

WEB & MOBILE APPLICATIONS

Composer

Outline

- Librerie di terze parti e problema delle dipendenze
- installazione di composer
- il funzionamento di composer



Chi sono

Stefano Bianchini

mail: stefano.bianchini@simplenetworks.it

Founder & CTO

Simplenetworks SRL

<https://www.simplenetworks.it>

<https://www.linkedin.com/in/stefanobianchini/>



Composer



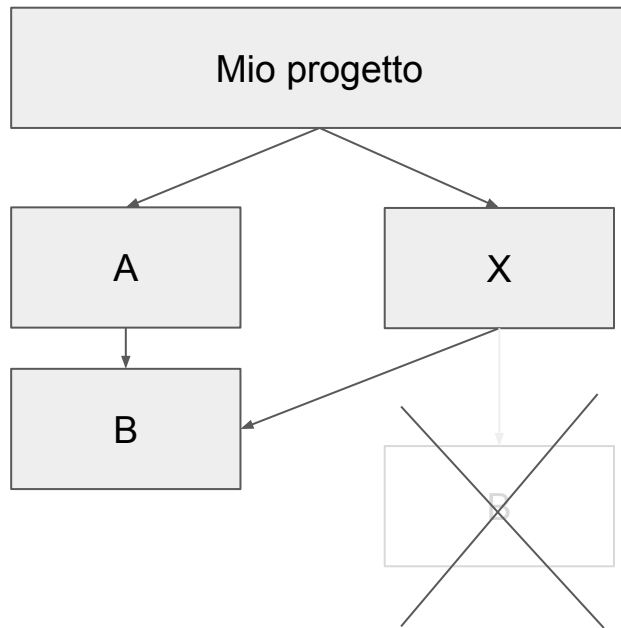
Riutilizzo del codice: librerie di terze parti

- È normale in un progetto Php utilizzare librerie / script esterni creati da altri sviluppatori (terze parti)
- Ad esempio potrei utilizzare
 - Una libreria per l'invio di email tramite SMTP
 - Una libreria per l'esportazione di dati in formato Excel
 - Una libreria per la creazione di documenti PDF
- Come gestisco l'inclusione di queste librerie nei miei progetti?
- Vecchia soluzione... **a mano!** Nuova soluzione... **composer**

Il problema delle dipendenze

- Includendo varie librerie in un progetto, ognuna di queste può a sua volta dipendere da altre funzionalità
- Se ogni singola libreria dovesse portarsi dietro (all'interno di sé stessa) queste funzionalità, ci sarebbe una ripetizione dello stesso codice e quindi uno spreco di risorse
- Ad esempio: la libreria A sfrutta una funzionalità della libreria B. La libreria X sfrutta un'altra funzionalità ma sempre della libreria B.
Potrei quindi avere installate, nel mio progetto, AB e XB dove B è presente due volte
- Le librerie potrebbero avere dipendenze a cascata (B dipende a sua volta da Z e così via)

Il problema delle dipendenze



Versioni delle librerie necessarie

- Ho bisogno di gestire delle versioni specifiche delle librerie che utilizza il mio progetto.
- Ho bisogno di condividerle con i miei colleghi, con l'ambiente di sviluppo e quello di produzione.
- Riguardando la slide precedente, potrei aver bisogno della libreria A in versione 1.0 e la libreria X in versione 2.3.1
- Sarebbe utile un sistema che, in caso di nuove versioni delle librerie, me ne permetta l'aggiornamento se non ci sono conflitti

La soluzione

- Le problematiche illustrate erano comuni a tutti gli sviluppatori Php; per questa ragione è nato **Composer**
- Composer è un gestore di dipendenze molto diffuso
- L'installazione delle dipendenze è legata al progetto (non globale)
- Tutte le dipendenze sono gestite da un unico file
- Repository centralizzato (<https://packagist.org/>) che contiene un registro di tutte le librerie scaricabili
- Permette di specificare determinate versioni di librerie

Installazione di composer

Su sistema operativo Windows esiste un comodo installer.

<https://getcomposer.org/doc/00-intro.md#installation-windows>

Sugli altri sistemi esiste una procedura di installazione

<https://getcomposer.org/download/>

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains three lines of PHP code for installing Composer.

```
php -r "copy('https://getcomposer.org/installer', 'composer-setup.php');"
php composer-setup.php
php -r "unlink('composer-setup.php');"
```

Inizializzare composer in un progetto

- posizionarsi nella cartella relativa al progetto
- eseguire composer init

```
Terminal
+ D:\dev\work\laravel\greetr\packages\simplexi\greetr>composer init
x
Welcome to the Composer config generator

This command will guide you through creating your composer.json config.

Package name (<vendor>/<name>) [ph_devpc/greetr]: simplexi/greetr
Description []: Greetr - A simple Laravel package.
Author [Francis Macugay <francis01@simplexi.com.ph>, n to skip]:
Minimum Stability []: dev
Package Type (e.g. library, project, metapackage, composer-plugin) []: project
License []: MIT

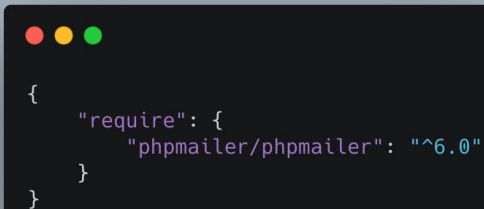
Define your dependencies.

Would you like to define your dependencies (require) interactively [yes]? no
Would you like to define your dev dependencies (require-dev) interactively [yes]? no
```

I file principali di composer

- Il file `composer.json`
 - contiene la specifica delle librerie da utilizzare
- Il file `composer.lock`
 - contiene la specifica delle librerie attualmente installate

Esempio di `composer.json`

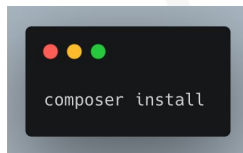


```
{  
  "require": {  
    "phpmailer/phpmailer": "^6.0"  
  }  
}
```

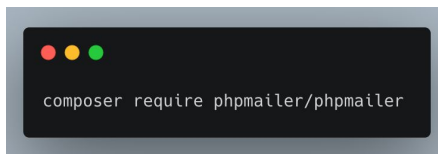
Inserimento di una dipendenza di progetto

Esistono due strade per inserire la dipendenza di una libreria (vengono chiamate package) all'interno del nostro progetto.

- Possiamo impostare a mano il file `composer.json` (o modificarlo) ed eseguire



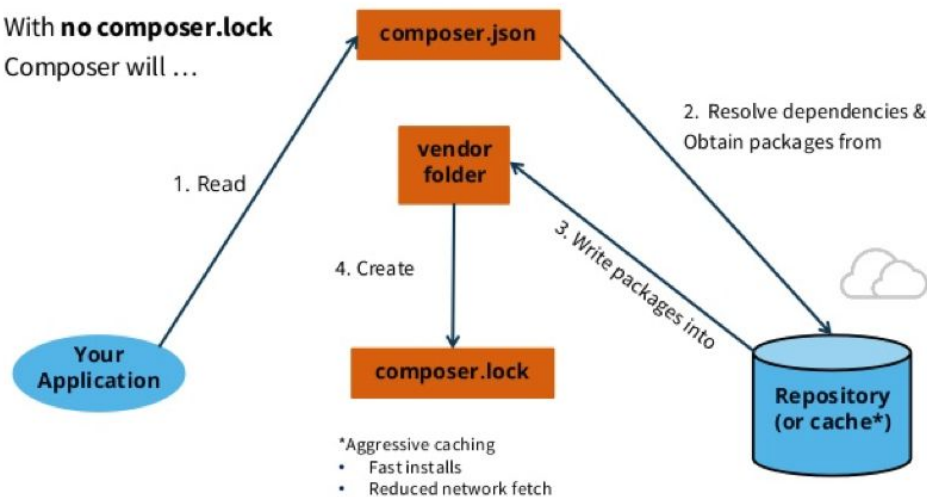
- oppure utilizzare una strada diversa utilizzando il comando **require**.
Ad esempio, per richiedere la dipendenza vista nella slide precedente



Il funzionamento di composer install

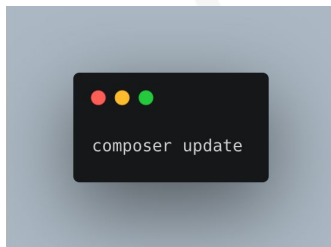
Initial **composer install**

With **no composer.lock**
Composer will ...



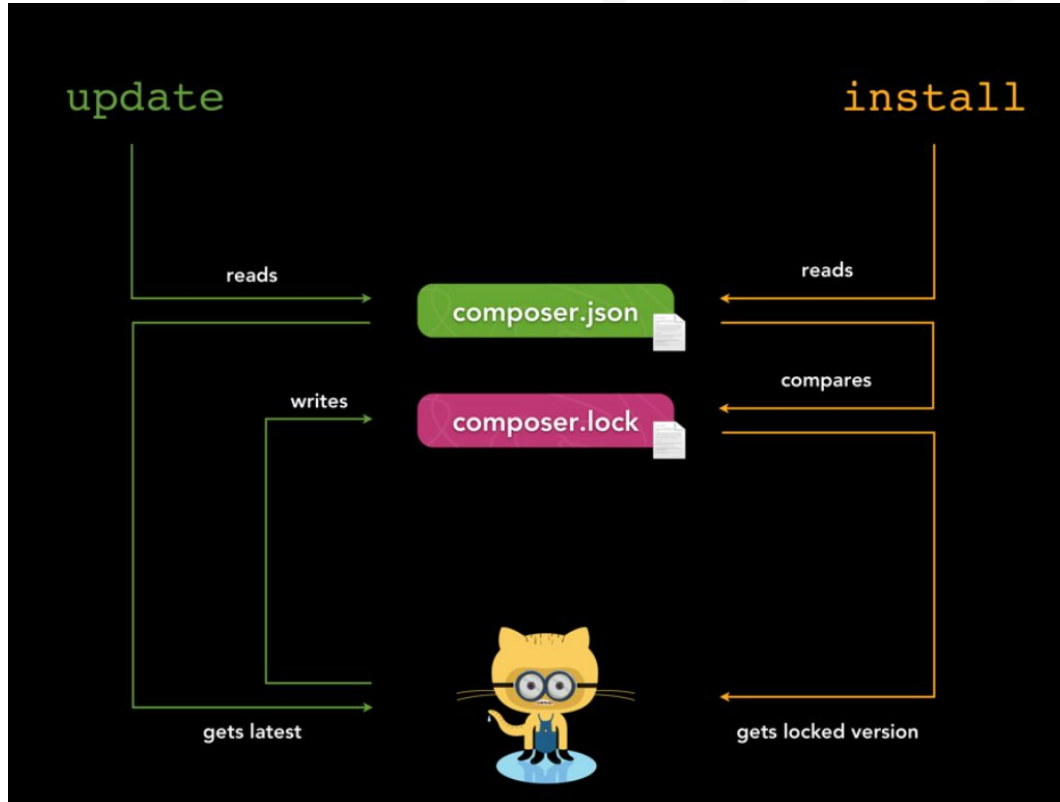
Aggiornamento dei package

Con il passare del tempo, le varie dipendenze (che solitamente sono in continua evoluzione) dovranno essere aggiornate alle nuove versioni. meno bug (in teoria), più stabilità, nuove caratteristiche



In questo caso, l'esecuzione del comando aggiorna anche il file `composer.lock`, a differenza di **composer install**

Funzionamenti differenti per install e update



Come vengono incluse le librerie scaricate?

Composer installa tutte le librerie scaricate nella cartella **vendor**.

Inoltre, genera un file chiamato **autoload.php** che si occupa del caricamento dei package necessari

Sarà l'unico file da includere negli script php del nostro progetto



```
<?php
```

```
require( 'vendor/autoload.php' );
```

Altre funzionalità di composer

- Bootstrap di un progetto
 - ovvero la possibilità di partire, ad esempio con un intero framework preinstallato nel nostro progetto
- Esecuzione di script all'occorrenza di determinati eventi
- Dipendenze installate a livello globale, ad esempio installer o compilatori, come succede per il cugino npm (gestore delle dipendenze di nodejs)

Bootstrap di un progetto

Composer ci permette anche di creare lo scheletro di un progetto specifico (se supportato) e gestirne contemporaneamente le dipendenze



```
composer create-project laravel/laravel example-app
```

Un test pratico (<https://packagist.org/packages/ramsey/uuid>)

- Posizionarsi nella cartella del progetto ed eseguire **composer init**
- Eseguire **composer require ramsey/uuid**
- Creare un file index.php con il seguente contenuto

```
<?php
//Includo le librerie importate da composer
require('vendor/autoload.php');
//Utilizzo la use per caricare la classe dal namespace corretto
use Ramsey\Uuid\Uuid;
//Eseguo una funzionalità della libreria
$uuid = Uuid::uuid4();
//Ne stampo il risultato
echo $uuid.PHP_EOL
```



Q & A