L3 MI-AW : Projet PHP

Réalisation MVC avec Front Controller

GEORGES Lucas

LEBOC Philippe

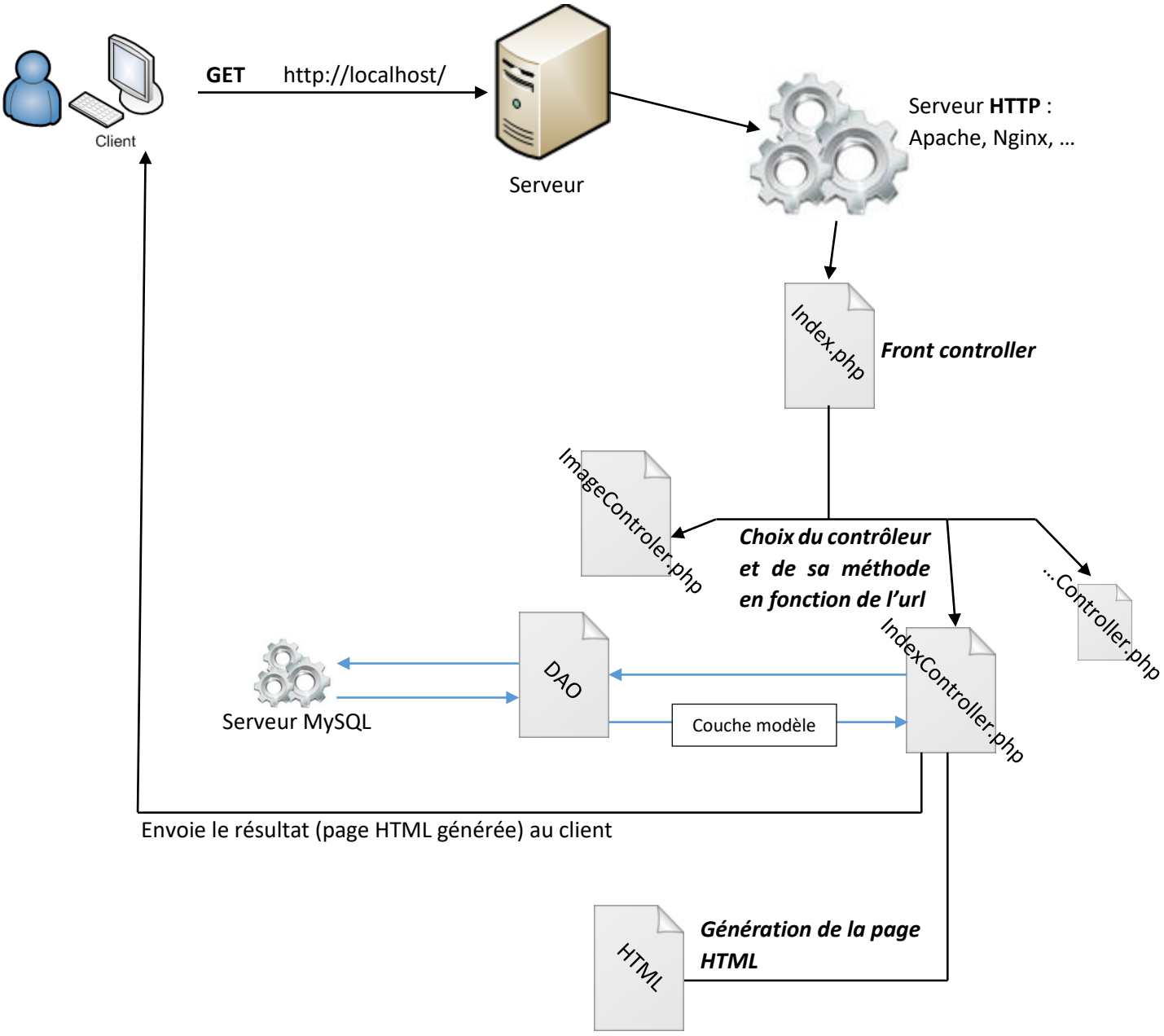
Le 9 novembre 2016

Table des matières

Description du projet

Le but de ce projet est de manipuler le langage PHP afin de réaliser un site internet servant de base à la compréhension du modèle MVC (Modèle-Vue-Contrôleur) qui a pour but d'apporter une organisation structurée du code. Dans ce projet, nous avons voulu pousser le modèle MVC un plus loin en effectuant une réelle séparation des langages. Par exemple, nos vues ne possèdent pas de code PHP destiné à afficher des variables.

Ci-dessous, un résumé de la structure adoptée dans ce projet en considérant que si on est en localhost, il faut bien entendu prendre en compte le fait que le client et le serveur sont exactement la même machine.



Outils utilisés

Git

Il s'agit d'un outil de versionnement du code permettant aux développeurs d'avoir une gestion des versions de leurs fichiers avec un historique des changements. Il est possible de faire du développement en parallèle sur plusieurs branches différentes. Le principe étant de développer des modules sur des branches à part et de les rapatriées sur la branche principale (master par défaut) une fois que le module est implémenté et fonctionnel.

Chaque changement apportant une plus-value au code est transformé en « commit » qui est un paquet prêt à être partagé.

Pour notre part nous avons également utilisé des normes de nommage des commits : « opération (qui cela concerne) : commentaire » Ci-dessous les différentes opérations possibles :

- Chore : modification de configuration
- Feat : ajout d'une nouvelle fonctionnalité
- Refactor : changement (réécriture) dans le code ou les fichiers
- Fix : résolution d'un bug

Github

Il s'agit d'une plateforme web utilisant git et permettant de mettre en ligne notre code sur des dépôts privés ou publiques.

Voici notre dépôt : https://github.com/Mathael/mi3-projet

Slack

Il s'agit d'un outil utilisé pour la communication de l'équipe. Étant à distance, nous avions besoin de pouvoir communiquer facilement par écrit (Slack propose des fonctionnalités spéciales pour les développeurs permettant de mettre en avant certains mots dans les textes tapés dans le chat) et expliquer clairement à l'aide de bout de code indenté et avec coloration syntaxique (fonctionnalité appelée « snippet »).

Nous avions également besoin de savoir quand est-ce que l'un d'entre nous travail sur le projet. Slack propose alors d'établir un Webhook (un bot) qui s'occupe de notifier l'équipe lors d'un « push » sur Github et propose des liens permettant directement d'aller voir le travail qui vient d'être fait par le collègue.

JetBrains – PHPStorm

Un IDE gratuit pour les étudiants et les professeurs et permettant de centraliser toute une panoplie d'outils.

MySQL

Nous avons fait le choix d'utiliser une base de données MySQL afin de s'entrainer sur la mise en place d'un cas de figure que nous rencontrerons bien plus souvent que pour du SQLITE.

Fonctionnalités

Fonctionnalités demandées

Description	Réalisée ?
Filtrage de l'affichage des images par catégorie	Oui
Modifications des catégories et des commentaires des images	Oui
Ajout de nouvelles images	Oui
Jugement (anonyme) des photos	Oui
Construction d'albums	Oui
Identification des utilisateurs	Oui

Fonctionnalités réalisées en supplément (non demandées)

Création d'un Template Manager (outil de gestion des vues HTML - objectif 0 php dans les vues)
Création d'un Autoloader (charge automatiquement les Class sans avoir besoin de le faire manuellement)
Création d'un fichier de configuration (centralise la configuration)
Ajout des Namespace (espaces de noms)
Sécurité : restrictions d'accès aux sous dossiers/fichiers de l'application
Front controller réalisé nous même
Utilisation de la ReflectionClass de PHP dans le TemplateManager
Réalisation de ce document ☺

Énumération des fonctionnalités de la grille d'évaluation

Description	Réalisée ?
Image : Affichage première, image suivante, précédente, au hasard	Oui
Groupe : Affichage d'un groupe de plusieurs images à partir de l'image courante	Oui
Groupe, début : Affichage du premier groupe de plusieurs images	Oui
Groupe, suivant : Affichage du groupe suivant de plusieurs images	Oui
Groupe, précédant : Affichage du groupe précédant de plusieurs images	Oui
Groupe aléatoire : Affichage d'un groupe de plusieurs images	Non
Taille du groupe : Chargement du nombre d'images affiché dans le groupe	Oui
Informations : Affichage de la catégorie et du commentaire d'une image	Oui
Catégorie : Mise en place du filtre de la catégorie sur l'image	Oui
Catégorie, début : Affichage de la première image d'une catégorie	Non
Catégorie, suivant : Affichage de l'image suivante de la même catégorie	Non
Catégorie, précédant : Affichage de l'image suivante de la même catégorie	Non
Catégorie, aléatoire : Affichage d'une image aléatoirement dans la catégorie	Non
Catégorie, fin : Fin du filtrage par catégorie	Non
Groupe, catégorie, début : Affichage du premier groupe d'une catégorie	Non
Groupe, catégorie, suivant : affichage du groupe suivant de la même catégorie	Non
Groupe, catégorie, précédant : affichage du groupe précédant de la même catégorie	Non
Groupe, catégorie, aléatoire : affichage aléatoire d'un groupe de la même catégorie	Non
Commentaire, modification : modification du commentaire d'une image	Oui
Catégorie, modification :modification de la catégorie	Oui
Catégorie, ajout : ajout d'une nouvelle catégorie	Non
Catégorie, suppression : suppression d'une catégorie	Non
Image, ajout par URL : ajout d'une image par URL	Non
Image, ajout local : ajout d'une image par téléchargement (upload)	Oui
Jugement, ajout : jugement d'une image	Oui
Jugement, affichage : affichage des images les plus populaires	Non
Album, création : création vide ou avec une image	Oui
Album, destruction : suppression d'album	Non

Album, modification : ajout, suppression d'une image	Oui
Album, visualisation : visualisation d'album	Oui
Design : personnalisation CSS	Oui

IMPORANT : A noter que l'ensemble des fonctionnalités de la grille d'évaluation ne nous ont pas été demandées. Dans le cas contraire, l'ensemble des fonctionnalités auraient été réalisées. Le projet a été réalisé dans le respect du principe MVC et la qualité du code a été un facteur important durant le développement avec des noms de variables permettant une bonne maintenabilité ainsi que des commentaires sur les algorithmes les plus difficiles à comprendre (l'utilisation de la Reflection est fortement documentée dans le code tandis qu'un contrôleur ne l'est presque pas à cause de son extrême niveau de facilité).

Explications détaillées

En plus des fonctionnalités demandées, nous avons décidé d'ajouter d'autres fonctionnalités afin de rendre le projet plus intéressants et de se rapprocher des Framework (en terme de règle de nommage ou fonctionnalités) d'aujourd'hui.

Front controller

Le front contrôleur attend par défaut 2 paramètres : « page » et « action ».

Ces paramètres servent à définir le contrôleur à appeler ainsi que la fonction à exécuter. Un contrôle accru est fait en amont avant d'appeler le contrôleur et sa fonction pour s'assurer qu'ils sont bien des chemins possibles de notre application web. Dans le cas contraire, nous redirigeons l'utilisateur vers la page d'accueil.

Controller

Le contrôleur est appelé par le front contrôleur et celui-ci **implémente** forcément l'**interface** « DefaultController » qui permet d'obliger les contrôleurs à définir la fonction statique « IndexAction » qui est la fonction par défaut de chaque contrôleur. Bien entendu, nous aurions pu faire la même chose de différentes manières plutôt que de passer par un interface cependant cela permet de montrer que nous savons utiliser des interfaces.

DAO

Cette class est appelée uniquement par un contrôleur et permet, comme son nom l'indique, d'avoir accès aux données stockées en base de données. Nos DAO implémentent l'**interface** « CrudDAO » permettant de s'assurer que nos DAO aient tous les fonctions **CRUD** et avec le même nom pour faciliter le développement ainsi que pour la cohérence de l'application.

Database

Notre objet « Database » est l'objet qui ouvre la connexion à la base de données. Il est donc un **Singleton** afin de garantir qu'il n'y ait pas plus d'une seule instance de notre objet Database et donc qu'une seule connexion ouverte à celle-ci.

Template Manager

Cet objet joue le rôle d'interface entre le contrôleur et la vue. Il s'occupe de récupérer le contenu de la page à afficher et offre la fonction « assign » qui prend en paramètre une clé et une valeur associée. La clé se trouve dans le HTML englobée par des doubles accolades et la valeur est une variable, un objet ou un tableau d'objets.

Exemple d'utilisation :

```php
$template->init('image/image');
$template->assign('image', $image);
$template->assign('id', $image->getId());
$template->show();
```

Dans cet exemple, le TemplateManager a été instancié dans le front contrôleur. Lors de son utilisation, on commence par l'initialiser avec la page que l'on souhaite afficher grâce à la fonction « init » (le TemplateManager ira chercher l'HTML dans le répertoire « view »). On lui affecte ensuite les clés – valeurs pour remplacer les clés présentes dans le HTML par leurs variables. Ainsi, la clé {{id}} est remplacée par l'id de l'image.

Dans le TemplateManager, il a été nécessaire d'utiliser la ReflectionClass pour avoir accès aux attributs privés d'une class passée en paramètre (peu importe la class). Le fonctionnement est détaillé dans les commentaires. Il était ici question de ne pas casser ce respect des attributs privé (cf : cours de modélisation UML).

Sessions

Dans un soucis d'harmonisation du code, nous avons opté pour « toute connexion correspond à un objet User ». Actuellement les mots de passes sont enregistrés en clair dans la base de données. On pourrait imaginer les passer par un Hash avant de les enregistrer et effectuer une comparaison des Hash pour se connecter.

Prise en main du projet

Afin de centraliser la configuration de l'application web, nous avons mis en place un fichier de configuration dans le répertoire « config ». Il suffit de suivre les instructions des commentaires pour établir la configuration générale du projet.

```php
// Configurations relatives à la base de données
const DATABASE_URL = 'localhost';
const DATABASE_TYPE = 'mysql';
const DATABASE_PORT = '3306';
const DATABASE_NAME = 'php-image-project';
const DATABASE_USER = 'root';
const DATABASE_PASSWORD = '';

// Url du site : localhost par défaut
const APP_URL = 'http://localhost/';

// Nom du répertoire direct après 'www' sans ajouter de slash ou d'antislash
// Si le projet possède 'www' pour racine, alors laisser cette variable vide : ''
const APP_DIRECTORY = 'tp1/app';
```

L'installation de la base de donnée est réalisée automatiquement en se rendant à l'adresse ci-dessous :

http://votre-url/votre-dossier/?page=install

Le script retourne une page d'explication une fois exécuté.