Measuring Code Coverage

Anna Filina
TESTING AND LEGACY EXPERT
@afilina afilina.com

Summary

Collect

Analyze

Best practices

Collect

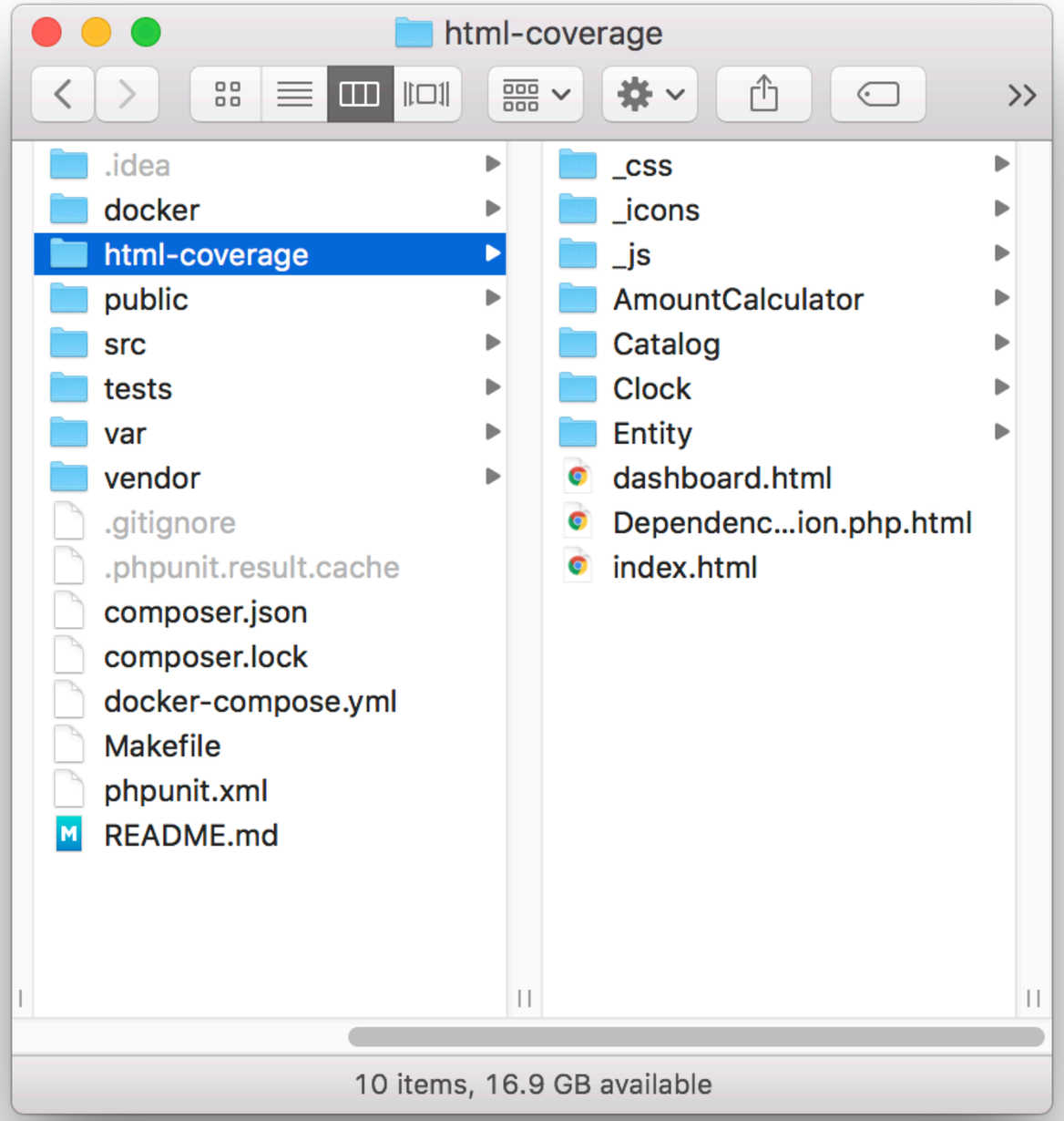
phpunit.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<phpunit bootstrap="tests/bootstrap.php"
         colors="true">

    <testsuites>
        <testsuite name="all">
            <directory suffix="Test.php">tests</directory>
       </testsuite>
    </testsuites>

    <coverage>
        <include>
            <directory suffix=".php">src</directory>
        </include>
        <report>
            <html outputDirectory="html-coverage" lowUpperBound="80"
highLowerBound="100"/>
        </report>
    </coverage>

</phpunit>
```


html-coverage
.idea
docker
html-coverage
public
src
tests
var
vendor
.gitignore
.phpunit.result.cache
composer.json
composer.lock
docker-compose.yml
Makefile
phpunit.xml
README.md
_css
_icons
_js
AmountCalculator
Catalog
Clock
Entity
dashboard.html
Dependenc...ion.php.html
index.html
10 items, 16.9 GB available

Analyze

tests/AmountCalculator/Operation/DiscountOperationTest.php

```php
/** @covers \App\AmountCalculator\Operation\DiscountOperation */
final class DiscountOperationTest extends TestCase
{
```

tests/Catalog/Value/DiscountTest.php

```php
<?php
namespace App\Tests\Catalog\Value;

use App\Catalog\Value\Amount;
use App\Catalog\Value\Discount;
use PHPUnit\Framework\TestCase;

/** @covers \App\Catalog\Value\Discount */
final class DiscountTest extends TestCase
{
    /** @test */
   public function getDiscountAmountForPrice_WithPercent_ReturnsPercentOfPrice(): void
    {
        $discount = Discount::fromPercent(0.20);
        self::assertEquals(
            new Amount(20),
            $discount->getDiscountAmountForPrice(new Amount(100))
        );
    }
}
```

tests/Catalog/Value/DiscountTest.php

```php
/** @test */
public function getDiscountAmountForPrice_WithAmount_ReturnsSameAmount(): void
{
    $discount = Discount::fromAmount(10);
    self::assertEquals(
        new Amount(10),
        $discount->getDiscountAmountForPrice(new Amount(500))
    );
}
```

Best Practices

~~Objective: increase code coverage to 100%~~

Objective: find bugs during initial dev

tests/Catalog/Value/DiscountTest.php

```php
/** @test */
public function getDiscountAmountForPrice_WithAmount_ReturnsSameAmount(): void
{
    $discount = Discount::fromAmount(100);
    $discount->getDiscountAmountForPrice(new Amount(100));
    self::assertTrue(true);
}
```

Cyclomatic Complexity

Execution Paths

src/Catalog/Value/Amount.php

```php
public function __construct(int $cents)
{
    if ($cents < 0) {
        throw AmountBelowZero::fromInt($cents);
    }
    $this->cents = $cents;
}
```

Up Next:
Writing Integration Tests