Travail de recherche

PHP

Par DUPIN ALEXIS

BACHELOR 2 CYBERSECURITE

SOMMAIRE

Partie de recherche & préparation au développement PHP

Architecture MVC :

- **Qu'est-ce que l'architecture MVC (Modèle – Vue – Contrôleur) ?**

  Il s'agit d'un motif d'architecture logicielle.

  Il est composé de trois types de modules ayant trois responsabilités différentes : les modèles, les vues et les contrôleurs.

  - Un modèle contient les données à afficher
  - Une vue contient l'interface graphique
  - Un contrôleur contient la logique des actions utilisateur.

- **Quel est le rôle de chaque composant (Modèle, Vue, Contrôleur) dans une application PHP ?**

  Modèle : Il représente la couche des données et la logique métier de l'application. Il gère l'accès aux données et les interactions avec la base de données. Il ne s'occupe pas de l'affichage des données. Il se focus sur le traitement de celle-ci.

  Vue : La vue est responsable de l'affichage des données aux utilisateurs. Elle récupère les données du modèle et les présente dans un format compréhensible (souvent HTML). Vue ne contient aucune logique métier.

Contrôleur : Le contrôleur agit en intermédiaire entre le modèle et la vue. Il reçoit les requêtes client, interagit avec le modèle pour récupérer les données nécessaires. Il transmet ces données à la vue pour les afficher. Le contrôleur gère le flux de l'application.

- **Expliquez le cheminement d'une requête HTTP dans une application MVC.**

  Tout d'abord, l'utilisateur fait une requête qui est envoyée au contrôleur. Le contrôleur fait suivre cette requête au modèle qui va traiter la demande. Le model envoie ensuite les données nécessaires au contrôleur qui va les transcrire en html, puis le contrôleur va les faire suivre à la partie vue. La vue va ensuite envoyer les données rendue compréhensible à l'utilisateur au contrôleur qui donnera la réponse à la requête utilisateur.

- **Pourquoi les requêtes SQL ne doivent-elles pas se trouver dans les vues ?**

  Placer des requêtes SQL directement dans les vues n'est pas logiquement viable, car l'interface ne doit gérer que l'affichage et non l'accès aux données. Cela rend le code difficile à maintenir et répétitif, car toute modification de la structure de la base oblige à corriger plusieurs fichiers d'interface. Elle expose également l'application à des risques majeurs de sécurité, notamment des injections SQL, en contournant les protections en back-end.

Front Controller et Routage :

- **Qu'est-ce qu'un Front Controller et quel est son rôle dans une application PHP ?**

  Un front controller consiste à avoir un point d'entrée unique pour l'application web (index.php par exemple) qui gère toutes les requêtes. Ce code d'entrée est responsable du chargement des dépendances, du traitement de la requêtes et de l'envoi de la réponse au navigateur.

- **Pourquoi centraliser toutes les requêtes dans un fichier index.php ?**

  Centraliser les requêtes via un fichier index.php est utile pour structurer proprement une application web.
  Cette approche permet de gérer des URLs personnalisées, indépendantes des fichiers physiques, ce qui améliore grandement le référencement. Elle respecte le principe DRY en évitant la duplication de code, car l'initialisation de la base de données et des sessions ne se fait qu'une seule fois au démarrage. La sécurité est renforcée puisque ce point d'entrée unique agit comme un filtre global pour vérifier les accès et protéger les fichiers sensibles du dossier public. Enfin, cette architecture facilite la maintenance et l'évolution future du projet en séparant clairement la logique de traitement de l'affichage.

- **Qu’est-ce qu’un routeur et à quoi sert-il ?**

  En développement web, le routeur est le composant logiciel essentiel qui fait le lien entre l'URL affichée dans le navigateur et le code précis à exécuter sur le serveur. Il agit comme un aiguilleur après le point d'entrée unique, il analyse chaque requête pour comprendre quelle page ou action est demandée par l'utilisateur. Il dirige la demande vers le bon contrôleur. C'est grâce au routeur que l'on peut séparer l'adresse web de l'organisation physique des fichiers, offrant ainsi des URLs propres et une architecture flexible.

- **Présentez une méthode simple de routage sans framework en PHP.**

  Une méthode basique consiste à récupérer l'URL demandée par l'utilisateur via $_SERVER['REQUEST_URI'] au sein du fichier index.php central. On peut ensuite utiliser une structure conditionnelle simple, comme un switch, pour comparer cette adresse à une liste de routes que nous aurions prédéfinies. Pour chaque correspondance trouvée, comme /accueil ou /contact, le script détermine quel fichier PHP spécifique doit être chargé pour traiter la demande. L'instruction require ou include est exécutée pour importer le code de la page visée, assurant ainsi l'affichage du bon contenu. Il faut prévoir un cas par défaut pour renvoyer une erreur 404 si l'URL ne correspond à aucune route connue.

Accès aux données avec PDO :

- **Qu'est-ce que PDO et pourquoi est-il recommandé pour accéder à une base de données SQL ?**

  PDO (PHP Data Object) est une extension qui défini l'interface pour accéder à une base de données avec PHP. PDO est orientée objet (POO). PDO facilite la migration vers un autre SGBD (Système de Gestion de Base de Données) puisqu'il n'est plus nécessaire de changer le code déjà développé.

- **Quelle est la différence entre query() et prepare() / execute() ?**

  Prepare() : Prépare une requête SQL à être exécutée par la méthode PDOStatement::execute(). Le modèle de déclaration peut contenir zéro ou plusieurs paramètres nommés (:nom) ou marqueurs (?) pour lesquels les valeurs réelles seront substituées lorsque la requête sera exécutée.

  - public **PDO::prepare**(string $query, array $options = []): PDOStatement|false

  Execute() : Exécute une requête préparée avec PDO ::prepare().

  - Public PDOStatement ::execute(?array $params = null):bool

Query() : prépare et exécute une requête SQL en un seul appel de fonction, retournant la requête en tant qu'objet PDOStatement.

- public **PDO::query**(string $query, ?int $fetchMode = **null**): PDOStatement|false

On peut donc en déduire que prepare() et execute() ont besoin l'une de l'autre pour fonctionner. Prepare() prépare la requête que execute() devra exécuter.

Query() prépare et exécute elle-même la requête.

- **Pourquoi les requêtes préparées sont-elles indispensables dans une application web ?**

  Les requêtes préparées permettent de protéger l'application web de potentielles injections SQL, et permettent de gagner en performance si une requête est répétée dans la même sessions. Mais l'aspect le plus important est la sécurisation de l'application contre les injections SQL.

- **Quelle est la différence entre fetch() et fetchAll() ?**

  Fetch() récupère une ligne depuis un jeu de résultats associé à l'objet PDOStatement tandis que fetchAll() les récupère toutes.

- **Comment gérer les erreurs lors de l'exécution d'une requête SQL avec PDO ?**

  On peut utiliser la fonction PDO::errorInfo() qui retourne les informations associées à l'erreur lors de la dernière opération **sur la base de données.**

  - public **PDO::errorInfo**(): array

  Mais aussi, PDOStatement::errorInfo() qui récupère les informations sur l'erreur associée lors de la dernière opération **sur la requête.**

  - public **PDOStatement::errorInfo**(): array

Gestion des requêtes et couche Model :

- **Quel est le rôle de la couche Model dans une architecture MVC ?**

  Comme dit précédemment, je cite :

  Modèle : Il représente la couche des données et la logique métier de l'application. Il gère l'accès aux données et les interactions avec la base de données. Il ne s'occupe pas de l'affichage des données. Il se focus sur le traitement de celle-ci.

- **Pourquoi le contrôleur ne doit-il pas contenir de requêtes SQL ?**

  Le contrôleur ne doit pas contenir de SQL pour respecter la séparation des responsabilités dans le modèle MVC, pour faciliter la maintenance et la réutilisation du code.

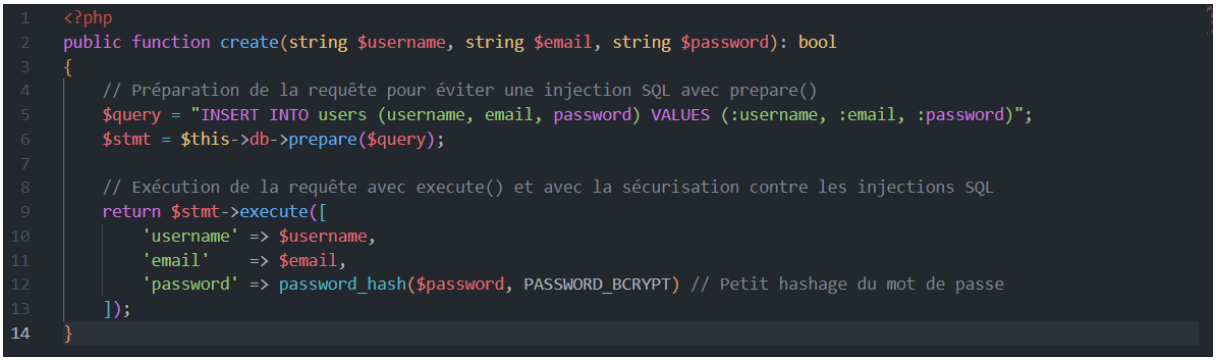
Il délègue la gestion des données au Modèle afin de rester un simple intermédiaire entre l'utilisateur et la base de données.

- **Comment organiser les méthodes d'un Model pour interagir avec la base de données ?**

  Il nous faut organiser les méthodes selon le CRUD (Create, Read, Update, Delete). Il nous fait répartir chaque fonction en accord avec son utilité dans le CRUD.

- **Donnez un exemple de méthode de Model permettant de lire ou d'écrire des données (pseudo-code accepté).**

  Code permettant d'écrire un nouvel utilisateur en utilisant prepare() et execute() vu précédemment :



```php
<?php
public function create(string $username, string $email, string $password): bool
{
    // Préparation de la requête pour éviter une injection SQL avec prepare()
    $query = "INSERT INTO users (username, email, password) VALUES (:username, :email, :password)";
    $stmt = $this->db->prepare($query);

    // Exécution de la requête avec execute() et avec la sécurisation contre les injections SQL
    return $stmt->execute([
        'username' => $username,
        'email'    => $email,
        'password' => password_hash($password, PASSWORD_BCRYPT) // Petit hashage du mot de passe
    ]);
}
```

Pattern Singleton :

- **Qu’est-ce que le pattern Singleton ?**

  C'est un design pattern qui garantit qu'une classe ne possède qu'une seule et unique instance accessible.

- **Pourquoi utilise-t-on souvent le Singleton pour la connexion à la base de données ?**

  On l’utilise pour éviter d'ouvrir des connexions multiples inutiles, ce qui permet d’économiser les ressources du serveur.

- **Citez un avantage et une limite du pattern Singleton.**

  Un avantage du pattern Singleton est qu’il a un point d’accès unique et un contrôle total sur l’instance.

  Une de ses limites est qu’il nous empêche d’ouvrir une deuxième connexion à une base de données différente (pour une migration par exemple).

Sources :

https://www.php.net/manual/fr **(Pour toutes les fonctions PHP)**

https://openclassrooms.com/fr/courses/4670706-adoptez-une-architecture-mvc-en-php

https://developer.mozilla.org/fr/docs/Glossary/CRUD

https://fr.wikipedia.org/wiki/PHP_Data_Objects

https://zestedesavoir.com/tutoriels/730/administrez-vos-bases-de-donnees-avec-mysql/952_securiser-et-automatiser-ses-actions/3954_requetes-preparees/

https://fr.wikipedia.org/wiki/Modèle-vue-contrôleur

https://www.reddit.com/r/PHP/comments/1an3emv/why_is_the_performance_benefit_of_prepared/?tl=fr

Glossaire :

Architecture MVC : Une méthode d'organisation du code qui sépare une application en trois parties distinctes (Modèle, Vue, Contrôleur) pour faciliter la maintenance et le travail en équipe.

CRUD : Acronyme pour Create, Read, Update, Delete. Ce sont les quatre opérations de base que l'on peut effectuer sur des données.

Dépendances : Fichiers, bibliothèques ou outils externes dont un programme a besoin pour fonctionner correctement.

Front Controller : Un fichier unique (souvent index.php) par lequel passent toutes les demandes des utilisateurs avant d'être redirigées vers la bonne section du site.

Injection SQL : Une cyberattaque où un utilisateur malveillant insère du code SQL pirate dans un formulaire pour voler ou détruire les données d'un site.

Logique métier : Ensemble des règles spécifiques à l'application (calculs, vérifications, gestion des droits) qui dictent comment les données doivent être traitées avant d'être affichées ou enregistrées.

Pattern Singleton : Un modèle de conception qui force une classe à ne créer qu'une seule et unique instance d'elle-même (très utilisé pour ne pas ouvrir plusieurs connexions à la base de données inutilement).

PDO (PHP Data Objects) : Une interface intégrée à PHP qui permet de communiquer avec une base de données de manière sécurisée et uniforme, quel que soit le type de base utilisé.

PHP : Un langage de programmation principalement utilisé pour créer des sites web dynamiques. Il s'exécute sur le serveur pour générer le code que l'utilisateur voit dans son navigateur.

Principe DRY : Acronyme de "Don't Repeat Yourself" (Ne vous répétez pas). C'est une règle de programmation qui consiste à éviter la duplication de code pour réduire les erreurs.

Requête HTTP : Message envoyé par un navigateur (le client) au serveur pour demander une action, comme afficher une page ou valider un formulaire.

Requête SQL : Commande textuelle envoyée à une base de données pour lire, ajouter, modifier ou supprimer des informations (ex: "Sélectionne tous les utilisateurs").

Requêtes préparées : Technique consistant à envoyer le modèle de la requête SQL séparément des données de l'utilisateur. C'est la méthode la plus efficace pour bloquer les injections SQL.

Routage : Le système qui analyse l'adresse tapée dans le navigateur (l'URL) pour décider quel contrôleur et quelle action doivent être déclenchés.

SGBD : Système de Gestion de Base de Données. C'est le logiciel qui stocke et gère réellement les données (ex: MySQL, PostgreSQL).

(Glossaire réalisé à l'aide d'une Intelligence Artificielle)