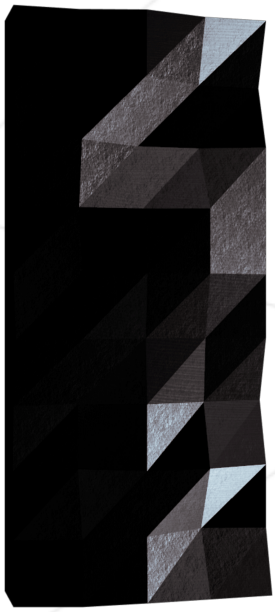

Formation PHP - Symfony

D08 - Mise en prod

Résumé: Toute application doit être développée , certes , mais son déploiement est une étape importante afin de la transposer dans un environnement de production..

Table des matières

Chapitre I

Préambule

The void type, in several programming languages derived from C and Algol68, is the type for the result of a function that returns normally, but does not provide a result value to its caller. Usually such functions are called for their side effects, such as performing some task or writing to their output parameters. The usage of the void type in such context is comparable to procedures in Pascal and syntactic constructs which define subroutines in Visual Basic. It is also similar to the unit type used in functional programming languages and type theory.

Chapitre II

Consignes

Sauf contradiction explicite, les consignes suivantes seront valables pour tous les jours de cette Piscine.

- Seul ce sujet sert de référence : ne vous fiez pas aux bruits de couloir.
- Le sujet peut changer jusqu'à une heure avant le rendu.
- Les exercices sont très précisément ordonnés du plus simple au plus complexe. En aucun cas nous ne porterons attention ni ne prendrons en compte un exercice complexe si un exercice plus simple n'est pas parfaitement réussi.
- Attention aux droits de vos fichiers et de vos répertoires.
- Vous devez suivre la procédure de rendu pour tous vos exercices. L'url de votre dépot `GIT` pour cette journée est disponible sur votre intranet.
- Vos exercices seront évalués par vos camarades de Piscine.
- En plus de vos camarades, vous pouvez être évalués par un programme appelé la Moulinette. La Moulinette est très stricte dans sa notation car elle est totalement automatisée. Il est donc impossible de discuter de sa note avec elle. Soyez d'une rigueur irréprochable pour éviter les mauvaises surprises.
- Les exercices shell doivent s'éxcuter avec `/bin/sh`.
- Vous ne devez laisser aucun autre fichier que ceux explicitement specifiés par les énoncés des exercices dans votre dépot de rendu.
- Vous avez une question ? Demandez à votre voisin de droite. Sinon, essayez avec votre voisin de gauche.
- Toutes les réponses à vos questions techniques se trouvent dans les `man` ou sur Internet.
- Pensez à discuter sur le forum Piscine de votre Intra et sur Slack !
- Lisez attentivement les exemples car ils peuvent vous permettre d'identifier un travail à réaliser qui n'est pas précisé dans le sujet à première vue.
- Réfléchissez. Par pitié, par Thor, par Odin !

Chapitre III

Règles spécifiques de la journée

- Chaque exercice doit être rendu dans un répertoire différent.
- Pour la résolution des exercices vous devez respecter des demandes des énoncés. Vos fichiers doivent être nommés correctement. Si une de ces conditions n'est pas remplies, l'exercice ne reçoit pas de points et dont être considéré incomplet.

Chapitre IV

Exercice 00

	Exercice : 00
Exercice 00 : Créer la machine virtuelle	
Dossier de rendu : *ex*00/	
Fichiers à rendre : `Fichiers et répertoires de votre application`	
Fonctions Autorisées : `Toutes méthodes permises`	

Pour cet exercice vous devez créer une machine virtuelle (VM), en utilisant Vagrant, afin de simuler un serveur de production. La VM doit contenir une installation ***"ubuntu/trusty64"***. La mémoire allouée doit être paramétrée à ***1024***.

Afin d'avoir un serveur fonctionnel, vous devez installer les composants suivants sur votre VM :

- install *curl*
- install *php5*
- install *apache2*
- install *mysql*
- install *git*
- install *composer*

Ces commandes seront incluses dans un fichier provision qui sera passé à Vagrant.

Note :
Le nom du fichier provision devrait être **create_server.sh**.
Afin de prévenir les avertissements de type *"stdin : is not a tty"*, ajoutez les lignes suivantes dans votre fichier provision :

```
config.vm.provision "fix-no-tty", type: "shell" do |s|
   s.privileged = false
   s.inline = "sudo sed -i '/tty/!s/mesg n/tty -s \\&\\& mesg n/' /root/.profile"
```

Chapitre V

Exercice 01

	Exercice : 01
Exercice 01 : Ajouter deployer.php	
Dossier de rendu : $ex01/$	
Fichiers à rendre : `Fichiers et répertoires de votre application`	
Fonctions Autorisées : `Toutes méthodes permises`	

Cet exercice est dédié à la partie ***déploiement*** proprement-dite. Pour ce faire, nous allons utiliser **deployer.php**, une bibliothèque pour déploiement automatique sur serveur de production. Ayant le serveur de production configuré dans l'exercice précédent, nous devons y ajouter notre projet.
Ajouter le fichier **deploy.php** dans le projet courant. Ce fichier sera configuré pour déployer le **Symfony-Repo** (https://github.com/symfony/symfony-standard.git) sur votre VM Vagrant. Le nom du serveur sera **production**. Le chemin de votre projet sur le serveur sera **/var/www/production** et la branche **master** sera déployée ici.

Pour accéder à votre serveur de production, vous devez configurer un VirtualHost pour votre VM vagrant. Cette action pourrait être ajoutée dans le fichier provision fourni à votre instance Vagrant.

Note :
La première étape de l'exercice consiste à copier le fichier Vagrant de l'exercice spécédent afin de créer votre serveur de production.
Afin d'obtenir des configurations, utilisez la commande **"vagrant ssh-config"**.

Chapitre VI

Exercice 02

	Exercice : 02
Exercice 02 : Ajouter une tache au déployeur	
Dossier de rendu : $ex02/$	
Fichiers à rendre : `Fichiers et répertoires de votre application`	
Fonctions Autorisées : `Toutes méthodes permises`	

Créer une nouvelle tache dans le fichier **deploy.php**tache qui mettra à jour un fichier **VERSION.txt** avec le dernier hash commit. Le nom de la tache sera **"update :hash :version"**.