S1 : Introduction aux Frameworks (1)

Introduction

A partir de l'architecture MVC que vous avez construite, on a vu comment cela peut organiser le code de manière organisé.

Maintenant, à partir de ce que vous avez fait, est-ce que vous pouvez me dire ce que tu vous avez construit de votre côté pour gérer ton application ?

- Le routing pour diriger les requêtes vers les bons contrôleurs et actions
- Les vues pour afficher du HTML.

Au-delà du **routing**, qu'est-ce qu'on a dû faire d'autre pour que tout fonctionne correctement ?

- Configurer le serveur pour qu'il redirige correctement les requêtes HTTP vers le fichier index.php.
- Configurer la base de données manuellement avec PhpMyAdmin, en créant des tables et en structurant les relations entre elles. Chaque fois qu'il y a un petit changement, on doit refaire tout le travail manuellement.

C'est là qu'on peut voir la différence avec un framework comme **Laravel**. Quand tu on crée un **MVC maison**, même si cela permet de comprendre comment les composants fonctionnent, **on doit se charger de beaucoup de choses à la main**.

- Comme la configuration du serveur
- La gestion de la base de données
- Le routage.

⇒ Cela peut devenir difficile à maintenir, et on risque de perdre du temps à

gérer ces aspects répétitifs.

Un framework permet de gagner beaucoup de temps en vous offrant des **outils préfabriqués pour des tâches courantes.**

⇒ Au lieu de devoir configurer manuellement chaque aspect de ton projet, un frameworks PHP comme **Laravel** gère:

- le routage,
- la **configuration de la base de données** se fait via un outil qui permet de **créer** et de **modifier** des tables dans ta base de données **directement via du code PHP**(plus besoin de PhpMyAdmin), et il n'y a plus à configurer le serveur pour chaque petit changement.

Ce qui permet de se concentrer sur les fonctionnalités du projet, plutôt que de s'occuper de la configuration.

---

1. Qu'est-ce qu'un Framework ? (5 min)

Un framework est un **ensemble d'outils, de bibliothèques et de conventions qui simplifient et organisent le développement d'applications**.

C'est un cadre de travail qui fournit
une structure de code préétablie conçu pour aider à développer des logiciels plus rapidement et plus efficacement.

Les frameworks sont **utilisés dans différents langages de programmation** (JavaScript, Python, Ruby, PHP, etc.)

Un framework PHP est une collection de classes et de bibliothèques qui vous aide à développer des applications web.

2. Caractéristiques d'un Framework :

- **Structure organisationnelle** : Il impose une structure qui permet d’avoir une organisation claire du code.
- **Modularité** : Il permet de réutiliser du code à travers des modules et des bibliothèques.
- **Réutilisation du code** : Il fournit des fonctions prêtes à l'emploi pour éviter de réécrire des fonctionnalités de base (authentification, gestion des sessions, validation, etc.).
- **Gestion des erreurs** : Des outils pour gérer les erreurs et exceptions.

2. Pourquoi Utiliser un Framework PHP ? (7 min)

Les frameworks PHP comme **Laravel, Symfony, ou CodeIgniter** sont des outils puissants qui rendent le développement d'applications web plus rapide, sécurisé et maintenable.

- Open source
- Conçu pour rendre le processus de développement web **plus rapide, plus propre et plus maintenable.**

1. Gain de Temps

Un framework vous permet de gagner beaucoup de temps en vous offrant des **outils préfabriqués** pour des tâches courantes.

- **Exemple** :
  - Système de routage très simple à configurer,
  - Fonctionnalités de validation des formulaires,
  - Gestion des sessions
  - Gestion de base de données avec l'ORM Eloquent, etc.

⇒ Facile et rapide à installer avec une **commande Composer**.

2. Réutilisation du Code

Avec un framework, vous utilisez du code **déjà testé et validé**. Cela signifie moins d'erreurs et un code plus robuste. Vous pouvez réutiliser des composants comme les routes, les contrôleurs, et les vues **sans avoir à les re-développer à chaque fois.**

- **Exemple** : Dans **Laravel**, vous pouvez créer des modèles Eloquent qui interagissent automatiquement avec la base de données et évitent les requêtes SQL complexes.

3. Sécurité

Les frameworks PHP, en particulier Laravel, intègrent des **mesures de sécurité** par défaut. Ils vous protègent contre des attaques courantes telles que les **attaques par injection SQL**, les **attaques XSS** (Cross-Site Scripting), ou encore les attaques **CSRF** (Cross-Site Request Forgery).

- **Exemple** : Laravel protège automatiquement les applications contre les attaques CSRF en générant des tokens uniques pour chaque formulaire.

4. Maintenance Simplifiée

Les frameworks sont conçus pour promouvoir une **architecture propre** et **structurée** (souvent avec une architecture MVC - Modèle-Vue-Contrôleur). Cela permet de maintenir plus facilement un projet à long terme.

- **Exemple** : **Laravel** suit l'architecture MVC, ce qui signifie que le code de la logique métier, de l'affichage et des données est séparé, ce qui facilite la maintenance et les évolutions du projet.

5. Communauté et Documentation

La plupart des frameworks PHP populaires ont une **grande communauté** et une **documentation complète**. On trouve facilement **des ressources**, **des tutoriels** et **obtenir de l'aide sur des forums ou Stack Overflow**.

- **Exemple** : **Laravel** a une très grande communauté active et une documentation claire qui aide les développeurs à résoudre rapidement des problèmes.

Conclusion (30 secondes)

En résumé, un **framework PHP** est un outil qui vous aide à développer plus rapidement, en vous offrant des outils et des structures pré-configurées pour gérer des tâches courantes comme l'interaction avec la base de données, la gestion des utilisateurs, la sécurité, et plus encore.

Utiliser un framework permet **de se concentrer sur les fonctionnalités spécifiques** de votre projet tout en bénéficiant de la puissance d'un cadre éprouvé.

S1: Exercices

S1: Structure du projet Laravel