参考：<http://blog.csdn.net/think2me/article/details/38456931>

[Composer是干嘛的，怎么用它。(2) 使用](http://blog.csdn.net/think2me/article/details/38456931)

2014-08-09 15:34 2621人阅读 [评论](http://blog.csdn.net/think2me/article/details/38456931#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/think2me/article/details/38456931#report)

 分类：

PHP（55）  composer（1）

版权声明：本文为博主原创文章，未经博主允许不得转载。

前言：

我们在前一篇使用详细介绍了composer的下载安装，以及出现的问题和解决方案。这一章着重看composer的具体用法。

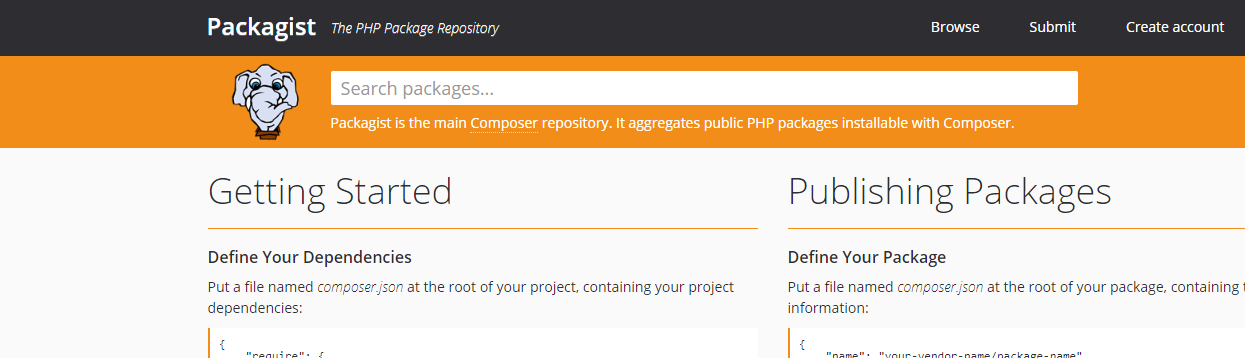
主要几点：

* composer贡献的代码网站<https://packagist.org> PHP Package Repository
* 如何下载源码
* 如何管理（跟新，删除）源码
* 如何提交自己的代码贡献给他人

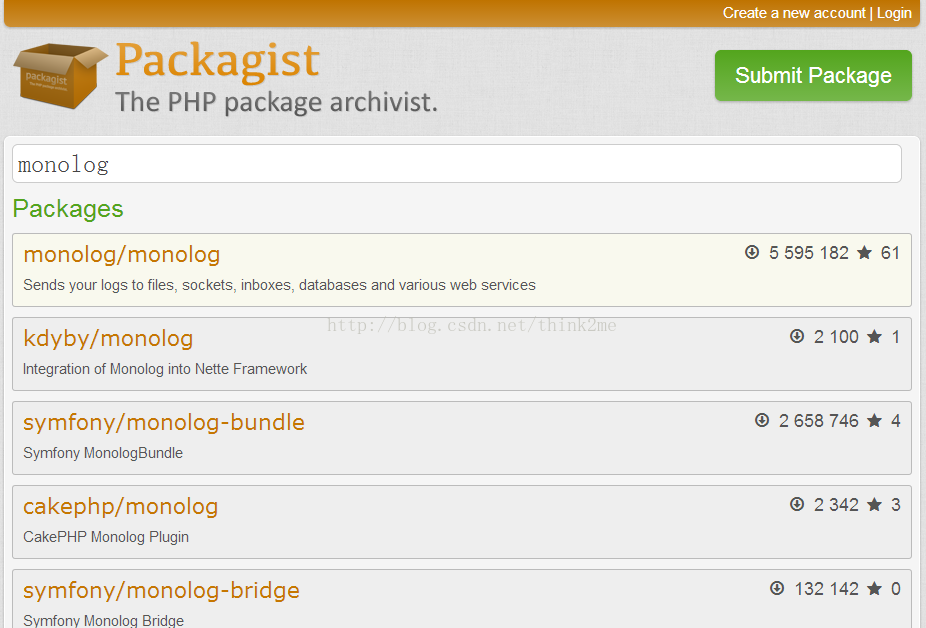
**1、我们如何寻找自己需要的项目或源码**

这个是第一步，composer网站给我们提供一个网站https://packagist.org。这个网站提供了类似npm 网站https://www.npmjs.org/ 给我们的功能，我们能在这里搜寻以及提交我们的源代码。

打开这个网站后，左边是如何安装和使用composer，第一篇我们已经讲过了，页面中间是一个搜索框，可以模糊搜索出来我们需要的项目或源码。

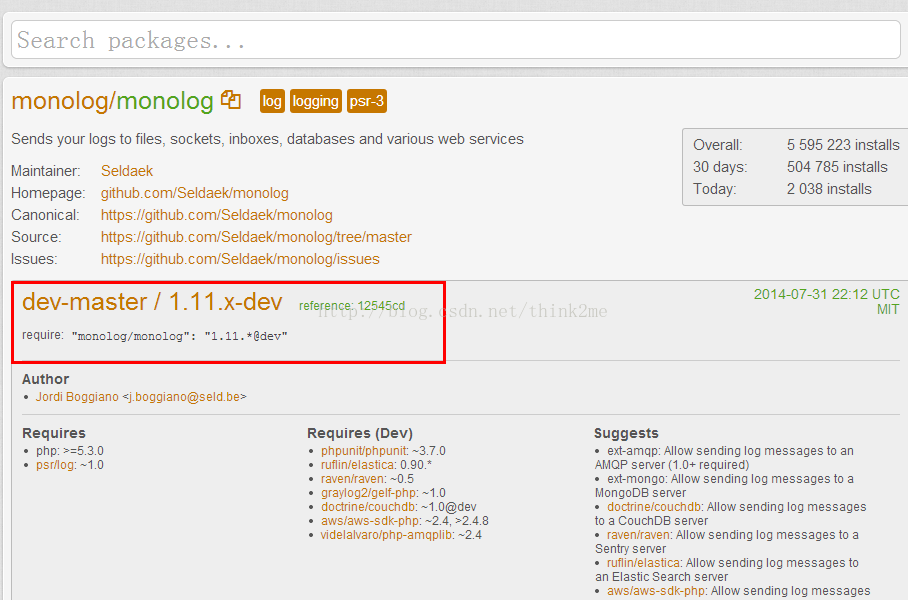


比如我们搜索monolog，是一个很有名很有用的日志记录类库，很强大。我们搜索：monolog

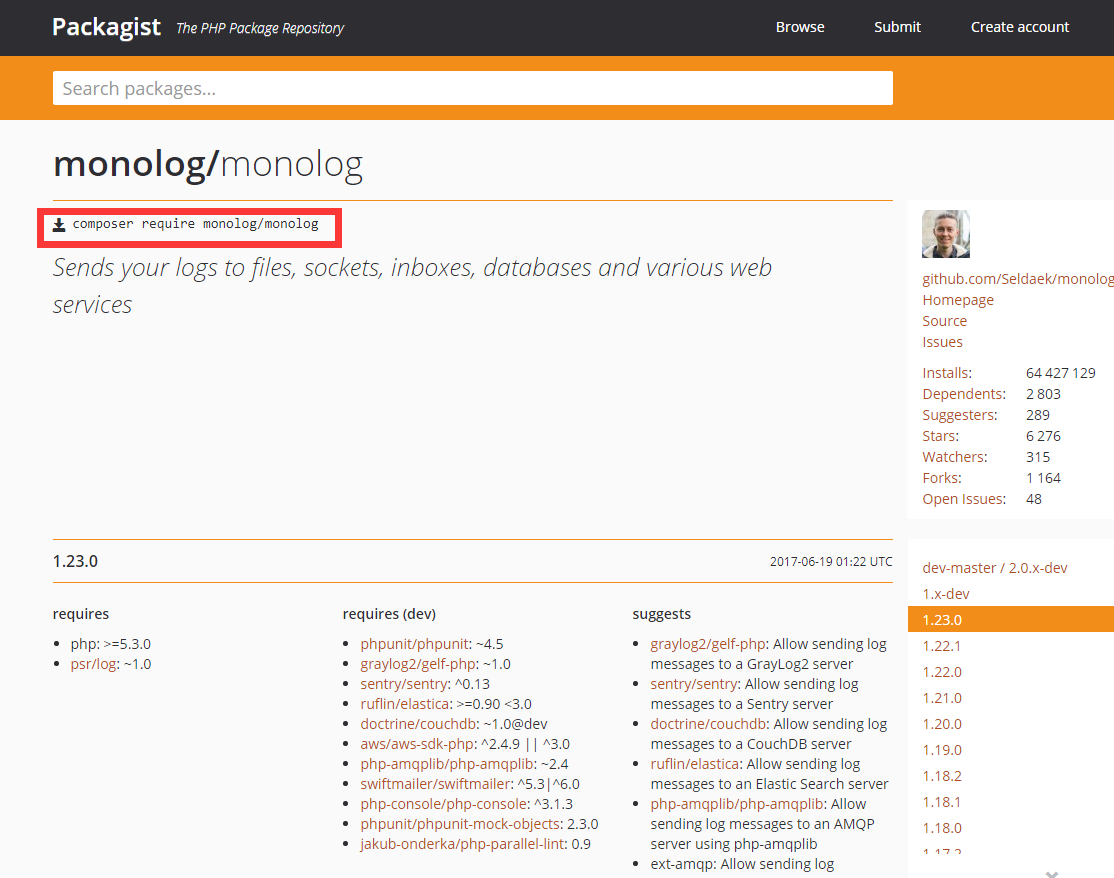


很显然第一个排名开抢，且下载量很高。所以我们就用这个了。我们点进去：<https://packagist.org/packages/monolog/monolog>

那么我们搜索到了找个类库，我们如何把它拿到自己的项目中用呢？我们看打开后的主页：



现在的网站模样：



我们主要的就是红框中的，找个是我们需要的，我们就要这个dev-master版本：

require: "monolog/monolog": "1.11.\*@dev"

**2. 使用composer.json来加载和下载类库源码**

我们已经找到了monolog的加载master路径了。那么我们怎么搞到我们的项目中的。我们需要借助于composer.json文件来帮我们，说到composer.json，我们就不得不说nodejs中的package.json，如果我们知道package.json。我们就完全能了解composer.json的作用了。它是一个代码依赖的约束文件，格式是json格式。通过这个文件我们用composer install 命令时，他会自动去下载里面的约束关系。

好。那我们把我们需要的monglog加进来，需要用require 字段：

{

"require": {

"monolog/monolog": "1.11.\*@dev"

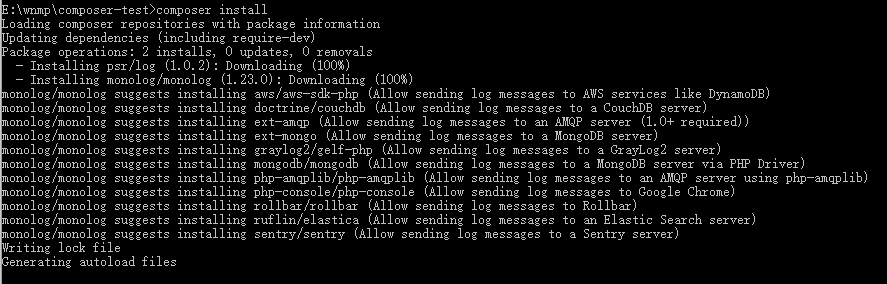
}

}

你可以看到， require 需要一个 包名称 （例如 monolog/monolog） 映射到 包版本 （例如 1.11.\*@dev） 的对象。

我们就写这一个最简单的require，我们保存composer.json。然后我们打开命令行工具，使用命令：composer install

$ composer install



命令执行完成后，就开始自动下载了，我们打开文件夹：



文件结构是：vendor文件夹+composer.lock+composer.json。我们过会说composer.lock是干嘛的有什么用。我们点击vendor进去：



里面是一个一个类库文件夹，有我们需要的mongolog，还有一个psr是在mongolog里面的composer.json里reqiure了，所以一起下载了。从这里可以看出用composer的好处了，就是一层一层的依赖关系都能加载进来，很强大很方便。

我们注意到里面有个autoload.php找个我们也是在后面来说，它是一个自动加载类库，很强大，我们只需要加载找个文件，就能自动加载使用其他的类库了，不需要再手动加载。

**3. composer.lock是干嘛的？**

我们注意到了，composer install 的时候，系统自动帮我们新建一个composer.lock文件，那这个锁文件是干嘛的呢？

我们可以打开这个文件，看看里面写的是什么。打开后，我们发现这几乎和composer.json是一样的格式和数据，也是各种约束关系，require文件等。

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。这将锁定该项目的特定版本。以后就会以这个版本为主，只要有这个文件在，就会忽略所有的composer.json文件。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conposer.lock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composer.lock。

我们想想这样有什么作用呢？假设一个场景，多人开发，肯定会有人新的人开发完成将要后续开发，那么他肯定会去更新composer.json的依赖关系，那么其他人更新后，也会同样去更新，这样就乱了。他本来不想去更新，只想用原来的。所以有个这个composer.lock 文件，所有的人都会用一样的依赖关系版本。等全部都开发好了。再将这个composer.lock文件删掉就可以了。

这意味着如果你的requie的依赖更新了新的版本，你也不会获得任何更新。因为有composer.lock文件存在，此时要想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composer.lock

如果我们确实需要更新依赖呢？我们可以使用 update 命令，它将重新生成lock文件

$ composer update

如果只想安装或更新一个依赖，你可以这样单独更新它：

$ composer update monolog/monolog [...]

**4. 包版本的规则是啥**

在前面的例子中，我们引入的 monolog 版本指定为 1.0.\*。这表示任何从 1.0 开始的开发分支，它将会匹配 1.0.0、1.0.2或者 1.0.20。

版本约束可以用几个不同的方法来指定。

| **名称** | **实例** | **描述** |
| --- | --- | --- |
| 确切的版本号 | 1.0.2 | 你可以指定包的确切版本。 |
| 范围 | >=1.0 >=1.0,<2.0>=1.0,<1.1|>=1.2 | 通过使用比较操作符可以指定有效的版本范围。  有效的运算符：>、>=、<、<=、!=。  你可以定义多个范围，用逗号隔开，这将被视为一个逻辑AND处理。一个管道符号|将作为逻辑OR处理。  AND 的优先级高于 OR。 |
| 通配符 | 1.0. | 你可以使用通配符来指定一种模式。1.0.\*与>=1.0,<1.1是等效的。 |
| 赋值运算符 | ~1.2 | 这对于遵循语义化版本号的项目非常有用。~1.2相当于>=1.2,<2.0。想要了解更多，请阅读下一小节。 |

**5. 下一个重要版本（波浪号运算符）**

~ 最好用例子来解释：

 ~1.2 相当于 >=1.2,<2.0，而 ~1.2.3 相当于 >=1.2.3,<1.3。正如你所看到的这对于遵循 语义化版本号 的项目最有用。一个常见的用法是标记你所依赖的最低版本，像 ~1.2 （允许1.2以上的任何版本，但不包括2.0）。由于理论上直到2.0应该都没有向后兼容性问题，所以效果很好。你还会看到它的另一种用法，使用 ~ 指定最低版本，但允许版本号的最后一位数字上升。

默认情况下只有稳定的发行版才会被考虑在内。如果你也想获得 RC、beta、alpha 或 dev 版本，你可以使用 稳定标志。你可以对所有的包做 最小稳定性 设置，而不是每个依赖逐一设置。

**6. 在我们自己的php项目中使用**

既然代码库都下载好了。那么我们就应该可以使用了。前面说过，在vendor文件夹下面有一个autoload.php文件，这个文件有大大的用处，它是一个自动加载的文件，我们只要加载了它，那么我们就相当于将其他的类库文件都加载了：

require 'vendor/autoload.php';

我们在对的路径加载这一句就可以了。比如，我们在加载了这一句后，想调用mongolog，我们就可以直接用了，不需要另外加载它：

$log = new Monolog\Logger('name');

$log->pushHandler(new Monolog\Handler\StreamHandler('app.log', Monolog\Logger::WARNING));

$log->addWarning('Foo');

很方便。

**7. composer的其他用法**

一般我们使用composer.json来下载第三方类库，用require 'vendor/autoload.php';使用它（me:下载后就会生成composer.lock文件）。当然composer的使用远远不止于此

我们可以直接敲入用composer命令，就会出现一个composer的命令list：

   \_\_\_\_\_\_

  / \_\_\_\_/\_\_\_  \_\_\_\_ \_\_\_  \_\_\_\_  \_\_\_\_  \_\_\_\_\_\_\_\_  \_\_\_\_\_

/ /   / \_\_ \/ \_\_ `\_\_ \/ \_\_ \/ \_\_ \/ \_\_\_/ \_ \/ \_\_\_/

/ /\_\_\_/ /\_/ / / / / / / /\_/ / /\_/ (\_\_  )  \_\_/ /

\\_\_\_\_/\\_\_\_\_/\_/ /\_/ /\_/ .\_\_\_/\\_\_\_\_/\_\_\_\_/\\_\_\_/\_/

                    /\_/

Composer version 0209bd31a0ac3aeb2a68fc81e2d03c71072bef33

Usage:

  [options] command [arguments]

Options:

  –help           -h Display this help message.

  –quiet          -q Do not output any message.

  –verbose        -v Increase verbosity of messages.

  –version        -V Display this application version.

  –ansi              Force ANSI output.

  –no-ansi           Disable ANSI output.

  –no-interaction -n Do not ask any interactive question.

  –profile           Display timing and memory usage information

  –working-dir    -d If specified, use the given directory as working directory

.

Available commands:

  about            Short information about Composer

  archive          Create an archive of this composer package

  config           Set config options

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

  depends          Shows which packages depend on the given package

  diagnose         Diagnoses the system to identify common errors.

  dump-autoload    Dumps the autoloader

  dumpautoload     Dumps the autoloader

  help             Displays help for a command

  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.

  list             Lists commands

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

m

  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

  update           Updates your dependencies to the latest version according to

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

  validate         Validates a composer.json

这里面有很多命令，我们可以一个个的尝试下看看。最常用的就是create-project命令了，我们常见的框架symfony 和 Laravel 框架就是这种方式的：

E:\wnmp>composer create-project -h

Usage:

create-project [options] [--] [<package>] [<directory>] [<version>]

composer create-project laravel/laravel your-project-name 2.1.2

composer create-project symfony/framework-standard-edition  your-project-name  2.1.2

**8. composer的一些常用的命令：**

composer list 列出所有可用的命令

composer init 初始化composer.json文件(就不劳我们自己费力创建啦)，会要求输入一些信息来描述我们当前的项目，还会要求输入依赖包

composer install 读取composer.json内容，解析依赖关系，安装依赖包到vendor目录下，默认生成verdor文件目录和composer.lock文件

composer update 更新最新的依赖关系到compsoer.lock文件，解析最新的依赖关系并且写入composer.lock文件

composer search packagename 搜索包，packagename替换为你想查找的包名称

composer require packagename 添加对packagename的依赖，packagename可修改为你想要的包名称

composer show packagename

composer self-update 更新 composer.phar文件自身

composer command --help 以上所有命令都可以添加--help选项查看帮助信息