Composer

Achref El Mouelhi

Plan

Composer

Dans un projet informatique

- on a souvent un (ou des) bloc(s) de code se répète(nt)
- on essaye donc de le factoriser
  - soit tout mettre dans une fonction et appeler cette fonction
  - soit tout mettre dans un fichier et importer ce fichier

Composer

Exemple

- En PHP, on peut utiliser des fonctions et/ou des classes définies dans d'autres fichiers en faisant `require 'file.php';` ou include 'file.php' ;
- En Java, on peut faire la même chose en écrivant `import java.math.*;`

Là, on parle de dépendances

Il s'agit de toutes les relations avec les librairies dont notre projet dépend pour fonctionner, correctement et sans erreurs.

Composer

Composer est un gestionnaire de dépendance pour PHP

- Le code de ces librairies peut changer avec le temps : donc Composer doit les maintenir à jour dans mon projet
- Ces librairies peuvent dépendre d'autres : donc Composer doit s'en charger de gérer ses sous-dépendances
- il dispose d'un autoloader qui inclut les librairies dont on a besoin sans utiliser `require` ou `include`

Installation

Téléchargement et installation

https ://getcomposer.org/download/

Installation

Téléchargement et installation

https ://getcomposer.org/download/

on peut utiliser composer comme commande dans une ligne de commande

Des librairies PHP

Où je peux les trouver ?

https ://packagist.org/

Composer

Exemple

Supposant que je dispose d'une page PHP contenant l'affichage d'un texte écrit avec des balises Markdown (** et * sont des balises Markdown)

- ** pour un texte en gras
- * pour un texte en italic

```
//testcomposer.php
<?php
        echo "bonjour **tout le monde** ceci est un
         contenu *Markdown*";
?>
```

Composer

Étapes à suivre

- Chercher une librairie qui traite les Markdown dans **Packagist**
- Choisir une s'il y en a plusieurs
- Lire la description et comprendre comment elle fonctionne
- L'intégrer dans mon projet

Composer

J'ai choisis une librairie comment je fais pour l'intégrer ?

- Dans l'invite de commandes, je me situe à l'intérieur de mon projet
- Ensuite je saisis `composer init`
- Plusieurs questions seront posées :
  - ...
  - `yes` pour la gestion de dépendance
  - je précise le nom du package (`michelf/php-markdown` pour mon exemple)
  - puis j'indique la version que j'ai trouvé dans packagist : (1.7.0 la dernière version que j'ai trouvée)
  - répondre `yes` à la confirmation pour le fichier .json
- Enfin j'écris `composer install` pour télécharger les différentes dépendances

Composer

Pour l'utiliser, il faut lire la description

- Then include Composer's generated `vendor/autoload.php` to enable autoloading : `require 'vendor/autoload.php';`
- use Michelf\SMarkdown ;
- $my_html = Markdown : :defaultTransform($my_text) ;

Composer

```
//testcomposer.php
<?php
    require "vendor/autoload.php";
    echo "bonjour␣**tout␣le␣monde**␣ceci␣est␣un␣
     contenu␣*Markdown*";
?>
```

Composer

```
//testcomposer.php
<?php
    require "vendor/autoload.php";
    use Michelf\Markdown;
    echo Markdown::defaultTransform("bonjour␣**tout␣
     le␣monde**␣ceci␣est␣un␣contenu␣*Markdown*");
?>
```

Et donc là ça affiche : bonjour **tout le monde** ceci est un contenu *Markdown*

Composer

Si je change de librairie

- Je dois relire la description et l'appliquer
- Au lieu de faire `composer install`, il faut faire `composer update`

Ouvrir le composer.json

Pour charger plusieurs librairies PHP avec Composer, on peut tout indiquer dans la partie `require` du `composer.json`

```
//composer.json
{
    "name": "Utilisateur/testcomposer",
    "require": {
        //ici on ajoute les librairies ainsi que leur versions
        "michelf/php-markdown": "^1.7",
        "phpunit/phpunit": "^4.8"
    },
    "authors": [
        {
            "name": "elmouelhi",
            "email": "elmouelhi.achref@gmail.com"
        }
    ]
}
```

N'oublions pas de faire `composer update` après modification

Effectuer des tests unitaires avec `PHPunit`

Le rôle des tests unitaires est de s'assurer que les différents fragments de notre projet fonctionnent correctement et donnent les résultats attendus.

Comment faire ?

- On commence par se positionner dans notre projet
- Ensuite télécharger `PHPUnit` avec Composer
- Et enfin `composer install`

Ensuite

Considérant la classe suivante que nous voudrons tester sa méthode division

```
//Calcul.php
<?php
        class Calcul{
                private $_nbr1;
                private $_nbr2;
                public function __construct($x,$y){
                        $this->_nbr1 = $x;
                        $this->_nbr2 = $y;
                }
                public function division(){
                        return ($this->_nbr1)/($this->_nbr2);
                }
        }
?>
```

Ensuite

Nous créons aussi une classe dont le nom se termine par Test : nous l'appellerons CalculTest.php.

```
//CalculTest.php
<?php
        include "Calcul.php"; // inclure la classe Calcul
        class CalculTest extends PHPUnit_Framework_TestCase{
        // etendre la classe PHPUnit_Framework_TestCase
                public function testDivision(){
                        $c1=new Calcul(10,2);
                        $this->assertEquals(5,$c1->division());
                        $c2=new Calcul(12,2);
                        $this->assertEquals(6,$c2->division());
                }
        }
?>
```

Donc on va tester cette méthode avec deux couples de valeurs différents et on indique chaque ce qu'on attend comme retour (avec la méthode `assertEquals()`.

Effectuer des tests unitaires avec `PHPunit`

Et après

- On place `Calcul.php` et CalculTest.php dans un même dossier `Fichier` (par exemple) qui se situe directement sous mon projet
- Dans l'invite de commande, on fait un `cd vendor/bin`
- Enfin exécuter `phpunit ../../Fichier`

Le résultat :

```
Time: 1.39 seconds, Memory: 3.00MB
OK (1 test, 2 assertions)
```

Remarques

Et après

- Le nom des classes de tests doivent se terminer par **T**est (par exemple CalculTest)
- Les méthodes de cette classe doivent contenir le mot test
- Et elle doivent être publiques

Autres méthodes pour les assertions

Exemple

- `assertArraySubset()`
- `assertClassHasAttribute()`
- `assertClassHasStaticAttribute()`
- `assertContains()`
- `assertCount()`
- `assertDirectoryExists()`
- `assertFalse()`
- La liste est longue. Voir :

  `https://phpunit.de/manual/current/en/appendixes.assertions.html`