



RÉSUMÉ THÉORIQUE – FILIÈRE DÉVELOPPEMENT WEB FULL STACK

M205 : « Développement en back-end »

Elaboré par :

Mariam MAHDAOUI Formatrice à ISTA Sidi Moumen CASABLANCA

Mohamed CHCHAB Formateur à ISTA BENGUERIR

Adapté par:

Asmae YUALA Formatrice à ISTA ALADARISSA FES



120 heures



Equipe de rédaction de validation

Equipe de rédaction :

Cours rédigé par:

- Mme MARIAM MAHDAOUI : Formatrice en Développement Digital
- M. MOHAMED CHCHAB : Formateur en Développement Digital

Adapté par:

- Mme Asmae YOUALA : Formatrice en Développement Digital – option web full stack



SOMMAIRE

1. Découvrir le Framework PHP Laravel

Découvrir les notions fondamentales des Frameworks PHP
Préparer l'environnement de Laravel

2. Programmer avec Laravel

Installation des outils pour Laravel
Créer un nouveau projet Laravel
Architecture d'un projet Laravel
Laravel Artisan
Lancer un projet Laravel

3. Approfondir la programmation Laravel

Gérer la sécurité
Interagir avec la base de données
Manipuler l'ORM Eloquent

4. Administrer un site à l'aide d'un CMS

Manipuler les éléments essentiels d'un CMS
Personnaliser graphiquement un site à l'aide d'un CMS
Manipuler les outils avancés d'un CMS

MODALITÉS PÉDAGOGIQUES



1

LE GUIDE DE SOUTIEN

Il contient le résumé théorique et le manuel des travaux pratiques



2

LA VERSION PDF

Une version PDF est mise en ligne sur l'espace apprenant et formateur de la plateforme WebForce Life



3

DES CONTENUS TÉLÉCHARGEABLES

Les fiches de résumés ou des exercices sont téléchargeables sur WebForce Life



4

DU CONTENU INTERACTIF

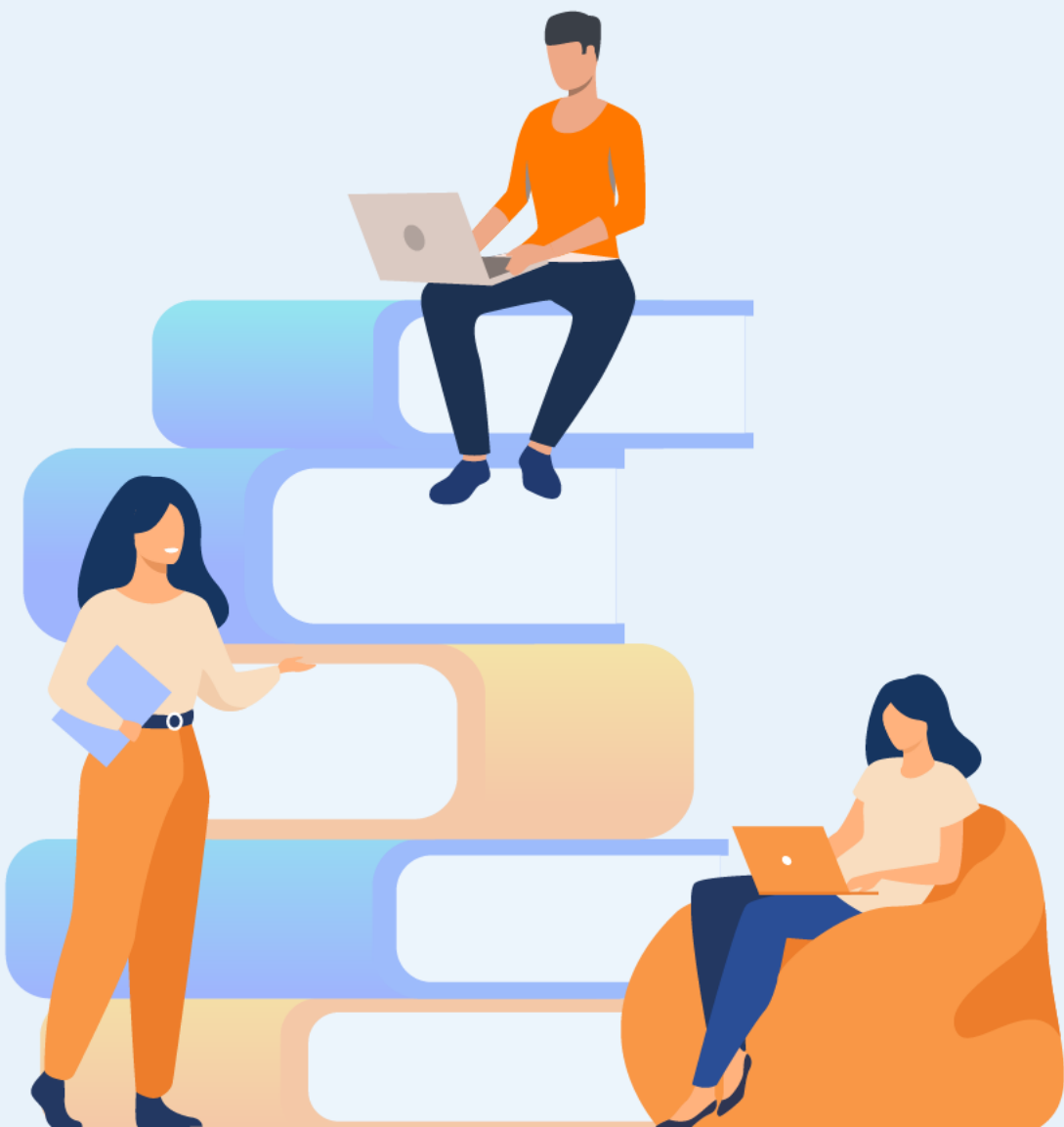
Vous disposez de contenus interactifs sous forme d'exercices et de cours à utiliser sur WebForce Life



5

DES RESSOURCES EN LIGNES

Les ressources sont consultables en synchrone et en asynchrone pour s'adapter au rythme de l'apprentissage



PARTIE 1

Découvrir le Framework PHP Laravel

Dans ce module, vous allez :

- Découvrir les notions fondamentales des Frameworks PHP
- Préparer l'environnement de Laravel



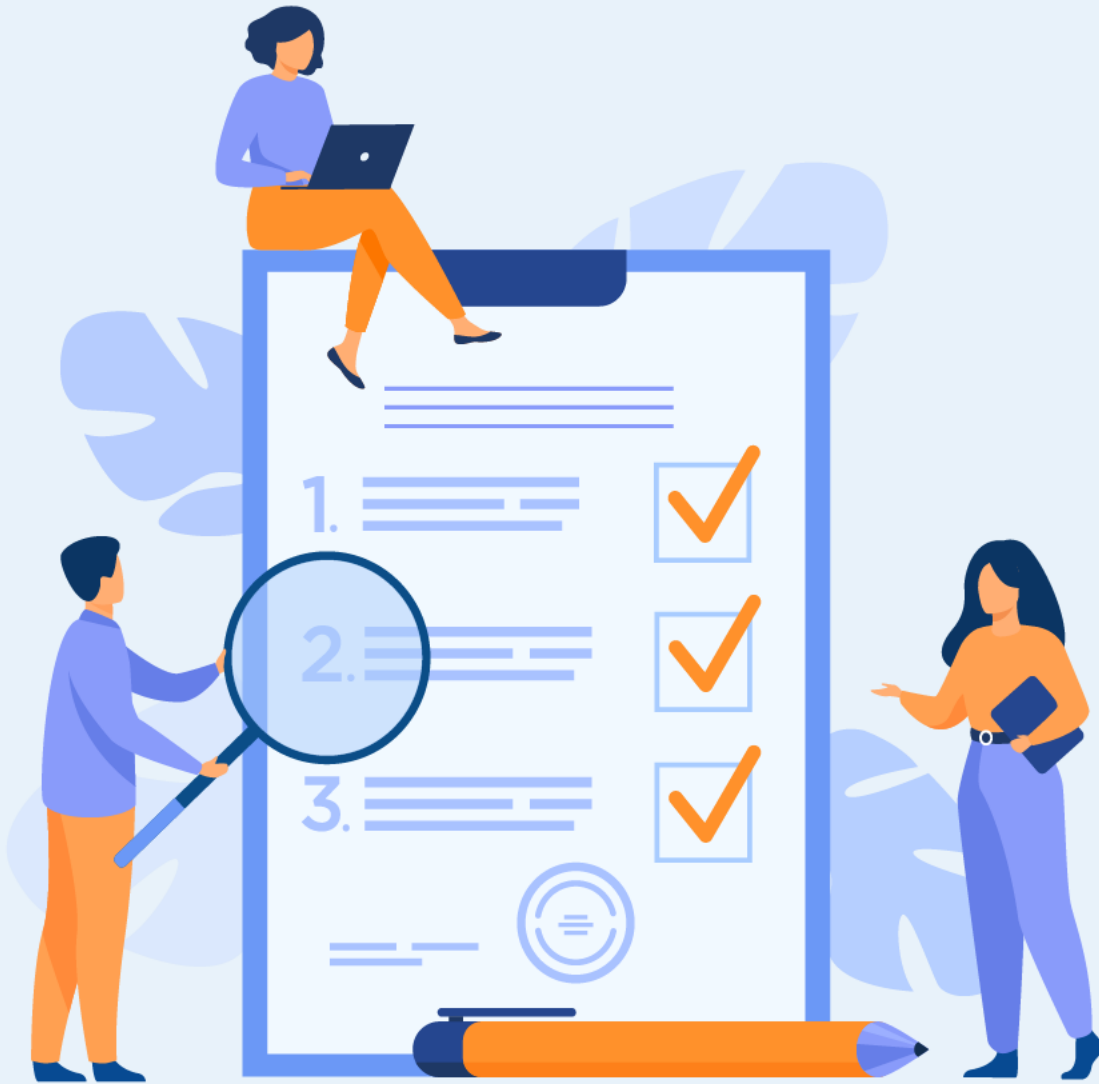
20 heures

CHAPITRE 1

Découvrir les notions fondamentales des Frameworks PHP

Ce que vous allez apprendre dans ce chapitre :

1. **Présentation des Frameworks PHP**
2. L'architecture MVC
3. Intérêt du Framework Laravel



02 heures

01 - Découvrir les notions fondamentales des Frameworks PHP

Présentation des Frameworks PHP



Qu'est-ce qu'un Framework web ?

- Un Framework est **une boîte à outils** pour **aider** les développeurs dans la réalisation de leurs tâches. **Frame (cadre) et work (travail).**
- Un Framework contient des composants autonomes qui permettent de **résoudre les problèmes souvent rencontrés par les développeurs** (CRUD, arborescence, normes, sécurités, etc.).
- Un Framework **n'est pas uniquement une boîte à outils**. Il peut aussi désigner une **méthodologie**.
- Un framework est donc perçu comme un **squelette**, regroupant chaque os et articulations (ici composants/librairies) de manière harmonieuse mais surtout de manière fiable !

Pourquoi utiliser un framework ?

L'objectif de l'utilisation d'un Framework est de :

- **maximiser la productivité** du développeur qui l'utilise.
- **faciliter la maintenance** de l'application Web réalisée.

Voici les trois raisons à utiliser un Framework :

- **Rapidité** : le Framework permet un gain de temps et une livraison beaucoup plus rapide qu'un développement à zéro.
- **Organisation** : l'architecture d'un Framework favorise une bonne organisation du code source (modèle MVC par exemple).
- **Maintenabilité** : l'organisation du Framework facilite la maintenance du logiciel et la gestion des évolutions.

01 - Découvrir les notions fondamentales des Frameworks PHP

Présentation des Frameworks PHP



Les Frameworks PHP les plus populaires

Il est difficile d'obtenir une liste définitive des frameworks PHP. Wikipédia liste 40 frameworks PHP. Voici quelques-uns des meilleurs frameworks PHP en usage aujourd'hui :

1. Laravel
2. Symfony
3. CodeIgniter
4. CakePHP
5. Zend Framework / Laminas Project
6. Yii Framework 2
7. Slim Framework
8. ...

01 - Découvrir les notions fondamentales des Frameworks PHP

Présentation des Frameworks PHP



Le framework Symfony

Symfony est un excellent choix pour les sites web et les applications qui doivent être évolutifs, il a plusieurs fonctionnalités:

- Supporte la **plupart des bases de données**.
- a son propre ORM **Doctrine**.
- Utilise le moteur de templating **Twig**, qui est facile à apprendre, rapide et sûr.
- **Packagist** * liste plus de **4 000 paquets Symfony** téléchargeable prête à utiliser.
- dispose d'un support **support professionnel**(support commercial de Sensio Labs), contrairement à la plupart des autres frameworks PHP.
- S'intègre facilement avec des frameworks populaires comme Drupal.



* Packagist est une sorte de répertoire public de packages PHP utilisables via Composer. Composer est un gestionnaire de dépendances entre applications et librairies.

Composer permet de gérer pour chaque projet, la liste des modules et bibliothèques nécessaires à son fonctionnement ainsi que leurs versions. Il est utilisable via la console en ligne de commande. De plus, il permet de mettre en place un système d'autoload pour les bibliothèques compatibles.

Le framework Laravel

- Laravel est présenté comme « **Le framework PHP pour les artisans du web** ».
- Lancé en 2011, Laravel se trouve en tête des classements grâce à ses fonctionnalités,
- Laravel :
 - Offre une solution **MVC** complète.
 - a son propre ORM **Eloquent ORM** .
 - Utilise le moteur de templating **Blade**. On peut utiliser PHP dans Blade, contrairement aux autres.
 - **Packalyst**, une collection de paquets Laravel, compte plus de 15 000 paquets utilisable .
 - Dispose de l'outil de ligne de commande **Artisan Console** qui permet d'automatiser les tâches répétitives et de générer rapidement du code squelette.
 - Dispose de plusieurs outils utiles tels que **Mix** pour la compilation des actifs CSS et JS, et **Socialite** pour l'authentification OAuth.
 - Bénéficie d'une large communauté de développeurs



Le Zend framework/ Laminas Project

Zend Framework est un framework PHP établi de longue date qui est maintenant en transition vers Laminas Project. La migration vers Laminas est fortement recommandée, car Zend n'est plus mis à jour.

- C'est un framework complètement orienté objet.
- Basé sur une méthodologie agile.
- Il suit strictement le modèle de conception MVC
- La communauté de Laminas dispose d'un forum et d'un groupe Slack pour la collaboration et le support.
- Le seul inconvénient de Zend réside dans le fait qu'il n'est pas aussi facile à maîtriser.



Le Yii Framework 2

Le nom de ce framework, **Yii**, signifie « simple et évolutif » en chinois. Il signifie également « **Yes, It Is !** ». Les principaux avantages de l'utilisation du framework Yii sont:

- C'est assez facile à installer et à démarrer
- Il est préchargé avec un modèle Bootstrap
- Il est livré avec un design élégant pour commencer rapidement
- C'est un framework PHP qui est également livré avec un débogueur
- Il est livré avec un générateur de code de classe extrêmement puissant appelé **Gii**. Le **générateur de code Gii** peut rapidement construire un squelette de code pour vous, ce qui permet de gagner du temps.
- La communauté Yii offre un support en direct via Slack ou IRC.



Ce que vous devez savoir avant d'utiliser un framework PHP

Avant d'utiliser un framework PHP, il faut connaître :

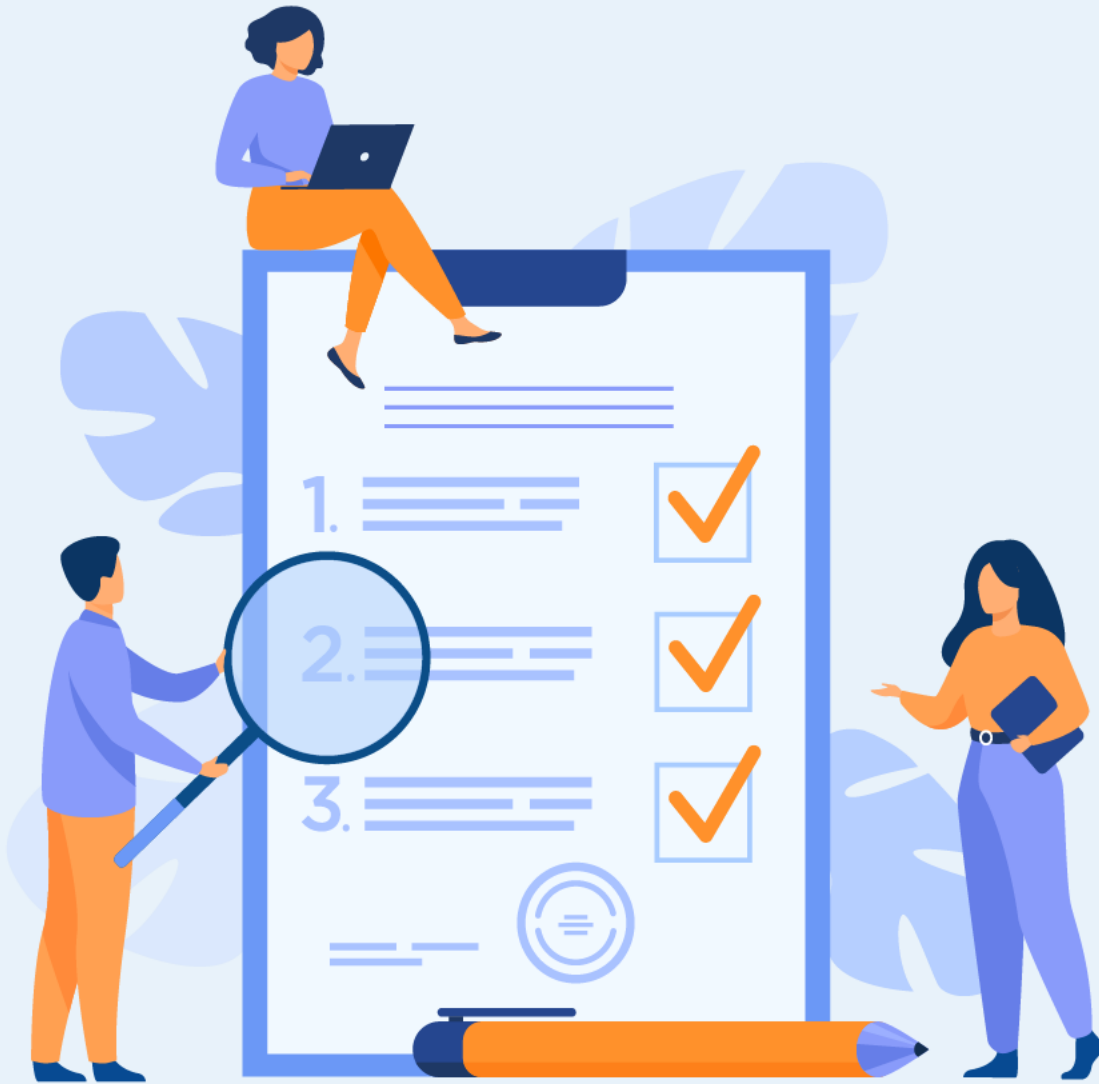
- Le **langage PHP** lui-même. La plupart des frameworks fonctionnent avec PHP version 7.2 ou supérieure.
- Le PHP **orienté objet**, car la plupart des frameworks PHP modernes sont orientés objet.
- **Les bases de données et la syntaxe SQL**. Chaque framework PHP a sa propre liste de bases de données supportées.
- **ORM** (Object-Relational-Mapping) est une méthode d'accès aux données d'une base de données utilisant une syntaxe orientée objet au lieu d'utiliser le langage SQL.
- De nombreux frameworks PHP ont leur propre ORM intégré.
- Le fonctionnement **des serveurs web** comme Apache et Nginx.
- Architecture **MVC** (Modèle – Vue – Contrôleur): Les frameworks PHP suivent généralement le modèle de conception MVC
- L'utilisation d'une **interface en ligne de commande** (commande-line interface ou CLI) permet de se sentir à l'aise dans un framework PHP.

CHAPITRE 1

Découvrir les notions fondamentales des Frameworks PHP

Ce que vous allez apprendre dans ce chapitre :

1. Présentation des Frameworks PHP
2. **L'architecture MVC**
3. Intérêt du Framework Laravel



02 heures

L'architecture MVC

M



Model : Les modèles seront les éléments qui se chargeront des échanges avec la base de données (CRUD). On ne mettra pas de traitement dans ces fichiers, uniquement des requêtes.

V



View : Les vues contiendront uniquement le code HTML destiné à structurer les pages.

C



Controller : Les contrôleurs contiendront toute l'intelligence de l'application, le traitement des données en vue de leur affichage, par exemple.

L'architecture MVC

MVC (Modèle-Vue-Contrôleur) est un modèle de conception, il est donc indépendant du langage de programmation. Il met l'accent sur la séparation entre la logique métier et l'affichage du logiciel.

- Les modèles et les contrôleurs sont généralement des classes.
- Les vues sont généralement des Templates HTML ou PDF

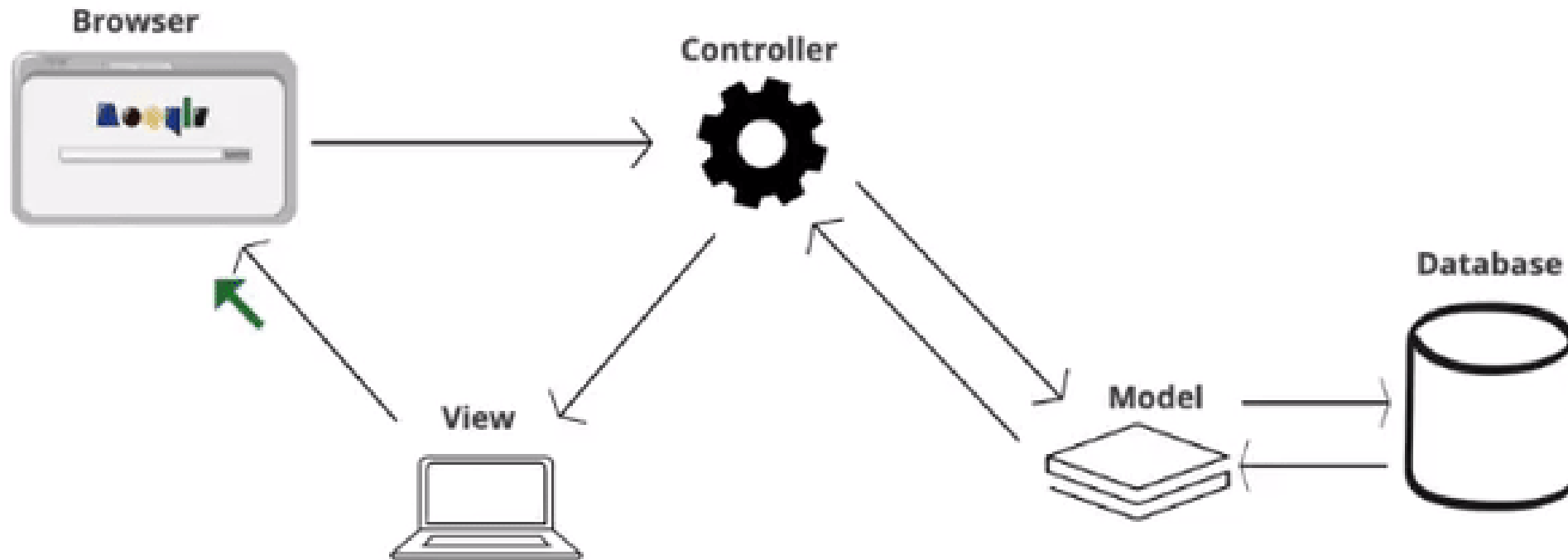
Laravel propose le modèle MVC mais ne l'impose pas.

01 - Découvrir les notions fondamentales des Frameworks PHP

Présentation des Frameworks PHP

L'architecture MVC : Fonctionnement:

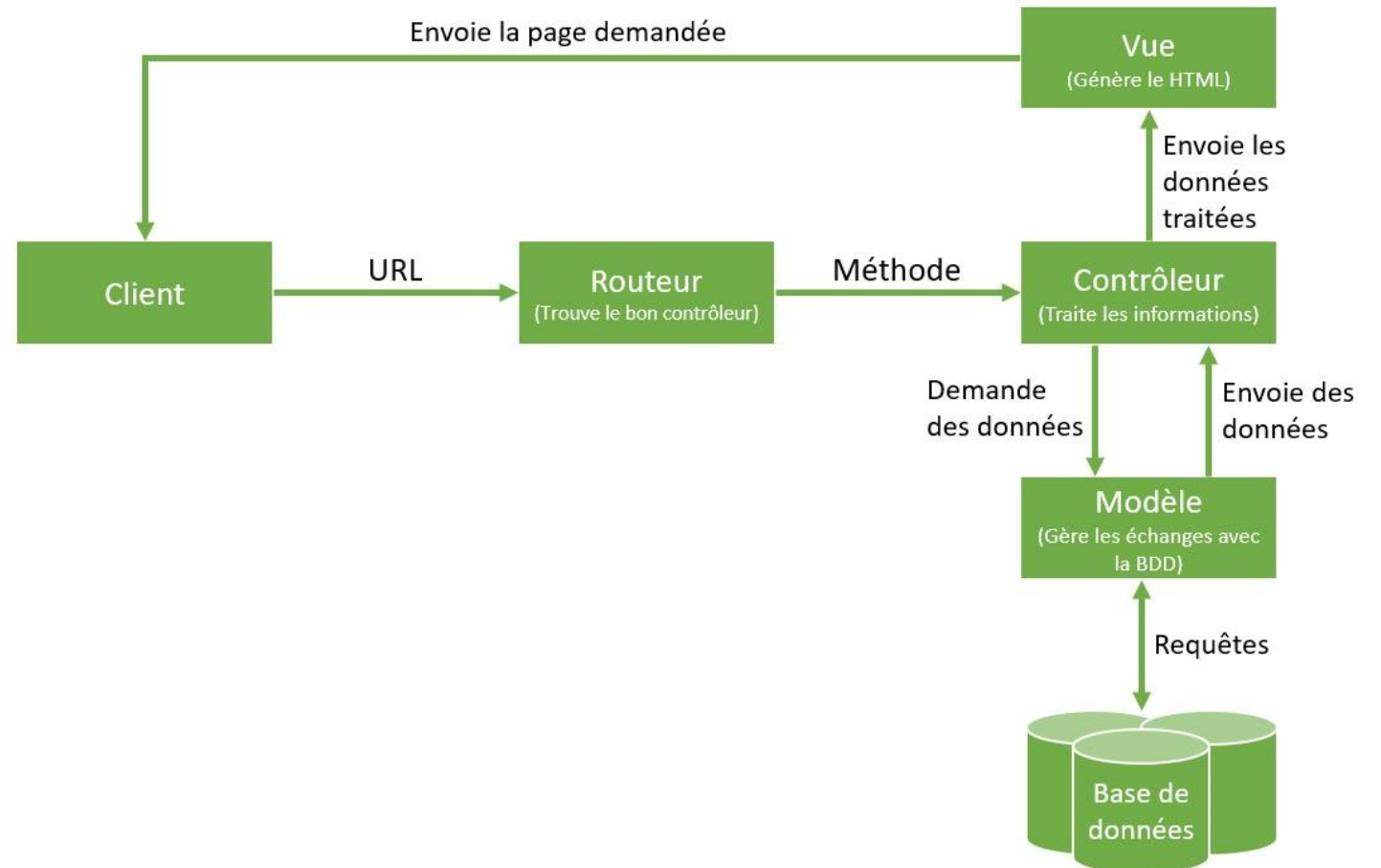
1. Le contrôleur reçoit des requêtes de l'utilisateur.
2. Le contrôleur va demander au modèle d'effectuer certaines actions (lire des informations depuis une base de données,...) et de lui renvoyer les résultats.
3. Le contrôleur va *adapter* ce résultat et le donner à la vue.
4. Enfin, le contrôleur va renvoyer la nouvelle page HTML, générée par la vue, à l'utilisateur.



L'architecture MVC : Fonctionnement:

Le routeur

- Dans la structure MVC, un **seul** et **unique** fichier est le point **d'entrée** de l'application, quelle que soit la page affichée.
- Il est **systématiquement appelé**, et envoie la demande au bon contrôleur.
- Il est chargé de **trouver le bon chemin** pour que l'utilisateur récupère la bonne page, d'où le nom de **routeur**.



CHAPITRE 1

Découvrir les notions fondamentales des Frameworks PHP

Ce que vous allez apprendre dans ce chapitre :

1. Présentation des Frameworks PHP
2. L'architecture MVC
3. **Intérêt du Framework Laravel**



02 heures

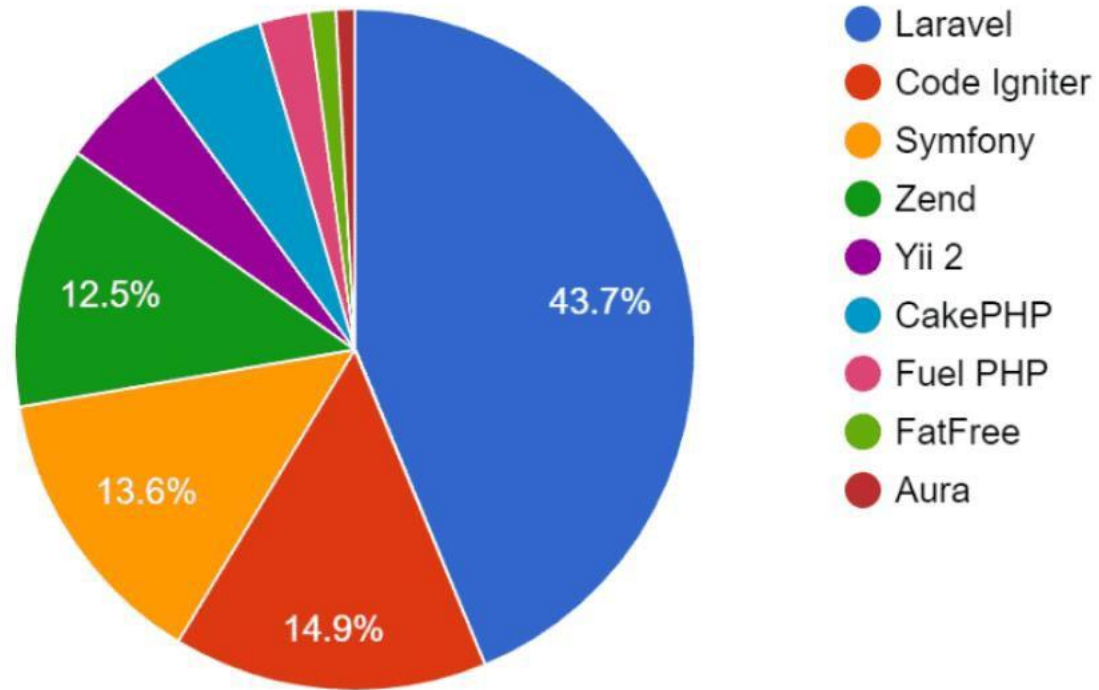
01 - Découvrir les notions fondamentales des Frameworks PHP

Intérêt du Framework Laravel



Pourquoi choisir le Framework Laravel ?

Laravel est le framework PHP le plus utilisé au monde



<https://laravel.sillo.org/cours-laravel-8-les-bases-presentation-generale/>

Pourquoi choisir le Framework Laravel ?

1. **une sécurité de haut niveau:** Avec Laravel, votre application Web ne présente aucun risque d'injections SQL involontaires et cachés.
2. **Performance:** Laravel propose divers outils qui aident les développeurs à améliorer leurs performances.
3. **Bibliothèques orientées objet:** Laravel possède des bibliothèques orientées objet et d'autre pré installées, qui ne se trouvent dans aucun autre framework PHP, comme la bibliothèque d'authentification
4. **Documentation et communauté:** Laravel possède une puissante communauté de développeurs qui fournit en une assistance en permanence.
5. **Tests unitaires:** Avec les tests unitaires de Laravel, chaque module de votre application Web est testé avant la mise en ligne du site.
6. **Intégration aux services de messagerie :** Il fournit également des pilotes pour Mailgun, SMTP, Mandrill, SparkPost, la fonction de courrier électronique de PHP et Amazon SES.
7. **Créateur d'applications multilingues:** Le framework Laravel vous aide donc à créer facilement et rapidement votre application Web dans différentes langues.
8. **Tutoriels Laracasts :** Laravel propose des fonctionnalités de Laracasts, un mélange de tutoriels vidéo gratuits et payants qui vous montrent comment utiliser Laravel pour le développement.

Résumé

- Un framework fait gagner du temps et donne l'assurance de disposer de composants bien codés et fiables.
- Laravel est un framework novateur, complet, qui utilise les possibilités les plus récentes de PHP et qui est impeccablement codé et organisé.
- La documentation de Laravel est complète, précise et de plus en plus de tutoriels et exemples apparaissent sur la toile.
- Laravel adopte le patron MVC, mais ne l'impose pas.
- Laravel est totalement orienté objet.

CHAPITRE 2

Préparer l'environnement de Laravel

1. **Installation des outils pour Laravel**
2. Créer un nouveau projet Laravel
3. Architecture d'un projet Laravel
4. Laravel Artisan
5. Lancer le projet Laravel



02 - Préparer l'environnement de Laravel

Installation des outils pour Laravel



Les outils pour Laravel

Pour travailler avec Laravel, vous avez besoin de :

1. PHP
2. un serveur Local (Apache ou Nginx), permettant l'exécution du PHP
3. un serveur de base de données pour gérer vos bases de données.
4. Composer, un gestionnaire de dépendances pour PHP
5. IDE et Editeurs de code (Visual Studio Code, Atom, PHPStorm ...)
6. Navigateurs Web : Laravel est compatible sur tout les navigateurs récents (Chrome, Firefox ou Safari)

Nous vous recommandons d'installer les dernières versions de ces logiciels ayant une version PHP plus récente (PHP 8.X pour Laravel 10, PHP 7.4 pour Laravel 8).

Document officielle : <https://laravel.com/docs/10.x>

02 - Préparer l'environnement de Laravel

Installation des outils pour Laravel



Téléchargement et installation de PHP

- Pour installer PHP vous allez devoir vous rendre sur la page de téléchargement Windows du site officiel : <http://windows.php.net/download/>.
- Installer la version 8.1 ou plus
- Vérifier que php est bien installé en exécutant la commande dans la console `php --version`.



Configuration des extensions nécessaires pour PHP

Les extensions **ZIP**, **PDO**, **Tokenizer**, **OpenSSL** et **Mbstring** de PHP doivent être activées.

Dans le fichier Le fichier de configuration de PHP **php.ini** , décommenter les lignes ci-dessous en enlevant le point virgule au début des lignes :

```
;extension=php_mbstring.dll  
;extension=php_openssl.dll  
;extension=zip
```

02 - Préparer l'environnement de Laravel

Installation des outils pour Laravel



Composer - Définition

Composer est un logiciel gestionnaire de dépendances libre écrit en PHP. Il permet à ses utilisateurs de déclarer et d'installer les bibliothèques dont le projet principal a besoin. Il permet de télécharger et de mettre à jour des bibliothèques externes.

Les bibliothèques externes permettent de réutiliser le code écrit par d'autres personnes pour simplifier le développement.

Exemple:

- Pour gérer des dates, vous pouvez utiliser **Carbon**
- Pour gérer les paiements Paypal, vous pouvez utiliser la bibliothèque officielle **PayPal PHP SDK**).



Composer permet également de créer des projets Laravel et de télécharger le framework.

Le framework Laravel est un simple assemblage de plusieurs dizaines de bibliothèques.

02 - Préparer l'environnement de Laravel

Installation des outils pour Laravel



Installation de Composer

Pour installer Composer, il suffit de télécharger un installeur <https://getcomposer.org/download/> et téléchargez **Composer-Setup.exe**.

Vérifiez lors de l'installation que le chemin par défaut vers PHP est bien **C:\PHP\php.exe**. car Composer est un fichier PHP et a besoin d'être exécuté.

Vérifications

Pour vérifier que tout fonctionne exécuter `composer` sur la ligne de commande comme suit:

```
C:\Users\HINNOVIS>composer

Composer version 2.3.10 2022-07-13 15:48:23
```

ici on peut voir que j'ai la version 2.3.10 de composer installée.

CHAPITRE 2

Préparer l'environnement de Laravel

1. Installation des outils pour Laravel
- 2. Créer un nouveau projet Laravel**
3. Architecture d'un projet Laravel
4. Laravel Artisan
5. Lancer le projet Laravel



02 - Préparer l'environnement de Laravel

Installation de Laravel



Utiliser l'installateur de Laravel

- **Installer Laravel dans notre ordinateur** : Ouvrez votre terminal (invite de commande) et tapez la ligne suivante:

C:\> Invite de commandes

```
Microsoft Windows [version 10.0.19043.1766]  
(c) Microsoft Corporation. Tous droits réservés.
```

```
C:\Users\HINNOVIS>composer global require laravel/installer
```

require: ajoute la bibliothèque en paramètre au fichier *composer.json* et l'installe.

Créer notre premier projet laravel (project_laravel) – **Méthode 1** :

Taper la commande ***laravel new project_laravel*** en plaçant le terminal dans le répertoire de travail (ex. www ou htdocs):

C:\> Invite de commandes - laravel new project_laravel

```
Microsoft Windows [version 10.0.19043.1766]  
(c) Microsoft Corporation. Tous droits réservés.
```

```
C:\Users\HINNOVIS>cd C:\Apache24\htdocs
```

```
C:\Apache24\htdocs>laravel new project_laravel
```

02 - Préparer l'environnement de Laravel

Créer un nouveau projet Laravel




Créer un nouveau projet Laravel avec Composer – Méthode 2:

Créer un deuxième projet à la racine du serveur nommé `project_laravel2` : avec la commande suivante:

- Dirigez-vous vers votre dossier à la racine de votre serveur `www` ou `htdocs` et exécuter la commande :

composer create-project laravel/laravel project_laravel2

 Invite de commandes -

```
Microsoft Windows [version 10.0.19043.1766]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\HINNOVIS>cd C:\Apache24\htdocs

C:\Apache24\htdocs>composer create-project laravel/laravel project_laravel2
```

Ici nous avons demandé à composer de créer une installation Laravel dans le dossier «`project_laravel2`». Cette commande installera automatiquement la dernière version stable de Laravel.

CHAPITRE 2

Préparer l'environnement de Laravel

1. Installation des outils pour Laravel
2. Créer un nouveau projet Laravel
3. **Architecture d'un projet Laravel**
4. Laravel Artisan
5. Lancer le projet Laravel



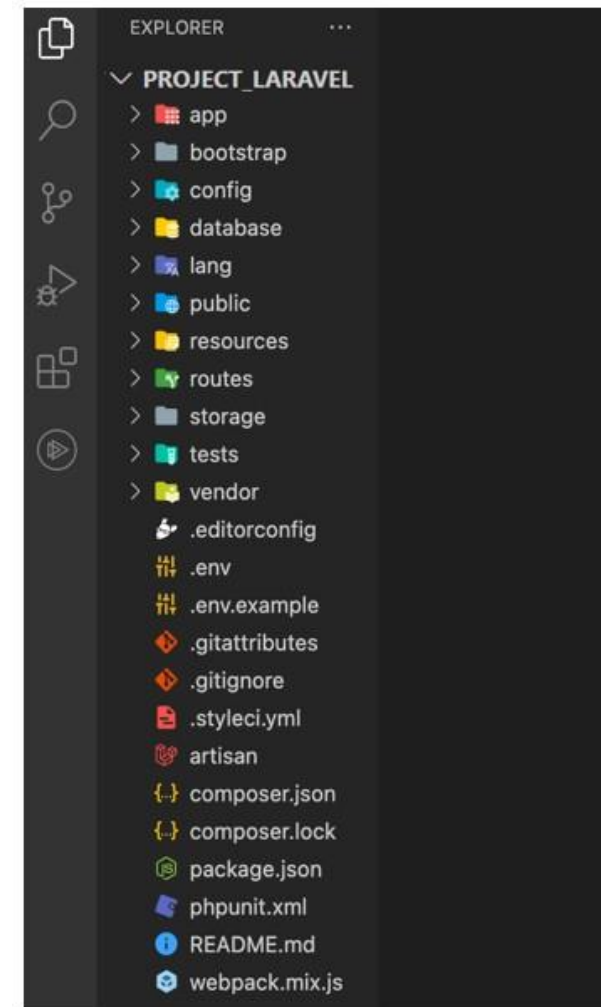
02 - Préparer l'environnement de Laravel

Architecture d'un projet Laravel



Architecture d'un projet Laravel

Ouvrez le dossier project_laravel avec l'éditeur Visual Code ou autre ,
vous devez avoir ceci comme résultat :



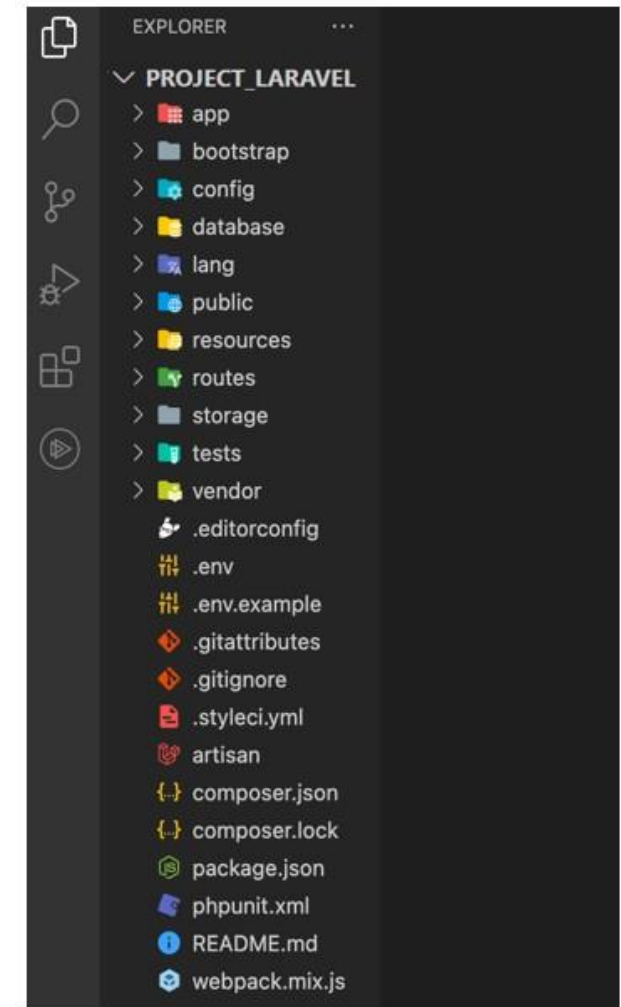
02 - Préparer l'environnement de Laravel

Architecture d'un projet Laravel



Les répertoires racines d'un projet Laravel

- ❑ **/app** : contient le code principal de votre application. Il contient deux répertoires principal http et Models.
 - ❑ **Le répertoire Http** : contient les contrôleurs et les middleware.
 - ❑ **Le répertoire Models** : contient toutes les classes de modèle Eloquent.
- ❑ **/config** : contient tous les fichiers de configuration de votre application, authentification, namespace, mails, base de données etc...
- ❑ **/database** : contient vos migrations de base de données, les données factices (faux data) de vos modèles.
- ❑ **/lang** : contient tous les fichiers de langue de votre application. Vous permettant de gérer plusieurs langues dans votre site web. *Ce répertoire est devenu racine que à partir de la version 9 de Laravel, dans les versions ultérieures, le répertoire lang se trouve dans le dossier public.*



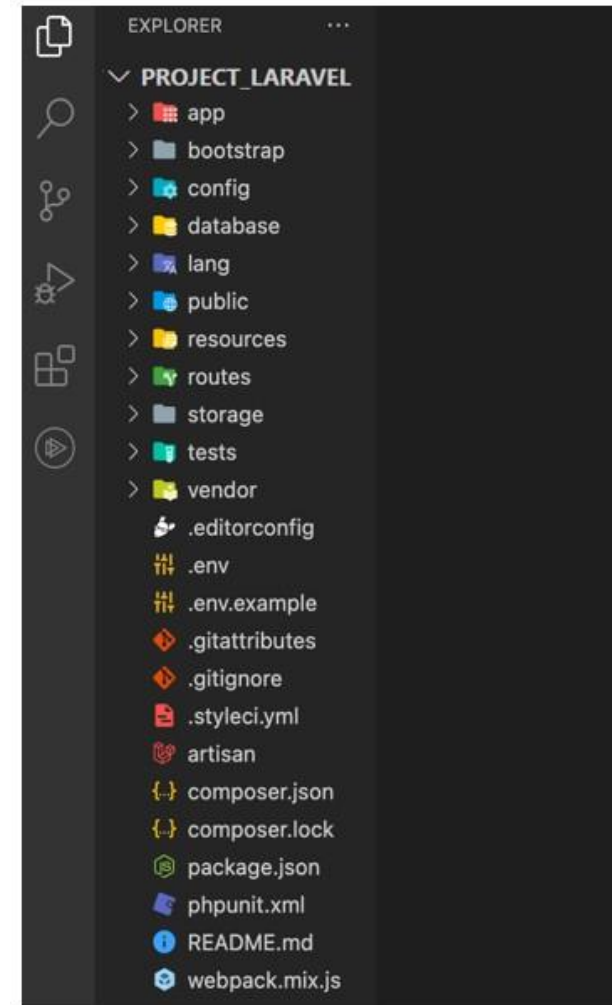
02 - Préparer l'environnement de Laravel

Architecture d'un projet Laravel



Les répertoires racines d'un projet Laravel

- ❑ **/public:** par convention comme pour la majorité des frameworks il s'agit du seul dossier accessible depuis le serveur où les fichiers sont accessibles depuis votre site (images, feuilles de style et scripts principalement). Contient le fichier **index.php**, qui est le point d'entrée pour toutes les demandes entrant dans votre application et configure le chargement automatique.
- ❑ **/resources:** contient vos **vues** ainsi que vos fichiers bruts non compilés tels que CSS, SASS ou JavaScript
- ❑ **/routes :** contient toutes les définitions des URLs pour votre application.
Par défaut, 4 fichiers de route sont inclus avec Laravel : web.php, api.php, console.php et channels.php.
 - **Le fichier web.php:** contient des routes placés dans le **RouteServiceProvider** (web groupe middleware), qui fournit l'état de la session, la protection CSRF et le cryptage des cookies.



02 - Préparer l'environnement de Laravel

Architecture d'un projet Laravel

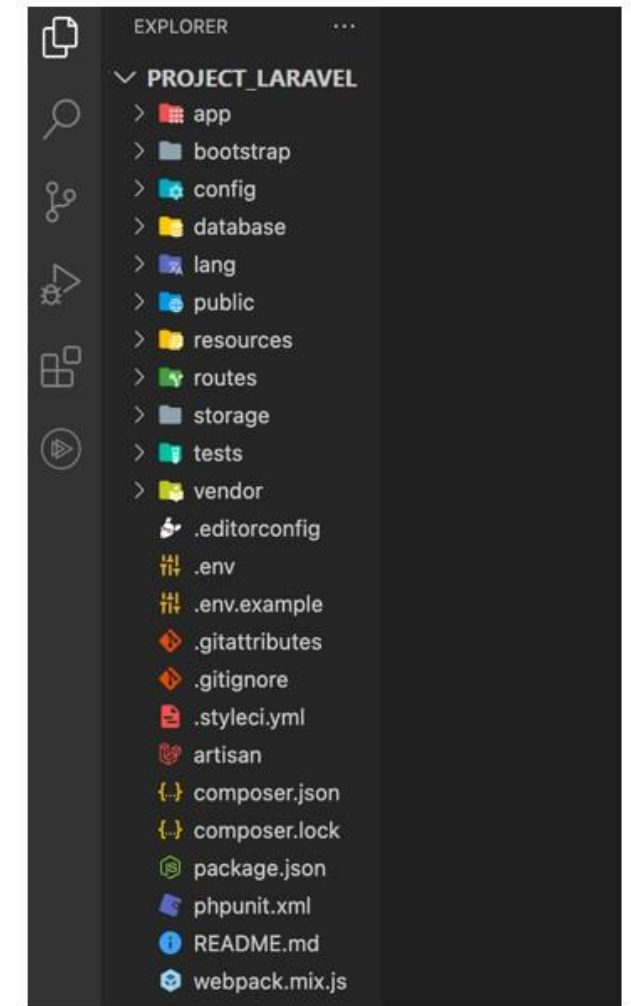


- ❑ **/storage:** contient vos logs, modèles de vues (blades) compilés, sessions basées sur des fichiers, fichiers caches et autres fichiers générés par le framework.

Ce répertoire est divisé en répertoires **app**, **framework** et **logs**.

- **storage/app** peut être utilisé pour stocker tous les fichiers générés par votre application.
- **storage/framework** est utilisé pour stocker les fichiers et les caches générés par le framework.
- **storage/logs** contient les fichiers journaux de votre application.
- **storage/app/public:** peut être utilisé pour stocker des fichiers générés par l'utilisateur, tels que des avatars de profil, les images des vos publications, qui doivent être accessibles au public.
 - Vous devez créer un lien symbolique(raccourcis) **public/storage** pointant vers ce répertoire.
 - Vous pouvez créer le lien à l'aide de la commande Artisan : **php artisan storage:link**

- ❑ **/bootstrap** : démarrage de l'app.



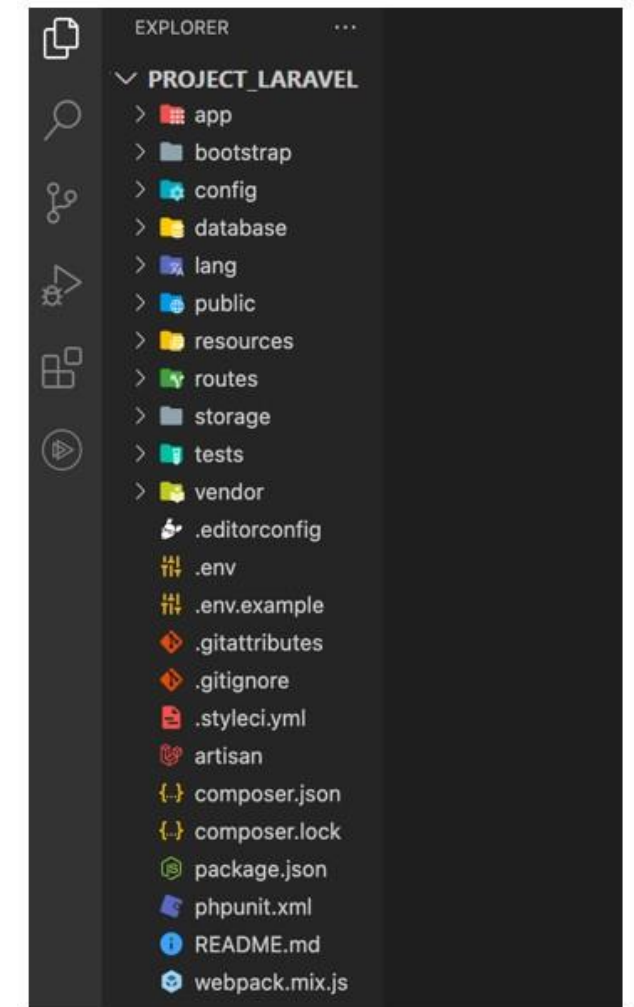
02 - Préparer l'environnement de Laravel

Architecture d'un projet Laravel



Les répertoires racines d'un projet Laravel

- ❑ **/tests:** contient vos tests automatisés. Des exemples de tests unitaires PHPUnit et de tests de fonctionnalités sont fournis prêts à l'emploi.
 - Chaque classe de test doit être suffixée par le mot Test.
 - Vous pouvez exécuter vos tests à l'aide des commandes `php unit` ou `php vendor/bin/phpunit`.
 - Ou, si vous souhaitez une représentation plus détaillée et plus belle de vos résultats de test, vous pouvez exécuter vos tests à l'aide de la commande Artisan `php artisan test`.



02 - Préparer l'environnement de Laravel

Architecture d'un projet Laravel

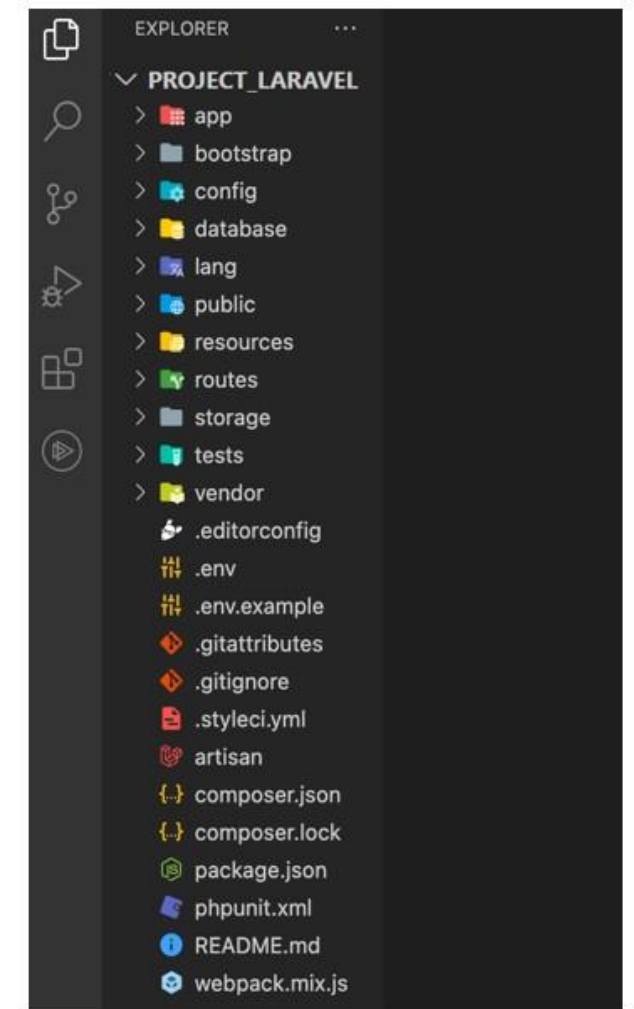


Qu'est ce que le fichier composer.json ?

C'est le fichier qui stocke tous les paquets (dépendances) d'un projet, ainsi que leurs versions, mais peut aussi être utilisé pour définir un projet quand il est publié sur Git, par exemple.

```
{
  "name": "laravel/laravel",
  "type": "project",
  "description": "The skeleton application for the Laravel framework.",
  "keywords": ["laravel", "framework"],
  "license": "MIT",
  "require": {
    "php": "^8.1",
    "guzzlehttp/guzzle": "^7.2",
    "laravel/framework": "^10.10",
    "laravel/sanctum": "^3.3",
    "laravel/tinker": "^2.8"
  },
  "require-dev": {
    "fakerphp/faker": "^1.9.1",
    "laravel/pint": "^1.0",
    "laravel/sail": "^1.18",
    "mockery/mockery": "^1.4.4",
    "nunomaduro/collision": "^7.0",
    "phpunit/phpunit": "^10.1",
    "spatie/laravel-ignition": "^2.0"
  }
}
```

(ici ma version de Laravel est la 10.10 et nécessite une version php qui soit au moins égal à la 8.1)



CHAPITRE 2

Préparer l'environnement de Laravel

1. Installation des outils pour Laravel
2. Créer un nouveau projet Laravel
3. Architecture d'un projet Laravel
4. **Laravel Artisan**
5. Lancer le projet Laravel



02 - Préparer l'environnement de Laravel

Laravel Artisan



Qu'est ce que Laravel Artisan?

Laravel Artisan est une Interface en Ligne de Commande (CLI) qui va vous permettre de gérer votre application en lançant des commandes via le terminal (effacer le cache de l'application, gérer des modèles, des contrôleurs, des routes...)

Laravel Artisan: commandes utiles

1. **Afficher la liste des commandes artisan**

```
php artisan list
```

2. **Pour afficher un écran d'aide, faites précéder le nom de la commande de help**

```
php artisan help migrate
```

3. **Pour créer un modèle**

```
php artisan make:model NomDuModel
```

4. **Vérifier la version de Laravel installée**

```
php artisan -version
```

02 - Préparer l'environnement de Laravel

Laravel Artisan



Laravel Artisan: commandes utiles (suite)

5. lancer le serveur , lancer le projet Laravel

```
php artisan serve
```

6. Le serveur est lancé par défaut sur le port 8000, si on veut le lancer sur un autre port on peut utiliser :

```
php artisan serve --port 3000
```

6. voir toutes les routes de votre application

```
php artisan route:list
```

7. Exécuter toutes les migrations:

```
php artisan migrate
```

8. faire un retour en arrière pour la dernière migration

```
php artisan migrate :rollback
```


CHAPITRE 2

Préparer l'environnement de Laravel

1. Installation des outils pour Laravel
2. Créer un nouveau projet Laravel
3. Architecture d'un projet Laravel
4. Laravel Artisan
5. **Lancer le projet Laravel**



02 - Préparer l'environnement de Laravel

Lancer le projet Laravel



Lancer le projet Laravel

1. Placez le terminal dans le dossier racine de votre projet et taper les commandes: ***php artisan key:generate*** ensuite ***php artisan serve*** :

```
C:\Apache24\htdocs>cd project_laravel  
  
C:\Apache24\htdocs\project_laravel>php artisan key:generate  
  
INFO Application key set successfully.  
  
C:\Apache24\htdocs\project_laravel>php artisan serve  
  
INFO Server running on [http://127.0.0.1:8000].  
  
Press Ctrl+C to stop the server
```

Sous windows 10 , en cas d'erreur exécuter la commande suivante : ***composer install --ignore-platform-reqs***

On peut utiliser ***php artisan key:generate*** dans le cas d'un clone d'un autre projet Laravel pré-crée (afin de régénérer les clés du projet)

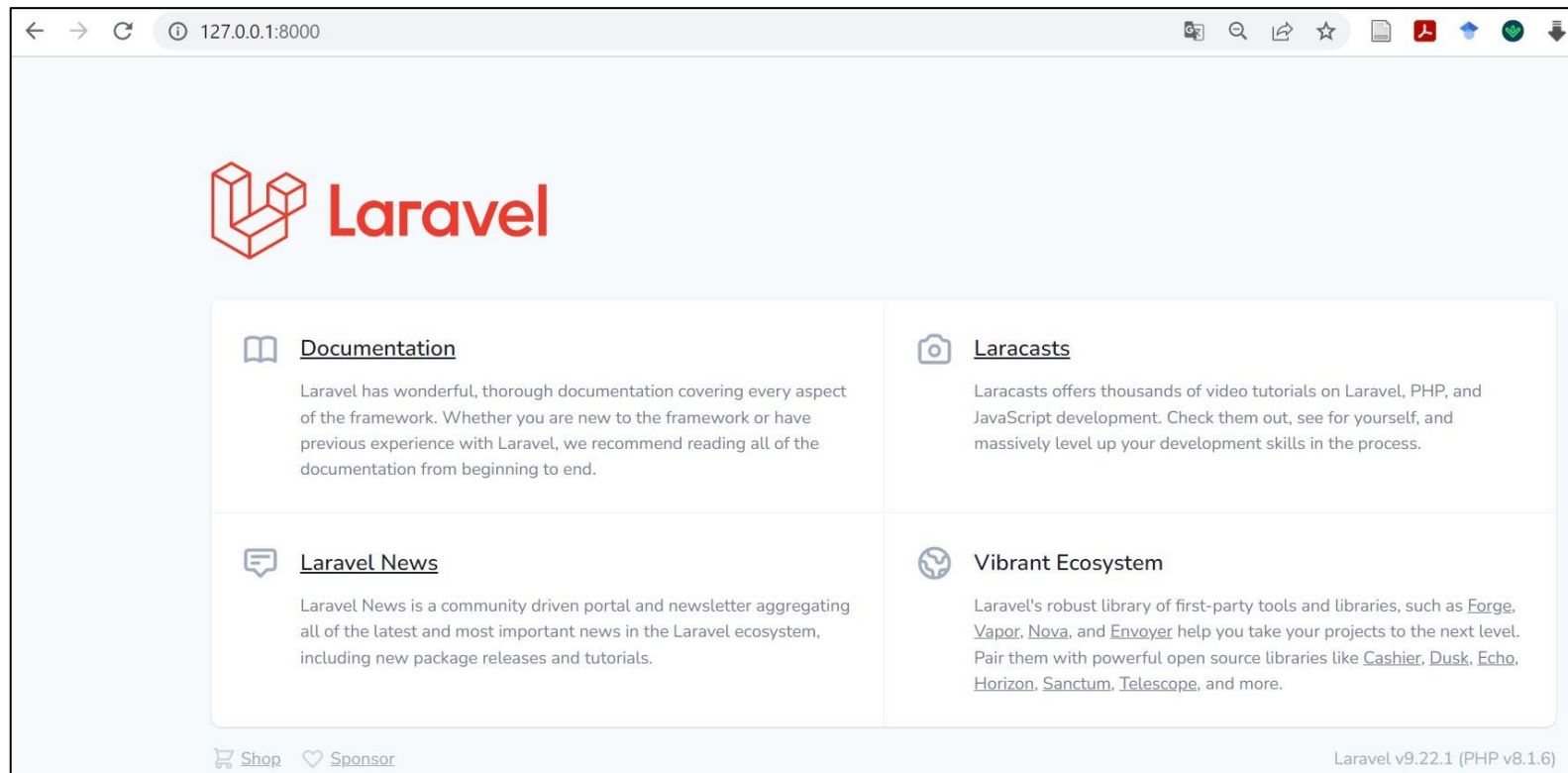
02 - Préparer l'environnement de Laravel

Lancer le projet Laravel



Lancer le projet Laravel

1. Ouvrez le navigateur et allez sur <http://localhost:8000>, et vous obtiendrez le résultat suivant :



02 - Préparer l'environnement de Laravel

Résumé



Résumé

- Pour son installation et sa mise à jour, Laravel utilise le gestionnaire de dépendances Composer.
- La création d'une application Laravel se fait à partir de la console avec une simple ligne de commande.
- Laravel est organisé en plusieurs dossiers.
- Le dossier public est le seul qui doit être accessible par le serveur.

PARTIE 1

Programmer avec Laravel

Dans ce module, vous allez :

- Connaître les fondements du modèle MVC Laravel
- Maîtriser le Framework Laravel



30 heures

CHAPITRE 1

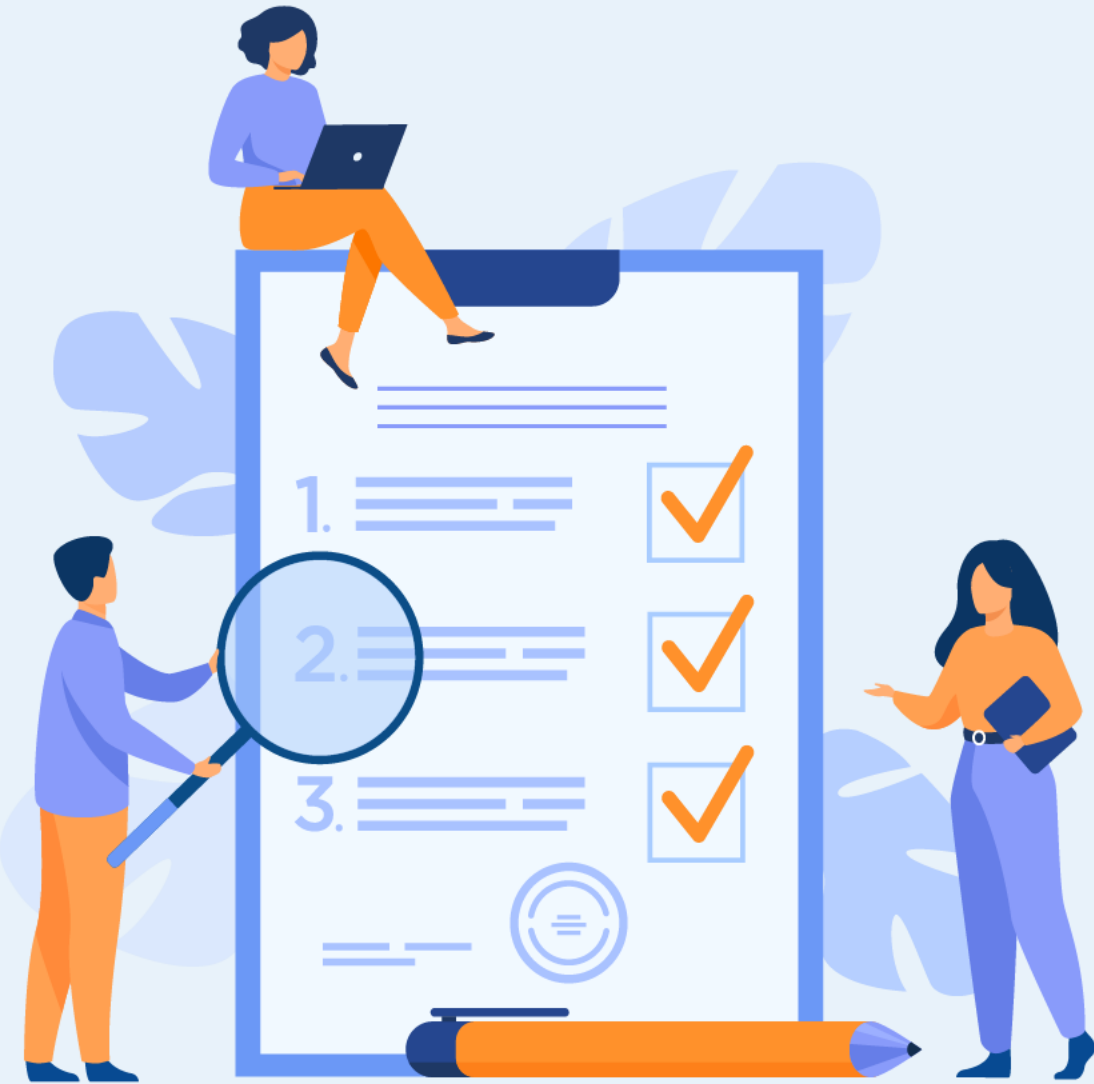
Gestion du routage

Ce que vous allez apprendre dans ce chapitre :

- Routage de base (redirection, affichage, liste de routes)
- Paramètres de routage
- Routes nommées
- Groupe de routage
- Liaisons de modèles de routes
- Routes de repli
- (spoofing) de la méthode du formulaire
- Accès à la route courante
- Cross Origin Resource Sharing (CORS)
- Mise en cache des routes



05 heures



CHAPITRE 1

Gestion du routage

1. Routage de base (redirection, affichage, liste de routes)
2. Paramètres de routage
3. Routes nommées
4. Groupe de routage
5. Liaisons de modèles de routes
6. Routes de repli
7. (spoofing) de la méthode du formulaire
8. Accès à la route courante
9. Cross Origin Resource Sharing (CORS)
10. Mise en cache des routes

01 - Gestion du routage

Routage de base



- Les routes **Laravel** les plus basiques acceptent un URI et une fermeture, fournissant une méthode très simple et expressive de définition des routes et du comportement sans fichiers de configuration de routage compliqués :

```
use Illuminate\Support\Facades\Route;

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


01 - Gestion du routage

Routage de base



Les fichiers de routage par défaut

- Toutes les routes **Laravel** sont définies dans vos fichiers de route, qui se trouvent dans le répertoire **routes**.
- Ces fichiers sont automatiquement chargés par le fichier **App\Providers\RouteServiceProvider**.
- Le fichier **routes/web.php** définit les itinéraires qui sont destinés à votre interface Web.
- Ces routes sont affectées au webgroupe middleware, qui fournit des fonctionnalités telles que l'état de session et la protection CSRF. Les routes dans **routes/api.php** sont sans état et sont affectées au **api** groupe middleware.
- Pour la plupart des applications, vous commencerez par définir des routes dans votre fichier **routes/web.php**. Les itinéraires définis dans **routes/web.php** sont accessibles en saisissant l'URL de l'itinéraire défini dans votre navigateur. Par exemple, vous pouvez accéder à l'itinéraire suivant en naviguant vers <http://example.com/user> dans votre navigateur :

```
use Illuminate\Support\Facades\Route;
```

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

- Les routes définies dans le fichier **routes/api.php** sont imbriquées dans un groupe de routes par le **RouteServiceProvider**. Dans ce groupe, le préfixe URI **/api** est automatiquement appliqué, vous n'avez donc pas besoin de l'appliquer manuellement à chaque route du fichier. Vous pouvez modifier le préfixe et d'autres options de groupe de routage en modifiant votre classe **RouteServiceProvider**.

01 - Gestion du routage

Routage de base



Méthodes de routeur disponibles

- Le routeur vous permet d'enregistrer des routes qui répondent à n'importe quel verbe HTTP :

```
Route::get($uri, $callback);  
Route::post($uri, $callback);  
Route::put($uri, $callback);  
Route::patch($uri, $callback);  
Route::delete($uri, $callback);  
Route::options($uri, $callback);
```

- Parfois, vous devrez peut-être enregistrer une route qui répond à plusieurs verbes HTTP. Vous pouvez le faire en utilisant la méthode **match**. Ou, vous pouvez même enregistrer une route qui répond à tous les verbes HTTP en utilisant la méthode **any** :

```
Route::match(['get', 'post'], '/', function () {  
    //  
});  
  
Route::any('/', function () {  
    //  
});
```

01 - Gestion du routage

Routage de base



Injection de dépendance

- Vous pouvez indiquer toutes les dépendances requises par votre itinéraire dans la signature de rappel de votre itinéraire. Les dépendances déclarées seront automatiquement résolues et injectées dans le rappel par le conteneur de service **Laravel** .
- Par exemple, vous pouvez donner un indice de type à la classe **Illuminate\Http\Request** pour que la requête HTTP actuelle soit automatiquement injectée dans votre rappel de route :

```
use Illuminate\Http\Request;

Route::get('/users', function (Request $request) {
// ...
});
```

Figure 1 : les navigateurs les plus utilisés

01 - Gestion du routage

Routage de base



Redirection

- Si vous définissez une route qui redirige vers un autre **URI**, vous pouvez utiliser la méthode **Route::redirect**. Cette méthode fournit un raccourci pratique pour que vous n'ayez pas à définir une route ou un contrôleur complet pour effectuer une simple redirection :

```
Route::redirect('/ici', '/là-bas');
```

- Par défaut, **Route::redirect** renvoie un code d'état **302**. Vous pouvez personnaliser le code d'état à l'aide du troisième paramètre facultatif :

```
Route::redirect('/ici', '/là-bas', 301);
```

- Ou, vous pouvez utiliser la méthode **Route::permanentRedirect** pour renvoyer un code d'état **301**:

```
Route::permanentRedirect ('/ici', '/là-bas');
```

- Une redirection 301 est utilisée lorsque vous souhaitez rediriger de manière **permanente** une URL vers une autre URL. Cette redirection indique aux utilisateurs et aux moteurs de recherche qu'une page a été déplacée de façon permanente. Elle est le plus souvent utilisée lorsque des pages sont déplacées de façon permanente vers une autre URL ou ont été purement et simplement supprimées.

À l'inverse, une redirection 302 est une redirection temporaire. Les redirections 302 sont utilisées lorsque le webmaster a l'intention de revenir à l'URL d'origine à l'avenir.

01 - Gestion du routage

Routage de base



Afficher les routes

- Si votre itinéraire ne doit renvoyer qu'une vue , vous pouvez utiliser la méthode **Route::view**.
- Comme la méthode **redirect**, cette méthode fournit un raccourci simple pour que vous n'ayez pas à définir une route ou un contrôleur complet. La méthode **view** accepte un URI comme premier argument et un nom de vue comme second argument. De plus, vous pouvez fournir un tableau de données à transmettre à la vue en tant que troisième argument facultatif :

```
Route::view('/welcome', 'welcome');
```

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

Source : <https://www.wddonline.com/2016/09/27/history-world-wide-web/>

01 - Gestion du routage

Routage de base



La liste des routes

- La commande **Artisan route:list** peut facilement fournir une vue d'ensemble de toutes les routes définies par votre application :

```
php artisan route:list
```

- Par défaut, le middleware de route affecté à chaque route ne sera pas affiché dans la sortie **route:list** ; cependant, vous pouvez demander à **Laravel** d'afficher le middleware de route en ajoutant l'option -v à la commande :

```
php artisan route:list -v
```

- Vous pouvez également demander à **Laravel** de n'afficher que les routes commençant par un **URI** donné :

```
php artisan route:list --path=api
```

- De plus, vous pouvez demander à **Laravel** de masquer toutes les routes définies par des packages tiers en fournissant l'option **--except-vendor** lors de l'exécution de la commande **route:list** :

```
php artisan route:list --except-vendor
```

01 - Gestion du routage

Routage de base



La liste des routes

- De même, vous pouvez également demander à **Laravel** de n'afficher que les routes définies par des packages tiers en fournissant l'option **--only-vendor** lors de l'exécution de la commande **route:list** :

```
php artisan route:list
```

CHAPITRE 1

Gestion du routage

1. Routage de base (redirection, affichage, liste de routes)
2. Paramètres de routage
3. Routes nommées
4. Groupe de routage
5. Liaisons de modèles de routes
6. Routes de repli
7. (spoofing) de la méthode du formulaire
8. Accès à la route courante
9. Cross Origin Resource Sharing (CORS)
10. Mise en cache des routes



01 - Gestion du routage

Paramètres de routage



Paramètres requis

- Parfois, vous devrez capturer des segments de l'URI dans votre route. Par exemple, vous devrez peut-être capturer l'ID d'un utilisateur à partir de l'URL. Vous pouvez le faire en définissant les paramètres de route :

```
Route::get('/user/{id}', function ($id) {  
    return 'User '.$id;  
});
```

- Vous pouvez définir autant de paramètres de route que requis par votre route :

```
Route::get('/posts/{post}/comments/{comment}', function ($postId, $commentId) {  
    //  
});
```

- Les paramètres de route sont toujours entourés d'accolades {} et doivent être composés de caractères alphabétiques. Les traits de soulignement (_) sont également acceptables dans les noms de paramètre de route. Les paramètres de route sont injectés dans les rappels/contrôleurs de route en fonction de leur ordre - les noms des arguments de rappel/contrôleur de route n'ont pas d'importance.

Source : <https://www.wddonline.com/2016/09/27/history-world-wide-web/>

01 - Gestion du routage

Paramètres de routage



Paramètres et injection de dépendance

- Si votre route a des dépendances que vous aimeriez que le conteneur de service **Laravel** injecte automatiquement dans le rappel de votre route, vous devez lister vos paramètres de route après vos dépendances.

```
use Illuminate\Http\Request;
```

```
Route::get('/user/{id}', function (Request $request, $id) {  
    return 'User '.$id;  
});
```

01 - Gestion du routage

Paramètres de routage



Paramètres facultatifs

- Parfois, vous devrez peut-être spécifier un paramètre de route qui n'est pas toujours présent dans l'URI. Vous pouvez le faire en plaçant une ? marque après le nom du paramètre. Assurez-vous de donner une valeur par défaut à la variable correspondante de la route :

```
Route::get('/user/{name?}', function ($name = null) {  
    return $name;  
});
```

```
Route::get('/user/{name?}', function ($name = Ahmed') {  
    return $name;  
});
```

CHAPITRE 1

Gestion du routage

1. Routage de base (redirection, affichage, liste de routes)
2. Paramètres de routage
3. **Routes nommées**
4. Groupe de routage
5. Liaisons de modèles de routes
6. Routes de repli
7. (spoofing) de la méthode du formulaire
8. Accès à la route courante
9. Cross Origin Resource Sharing (CORS)
10. Mise en cache des routes



01 - Gestion du routage

Routes nommées



- Les routes nommées permettent la génération pratique d'URL ou de redirections pour des routes spécifiques. Vous pouvez spécifier un nom pour une route en enchaînant la méthode **name** sur la définition de route :

```
Route::get('/user/profile', function () {  
    //  
})->name('profile');
```

- Vous pouvez également spécifier des noms de route pour les actions du contrôleur :

```
Route::get(  
    '/user/profile',  
    [UserProfileController::class, 'show']  
)->name('profile');
```

01 - Gestion du routage

Routes nommées



Génération d'URL vers des routes nommées

- Une fois que vous avez attribué un nom à une route donnée, vous pouvez utiliser le nom de la route lors de la génération d'URL ou de redirections via les fonctions de **Laravel** `route` et d'assistance **redirect** :

```
// Generating URLs...
$url = route('profile');

// Generating Redirects...
return redirect()->route('profile');
```

- Si la route nommée définit des paramètres, vous pouvez passer les paramètres comme deuxième argument à la fonction **route**. Les paramètres donnés seront automatiquement insérés dans l'**URL** générée dans leurs positions correctes :

```
Route::get('/user/{id}/profile', function ($id) {
    //
})->name('profile');

$url = route('profile', ['id' => 1]);
```

01 - Gestion du routage

Routes nommées



- Si vous transmettez des paramètres supplémentaires dans le tableau, ces paires **clé/valeur** seront automatiquement ajoutées à la chaîne de requête de l'URL générée :

```
Route::get('/user/{id}/profile', function ($id) {  
    //  
    }->name('profile');  
  
    $url = route('profile', ['id' => 1, 'photos' => 'yes']);  
  
    // /user/1/profile?photos=yes
```