## composer笔记

PHP-FIG PHP框架可互用性小组

这个小组对PHP的未来发展做出了很多规范，有很多框架都已经遵循了这个规范

composer是PHP用来管理项目依赖的工具

依赖关系：指项目中需要使用的外部工具类或者叫组件，组件是一组打包的代码，是相关的类或接口，用于帮助我们解决PHP应用中的某个具体问题

https://getcomposer.org

安装composer前提 PHP5.3.2 开启openssl扩展

将php.exe添加到环境变量中

直接在地址栏输入cmd可以直接进入该目录的命令行

composer.phar 文件 就是composer的执行文件

运行：php composer.phar

利用composer来实现类的自动加载

安装：

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

php composer-setup.php

php -r "unlink('composer-setup.php');"

上述 3 条命令的作用依次是：

下载安装脚本 － composer-setup.php － 到当前目录。

执行安装过程。

删除安装脚本。

下载 Composer 的过程正确执行完毕后，可以将 composer.phar 文件复制到任意目录（比如项目根目录下），然后通过 php composer.phar 指令即可使用 Composer 了！

使用帮助：

Available commands:

about Short information about Composer.

archive Create an archive of this composer package.

browse Opens the package's repository URL or homepage in your browser

.

clear-cache Clears composer's internal package cache.

clearcache Clears composer's internal package cache.

config Set config options.

create-project Create new project from a package into given directory.

depends Shows which packages cause the given package to be installed.

diagnose Diagnoses the system to identify common errors.

dump-autoload Dumps the autoloader.

dumpautoload Dumps the autoloader.

exec Execute a vendored binary/script.

global Allows running commands in the global composer dir ($COMPOSER\_

HOME).

help Displays help for a command

home Opens the package's repository URL or homepage in your browser

.

info Show information about packages.

init Creates a basic composer.json file in current directory.

install Installs the project dependencies from the composer.lock file

if present, or falls back on the composer.json.

licenses Show information about licenses of dependencies.

list Lists commands

outdated Shows a list of installed packages that have updates available

, including their latest version.

prohibits Shows which packages prevent the given package from being inst

alled.

remove Removes a package from the require or require-dev.

require Adds required packages to your composer.json and installs them

.

run-script Run the scripts defined in composer.json.

search Search for packages.

self-update Updates composer.phar to the latest version.

selfupdate Updates composer.phar to the latest version.

show Show information about packages.

status Show a list of locally modified packages.

suggests Show package suggestions.

update Updates your dependencies to the latest version according to c

omposer.json, and updates the composer.lock file.

validate Validates a composer.json and composer.lock.

why Shows which packages cause the given package to be installed.

why-not Shows which packages prevent the given package from being inst

alled.

全局安装

将 composer.phar 复制到 PHP 的安装目录下面，也就是和 php.exe 在同一级目录

在 PHP 安装目录下新建一个 composer.bat 文件，并将下列代码保存到此文件中。

复制

@php "%~dp0composer.phar" %\*

有时间要简单学一下 .bat编程

经常执行 composer selfupdate 以保持 Composer 一直是最新版本

可以通过php -r "php代码" 直接在控制台执行PHP代码！！！

php -r "readfile('https://getcomposer.org/installer');" | php

解释：先读取文件内容，然后用php命令执行这个文件，其实与上面的方法差不多，只是合二为一！！！

这将检查一些 PHP 的设置，然后下载 composer.phar 到你的工作目录中。这是 Composer 的二进制文件。这是一个 PHAR 包（PHP 的归档），这是 PHP 的归档格式可以帮助用户在命令行中执行一些操作

这是一个惯例把第三方的代码到一个指定的目录 vendor。如果是 monolog 将会创建 vendor/monolog/monolog 目录。

在安装依赖后，Composer 将把安装时确切的版本号列表写入 composer.lock 文件。这将锁定改项目的特定版本。

请提交你应用程序的 composer.lock （包括 composer.json）到你的版本库中

这是非常重要的，因为 install 命令将会检查锁文件是否存在，如果存在，它将下载指定的版本（忽略 composer.json 文件中的定义）。

如果不存在 composer.lock 文件，Composer 将读取 composer.json 并创建锁文件。

此时要更新你的依赖版本请使用 update 命令。这将获取最新匹配的版本（根据你的 composer.json 文件）并将新版本更新进锁文件。

如果只想安装或更新一个依赖，你可以白名单它们：

php composer.phar update monolog/monolog [...]

如果没有composer.json 文件，在首次执行 composer require 公司/软件名

这是会自动创建composer.json 文件 并自动下载安装指定依赖，安装完后也会加锁！！！

镜像用法

方法一： 修改 composer 的全局配置文件（推荐方式）

打开命令行窗口（windows用户）或控制台（Linux、Mac 用户）并执行如下命令：

composer config -g repo.packagist composer https://packagist.phpcomposer.com

方法二： 修改当前项目的 composer.json 配置文件：

打开命令行窗口（windows用户）或控制台（Linux、Mac 用户），进入你的项目的根目录（也就是 composer.json 文件所在目录），执行如下命令：

composer config repo.packagist composer https://packagist.phpcomposer.com

手动添加：

"repositories": {

"packagist": {

"type": "composer",

"url": "https://packagist.phpcomposer.com"

}

}

注意：这个json文件全部都要使用双引号！！！

在json文件中添加：

"autoload": {

"psr-4": {"Acme\\": "src/"}

}

注意这里的psr-4必须是小写的！！！

Composer 将注册一个 PSR-4 autoloader 到 Acme 命名空间

定义一个从命名空间到目录的映射。此时 src 会在你项目的根目录，与 vendor 文件夹同级

在PHP代码中：

require 'vendor/autoload.php';

引用这个文件也将返回 autoloader 的实例

$loader = require 'vendor/autoload.php';

$loader->add('Acme\\Test\\', \_\_DIR\_\_);

只需要在控制台运行composer require laravel/laravel 即可安装到当前目录！！！而后面的名字可以到packagelist网站查找！！！

当修改了composer.json 文件就一定要重新运行 composer install 进行安装或者更新依赖，才能在项目中生效,运行composer update命令也是可以的！！！

在哪个目录下运行composer就在哪个目录下生成vendor，第三方类库！！！

PHP中 . 表示当前文件所在的目录

注意：设置命名空间是不用更新 composer.json文件的，只需要在类文件中

开头 namespace 这个文件所在的文件夹;

注意要有空行

在需要new这个类的文件中，这样写

new \t1\class();即可

如果是多层目录呢？t1目录下的t2目录下的PHP文件

namespace t1\t2;

$class2 = new \t1\Class2();

$class3 = new \t1\t2\Class3();

因为本地的路径都是用正斜线的， \\\

"require":{

"phptestsoft/json":"^1.0"

}

"仓库名":"版本号"

注意版本号这里 ^

每个composer.json 文件之间是有联系的，一个引另外一个！！

-------------------------------

//注意：这个phptestsoft对应的是在composer.json中所对应的目录值，只要可以用来表示当前目录即可

namespace phptestsoft;

-------------------------------

"psr-4":{"":"class/"}

注意：这里也是命名空间，前面什么都不写就表示class/这个目录 ！！！

注意在composer.json自定义命名空间必须 \\ss\\ 两个\\作为分隔符！！！