Structurer ses applications avec le pattern MVC

1. Objectif

- **Vous apprendrez à :**
  - Comprendre le pattern MVC (Modèle – Vue – Contrôleur) appliqué à PHP.
  - Organiser correctement un projet PHP en architecture MVC.
  - Configurer une connexion sécurisée à une base de données MySQL avec PDO.
  - Séparer la logique métier, l'accès aux données et l'affichage.
  - Utiliser un **Autoloader** pour charger les classes automatiquement sans `require_once`.
  - Utiliser un fichier **helpers.php** pour générer des URLs compatibles avec tout environnement.
  - Créer un contrôleur capable de récupérer des données depuis un modèle.
  - Afficher les données dans une vue sans écrire de SQL.
  - Comprendre le trajet d'une requête HTTP dans une application MVC.

2. Présentation

1. L'analogie du Restaurant
   - Imaginez un restaurant. Pour que cela fonctionne bien, on sépare les tâches entre trois personnes :
     - **Le Serveur (Le Contrôleur)**: C'est lui qui parle au client. Il prend la commande et dit aux autres quoi faire. Il ne cuisine pas lui-même.
     - **Le Cuisinier (Le Modèle)** : Il est dans la cuisine. Il manipule la nourriture (les données). C'est lui qui sait comment préparer le plat.
     - **L'Assiette Dressée (La Vue)**: C'est ce que le client voit et mange. C'est la présentation finale, toute belle.
   - **Les 3 lettres expliquées simplement**
   - Voici ce que chaque lettre signifie techniquement, mais avec des mots simples :
   - **M** comme **Modèle** (Le Cerveau / La Cuisine)
     - C'est la partie qui gère les données.

apcpedagogie.com

        - Si vous avez un site de météo, le Modèle va chercher la température dans la base de données.
        - Il ne sait pas à quoi ressemble le site, il s'occupe juste des calculs et de l'information brute.
    - **V** comme **Vue** (Le Visuel / L'Assiette)
        - C'est ce que l'utilisateur voit à l'écran.
        - C'est le code HTML et CSS.
        - La Vue est "bête" : elle attend juste qu'on lui donne des informations pour les afficher joliment.
    - **C** comme **Contrôleur** (Le Chef d'orchestre / Le Serveur)
        - C'est le lien entre l'utilisateur, le Modèle et la Vue.
        - L'utilisateur clique sur un bouton ? Le Contrôleur reçoit l'ordre.
        - Il demande au Modèle : "Donne-moi les infos".
        - Puis il donne ces infos à la Vue en disant : "Affiche ça".
    - **Pourquoi on s'embête à faire ça ?**
        - Si vos apprenants demandent "Pourquoi ne pas tout mettre dans un seul fichier ?", voici les deux arguments choc :
        - Le rangement : Si on veut changer la décoration du restaurant (la Vue), on n'a pas besoin de changer la recette de cuisine (le Modèle).
        - Le travail d'équipe : Un graphiste peut travailler sur la Vue pendant qu'un développeur travaille sur le Modèle, sans se marcher sur les pieds.

2. Le **pattern MVC** (Model – View – Controller)
    - Le **pattern MVC** (Model – View – Controller) est une architecture de développement utilisée pour structurer proprement une application web.
    - En PHP, MVC permet de **séparer clairement les responsabilités**:
        - **Le Modèle** s'occupe des données et de la base de données.
        - **La Vue** se charge uniquement de l'affichage HTML.
        - **Le Contrôleur** fait le lien entre le modèle et la vue.
    - ? **Objectif principal du MVC :** rendre le code plus lisible, plus maintenable et plus facile à faire évoluer.
    - Dans ce tutoriel, nous allons créer une **mini-application PHP MVC** sans framework, en utilisant : **PHP**, **PDO** et **MySQL**. Nous y ajoutons deux fichiers essentiels absents des tutoriels classiques : l'**Autoloader** et le fichier **helpers.php**.

apcpedagogie.com

3. Le concept MVC (Model – View – Controller)

- **Modèle (Model)**
  - Gère la connexion à la base de données.
  - Contient les requêtes SQL (SELECT, INSERT, UPDATE, DELETE).
  - Ne contient aucun code HTML.
- **Vue (View)**
  - Contient le HTML et l'affichage des données.
  - Affiche les informations transmises par le contrôleur.
  - Ne contient aucune requête SQL.
- **Contrôleur (Controller)**
  - Reçoit la requête de l'utilisateur.
  - Appelle les méthodes du modèle.
  - Transmet les données à la vue.
- **Règle d'or :** *Le contrôleur décide, le modèle travaille, la vue affiche.*

4. Architecture du projet MVC

- Le projet est organisé selon une architecture MVC claire et sécurisée, avec un point d'entrée unique (`public/index.php`). Deux fichiers supplémentaires sont ajoutés dans `core/`: `Autoloader.php` et `helpers.php`.

```
mvc-users/
│
├── app/
│   ├── controllers/        ← ? Contrôleurs (UserController.php)
│   ├── models/             ← ? Modèles (User.php)
│   └── views/              ← ?  Vues (index.php, create.php, edit.php)
│       └── users/
│
├── config/                 ← ⚙  Configuration base de données
│   └── database.php
│
├── core/                   ← ? Classes principales du framework maison
│   ├── Autoloader.php      ←  Chargement automatique des classes
│   ├── Controller.php      ← Classe parente des contrôleurs
│   ├── helpers.php         ←  Fonctions utilitaires (génération
d'URLs)
│   ├── Model.php           ← Connexion PDO centralisée
```

apcpedagogie.com

```
│   └── Router.php          ← Routeur URL → contrôleur/action
│
├── public/                 ← ? Seul dossier accessible depuis le
navigateur
│   ├── assets/
│   │   └── css/
│   │       └── style.css   ← Feuille de style
│   └── index.php           ← Point d'entrée unique (Front Controller)
│
├── routes/
│   └── web.php             ← Déclaration des routes de l'application
│
├── .htaccess               ← Redirection de toutes les URLs vers
public/index.php
└── README.md
```

5. Rôle de chaque dossier et fichier

- **public/**
  - Seul dossier accessible depuis le navigateur.
  - Contient le fichier `index.php` (Front Controller) et les assets CSS/JS.
  - Démarre toute l'application MVC.
- **app/models/**
  - Contient les classes représentant les données.
  - Utilise PDO pour communiquer avec MySQL.
  - Exemple : `User.php` avec les méthodes `getAll()`, `create()`, `update()`, `delete()`.
- **app/controllers/**
  - Contient la logique de traitement des requêtes.
  - Appelle les méthodes du modèle.
  - Charge les vues avec les données.
- **app/views/**
  - Contient les fichiers HTML/PHP d'affichage.
  - Affiche les données reçues du contrôleur.
  - N'effectue aucun accès direct à la base de données.
- **config/**
  - Contient les paramètres de connexion à la base de données.
  - À ne jamais versionner en production (ajouter à `.gitignore`).
- **routes/**
  - Déclare toutes les URLs disponibles dans l'application.
  - Associe chaque URL à un contrôleur et une action.

apcpedagogie.com

- **core/Autoloader.php** ← *nouveauté importante*
  - Charge automatiquement toutes les classes PHP sans `require_once`.
  - Évite les erreurs d'oubli et allège le code.
  - S'enregistre via la fonction native PHP `spl_autoload_register()`.
- **core/helpers.php** ← *nouveauté importante*
  - Contient des fonctions utilitaires disponibles dans toute l'application.
  - La fonction `url()` génère des URLs correctes que l'application soit à la racine du serveur ou dans un sous-dossier (ex : `/mvc-users/`).

3. Base de données

```
CREATE DATABASE IF NOT EXISTS `mvc_app`
CHARACTER SET utf8mb4
COLLATE utf8mb4_unicode_ci;
```

```
CREATE TABLE IF NOT EXISTS `users` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(100) NOT NULL,
  `email` VARCHAR(150) NOT NULL,
  PRIMARY KEY (`id`),
  UNIQUE KEY `email` (`email`)
) ENGINE=InnoDB
DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_unicode_ci;
```

4. Fichier `.htaccess`

```
RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule ^(.*)$ public/index.php [QSA,L]
```

5. Fichier `README.md`

README.md

apcpedagogie.com

```
# mvc-users — Application MVC PHP / PDO / MySQL

Application CRUD complète construite en PHP pur selon le pattern MVC, sans
framework.

## Prérequis
- PHP >= 7.4
- MySQL / MariaDB
- Apache avec mod_rewrite activé (XAMPP, WAMP, Laragon...)

## Installation

### 1. Créer la base de données
```sql
CREATE DATABASE mvc_app CHARACTER SET utf8mb4 COLLATE utf8mb4_unicode_ci;
USE mvc_app;

CREATE TABLE users (
    id    INT AUTO_INCREMENT PRIMARY KEY,
    name  VARCHAR(100) NOT NULL,
    email VARCHAR(150) NOT NULL UNIQUE
);

INSERT INTO users (name, email) VALUES
    ('Alice Martin', 'alice@example.com'),
    ('Bob Dupont',   'bob@example.com');


### 2. Configurer la connexion
Modifier `config/database.php` selon votre environnement.

### 3. Placer le projet dans le dossier web
- XAMPP : C:/xampp/htdocs/mvc-users/
- WAMP  : C:/wamp64/www/mvc-users/

### 4. Accéder à l'application
http://localhost/users

## Routes

| Méthode | URL                | Action                      |
|---------|--------------------|-----------------------------|
```

apcpedagogie.com
```

```
| GET     | /users             | Liste des utilisateurs     |
| GET     | /users/create      | Formulaire de création     |
| POST    | /users/store       | Enregistrer                |
| GET     | /users/edit/{id}   | Formulaire de modification |
| POST    | /users/update/{id} | Mettre à jour              |
| GET     | /users/delete/{id} | Supprimer                  |
```

6. Le dossier appp

1. Le fichier `controllers/UserController.php`

UserController.php

```php
<?php

// □ Plus besoin de require_once grâce à l'Autoloader
// require_once '../core/Controller.php';  ← supprimé
// require_once '../app/models/User.php';  ← supprimé

/**
 * Classe UserController – Gère toutes les actions CRUD sur les utilisateurs.
 */

class UserController extends Controller
{
    // GET /users
    public function index()
    {
        $model = new User();
        $users = $model->getAll();
        require '../app/views/users/index.php';
    }

    // GET /users/create
    public function create()
    {
        require '../app/views/users/create.php';
    }

    // POST /users/store
    public function store()
```

apcpedagogie.com

```
    {
        $name  = trim($_POST['name']  ?? '');
        $email = trim($_POST['email'] ?? '');

        if (!empty($name) && filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $model = new User();
            $model->create($name, $email);
        }

        $this->redirect('/users');
    }

    // GET /users/edit/{id}
    public function edit($id)
    {
        $model = new User();
        $data  = $model->getById($id);

        if (!$data) {
            $this->redirect('/users');
        }

        require '../app/views/users/edit.php';
    }

    // POST /users/update/{id}
    public function update($id)
    {
        $name  = trim($_POST['name']  ?? '');
        $email = trim($_POST['email'] ?? '');

        if (!empty($name) && filter_var($email, FILTER_VALIDATE_EMAIL)) {
            $model = new User();
            $model->update($id, $name, $email);
        }

        $this->redirect('/users');
    }

    // GET /users/delete/{id}
    public function delete($id)
    {
        $model = new User();
```

apcpedagogie.com

```
        $model->delete($id);
        $this->redirect('/users');
    }
}
```

2. **Le fichier `models/User.php`**

User.php

```
<?php

// □ Plus besoin de require_once grâce à l'Autoloader
// require_once '../core/Model.php';  ← supprimé

/**
 * Classe User — Modèle représentant la table `users`.
 * Contient UNIQUEMENT les opérations SQL.
 */
class User extends Model
{
    public function getAll()
    {
        return $this->db
                    ->query("SELECT * FROM users ORDER BY id DESC")
                    ->fetchAll(PDO::FETCH_ASSOC);
    }

    public function getById($id)
    {
        $stmt = $this->db->prepare("SELECT * FROM users WHERE id = ?");
        $stmt->execute([$id]);
        return $stmt->fetch(PDO::FETCH_ASSOC);
    }

    public function create($name, $email)
    {
        $stmt = $this->db->prepare(
            "INSERT INTO users (name, email) VALUES (?, ?)"
        );
        return $stmt->execute([$name, $email]);
    }
```

apcpedagogie.com

```
    public function update($id, $name, $email)
    {
        $stmt = $this->db->prepare(
            "UPDATE users SET name = ?, email = ? WHERE id = ?"
        );
        return $stmt->execute([$name, $email, $id]);
    }

    public function delete($id)
    {
        $stmt = $this->db->prepare("DELETE FROM users WHERE id = ?");
        return $stmt->execute([$id]);
    }
}
```

3. **Le fichier `views/users/index.php`**

index.php

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Liste des utilisateurs</title>
    <link rel="stylesheet" href="/mvc-users/public/assets/css/style.css">
</head>
<body>
<div class="container">

    <h1>? Liste des utilisateurs</h1>

    <a href="/mvc-users/public/users/create" class="btn btn-primary">□ Ajouter
un utilisateur</a>

    <table>
        <thead>
            <tr>
                <th>ID</th>
                <th>Nom</th>
                <th>Email</th>
                <th>Actions</th>
```

apcpedagogie.com

```
            </tr>
        </thead>
        <tbody>

        <?php if (empty($users)): ?>
            <tr>
                <td colspan="4" class="empty-msg">Aucun utilisateur
trouvé.</td>
            </tr>
        <?php else: ?>
            <?php foreach ($users as $user): ?>
            <tr>
                <td data-label="ID"><?= (int)$user['id'] ?></td>
                <td data-label="Nom"><?= htmlspecialchars($user['name'],
ENT_QUOTES, 'UTF-8') ?></td>
                <td data-label="Email"><?= htmlspecialchars($user['email'],
ENT_QUOTES, 'UTF-8') ?></td>
                <td data-label="Actions">
                    <a href="<?= url('users/edit/' . (int)$user['id']) ?>"
class="btn btn-warning">✏ Modifier</a>
                    <a href="<?= url('users/delete/' . (int)$user['id']) ?>"
                       class="btn btn-danger"
                       onclick="return confirm('Confirmer la suppression ?')">
                       ? Supprimer
                    </a>
                </td>
            </tr>
            <?php endforeach; ?>
        <?php endif; ?>

        </tbody>
    </table>

</div>
</body>
</html>
```

4. **Le fichier `views/users/edit.php`**

edit.php

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Modifier l'utilisateur</title>
     <link rel="stylesheet" href="/mvc-users/public/assets/css/style.css">
</head>
<body>
<div class="container">

    <h1>✏ Modifier l'utilisateur</h1>

    <div class="form-card">
        <form method="post" action="<?= url('users/update/' .
(int)$data['id']) ?>">

            <div class="form-group">
                <label for="name">Nom</label>
                <input type="text"
                       id="name"
                       name="name"
                       value="<?= htmlspecialchars($data['name'], ENT_QUOTES,
'UTF-8') ?>"
                       required
                       maxlength="100">
            </div>

            <div class="form-group">
                <label for="email">Email</label>
                <input type="email"
                       id="email"
                       name="email"
                       value="<?= htmlspecialchars($data['email'], ENT_QUOTES,
'UTF-8') ?>"
                       required
                       maxlength="150">
            </div>

            <div class="form-actions">
                <button type="submit" class="btn btn-primary">✏ Mettre à
jour</button>
```

apcpedagogie.com

```
                <a href="/users" class="btn btn-secondary">Annuler</a>
            </div>

        </form>
    </div>

</div>
</body>
</html>
```

5. **Le fichier `views/users/create.php`**

create.php

```
<!DOCTYPE html>
<html lang="fr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ajouter un utilisateur</title>
     <link rel="stylesheet" href="/mvc-users/public/assets/css/style.css">
</head>
<body>
<div class="container">

    <h1> Ajouter un utilisateur</h1>

    <div class="form-card">
        <form method="post" action="<?= url('users/store') ?>">

            <div class="form-group">
                <label for="name">Nom</label>
                <input type="text"
                       id="name"
                       name="name"
                       placeholder="Ex : Alice Martin"
                       required
                       maxlength="100">
            </div>

            <div class="form-group">
                <label for="email">Email</label>
```

apcpedagogie.com

```
                <input type="email"
                       id="email"
                       name="email"
                       placeholder="Ex : alice@example.com"
                       required
                       maxlength="150">
            </div>

            <div class="form-actions">
                <button type="submit" class="btn btn-primary">?
Enregistrer</button>
                <a href="/users" class="btn btn-secondary">Annuler</a>
            </div>

        </form>
    </div>

</div>
</body>
</html>
```

7. Le dossier config

1. Le fichier `config/database.php`

database.php

```
<?php

// Retourne un tableau de configuration pour la connexion PDO
return [
    'host'     => 'localhost',
    'dbname'   => 'mvc_app',
    'user'     => 'root',
    'password' => ''
];
```

apcpedagogie.com

8. Le dossier core

1. Le fichier `core/Autoloader.php`

- Sans Autoloader, chaque fichier PHP doit inclure manuellement toutes les classes dont il a besoin. Cela produit un code répétitif et difficile à maintenir :

```
<?php
// ❌ Sans Autoloader : require_once partout dans chaque fichier
require_once '../core/Model.php';
require_once '../core/Controller.php';
require_once '../app/models/User.php';
require_once '../app/controllers/UserController.php';
```

- Avec l'Autoloader, une seule ligne dans `public/index.php` suffit. PHP chargera automatiquement chaque classe au moment où elle est utilisée :

```
<?php
// ✅ Avec Autoloader : une seule ligne dans public/index.php
require_once '../core/Autoloader.php';
Autoloader::register();
// Désormais toutes les classes (Model, Controller, User, UserController...)
// sont chargées automatiquement dès qu'elles sont instanciées.
```

- **Contenu de `core/Autoloader.php`:**
Autoloader.php

```
<?php
/**
 * Classe Autoloader — Chargement automatique des classes PHP.
 * Fonctionne sans namespace : cherche le fichier NomClasse.php
 * dans les dossiers déclarés.
 */
class Autoloader
{
    /**
     * Enregistre l'autoloader auprès de PHP via spl_autoload_register().
     * À appeler une seule fois dans public/index.php
```

apcpedagogie.com

```
     */
    public static function register()
    {
        spl_autoload_register(function ($className) {

            // Racine du projet (un niveau au-dessus de /public)
            $baseDir = dirname(__DIR__) . '/';

            // Liste COMPLÈTE des dossiers où chercher les classes
            $directories = [
                $baseDir . 'core/',               // Model.php,
Controller.php, Router.php
                $baseDir . 'app/models/',          // User.php
                $baseDir . 'app/controllers/',     // UserController.php
            ];

            foreach ($directories as $dir) {
                $file = $dir . $className . '.php';

                if (file_exists($file)) {
                    require_once $file;
                    return; // Classe trouvée → on arrête
                }
            }

            // Classe introuvable dans tous les dossiers → erreur explicite
            throw new Exception("Classe introuvable : {$className}. Vérifiez
le nom du fichier et le dossier.");
        });
    }
}
```

2. **Le fichier `core/Controller.php`**

Controller.php

```
<?php

/**
 * Classe Controller – Classe parente des contrôleurs.
 */
class Controller
```

```
{
    protected function redirect($url)
    {
        header("Location: " . $url);
        exit;
    }
}
```

3. **Le fichier `core/helpers.php`**

   - Pourquoi un fichier helpers.php ?
     - Quand une application PHP est déployée dans un **sous-dossier** (ex : `http://localhost/mvc-users/`), les liens absolus comme `/users/create` ne fonctionnent pas — ils pointent vers `http://localhost/users/create` au lieu de `http://localhost/mvc-users/users/create`.
     - Le fichier `core/helpers.php` contient la fonction `url()` qui résout ce problème en détectant automatiquement le sous-dossier :

       helpers.php

```
<?php

/**
 * Génère une URL correcte selon que l'app est
 * en sous-dossier ou à la racine.
 */
function url($path = '')
{
    // Récupère le sous-dossier : /mvc-users
    $basePath = dirname(dirname($_SERVER['SCRIPT_NAME']));
    if ($basePath === '/' || $basePath === '') {
        $basePath = '';
    }
    return $basePath . '/' . ltrim($path, '/');
}
```

   - **Le fichier `core/Models.php`**

     Models.php

apcpedagogie.com

```
<?php

/**
 * Classe Model – Classe parente de tous les modèles.
 * Centralise la connexion PDO.
 */
class Model
{
    protected $db;

    public function __construct()
    {
        $config = require '../config/database.php';

        $this->db = new PDO(

"mysql:host={$config['host']};dbname={$config['dbname']};charset=utf8mb4",
            $config['user'],
            $config['password']
        );

        $this->db->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
        $this->db->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE,
PDO::FETCH_ASSOC);
    }
}
```

- **Le fichier `core/Routers.php`**

Routers.php

```
<?php

/**
 * Classe Router – Analyse l'URL et dispatche vers le bon contrôleur et
la bonne action.
 * Compatible avec un déploiement en sous-dossier (ex: /mvc-users/) ou à
la racine.
 */
class Router
```

apcpedagogie.com

```
{
    private $routes = [];

    /**
     * Enregistre une route GET.
     */
    public function get($path, $callback)
    {
        $this->routes['GET ' . $path] = $callback;
    }

    /**
     * Enregistre une route POST.
     */
    public function post($path, $callback)
    {
        $this->routes['POST ' . $path] = $callback;
    }

    /**
     * Démarre le routeur : compare l'URL courante aux routes
enregistrées.
     */
    public function run()
    {
        $method = $_SERVER['REQUEST_METHOD'];

        // URI complète ex: /mvc-users/users
        $uri = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);

        // Supprime le sous-dossier du début de l'URI
        // SCRIPT_NAME = /mvc-users/public/index.php
        // dirname x2  = /mvc-users
        // URI finale  = /users
        $basePath = dirname(dirname($_SERVER['SCRIPT_NAME']));
        if ($basePath !== '/') {
            $uri = '/' . ltrim(substr($uri, strlen($basePath)), '/');
        }

        foreach ($this->routes as $route => $callback) {
            [$routeMethod, $routePath] = explode(' ', $route, 2);

            // Convertit {id} en groupe de capture regex
```

apcpedagogie.com

```
            $pattern = preg_replace('/\{[a-z]+\}/', '([^/]+)',
$routePath);
            $pattern = '#^' . $pattern . '$#';

            if ($routeMethod === $method && preg_match($pattern, $uri,
$matches)) {
                array_shift($matches);
                call_user_func_array($callback, $matches);
                return;
            }
        }

        // Aucune route trouvée
        http_response_code(404);
        echo "<h1>404 – Page non trouvée</h1>";
    }
}
```

9. Le dossier public

1. Le fichier `public/index.php`

- Toutes les requêtes HTTP passent par ce fichier unique grâce au `.htaccess`. Il charge l'Autoloader, les helpers, puis démarre le routeur :
  // 1. Chargement de l'Autoloader // Toutes les classes seront chargées automatiquement require_once '../core/Autoloader.php'; Autoloader::register(); // 2. Chargement des fonctions utilitaires (url(), etc.) require_once '../core/helpers.php'; // 3. Instanciation du routeur $router = new Router(); // 4. Chargement des routes déclarées dans routes/web.php require_once '../routes/web.php'; // 5. Démarrage : analyse l'URL et dispatche vers le bon contrôleur $router->run();

index.php

```
<?php
require_once '../core/Autoloader.php';
Autoloader::register();

// Chargement des fonctions helpers
require_once '../core/helpers.php';
```

apcpedagogie.com

```
$router = new Router();
require_once '../routes/web.php';
$router->run();
```

2. **Le fichier `public/assets/css/style.css`**

style.css

```
/* ============================================================
   mvc-users — Feuille de style globale
   ============================================================ */

/* ---------- Reset de base ---------- */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f4f6f9;
    color: #333;
    min-height: 100vh;
}

/* ---------- Conteneur principal ---------- */
.container {
    max-width: 900px;
    margin: 40px auto;
    padding: 0 20px;
}

/* ---------- Titres ---------- */
h1 {
    font-size: 1.8rem;
    margin-bottom: 20px;
    color: #2c3e50;
    border-left: 4px solid #3498db;
    padding-left: 12px;
}
```

apcpedagogie.com

```
/* ---------- Lien Ajouter ---------- */
.btn {
    display: inline-block;
    padding: 8px 16px;
    border-radius: 5px;
    text-decoration: none;
    font-size: 0.9rem;
    cursor: pointer;
    border: none;
    margin-bottom: 16px;
    transition: background 0.2s ease;
}

.btn-primary {
    background-color: #3498db;
    color: #fff;
}

.btn-primary:hover {
    background-color: #2980b9;
}

.btn-secondary {
    background-color: #95a5a6;
    color: #fff;
}

.btn-secondary:hover {
    background-color: #7f8c8d;
}

.btn-danger {
    background-color: #e74c3c;
    color: #fff;
}

.btn-danger:hover {
    background-color: #c0392b;
}

.btn-warning {
    background-color: #f39c12;
    color: #fff;
```

apcpedagogie.com

```
}

.btn-warning:hover {
    background-color: #d68910;
}

/* ---------- Tableau ---------- */
table {
    width: 100%;
    border-collapse: collapse;
    background-color: #fff;
    border-radius: 8px;
    overflow: hidden;
    box-shadow: 0 2px 8px rgba(0,0,0,0.08);
}

thead {
    background-color: #2c3e50;
    color: #fff;
}

thead th {
    padding: 12px 16px;
    text-align: left;
    font-weight: 600;
    font-size: 0.9rem;
    text-transform: uppercase;
    letter-spacing: 0.05em;
}

tbody tr {
    border-bottom: 1px solid #ecf0f1;
    transition: background 0.15s ease;
}

tbody tr:last-child {
    border-bottom: none;
}

tbody tr:hover {
    background-color: #f0f4f8;
}
```

apcpedagogie.com

```
tbody td {
    padding: 12px 16px;
    font-size: 0.95rem;
}

/* ---------- Formulaires ---------- */
.form-card {
    background-color: #fff;
    border-radius: 8px;
    padding: 30px;
    box-shadow: 0 2px 8px rgba(0,0,0,0.08);
    max-width: 480px;
}

.form-group {
    margin-bottom: 18px;
}

label {
    display: block;
    font-weight: 600;
    margin-bottom: 6px;
    font-size: 0.9rem;
    color: #555;
}

input[type="text"],
input[type="email"] {
    width: 100%;
    padding: 10px 12px;
    border: 1px solid #ccc;
    border-radius: 5px;
    font-size: 0.95rem;
    transition: border-color 0.2s ease;
    outline: none;
}

input[type="text"]:focus,
input[type="email"]:focus {
    border-color: #3498db;
    box-shadow: 0 0 0 3px rgba(52,152,219,0.15);
}
```

apcpedagogie.com

```
.form-actions {
    display: flex;
    gap: 10px;
    margin-top: 24px;
}

/* ---------- Message vide ---------- */
.empty-msg {
    text-align: center;
    color: #999;
    font-style: italic;
    padding: 24px;
}

/* ---------- Responsive ---------- */
@media (max-width: 600px) {
    table, thead, tbody, th, td, tr {
        display: block;
    }

    thead {
        display: none;
    }

    tbody tr {
        background: #fff;
        border-radius: 8px;
        margin-bottom: 12px;
        padding: 12px;
        box-shadow: 0 1px 4px rgba(0,0,0,0.1);
    }

    tbody td {
        padding: 6px 0;
        font-size: 0.9rem;
    }

    tbody td::before {
        content: attr(data-label) " : ";
        font-weight: bold;
        color: #2c3e50;
    }
}
```

apcpedagogie.com

10. **Le dossier routes**

    1. **Le fichier `routes/web.php`**

       web.php

```
<?php

// □ Plus besoin de require_once grâce à l'Autoloader
// require_once '../app/controllers/UserController.php';  ← supprimé

$controller = new UserController();

$router->get('/users', function() use ($controller) {
    $controller->index();
});

$router->get('/users/create', function() use ($controller) {
    $controller->create();
});

$router->post('/users/store', function() use ($controller) {
    $controller->store();
});

$router->get('/users/edit/{id}', function($id) use ($controller) {
    $controller->edit($id);
});

$router->post('/users/update/{id}', function($id) use ($controller) {
    $controller->update($id);
});

$router->get('/users/delete/{id}', function($id) use ($controller) {
    $controller->delete($id);
});
```

- ## Flux d'exécution d'une requête

  - L'utilisateur tape une URL dans le navigateur (ex : `http://localhost/mvc-users/users`).
  - Le `.htaccess` redirige la requête vers `public/index.php`.

apcpedagogie.com

- `index.php` charge l'Autoloader et les helpers, puis instancie le Router.
- Le Router supprime le préfixe `/mvc-users` et obtient l'URI propre `/users`.
- Le Router trouve la route correspondante et appelle `UserController::index()`.
- Le contrôleur instancie `User` (chargé automatiquement par l'Autoloader).
- Le modèle exécute la requête SQL et retourne les données.
- Le contrôleur transmet les données à la vue.
- La vue génère le HTML avec les données et l'envoie au navigateur.
- **Important :** À aucun moment la vue n'accède directement à la base de données.

```
Navigateur  →  http://localhost/mvc-users/users
     │
     ▼
.htaccess   →  Redirige vers public/index.php
     │
     ▼
public/index.php
     │  1. Autoloader::register()   → charge les classes automatiquement
     │  2. require helpers.php      → fonction url() disponible partout
     │  3. new Router()
     │  4. require routes/web.php
     │  5. $router->run()
     │
     ▼
core/Router.php
     │  URI reçue  : /mvc-users/users
     │  Après fix  : /users
     │  Route match: GET /users → UserController::index()
     │
     ▼
app/controllers/UserController.php  →  index()
     │  new User()   (chargé par Autoloader)
     │
     ▼
app/models/User.php  →  getAll()
     │  SELECT * FROM users
     │  Retourne $users
     │
     ▼
app/views/users/index.php
     │  Affiche le tableau HTML avec $users
     │  url('users/create') → /mvc-users/users/create
```

apcpedagogie.com

```
     |
     ▼
Navigateur  ←  Page HTML finale
```

- ## À retenir

  - Le **Modèle** contient tout le SQL – rien d'autre.
  - Le **Contrôleur** reçoit la requête, appelle le modèle, choisit la vue.
  - La **Vue** affiche les données – aucune logique métier, aucun SQL.
  - L'**Autoloader** charge les classes automatiquement – plus de `require_once` dispersés.
  - Le fichier **helpers.php** avec la fonction `url()` garantit des liens corrects en sous-dossier comme à la racine.
  - Le **Router** supprime le préfixe du sous-dossier pour que les routes restent propres (`/users` et non `/mvc-users/users`).
  - Pas de SQL dans les vues.
  - Pas de HTML dans les modèles.
  - Pas de liens en dur avec `/chemin` absolu sans passer par `url()`.
  - **MVC = structure claire, maintenable et professionnelle.**

Cours PHP

Source : https://apcpedagogie.com/structurer-ses-applications-avec-le-pattern-mvc · téléchargé le 07/11/2026 · © APC Pédagogie · apcpedagogie.com

apcpedagogie.com