PUNITIVE
   OR EXEMPLARY DAMAGES ARISING OUT OF THIS LICENSE OR THE USE OF
   THE WORK, EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY
   OF SUCH DAMAGES.

7. Termination

   a. This License and the rights granted hereunder will terminate
      automatically upon any breach by You of the terms of this
      License. Individuals or entities who have received Adaptations
      or Collections from You under this License, however, will not
      have their licenses terminated provided such individuals or
      entities remain in full compliance with those licenses.
      Sections 1, 2, 5, 6, 7, and 8 will survive any termination of
      this License.

   b. Subject to the above terms and conditions, the license granted
      here is perpetual (for the duration of the applicable
      copyright in the Work). Notwithstanding the above, Licensor
      reserves the right to release the Work under different license
      terms or to stop distributing the Work at any time; provided,
      however that any such election will not serve to withdraw this
      License (or any other license that has been, or is required to
      be, granted under the terms of this License), and this License
      will continue in full force and effect unless terminated as
      stated above.

8. Miscellaneous

   a. Each time You Distribute or Publicly Perform the Work or a
      Collection, the Licensor offers to the recipient a license to
      the Work on the same terms and conditions as the license
      granted to You under this License.

   b. Each time You Distribute or Publicly Perform an Adaptation,
      Licensor offers to the recipient a license to the original
      Work on the same terms and conditions as the license granted
      to You under this License.

   c. If any provision of this License is invalid or unenforceable
      under applicable law, it shall not affect the validity or
      enforceability of the remainder of the terms of this License,
      and without further action by the parties to this agreement,
      such provision shall be reformed to the minimum extent
      necessary to make such provision valid and enforceable.

   d. No term or provision of this License shall be deemed waived
      and no breach consented to unless such waiver or consent shall
      be in writing and signed by the party to be charged with such
      waiver or consent.

   e. This License constitutes the entire agreement between the
```

```
      parties with respect to the Work licensed here. There are no
      understandings, agreements or representations with respect to
      the Work not specified here. Licensor shall not be bound by
      any additional provisions that may appear in any communication
      from You. This License may not be modified without the mutual
      written agreement of the Licensor and You.

   f. The rights granted under, and the subject matter referenced,
      in this License were drafted utilizing the terminology of the
      Berne Convention for the Protection of Literary and Artistic
      Works (as amended on September 28, 1979), the Rome Convention
      of 1961, the WIPO Copyright Treaty of 1996, the WIPO
      Performances and Phonograms Treaty of 1996 and the Universal
      Copyright Convention (as revised on July 24, 1971). These
      rights and subject matter take effect in the relevant
      jurisdiction in which the License terms are sought to be
      enforced according to the corresponding provisions of the
      implementation of those treaty provisions in the applicable
      national law. If the standard suite of rights granted under
      applicable copyright law includes additional rights not
      granted under this License, such additional rights are deemed
      to be included in the License; this License is not intended to
      restrict the license of any rights under applicable law.

Creative Commons is not a party to this License, and makes no
warranty whatsoever in connection with the Work. Creative Commons
will not be liable to You or any party on any legal theory for any
damages whatsoever, including without limitation any general,
special, incidental or consequential damages arising in connection
to this license. Notwithstanding the foregoing two (2) sentences,
if Creative Commons has expressly identified itself as the Licensor
hereunder, it shall have all rights and obligations of Licensor.

Except for the limited purpose of indicating to the public that the
Work is licensed under the CCPL, Creative Commons does not authorize
the use by either party of the trademark "Creative Commons" or any
related trademark or logo of Creative Commons without the prior
written consent of Creative Commons. Any permitted use will be in
compliance with Creative Commons' then-current trademark usage
guidelines, as may be published on its website or otherwise made
available upon request from time to time. For the avoidance of
doubt, this trademark restriction does not form part of this
License.

Creative Commons may be contacted at http://creativecommons.org/.

======================================================================
```