

DAWS – 2º

Composer para PHP

JESÚS MOLINA HERNÁNDEZ

jmolina@florida-uni.es



Florida

Secundària

Índice

- 1. Introducción a Composer**
- 2. composer.json: Definiendo dependencias**
- 3. Instalando dependencias con Composer**
- 4. Autoloading en Composer**
- 5. Actualización de dependencias**
- 6. Buenas prácticas con Composer**

Introducción a Composer

¿Qué es Composer?

- Composer es un gestor de dependencias para PHP.
- Permite manejar bibliotecas y herramientas externas en tus proyectos.
- Funciona declarando las bibliotecas que necesita tu proyecto, y Composer se encarga de instalarlas y actualizarlas.

¿Por qué usar Composer?

- Simplifica el manejo de librerías y versiones.
- Facilita el autoloading de clases. → **Incluso útil para el autoloading de tus propias clases**
- Centraliza la gestión de dependencias para que no tengas que preocuparte por instalar todo manualmente.

composer.json: Definiendo dependencias

¿Qué es composer.json?

- Es el archivo que contiene la configuración del proyecto: nombre, versión, autor, y dependencias.

El fichero composer.json tiene muchas más secciones pero las que más nos importan son la de **require** y la de **autoload**.

```
1  {
2      "name": "nombre-del-proyecto/proyecto",
3      "description": "Descripción del proyecto",
4      "type": "project",
5      "license": "MIT",
6      "authors": [
7          {
8              "name": "Jesus Molina",
9              "email": "jesus.molina@florida.com"
10         }
11     ],
12     "require": {
13         "php": "^7.4 || ^8.0",
14         "monolog/monolog": "^2.0"
15     },
16     "autoload": {
17         "psr-4": {
18             "App\\": "src/"
19         }
20     }
21 }
```

Instalando dependencias con Composer

Comando:

```
composer install
```

→ Se ejecuta en el terminal.

→ Crea una carpeta vendor/ con todas las dependencias necesarias

¿Cómo cargo un nuevo paquete de terceros?

NO editar a mano la sección “require” del composer.json

Pasos:

1. Inicia Composer en tu proyecto (crea el composer.json) `composer init`
2. Agregar el paquete al proyecto `composer require justinrainbow/json-schema`
3. Verificar la instalación del paquete: Ver fichero composer.json actualizado y la carpeta /vendor con los nuevos paquetes.
4. Incluir el autoload en tu proyecto
5. Usar el paquete en tu código

Autoloading en Composer

Esto enlaza (y sustituye) al autoloader de clases que ofrece PHP.

→ Composer incluye automáticamente las clases de tus dependencias usando el autoloader.

```
111     require 'vendor/autoload.php';
112
113     use Monolog\Logger;
114     use Monolog\Handler\StreamHandler;
115
116     // Crear un nuevo logger
117     $log = new Logger('nombre');
118     $log->pushHandler(new StreamHandler('app.log', Logger::WARNING));
119
```

Esto evita tener que hacer require manual de cada archivo de clase.

Autoloading en Composer

Cómo se realiza Autoload de mis propias clases?

- **Instalar Composer.**
- **Inicializar Composer en el proyecto** (crear composer.json).
- **Configurar el autoloading PSR-4** en composer.json para tus clases.
- **Estructurar tus clases** con namespaces que coincidan con la estructura de carpetas.
- **Ejecutar composer dump-autoload** para generar el autoloader.
- **Incluir vendor/autoload.php** en tu archivo principal y usar tus clases normalmente

Claves

1. Usar PSR-4 en tu código.
2. Tener todas las clases dentro de la carpeta /src.

Actualización de dependencias

Comando para actualizar todas las dependencias:

```
composer update
```

Comando para actualizar una dependencia específica:

```
composer update monolog/monolog
```

Comando para actualizar sólo las clases locales:

```
composer dump-autoload
```

Buenas prácticas con Composer

Definir correctamente las restricciones de versión: Usa ^ para compatibilidad de versiones mayores.

Usa ~ para más restricciones en actualizaciones

Ignorar vendor/ en repositorios

Añade la carpeta vendor/ a .gitignore para no subir las dependencias al control de versiones

Usar composer.lock

Asegúrate de **versionar composer.lock** para garantizar que todos los desarrolladores usen las mismas versiones de las dependencias.