Pecee/simple-router

[Installation 2](#_Toc131579619)

[Composer 2](#_Toc131579620)

[Simple-router 2](#_Toc131579621)

[Autoload 2](#_Toc131579622)

[VirtualHost 2](#_Toc131579623)

[Création 2](#_Toc131579624)

[apache2.conf 3](#_Toc131579625)

[Hosts 3](#_Toc131579626)

[Modifications 3](#_Toc131579627)

[autoload-psr4.php 3](#_Toc131579628)

[autoload\_static.php 4](#_Toc131579629)

[.htaccess 4](#_Toc131579630)

[composer.json 5](#_Toc131579631)

[index.php 5](#_Toc131579632)

[wep.php 5](#_Toc131579633)

Installation

Composer

Pour commencer, tapez la commande suivante : « curl -sS https://getcomposer.org/installer -o composer-setup.php ».

Ensuite, tapez la commande suivante afin d’installer Composer de manière globale : « sudo php composer-setup.php --install-dir=/usr/local/bin --filename=composer »

Vous pouvez ensuite relancer votre Ubuntu et tapez « composer --version » pour vérifier que composer est bien installé.

Simple-router

Ensuite Il faut suivre la documentation qui explique de l’installation à l’utilisation de simple-router ou continue cette documentation qui explique chaque étape de l’insallation :

<https://packagist.org/packages/pecee/simple-router>

En cas de problème avec l’installation de simple-router il est possible de récupérer les fichiers générer directement dans ce GitHub :

<https://github.com/skipperbent/simple-php-router>

ou tapper la commande : « composer require pecee/simple-router »

Autoload

Autoload permet de charger automatiquement les classes sans avoir à les inclure via « require\_once » ou « include\_once » .

Une fois simple-router installer il est possible d’intégrer autoload en suivant ces 2 étapes :

« sudo apt-get update -y »

« sudo apt-get install -y php-autoload »

Pour mettre à jour il suffit de tapper cette commande :

« composer dump-autoload »

VirtualHost

## Création

Pour créer un virtual host il faut aller dans le serveur et taper la commande :

« sudo vim /etc/apache2/sites-available/[nom\_du\_domaine].conf »

Dans ce fichier il faut ajouter ce code et changer les éléments entre crochet en fonction du projet.

<VirtualHost \*:[Port]>

DocumentRoot ["Chemin jusqu’à l’index"]

ServerName [www.example.com]

</VirtualHost>

Une fois terminé il faut faire un lien symbolique vers « sites-enabled». Pour ce faire il faut aller dans le ce dossier puis tapper la commande :

« sudo a2ensite [nom\_du\_domaine].conf »

## apache2.conf

Ensuite il faut aller dans le fichier apache2.conf en tapant la commande suivante :

« sudo vim /etc/apache2/apache2.conf »

Dans ce fichier il faut ajouter une balise Directory comme ceci :

<Directory ["Même chemin que DocumentRoot"]>

AllowOverride all

Require all granted

</Directory>

Toujours dans ce fichier il faut ajouter (pas obligatoire) :

ServeurName [exemple.com]

## Hosts

Après il faut ouvrir le fichier hosts en administrateur qui se trouve dans le chemin suivant :

« C:\Windows\System32\drivers\etc »

Une fois ouvert ajouter à ce fichier :

127.0.0.1 [exemple.com]

Pour écouter le port, il faut aller dans « /etc/apache2/ports.conf » et ajouter une ligne avec votre port :

Listen [port]

Commande pour voir les ports qui sont ouverts :

« sudo apache2ctl -S »

Modifications

autoload-psr4.php

Voici le code de base :

return array(

    'Pecee\\' => array($vendorDir . '/pecee/simple-router/src/Pecee'),

);

Voici a quoi ressemble le code après modification :

return array(

    'projet\_exemple\\' => array($baseDir . '/src'),

    'Pecee\\' => array($vendorDir . '/pecee/simple-router/src/Pecee'),

);

La modification permet ajouter un deuxième préfixe pour créer un nouveau namespace spécifique au projet.

autoload\_static.php

Remplacez :

 public static $prefixLengthsPsr4 = array (

        'P' =>

        array (

            'Pecee\\' => 6,

        ),

    );

    public static $prefixDirsPsr4 = array (

        'Pecee\\' =>

        array (

            0 => \_\_DIR\_\_ . '/..' . '/pecee/simple-router/src/Pecee',

        ),

    );

Par :

public static $prefixLengthsPsr4 = array (

        '[**premiere lettre du projet**]' =>

        array (

            'projet\_exemple\\' => 15,

        ),

        'P' =>

        array (

            'Pecee\\' => 6,

        ),

    );

    public static $prefixDirsPsr4 = array (

        'projet\_exemple\\' =>

        array (

            0 => \_\_DIR\_\_ . '/../..' . '/src',

        ),

        'Pecee\\' =>

        array (

            0 => \_\_DIR\_\_ . '/..' . '/pecee/simple-router/src/Pecee',

        ),

    );

Cette modification de l'autoload\_static.php sert à ajouter un nouveau préfixe de chargement de classes pour le projet, qui permettra de charger les classes du projet à partir d'un dossier spécifique.

.htaccess

Si la documentation d’installation n’a pas été suivie, il ne faut pas oublier de créer un fichier .htaccess dans le dossier public qui permet de rediriger vers la page index constamment. Pour ce faire, il faut ajouter ces lignes :

RewriteEngine on

RewriteCond %{SCRIPT\_FILENAME} !-f

RewriteCond %{SCRIPT\_FILENAME} !-d

RewriteCond %{SCRIPT\_FILENAME} !-l

RewriteRule ^(.\*)$ index.php/$1

composer.json

Il faut ajouter :

"autoload": {

        "psr-4": {

            "projet\_exemple\\": "src/"

        }

    },

index.php

Le fichier index devrait ressembler à ça :

<?php

require '../vendor/autoload.php';

require '../routes/routes.php';

use Pecee\SimpleRouter\SimpleRouter as Router;

// Enregistrement du namespace par défaut des controllers

Router::setDefaultNamespace('\projet\_exemple\Controlleurs');

// Lancement du router

Router::start();

routes.php

Le fichier routes.php doit être mis dans un dossier routes par exemple à la racine du projet. Il contient tout le routage et les appellent aux contrôleurs.

Voici un exemple d’utilisation :

Dans cette ligne on utilise la classe « AccueilControlleur » pour pouvoir appeler les fonctions qui sont dedans.

use projet\_exemple\Controlleurs\AccueilControlleur;

Ensuite il est possible de faire le routage comme dans cet exemple :

Router::get('/', [AccueilControlleur::class, 'accueil']);

Cette ligne permet de dire que quand on arrive sur le site ou que la page est « / » on appelle le contrôleur accueil puis on appelle la fonction accueil qui est dans le contrôleur.