

# **Installer un bundle avec Composer**

# Table des matières

<b>I. Introduction</b>	<b>3</b>
<b>II. Composer</b>	<b>3</b>
A. Introduction.....	3
B. Qu'est-ce qu'un bundle ? .....	5
C. Exercice : Quiz.....	6
<b>III. Installation, mise à jour et suppression</b>	<b>7</b>
A. Installation de Composer.....	7
B. Installation de Git .....	8
C. Installer un bundle .....	11
D. Symfony Flex .....	12
E. La gestion de versions .....	13
F. Supprimer un bundle .....	13
G. Exercice : Quiz.....	13
<b>IV. L'essentiel</b>	<b>14</b>
<b>V. Auto évaluation</b>	<b>14</b>
A. Exercice .....	14
B. Test.....	15
<b>Solutions des exercices</b>	<b>15</b>

# I. Introduction

**Durée :** 1 h

**Environnement de travail :** PC connecté à internet et IDE VsCode

## Contexte

Ce cours traite de l'installation d'un bundle dans un projet Symfony avec le gestionnaire de dépendance Composer. Il permettra de comprendre ce qu'est un bundle, l'utilité de Git et Composer et comment les utiliser.

Dans le monde du développement web, Composer est omniprésent, il est donc crucial de l'appréhender pour le comprendre et ainsi le maîtriser le plus rapidement possible, ce qui permettra de produire un code propre et compréhensible pour tous les intervenants sur un projet.

On utilise fréquemment Git en tant que développeur web afin de pouvoir collaborer sur un projet notamment hébergé sur Github, Gitlab ou un autre *provider*. Il est facile de constater la puissance de Git lors d'un projet collaboratif hébergé sur Github, puisqu'il permet à chaque développeur participant à un projet de travailler indépendamment sur sa branche, et ainsi éviter l'écrasement des modifications apportées par un autre développeur, mais également de voir les modifications apportées au code.

De plus, il est également possible de revenir à la version précédente et de minimiser la perte de travail. Il s'utilise également comme système de gestion de versions. Dans le cadre d'un projet évolutif, il sera possible de le faire évoluer sans écraser la version précédente et ainsi permettre aux utilisateurs d'utiliser les versions précédentes si nécessaire. Les utilisateurs ont la possibilité de mettre des issues pour déclarer un bug ou même un besoin.

## II. Composer

### A. Introduction

#### Définition

Composer est ce qu'on appelle un gestionnaire de dépendances parce qu'il permet d'agir sur les dépendances en PHP au sein d'un projet. Les dépendances dans un projet sont toutes les bibliothèques dont le projet dépend pour fonctionner.

#### Exemple

Le projet utilise la bibliothèque CKEditor pour intégrer un éditeur de texte, il dépend donc de CKEditor. Ainsi, dans notre cas, CKEditor est une dépendance dans le projet.

Si par mégarde, un bundle n'a pas été installé (ici CKEditor), que ce soit sur un serveur de développement ou un serveur de production, et que du code dépend de ce bundle, une erreur sera générée et celui-ci ne fonctionnera pas.

De plus, quand bien même Composer est un véritable facilitateur, il possède aussi quelques inconvénients :

- Composer nécessite une connexion internet pour télécharger les dépendances nécessaires, ce qui peut-être un problème si vous travaillez en mode déconnecté ou si votre connexion est instable.
- Composer peut parfois être un peu lent pour télécharger les dépendances, en particulier si vous installez un grand nombre de paquets ou si vous utilisez des serveurs de dépôt lent.
- Composer peut être un peu difficile à apprendre pour les développeurs qui n'ont pas l'habitude de travailler avec des gestionnaires de dépendances. Il y a une courbe d'apprentissage pour comprendre comment utiliser Composer et configurer correctement votre projet.
- Composer ne prend pas en charge certaines anciennes versions de PHP, ce qui peut être un problème si vous devez utiliser une version obsolète pour une raison quelconque.

- Composer ne peut pas être utilisé pour gérer les dépendances de projets qui ne sont pas basés sur PHP, bien qu'il existe des alternatives similaires pour d'autres langages de programmation.
- Les dépendances cessent d'être mises à jour automatiquement pour la majorité et requièrent d'être mises à jour manuellement (ce qui est très important pour corriger les bogues ou même les potentielles failles).
- En cas de changement de version de Symfony, il faudra également mettre à jour les dépendances en fonction de la version de Symfony installée.
- Les dépendances doivent être mises à jour sans précipitation, parce qu'il est possible que des erreurs se produisent, et notamment en utilisant un bundle qui n'est pas à jour.
- Il se peut que certaines dépendances soient dépendantes d'autres dépendances et de ce fait celles-ci doivent également être mises à jour ou installées. Si cela n'est pas fait, la dépendance fille ne fonctionnera pas.

#### Rappel

Pour que Composer puisse trouver une dépendance (un composant ou encore un bundle), il va chercher sa source dans une banque de répertoires Github qui centralise tous les composants PHP pour des frameworks comme Symfony ou Laravel. Il s'agit de Packagist<sup>1</sup>.

La centralisation est cruciale, puisqu'elle permet de tout retrouver, ce qui est important dans le cas de Composer pour qu'il puisse faire ce à quoi il est destiné, faciliter la vie.

Par ailleurs, en consultant une page consacrée à un composant sur le site Packagist<sup>2</sup>, de nombreuses informations sont à disposition. Il faudra s'intéresser davantage à la source et aux *requires*, puisqu'il s'agit des éléments qui seront les plus pertinents.

Il est impératif de bien regarder les *requires* pour éviter le plus possible une incompatibilité et de générer une erreur.

Composer n'est d'ailleurs pas le seul gestionnaire de dépendance dans le monde de l'informatique, il existe d'autres gestionnaires de dépendances dans d'autres langages de programmation, comme Bundler pour Ruby ou NPM pour JavaScript.

APT (Advanced Packaging Tool) est un gestionnaire de paquets pour les systèmes d'exploitation basés sur Debian, comme Ubuntu. Il permet de gérer les logiciels installés sur votre système, de les mettre à jour et de les désinstaller, ainsi que de gérer les dépendances entre ces logiciels. APT est similaire à d'autres gestionnaires de paquets tels que Yum pour les systèmes basés sur Red Hat ou Pacman pour Arch Linux.

#### Définition

Les *requires* d'un projet Symfony sont les dépendances externes nécessaires à votre projet. Elles sont généralement indiquées dans le fichier `composer.json` de votre projet. Ces dépendances peuvent être des bibliothèques PHP, des frameworks ou encore des outils de développement. Lorsque vous installez un projet Symfony, le gestionnaire de dépendances Composer télécharge et installe automatiquement ces dépendances pour vous, afin que votre projet puisse fonctionner correctement. Les *requires* peuvent être spécifiés de manière explicite dans le fichier `composer.json`, ou bien être implicite, lorsqu'une dépendance est requise par une autre dépendance que vous avez explicitement ajoutée à votre projet.

Composer est un outil qui utilise les informations contenues dans le fichier `composer.json` pour gérer les dépendances de votre projet Symfony, appelées *requires*.

<sup>1</sup> <https://packagist.org/>

<sup>2</sup> <https://packagist.org/>

## B. Qu'est-ce qu'un bundle ?

Depuis la version 4.0 de Symfony, il n'est plus conseillé d'organiser une application web sous Symfony en organisant le code avec les bundles, parce que le but d'un bundle est de partager le code d'une fonctionnalité entre plusieurs pages ou même applications.

Cela permet de garder un code propre ainsi qu'une arborescence claire et qui constitue en une bonne pratique du codage. Il existe de nombreuses bonnes pratiques mais pour l'instant, mieux vaut se concentrer sur la propreté et la clarté du code. La norme, quant à elle, sera traitée par la suite dans le cours.

Un bundle ressemble à une extension de navigateur web ou même à un CMS, il est constitué de la même manière sauf en ce qui concerne le contenu.

Ce sont des packages PHP qui étendent les fonctionnalités de Symfony en ajoutant de nouveaux composants, de nouvelles configurations et de nouvelles fonctionnalités à l'application. Ils sont généralement développés et maintenus par la communauté Symfony, mais on peut aussi créer nos propres bundles pour l'application. Les bundles peuvent être utilisés indépendamment les uns des autres ou être combinés pour ajouter de nouvelles fonctionnalités à votre application.

Un bundle peut également inclure d'autres éléments tels que des contrôleurs, des services, des événements, etc. En général, un bundle est un package tout-en-un qui regroupe tous les éléments nécessaires à une fonctionnalité spécifique de votre application. Il ne faut cependant pas le confondre avec un composant (tel un composant Symfony) qui correspond plus à une librairie, généralement une succession simple de classes que vous pouvez utiliser indépendamment de chaque projet, sans logique métier, ni contrôleurs.

Les bundles doivent être activés ou désactivés dans le fichier `config/bundles.php` mais il n'est pas nécessaire de toucher à ce fichier si l'application contient Symfony Flex, ce qui est logiquement le cas, et qui lui permet de s'autogérer lorsque des bundles sont installés ou supprimés.

En revanche, lorsque l'on crée son propre bundle, il faudra l'intégrer manuellement dans le fichier `bundles.php` ainsi que dans le fichier `composer.json` en respectant la norme PSR-4.

### Remarque

Les normes PSR ressemblent à ce que sont les règles grammaticales (ponctuation, syntaxe, etc.) en littérature mais pour le développement web. Ainsi, pour faciliter la collaboration entre développeurs, il est primordial de respecter les PSR (PHP Standard Recommendation) qui sont un ensemble de normes pour le langage PHP. Ces normes améliorent la compréhension et permettent également aux différents composants de communiquer de la même manière. Ces normes viennent du PHP Framework Interop Group (PHP-FIG) qui est le groupe de travail qui est à l'origine des PSR. Il est important de garder à l'esprit que les normes PSR ne sont que des recommandations et non des obligations. Il en existe un total de 20.

### La composition d'un bundle

Les classes PHP présentes dans un bundle peuvent étendre ou surcharger les fonctionnalités de Symfony, ou encore ajouter de nouvelles fonctionnalités à l'application. Par exemple, elles peuvent inclure des classes qui étendent les contrôleurs de Symfony pour renforcer la gestion des routes, tandis que d'autres peuvent étendre les modèles de données pour offrir de nouvelles possibilités de persistance de données.

Quant aux fichiers de configuration inclus dans un bundle, ceux-ci définissent comment le bundle doit être utilisé et configuré dans l'application. Ils peuvent inclure des paramètres de configuration, des routes, des services, des événements, etc. qui sont nécessaires pour le fonctionnement du bundle ou de l'application.

Concernant les vues intégrées au bundle, ce sont des fichiers de template Twig qui permettent de visualiser le contenu de l'application. Elles comprennent du code HTML, PHP et Twig qui est interprété et transformé en HTML lors de l'affichage de l'application, afin de présenter de manière cohérente les données.

De plus, les ressources comprises dans un bundle sont des fichiers tels que des images, des feuilles de style ou des scripts JavaScript qui sont nécessaires pour le fonctionnement du bundle ou de l'application. Ces fichiers peuvent être inclus dans l'application via des liens statiques ou des appels de ressources dans les vues ou les fichiers de configuration, de sorte à pouvoir les utiliser facilement.

Enfin, les commandes console jointes au bundle sont des classes qui peuvent être exécutées via le terminal pour accomplir des tâches spécifiques. Elles peuvent automatiser des processus tels que la création d'entités, la mise à jour de la base de données, la génération de code, etc. pour améliorer l'efficacité.

### C. Exercice : Quiz

[solution n°1 p.17]

#### Question 1

Composer est un :

- ☐ Gestionnaire de dépendance
- ☐ Un framework
- ☐ Un bundle

#### Question 2

Composer permet d'agir sur les dépendances en :

- ☐ Python
- ☐ JavaScript
- ☐ PHP

#### Question 3

Les dépendances se mettent à jour automatiquement.

- ☐ Vrai
- ☐ Faux

#### Question 4

Les dépendances peuvent dépendre d'autres dépendances.

- ☐ Vrai
- ☐ Faux

#### Question 5

Quelle est la source qui centralise toutes les dépendances ?

- ☐ packagist.net
- ☐ packagist.org
- ☐ packagist.fr

### III. Installation, mise à jour et suppression

#### A. Installation de Composer

##### Méthode

Pour installer Composer sur votre ordinateur, vous pouvez suivre les étapes suivantes :

- Téléchargez l'installateur de Composer à partir du site web de Composer : [getcomposer.org](https://getcomposer.org)<sup>1</sup>.
- Exécutez l'installateur pour Windows et Mac et suivez les instructions à l'écran pour installer Composer.

Sous Linux, il faut taper les commandes qui sont écrites dans le bloc Command-line installation de la page Download du site [getcomposer.org](https://getcomposer.org)<sup>2</sup> sur son terminal.

À savoir que vous pouvez télécharger le binaire directement depuis Github ou via la commande CURL, qu'il faudra s'assurer d'installer au préalable.

Une fois l'installation terminée, vous pouvez vérifier que Composer est correctement installé en ouvrant une fenêtre de terminal ou de commande et en tapant la commande suivante :



Cette commande affiche la version de Composer installée sur votre ordinateur. Si Composer est correctement installé, vous devriez voir s'afficher la version de Composer.

<sup>1</sup> <https://getcomposer.org/download/>

<sup>2</sup> <https://getcomposer.org/download/>

Voici un exemple de sortie de cette commande :



```
Composer version 2.5.1 2022-12-22 15:33:54
```

Pour mettre à jour Composer sous Linux et sous Windows, il faut taper la commande `composer self-update` dans leur terminal natif.

Pour ne pas avoir de limitation dans le choix des composants, il faut également installer un autre outil, Git, qui est nécessaire pour la plupart des bibliothèques.

## B. Installation de Git

Pour installer Git sur l'ordinateur, il faut commencer par télécharger l'installateur de Git à partir du site web de Git : [git-scm.com](https://git-scm.com)<sup>1</sup>.

Sous Windows, pour installer Git, il faut installer le fichier binaire avec l'extension .exe qui est sur le site officiel de Git. Il existe 2 versions de Git, la version standalone et la version portable, il est recommandé d'utiliser la version standalone pour avoir une version complète.

Ces 2 versions ont elles-mêmes 2 installateurs différents en fonction de la version du système Windows installée sur la machine (32-bit ou 64-bit). Sous Linux, il faudra exécuter la commande en ligne de commande comme pour Composer et la commande diffère en fonction de la distribution Linux. Sous Mac, il est possible de passer soit par la ligne de commande, soit avec le fichier binaire mais comme sous Linux, ceci dépendra de la distribution. Pour Linux et MacOS il est également possible d'installer Git via les gestionnaires de paquet tel que APT ou BREW.

---

<sup>1</sup> <https://git-scm.com/downloads>




Exécutez ensuite l'installateur et suivez les instructions à l'écran pour installer Git. Une fois l'installation terminée, ouvrez une fenêtre de terminal ou de commande puis vérifiez que Git est correctement installé en tapant la commande suivante :

A stylized illustration of a terminal window. It has a light gray background and rounded corners. At the top left, there are three colored circles: red, yellow, and green. In the center of the window, the text 'git --version' is displayed in a green, monospaced font.

```
git --version
```

Cette commande affiche la version de Git installée sur l'ordinateur. Si Git est correctement installé, on devrait voir s'afficher la version de Git.

Voici un exemple de sortie de cette commande :

A terminal window with a white background and rounded corners. At the top, there are three colored circles: red, yellow, and green. Below them, the text "git version 2.39.0" is displayed in a monospaced font, with "git" in orange and "version 2.39.0" in black.

```
git version 2.39.0
```

Git est maintenant installé sur l'ordinateur et prêt à être utilisé. On peut maintenant cloner des répertoires Git, créer des dépôts Git et effectuer toutes les autres opérations de gestion de version que vous souhaitez avec Git.

Il existe également des distributions de Git qui incluent un Environnement de Développement Intégré (IDE) et d'autres outils de développement. Si on souhaite utiliser ces distributions, on devra télécharger et installer l'une de ces distributions au lieu de l'installateur de Git standard.

#### Méthode

Pour déterminer si la machine fonctionne sous la version 32 bits ou 64 bits de Windows il faut :

- Se rendre dans le menu Démarrer,
- Se rendre dans Paramètres,
- Se rendre dans Système,
- Se rendre dans À propos de,
- Regarder dans la partie « *Spécifications de l'appareil* » et plus particulièrement à la ligne « *Type du système* »,
- Vérifier dans la partie « *Spécifications Windows* » l'édition et la version de Windows exécutée par l'appareil.

## C. Installer un bundle

Pour chaque projet, il y a un fichier `composer.json`. Le fichier `composer.json` est utilisé pour gérer les dépendances d'un projet PHP. Il est utilisé pour définir les bibliothèques et les packages dont un projet a besoin pour fonctionner correctement. Le fichier `composer.json` peut être créé pour n'importe quel type de projet PHP, que ce soit un site web, une application web, une application de bureau, etc.

Le fichier `composer.json` définit les informations suivantes pour chaque dépendance :

- Le nom du package,
- La version requise,
- Les dépendances supplémentaires nécessaires.

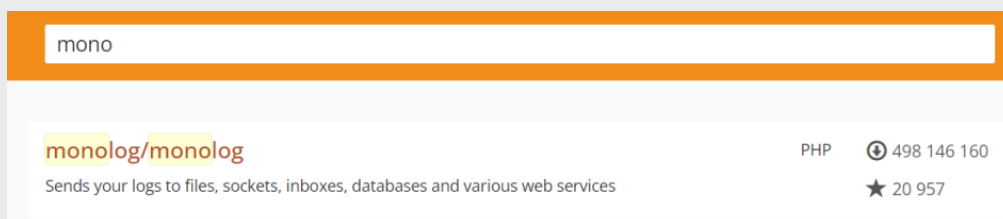
En utilisant Composer, vous pouvez automatiser le processus d'installation et de mise à jour de ces dépendances, ce qui vous permet de vous concentrer sur le développement de votre projet sans avoir à vous soucier de la gestion des dépendances.

### Recherche d'un bundle sur Packagist

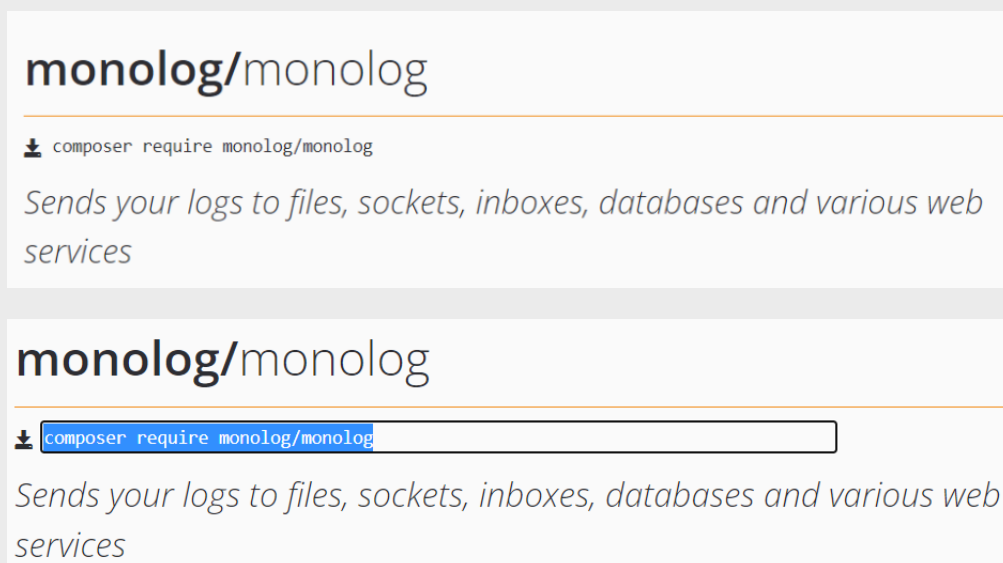
L'installation d'un bundle se fait en plusieurs étapes. Maintenant que nous avons installé Composer, il faut nous rendre sur le site Packagist.org. À la suite de quoi, on utilisera la barre de recherche pour trouver un bundle qui correspond à nos besoins. Il est donc de votre ressort de choisir celui que vous souhaitez.

#### Exemple

Imaginons que l'on souhaite intégrer le bundle « *monolog/monolog* ». Voici ce que l'on devra obtenir sur notre page :



On va venir cliquer sur le lien menant à notre package et copier la première ligne qui s'offre à nous, ici, la ligne de commande qui nous intéresse est `composer require monolog/monolog`



#### Méthode Ajout du bundle au fichier composer.json

Vous allez ensuite ouvrir votre terminal en prenant soin de vous placer dans le répertoire de votre projet puis vous allez coller la ligne de commande précédemment copiée, ici, `composer require monolog/monolog`.

Cela fait, vous n'aurez plus qu'à valider les dépendances ajoutées en répondant « yes » à la demande du terminal.

#### Méthode Exécution de la commande Composer pour installer le bundle

Toujours dans le terminal dans le répertoire de votre projet PHP, vous allez exécuter la commande `composer install`. Cette commande téléchargera les fichiers du bundle et les dépendances associées à partir du dépôt Packagist et les installera dans le répertoire `vendor`. Enfin, vous n'aurez plus qu'à attendre la fin de l'installation.

Après l'exécution de la commande `composer install`, le bundle est maintenant installé et prêt à être utilisé dans votre projet PHP. Vous pouvez maintenant vérifier l'installation en consultant le fichier `composer.lock` et en vérifiant la présence du dossier du bundle dans le répertoire `vendor`.

Pour mettre à jour les dépendances du projet, il faut utiliser une commande Composer dans le terminal. Sous Windows et Linux il s'agit de `composer update`. Elle télécharge les dernières versions des packages nécessaires et met à jour les fichiers `Composer.lock`, `symfony.lock` et le dossier `vendors` pour refléter ces mises à jour.

## D. Symfony Flex

Symfony Flex est un outil qui facilite l'installation, la configuration et la gestion des dépendances dans les projets Symfony. Il utilise des « recettes » pour automatiser ces tâches en fournissant des instructions pour installer et configurer des packages spécifiques.

Lorsqu'un package est installé avec Symfony Flex, il est ajouté à un fichier appelé `symfony.lock`, qui enregistre la version exacte de chaque package installé ainsi que toutes ses dépendances. Cela garantit que les versions des packages utilisés dans le projet sont cohérentes, ce qui réduit les risques de conflits de dépendances et de comportement inattendu.

Les recettes Symfony Flex sont des scripts qui automatisent l'installation et la configuration de packages. Ils peuvent ajouter des fichiers de configuration, des services, des routes et même modifier des fichiers existants dans le projet pour s'assurer que le package est correctement intégré. Les recettes peuvent également définir des scripts qui seront exécutés après l'installation pour effectuer des tâches supplémentaires telles que la création de tables de base de données.

Les recettes sont stockées dans un répertoire appelé « *recipes* » à la racine du projet et sont téléchargées automatiquement lorsqu'un nouveau package est installé. Les recettes peuvent être personnalisées pour répondre aux besoins spécifiques d'un projet en modifiant les fichiers existants ou en ajoutant de nouveaux fichiers à la recette.

En utilisant Symfony Flex, les développeurs peuvent gagner du temps en automatisant les tâches d'installation et de configuration de packages, tout en garantissant la cohérence des versions des packages utilisés dans le projet. Le fichier `symfony.lock` assure que les versions des packages restent stables et cohérentes, ce qui facilite la gestion des dépendances et la résolution des conflits.

## E. La gestion de versions

La gestion de versions est une pratique essentielle dans le développement logiciel qui permet de suivre les modifications apportées à un code source au fil du temps. Lorsqu'il s'agit de gérer les dépendances d'un projet, il est important de prendre en compte les bonnes pratiques pour éviter les conflits et les erreurs.

L'un des principaux éléments à prendre en compte lors de la gestion des dépendances est la gestion des fichiers de verrouillage. Ces fichiers, tels que `composer.lock` et `symfony.lock`, enregistrent les versions exactes des packages installés et leurs dépendances. Il est important de s'assurer que ces fichiers sont toujours à jour et de les inclure dans le système de gestion de version, afin que tous les membres de l'équipe travaillent avec la même version de chaque dépendance.

Cependant, il ne faut pas versionner le dossier « *vendors* » qui contient les packages installés eux-mêmes, car cela peut entraîner des conflits inutiles et augmenter considérablement la taille de l'historique de version. Au lieu de cela, il est préférable d'ajouter le dossier « *vendors* » à un fichier `.gitignore` ou équivalent, afin qu'il ne soit pas suivi par le système de gestion de version.

En cas de conflits entre les branches sur les fichiers de verrouillage, il est important d'être préparé à les résoudre de manière appropriée. En général, il est recommandé de commencer par résoudre les conflits manuellement, en comparant les différences entre les 2 fichiers de verrouillage et en fusionnant les changements. Si les conflits sont trop nombreux ou trop complexes à résoudre manuellement, il peut être nécessaire de supprimer les fichiers de verrouillage et de monter les versions des packages dans le fichier `composer.json` ou équivalent. Dans ce cas, il est important de veiller à ce que les versions montées soient cohérentes et fonctionnent correctement avec le reste du projet.

En somme, la gestion de version avec les dépendances est une pratique importante qui permet de garantir la cohérence et la fiabilité des projets logiciels. En s'assurant de maintenir à jour les fichiers de verrouillage, de ne pas versionner les dossiers *vendors* et de résoudre les conflits de manière appropriée, les développeurs peuvent éviter les erreurs et garantir la stabilité du projet.

## F. Supprimer un bundle

Pour supprimer un bundle installé avec Composer, vous devez commencer par ouvrir le terminal et vous déplacer vers le répertoire du projet PHP. Par la suite, exécutez la commande `composer remove nom_du_paquet` en remplaçant `nom_du_paquet` par le nom du paquet du bundle que vous souhaitez supprimer. Attendez ensuite la fin de la suppression.

Cette commande supprimera le paquet du bundle et les dépendances associées de votre projet. Cependant, notez que les dépendances qui sont également utilisées par d'autres paquets peuvent être conservées. Il est donc important de vérifier les dépendances avant de supprimer un bundle pour éviter les problèmes futurs.

De plus, après avoir exécuté la commande `composer remove`, n'oubliez pas de mettre à jour le fichier `composer.json` en enlevant le nom du paquet du bundle. Enfin, mettez à jour le fichier `composer.lock` pour refléter les modifications apportées à votre projet.

## G. Exercice : Quiz

[solution n°2 p.18]

### Question 1

Comment doit-on installer Composer sur Windows ?

- ☐ En ligne de commande
- ☐ Via un App Store
- ☐ En utilisant l'installateur

### Question 2

Quelle commande faut-il utiliser pour mettre à jour Composer ?

- ☐ Composer self-update
- ☐ Composer update
- ☐ Composer install

Question 3

Comment doit-on installer Git sur Windows ?

- ☐ En ligne de commande
- ☐ En utilisant le binaire
- ☐ Aucune de ces propositions

Question 4

Le fichier qui gère les dépendances est :

- ☐ Composer.lock
- ☐ Symfony.lock
- ☐ Composer.json

Question 5

Quelle commande utilise-t-on pour mettre à jour les dépendances du projet ?

- ☐ Composer install
- ☐ Composer self-update
- ☐ Composer update

## IV. L'essentiel

Composer, qui est un gestionnaire de dépendance pour PHP, utilise un fichier nommé `composer.json` pour chaque bundle nécessaire pour un projet. Ce fichier permet de gérer les dépendances du projet ainsi que leur version.

En combinant Composer avec Git, il est possible d'installer un nombre considérable de bundles trouvés sur [packagist.org](https://packagist.org) dans un projet. Lors de l'installation d'un bundle, Composer installe la dernière version disponible, mais si cela n'est pas possible en raison de la compatibilité, une version antérieure peut être utilisée.

Pour mettre à jour les dépendances, il faut le faire sur le terminal de commande en tapant la commande correspondante, tout comme pour installer les bundles ou même pour supprimer les bundles. En ce qui concerne la mise à jour de la version du bundle, il suffit de modifier le numéro de la version du bundle correspondant dans le fichier `composer.json`. Enfin, pour supprimer un bundle manuellement, il faut le supprimer du fichier `composer.json` en effaçant la ligne correspondant au bundle et en supprimant les fichiers et les dossiers correspondant au bundle.

## V. Auto évaluation

### A. Exercice

Vous êtes chargé de vous occuper d'un projet en Symfony pour une agence de communication et de développement web. Votre patron vous a demandé d'ajouter le bundle `var_dump` à votre projet. Le bundle `var_dump` est un bundle pour le framework Symfony qui permet d'afficher des informations détaillées sur les variables dans le code PHP. Cela peut être très utile pour le débogage et la compréhension du comportement de ces variables.

**Question 1**

[solution n°3 p.19]

Comment allez-vous procéder à l'installation de ce bundle ?

**Question 2**

[solution n°4 p.19]

Toujours dans le cadre de ce projet, votre patron souhaite que vous supprimiez le bundle « *symfony/translation* ». Comment allez-vous procéder ?

**B. Test****Exercice 1 : Quiz**

[solution n°5 p.19]

## Question 1

Quelle ligne de commande permet d'installer le bundle ?

- ☐ composer require symfony/symfony
- ☐ composer self-require symfony/symfony
- ☐ composer install symfony/symfony

## Question 2

Quelle commande doit-on taper pour vérifier la version de GIT installée ?

- ☐ git /version
- ☐ git --version
- ☐ git -version

## Question 3

Quelle commande sert à supprimer un bundle ?

- ☐ composer update
- ☐ composer self-update
- ☐ composer remove

## Question 4

Symfony flex gère l'activation et la désactivation du bundle.

- ☐ Vrai
- ☐ Faux

## Question 5

Depuis quelle version il n'est plus conseillé d'organiser une application en organisant le code avec des bundles ?

- ☐ 4.0
- ☐ 5.0
- ☐ 6.0


**Solutions des exercices**






**Exercice p. 6 Solution n°1****Question 1**

Composer est un :

- ☒ Gestionnaire de dépendance
- ☐ Un framework
- ☐ Un bundle
-  Il permet d'agir sur les dépendances d'un projet.


**Question 2**

Composer permet d'agir sur les dépendances en :

- ☐ Python
- ☐ JavaScript
- ☒ PHP
-  Il agit seulement sur les dépendances en PHP.


**Question 3**

Les dépendances se mettent à jour automatiquement.

- ☐ Vrai
- ☒ Faux
-  Il faut les mettre à jour manuellement via Composer update.

**Question 4**


Les dépendances peuvent dépendre d'autres dépendances.

- ☒ Vrai
- ☐ Faux
-  Ce n'est pas le cas de toutes, mais certaines dépendances peuvent dépendre d'autres. Pour qu'elles fonctionnent, on doit installer les dépendances requises.

**Question 5**

Quelle est la source qui centralise toutes les dépendances ?


- ☐ packagist.net
- ☒ packagist.org
- ☐ packagist.fr

 Le domaine qui recense toutes les sources est packagist.org.

### Exercice p. 13 Solution n°2


#### Question 1

Comment doit-on installer Composer sur Windows ?

- ☐ En ligne de commande
- ☐ Via un App Store
- ☒ En utilisant l'installateur
-  On peut l'installer en ligne de commande uniquement sous Linux et non sous Windows.


#### Question 2

Quelle commande faut-il utiliser pour mettre à jour Composer ?

- ☒ Composer self-update
- ☐ Composer update
- ☐ Composer install
-  C'est la seule commande qui permet de le faire parce que install permet d'installer les dépendances et update de mettre à jour les dépendances.


#### Question 3

Comment doit-on installer Git sur Windows ?

- ☐ En ligne de commande
- ☒ En utilisant le binaire
- ☐ Aucune de ces propositions
-  On peut l'installer en ligne de commande mais uniquement sous Linux et non sous Windows.


#### Question 4

Le fichier qui gère les dépendances est :

- ☐ Composer.lock
- ☐ Symfony.lock
- ☒ Composer.json
-  Il sert à y mettre les informations minimums requises d'installation de chacun des composants de chaque bibliothèque.

#### Question 5

Quelle commande utilise-t-on pour mettre à jour les dépendances du projet ?

- ☐ Composer install
- ☐ Composer self-update
- ☒ Composer update
-  C'est la seule qui le permet parce que self-update met seulement à jour Composer.

#### p. 15 Solution n°3

Pour ce faire, vous allez commencer par vous rendre sur le site internet Packagist.org puis vous allez taper dans sa barre de recherche le nom du bundle que vous souhaitez installer. Ici il s'agit de var-dumper. Vous allez venir copier la ligne de commande qui vous est donnée sur la page dédiée à cet bundle puis vous allez vous rendre sur votre terminal dans le répertoire de votre projet et taper « `composer require symfony/var-dumper` ». Vous validerez ensuite les dépendances avant de taper la commande « `composer install` » pour installer le bundle.

#### p. 15 Solution n°4


Pour ce faire, vous allez commencer par ouvrir votre terminale dans le répertoire de votre projet PHP. Une fois cela fait, vous devez taper la commande « `composer remove symfony/translation` » et attendre la fin du processus de suppression. Après coup, il ne vous restera plus qu'à mettre à jour le fichier `composer.json` et le fichier `composer.lock` en effectuant la commande « `composer update` ».

#### Exercice p. 15 Solution n°5

##### Question 1

---


Quelle ligne de commande permet d'installer le bundle ?

- ☒ `composer require symfony/symfony`
- ☐ `composer self-require symfony/symfony`
- ☐ `composer install symfony/symfony`
-  En effet, si l'on se rend sur Packagist.org et que l'on recherche le bundle `symfony/symfony`, la première ligne que l'on retrouvera sera `composer require symfony/symfony`.

##### Question 2

---


Quelle commande doit-on taper pour vérifier la version de GIT installée ?

- ☐ `git /version`
- ☒ `git --version`
- ☐ `git -version`
-  Il s'agit en effet de la commande `git --version`. D'ailleurs, si l'on remplace « `git` » par « `composer` », on pourra obtenir la version de Composer installée.

##### Question 3

---


Quelle commande sert à supprimer un bundle ?

- ☐ composer update
- ☐ composer self-update
- ☒ composer remove
-  C'est la seule commande qui permet de supprimer un bundle car `composer update` permet de mettre à jour les dépendances et `composer self-update` permet de mettre à jour Composer.

#### Question 4

---


Symfony flex gère l'activation et la désactivation du bundle.

- ☒ Vrai
- ☐ Faux
-  Symfony flex gère l'activation ou la désactivation du bundle lors de son installation ou de sa désinstallation.

#### Question 5

---

Depuis quelle version il n'est plus conseillé d'organiser une application en organisant le code avec des bundles ?

- ☒ 4.0
- ☐ 5.0
- ☐ 6.0
-  Depuis la version 4.0, cette pratique est déconseillée pour avoir un code plus propre et plus clair.