UE L313 - EF

Hugo Ligneres

08/01/2025

Partie 1

Question 1

L'acronyme MVC signifie "Modèle - Vue - Contrôleur". Cette architecture, utilisée dans de nombreux frameworks, permet de séparer les rôles des fichiers d'un projet de manière très claire, en trois catégories :

- **Le contrôleur** : il gère la logique au sein de l'application. Il sert aussi de passerelle entre la vue et le modèle, avec qui il interagit directement ;
- **La vue** : responsable de l'affichage du contenu renvoyé par le contrôleur sur le navigateur de l'utilisateur ;
- **Le modèle** : son rôle est de gérer les requêtes auprès de la base de données de l'application, et de récupérer les données obtenues.

Prenons un exemple simple d'une page web qui simule une bibliothèque. Un utilisateur demande à consulter les livres écrit par un auteur X. Voici comment l'architecture MVC procède :

1. La vue va récupérer la requête formulée par l'utilisateur, pour l'envoyer au contrôleur ;
2. Le contrôleur va analyser la requête formulée, pour la transmettre auprès du modèle ;
3. le modèle va interroger la base de données pour récupérer les informations voulues par l'utilisateur ;
4. Une fois les données obtenues, le modèle va transmettre ces informations auprès du contrôleur ;
5. Le contrôleur va transformer les données pour les retourner dans une forme facile à utiliser par la vue, par exemple sous forme de liste ;
6. Le contrôleur transmet cette liste auprès de la vue ;
7. Enfin, la vue s'occupe d'afficher la liste des livres de l'auteur X sur le navigateur de l'utilisateur.

Question 2

Cette architecture permet de faciliter le développement d'une application ou d'un site web relativement complexe. Elle permet également de rendre la maintenabilité d'un projet plutôt aisée, même si ce dernier augmente en taille et en complexité au fil du temps. Elle trouve aussi son utilité dans les projets sur lequels plusieurs personnes travaillent dessus, notamment si chaque personne à un rôle différent. Par exemple, la personne responsable de la partie frontend d'un projet travaillera principalement sur les fichiers de la vue. D'une manière similaire, le développeur backend travaillera surtout sur les fichiers du contrôleur et/ou du modèle.

Question 3

Composer est un gestionnaire de dépendances pour PHP. On peut le voir comme une grande base de données qui contient tous les frameworks et bibliothèques utilisables par les développeurs PHP.
L'un des intérêts de Composer, c'est son fichier *composer.json*. Il permet d'ajouter facilement et rapidement toutes les bibliothèques et autres outils à un projet PHP. Pour cela, on ajoute dans le fichier *composer.json* ce que l'on souhaite inclure au projet.
Une fois que cela est fait, on a juste à entrer *composer install* dans le terminal, à la racine du projet, pour que tout soit installer directement. Bien entendu, si l'on souhaite ajouter d'autres éléments à projet dans le futur, il suffit juste de modifier le fichier *composer.json* et d'entrer à

nouveau la commande *composer install*. Ce procédé permet deux choses :

— Il évite au développeur d'avoir à installer manuellement, un par un, les dépendances souhaitées à un projet
— Cela facilite la maintenabilité du projet, car toutes les dépendances du projet sont consultables dans un seul et même fichier. Ainsi, pas besoin de fouiller dans plusieurs fichiers pour savoir ce qui est installé ou non.

Question 4

Un flux RSS est un format de données qui permet de stocker, d'afficher et de diffuser des informations. Cet outil permet à des utilisateurs de récupérer et d'extraire facilement les informations envoyés par ce flux, par exemple en utilisant un agrégateur.

Le RSS est principalement composé d'un ou plusieurs fichiers XML, qui sont utilisés pour ordonner et catégoriser les données dans des balises. C'.

Question 5

Question non-traitée.

Partie 2

Question 1

Ces fichiers seront tous à installér dans le répertoire *web/* d'un projet développé avec le framework Silex. Ce répertoire correspond à la partie publique du site, c'est donc ici que doivent se trouver les éléments utilisés par le navigateur, tels que les images ou le style d'une page.

Question 2

La correspondance entre les URL et les fonctions PHP exécutées se situe dans le fichier *routes.php*, lui-même situé dans le répertoire *app/* d'un projet Silex.
Dans ce fichier *routes.php*, on associe la variable *$app* à une méthode. Cette méthode permet d'associer un URL à un chemin qui mène à une fonction dans un fichier, voire même le fichier entier, situé.e le plus souvent dans le répertoire *Controller/* du projet.
La variable *$app* provient du fichier *app.php*, lui aussi situé dans le répertoire *app/*.
On peut donc en déduire que les fichiers concernés pour assurer la correspondance entre les URL et les fonctions PHP sont les fichier *routes.php* et *app.php*, situés dans le même répertoire.

Question 3

Le code de cette capture d'écran se trouve dans le contrôleur. On peut déduire cela grâce aux deux lignes dans la fonction *indexAction()*.

Dans la première ligne, on cherche tous les liens d'une base de données, en utilisant le DAO de l'application. Hors, le DAO (Data Access Object) est le système utilisé par le modèle pour interogger la base de données. Donc, dans cette première ligne, on communique avec le modèle.

Dans la deuxième ligne, on retourne les liens trouvés vers le fichier twig prévu à cet effet. C'est la vue qui s'occupe de gérer l'affichage des données dans les fichiers twig. Donc, à cette ligne, on communique avec la vue pour indiquer quelles données afficher.

Hors, le seul élément capable de communiquer directement avec le modèle et la vue, c'est le contrôleur. Ce code se situe donc dans le contrôleur du projet.

Question 4

Question non-traitée.

Question 5

Avant de procéder à l'installation du projet, il faut déterminer quels outils seront utilisés pendant la phase de développement. Surtout, il faut déterminer quel outil utiliser pour simuler un serveur. On peut utiliser un serveur local, tel que LAMP ou WAMP, ou une solution de conteneurisation, comme Docker ou Kubernetes. Une fois que cette question est réglée, on peut passer à l'installation du projet sur les ordinateurs.

D'abord, on doit clôner les fichiers du projet, situés sur le dépôt Git, sur les ordinateurs. Pour cela, on utilise la commande *git clone <url du dépôt>*. Avec cette commande, les fichiers du dépôt sont automatiquement installés.

Ensuite, dans la très grande majorité des projets Silex, il y a un nombre de dépendances à installer pour que le projet fonctionne.
Pour cela, on s'assure que le fichier *composer.json* du projet soit bien situé à la racine du projet, et que les dépendances nécessaires sont listées dans ce fichier. Une fois cette vérification faite, on entre la commande *composer install* depuis la racine du projet. Cela devrait installer toutes les dépendances, et rendre le projet fonctionnel, comme sur l'ordinateur de la personne qui a déposé le code source sur le dépôt Git.

La dernière étape d'installation est de configurer la connexion à la base de données du projet : le port, le mot de passe, le nom d'utilisateur, etc. Pour cela, on doit créer un fichier *.env.local*, qui contient toutes les informations de configuration de la base de données.

Une fois ces étapes passées, le développement de l'application peut démarrer.

Enfin, pour faciliter le travail en asynchrone, il est préfèrable d'utiliser un dépôt Git sur lequel on ajoutera les modifications apportées au projet. Sauf si l'on doit envoyer ces modifications directement sur le dépôt originel.
Attention cependant, certains dossiers et fichiers ne doivent pas apparaître sur le dépôt, comme le répertoire vendor ou le fichier de configuration de la base de données. Le premier car il est trop volumineux, et le deuxième car il est trop sensible. Pour cela, on utilise le fichier *.gitignore*. On y inscrit dedans les fichiers et dossiers à ignorer, et lorsque l'on veut ajouter des fichiers modifiés au dépôt, les fichiers présents dans le fichier *.gitignore* ne seront pas ajoutés.