

Arrange-Act-Assert : Une structure de test reconnaissable au premier coup d'œil

Avoir une structure commune à tous les tests permet de gagner du temps. Il est recommandé de suivre le pattern Arrange-Act-Assert.

Les trois parties

Arrange

Dans la partie Arrange, on met en place le petit monde dans lequel le test va être exécuté. Si l'on imagine le test comme une histoire, c'est le point de départ de celle-ci.

Act

Dans la seconde partie, Act, on agit sur notre système. C'est à cet endroit que l'on déclenche le comportement que l'on veut vérifier.

Assert

Dans la troisième partie, Assert, on effectue les vérifications. On s'assure que le comportement s'est déroulé conformément à nos attentes. On peut dire que l'on vérifie l'état du monde à la fin de l'histoire.



Cette structure en 3 parties est similaire à la structure sous forme d'instruction Given, When, Then, que l'on trouve dans le langage Gherkin, utilisé dans les outils de la communauté Behaviour-Driven Design.

Exemple

Dans l'exemple suivant, nous créons un burrito non-végétarien, puisqu'il contient du thon, dans la partie Arrange du test. Dans un second temps, nous agissons. C'est la partie Act; ici, nous appelons la méthode **rendreVégétarien**. Et enfin, nous nous assurons que le burrito est bien végétarien et qu'il contient des œufs à la place du thon dans la partie Assert.

```

/**
 * @test
 */
public function
remplace_les_ingredients_non_végétariens_par_des_oeufs_pour_rendre_le_burrito_végétarien(): void {

    // Arrange
    $burrito = new Burrito(['thon', 'salade', 'oignon', 'avocat', 'riz']);

    // Act
    $burrito->rendreVégétarien();

    // Assert
    $this->assertTrue($burrito->estVégétarien());
    $this->assertEquals(['oeuf', 'salade', 'oignon', 'avocat', 'riz'], $burrito->
    ingrédients());
}

```



Pour prendre l'habitude d'utiliser le pattern Arrange-Act-Assert, il peut être intéressant d'utiliser des commentaires pour bien délimiter les 3 parties.

Variation

Parfois, lorsque le comportement testé est assez simple, il n'est pas nécessaire d'utiliser les 3 parties. C'est le cas lorsqu'il n'y a pas vraiment d'action - par exemple lorsqu'on interroge le sujet du test sur son état - ou quand le test peut être tout aussi lisible sans utiliser de variable intermédiaire.

```

/**
 * @test
 */
public function nest_pas_végétarien_sil_contient_du_poisson(): void {

    // Arrange
    $burrito = new Burrito(['thon']);

    // Act & Assert
    $this->assertFalse($burrito->estVégétarien());
}

```