Desarrollo de Aplicaciones Web

COMPOSER

ITCA-ZACATECOLUCA

Técnico en Ingeniería de Sistemas Informáticos

Cristhian Alexander González Puro 22 "B"

14-5-2015

COMPOSER

Composer es una herramienta para gestionar las dependencias de las aplicaciones PHP. Una vez declaradas las librerías de las que depende tu proyecto, Composer es capaz de descargar e instalar automáticamente las versiones correctas de cada una de esas librerías.

Características:

- ◊ Composer es un gestor de dependencias y no de paquetes.
- ◊ La instalación siempre es local para cada proyecto.
- ◊ Permite declarar las librerías de las cuales el proyecto depende o necesita y éste las instala en el proyecto.
- ◊ Composer requiere PHP 5.3.2+ para correr.
- ◊ Es una herramienta multiplataforma, por lo que funciona igual de bien en servidores Windows, Linux y Mac OS X.

Instalación y configuración de Composer en Windows.

La instalación puede hacerse de dos formas:

1 Usando el instalador:

Consiste en descargar y ejecutar el archivo “Composer-Setup.exe”, que instala la versión más reciente de Composer y actualiza el PATH de tu ordenador para que puedas ejecutar Composer simplemente escribiendo el comando Composer.

2 Haciendo una instalación manual:

Entrar con la consola de comandos en cualquier directorio que se encuentre dentro de `PATH` y ejecutar lo siguiente para descargar el archivo `composer.phar`:

```
C:\Users\username>cd C:\bin

C:\bin>php -r "eval('?>'.file_get_contents('https://getcomposer.org/installer'));"
```

A continuación, crea un nuevo archivo llamado `composer.bat` en el mismo directorio donde se encuentre `composer.phar`. Los contenidos de ese archivo `composer.bat` son los siguientes:

```
C:\bin>echo @php "%~dp0composer.phar" %*>composer.bat
```

Cierra la consola de comandos en la que se han hecho todos los cambios y prueba que todo ha salido bien ejecutando lo siguiente:

```
C:\Users\username>composer -V

Composer version 27d8904

C:\Users\username>
```

El Archivo “composer.json”

Una vez instalado Composer, ya se puede utilizar para instalar las dependencias de cualquier proyecto. Para que los comandos funcionen correctamente, es necesario crear primero un archivo llamado `composer.json` dentro del proyecto.

Las propiedades que deben ir dentro del paquete “composer.json” son:

1 La propiedad *name:*

Establece el nombre del paquete, formado por una primera parte que hace referencia a su creador y una segunda parte que describe el nombre del proyecto. Las dos partes tienen que estar separadas por el carácter /. Ejemplos:

- monolog/monolog
- igorw/event-source

Esta propiedad es obligatoria para los paquetes publicados como librerías instalables por terceros.

2 La propiedad *description.*

Se emplea para definir brevemente el propósito del paquete. Suele ser suficiente con una descripción de una sola línea.

Esta propiedad es obligatoria para los paquetes publicados como librerías instalables por terceros.

Cualquier directorio que contenga un archivo llamado composer.json, se convierte automáticamente en un paquete. Si además añades la opción require en ese archivo, estás haciendo un paquete que depende de otros paquetes. La única diferencia entre tu proyecto y una librería es que tu proyecto en realidad es un paquete sin nombre.

Para convertir tu proyecto en un paquete instalable, tienes que darle un nombre. Para ello, añade la opción `name` en el archivo `composer.json`:

```
{
    "name": "acme/hello-world",
    "require": {
        "monolog/monolog": "1.0.*"
    }
}
```

Algunos comandos básicos de Composer.

1 El comando *init*

Es un comando que puede crear el archivo composer.json solo con contestar preguntas cuyos valores se utilizan en tal archivo:

2 El comando *install*

El comando install procesa el archivo composer.json existente en ese mismo directorio, resuelve las dependencias y las instala bajo el directorio vendor/.

3 El comando *update.*

El comando update instala las versiones más recientes disponibles de tus dependencias y actualiza el contenido del archivo composer.lock.

Otros comandos son:

- ◊ **validate**: Comprueba si el archivo composer.json es válido.
- ◊ **selfupdate**: Actualiza Composer a la última versión disponible. Si la versión de Composer es demasiado antigua – 30 días o más -, se muestra un mensaje de alerta cuando se ejecuta cualquier comando.
- ◊ **search**: Busca paquetes en Packagist – por ejemplo, “composer search yaml” -.
- ◊ **show**: Muestra información sobre los paquetes. Por ejemplo, “composer show –installed” muestra un listado de los paquetes instalados.
- ◊ **create-project**: Crea un nuevo proyecto a partir de un paquete. Es bastante útil para frameworks – por ejemplo, para crear un nuevo proyecto Symfony2: “php composer.phar create-project symfony/framework-standard-edition path/ 2.4.2″.