Introduction au MVC

templates php, composer, HttpFoundation, routage

Dans ce TD, vous allez commencer à vous familiariser avec les composants d'un Framework MVC et voir l'utilité de recourir à de tels outils.

Une introduction générale à ce sujet se trouve ici :
`http://symfony.com/doc/current/book/from_flat_php_to_symfony2.html` et un petit exemple plus détaillé se trouve là :
`http://fr.openclassrooms.com/informatique/cours/adopter-un-style-de-programmation-c application-pratique-un-systeme-de-news`

Exercice 1. Du PHP pur aux templates PHP

Considérons le code suivant en interrogeant la table CARNET depuis PHP avec PDO

```php
require("connect.php");
$dsn="mysql:dbname=".BASE.";host=".SERVER;
    try{
      $connexion=new PDO($dsn,USER,PASSWD);
    }
    catch(PDOException $e){
      printf("Echec connexion : %s\n", $e->getMessage());
      exit();
    }
$sql="SELECT * from CARNET";
if(!$connexion->query($sql))
  echo "Pb pour acceder au CARNET";
else {
    foreach ($connexion->query($sql) as $row)
      echo $row['NOM']<br/>\n";
}
```

On peut observez quelques défauts :

— Réutilisabilté du code très réduite

— Si on fabrique un formulaire avec les entrées du carnet où mettre le code ?

Exercice 2. Un template PHP On peut améliorer un peu les choses :

```php
require("connect.php");
$dsn="mysql:dbname=".BASE.";host=".SERVER;$
    try{
      $connexion=new PDO($dsn,USER,PASSWD);
    }
    catch(PDOException $e){
      printf("Echec connexion : %s\n", $e->getMessage());
      exit();
    }
$sql="SELECT * from CARNET";
if(!$connexion->query($sql)) echo "Pb pour acceder au CARNET
   ";
else{
    $amis=Array();
    foreach ($connexion->query($sql) as $row)
    $amis[]=$row;
    require "templates/listeamis.php";
}
```

Avec un template du style suivant à placer dans templates/listeamis.php.

```php
<!DOCTYPE html>
<html lang='fr'>
    <head>
        <meta charset="utf-8"/>
        <link rel="stylesheet" href="css/monstyle.css"/>
        <title>Liste de mes Amis</title>
    </head>
    <body>
        <h1>List of friends</h1>
        <ul>
            <?php foreach ($amis as $ami): ?>
              <li>
              <a href=
            "../recherche.php?id=<?php echo $ami['ID'] ?>">
              <?php echo $ami['PRENOM'].' '.$ami['NOM']; ?>
              </a>
              </li>
            <?php endforeach; ?>
        </ul>
    </body>
</html>
```

On commence à séparer la présentation du codage "métier".

Exercice 3. Isoler encore la logique applicative

```php
<?php
//modele.php
require("connect.php");
function connect_db(){
$dsn="mysql:dbname=".BASE.";host=".SERVER;$
    try{
      $connexion=new PDO($dsn,USER,PASSWD);
    }
    catch(PDOException $e){
      printf("Echec connexion : %s\n", $e->getMessage());
      exit();
    }
    return $connexion;
    }
// Puis
function get_all_friends(){
  $connexion=connect_db();
  $amis=Array();
  $sql="SELECT * from CARNET";
  foreach ($connexion->query($sql) as $row){
    $amis[]=$row;
  }
  return $amis;
}
```

On peut maintenant avoir un **controleur** très simple :

```php
<?php
//c-list.php
require_once 'modele.php';

$amis = get_all_friends();

require 'templates/listamis.php';
```

Exercice 4. Layout

Il reste une partie non réutilisable dans le code : le layout. Essayons de remédier à ça :

```php
<!-- templates/baseLayout.php -->
<!DOCTYPE html>
<html lang='fr'>
    <head>
        <meta charset="utf-8"/>
        <link rel="stylesheet" href="css/monstyle.css"/>
        <title><?php echo $title ?></title>
    </head>
    <body>
        <?php echo $content ?>
    </body>
</html>
```

Le template listamis.php peut maitenant récupérer cette structure de base

Exercice 5. Héritage de templates

```php
# templates/t-list.php
<?php
  $title = 'Liste des amis';
  ob_start();
?>
    <h1>Liste de mes amis</h1>
    <ul>
        <?php foreach ($amis as $ami): ?>
        <li>
            <a href=
            "../recherche.php?id=<?php echo $ami['ID'] ?>"
            >
                <?php echo $ami['PRENOM'].' '.$ami['NOM'];
                  ?>
            </a>
        </li>
        <?php endforeach; ?>
    </ul>
<?php
  $content = ob_get_clean();
  include 'baseLayout.php'
?>
```

Observez l'utilisation de la bufferisation en PHP avec `ob_start()` et `ob_get_clean()`. Cette dernière fonction récupère le contenu bufferisé et nettoie ensuite le buffer.

Exercice 6. Affichage des détails d'une personne

On va ajouter à notre modèle une fonction pour afficher les détails d'une personne :

```
1 function get_friend_by_id($id){
2   $connexion=connect_bd();
3   $sql="SELECT * from CARNET where ID=:id";
4   $stmt=$connexion->prepare($sql);
5   $stmt->bindParam(':id', $id, PDO::PARAM_INT);
6   $stmt->execute();
7   return $stmt->fetch();
8 }
```

On peut maintenant créer un nouveau controleur c-details.php :

```
1 <?php
2 //c-details.php
3 require_once 'modele.php';
4 $pers = get_friend_by_id($_GET['id']);
5 require 'templates/t-details.php';
6 ?>
```

Qui utilise le template :

```
1 <?php
2   //templates/t-details.php
3   $title = $pers['NOM'];
4   ob_start();
5 ?>
6 <h1>details sur
7 <?php echo $pers['PRENOM'].' '.$pers['NOM'] ?>
8 </h1>
9 <p>
10 <?php
11   echo ' Ne le '.$pers['NAISSANCE'];
12   echo '<br/>Ville:'.$pers['VILLE'];
13   $content = ob_get_clean();
14   include 'baseLayout.php'
15   ?>
```

Vous pouvez tester en entrant l'URL de c-details.php avec un paramètre id. Le code est similaire à celui du premier template et nous pouvons réutiliser le template de base, mais il subsiste plusieurs problèmes. Si le paramètre id n'est pas fourni, notre application va provoquer une erreur. En outre nous n'avons pas de controleur principal. Regroupons d'abord le code des 2 contrôleurs (c-list.php et c-details.php) dans un fichier unique controllers.php

```php
function list_action()
{
    $amis = get_all_friends();
    require 'templates/t-list.php';
}

function detail_action($id)
{
    $pers = get_friend_by_id($id);
    require 'templates/t-detail.php';
}
```

Puis nous pouvons proposer un controleur principal (Front Controller) `index.php` :

```php
<?php
// index.php
// On charge les modeles et les controleurs
require_once 'modele.php';
require_once 'controllers.php';

// gestion des routes
$uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
if ('/index.php' == $uri) {
    list_action();
} elseif ('/index.php/detail' == $uri && isset($_GET['id'])) {
    detail_action($_GET['id']);
} else {
    header('Status: 404 Not Found');
    echo '<html><body><h1>Page Not Found</h1></body></html>';
}
```

On peut améliorer encore un peu les choses en mettant en place une classe de connexion à la BD (Pattern Singleton ?) et une classe `Contacts` également dans le modèle permettant d'autres actions : suppression ou mise à jour d'un contact de façon à obtenir un **CRUD** complet. Au niveau des templates, nous utiliserons plutôt Twig qui est plus simple et plus performant que les templates PHP. Vous pouvez observer un code correspondant sur le dépôt `https://github.com/roza/php-basic-mvc` Ce système de routage est encore très incomplet et nous allons plutôt utliser pour continuer sur de bonnes bases un framework Web comme Symfony.