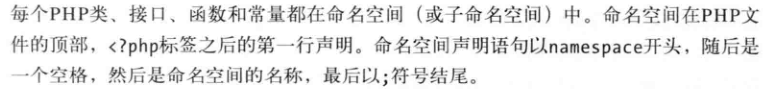
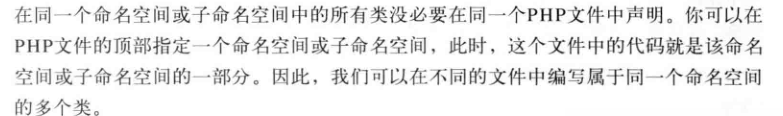
命名空间只是PHP语言的一种记号，PHP解释器会将其作为前缀加到类、接口函数和常量的名称前。

**声明命名空间**





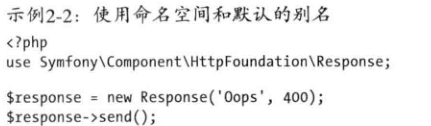
**导入类（use）**

没有使用use导入类时，我们可以这样使用其他类



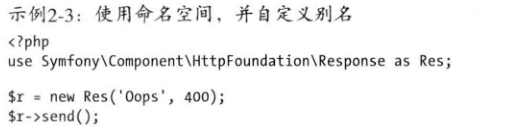
注意，此时要加上命名空间前面的\，而且每次实例化时都内写完整的命名空间

使用use 导入类



注意，使用use导入后只需要输入一次完全限定的类名，随后的实例化不用再输入完整的类名。

使用as指定类的别名，（不使用as默认别名就是类名）



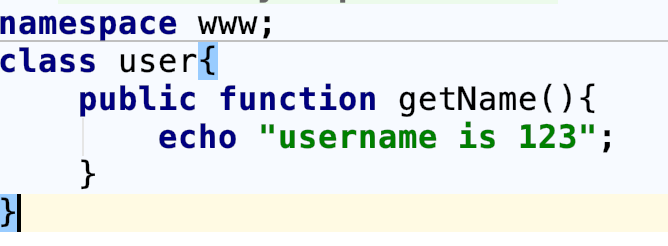
注意：应该在PHP文件顶部使用use关键字导入代码，而且要放到命名空间声明语句之后。Use关键字必须在全局作用域中，不能在类或者函数里。

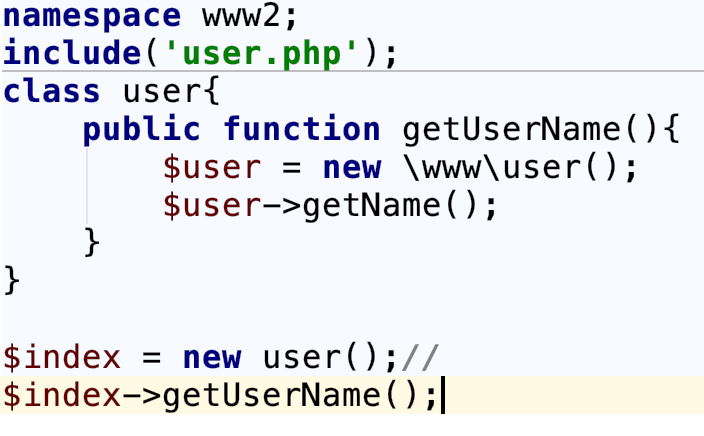
使用use关键字时无需在开头加上\符号，因为PHP假定导入的是完全限定的命名空间。

**命名空间和include()**

注意，命名空间只是为了降低不同文件重名的冲突，相当于类、常量、方法的前缀，**include()和require是引入文件，两种概念。**

例如下面的例子，相同类名，使用命名空间区分：





**全局命名空间**

有些引用的类没有命名空间，PHP会默认在当前命名空间中，此时会找不到引入的类，实例化时错误，例如PHP原生类Exception，需要在类的前面加上\前缀，这么做为了告诉PHP别在当前命名空间中查找，要在全局命名空间中查找，

<?php

namespace Test;

try {

　　throw new \Exception("抛出异常");//不加\会报错

} catch (\Exception $e) {

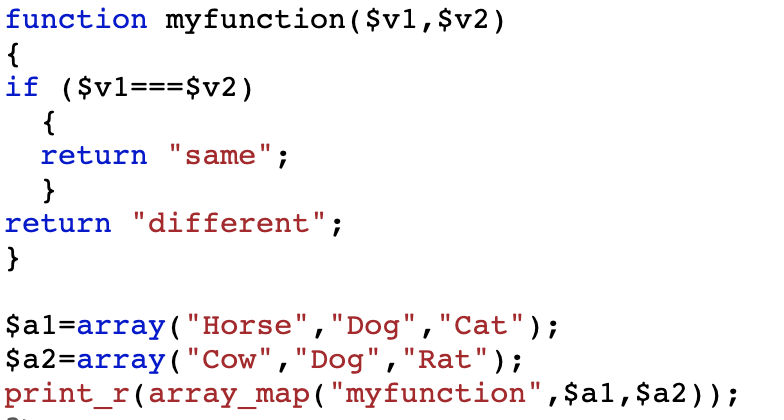
　　echo '捕获到异常'.$e->getMessage();

}

知识点：array\_map(‘myfunction’,array1,array2,array3...)

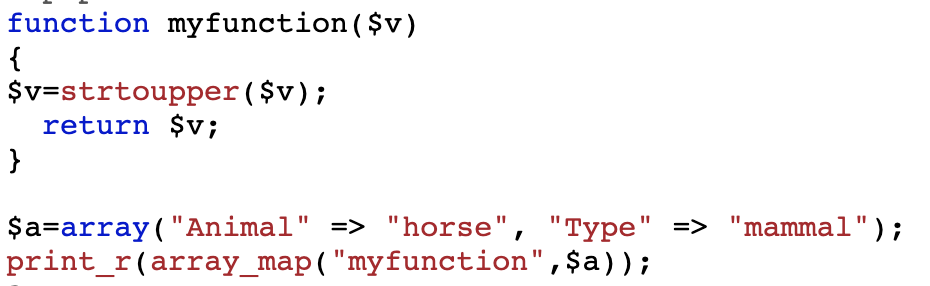
array\_map作用将数组里的每个值作为参数带入自定义函数，函数参数和数组个数对应，

例如1，比较两个数组的值



输出结果：Array ( [0] => different [1] => same [2] => different )

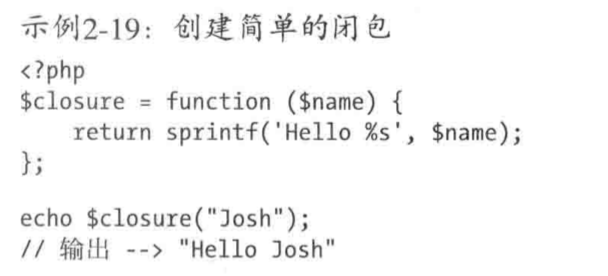
例如2:将数据值小写转大写



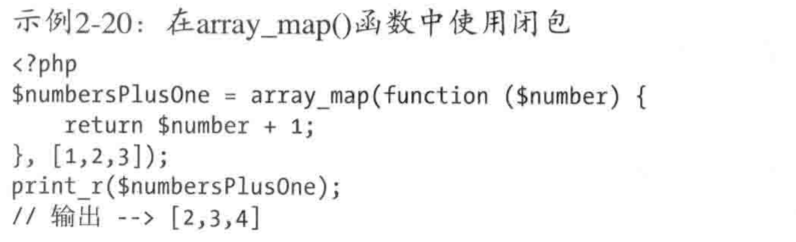
输出结果：Array ( [Animal] => HORSE [Type] => MAMMAL )

闭包

闭包和函数很像，使用一个匿名函数定义给一个变量：



通常把PHP闭包当作函数和方法的回调使用，**闭包可以作为参数传入其他PHP函数**，例如在



PHP-FIG

PHP-FIG的使命是实现框架的互操作性，通过接口、动加载机制和标准，让框架互相合作

59

**组件**

组件时打包的代码，用于帮助结局PHP应用中某个具体的问题，严格来说，组件时一系列的相关的类、接口和性状，用于解决某个具体问题。组件中的类、接口、性状通常放在同一个命名空间中。

组件特征：

作用单一，能很好的专注解决一个问题

小型，只包含解决某个问题所需的最少代码，可以只有一个PHP类，也可以有多个类，分别放在不同的子命名空间中。

合作，组件把代码放到自己的命名空间，防止与其他组件由名称的冲突

流行的PHP框架：Aura Laravel Symfony Yii Zend

使用组件

Packagist是查找组件的地方，Composer是安装PHP组件的工具。Composer是PHP组件的依赖管理器，运行在命令行中。Composer会从Packagist下载需要的PHP组件并自动加载到项目中。Composer是依赖管理器，因此还能解析并下载组件的依赖（以及依赖的依赖，等等）

自动加载是指，在不使用require()require\_once() include()或者include\_once()函数的情况下，按需自动加载PHP类。旧版本PHP会使用\_\_autoload(),新版本引入spl\_autoload\_register()函数。Composer会为项目中所有PHP组件自动生成符合PSR标准的自动加载器。

**安装Composer**

打开终端，执行下面命令

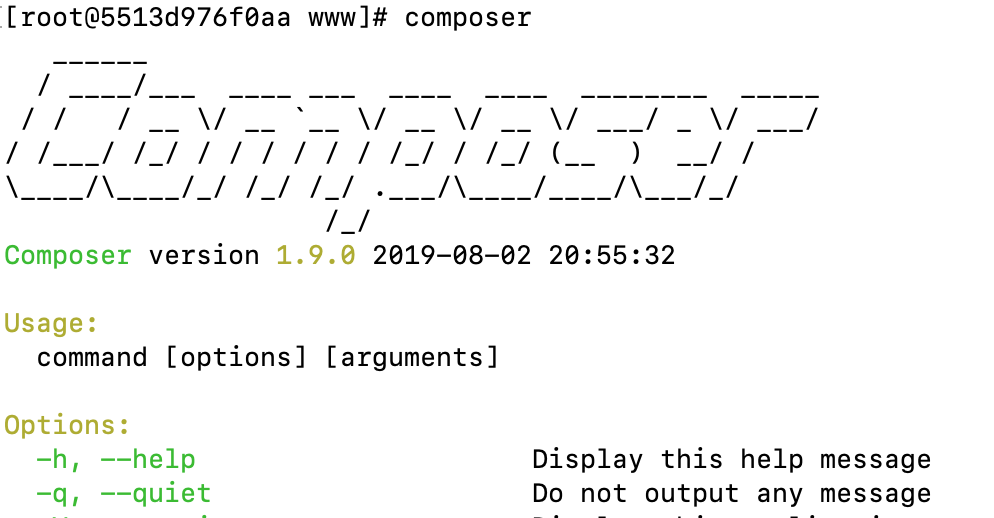
curl -sS https://getcomposer.org/installer | php

这个命令使用curl下载Composer的安装脚本，然后使用PHP执行安装脚本，最后在当前工作目录中创建composer.phar文件（Composer二进制文件）

注意：如果你把它放在系统的 PATH 目录中，你就能在全局访问它。

mv composer.phar /usr/local/bin/composer

可以输出Composer命令，查看Composer的选项列表：



使用Composer安装组件

可以去<https://packagist.org/> 查找要安装的组件，注意每个组件都由厂商和包名组成，例如league/flysystem，厂商是league包名是flysystem,中间反斜杠/分割，厂商名是全局唯一的，这是全局标识符，用于识别名下的包属于谁，避免PHP组件名称冲突。

每个组件可能会有多个版本，Composer会默认选择最新稳定的版本号。

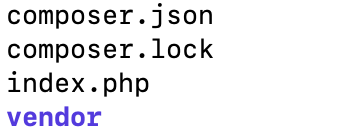
在终端中进入项目的最顶层目录，然后为所需的每个PHP组件执行一次下列命令：

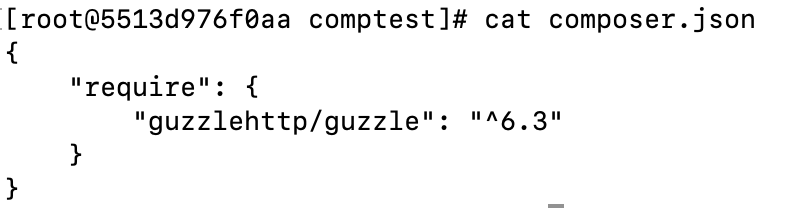
composer require league/flysystem

composer require guzzlehttp/guzzle

注意：这个命令还可以更新组件的版本（最新稳定版）

安装命令完后，会新建下列文件（除index.php）

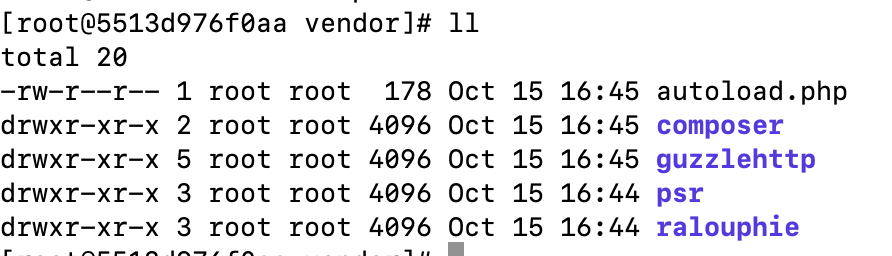




Composer.lock文件：这个文件会列出项目使用的所有PHP组件，以及组件的具体版本号（主版本号、此版本号和修订版本号）。这其实是锁定了项目，让项目只能使用具体版本的PHP组件，Composer只能下载这个文件列出的版本号组件。

注意，这个文件需要版本控制，保证其他人的组件版本一致

Vendor目录：组件下载到该目录下



**自动加载组件**

Composer在下载组件时会为了项目的所有依赖创建一个符合PSR标准的自动加载器(其实就是vendor/autoload.php文件)，我们只需在文件顶部使用require函数导入Composer创建的自动加载器：

<?php

require "vendor/autoload.php";

$client = new \GuzzleHttp\Client();

$url = "http://www.baidu.com";

$httpRequest = $client->options($url);

echo $httpRequest->getStatusCode();