Programación III
Composer

Clase 4 - parte 2

Maximiliano Neiner

Temas a Tratar

- Composer

Temas a Tratar

- Composer
  - Generalidades
  - Packagist
  - Esquema - composer.json

Composer

Composer es una herramienta para la gestión de dependencias en PHP.

Permite declarar las bibliotecas de las que depende tu proyecto y las administrará (instalará/actualizará) para vos.

¿Por qué usarlo?

- Cuando se comienza con un proyecto en PHP, ya de cierta complejidad, no sirve sólo con la librería de funciones nativas de PHP. Generalmente se usan librerías de terceros.
- Ya sea un framework, sistemas de envío de email, validación de formularios, etc.
- Los gestores de paquetes ayudan a resumir las tareas de descarga y mantenimiento de las versiones del proyecto para que estén siempre actualizadas.
- Los gestores de paquetes ya existían en otros lenguajes de programación como ***npm*** en NodeJS.

Instalación

- El proceso es bien simple. Se reduce a estas acciones, que dependen del sistema operativo.
- **Windows:**
  Se usará un instalador (el de toda la vida). Ningún secreto. Es un asistente y vas yendo a través de varias ventanas, siguiente, siguiente, finalizar.

- **Linux / Mac:**
  Se usará la línea de comandos para instalar Composer. Es tan sencillo como ejecutar esta instrucción:

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

¿Cómo funciona?

- Para beneficiarnos del workflow que nos propone Composer, simplemente tenemos que escribir un archivo de configuración.

- El archivo es un simple JSON, en el que se indican cosas como: el autor del proyecto, las dependencias, etc.

- El archivo JSON debe tener un nombre específico: ***composer.json***

Ejemplo - composer.json (1/2)

```
{
    "name" : "vendor/nombre_proyecto",
    "description" : "descripcion_del_paquete",
    "require" : {
        "nombre_paquete_1" : "version",
        "nombre_paquete_2" : "version",
    }
}
```

- Una vez definidas las dependencias se debe instalarlas. Esto se consigue con un simple comando en el terminal:

```
composer install
```

Ejemplo - composer.json (2/2)

- Una vez finalizado el proceso en la consola de comandos se podrá encontrar en la carpeta raíz del proyecto un directorio llamado "vendor" donde estarán las librerías declaradas.
- Sólo queda hacer los includes para que estén disponibles en las aplicaciones y para ello también nos ayuda Composer.
- Simplemente tendremos que hacer un único include o require en nuestro código y todas las librerías estarán disponibles para usar.

```php
<?php

require_once "./vendor/autoload.php";
```

Temas a Tratar

- Composer
  - Generalidades
  - Packagist
  - Esquema - composer.json

Packagist

- Se trata del repositorio de paquetes que son instalables por medio de Composer.
- En la página de Packagist encontrarás un buscador. Simplemente busca y obtendrás varias opiniones clasificadas por popularidad, descargas, etc.
- Además sobre cada paquete hay información y el código necesario para declarar la dependencia en el JSON de Composer.

Temas a Tratar

- Composer
  - Generalidades
  - Packagist
  - Esquema - composer.json

Esquema composer.json (1/4)

- Define la estructura del documento así como los valores posibles que tengan cada uno de sus campos.
- **name:**
  Indicar el nombre del autor, se compone de dos partes, el "vendor" (la empresa o nick del desarrollador o grupo que lo ha creado) y el nombre del proyecto propiamente dicho.
- **description:**
  Es la descripción que ofrecemos de este paquete. Es un texto normalmente de una única línea.
- **homepage:**
  Una URL del sitio web del proyecto.

Esquema composer.json (2/4)

- **authors:**
  Es un array con los autores del proyecto. A cada uno de los elementos se le puede indicar: name, email, homepage, role (rol dentro del proyecto).

```
"authors": [
    {
        "name": "Oso Pratto",
        "email": "oso@pratto.com",
        "homepage": "http://www.modo_oso.com",
        "role": "El primero"
    },
    {
        "name": "JuanFer Quintero",
        "email": "juanfer@quintero.com.co",
        "homepage": "http://www.juanfer.com.co",
        "role": "El segundo"
    },
    {
        "name": "Pity Martínez",
        "email": "el-pity-martinez@que-loco-que-esta.com",
        "homepage": "http://www.elpitymartinez.com.ar",
        "role": "Y va el tercero, y va el tercero..."
    }
]
```

Esquema composer.json (3/4)

- El campo más importante donde debemos centrarnos es en la definición de las dependencias, así como las versiones que deseamos que sean instaladas, o actualizadas de cada una de esas dependencias.
- **require:**
  Es un objeto con una serie de pares clave/valor que definen cada una de las dependencias que Composer debe instalar para nuestro proyecto. En la clave debemos de indicar el nombre del paquete que depende (que se obtiene del sitio de Packagist) y como valor indicamos la versión que deseamos que esté instalada, o el rango de versiones.

Esquema composer.json (4/4)

- **Versión exacta:** indica que se debe instalar una versión exacta, y solo esa. Quiere decir que nunca te va a actualizar el paquete, porque tu proyecto debe tener esa versión y no otra. Por ejemplo "4.3.1".
- **Rango de versiones:** permite indicar versiones que sean mayor que una determinada, menor o que esté entre una versión y otra. Por ejemplo ">=2.0".
- **Comodín:** Permite decir cualquier versión de una release mayor. Ejemplo "4.*" para indicar que se deje siempre la versión 4 y cualquier cosa.
- **Virgulilla (~):** permite indicar la versión de una manera diferente, "próxima versión significativa". Por ejemplo, "~2.2" siempre te dejará la versión mayor o igual a la 2.2 y menor que la 3.0.

Ejercitación