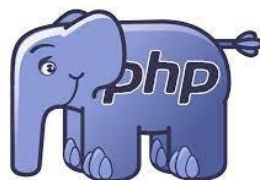


# Cours PHP Laravel

2023-2024 (laravel 10)

**L4 UBO PDW**



# Programme laravel



## **I- Introduction à Laravel**

Présentation de Laravel et son écosystème.

Comparaison avec d'autres frameworks.

## **II- Installation et Configuration**

Guide d'installation de Laravel via Composer.

Configuration initiale et compréhension de l'architecture.

## **III- Console Artisan**

Pourquoi utiliser artisan pour simplifier les tâches courantes ?

## **IV- Concepts de base de Laravel**

Routing (Comprendre le système de routage de Laravel).

Création de contrôleurs et gestion des actions, ou des requêtes http.

Blade Templating Engine (Moteur de template blade).

Bases de données et ORM Eloquent utiliser dans laravel.

## **V- Gestion des Formulaires et Validation**

Formulaire HTML et CSRF Protection

Validation des Données

# Introduction


## **Présentation de Laravel et son écosystème.**

Conçu par Taylor Otwell en tant que framework web PHP open source gratuit, Laravel vise à simplifier et accélérer le développement d'applications web, mettant particulièrement l'accent sur la simplicité. Il suit le modèle architectural Modèle-Vue-Contrôleur (MVC), ainsi que les normes de codage PSR-2 et de chargement automatique PSR-4.

Le développement piloté par les tests (TDD) avec Laravel est à la fois plaisant et simple à mettre en œuvre. Hébergé sur GitHub et accessible à l'adresse <https://github.com/laravel/laravel>, Laravel se distingue par une architecture de micro-services, ce qui le rend extrêmement extensible. Cette extensibilité est facilement réalisable grâce à l'utilisation de paquets tiers personnalisés ou existants.



## **Comparaison avec d'autres frameworks**

Laravel est l'un des nombreux frameworks web populaires, et sa comparaison avec d'autres dépend souvent des besoins spécifiques d'un projet. Pour comprendre mieux, faisons une comparaison de laravel  face autres framework comme :

- Django (un framework Python)
- et Ruby on Rails (un framework Ruby) :

### 1. Langage de Programmation :

- Laravel est basé sur PHP.
- Django est basé sur Python.
- Ruby on Rails est basé sur Ruby.

### 2. Architecture :

- Laravel suit le modèle architectural Modèle-Vue-Contrôleur (MVC).
- Django suit également le modèle MVC, appelé Modèle-Vue-Template (MVT) dans Django.
- Ruby on Rails suit lui aussi le concept de MVC.

### 3. Base de Données :

- Laravel utilise Eloquent ORM pour la gestion de la base de données.
- Django utilise son propre ORM, appelé Django ORM.
- Ruby on Rails utilise ActiveRecord pour la gestion de la base de données.

#### 4. Migrations de Base de Données :

- Laravel propose des migrations pour faciliter la gestion des schémas de base de données.
- Django a des migrations intégrées pour gérer les évolutions de base de données.
- Ruby on Rails propose également des migrations.

#### 5. Facilité d'Apprentissage :

- Laravel est souvent considéré comme convivial pour les développeurs débutants, donc facile à apprendre.
- Django offre une courbe d'apprentissage modérée, mais sa documentation est complète.
- Ruby on Rails est réputé pour sa simplicité, mais certains aspects peuvent être moins évidents pour les débutants.

#### 6. Écosystème et Communauté :

- Laravel a une communauté active et propose un écosystème de packages via Composer.
- Django a une communauté solide et propose le Python Package Index (PyPI) pour les packages.
- Ruby on Rails bénéficie d'une communauté dynamique et d'une gestion des dépendances via RubyGems.

En fin de compte, le choix entre Laravel, Django et Ruby on Rails ou tout autre framework web dépend des préférences personnelles, des besoins spécifiques du projet et d'autres facteurs contextuels. Chacun de ces frameworks a ses forces et ses faiblesses, et il est important de les évaluer en fonction des exigences de votre application.

## **Concept de laravel**

L'architecture de Laravel suit le modèle MVC (Modèle-Vue-Contrôleur), qui est un design pattern couramment utilisé dans le développement d'applications web. Dans Laravel, on retrouve :

### **1.Modèle (Model) :**

- Le modèle représente la couche d'accès aux données de l'application. Il est responsable de récupérer et de stocker les données dans la base de données.

- Dans Laravel, Eloquent est le système ORM (Object-Relational Mapping) utilisé pour interagir avec la base de données.

### **2. Vue (View) :**

- La vue est responsable de l'affichage des données et de l'interface utilisateur. Elle reçoit les données du contrôleur et les présente à l'utilisateur.

- Laravel utilise le moteur de template Blade pour la gestion des vues. Blade offre une syntaxe simple, et expressive et permet d'inclure des directives pour le contrôle de flux et la gestion de la logique dans les vues en laravel.

### **3. Contrôleur (Controller) :**

- Le contrôleur gère la logique de l'application et agit comme un intermédiaire entre le modèle et la vue. Il reçoit les requêtes de l'utilisateur, interagit avec le modèle pour récupérer les données nécessaires, puis les transmet à la vue pour l'affichage.

- Les contrôleurs dans Laravel sont généralement responsables de la gestion des actions (requête HTTP) liées aux routes définies dans l'application.

#### 4. Routes :

- Les routes définissent les points d'entrée de l'application et spécifient quel contrôleur et quelle action (méthode) doivent être appelés en fonction de la requête de l'utilisateur.
- Laravel propose un fichier de configuration de routes où vous pouvez définir vos itinéraires de manière claire et lisible.

#### **5. Middleware :**

- Le middleware est une couche intermédiaire qui peut effectuer des actions avant ou après le traitement d'une requête. Il est utilisé pour filtrer les requêtes HTTP, gérer l'authentification, la vérification des autorisations, etc.
- Laravel propose des middleware pré-définis et permet aux développeurs de créer leur propre middleware personnalisé.

#### **6. Services et Providers :**

- Les services et les providers dans Laravel sont utilisés pour organiser et lier des composants de l'application. Les services peuvent être des classes utilisées pour effectuer des tâches spécifiques, tandis que les providers sont responsables de la configuration des services et de leur enregistrement dans l'application.

#### **7. Base de données et Migrations :**

- Laravel utilise Eloquent ORM pour la gestion de la base de données. Les migrations facilitent la création et la modification de la structure de la base de données de manière versionnée et cohérente.

L'architecture de Laravel favorise la séparation des préoccupations, ce qui rend l'application plus organisée, facile à maintenir et extensible. La modularité et la clarté du code sont des aspects importants de l'architecture Laravel, contribuant à une expérience de développement efficace.

## **Architecture de dossier et de fichier dans un projet laravel**

L'architecture de fichiers dans Laravel est conçue pour offrir une structure organisée et cohérente, facilitant le développement, la maintenance et le déploiement d'applications.

### **- Le répertoire "app" :**

Contient le code de l'application, y compris les modèles, les contrôleurs, les politiques, etc.

Les modèles sont généralement stockés dans le sous-répertoire "app/Models".

Les contrôleurs sont stockés dans le sous-répertoire "app/Http/Controllers".

### **- Le répertoire "bootstrap" :**

Contient des fichiers permettant de démarrer l'application, y compris le chargement automatique des classes et la configuration de l'application.

### **- Le répertoire "config" :**

Contient les fichiers de configuration de l'application, tels que les paramètres de base de données, les fichiers de service, etc.

### **- Le répertoire "database" :**

Contient les migrations de base de données et les seeders.

Les migrations sont utilisées pour versionner la structure de la base de données.

Les seeders permettent de remplir la base de données avec des données de test.



### **- Le répertoire "public" :**

Contient le point d'entrée de l'application, le fichier "index.php".

Les fichiers publics tels que les images, les feuilles de style et les scripts JavaScript peuvent être stockés ici.

### **Le répertoire "resources" :**

Contient les fichiers de ressources, tels que les vues, les fichiers de langues, les fichiers de configuration front-end, etc.

Les vues Blade sont souvent stockées dans "resources/views".

### **- Le répertoire "routes" :**

Contient les fichiers de définition des routes de l'application.

Le fichier "web.php" définit les routes pour les interactions web, tandis que "api.php" peut être utilisé pour les routes d'API.

### **- Le répertoire "storage" :**

Stocke les fichiers générés par l'application, tels que les fichiers journaux, les fichiers de cache, etc.

Le sous-répertoire "storage/app" est souvent utilisé pour stocker des fichiers générés par l'application.

### **- Le répertoire "tests" :**

Contient les tests unitaires et fonctionnels de l'application.

### **- Le fichier ".env" :**

Contient la configuration de l'environnement, y compris les informations de base de données, les clés d'application, etc.

Ce fichier n'est pas versionné et est spécifique à chaque environnement.

**- Le fichier "composer.json" :**

Définit les dépendances de l'application et les scripts Composer pour automatiser certaines tâches.