Менеджер пакетов PHP: composer

Composer. Понятие, применение

Composer — менеджер пакетов для PHP. Этот инструмент позволяет не только устанавливать сторонние пакеты, но и обновлять их при выходе более новых версий. Также с помощью Composer можно легко создавать пакеты для своих библиотек.

Язык программирования PHP стремительно развивается с каждым годом. Каждый месяц регистрируют десятки, а то и сотни библиотек для работы с PHP проектами.

На сайте packagist указано, что с 2012 года было зарегистрировано более 330 тысяч библиотек. Чтобы использовать любую из них в конкретном проекте достаточно установить composer и ввести одну строку в терминале:

composer install <package_name>

Composer. Установка

Самый простой и верный способ установить composer - перейти на официальный сайт (https://getcomposer.org/) и следовать инструкциям (https://getcomposer.org/download/).

Тем не менее, есть более простой способ установить composer: воспользоваться пакетным менеджеров в OS Linux. Например, для Debian команда установки будет выглядеть так:

apt install composer.

Для установки composer в Docker контейнере необходимо написать подобную инструкцию в Dockerfile:

```
COPY --from=composer:latest /usr/bin/composer /usr/bin/composer
```

При этом docker в своем репозитории найдет образ composer и скопирует bin файл в контейнер.

Composer. Синтаксис и опции

Composer — это консольная утилита, у неё нет графического интерфейса. Для работы с composer необходимо использовать командную строку.

Список всех опций и команд рассмотрим ниже:

- **-h** — вывести справку по утилите
- **-q** — сокращённый вариант вывода
- **-V** — показать версию утилиты
- **-n** — не задавать интерактивные вопросы
- **-v**, **-vv, -vvv** — настройка подробности вывода
- **-d** — использовать указанную рабочую директорию

Composer. Синтаксис и опции

- **archive** — архивирует текущий проект в качестве библиотеки для отправки в сеть
- **check-platform-reqs** — проверяет, соблюдены ли системные требования
- **create-project** — создаёт проект на основе пакета в указанную директорию
- **depends** — выводит зависимости пакета
- **dump-autoload** — обновляет систему автозагрузки классов
- **exec** — позволяет выполнять скрипты из установленных пакетов
- **init** — создает пустой проект в текущей папке
- **list** — выводит список доступных команд
- **outdated** — выводит список пакетов, для которых есть обновления
- **prohibits** — выводит названия пакетов, которые мешают установить указанный пакет
- **search** — поиск пакетов в репозиториях
- **self-update** — обновление Composer до последней версии, работает только при локальной установке
- **show** — информация о пакете
- **update** — обновляет все пакеты до самой актуальной версии

Composer. Синтаксис и опции

Composer — это консольная утилита, у неё нет графического интерфейса. Для работы с composer необходимо использовать командную строку.

Список всех опций и команд рассмотрим ниже:

- **-h** — вывести справку по утилите
- **-q** — сокращённый вариант вывода
- **-V** — показать версию утилиты
- **-n** — не задавать интерактивные вопросы
- **-v**, **-vv, -vvv** — настройка подробности вывода
- **-d** — использовать указанную рабочую директорию

Composer. Пример composer.json

```
{
    "name": "laravel/laravel",
    "type": "project",
    "description": "The Laravel Framework.",
    "keywords": ["framework", "laravel"],
    "license": "MIT",
    "require": {
        "php": "^8.0.2",
        "laravel/framework": "^9.19"
    },
    "require-dev": {
        "fakerphp/faker": "^1.9.1",
        "laravel/pint": "^1.0"
    },
    "autoload": {
        "psr-4": {
            "App\\": "app/",
            "Database\\Factories\\": "database/factories/",
            "Database\\Seeders\\": "database/seeders/"
        }
    },
    "autoload-dev": {
        "psr-4": {
            "Tests\\": "tests/"
        }
    },
    "scripts": {
        "post-autoload-dump": [
            "cmd"
        ],
        "post-update-cmd": [
            "cmd"
        ],
        "post-root-package-install": [
            "cmd"
        ],
        "post-create-project-cmd": [
            "@php artisan key:generate --ansi"
        ]
    },
    "repositories": [
        {
            "type": "git",
            "url": "https://github.com/someVendor/someRepo",
        }
    ],
    "minimum-stability": "stable",
    "prefer-stable": true
}
```

Composer. Require

Рассмотрим установку пакетов с помощью composer.

Для того, чтобы добавить пакет к проекту необходимо вызвать команду

`composer require vendor/package`

которая добавит требуемый пакет в файл `composer.json` и установит его в проект.

Команда `require` изменяет `composer.json`, находящийся в текущей папке. Если пакету требуются зависимости, то они будут установлены или обновлены. А также будет обновлён `composer.lock`.

Команда require позволит вам установить пакет самой свежей версии, совместимой с остальными пакетами. Чтобы указать конкретную версию пакета необходимо после его названия указать версию через знак “:”.

`composer require vendor/package:1.2.0`

Composer. Require

При вызове команды require заданный пакет ищется на сайте https://packagist.org/

Сопоставляя настройки проекта, composer производит анализ зависимостей и их версий, которые необходимо установить.

Как правило, бОльшая часть проектов на packagist содержится на github, который является удобным инструментом для хранения и управления кодом, а также выпуском релизов.

Бывают ситуации, когда нужный пакет не может быть установлен, т.к. не отвечает требованиям стабильности, описанным в composer.json. В этом случае предлагается либо заменить пакет, либо понизить уровень стабильности в настройках.

Третьим способом, не самым лучшим, является форк (fork) необходимого репозитория и создание тега (версии) для него.

Composer. Json vs Lock

При работе с composer в проекте появляются два файла: composer.json и composer.lock.

В первом файле хранится информация о тех пакетах, которые необходимо установить, а во втором - пакеты, которые установлены.

Важно отметить, что в файле composer.json информация о пакетах и их версиях может быть не точной, а задана рамками. Например,

```
"require": {
        "php": ">=8.0.2",
        "laravel/framework": "9.0.*"
    }
```

в секции require указано, что пакет laravel/framework должен иметь версию релиза 9, мажорную версию 0 и любую минорную.

В composer.lock запишется, что установлена конкретная версия, например, 9.0.1.

Composer. Json vs Lock

Достаточно часто у неопытных разработчиков появляется вопрос: “а нужно ли хранить composer.lock в проекте? Ведь он описывает версии пакетов, которые установлены на моем проекте, в моей локальной версии”?

Ответ: Нужно в 99,9% случаях.

Выпуск новой версии пакета говорит о том, что произошли какие-либо изменения в коде, т.е. в логике работы пакета. Соответственно, логика работа программы, связанная с этим пакетом, может быть изменена и работать иначе, чем задумывал разработчик, работая с предыдущей версией пакета.

Стабильность работы программного обеспечения основывается на использовании одних и тех же версий основного и дополнительного ПО в разных средах разработки.

Composer. Composer.lock

Случаи, когда composer.lock не нужно присоединять к проекту:

1. Когда проект предназначен только для использования на одной машине и разработка на проекте ведется только на этой машине.
2. Когда проект является open-source и разработчики не хотят, чтобы информация о конкретных версиях зависимостей была доступна всем пользователям.
3. Когда в проекте используются зависимости, которые часто обновляются и обновления не приводят к изменению функциональности проекта. В этом случае, можно использовать символьные версии зависимостей в composer.json без указания конкретных версий, что позволит Composer автоматически устанавливать самые последние версии зависимостей.