



# PHP – Composer

---



**Ciclo:** DAW  
**Módulo:** DWES  
**Curso:** 2020-2021  
**Autor:** César Guijarro Rosaleny



<b>Introducción.....</b>	<b>3</b>
<b>Instalar Composer.....</b>	<b>4</b>
Hacer accesible Composer desde cualquier sitio.....	6
<b>Usar Composer.....</b>	<b>7</b>
Inicializar Composer.....	7
Añadir librerías externas.....	8
La carpeta vendor.....	13
<b>Repositorio de paquetes PHP.....</b>	<b>15</b>

## Introducción

---

Composer es un gestor de dependencias para PHP. Esto significa que nos permite gestionar los paquetes usados en nuestros proyectos de forma automática, sin necesidad de tener que gestionarlos nosotros manualmente.

Cuando trabajamos en un proyecto PHP es habitual incorporar librerías de terceros para evitar tener que hacer todo el trabajo desde cero. Para hacerlo, debemos ir a la web donde está publicada la librería que queremos usar, descargarla y añadirla a nuestro proyecto. Si a mitad de nuestro proyecto cambian la versión de esa librería, deberíamos borrar la anterior, volver a descargar y añadir la nueva.

Además, muchas librerías de terceros dependen, a su vez, de otras, con lo que tendríamos que mantenerlas a mano.

Para solucionar todos estos problemas (y otros), tenemos los gestores de paquetes, que nos ayudan a automatizar todas esas tareas.

Los desarrolladores en Node.js ya están familiarizados con un gestor de paquetes como **npm**, y, desde hace unos años, los programadores PHP contamos con **Composer**, lo cual nos ahorra mucho tiempo y esfuerzo.

## Instalar Composer

---

Composer es multiplataforma (funciona el Linux, Windows y Mac).

Para instalarlo de forma sencilla en Linux o Mac, podemos ir a su página de [descarga](#), copiar el script de instalación en la terminal y ejecutarlo.

Este script descargará el instalador de composer (*composer-setup.php*), verificará el archivo descargado, ejecutará el instalador y por último lo eliminará.

Si te da un error a mitad de la ejecución del script anterior como el siguiente:

```
cesar@Asus:~$ php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
cesar@Asus:~$ php -r "if (hash_file('sha384', 'composer-setup.php') === 'e5325b19b381bfd88ce90a5ddb7823406b2a38cff6bb704b0acc289a09c8128d4a8ce2bbafcd1fcbdc38666422fe2806') { echo 'Installer verified'; } else { echo 'Installer corrupt'; unlink('composer-setup.php'); } echo PHP_EOL;"
Installer verified
cesar@Asus:~$ php composer-setup.php
All settings correct for using Composer
Unable to write keys.dev.pub to: /home/cesar/.composer
cesar@Asus:~$
```

Puede que tengas que darle permiso para ejecutar la orden *php composer-setup.php*:

```
cesar@Asus:~$ sudo php composer-setup.php
All settings correct for using Composer
Downloading...

Composer (version 1.10.10) successfully installed to: /home/cesar/composer.phar
Use it: php composer.phar
```

Con esto ya tenemos instalado Composer. Para comprobar nuestra instalación ejecutamos ***php composer.phar*** en la terminal y debería salirnos la versión instalada y las opciones.

```
cesar@Asus:~$ php composer.phar

Composer version 1.10.10 2020-08-03 11:35:19

Usage:
  command [options] [arguments]

Options:
  -h, --help                Display this help message
  -q, --quiet               Do not output any message
  -V, --version             Display this application version
  --ansi                    Force ANSI output
```

Para instalarlo en Windows simplemente tenemos que bajarnos el [instalador](#) y ejecutarlo.

## Hacer accesible Composer desde cualquier sitio

Si hemos instalado Composer en Windows, el instalador ya se encarga de configurar el PATH para que podamos ejecutar Composer desde cualquier directorio.

Por el contrario, si lo hemos instalado en Linux o Mac, sólo podremos utilizarlo en la carpeta donde lo hayamos instalado. Para que sea accesible desde cualquier sitio debemos mover Composer a un directorio global.

```
sudo mv composer.phar /usr/local/bin/composer
```

Ahora deberíamos poder utilizar Composer en cualquier proyecto que creemos sin necesidad de tener que instalarlo cada vez.

[illegible]

## Usar Composer

---

Composer tiene una lista de comandos que nos permite realizar diferentes tareas. Nosotros vamos a utilizar los comandos básicos con un ejemplo para ir familiarizándonos con él.

### Inicializar Composer

Copia el archivo *actores.php* de la práctica 1.1 en la carpeta *apache\_htdocs/DWS/ejemplos/T3/composer* y crea el archivo *index.php* con el siguiente contenido:

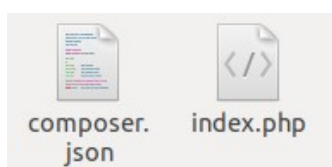
```
<?php  
$actores = include "actores.php";
```

A continuación, abre una terminal en la carpeta y ejecuta la siguiente orden:

```
composer init
```

Esta orden inicializará Composer y tendremos que definir una serie de parámetros de nuestro proyecto (puedes dejar las opciones que vienen por defecto).

Si miramos nuestra carpeta, veremos que nos ha creado un nuevo archivo llamado **composer.json**:



cuyo contenido será algo parecido a ésto:

```
{  
    "name": "cesar/composer",  
    "authors": [  
        {  
            "name": "César Guijarro",  
            "email": "cguijarro@fpmislata.com"  
        }  
    ],  
    "require": {}  
}
```

En este archivo tenemos el nombre del proyecto, los autores y las librerías que utilizamos (en principio ninguna).

## Añadir librerías externas

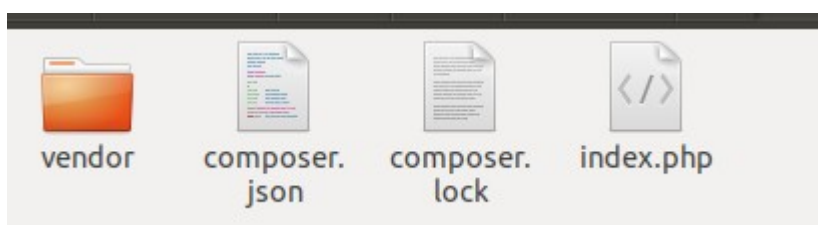
Vamos a añadir a nuestro proyecto una librería llamada *mtdowling/jmespath* para tratar *arrays* en PHP de forma más cómoda.

Ejecuta la siguiente orden en la terminal:

```
composer require mtdowling/jmespath.php
```



La orden **composer require** se utiliza para instalar librerías externas y sus dependencias. Si miramos nuestra carpeta del proyecto veremos que se ha creado un archivo nuevo y una nueva carpeta:



Por ahora podemos olvidarnos del archivo *composer.lock* (**no** debemos borrarlo). Este archivo tiene el registro exacto de las versiones de dependencia que se han instalado.

La carpeta *vendor* es la que contendrá todos los archivos de las librerías que hemos instalado en nuestro proyecto (y otras librerías si éstas últimas las necesitasen). Si añadiésemos otras librerías, se añadirían a esta carpeta automáticamente, con lo que nosotros no tendríamos que preocuparnos de nada.

Ahora que ya hemos instalado nuestra librería, veamos como se usa.

JMESPath es un lenguaje para extraer datos de un documento JSON. En su [web](#) podemos ver tutoriales y ejemplos de como usarla.

La librería *mtdowling/jmespath* nos permite usar ese lenguaje con estructuras de datos PHP (como los *arrays*).

Por ejemplo, si queremos sacar el nombre del actor con id = 5, no hace falta recorrer el *array* con un bucle. Podemos utilizar la librería de la siguiente forma para mostrar el nombre:

```
<?php

$actores = include "actores.php";

$expression = "[?id == `5`.nombre";
$result = JmesPath\search($expression, $actores);
var_dump($result);
```

De esta forma, utilizamos el método **JmesPath\search** de la librería para pasarle una expresión y un array y obtener un resultado. En este caso con la expresión "[?id == `5`.nombre" le estamos diciendo que nos busque los nombres de los actores con id = 5 (el método devolverá un *array* aunque sólo exista un resultado).

Si ejecutamos el ejemplo tal y como está no nos funcionará, ya que primero tenemos que incluir los ficheros de la librería mediante **include** o **require**.

(!) Fatal error: Uncaught Error: Call to undefined function JmesPath\search() in /var/www/html/DWS/ejemplos/T3/index.php on line 8				
(!) Error: Call to undefined function JmesPath\search() in /var/www/html/DWS/ejemplos/T3/index.php on line 8				
Call Stack				
#	Time	Memory	Function	Location
1	0.1678	398960	{main}()	.../index.php:0

Ésto puede ser algo tedioso, ya que si usamos varias librerías de terceros deberemos incluir todas ellas en nuestro archivo PHP.

Por fortuna, hay una solución sencilla para ello: usar un **Autocarga de clases**. Básicamente, PHP dispone de un mecanismo para intentar cargar las clases que no están definidas todavía como último recurso.

Si intentamos acceder a una clase que no hemos definido, PHP comprobará si existe un **Autoload**, y, si es así, ejecutará el código incluido en él. Ésto se suele utilizar con frecuencia para no tener que incluir todos los archivos necesarios en nuestra aplicación PHP.

Composer ya viene con un *autoloader* propio, con lo que para utilizarlo sólo tendremos que incluirlo (sólo el *autoloader*, sin ningún otro archivo de las librerías que utilicemos) al principio de nuestro archivo de la siguiente forma:

```
<?php

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

$actores = include "actores.php";

$expression = "[?id == `5`.nombre";
$result = JmesPath\search($expression, $actores);
var_dump($result);
```

Ahora sí, al ejecutar el archivo debería mostrarnos el resultado:



```
/var/www/html/DWS/ejemplos/T3/index.php:9:
array (size=1)
  0 => string 'Diane Keaton' (length=12)
```

Si quisiésemos, por ejemplo, sacar por pantalla el nombre y el año de nacimiento de los actores con id 3, 4 y 6:

```
<?php

$expression = '[?contains(["3", "4", "6"], id)].{Nombre: nombre, Anyo:
anyo}';
$result = JmesPath\search($expression, $actores);
var_dump($result);
```



```
Aplicaciones cesguir Instituto ubuntu C
/var/www/html/DWS/ejemplos/T3/index.php:13:
array (size=3)
  0 =>
    array (size=2)
      'Nombre' => string 'Robert Duvall' (length=13)
      'Anyo' => int 1931
  1 =>
    array (size=2)
      'Nombre' => string 'James Caan' (length=10)
      'Anyo' => int 1940
  2 =>
    array (size=2)
      'Nombre' => string 'Robert de Niro' (length=14)
      'Anyo' => int 1943
```

Como hemos dicho antes, en la web de [JMESPath](https://jmespath.org/) hay tutoriales y ejemplos de como utilizar este lenguaje para sacar datos de un *array* (en realidad es para sacar datos de un documento JSON, pero gracias a la librería *mtdowling/jmespath* podemos usarlo con *arrays PHP*).

## La carpeta vendor

Como hemos visto, la carpeta *vendor* se crea automáticamente cuando instalamos alguna librería externa a nuestro proyecto, y es donde se almacena el código de todas esas librerías.

Por esta razón, esta carpeta puede llegar a ocupar bastante tamaño, con lo que subir nuestro proyecto a un servidor o a otro ordenador puede tardar mucho tiempo (incluso puede haber servidores que no te permitan subir todo el proyecto por excesivo tamaño).

Afortunadamente, Composer tiene una solución para eso. A la hora de trasladar nuestro proyecto podemos ignorar la carpeta *vendor* y utilizar la siguiente orden para instalar de nuevo todas nuestras dependencias:

**composer install**

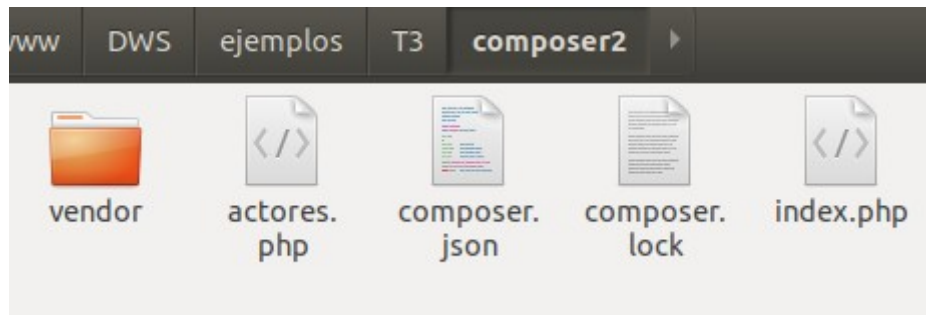
Vamos a hacer una prueba. Copia la carpeta *composer* en la misma ubicación y cámbiale el nombre a *composer2*. A continuación, borra la carpeta *vendor* de *composer2*. Obviamente, si intentamos ejecutar nuestra aplicación, nos dará un error, ya que no encuentra el archivo *vendor/autoload.php*.

**Warning:** require(/var/www/html/DWS/ejemplos/T3/composer2/vendor/autoload.php): failed to open stream: No such file or directory in /var/www/html/DWS/ejemplos/T3/composer2/index.php on line 3

Call Stack				
#	Time	Memory	Function	Location
1	0.2862	398416	{main}()	.../index.php:0

**Fatal error:** require(): Failed opening required '/var/www/html/DWS/ejemplos/T3/composer2/vendor/autoload.php' (include\_path='.:usr/local/lib/php') in /var/www/html/DWS/ejemplos/T3/composer2/index.php on line 3

Abre ahora una terminal en la carpeta y ejecuta la orden anterior (**composer install**). Lo que hará esta orden es leer el archivo *composer.json* y si existe alguna librería externa, crear la carpeta *vendor* y descargar las librerías usadas:



Si abrimos nuestra web en un navegador, veremos que ahora todo funciona como toca.

```

Aplicaciones  cesguirio  Instituto  ubuntu  Cosas

/var/www/html/DWS/ejemplos/T3/composer2/index.php:13:
array (size=3)
  0 =>
    array (size=2)
      'Nombre' => string 'Robert Duvall' (length=13)
      'Anyo' => int 1931
  1 =>
    array (size=2)
      'Nombre' => string 'James Earl Ray' (length=13)
      'Anyo' => int 1938

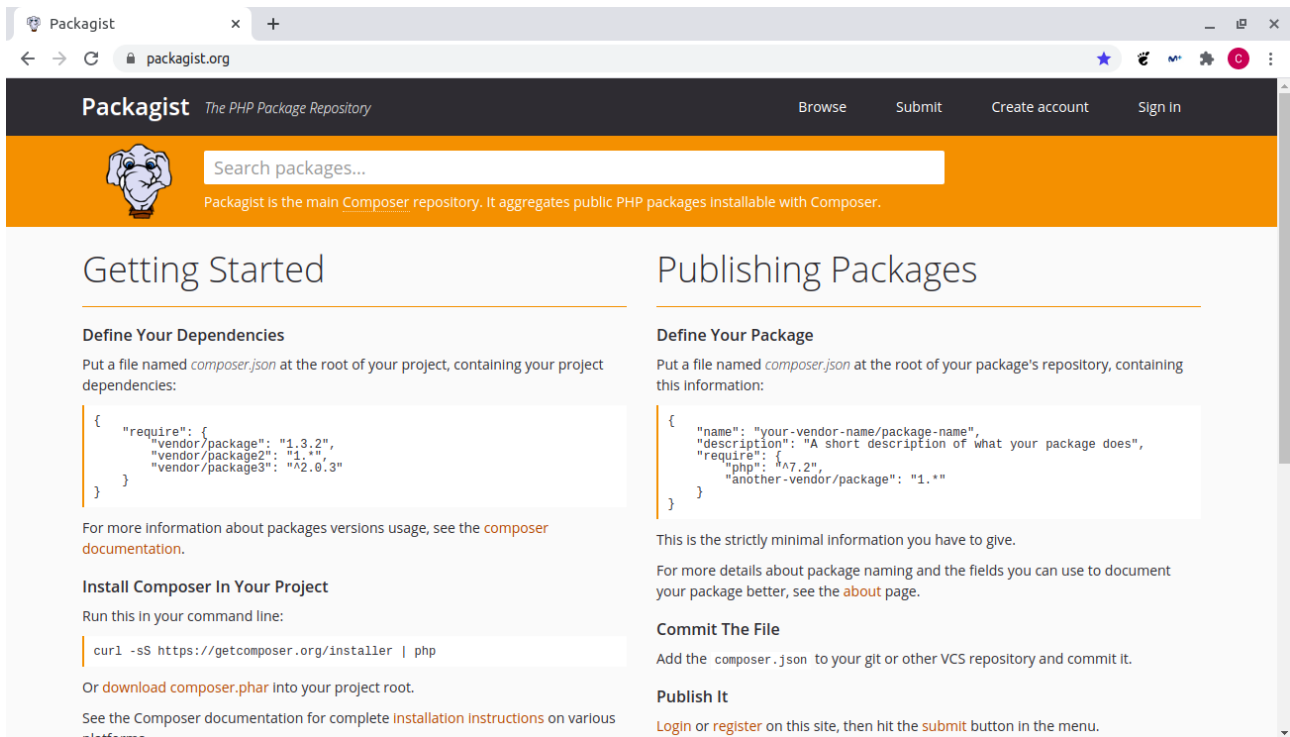
```

Por lo tanto, siempre es recomendable no subir la carpeta *vendor* y ejecutar primero la sentencia **composer install** cuando instalemos nuestro proyecto en otra ubicación, con lo que nos ahorramos subir todos los archivos de las librerías usadas.

Existen muchas otras opciones al utilizar Composer (como **composer update**, que actualiza nuestras librerías externas). Veremos algunas de ellas a lo largo del curso.

# Repositorio de paquetes PHP

Para encontrar librerías externas útiles a nuestro proyecto, podemos utilizar [Packagist](https://packagist.org), web que es un repositorio de paquetes php.



The screenshot shows the Packagist website interface. At the top, there's a navigation bar with 'Packagist The PHP Package Repository' and links for 'Browse', 'Submit', 'Create account', and 'Sign in'. Below this is a search bar with the placeholder 'Search packages...'. The main content area is divided into two columns: 'Getting Started' and 'Publishing Packages'.

**Getting Started**

**Define Your Dependencies**

Put a file named `composer.json` at the root of your project, containing your project dependencies:

```
{
  "require": {
    "vendor/package": "1.3.2",
    "vendor/package2": "1.*",
    "vendor/package3": "^2.0.3"
  }
}
```

For more information about packages versions usage, see the [composer documentation](#).

**Install Composer In Your Project**

Run this in your command line:

```
curl -sS https://getcomposer.org/installer | php
```

Or [download composer.phar](#) into your project root.

See the [Composer documentation](#) for complete [installation instructions](#) on various platforms.

**Publishing Packages**

**Define Your Package**

Put a file named `composer.json` at the root of your package's repository, containing this information:

```
{
  "name": "your-vendor-name/package-name",
  "description": "A short description of what your package does",
  "require": {
    "php": "^7.2",
    "another-vendor/package": "1.*"
  }
}
```

This is the strictly minimal information you have to give.

For more details about package naming and the fields you can use to document your package better, see the [about](#) page.

**Commit The File**

Add the `composer.json` to your git or other VCS repository and commit it.

**Publish It**

[Login](#) or [register](#) on this site, then hit the [submit](#) button in the menu.

Aquí podemos buscar las librerías hechas por terceros que nos pueden ayudar en nuestro proyecto. Además, podemos publicar nuestras propias librerías, y tiene una guía rápida para utilizar Composer.