Formation Laravel

IMIE Angers – Mai 2019

Déroulé

- Présentation et mécanique de base de Laravel (MVC, injection de dépendances …)
- Installation / Configuration de Laravel
- Routage d’une requête / Contrôleurs
- Vues avec Blade
- Connexion à la base de données / ORM Eloquent
- Gestion de formulaires
- Authentification
- Laravel avancé

Pré-requis

- Maitriser la programmation orientée objet en PHP
- Connaitre le patron de conception MVC est un plus
- Environnement WAMP / MAMP / LAMP fonctionnel
- Utilisation de PHPStorm recommandée

Chapitre 1 : Présentation

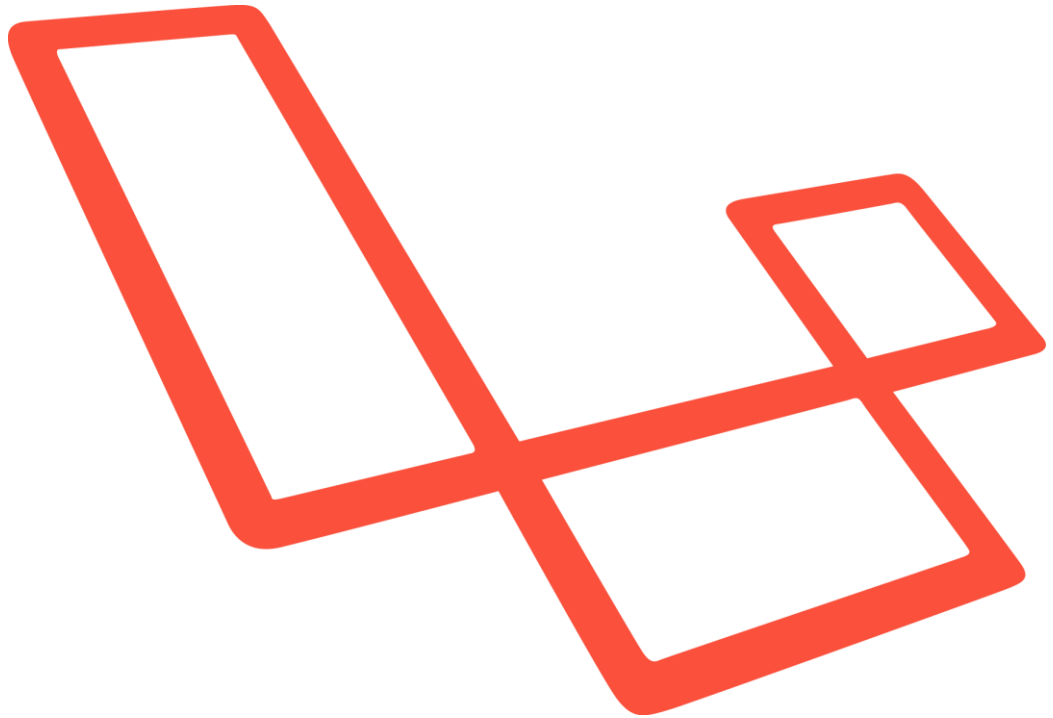
Présentation succincte de Laravel, fonctionnement de base

Objectifs

- Qu’est-ce qu’un framework ? Intérêts ?
- Les alternatives à Laravel
- Pourquoi Laravel ?
- Rappels sur le patron de conception MVC

Laravel

- Framework PHP open-source
- Créé en 2011 par Taylor Otwell
- Principe MVC
- Dernière version majeure : 5.8
- S’appuye sur plusieurs composant Symfony



Pourquoi un framework ?

- Offre une structure de code permettant un travail en équipe
- Bibliothèque de composants
- Évite de réinventer la roue
- Productivité accrue
- Produit du code de qualité

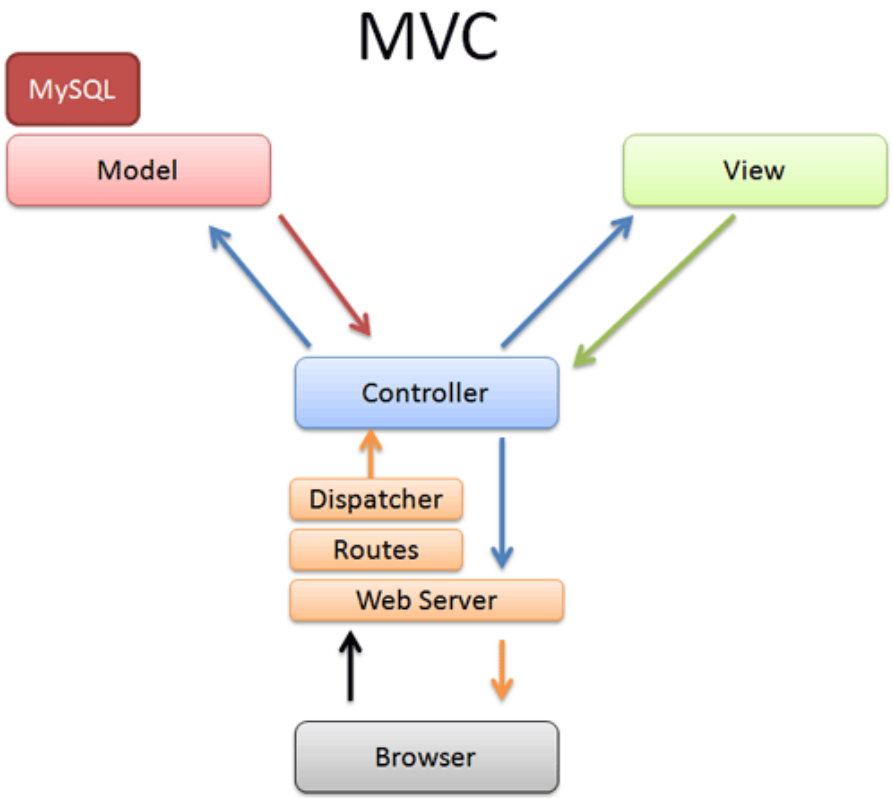
Alternatives à Laravel

- Frameworks
  - Symfony
  - Cake PHP
  - Phalcon PHP
  - Code Igniter
  - …
- CMS
  - WordPress
  - Joomla
  - Drupal
  - Prestashop
  - Magento

Pourquoi utiliser Laravel ?

- Framework N°1 sur Github (+50K stars)
- Très grosse communauté de développeurs
- Nombreux packages connexes permettant d’ajouter des fonctionnalités
- Plus facile à appréhender que Symfony

Patron de conception MVC



Patron de conception MVC

Sans MVC : une ressource par requête

http://www.monsite.com/	/home/monsite.com/htdocs/index.php
http://www.monsite.com/page1.php	/home/monsite.com/htdocs/page1.php
http://www.monsite.com/page2.php	/home/monsite.com/htdocs/page2.php
http://www.monsite.com/dossier/test.php	/home/monsite.com/htdocs/dossier/test.php

Patron de conception MVC

Avec MVC : un seul point d'entrée, la requête sera routée en fonction des paramètres

Les URLs peuvent être « plus propres » grâce à la réécriture d'URL

http://www.monsite.com/	/home/monsite.com/htdocs/index.php
http://www.monsite.com/nous-contacter/	/home/monsite.com/htdocs/index.php?controller=page&action=contact
http://www.monsite.com/news/	/home/monsite.com/htdocs/index.php?controller=news&action=index
http://www.monsite.com/news/5-titre-article.html	/home/monsite.com/htdocs/index.php?controller=news&action=view&id=5

Injection de dépendances

- Permet de séparer les « tâches » de la logique applicative
- La dépendance ne sera injectée qu'au cas par cas
- Economie de ressources, temps de chargement accéléré
- Système de facades (appel statique, binding de dépendances)

Chapitre 2 : Installation / Configuration

Configuration de l’environnement, installation de Laravel, configuration d’un projet Laravel

Pré-requis

- Version PHP >= 7.1.3
- Extensions PHP à activer : OpenSSL, PDO, MbString, XML (les autres sont normalement actives par défaut, sauf installation personnalisée)
- Environnement Apache ou NGINX

Solution 1 : Laravel Homestead

Avantages

- Stack complète et compatible
- Solution « Clé en main »
- Performances

Inconvénients

- Première installation longue et fastidieuse
- Plugin vagrant payant pour VMWare (Virtualbox peut être utilisé)
- Ne pas utiliser en production

[Windows] Solution 2 : WSL (W10)

Avantages

- Stack Linux
- Développement sous Windows, exécution sous Linux

Inconvénients

- Première installation pouvant être fastidieuse si non-habitué aux stacks Web Linux
- Performances OK mais non optimales

[Windows] Solution 3 : WAMP

Avantages

- Simplicité et rapidité d'installation
- Prêt à l'emploi

Inconvénients

- Performances médiocres
- Problèmes récurrents avec la version 64 bits de WAMP (nécessité d'installer des packages VC)

Solution 4 : PHPStorm Built-in server

Avantages

- Fourni avec PHPStorm (nécessite l'installation d'une DB si besoin)
- Performances OK (NGINX)

Inconvénients

- Obligation d'utiliser PHPStorm
- Le projet doit être correctement configuré

PHPStorm

- Environnement de développement complet et optimisé pour le web
- Nombreux plugins (dont Laravel)
- Built-in server
- Console intégrée

Composer

- Gestionnaire de paquet PHP
- Gestion des dépendances
- Gestionnaire d'auto-chargement des dépendances
- Liste des paquets disponibles : https://packagist.org/
- Accès à internet requis
- Installateur disponible sous Windows

Ajouter composer dans le PATH lors de l'installation !

Installation de Laravel

Via l'installateur Laravel

- composer global require laravel/installer
- laravel new mysite

Via Composer

- composer create-project --prefer-dist laravel/laravel mysite

Configuration d'un Virtual Host (Apache)

1. Modification du fichier HOSTS Windows

```
127.0.0.1       localhost
127.0.0.1       www.laravel.local
```

2. Création d'un virtual host Apache

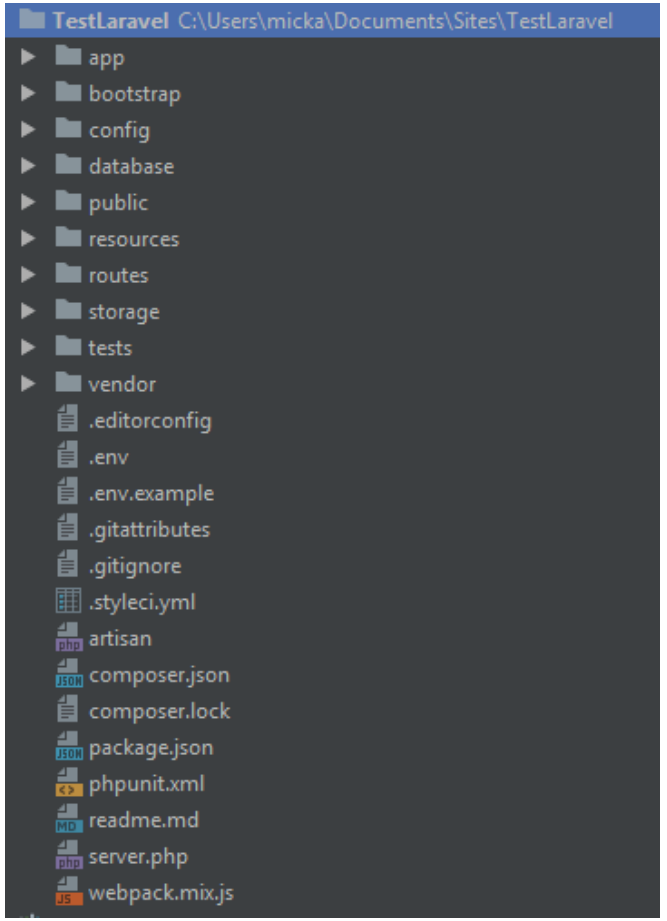
```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        ServerName www.laravel.local
        DocumentRoot /home/mickael/Sites/laravel/public

        <Directory /home/mickael/Sites/laravel/public/>
                Options Indexes FollowSymLinks
                AllowOverride All
                Require all granted
        </Directory>
</VirtualHost>
```

Architecture de Laravel

- app : dossier de l'application web
- database : emplacement pour mettre le « schema » de la DB ainsi que les données à insérer automatiquement
- public : Document root du site, assets compilés
- resources : vues, assets non compilés, traductions
- storage : cache, sessions, logs …
- vendor : dépendances composer

Interdiction formelle de toucher au dossier « vendor » !



Configuration Laravel

- /config
- Fichier configuration général : app.php (à éditer pour les Facades)
- Il n’est plus nécessaire de déclarer les providers de packages externes (Laravel auto discover)
- Possibilité de créer un fichier par package / fonctionnalité
- Récupération d’une variable :

```
config(`[filename].[keyname]', $defaultValue);
```

```
$siteURL = config('app.url', 'http://www.monsite.com');
```

DOTENV Laravel

- Fichier .env
- https://github.com/vlucas/phpdotenv
- Très similaire au DOTENV Symfony

```php
$var = env('APP_URL', 'http://www.monsite.com');
```

Ligne de commande « artisan »

- Permet d'exécuter des commandes relative au projet associé
- Commande : « *php artisan* »
- Lister les commandes disponibles : « *php artisan list* »
- Aide sur une commande

```
php artisan help [COMMAND]
```

Ligne de commande « artisan »

- Générer clé de sécurité : *php artisan key:generate*
  - A quoi sert cette clé ?
- Générer du code : *php artisan make:XXX*
  - Contrôleurs
  - Modèles
  - …
- Nettoyer le cache : *php artisan cache:clear*
  - Possibilité de « catégoriser » les caches via des « stores » (DB, vues …)
- Base de données : *php artisan db:XXX / php artisan migrate*

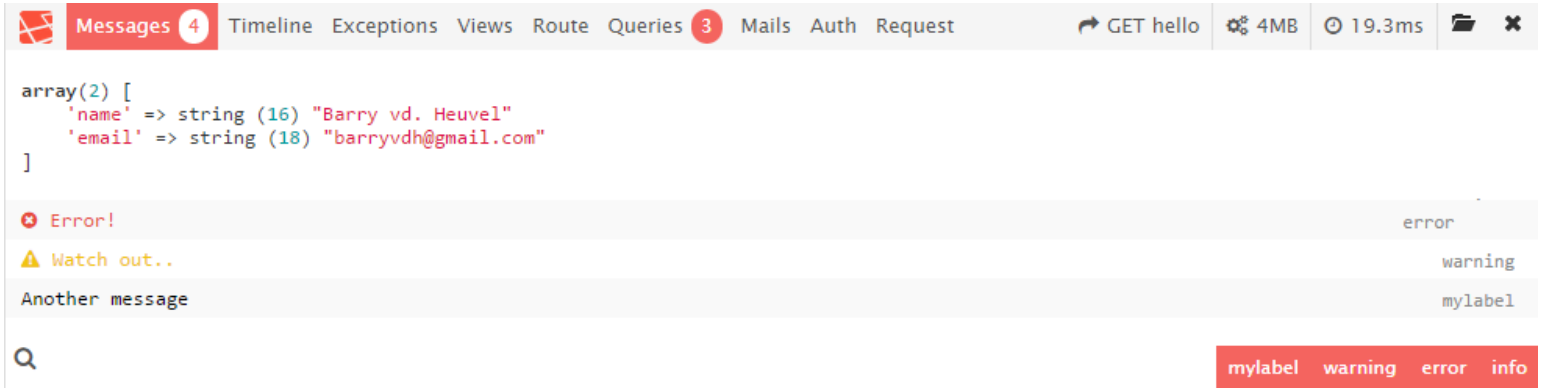
Troubleshoot

- Droits d’écriture dossier storage
- Mod Rewrite activé
- VirtualHost doit pointer dans le sous-dossier « public »
- Serveur rechargé ?
- Clé générée ?

Bonus : Bar de debug en DEV

- Package barryvdh/laravel-debugbar
- A ne mettre qu'en développement (même si une sécurité est présente)

composer require barryvdh/laravel-debugbar --dev



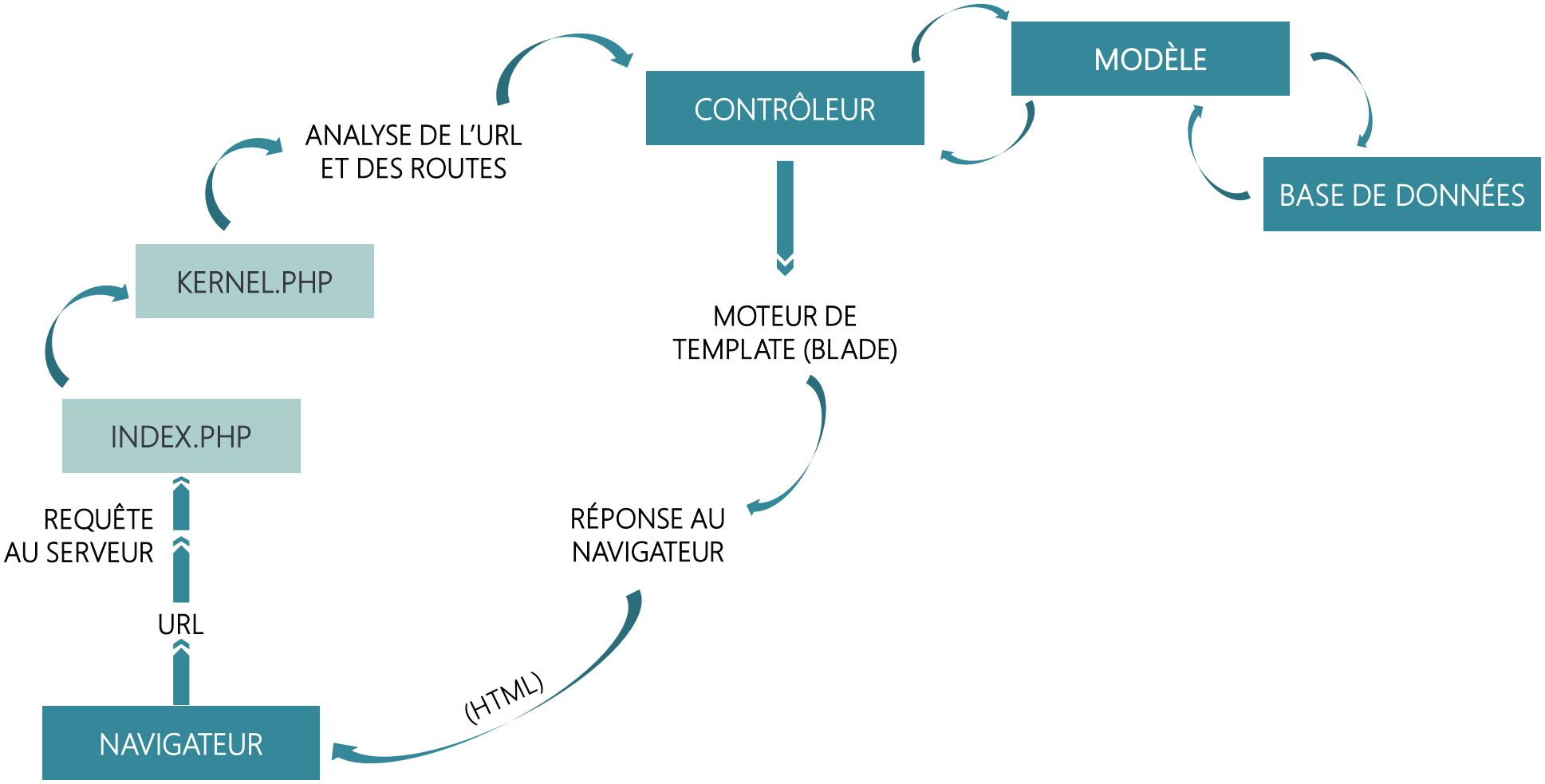
TP 1

Installation de la vidéothèque « Video Turfu »

Chapitre 3 : Routage / Contrôleurs

Comprendre et utiliser le mécanisme de routage afin de faire correspondre une URI à une action

Requête Laravel



Requête Laravel

- Consultation via une URI
  Ex : **/video/15-avatar.html**
- Le serveur Web intercepte la requête et consulte index.php grâce à la réécriture d'URL
- Index.php charge le Kernel, qui appelle le routeur
- Le routeur tente de faire matcher l'URI saisie avec une correspondance présente dans la configuration
- Si une route correspond : appel du contrôleur et de l'action associée

Routeur Laravel

- Routeur écrit en PHP
- Utilisation automatique de la Facade « Route »
- /routes/web.php
- La fonction utilisée exprime le verbe HTTP correspondant

```
Route::get('/', 'HomeController@index');
```

Routeur Laravel

- Possibilité d'écrire une closure au lieu d'appeler un contrôleur (micro-api, fonction simple, déclanchement de mail …)

```
Route::get('foo', function () {
    return 'Hello World';
});
```

Contrôleurs

- /app/Http/Controllers/
- Ne pas oublier le namespace
- Ils étendent Illuminate\Routing\Controller
- Les actions ne nécessitent pas de suffixe (contrairement à Symfony)
- Génération d'un contrôleur
- Possibilité de créer un contrôleur lié à un modèle (-m=XXX) ou contrôleur CRUD (-r)

```
php artisan make:controller TotoController
```

Action

- Une action = une fonction dans le contrôleur
- Retour d'une action
  - Retour d'une response (redirection)
  - Si interdiction : abort(404);
  - Retour d'une view (*return view()*): affichage HTML (via Blade)
  - Retour de texte : affichage brut
  - Retour de tableau : affichage JSON
  - Retour d'une instance de modèle : affichage JSON
  - Aucun retour : page blanche (pas d'erreur)

Routing

- Redirection via router

```
Route::redirect('/news', '/actualites', 301);
```

- Route vers vue

```
Route::view('/welcome', 'welcome', ['name' => 'Taylor']);
```

- Nommage des routes

```
Route::get('/test', 'TestController@test')->name('test');
```

Routing - Paramètres

- Paramètre obligatoire

```
Route::get('/news/{id}', 'TestController@test')->name('test');
```

- Côté contrôleur

```
public function test($id)
{
    dd($id);
}
```

Routing - Paramètres

- Paramètre facultatif

```
Route::get('/news/{id?}', 'TestController@test')->name('test');
```

- Côté contrôleur

```
public function test($id=0)
{
    dd($id);
}
```

Routing - Paramètres

- Expression régulière

```
Route::get('/test/{id?}', 'TestController@test')
    ->where('id', '[0-9]+')
    ->name('test');
```

```
Route::get('/test/{id}/{name}', 'TestController@test')
    ->where([
        'id' => '[0-9]+',
        'name' => '[a-z]+'
    ])
    ->name('test');
```

Routing - Paramètres

- Contrainte globale sur un nom de paramètre
- Dans le fichier RouteServiceProvider.php

```php
public function boot()
{
    Route::pattern('id', '[0-9]+');
    parent::boot();
}
```

Reverse routing

- Principe : générer une URL à partir d'un nom de route
- Utile s'il est nécessaire de modifier par la suite, l'URL d'une page
- Les liens seront automatiquement mis à jour

```php
Route::get('/test/{id}', 'TestController@test')->name('test');
```

```php
$url = route('test', ['id' => 1]);
return response()->redirectToRoute('test', ['id' => 1]);
```

Groupement de routes, préfixes, …

```
Route::prefix('admin')
    ->name('admin.')
    ->namespace('Admin')
    ->group(function () {
        Route::get('/test/{id}', 'TestController@testAdmin')->name('testAdmin');
    });
```

Gestion de sous-domaines

```
Route::domain('admin.monsite.com')->group(function () {
    Route::get('/test/{id}', 'TestController@testAdmin')->name('testAdmin');
});
```

TP 2

Mise en place des principales pages

Chapitre 4 : Vues

Comprendre et utiliser le système de templating Blade. Gestion des assets

Blade ?

- Système de templating
- Contient une majorité de code HTML
- Compilé en PHP
- Avantages :
  - Lisible
  - Principe d’héritage (layout)
  - Cacheable
  - Ecriture simplifiée

Commentaire Blade

- Non compilé…
- … donc invisible dans le code source de la page rendue

```
{{-- Commentaire Blade --}}
```

Affichage d’une variable avec Blade

- Contrôleur

```php
return view('test', ['toto'=>'titi']);
```

- Vue

```
Hello {{ $toto }}
```

Affichage d’une variable avec Blade

- Par défaut, l’affichage est protégé contre les injections XSS (échappement HTML)
- Désactiver l’échappement : {!! $variable !!}

Affichage JSON

```
<script>
    var app = @json($array);
</script>
```

```
<example-component data-exemple='@json($array)'></example-component>
```

Conditions Blade

- Permet de conditioner l'affichage d'une information
- Compatible avec les opérateurs, syntaxe et fonctions PHP
- Cumulation des conditions possible (comme en PHP)
- @if / @elseif / @else

Conditions Blade

- Directive @unless
- Equivalent à @if(!...)

```
@unless ($gender == 'male')
    Madame
@endunless
```

Conditions Blade

- Directive @isset / @empty
- Gestion de l'authentification : @auth / @guest
- Possibilité de créer ses propres conditions

```php
// app/Providers/AppServiceProvider.php
Blade::if('numeric', function ($var) {
    return is_numeric($var);
});
```

```
@numeric('toto')
    C'est numérique
@elsenumeric(5)
    Ok, le deuxième est numérique
@else
    Aucun n'est numérique
@endnumeric
```

Conditions Blade

- @switch / @case / @break / @default
- Exactement le même principe qu'en PHP !

Boucles Blade

- @for / @endfor : Boucle avec compteur
- @foreach / @endforeach : Boucle à l'aveugle
- @forelse / @[COND] / @endforelse : Boucle à l'aveugle avec condition
- @while / @endwhile : While

- Possibilité d'ajouter :
  - @continue(COND)
  - @break(COND)

Boucles Blade

- Au sein d'une boucle, accès à la variable magique $loop
  - $loop->first : true si c'est la première itération
  - $loop->index : Index d'itération (débute à 0)
  - $loop->iteration : Nombre d'itération (débute à 1)
  - $loop->remaining : Nombre d'itération restantes
  - $loop->even / $loop->odd : Pair / impaire
  - $loop->count : Nombre d'itérations

- Itération imbriquée : Possibilité d'accéder au loop parent avec $loop->parent

Layouts

- Evite le code redondant en définissant un gabarit de page unique avec un header / footer fixe
- Principe de blocs surchargeable
- Principe d’héritage afin de restreindre à la vue fille une présentation définie par le layout

Layouts

- Dans le layout :
  - @yield pour définir un bloc à surcharger
  - @section / @show pour définir un bloc à surcharger avec un contenu par défaut

- Dans la vue :
  - @extends([FOLDER].[LAYOUT])
    - Ne pas mettre la partie en « .blade.php »
  - Redéfinir les « section » et « yield » avec @section / @endsection

Assets

- Gestion d’une pile de styles / scripts
- Ajout d’un élément à la pile : @push / @prepend
- Affichage de la stack @stack

```
@push('stackname')
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" crossorigin="anonymous"></script>
@endpush
@prepend('stackname')
    <script src="https://code.jquery.com/jquery-3.4.1.min.js" crossorigin="anonymous"></script>
@endprepend
```

```
@stack('stackname')
```

Assets

- Il faut pouvoir gérer le chemin des assets, quelque soit l'URI du projet
- {{ asset([CHEMIN RELATIF]) }}

```
{{ asset('css/styles.css') }}
```

Laravel MIX

- Gestion d’une stack frontend complète
- Regroupement / minification des ressources
- Repose sur Webpack

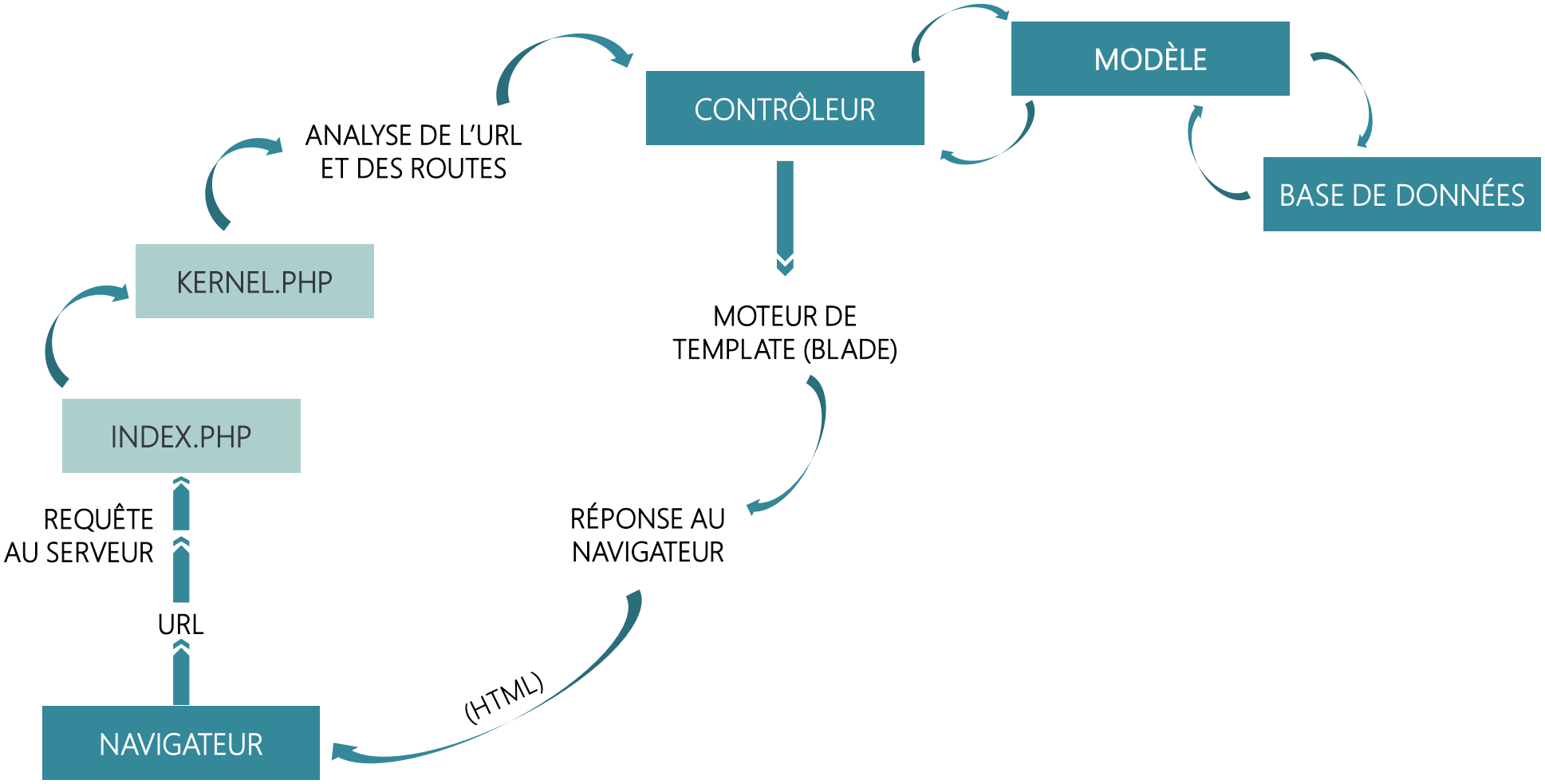
TP 3

Mise en place des vues ainsi que du Layout

Chapitre 5 : Modèles et base de données

Comprendre et utiliser le système de templating Blade. Gestion des assets

Requête Laravel



Problématique PHP

- Ecriture des requêtes en dur dans le modèle
- Comment être sûr que la requête est compatible avec le SGBD ?
- Allers / retours sous forme de tableaux : peu fiable

Solution Eloquent

- Utiliser un ORM !
- On gère des instances d’entité
- Abstraire la communication avec le SGBD
- Méthodes d’enregistrement, récupération …

Configuration Eloquent

- .env
  - DB_*

- Création de la base de données à la main

Fonctionnement de l'ORM

Entité

- 1 table = 1 entité
- 1 instance de l'entité = 1 row de la table
- Pas besoin de créer de propriétés + getters / setters

Base de données

- Le schéma est géré via des migrations
- Fichiers externes contenant le schéma de la base de données
- Synchro via Git

Migrations

- Située dans /database/migrations/
- Utilisation de la facade Schema et Blueprint
- Commande pour générer une migration
- --create=XXX pour créer une table
- --table=XXX pour modifier une table
- create_TABLE_table : création

```
php artisan make:migration create_[TABLE_NAME]_table
```

Migrations

- Deux fonctions :
  - Up() pour la migration ascendante
    - Schema::create
    - Schema::table
  - Down() pour le rollback
    - Schema::drop
    - Schema::dropIfExist

Migrations

- Types de colonnes : https://laravel.com/docs/5.8/migrations#columns
- ->timestamps() : permet de créer les champs created_at, updated_at
- ->softDelete() : gestion des soft deletes
  - Très utiles dans le cas d'une gestion avec des relations, car la cascade n'est plus à gérer
  - Permet d'annuler une suppression par erreur

Lancer une migration

- Migrer : **php artisan migrate**
  - --force pour forcer immédiatement sur un environnement de production
- Rollback sur la dernière migration : **php artisan migrate:rollback**
  - Paramètre optionnel –step=X permettant de reculer de X batchs
- Rollback l'intégralité des migrations : **php artisan migrate:reset**
- Rollback l'intégralité des migrations et les rejouer :
  php artisan migrate:refresh
- Vider la base de données et jouer les migrations :
  php artisan migrate:fresh

Modèle

- Etend la classe « Model » d'Eloquent
- 1 instance de modèle représentant 1 row, le nom du modèle est au singulier
- Ligne de commande :

```
php artisan make:model Book
```

- Possibilité de créer un modèle et sa migration en même temps

```
php artisan make:model Book -m
```

Modèle

- $table => nom de la table
  - Facultatif si le nom de la table correspond au pluriel du nom du modèle
- $primaryKey => nom de la colonne de clé primaire
  - Facultatif si la clé primaire se nomme « id »
- $timestamp => booléen permettant de préciser si nous avons mis les timestamps en base de données
  - Par défaut : true

Modèle

- $attributes => permet de préciser des valeurs par défaut à certains attributs
- $fillable / $guarded => permet de préciser quels champs peuvent être rempli par l'utilisateur, lesquels ne peuvent pas l'être
- $dates => quels champs sont de types dates / heures (mutator)
- $hidden => quel champs masquer lors de l'affichage front brut (json) d'une entité

Modèle

- Si votre modèle utilise le soft delete, préciser le trait « SoftDelete »

Seeders

- Avoir un schéma de la base de données, c'est bien…
- … avoir des données de test, c'est mieux

- Le seeder permet d'alimenter la base de données avec des données de test

```
php artisan make:seeder BooksTableSeeder
```

Seeders

- Coder le seed dans la fonction run()

```
DB::table('books')->insert([
    'name' => Str::random(10),
    'author' => Str::random(10),
    'publication_date' => date('Y-m-d H:i:s')
]);
```

Seeders

- Meilleur moyen pour les seeds : utiliser les factories !