MVC cheat

Glossaire

Entité

Une entité, c’est juste la représentation d’un objet ou d’un concept du monde réel dans ton application.

Logique métier

La logique métier, c’est l’ensemble des règles et processus qui définissent comment ton application doit fonctionner selon les besoins spécifiques d’une entreprise ou d’un domaine.

C’est ce qui dicte comment les données doivent être manipulées et utilisées pour répondre aux objectifs fonctionnels.

Et ça, peu importe les technologies que tu utilises.

Instancier / Instanciation

L'instanciation, c’est quand tu crées une instance d’une classe. Par exemple, si tu as une classe `Voiture`, instancier cette classe va créer un objet spécifique qui représente une voiture, avec des caractéristiques précises.

Héritage

L’héritage, c’est un concept où une classe (la "classe fille") hérite des propriétés et des méthodes d’une autre classe (la "classe mère"). Pratique pour éviter de tout réécrire !

Classe abstraite

Une classe abstraite, c’est une classe qui ne peut pas être instanciée directement. Elle sert de modèle pour d'autres classes qui vont l’hériter.

Les parties clés d’un MVC

Routeur

À quoi ça sert ?

Le `Routeur` gère les routes dans une application PHP MVC. En gros, il associe les URLs aux actions des contrôleurs. C'est lui qui fait le lien entre l'URL que tu tapes et l'action qui doit être exécutée.

Validator

À quoi ça sert ?

Le `Validator` sert à valider les données avant qu'elles ne soient insérées dans la base de données.

Pourquoi valider les données ?

Parce qu'on veut s'assurer que les données sont **cohérentes, complètes et sécurisées**.

- On veut éviter les erreurs comme des emails mal formatés ou des mots de passe trop courts.
- Cela permet aussi de prévenir des attaques comme les injections SQL.
- Et bien sûr, ça protège la base de données en garantissant l'intégrité des données.

En résumé, le `Validator` permet de :

1. Prévenir les erreurs de saisie.
2. Assurer que les données respectent les règles métier.
3. Éviter les failles de sécurité.
4. Assurer la cohérence des données.

Database

À quoi ça sert ?

La classe `Database` sert à configurer la connexion à la base de données et à interagir avec elle.

Elle :

1. Récupère les infos de connexion (hôte, base de données, utilisateur, mot de passe) depuis des variables d'environnement.

2. Crée une connexion avec la base via PDO.
3. Gère les erreurs en cas de problème de connexion.
4. Fournit l'accès à la connexion via la méthode `getConnection()`.

Les parties indispensables du MVC

Controllers

Les **contrôleurs**, c'est la logique de traitement des requêtes dans l'application.

Il y a deux types dans notre MVC :

1. **La classe abstraite** `Controller` : C'est une classe qui ne sera jamais instanciée directement. Tous les autres contrôleurs vont hériter de cette classe. Elle contient des méthodes utiles pour afficher les vues.
2. **Les contrôleurs concrets** : Ce sont ceux qui héritent de `Controller` et qui contiennent les actions spécifiques à chaque partie de l'application (par exemple, gérer les menus dans un restaurant).

À quoi ça sert ?

Les contrôleurs sont là pour :

- Recevoir les requêtes des utilisateurs,
- Les traiter en fonction de la logique métier,
- Retourner une réponse appropriée, comme afficher une vue.

Par exemple, le `MenuController` dans une application de restaurant va s'occuper des actions liées au menu, comme afficher les menus et gérer les commandes des utilisateurs.

Model

À quoi ça sert ?

Les **modèles** sont là pour représenter les entités de l'application et leur logique métier.

Les modèles sont responsables de la gestion des données et de l'interaction avec la base de données.

Fonctionnalités des modèles :

1. **Gestion des attributs** : Les modèles définissent les propriétés des entités.
2. **Validation des données** : Avant de stocker des données, le modèle les valide (par exemple, vérifier que l'email est bien formaté).
3. **Vérifications supplémentaires** : Par exemple, vérifier si l'email existe déjà en base de données.
4. **Accès aux données** : Les modèles permettent de récupérer ou manipuler les données sous forme d'objets ou de tableaux.
5. **Interaction avec la base de données** : Bien qu'un modèle ne gère pas directement la base (ça c'est le rôle de `Database`), il effectue les opérations comme l'insertion, la mise à jour, la suppression, etc.
6. **Gestion des erreurs** : Les modèles gèrent les erreurs de validation et permettent de les récupérer.

Voici un exemple avec la classe `User` qui représente un utilisateur :

```
php
Copier
<?php

namespace App\models;

use App\config\Database;

class User extends Database
{
    private $idClient;
    private $firstName;
    private $lastName;
    private $email;
    private $mdp;
    private $mdpValid;
    private $role = 'client';
    private $isValid = true;
    private $errors = array();
```

```
    // Méthodes pour définir les attributs (setters)...
    public function setFirstName($firstName)
    {
        // Validation ici...
    }

    // Vérification des mots de passe...
    public function checkMatchTwoPassword($arg1, $arg2)
    {
        if ($arg1 !== $arg2) {
            $this->isValid = false;
            return $this->errors += ["passwordMatch" => "Les mots de passe ne
correspondent pas"];
        }
    }

    public function checkIsValid()
    {
        return $this->isValid;
    }

    // Méthodes pour récupérer les données (getters)...
    public function getData()
    {
        return ['prenom' => $this->firstName, 'nom' => $this->lastName, 'email'
=> $this->email, 'mdp' => $this->mdp, 'role' => $this->role];
    }
}
```

Cette classe représente un utilisateur et gère tout ce qui est lié à ses informations personnelles : nom, prénom, email, mot de passe, etc. Elle permet de valider ces informations et de les enregistrer correctement en base.