TP 7 PWS

IUT BLAGNAC
NICOLAS BRIET

Table des matières

Introduction

Dans ce TP nous allons aborder le principe du MVC vu en TD pour savoir comment bien organiser la structure de son site Web.

1 – Adaptation d’une application MVC

Partie 1 : L’index:
Ce fichier ne change pas parce que c’est le principe même du MVC. L’index se contente toujours d’appeler le routeur

Partie 2: Routeur:

```php
public function routerRequete() {
// s'il y a un parametre 'entite'
    if (isset($_GET['entite'])) {
  // on détermine avec quelle entité on veut travailler
  switch($_GET['entite']) {
    case 'media' :
      // on détermine quelle action (CRUD) on veut effectuer sur l'entité choisie
      switch($_GET['action']) {
        case 'C' :  // 'C' = Create = ajout d'un étudiant...
              break;
        case 'R' :  // 'R' = Read = lecture des étudiants ou d'un seul s'il y a un parametre id
              if (isset($_GET['id'])) {
                $ctrlMedia = new ControleurMedia();
                $ctrlMedia->getMedia($_GET['id']);
              }
              else {
                $ctrlMedia = new ControleurMedia();
                $ctrlMedia->getListeMedia();
              }
              break;
        case 'U' :  // 'U' = Update = modification d'un étudiant à partir de son id
              break;
        case 'D' :  // 'D' = Delete = suppression d'un étudiant à partir de son id
              break;
        default:  // pour toutes les autres valeurs du parametre 'action', on affiche la liste des étudiants
              break;
      }
      break;

    case 'type' :
      switch($_GET['action']) {
        case 'C' :  // 'C' = Create = ajout d'un groupe...
              break;
        case 'R' :  // 'R' = Read = lecture des groupes ou d'un seul s'il y a un parametre id
              if (isset($_GET['id'])) {
                $ctrlType = new ControleurType();
                $ctrlType->getListeMediaByType($_GET['id']);
              }
```

Le principe du routeur reste le même, les choses qui ont changé par rapport au mvc donné sont:
Les **cases** pour que l’url soit plus en rapport avec ce dont le site traite.
Les **contrôleurs** par ce que les contrôleurs ne sont plus ceux d’Etudiant et de Groupe
Les **methodes** de ces contrôleurs due aux changement de contrôleurs.

Partie 3 : les contrôleurs ex avec *ControleurType.php*

```php
class ControleurType {
  private $modeleType;
  private $modeleMedia;

  public function __construct() {
    $this->modeleType = new ModeleType();
    $this->modeleMedia = new ModeleMedia();
    }

  public function getListeType() {
        $vListeType = $this->modeleType->getListeType();
        include 'Vue/VueListeType.php';
    }

  public function getListeMediaByType($idType) {
    $vListeMedia = $this->modeleMedia->getListeMediaByType($idType);
        include 'Vue/VueListeMedia.php';
    }
}
```

Les contrôleurs ont changé mais le principe reste le même: ces contrôleurs appellent des méthodes de modèle et ensuite les fichiers de vues qui vont utiliser le résultat des précédents appellent de méthode de modèle.

Partie 4 : Les modèles ex avec *ModelMedia.php*

```php
class ModeleMedia {
  // methode qui renvoie un tableau d'objets Etudiants
  // ce tableau est construit à partir d'une requête SQL sur la table Etudiants de la BD
    public function getListeMedia() {
    // cette instruction permet d'utiliser dans cette fonction la variable $conn
    // qui a été initialisée dans le script connect.inc.php
    global $conn;
    $res = $conn->prepare("Select * from Medias");
    $res->execute();
    foreach($res as $media) {
        $ListeMedia[] = new Etudiant($media["idMedia"], $media["idType"], $media["nomMedia"], $media["dateParution"]);
    }
    return $ListeMedia;
    }

    public function getMedia($idMedia) {
    global $conn;
    $res = $conn->prepare("Select * from Medias where idMedia = :pIdMedia");
    $res->execute(array('pIdMedia' => $idMedia));
    $media = $res->fetch();
    $unMedia = new Media($media["idMedia"], $media["idType"], $media["nomMedia"],$media["dateParution"]);
        return $unMedia;
    }

  public function getListeMediaByType($idType) {
    global $conn;
    $res = $conn->prepare("Select * from Medias where idType = :pidType");
    $res->execute( array ('pidType' => $idType) );
    foreach($res as $media) {
        $ListeMediaType[] = new Media($media["idMedia"], $media["idType"], $media["nomMedia"], $media["dateParution"]);
    }
    return $ListeMediaType;
    }
}
```

Ce sont les modèles qui vont donner les resultats aux vues en fonction des informations qu'ils piochent dans la base de donnée.

Partie 5 : Les vues ex avec *VueListeMedia.php*

```php
<html>
<head></head>
<body>
    <table border="2">
        <tbody>
          <tr><th>id Media</th><th>id Type</th><th>nom Media</th><th>date de Partution</th></tr>
        </tbody>
        <?php
      // s'il y a des étudiants à afficher
           if (count($vListeMedia) >= 1) {
        foreach ($vListeMedia as $vMedia) {
           echo '<tr><td><a href="index.php?entite=media&action=R&id='.$vMedia->idMedia.'">'.$vMedia->idMedia.'</a></td>';
           echo '<td>'.$vMedia->idType.'</td><td>'.$vMedia->nomMedia.'</td><td>'.$vMedia->dateParution.'</td></tr>';
        }
      }
      else {
        echo "Pas de media...<BR/>";
      }
        ?>
    </table>
</body>
</html>
```

Les vues se servent des résultats qu'on leurs fournis pour les afficher. Ici donc la vue possède une syntaxe html basique pour l'affichage et de plus en php parcours la liste *$vListeMedia* et l'affiche dans un tableau html.

Conclusion

Nous avons donc vu que le MVC est un principe permettant la maintenabilité et la réutilisation d'un code en partageant les différentes fonction du site dans des fichiers différents pour une meilleur organisation.