PHPUnit PHP testing framework

CONCEPTS

A test is a public method named test* or marked #[Test] in a class extending TestCase whose file ends in Test.php.

phpunit.xml suites, source
↓ discover
Test classes *Test.php
↓ run
Result dots, F, E, R
↓ report
Reports coverage, JUnit

Without a phpunit.xml PHPUnit takes the path argument as the tests to run; with one, a bare call runs the whole config.

Coverage needs a driver: Xdebug (xdebug.mode=coverage) or PCOV. Without one, coverage options only warn.

RUN AND SELECT

vendor/bin/phpunit
Run the configured tests

vendor/bin/phpunit *<path>*
Run one file or directory

vendor/bin/phpunit --filter *<pattern>*
Run tests whose name matches

--testsuite *<name>*
Run one suite from the config

--group *<name>*
Run tests in a group

EXECUTION

--stop-on-failure
Stop at the first failure

--order-by *<order>*
Reorder tests

OUTPUT

--testdox
Show results as readable sentences

--display-deprecations
Show deprecation details

--display-warnings
Show warning details

COVERAGE

--coverage-text
Print a coverage table

--coverage-html *<dir>*
Write an HTML report

XDEBUG_MODE=coverage vendor/bin/phpunit --coverage-html *<dir>*
Coverage with Xdebug

CONFIGURATION

--generate-configuration
Interactively create phpunit.xml

-c *<file>*
Use another config file

Minimal phpunit.xml

```
<phpunit bootstrap="vendor/
autoload.php" colors="true"
         failOnWarning="true"
failOnRisky="true"
cacheDirectory=".phpunit.cache">
  <testsuites>
    <testsuite name=
"unit"><directory>tests/unit</
directory></testsuite>
  </testsuites>
  <source><include><directory
suffix=".php">src</directory></
include></source>
</phpunit>
```

WRITING TESTS

A basic test

```
use PHPUnit\Framework\TestCase;

final class CartTest extends
TestCase
{
    protected function setUp():
void { $this→cart = new Cart; }

    #[Test]
    public function adds_items():
void
    {
        $this→cart-
>add('apple');
        $this→assertCount(1,
$this→cart);
    }
}
```

assertSame($expected, $actual)
Same type and value (===)

assertEquals($expected, $actual)
Loosely equal (==)

WRITING TESTS (cont.)

assertTrue($cond) / assertFalse($cond)
Boolean checks

assertCount($n, $items)
Number of elements

assertInstanceOf(*<Class>*::class, $v)
Value is of a class

$this→expectException(*<Class>*::class)
Expect a thrown exception

setUp() / tearDown()
Run before and after each test

ATTRIBUTES

#[Test]
Mark a method as a test

#[DataProvider('<*method*>**')]**
Feed data sets from a static method

#[TestWith([<*args*>**])]**
Inline data set

#[Group('<*name*>**')]**
Put a test in a group

Data provider

```
#[DataProvider('sums')]
public function test_sum(int $a,
int $b, int $want): void
{
    $this→assertSame($want, $a +
$b);
}

public static function sums():
array
{
    return ['zero' ⇒ [0, 0, 0],
'one' ⇒ [1, 0, 1]];
}
```

TEST DOUBLES

$this->createStub(<*Class*>**::class)**
Canned-answer double, no expectations

$this->createMock(<*Class*>**::class)**
Double that can verify calls

$stub→method('**<*m*>**')->willReturn($v)
Fix a return value

$mock→expects($this->once())→method('<*m*>**')**
Require exactly one call