

LARAVEL 11.X



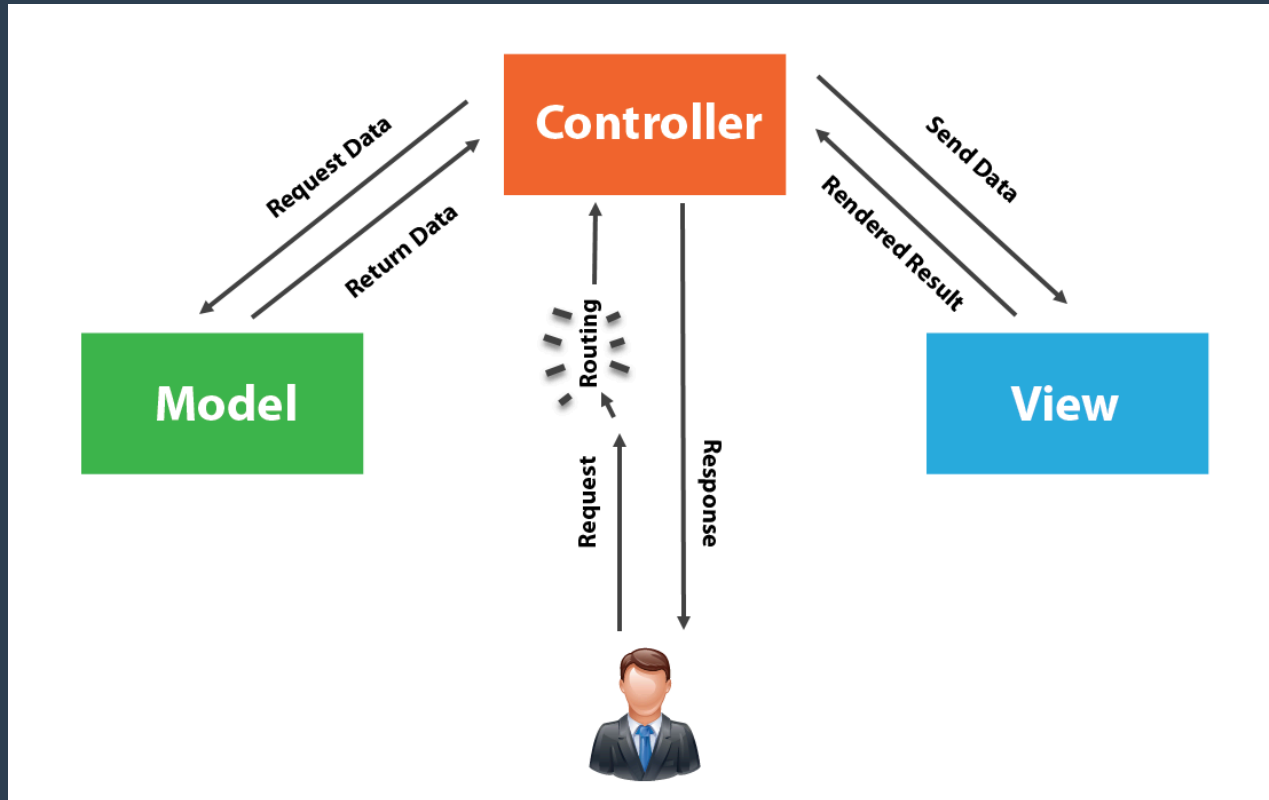
Steve Lebleu - 2CE-X75-A - 2024-2025

MODULE 1 - INTRODUCTION

QU'EST-CE QU'UN FRAMEWORK MVC ?

Séparation des préoccupations:

- Modèle (données)
- Vue (présentation)
- Contrôleur (logique)



POURQUOI LARAVEL ?

- Ecosystème très riche
- Nombreuses fonctionnalités
- Rapidité de développement
- Sécurité
- Communauté active



MODULE 2 - INSTALLATION

INSTALLATION AVEC DOCKER

```
1 # Build les images
2 docker compose build
3
4 # Lancer les containers
5 docker compose up -d
6
7 # Attacher un shell
8 docker compose exec app sh
```

```
1 # Build les images
2 docker compose build
3
4 # Lancer les containers
5 docker compose up -d
6
7 # Attacher un shell
8 docker compose exec app sh
```

```
1 # Build les images
2 docker compose build
3
4 # Lancer les containers
5 docker compose up -d
6
7 # Attacher un shell
8 docker compose exec app sh
```

```
1 # Build les images
2 docker compose build
3
4 # Lancer les containers
5 docker compose up -d
6
7 # Attacher un shell
8 docker compose exec app sh
```

```
1 # Dans le container, supprimer les fichiers locaux
2 rm -rf ./*
3
4 # Installer Laravel via Composer
5 composer create-project --prefer-dist laravel/laravel .
6
7 # Générer le fichier autoload de Composer
8 composer install
9
10 # Générer une clé d'application
11 php artisan key:generate
12
13 # Installer les dépendances Node.js
14 npm i
15
```

```
1 # Dans le container, supprimer les fichiers locaux
2 rm -rf ./*
3
4 # Installer Laravel via Composer
5 composer create-project --prefer-dist laravel/laravel .
6
7 # Générer le fichier autoload de Composer
8 composer install
9
10 # Générer une clé d'application
11 php artisan key:generate
12
13 # Installer les dépendances Node.js
14 npm i
15
```



```
1 # Dans le container, supprimer les fichiers locaux
2 rm -rf ./*
3
4 # Installer Laravel via Composer
5 composer create-project --prefer-dist laravel/laravel .
6
7 # Générer le fichier autoload de Composer
8 composer install
9
10 # Générer une clé d'application
11 php artisan key:generate
12
13 # Installer les dépendances Node.js
14 npm i
15
```

```
3
4 # Installer Laravel via Composer
5 composer create-project --prefer-dist laravel/laravel .
6
7 # Générer le fichier autoload de Composer
8 composer install
9
10 # Générer une clé d'application
11 php artisan key:generate
12
13 # Installer les dépendances Node.js
14 npm i
15
16 # Sortir du container
17 exit
```

```
3
4 # Installer Laravel via Composer
5 composer create-project --prefer-dist laravel/laravel .
6
7 # Générer le fichier autoload de Composer
8 composer install
9
10 # Générer une clé d'application
11 php artisan key:generate
12
13 # Installer les dépendances Node.js
14 npm i
15
16 # Sortir du container
17 exit
```

```
3
4 # Installer Laravel via Composer
5 composer create-project --prefer-dist laravel/laravel .
6
7 # Générer le fichier autoload de Composer
8 composer install
9
10 # Générer une clé d'application
11 php artisan key:generate
12
13 # Installer les dépendances Node.js
14 npm i
15
16 # Sortir du container
17 exit
```

```
1 # Copier la configuration vite.js optimisée
2 mv _vite.config.js ./src/vite.config.js
3
4 # Stopper les containers
5 docker compose down
6
7 # Relancer les containers
8 docker compose up -d
```

```
1 # Copier la configuration vite.js optimisée
2 mv _vite.config.js ./src/vite.config.js
3
4 # Stopper les containers
5 docker compose down
6
7 # Relancer les containers
8 docker compose up -d
```

```
1 # Copier la configuration vite.js optimisée
2 mv _vite.config.js ./src/vite.config.js
3
4 # Stopper les containers
5 docker compose down
6
7 # Relancer les containers
8 docker compose up -d
```

FACULTATIF

```
1 # Attache un shell au container
2 docker compose exec app sh
3
4 # Set permissions sur le répertoire de stockage de Laravel
5 chown -R www-data:www-data /var/www/html/storage
6
7 # Set permissions sur le répertoire des caches
8 chown -R www-data:www-data /var/www/html/bootstrap/cache
```


FACULTATIF

```
1 # Attache un shell au container
2 docker compose exec app sh
3
4 # Set permissions sur le répertoire de stockage de Laravel
5 chown -R www-data:www-data /var/www/html/storage
6
7 # Set permissions sur le répertoire des caches
8 chown -R www-data:www-data /var/www/html/bootstrap/cache
```

ACCÈS À L'APPLICATION

<http://localhost:8000>

WARNING

Docker is extremely slow when running laravel under
WSL2 on Windows

MODULE 3 - CONFIGURATION

ARTISAN CLI

Laravel embarque le CLI **Artisan**, qui permet d'interagir avec toutes les couches de l'application.

Connaître les commandes disponibles avec *artisan*:

```
1 docker compose exec app php artisan list
2 # ou Docker Desktop > Containers > [CONTAINER_NAME] > Exec
3 # ou extension VSCode > [CONTAINER_NAME] > Attach Shell
4 # ou docker compose exec app sh
5 php artisan list
```

Connaître le détail d'une configuration:

```
1 docker compose exec app php artisan config:show [CONFIGURATION]
2 # ou Docker Desktop > Containers > [CONTAINER_NAME] > Exec
3 # ou extension VSCode > [CONTAINER_NAME] > Attach Shell
4 # ou docker compose exec app sh
5 php artisan config:show [CONFIGURATION]
```


Outil puissant et efficace qui vous accompagnera tout
au long de vos développements Laravel !

LE FICHER .ENV

- Fichier d'environnement classique généré par Laravel à l'installation.
- Contient les valeurs par défaut.

Les informations de connexion à la base de données
doivent être mises à jour.

```
1 DB_CONNECTION=mysql
2 DB_HOST=mysql
3 DB_PORT=3306
4 DB_DATABASE=laravel
5 DB_USERNAME=root
6 DB_PASSWORD=root
7 ...
8 VITE_PORT=5173
9 VITE_HOST=0.0.0.0
```

Il est également nécessaire de modifier les valeurs par défaut dans le fichier *config/database*.

Créer les tables utilisateurs avec *artisan*:

```
1 docker compose exec app php artisan migrate
2 # ou Docker Desktop > Containers > [CONTAINER_NAME] > Exec
3 # ou extension VSCode > [CONTAINER_NAME] > Attach Shell
4 # ou docker compose exec app sh
5 php artisan migrate
```

STRUCTURE D'UN PROJET LARAVEL

- Contrôleurs et modèles dans *app*
- Migrations dans *database/migrations*
- Contrôleur frontal dans *public/index.php*
- Configurations dans *config*
- Vues et assets dans *resources*
- Routes dans *routes*
- Système de fichiers local dans *storage*
- Tests unitaires dans *tests*
- Dépendances PHP dans *vendors*

MODULE 4 - MODEL

ELOQUENT ORM

La couche **ORM** embarquée nativement par le framework Laravel.

Bénéficie de son intégration dans l'écosystème Laravel pour apporter simplicité et performance à la gestion des données.

LES MIGRATIONS

Opérations qui permettent de créer, modifier ou supprimer une structure (un schéma / une table) ou une donnée associée à l'utilisation d'un ORM.

Création d'une migration

```
1 php artisan make:migration [MIGRATION_NAME]
```


Génère une migration dans *database/migrations*.

```
1 use Illuminate\Database\Migrations\Migration;
2 use Illuminate\Database\Schema\Blueprint;
3 use Illuminate\Support\Facades\Schema;
4
5 return new class extends Migration
6 {
7     /**
8      * Run the migrations.
9      */
10    public function up(): void
11    {
12        Schema::create('my_schema', function (Blueprint $table) {
13            $table->id();
14            ...
15        });
16    }
17 }
```

```
3 use Illuminate\Support\Facades\Schema;
4
5 return new class extends Migration
6 {
7     /**
8      * Run the migrations.
9      */
10    public function up(): void
11    {
12        Schema::create('my_schema', function (Blueprint $table) {
13            $table->id();
14            ...
15        });
16    }
17
```

```
11     {
12         Schema::create('my_schema', function (Blueprint $table) {
13             $table->id();
14             ...
15         });
16     }
17
18     /**
19      * Reverse the migrations.
20      */
21     public function down(): void
22     {
23         Schema::dropIfExists('my_schema');
24     }
25 };
```

- Classe anonyme qui hérite de la classe *Migration*
- 2 méthodes publiques: *up* and *down*

Exécution des migrations:

```
1 php artisan migrate
```

LES MODÈLES

Classes qui représentent une table de la base de données.

Création d'un modèle

```
1 php artisan make:model [MODEL_NAME]
```

Génère une classe qui hérite de la classe *Model* dans *app/Models*.

```
1 namespace App\Models;
2
3 use Illuminate\Database\Eloquent\Factories\HasFactory;
4 use Illuminate\Database\Eloquent\Model;
5
6 class MyModel extends Model
7 {
8     use HasFactory;
9     protected $fillable = ['name', ...];
10
11     ...
12 }
```

MODULE 5 - CONTROLLER

Création d'un contrôleur avec artisan:

```
1 php artisan make:controller [CONTROLLER_NAME] [OPTIONS]
```

Génère un fichier dans *app/Http/Controllers*.

```
1 namespace App\Http\Controllers;
2
3 use Illuminate\Http\Request;
4
5 class MyController extends Controller
6 {
7     public function index()
8     {
9
10    }
11    ...
12 }
```

LE SYSTÈME DE ROUTAGE

Différents types de routes:

- GET
- POST
- PUT / PATCH
- DELETE

Définition des routes dans *routes/web.php*

```
1 Route::get('/', [MyController::class, 'index']);
```

MODULE 6 - VIEW

BLADE TEMPLATE ENGINE

Laravel embarque **Blade**, un moteur de templates directement intégré dans l'écosystème.

Comme tout l'écosystème, c'est moderne, puissant et ça apporte une vraie plus-value par rapport aux autres outils sur le marché.

ELÉMENTS DE SYNTAXE DE BASE

Affichage avec échappement

```
1 {{ $variable }}
```


Affichage sans échappement

```
1 {!! $variable !!}
```

Conditions

```
1 @if ($condition) ... @else ... @endif
```

Boucles

```
1 @foreach ($items as $item) ... @endforeach
```

Héritage

```
1 @extends('layouts.app')
```

CRÉATION D'UN LAYOUT

resources/views/layouts/app.blade.php

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Mon layout</title>
5   </head>
6   <body>
7     <div class="container">
8       @yield('content')
9     </div>
10  </body>
11 </html>
```

AFFICHAGE D'UN BLOCK

resources/views/games/index.blade.php

```
1 @extends('layouts.app')
2
3 @section('content')
4     <div>...</div>
5 @endsection
```


resources/views/games/index.blade.php

```
1 @extends('layouts.app')
2
3 @section('content')
4     <div>...</div>
5 @endsection
```

resources/views/games/index.blade.php

```
1 @extends('layouts.app')
2
3 @section('content')
4     <div>...</div>
5 @endsection
```