



企画：エンジニアの登壇を応援する会 / ワークショップチーム

制作：Tadouma, よしたく, FORTE, ariaki

# はじめに

---

## 注意事項

- PHP 7.1 以上が動作する PC が必要です
- PHP の基本構文を理解されている方が対象です
- PHPUnit 最新版のインストール が必要です（※セッション内で準備します）
- 以下をダウンロードする必要があります
  - ↓ <https://github.com/engineers-It/laracon>
- Wi-Fi 端末を希望の方はスタッフまでお声がけください

# はじめに

---

## ワークショップの流れ

- 説明時間と解答時間をそれぞれ設けます
  - 解答時間は自由に課題を解いてください
  - 課題を多く準備していますので、すべてを解く必要はありません
  - 解答例はダウンロードできますので、いつでも確認できます
- 質問は気軽にスタッフを呼び止めてください
- 解答時間中は、自由に途中入退室していただいて構いません

# はじめに

---

## 免責事項

このワークショップで配布される問題及び解答例は必ずしも最善なコードとは限りません。  
また入念な確認を行っておりますが、すべての環境における動作の正確性を保証しません。  
少しでも疑問に思った箇所がありましたら、必ず公式マニュアルをご参照ください。  
また、改善点などあれば是非スタッフまでご共有ください。

# はじめに

---

## 本日のスタッフ



@nekokotlin



@yoshitaku\_jp



@FORTEgp05



@kuwahara\_jsri



@chiroruxxxx



@ariaki4dev

## 制作／登壇

Tadouma

@nekokotlin

よしたく

@yoshitaku\_jp

FORTE

@FORTEgp05

## スライド作成

ariaki

@ariaki4dev

## 協力

chiro

@chiroruxxxx

k-kuwahara

@kuwahara\_jsri

tetsunosuke

@tetsunosuke

ねむ

@nemu1986

さっぴー川原

@sapi\_kawahara



# テストのメリット・デメリットを再確認

Tadouma | @nekokotlin

# はじめに

---

## 自己紹介

Tadouma | @nekokotlin

### 趣味



- 筋トレ
- 腹筋ローラー大好き
- 猫大好き
- 初心者(Laravel/Kotlin)



# はじめに

---

## このセクションで伝えること

- 1 テストの概要
- 2 自動テストのメリット／デメリット

# テストの概要

# テストの概要

---

## テストの定義

- プログラムが仕様どおりに動作するのを確認すること
- プログラミングの欠陥を見つける作業のこと
- テストコードを書くことで自動化できるもの

# テストの概要

---

## テストの種類

- 様々な観点で分類が可能
  - 工程での分類
  - 技法での分類
- 自動テストについて

# テストの概要

---

## テストの種類

- 工程での分類
  - 単体テスト
    - 関数、メソッド単位のテスト
  - 結合テスト
    - 単体テストで検証したプログラムを組み合わせで行うテスト
  - システムテスト
    - 実運用を想定したテスト

# テストの概要

---

## テストの種類

- 技法での分類
  - ホワイトボックステスト
    - 過程・処理が正しいか試すテスト
  - ブラックボックステスト
    - 出力結果が正しいか試すテスト
  - モンキーテスト
    - 対象箇所や操作手順を定めずに、思いつきで操作するテスト

# テストの概要

---

## 自動テストについて

- 自動テストフレームワーク
  - PHPUnit
    - いわゆるxUnit系
  - Laravelフレームワーク
    - PHPUnitが標準搭載

# 自動テストのメリット／デメリット



# 自動テストのメリット・デメリット

---

## メリット

- コードの品質向上、正確性向上
- コスト削減
- 保守性の向上
- 開発者のスキルアップ
- 仕様の把握や仕様漏れの減少

# 自動テストのメリット／デメリット

---

## メリット

- コードの品質向上、正確性向上
  - 自分のミスに気づく
  - 大規模プロジェクトや大量データを扱う際に役立つ
  - リグレッションの発見
- コスト削減
  - テスターに任せず自動テストでのカバー
  - テストカバレッジの提出がラクに
  - ロジックの正確性が担保され、レビュー時間も削減

# 自動テストのメリット／デメリット

---

## メリット

- 保守性の向上
  - リファクタリングを安心して行える
- 開発者のスキルアップ
  - テストを書くことによって実コードの書き方に工夫が生まれた
  - 初心者にとっては想定範囲を広げることにつながった
- 仕様の把握や仕様漏れの減少
  - 仕様の確認ができる
  - その結果仕様漏れや仕様の不具合に気づける

# 自動テストのメリット／デメリット

---

## デメリット

- 開発速度の低下
  - 実装の2倍、テストコード作成の時間がかかった
- 学習コスト
  - 学習コストがかかり、炎上案件では全面的な導入が難しかった
- 保守コスト
  - プログラムの規模が大きくなる/仕様変更によりテストコードの修正が必要



# 初めてのユニットテスト

よしたく | @yoshitaku\_jp


# はじめに

## 自己紹介



よしたく | @yoshitaku\_jp

### 趣味

 サッカー

 漫画

### アウトプット

 ブログ → <https://yoshitaku-jp.hatenablog.com/>

 #write\_blog\_every\_week

# はじめに

---

## このセクションで伝えること

1 PHPUnit の紹介

2 PHPUnit のインストール

32 PHPUnit の使い方

4 ワークショップ

# PHPUnit の紹介



# PHPUnit の紹介

---

## PHPUnit とは

- PHP 向けのユニットテストフレームワーク
  - xUnit の PHP 版
- 最新版は PHPUnit 8 (2019/2/16時点)
- PHP 7.2 - 7.4 に対応
- Laravel に標準搭載
- PHP のテストツールとしては一番使われている
  - phpspec
  - Atoum

# PHPUnit のインストール

# PHPUnit のインストール


## PHPUnit のインストール方法（ワークショップ用リポジトリ）

以下のコマンドで、必要なモジュールをダウンロードする

※ワークショップ用 GitHub リポジトリは初期設定済のため、インストールのみで良い

```
composer install
```

※インストールに成功すると、以下のようなファイル/ディレクトリが作成される

 vendor

# PHPUnit のインストール

## PHPUnit のインストール方法（初期ディレクトリ）

以下のコマンドで、composer を使用して最新版をインストールする

```
composer require phpunit/phpunit --dev
```

※インストールに成功すると、以下のようなファイル/ディレクトリが作成される



vendor



composer.json



composer.lock

# PHPUnit のインストール

---

## PHPUnit のインストール方法

以下のコマンドで、インストールされた PHPUnit のバージョンを確認する

```
vendor/bin/phpunit --version
```

※「PHPUnit 7.5.3 by Sebastian Bergmann and contributors.」のようにバージョンが表示されたら成功

# PHPUnit の使い方

# PHPUnit の使い方

---

## はじめてのテスト

以下のように「Hello, world」を返却するクラスのテストコードを書いてみよう

```
1 <?php
2 class Hello{
3     public function getMessage(){
4         return 'Hello, world';
5     }
6 }
```

# PHPUnit の使い方

## はじめてのテスト

前ページのテストケースは以下のように書く

```
1 <?
2 require_once 'Hello.php';
3 use PHPUnit\Framework\TestCase;
4
5 class HelloTest extends TestCase{
6     public function testGetMessage(){
7         $hello = new Hello;
8         $this->assertEquals('Hello, world', $hello->getMessage());
9     }
10 }
```



# PHPUnit の使い方

```
1 <?
2 require_once 'Hello.php';
3 use PHPUnit\Framework\TestCase;
4
5 class HelloTest extends TestCase{
6     public function testGetMessage(){
7         $hello = new Hello;
8         $this->assertEquals('Hello, world', $hello->getMessage());
9     }
10 }
```

TestCase を継承したクラスを作成する

test から始まるメソッドでテストを記述する

引数両者の値が**一致**する場合は**成功**としてテストケースを設定する

# PHPUnit の使い方

---

## はじめてのテスト

- Hello という名前のクラスのテストは、HelloTest という名前のクラスに記述します
- テストクラスは、ほとんどの場合 PHPUnit\Framework\TestCase を継承します
- テストメソッドは、test から始まるパブリックメソッドになります
- テストメソッドの中で assertEquals() のようなアサーションメソッドを記載します
- 困った場合はマニュアルを参照しましょう  
( <https://phpunit.readthedocs.io/ja/latest/index.html> )

# PHPUnit の使い方

## はじめてのテスト

以下のコマンドで、テストを実行します

```
vendor/bin/phpunit HelloTest
```

全てのテストが成功したら以下のように表示されるでしょう

```
PHPUnit 8.0.2 by Sebastian Bergmann and contributors.
```

```
.                                     1 / 1 (100%)
```

```
Time: 308 ms, Memory: 4.00MB
```

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

# PHPUnit の使い方

## はじめてのテスト

いずれかのテストに失敗した場合は以下のように表示されます

```
PHPUnit 8.0.2 by Sebastian Bergmann and contributors.
```

```
F                                     1 / 1 (100%)
```

```
Time: 327 ms, Memory: 4.00MB
```

```
There was 1 failure:
```

```
1) HelloTest::testGetMessage
```

```
Failed asserting that two strings are equal.
```

```
    : (中略)
```

```
FAILURES!
```

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

# PHPUnit の使い方

---

## アサーションメソッドの種類

- 一致
- 真偽値
- 文字列比較
- 数値比較
- その他

# PHPUnit の使い方

---

## アサーションメソッドの種類

- 一致

<b>assertEquals()</b>	引数の両者が等しくない場合にエラー
assertSame()	引数の両者が同型同値でない場合にエラー
assertNull()	引数の両者が Null でない場合にエラー

# PHPUnit の使い方

---

## アサーションメソッドの種類

- 真偽値

<code>assertTrue()</code>	引数が True 以外の場合にエラー
<code>assertFalse()</code>	引数が False 以外の場合にエラー

# PHPUnit の使い方

## アサーションメソッドの種類

- 文字列比較

<code>assertRegExp()</code>	引数が指定された正規表現に合致しない場合にエラー
<code>assertStringMatchesFormat()</code>	引数が指定された書式に合致しない場合にエラー
<code>assertStringStartsWith()</code>	引数が指定された文字列から開始されない場合にエラー
<code>assertStringEndsWith()</code>	引数が指定された文字列で終了しない場合にエラー



# PHPUnit の使い方

## アサーションメソッドの種類

- 数値比較

<code>assertGreaterThan()</code>	引数が指定された値より大きくない場合にエラー
<code>assertGreaterThanOrEqual()</code>	引数が指定された値以上でない場合にエラー
<code>assertLessThan()</code>	引数が指定された値より小さくない場合にエラー
<code>assertLessThanOrEqual()</code>	引数が指定された値以下でない場合にエラー

# PHPUnit の使い方

---

## アサーションメソッドの種類

- その他
  - 配列系
    - `assertArrayHasKey`, `assertArraySubset`, ...
  - クラス系
    - `assertClassHasAttribute`, ...
  - ファイル系
    - `assertFileExist`, `assertFileEquals`, `assertFileIsReadable`, `assertFileIsWritable`, ...
  - ディレクトリ系
    - `assertDirectoryExists`, `assertDirectoryIsReadable`, `assertDirectoryIsWritable`, ...

# PHPUnit の使い方

## assertEquals()

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

引数は以下のとおり

\$expected	期待値を記述する（※実際値と一致した場合にテストが通過する）
\$actual	実際値を記述する（※テストメソッドの実行結果など）
\$message	テスト失敗時に表示するメッセージ

# ワークショップ

# ワークショップ

---

## 注意点

- 課題をたくさん準備していますので、時間内に全問解答する必要はありません
- 不明点があれば気軽にスタッフまで声をかけてください
- ワークショップに必要な各ファイルは以下よりダウンロード可能です

↓ <https://github.com/engineers-It/laracon>

# ワークショップ

---

## 課題

1 Hello World をテストしよう

2 消費税計算をテストしよう

32 FizzBuzz をテストしよう

4 各月の日数をテストしよう

# ワークショップ

---

## 1. Hello World をテストしよう

- 「Hello World」を返却するクラスをテストしよう

 <https://github.com/engineers-It/laracon/blob/master/src/basic/WS1.php>

# ワークショップ

---

## 2. 消費税計算をテストしよう

- 単価と税率から税込金額を返却するクラスをテストしよう
  - 対象メソッドに引数がある場合にどう動くか確認しよう
  - 別の値を入れたときも動くか確認しよう

📄 <https://github.com/engineers-It/laracon/blob/master/src/basic/WS2.php>



# ワークショップ

---

## 3. FizzBuzz をテストしよう

- FizzBuzz クラスをテストしよう
  - 引数が 3 の場合にどんな動きをするか確認しよう
  - 引数が 5 の場合にどんな動きをするか確認しよう
  - 引数が 15 の場合にどんな動きをするか確認しよう
  - 引数が上記以外の場合にどんな動きをするか確認しよう



<https://github.com/engineers-It/laracon/blob/master/src/basic/WS3.php>

# ワークショップ

---

## 4. 各月の日数をテストしよう

- 月から日数を返却するクラス をテストしよう
  - 各月の日数が正しいか確認しよう
  - 存在しない月の場合の動作を確認し、例外に対処しよう

↓ <https://github.com/engineers-It/laracon/blob/master/src/basic/WS4.php>

課題の解答時間は

14:20 までです

- 自由に休憩して頂いて構いません
- 質問はお気軽にスタッフまで声をかけてください



# 本格的なテストのワークショップ

FORTE | @FORTEgp05

# はじめに

---

## 自己紹介

FORTE | @FORTEgp05

### アウトプット



aozora.fm → <https://fortegp05.github.io/aozorafm/>



技術書典6で『はじめる技術 つづける技術』を頒布予定

詳しくは #はじめる技術つづける技術 まで！

# はじめに

---

## このセクションで伝えること

1 課題の説明

2 ワークショップ

## 課題の説明

課題の解答時間は

15:50 までです

- 自由に休憩して頂いて構いません
- 質問はお気軽にスタッフまで声をかけてください



# エンジニアの登壇を応援する会

---



**connpass**

<https://engineers.connpass.com/>



**TECH PLAY**

<https://techplay.jp/community/engineers-lt>



**Slack**

<http://bit.ly/elt-slack>

# アンケート

---



[bit.ly / laracon-enquete](https://bit.ly/laracon-enquete)

エンジニアの登壇を応援する会

<http://portal.engineers-It.info/>