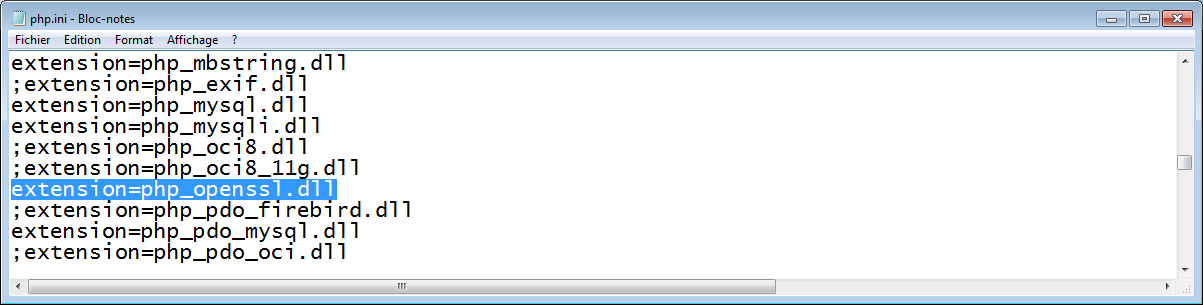
# Intégration de COMPOSER et de SILEX à votre application

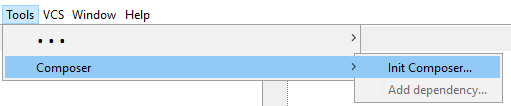
## Installation de COMPOSER et configuration de PHPStorm pour COMPOSER

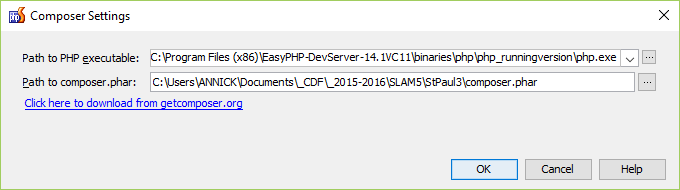
Sur Windows, les commandes php.exe et git.exe doivent être disponibles. Vérifier que la variable d’environnement PATH contient bien les chemins de ces exécutables.

Modifier le fichier php.ini pour prendre en compte l’extension openssl : oter le commentaire devant la ligne *extension=php\_openssl.dll*



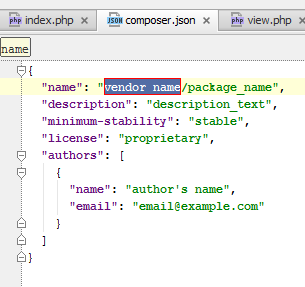
Sur PHPStorm :





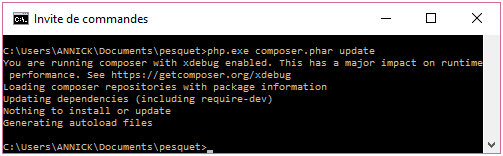
Suivre le lien *Click here to download…* pour installer composer.phar **dans le répertoire de l’application*. Faire attention aux chemins proposés.***

Init Composer crée le fichier composer.json comme suit. Il s’agit d’un squelette du fichier définitif.



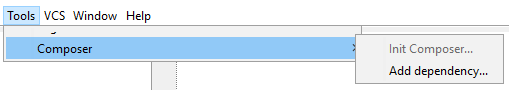
Vous pouvez modifier directement le fichier composer.json pour y répertorier tous les modules externes nécessaires à votre application.

La commande php.exe composer.phar update permet de vérifier les dépendances entre les modules ajoutés et de mettre à jour votre application.

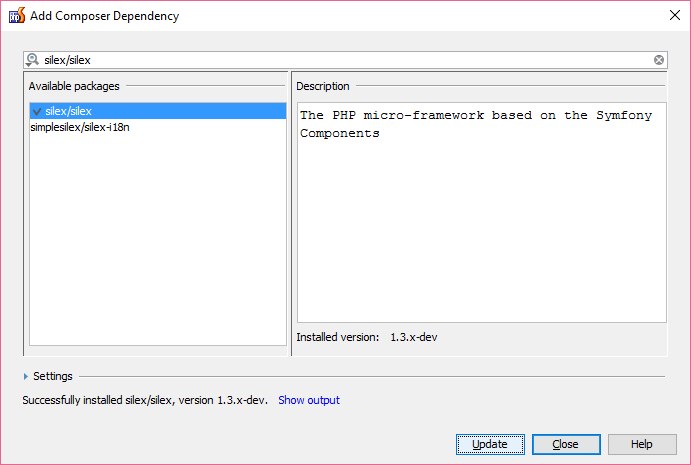


## Ajouter un module externe (SILEX par exemple) avec COMPOSER

PHPSTorm peut aussi prendre en charge ce travail ou ajouter de nouveaux modules externes nécessaires à votre application en faisant comme suit :



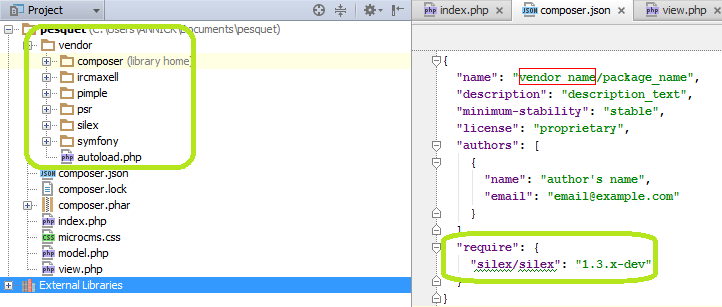
Add dependency permet d’ajouter SILEX à votre « boite à outils »



Show output vous permet de voir les packages ajoutés par l’installation.

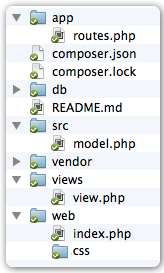
Le fichier composer.json contient maintenant SILEX.

Le projet contient le dossier VENDOR qui regroupe le code tiers utilisé par votre application.



# Modification de l’application de façon à utiliser SILEX

## Mise à jour de l’arborescence

Créez les sous-répertoires suivants :

* app, qui contiendra la configuration de l’application Silex ;
* src, qui contiendra les fichiers source PHP ;
* views, qui contiendra les vues de l’application ;
* web, qui contiendra les fichiers accessibles aux clients Web.

Déplacer vos fichiers de façon à obtenir l’arborescence ci-contre :

## Utiliser SILEX dans votre application

Editez index.php et remplacez son contenu par le code source ci-dessous.

**<?php  
  
require\_once *\_\_DIR\_\_***.**'/../vendor/autoload.php'**;  
  
$app = **new** Silex\Application();  
  
**require *\_\_DIR\_\_***.**'/../app/routes.php'**;  
  
$app->run();

Ce fichier constitue le **contrôleur frontal** de notre application Web. Il centralise la gestion des requêtes HTTP entrantes. Dans ce fichier, on instancie l’objet Silex principal $app puis on inclut la définition des routes de l’application (fichier routes.php).

Toujours dans web, créez un nouveau fichier texte nommé .htaccess contenant le texte ci-dessous. Ce fichier permet de rediriger toutes les requêtes entrantes vers index.php.

*# Redirect incoming URLs to index.php*<**IfModule mod\_rewrite.c**>  
 **Options -MultiViews  
  
 RewriteEngine On  
 RewriteCond %{REQUEST\_FILENAME} !-f  
 RewriteRule ^ index.php [QSA,L]**</**IfModule**>

Dans le sous-répertoire app, créez maintenant le fichier routes.php avec le contenu ci-dessous.

**<?php**$app->get(**'/ping'**, **function**() **use** ($app) {  
 **return 'ping'**;  
});  
  
$app->get(**'/hello'**, **function**() **use** ($app) {  
 **return 'Hello world'**;  
});  
  
$app->get(**'/hello/{name}'**, **function** ($name) **use** ($app) {  
 **return 'Hello '** . $app->escape($name);  
});

Silex permet de définir des **routes**, c’est-à-dire des points d’entrée dans l’application. A chaque route est associée une réponse construite par notre code.

La méthode get de $app indique quel type de requête est attendu (GET dans ce cas, il est aussi possible d’utiliser ->post() pour les requêtes POST, ->delete() pour les requêtes DELETE, etc.). Le premier argument représente l’url à capturer, ici “/ping”. A savoir qu’il est possible de passer des arguments, en utilisant la syntaxe {var} (ici **/hello/{name}**). La variable sera alors récupérée comme paramètre de la fonction inline qui vient en second (ici dans **$name**).

Vous pouvez tester les 3 url :

* *monProjet/*web/index.php/ping
* *monProjet/*web/index.php/hello
* *monProjet/*web/index.php/hello/Lulu

NB : il est possible de reconfigurer Apache pour avoir des URL plus simples et plus lisibles.

Réécrire routes.php avec le contenu nécessaire à l’application.

**<?php***//Retourne tous les séjours*$app->get(**'/'**, **function**() {  
  
 **require '../src/model.php'**;  
 $sejours = getAllSejours();  
  
 *ob\_start*(); *// start buffering HTML output* **require '../views/view.php'**;  
 $view = *ob\_get\_clean*(); *// assign HTML output to $view* **return** $view;  
});

La route ci-dessus correspond à l’URL racine de l’application (/). La [fonction anonyme](http://php.net/manual/fr/functions.anonymous.php) associée à cette route utilise la fonction getAllSejours du modèle pour récupérer la liste des séjours.

Chaque route doit renvoyer explicitement une réponse. Les fonctions PHP [ob\_start](http://php.net/manual/fr/function.ob-start.php) et [ob\_get\_clean](http://php.net/manual/fr/function.ob-get-clean.php) permettent de récupérer le résultat de l’appel à require (autrement dit la vue générée) dans une variable nommée $view. Cette variable est renvoyée à la fin de la fonction.

Vous pouvez tester votre application.