Programmer un site Internet en PHP

Avertissements

Bien que **Composer** soit un outil puissant pour les dépendances et les classes externes, certaines personnes préfèrent toujours créer leurs propres classes pour des raisons telles que :

- **Contrôle sur le code** : En créant ses propres classes, on peut contrôler le code source et le personnaliser pour répondre à ses besoins spécifiques.
- **Meilleure compréhension** : En créant ses propres classes, on peut mieux comprendre comment elles fonctionnent et les adapter à ses propres projets.
- **Prise en charge des anciens projets** : Si on a déjà utilisé ses propres classes pour la gestion de la base de données ou d'autres tâches dans d'autres projets, il peut être plus facile de les utiliser dans un nouveau projet plutôt que d'adopter un nouvel outil.
- **Exigences personnalisées** : Certaines personnes peuvent avoir des exigences spécifiques qui ne sont pas prises en charge par les bibliothèques externes existantes. La création de leurs propres classes leur permet de satisfaire ces besoins.

Le choix entre l'utilisation de **Composer** et la création de ses propres classes dépend des besoins et des préférences individuelles.

en cours de rédaction

Structure des dossiers d un projet php

Dans la plupart des cas, il est recommandé de créer un dossier **public** pour séparer les fichiers publics du code source. Le dossier **public** devrait contenir les fichiers accessibles directement via un navigateur web, tels que les fichiers **HTML**, **JavaScript**, **CSS** et **images**. Les autres fichiers, tels que les **classes PHP**, les **fichiers de configuration** et les **fichiers d'enregistrement**, devraient être placés dans un dossier séparé pour une meilleure sécurité.

Pour structurer un projet PHP avec des **classes**, **JavaScript** et **CSS**, voici une structure de dossiers suggérée :

```
web
|-- public
```

```
|   |-- css
|   |   |-- style.css
|   |-- js
|   |   |-- script.js
|   |-- images
|   |   |-- image1.jpg
|   |   |-- image2.png
|   |-- index.php
|-- classes
|   |-- class1.php
|   |-- class2.php
|-- config
|   |-- config.php
|-- tests
|   |-- Database
|       |-- DatabaseConnectorTest.php
|-- logs
|   |-- application.log
|-- vendor
|   |-- ..
|-- composer.json
|-- .env
```

Dans un projet PHP, les dossiers **classes**, **css** et **js** peuvent être utilisés pour organiser les fichiers associés à ces technologies.

- `classes/` : ce dossier peut contenir toutes les classes PHP utilisées pour la logique de l'application.
- `css/` : ce dossier peut contenir tous les fichiers CSS pour la mise en forme de l'interface utilisateur.
- `js/` : ce dossier peut contenir tous les fichiers JavaScript pour la logique client-side et l'interaction utilisateur.
- `images/` : ce dossier devrait être placé dans le dossier public pour être accessible directement via un navigateur web. Il contient les images.

Il est courant de placer les fichiers de configuration dans un dossier nommé `config` et les fichiers d'enregistrement dans un dossier nommé `logs`.

Cependant, la structure de dossiers dépend fortement des besoins spécifiques de chaque projet, il n'y a donc pas de solution universelle. Il est important de choisir une structure de dossiers qui facilite la maintenance et la compréhension de votre projet pour vous et les autres développeurs qui peuvent travailler dessus.

Créer un autoloader pour les classes

Un **autoloader** est une fonction en PHP qui charge automatiquement les classes nécessaires à l'exécution du code. Vous pouvez créer un **autoloader** en définissant une fonction qui inclura le fichier associé à une classe spécifique lorsque cette classe est utilisée pour la première fois.

Voici un exemple d'implémentation d'un **web/autoload** :

autoload.php

```php
<?php
spl_autoload_register(function ($className) {
    $classFile = 'classes/' . str_replace('\\', '/', $className) .
'.php';
    require_once $classFile;
});
```

Cet exemple utilise la fonction `spl_autoload_register` de PHP pour définir l'**autoloader**. La fonction `spl_autoload_register` accepte une fonction anonyme qui sera appelée chaque fois qu'une classe non trouvée sera utilisée.

La fonction anonyme convertit le nom de la classe en un nom de fichier en remplaçant les antislashs (\) par des slashs (/) et ajoute l'extension `.php` pour former le nom du fichier associé à la classe. Enfin, le fichier associé est inclu en utilisant `require_once` précédé du chemin pour accéder au classes (`/classes`).

Ce code suppose que les noms de classes correspondent aux noms de fichiers et que les classes sont rangées dans des dossiers hiérarchiques correspondant à leur **namespace**. Il peut être nécessaire de personnaliser cette implémentation en fonction de la structure de dossiers de votre projet.

index.php

```php
<?php

// Chargement de l'autoloader
require_once '../autoload.php';

// Utilisation des classes
$object = new MyClass();
```

Créer un fichier de configuration

Créer des fichiers de log

Créer une connexion à la base de données

— *Cédric ABONNEL (cedricabonnel)*, *CPT*