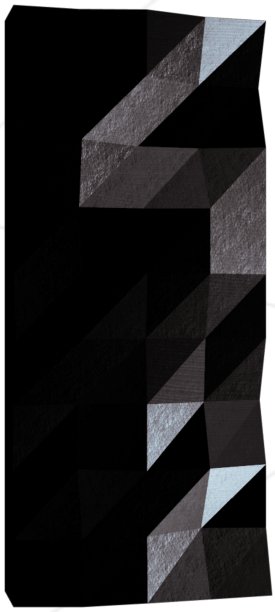

Formation PHP Symfony - 1

Composer

Résumé: Lors de cette quatrième 42 journée de formation, vous allez découvrir le Composer et la façon dont vous pouvez l'utiliser dans vos applications.

Version: 1

Table des matières

Chapitre I

Préambule

Lorsque vous écrivez des applications en PHP, vous aurez probablement trouver que vous devez réinventer la roue à chaque fois que vous devez fournir une fonctionnalité commune comme la gestion des utilisateurs, la gestion de la base de données, etc. C'est là que le Composer peut vous sauver. Le git pullComposer est gestionnaire de dépendances qui va intégrer toutes les libraries nécessaires et dépendances pour les gérer au même endroit, dans votre projet. Ce type de gestion de dépendances dans un projet n'est pas un concept nouveau. En fait, beaucoup de fonctionnalités du Composer sont inspirées du NPM de Node.js ou encore du Bundler de Ruby. En comparaison avec d'autres gestionnaires de packages PHP, comme PEAR par exemple, le Composer va vous permettre d'installer les packages sur la base de chaque projet et pas au niveau global de vôtre installation PHP.

Chapitre II

Règles communes

- Votre projet doit être réalisé dans une machine virtuelle.
- Votre machine virtuelle doit avoir tout les logiciels necessaire pour réaliser votre projet. Ces logiciels doivent être configurés et installés.
- Vous êtes libre sur le choix du systmème d'exploitation à utiliser pour votre machine virtuelle.
- Vous devez pouvoir utiliser votre machine virtuelle depuis un ordinateur en cluster.
- Vous devez utiliser un dossier partagé entre votre machine virtuelle et votre machine hote.
- Lors de vos évaluations vous allez utiliser ce dossier partager avec votre dépot de rendu.
- Vos fonctions de doivent pas s'arrêter de manière inattendue (segmentation fault, bus error, double free, etc) mis à part dans le cas d'un comportement indéfini. Si cela arrive, votre projet sera considéré non fonctionnel et vous aurez 0 au projet.
- Nous vous recommandons de créer des programmes de test pour votre projet, bien que ce travail **ne sera pas rendu ni noté**. Cela vous donnera une chance de tester facilement votre travail ainsi que celui de vos pairs.
- Vous devez rendre votre travail sur le git qui vous est assigné. Seul le travail déposé sur git sera évalué. Si Deepthought doit corriger votre travail, cela sera fait à la fin des peer-evaluations. Si une erreur se produit pendant l'évaluation Deepthought, celle-ci s'arrête.

Chapitre III

Consignes spécifiques à cette journée

- Si aucune information contraire n'est explicitement présente, vous devez assumer les versions de langages suivantes :
  - `PHP - Symfony version LTS`
  - `HTML 5`
  - `CSS 3`

Chapitre IV

Exercice 00

	Exercice : 00
Exercice 00 : Installer le Composer au niveau global	
Dossier de rendu : $ex00/$	
Fichiers à rendre : `Fichiers et répertoires de vôtre application`	
Fonctions Autorisées : `Toutes méthodes autorisées`	

Installez le **`Composer`** au niveau global, depuis le terminal.

Cet exercice est obligatoire. Afin de pouvoir résoudre tous les autres exercices vous devez absolument avoir résolu celui-ci.

Chapitre V

Exercice 01

	Exercice : 01
Exercice 01 : La spécification des versions	
Dossier de rendu : $ex01/$	
Fichiers à rendre : `Fichiers et répertoires de vôtre application`	
Fonctions Autorisées : `Toutes méthodes autorisées`	

Cet exercice comprend plusieurs tâches. Pour chacune d'entre elles vous devez créer un nouveau fichier composer.

Installez les versions suivantes du package **Monolog** en utilisant le **Composer**.

1. version supérieure à 2.3.0
2. version supérieure à 2.2.0 et inférieure ou égale à 2.3.5
3. version entre 2.1.0 et 2.2.0
4. version supérieure ou égale à 2.0.0 et inférieure à 2.0.2
5. version supérieure à 2.0.0 et inférieure à 2.3.5

Astuce : `Afin de résoudre les tâches, vous devrez utiliser le commande` **`composer install`**.

Chapitre VI

Exercice 02

	Exercice : 02
Exercice 02 : Development requirement	
Dossier de rendu : $ex02/$	
Fichiers à rendre : `Fichiers et répertoires de vôtre application`	
Fonctions Autorisées : `Toutes méthodes autorisées`	

Installez la version LTS du package PHPUnit en tant que development requirement.

Chapitre VII

Exercice 03

	Exercice : 03
Exercice 03 : Composer install vs. composer update	
Dossier de rendu : $ex03/$	
Fichiers à rendre : `composer.json, composer.lock, fichiers et répertoires de vôtre application`	
Fonctions Autorisées : `Toutes méthodes autorisées`	

Cet exercice contient deux tâches principales. Pour chacune d'entre elles vous devez créer un sous-répertoire contenant les fichiers du `composer`.
Avec les fichiers **composer.json** et **composer.lock** dans vôtre répertoire courant, vous devrez exécuter les commandes **composer install** et **composer update**. Observez les fichiers créés/mis à jour et quelle est la différence entre `install` et `update`.
Marche à suivre :

1. Copier **`composer.json`** et **`composer.lock`** dans vôtre répertoire courant
2. Exécuter la commande **composer install**
3. Exécuter la commande **composer update**

Chapitre VIII

Rendu et peer-évaluation

Rendez votre travail dans votre dépôt `Git` comme d'habitude. Seul le travail présent dans votre dépôt sera évalué en soutenance. Vérifiez bien les noms de vos dossiers et de vos fichiers afin que ces derniers soient conformes aux demandes du sujet.

`L'évaluation se déroulera sur l'ordinateur du groupe évalué.`