UNIVERSITÉ DE REIMS CHAMPAGNE-ARDENNE

Licence 2 Informatique
INFO0303 – 2020/2021
***Resp. :* C. Rabat**

Travaux dirigés n° 4

Programmation MVC

Dans ce TD, nous allons réaliser un micro-framework MVC. L'objectif est de comprendre le fonctionnement du modèle et d'arriver à positionner l'ensemble des éléments (scripts, ressources) d'une application Web. En travaux pratiques, nous utiliserons le *framework Laravel*.

Exercice 1 (Description de l'application)

Nous souhaitons développer une application Web qui permet d'afficher et gérer une liste d'actualités. Une actualité est liée à un utilisateur. Lorsqu'un visiteur arrive sur le site, les dernières actualités sont affichées. Depuis une autre page, il peut consulter la liste complète. Un utilisateur connecté pourra ajouter de nouvelles actualités, modifier ou supprimer ses actualités.

1. Donnez les cas d'utilisation.
2. Donnez les diagrammes de navigation.
3. Représentez le MLD.

Exercice 2 (Un micro-framework MVC)

Nous souhaitons développer un micro-*framework* dont les routes reposent sur des actions, regroupées par modes. Pour accéder à un mode `toto` et à une action associée `tata`, il suffit d'appeler le script à la racine du site `index.php?mode=toto&action=tata`.

1. Comment est appelé le script `index.php` dans le modèle MVC ? Quel est son rôle ?
2. Comment peut-on récupérer le mode et l'action en PHP ?
3. Nous supposons que le mode correspond à une classe donnée et que chaque action correspond à une méthode de cette classe. Expliquez comment exécuter la bonne méthode de la bonne classe en PHP.
4. Comment sont appelées chacune des classes correspondant à un mode dans le modèle MVC ?

Nous supposons que le squelette du script `index.php` est le suivant :

```php
// Configuration
// Démarrage de la session
// Récupération du mode et de l'action
// Vérification de la validité du mode
// Vérification de la validité de l'action
// Exécution de la méthode
$methodName();
```

5. À votre avis, quelles actions correspondent à chaque section du script ? Expliquez leurs intérêts.
6. Que signifie la dernière ligne ?
7. Donnez le code complet.

Notre micro-*framework* possède l'arborescence suivante :

```
├ class
├ config
├ controller
├ model
├ public
├ vendor
├ view
└ index.php
```

8. Selon vos connaissances en MVC, déterminez le rôle de chaque répertoire et donnez-en le contenu.
9. Quels répertoires ne doivent-ils pas être directement accessibles ?

Exercice 3 (Les vues)
Nous souhaitons mettre en place un template pour toutes les pages du site. Nous souhaitons également avoir un menu dans chaque page qui permet d'aller soit sur l'accueil, soit sur la liste des actualités.

1. Quelles données sont nécessaires dans le template ?
2. À votre avis, est-il judicieux de séparer le template du menu ?
3. Comment inclure le menu dans le template ?
4. Sachant que le contenu de la page affiché dans le template correspond à une vue en HTML (pas toujours la même), comment est-il possible de le fournir au template ?
5. Pour résoudre tous ces problèmes, nous souhaitons utiliser la classe `WebPage` développée dans un autre TD/TP. Montrez comment l'utiliser dans le template et quels attributs et méthodes sont nécessaires.

Exercice 4 (Les contrôleurs et modèles)
Dans cet exercice, nous nous intéressons à la gestion des actualités.

1. Rappelez le contenu de la classe `Actualite`.
2. Nous souhaitons écrire le modèle correspondant à la classe `Actualite`. Quelles méthodes sont nécessaires ?
3. Nous souhaitons maintenant écrire le contrôleur lié à la gestion des actualités. Quelles méthodes sont nécessaires ?
4. Quelles vues sont nécessaires ?
5. Donnez le contenu de la méthode `index` qui permet d'afficher la liste des actualités.

Annexes - Manuel PHP

Fonction **file_exists** : vérifie si un fichier existe.

```
bool file_exists ( string $filename )
```

Fonction **method_exists** : vérifie si une méthode existe pour un objet ou une classe.

```
bool method_exists ( mixed $object , string $method_name )
```

Fonction **ob_start** (version simplifiée) : enclenche la temporisation de sortie.

```
bool ob_start( void );
```

Fonction **ob_get_contents** : retourne le contenu du tampon de sortie.

```
string ob_get_contents ( void )
```

Fonction **ob_end_clean** : détruit le tampon et stoppe la temporisation.

```
bool ob_end_clean ( void )
```