*Laboratoire 1.1 – Introduction MVC*

**Remise** : Déposez sur Moodle un zip contenant tous les fichiers du site Web.

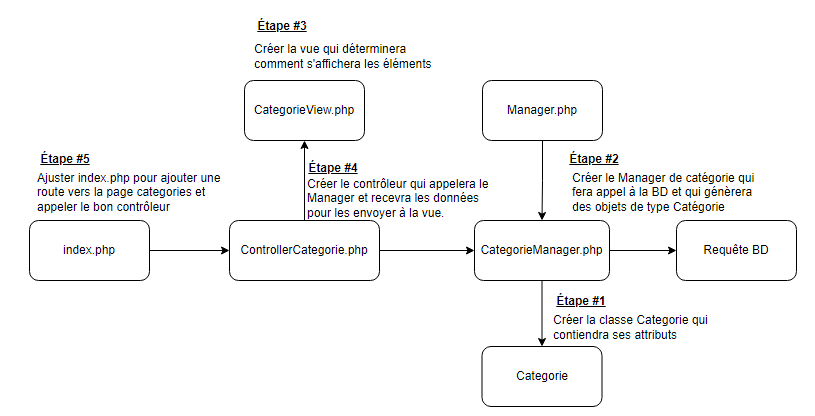
Dans ce laboratoire, vous partirez d’un modèle déjà en forme MVC afin d’y apporter des modifications.

**Étape préliminaire**

1. Créez une base de données MySQL qui se nomme « demomvc » et exécutez le script fourni dans ce zip.
2. Déplacez le dossier « mvc » contenu dans ce zip et placez-le dans le dossier htdocs de Xampp puis valider que le site Web est fonctionnel.

**Partie 1 - Ajout de la page des catégories (**[**voir vidéo pour le tutoriel**](https://web.microsoftstream.com/video/164993c1-bb08-4d12-bef2-cdc998839171)**)**Dans cette section nous allons ajouter une page qui permettra la consultation des catégories.

Pour vous aider à comprendre les étapes que nous ferons dans cette partie, voici un petit schéma qui représente les étapes qui vont suivre.

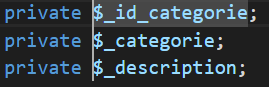
****

1. **La classe « Categorie ».**

Dans le répertoire « model », créez un fichier **Categorie.php**. Ce fichier contiendra la classe Categorie avec les propriétés : id\_categorie, categorie et description ainsi qu’un constructeur et les getters et setters.

*Inspirez-vous de la classe* ***Produits.php*** *pour la syntaxe.*

*Rappel* : Pour créer rapidement tous les getters et setters dans VS Code, créez vos propriétés puis placez le curseur devant le « $ » de la première propriété puis maintenez les touches « CTRL » et « ALT » et appuyez sur la touche « flèche en bas » pour ajouter un curseur à la même position dans la ligne suivante. Ajoutez un curseur sur toutes les propriétés (voir image ci-dessous) puis faites un clic-droit sur le curseur pour sélectionner l’option « Insert PHP Getter and Setter ». Voilà !



*Remarquez le curseur qui est étendu sur les 3 lignes.*

1. **Gestionnaire de catégorie.**

Dans le répertoire « model », créez un fichier **CategorieManager.php**. Ce fichier contiendra la classe **CategorieManager** qui contiendra les fonctions qui feront les requêtes à la BD et construiront les objets nécessaires. En vous inspirant de la classe **ProduitManager**, créez la classe sans les fonctions et en ajustant les require\_once().

Créez une fonction nommé **getCategories()** qui listera toutes les catégories.  
En vous inspirant de la fonction **getProduits()** dans le fichier **ProduitManager.php**, créez une fonction qui retourne un array d’objet de type Categorie en sélectionnant toutes les catégories trié par ordre d’id de catégorie.

1. **La vue Catégories**Dans le dossier view, créez un fichier nommé **categoriesView.php**.   
   En vous inspirant de **produitsView.php** créer la vue qui permettra l’affichage des catégories.

**Indice sur les changements à faire :**

* Changer la valeur de la variable $title pour « Categories »
* Changer la valeur du <h1> pour « Les catégories »
* Changer les variables de la boucle foreach
  + Remplacer $produits par $categories et $produit par $categorie
  + Ajuster le contenu de la boucle en modifiant le contenu du <h3>
  + Ajuster l’utilisation des getters appelant les bons getters de la classe Categorie.

1. **Contrôleur Categorie**

Dans le dossier controller, créez un fichier nommé **controllerCategorie.php**.

En vous inspirant de **controllerProduit.php**, ajoutez un require pour le gestionnaire catégorie soit **CategorieManager** et créez une fonction **listCategories()** qui instancie un nouvel objet de type **CategorieManager**. Ensuite remplissez une variable nommé $categories en appelant la fonction getCategories() du gestionnaire.

Il ne manque que l’ajout de la vue créé précédemment. Ajoutez un require de **categoriesView.php**.

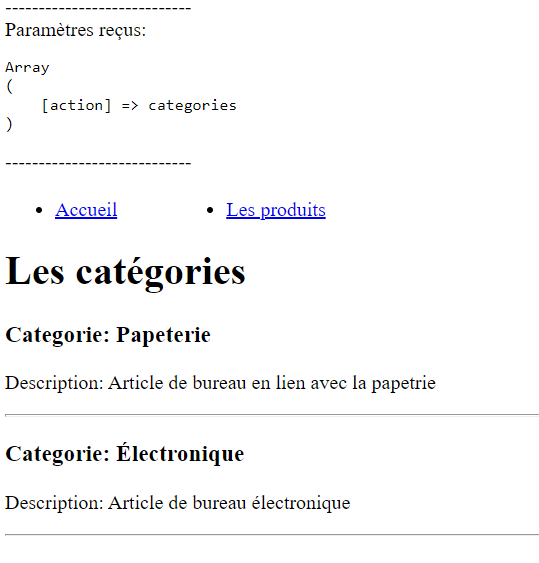
1. **Ajuster le routeur index.php**

Dans le fichier index.php, ajouter une condition elseif sous le elseif qui vérifie si la variable action est égale à « produit ». La nouvelle condition elseif devra valider si la variable action est égale à « categories ».

Dans cette condition, ajouter un require vers le contrôleur de catégorie et un appel à la fonction que vous avez créé dans cette dans ce fichier soit **listCategories()**.

Si tout va bien en tapant l’URL <http://localhost/mvc/categories> vous devriez voir la liste de vos catégories.

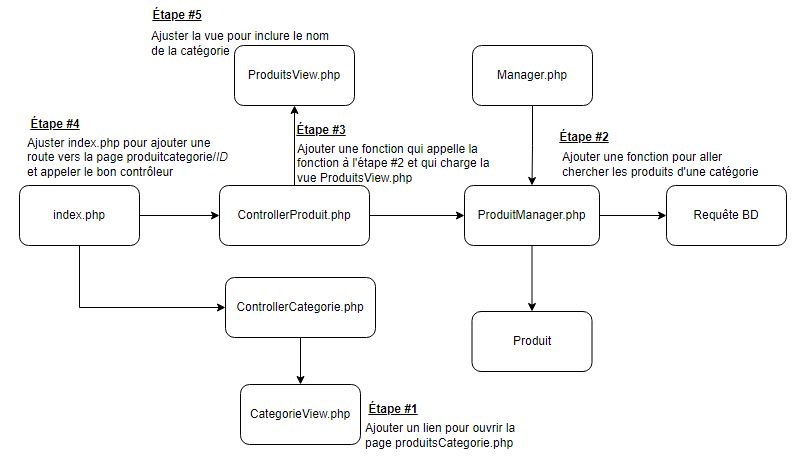
**Résultat attendu :**



**Partie 2 - Ajout de la page les produits d’une catégorie**

Dans cette section nous ajouterons un lien dans la page « categories » sur chaque catégorie pour consulter les produits d’une catégorie. Chaque catégorie aura un lien comme ceci <http://localhost/mvc/produitscategorie/1>. Où 1 représente l’id de la catégorie.

Pour vous aider à comprendre les étapes que nous ferons dans cette partie, voici un petit schéma qui représente les étapes qui vont suivre.



1. **Ajout du lien dans la vue categoriesView.php**

Sous la balise <p> qui affiche la description de la catégorie, ajoutez une balise <a> ayant comme texte « Voir les produits » qui mène à la page : **produitscategorie/*id\_categorie***

La valeur de id\_categorie doit être la valeur de l’id de la catégorie en cours.

*N.B. Ce lien ne fonctionnera pas pour le moment.*

1. **Ajout de la fonction dans le gestionnaire de produit (ProduitManager.php)**

Ajoutez une nouvelle fonction nommée getProduitsCategorie qui reçoit en paramètre l’id de la catégorie à filtrer. La fonction doit faire un SELECT dans la table Produit en sélectionnant seulement les produits ayant l’id de catégorie reçu en paramètre. Ajouter aussi le nom de la catégorie dans la requête. Construisez un array d’objet de type Produit nommé $produits avec le résultat de la requête puis retourner ce tableau.

*N.B. La fonction getProduit() ressemble beaucoup à ce qu’il faut faire.*

1. **Ajout de la fonction dans le contrôleur de Produit (ControllerProduit.php)**

Ajoutez une fonction nommée listProduitsCategorie() qui recevra en paramètre l’id de la catégorie.

Cette fonction instanciera un nouveau gestionnaire de produit et appellera la fonction getProduitsCategorie() que nous venons de créer en passant en paramètre l’id de la catégorie. Cet appel nous retournera un tableau de type Produit qui contiendra les produits de la catégorie passé en paramètre.

Ajoutez dans cette fonction une variable $categorie qui contiendra le nom de la catégorie filtrée. Utilisez le tableau $produits pour récupérer cette valeur.

L’affichage que nous souhaitons faire est la même que la liste de produits déjà faite à l’exception que seuls les produits d’une catégorie seront affichés. Nous pourrons donc réutiliser la vue produitsView.

Ajoutez un require de produitsView.

1. **Modification du routeur index.php pour ajouter le cas de l’appel à la page produitscategorie/id**

Ajouter un nouveau elseif sous celui qui vérifie si action vaut « categories ». Cette condition doit valider si action est égale à « produitscategorie » À l’intérieur de ce elseif, valider qu’il y a bien un paramètre $\_GET[‘id’] et qu’il est supérieur à 0. À l’intérieur de cette condition, ajouter un require du contrôleur de Produit et appelez la fonction listProduitsCategorie() que vous avez créé dans le contrôleur.

*N.B. Ce elseif ressemblera à celui qui valide que la variable action est égale à « produit ».*

À partir d’ici, les liens dans la page des catégories devraient fonctionner et afficher une liste de produit selon la catégorie choisie.

1. **Modification de la vue ProduitsView pour ajouter le nom de la catégorie**

Il serait intéressant de connaitre le nom de la catégorie lorsqu’il y a un filtre dans l’affichage des produits

Dans ProduitsView :

* Remplacez le contenu de la balise <h1> par le echo d’une variable nommée $titreH1.
* Déclarez cette nouvelle variable au début du fichier et affectez-y la valeur « Les produits ».
* Ajoutez un if qui vérifie si la variable $categorie existe et si c’est le cas concaténer sa valeur à la variable $titreH1. Voir l’exemple ci-dessous pour le format.

*N.B. la variable $categorie a été créé dans le Manager de Produit dans la fonction getProduitCategorie().*

Voilà vous devriez voir le nom de la catégorie dans la page <http://localhost/mvc/produitscategorie/1>

**Résultat attendu :**



**Astuces de débogage :**

* Est-ce que votre requête suit le bon chemin ?
  + Depuis index.php, faites des echo pour voir par où passe votre requête.
* Est-ce que vos variables contiennent les bonnes informations.
  + À quelques endroits stratégiques faites des echo ou des print\_r pour valider que vos variables contiennent quelque chose de cohérent.
  + Ce conseil vaut aussi pour les paramètres des fonctions
* Est-ce que votre requête à la BD fonctionne ?
  + Tester votre requête dans PHPMyAdmin pour valider que vous obtenez un bon résultat.
  + Si votre requête est construite dynamiquement ou qu’elle contient des variables, valider qu’elle se construisent bien à l’aide d’un echo.