LARAVEL 11.X

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

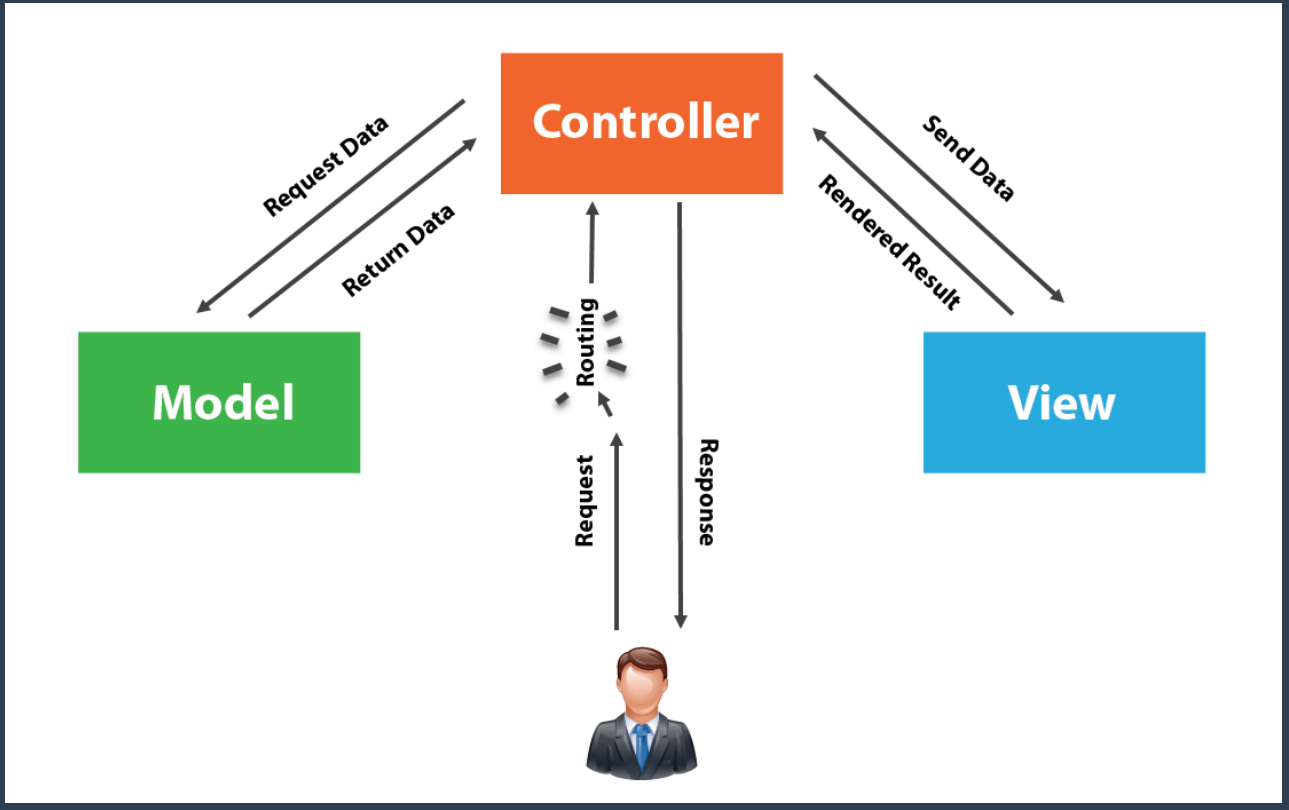
Konfer.

MODULE 1 - INTRODUCTION

QU'EST-CE QU'UN FRAMEWORK MVC ?

Séparation des préocupations:

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


Controller
Request Data
Return Data
Send Data
Rendered Result
Routing
Model
View
Request
Response

POURQUOI LARAVEL ?

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

Laravel

MODULE 2 - INSTALLATION

INSTALLATION AVEC DOCKER

Dockerfile

```
1  # Image de base légère avec PHP 8.2 et FPM
2  FROM php:8.2-fpm-alpine
3
4  # Installation des dépendances système avec mise en cache optimi
5  RUN apk add --no-cache \
6      libzip-dev \
7      zip \
8      nodejs \
9      npm \
10     git \
11     curl \
12     vim \
13     linux-headers \
14     $PHPIZE_DEPS
15
```

docker-compose.yaml

```yaml
services:
# Service PHP-FPM
app:
  build: .
  volumes:
    # Mount avec le flag :cached pour de meilleures performance
    - ./src:/var/www/html:cached
    # Volumes nommés pour les dépendances (évite les problèmes
    - vendor:/var/www/html/vendor
    - node_modules:/var/www/html/node_modules
  environment:
    - DOCKER_BUILDKIT=1
    # Variables d'environnement pour le développement
    - APP_ENV=local
    - PHP_IDE_CONFIG=serverName=docker
```

```
# Build les images
docker compose build

# Lancer les conteneurs
docker compose up -d

# Attacher un shell
docker compose exec app
```

```
# Build les images
docker compose build

# Lancer les conteneurs
docker compose up -d

# Attacher un shell
docker compose exec app
```

```
# Build les images
docker compose build

# Lancer les conteneurs
docker compose up -d

# Attacher un shell
docker compose exec app
```

```
# Dans le container, installer Laravel
composer create-project --prefer-dist laravel/laravel .

# Dans le container, générer une clé d'application
php artisan key:generate
```

```
# Dans le container, installer Laravel
composer create-project --prefer-dist laravel/laravel .

# Dans le container, générer une clé d'application
php artisan key:generate
```

FACULTATIF

```
1 # Set permissions
2 docker compose exec app chown -R www-data:www-data /var/www/html/s
3 docker compose exec app chown -R www-data:www-data /var/www/html/l
```

FACULTATIF

```
1 # Set permissions
2 docker compose exec app chown -R www-data:www-data /var/www/html/s
3 docker compose exec app chown -R www-data:www-data /var/www/html/l
```

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 php artisan list
4 # ou extension VSCode > [CONTAINER_NAME] > Attach Shell
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 php artisan config:show [CONFIGURATION]
4 # ou extension VSCode > [CONTAINER_NAME] > Attach Shell
5 php artisan config:show [CONFIGURATION]
```

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

LE FICHIER .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.

```
DB_CONNECTION=mysql
DB_HOST=mysql
DB_PORT=3306
DB_DATABASE=laravel
DB_USERNAME=root
DB_PASSWORD=root
```

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

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*.

```php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('my_schema', function (Blueprint $table) {
            $table->id();
            ...
        });
```

```php
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('my_schema', function (Blueprint $table) {
            $table->id();
            ...
        });
    }

```

```
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*.

```php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class MyModel extends Model
{
  use HasFactory;
  protected $fillable = ['name', ...];

  ...
}
```

MODULE 5 - CONTROLER

Création d'un contrôleur avec artisan:

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

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

```php
namespace App\Http\Controllers;

use Illuminate\Http\Request;

class MyController extends Controller
{
  public function index()
  {

  }
  ...
}
```

LE SYSTÈME DE ROUTAGE

Différents types de routes:

- GET
- POST
- PUT / PATCH
- DELETE

Définition des routes dans *routes/web.php*

```php
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
```