IUT de Montpellier - Programmation Web - TD5
Architecture MVC avancée

Rémi Coletta, Romain Lebreton, Bastien Vialla

Semaine du 20 Octobre 2014

Aujourd'hui nous allons développer notre site-école de covoiturage. Au fur et à mesure que le projet grandit, nous allons bénéficier du modèle MVC qui va nous faciliter la tâche de conception. En attendant de pouvoir gérer les sessions d'utilisateur, nous allons développer l'interface "administrateur" du site.

La semaine dernière, nous avions un site qui propose une gestion minimale des utilisateurs (Create / Read / Update / Delete). L'objectif de ce TD est d'avoir l'interface similaire pour les trajets. Nous allons dans un premier temps rendre notre MVC d'utilisateur plus générique. Cela nous permettra de l'adapter plus facilement aux trajets dans un second temps.

1 Mise en route

Pour ce TD, nous vous passons un squelette de site qui sera à compléter et à enrichir.

Q 1. Récupérez sur `http://www.lirmm.fr/~lebreton/teaching.html` l'archive TD5.zip qui vous servira de base pour ce TD. Décompressez cette archive dans votre `public_html`. **Rentrez vos informations de connexion dans `./config/Conf.php`.**

Si vous utilisez NetBeans, ce qui est conseillé, créez un nouveau projet à partir du répertoire TD5 (File → New Project → 'Php application from existing sources' → sélectionnez le répertoire TD5).

2 Vues modulaires

En l'état, certains bouts de code de nos vues se retrouvent dupliqués à de multiples endroits. Par exemple, l'affichage de la liste des utilisateurs, qui se trouve dans `viewListUtilisateur.php`, se retrouve en partie dans `viewCreatedUtilisateur.php`, `viewDeletedUtilisateur.php` et `viewUpdatedUtilisateur.php`.

Les prochaines questions vont vous aider à réorganiser le code pour éviter les redondances.

2.1 Mise en commun de l'en-tête et du pied de page

Q 2. Modifier la vue `viewListUtilisateur.php` pour ajouter en en-tête de page une barre de menu, avec deux liens :

- un lien vers la page d'accueil des utilisateurs `index.php?controller=utilisateur&action=readall`
- un lien vers la future page d'accueil des trajets `index.php?controller=trajet&action=readall`

Q 3. Nous souhaitons que notre site ait cette en-tête de page affichée partout. Au choix, dupliquer cette modification sur chacune des vues (mauvaise idée) ou traiter la question suivante.

Q 4. Créer une vue générique `TD5/view/view.php` avec le code suivant. La fonction de `view.php` est de charger une en-tête et un pied de page communs, ainsi que la bonne vue en fonction de la variable `$view`.

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <meta charset="UTF-8">
5         <title><?php echo $pagetitle; ?></title>
6     </head>
7     <body>
```

```
8           <nav>
9               <ul>
10                  <li>
11                      <a href="?controller=utilisateur&action=readall">
12                      Gestion des utilisateurs</a>
13                  </li><li>
14                      <a href="?controller=trajet&action=readall">
15                      Gestion des trajets</a>
16                  </li>
17              </ul>
18          </nav>
19  <?php
20  // Si le controleur est 'utilisateur' et la vue 'find',
21  // chargeons './view/utilisateur/viewFindUtilisateur.php'
22  require VIEW_PATH.$controller.DS.'view'.ucfirst($view).ucfirst($controller) . '.php';
23  ?>
24      </body>
25  </html>
```

Explication : La fonction `ucfirst` (UpperCase FIRST letter) sert à mettre en majuscule la première lettre d'une chaîne de caractère.

Q 5. Dans vos vues existantes, supprimer les parties du code correspondant aux header et footers, donc déjà incluses dans `view.php`.

Q 6. Reprendre le contrôleur pour que, par exemple, à la place d'un `require './view/utilisateur/viewFindUtilisateur.php'`, on initialise la variable `$view` avec `'find'` et `$pagetitle` avec le titre de page `'Détails de l'utilisateur'`.

Redéfinir le `VIEW_PATH` en début de fichier par `define('VIEW_PATH', ROOT . DS . 'view' . DS);`

Enfin, rajouter un `require VIEW_PATH . "view.php";` à la fin du fichier pour appeler notre vue générique.

2.2 Réutilisation du code pour la liste des utilisateurs

Q 7. Réutiliser le corps de `viewListUtilisateur.php` dans `viewCreatedUtilisateur.php`, `viewDeletedUtilisateur.php` et `viewUpdatedUtilisateur.php`.

Le fichier `viewDeletedUtilisateur.php` ne doit plus faire que deux lignes maintenant.

2.3 Un seul formulaire pour la création et la mise à jour

Q 8. Les vues `viewCreateUtilisateur.php` et `viewUpdateUtilisateur.php` sont quasiment identiques : elles affichent le même formulaire, et le ré-remplissent ou non. Nous allons donc fusionner `viewCreateUtilisateur.php` et `viewUpdateUtilisateur.php` en une unique page. En pratique,

- on supprime la vue `viewCreateUtilisateur.php` et on modifie le contrôleur de sorte que l'action `create` appelle la vue `viewUpdateUtilisateur.php`.
- **Attention :** quand on arrive sur la vue `viewUpdateUtilisateur.php` depuis l'action `create`, les variables `$n`, `$p`, ... ne sont pas renseignées. Penser à les initialiser à chaîne vide dans le contrôleur.
- le champ `login` du formulaire doit être `required` si l'action est `create` ou `readonly` si l'action est `update` (on ne peut pas modifier la clé primaire d'un tuple). Utiliser une variable dans le contrôleur pour permettre l'adaptation de la vue à ces deux actions.
- enfin, le champ `action` du formulaire doit être `save` si l'action est `create` ou `updated` si l'action est `update`. Là aussi, utiliser une variable spécifique.

3 CRUD pour les trajets

L'implémentation du CRUD pour les trajets est un code très similaire à celui pour les utilisateurs. Nous pourrions donc copier/coller le code des utilisateurs et changer les (nombreux) endroits nécessaires.

Pour éviter de perdre un temps conséquent à développer le CRUD pour chaque nouvel objet, nous allons le créer automatiquement autant que possible.

3.1 Création d'un modèle générique

Nous allons déplacer de `ModelUtilisateur.php` vers le modèle générique `Model.php` toutes les fonctions qui ne sont pas spécifiques aux utilisateurs.

Q 9. Commençons par la fonction `selectAll()` de `ModelUtilisateur.php`. Dans cette fonction, seul le nom de la table présent dans la requête SQL varie.

1. Déplacez la fonction `selectAll()` de `ModelUtilisateur.php` vers `Model.php`.
2. Créez dans `ModelUtilisateur.php` une variable `$table` qui est `protected` (accessible uniquement dans la classe courante et ses classes filles) et `static` (qui ne dépend que de la classe, pas des objets).
3. Utilisez cette variable dans la fonction `selectAll()` de `Model.php` pour faire la requête sur la bonne table. Pour cela, accéder à la variable `$table` avec `static::$table` dans `Model.php`.

   Plus d'explications : La syntaxe `static::$table` est quelque peu subtile. Dans notre cas, elle permet que lorsque l'on appelle `ModelUtilisateur::selectAll()`, qui est héritée de `Model::selectAll()`, la variable `static::$table` aille chercher `ModelUtilisateur::$table` et non pas `Model::$table`.
4. Testez que votre site marche toujours.

Q 10. Passons à la fonction `select()`. Dans cette fonction, le nom de la table et la condition `WHERE` varie.

1. Déplacez la fonction `select()` de `ModelUtilisateur.php` vers `Model.php`.
2. Utilisez la variable statique `$table` de `ModelUtilisateur.php` pour remplacer le nom de la table.
3. Créez une variable statique `$primary` dans `ModelUtilisateur.php` qui contiendra le nom du champ de la clé primaire. Utilisez cette variable pour remplacer le nom de la clé primaire dans `select()`.

Q 11. Répétez la question précédente avec la fonction `delete()`.

Q 12. Passons à la fonction `update()`. Pour reconstituer la requête

```
'UPDATE utilisateur SET nom=:nom,prenom=:prenom,email=:email,login=:login WHERE login=:login',
```

il est nécessaire de pouvoir lister les champs de la table 'utilisateur'. Ces champs sont les entrées du tableau `$data` et c'est ainsi que nous allons les récupérer.

1. Déplacez la fonction `update()` de `ModelUtilisateur.php` vers `Model.php`.
2. Remplacer la table et le nom de la clé primaire par les variables adéquates.
3. Nous allons générer la partie `SET` à partir des clés du tableau associatif `$data`. Autrement dit, si `$data['un_champ']` existe, nous voulons rajouter la condition `'un_champ = :un_champ'` à `SET`.

   Indice : Utilisez la boucle `foreach ($tableau as $cle => $valeur)` pour récupérer les clés du tableau. Googler aussi la fonction `rtrim` de PHP qui pourra vous être utile pour enlever la virgule de trop à la fin de votre requête.

Q 13. Répétez la question précédente avec la fonction `insert()`.

3.2 Adaptation du contrôleur

Pour mémoire, dans la variante du MVC que nous avons choisi d'implémenter, il y a un contrôleur par classe. Dans cette partie, nous allons nous donc créer un contrôleur pour la classe 'Trajets'.

Attention : NE PAS copier/coller bêtement `ControleurUtilisateur.php` en `ControleurTrajet.php`. Adaptez chacune des actions de `ControleurTrajet.php` et les tester une à une.

Q 14. Créer une vue `viewListTrajets`, et gérer l'action `readAll` de `ControleurTrajet.php`.

Q 15. Faire de même pour les autres actions. Nous vous conseillons de faire dans l'ordre les actions `read`, `delete`, `create`, `update`, `save` et `updated`.