Proposition de Correction (questions ouvertes)

DS 2017-2018 4ETI, TLI avancé, 1ère session

Avertissement

Les réponses ne sont pas toutes rédigées, vous avez cependant les principes de ce qui était attendu en réponse

Exercices

Compréhension technique (2.1)

Question 21 :

Voir le cours. Répondre en 2/3 phrases maximum.

Pour les exemples :

1. utilisation de la technique de l'erreur 403 ; détailler la mise en oeuvre
2. utilisation du module rewrite d'apache ; détailler la mise en oeuvre

Question 22 :

.htaccess

Question 23 :

A partir de la règle du fichier .htaccess :

RewriteEngine on

RewriteRule ^([a - zA - Z]*) /?([a - zA - Z]*) ?/?([a - zA - Z0 -9]*) ?/? $ index . php ?

controleur = $1 & action = $2 & id = $3 [NC , L]

En entrée :

1. chaine de caractères majuscule et minuscule, longueur nulle possible
2. éventuellement « / »
3. éventuellement chaine de caractères majuscule et minuscule, longueur nulle possible
4. éventuellement « / »
5. chaine de caractères majuscule et minuscule ET chiffres de 0 à 9, longueur nulle possible
6. éventuellement « / »

En sortie :

index.php?controleur=chaine de caractères majuscule et minuscule, longueur nulle possible&action=éventuellement chaine de caractères majuscule et minuscule, longueur nulle

possible&id=chaine de caractères majuscule et minuscule ET chiffres de 0 à 9, longueur nulle possible

Exemple : calcul/factorielle/123 → index.php?controleur=calcul&action=factorielle&id=123

Question 24 :

L'auteur n'utilise pas un moteur de template avec un langage propre.

Il s'agit d'inclusions de fichiers PHP, les variables du template sont des variables PHP assignées par le programme principal.

Il vaudrait mieux utiliser un vrai moteur de template avec un langage dédié (comme smarty par exemple).

Question 25 :

Il s'agit d'une méthode statique. Elle permet de récupérer les options de configuration via la classe Configuration qui maintient probablement une map associative clef/valeur.

Mise en œuvre (2.2)

Question 26 :

Il faut créer un objet connexion en PDO pour la variable $dbh

Question 27 :

Oui : l'utilisation de PDO devrait normalement empêcher toute injection

Question 28 :

Non :

1. on utilise directement une variable globale $_GET qui n'a pas été vérifiée auparavant.
2. le code n'est pas encapsulé dans une fonction ou une classe
3. le code ne permet pas de sauvegarder les résultats dans un objet ou une variable, il affiche simplement le résultat de la requête à l'écran.

Question 29 :

cf. bonnes pratiques vues en TP

Synthèse

Question 30 :

On a déjà vu :

1. pas d'utilisation d'un vrai moteur de template
2. pas d'encapsulation
3. pas de vérification des données entrantes (utilisation immédiate de $_GET)

Pour la réponse, bonnes pratiques à détailler :

1. protéger en lecture/écriture la base de données : avoir un utilisateur de base de données pour les consultations et un autre pour les modifications/écriture
2. utiliser un seul point d'entrée pour le site
3. structurer les scripts php en librairies, avec une hiérarchie de répertoire par exemple, voir des packages
4. utiliser un système de protection par droits (chmod) interdisant la lecture et l'écriture à tout processus n'étant pas lancé par www-data
5. ne pas utiliser des mots de passe en clair dans la base de données, mais se servir d'un hachage, voir d'un grain individuel pour chaque utilisateur : voir https://apprendre-php.com/tutoriels/tutoriel-35-scuriser-les-mots-de-passe-avec-les-hashs-et-les-salts.html ou https://patouche.github.io/2015/03/21/stocker-des-mots-de-passe/
6. etc.