PHP 包： <http://pecl.php.net/package-stats.php>

PHP 官网地址为：<https://www.php.net/>

PHPChina 开发者社区：<http://www.phpchina.com/>

PHP在线中文手册：<https://www.php.net/manual/zh/>

PHP代码规范：<http://laravel-china.github.io/php-the-right-way/>

# 基础知识

## PHP介绍

PHP官网：<https://www.php.net/>

在线英文文档：<https://www.php.net/docs.php>

在线中文文档：<https://www.php.net/manual/zh/>

下载chm格式的离线PHP手册。

### PHP是什么

Hypertext Preprocessor：超文本预处理器。

### PHP语言优势

### PHP7新特性

### PHP发展趋势

### PHP应用领域

### PHP学习资源

## 环境搭建

PHP环境搭建包括PHP运行环境的搭建以及PHP开发工具的搭建。

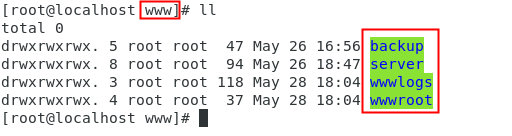
在CentOS系统中，宝塔安装命令：

|  |
| --- |
| yum install -y wget && wget -O install.sh http://download.bt.cn/install/install\_6.0.sh && sh install.sh |

宝塔安装后所在目录地址为：

|  |
| --- |
| /www |

如下图所示：



目录说明：

|  |  |
| --- | --- |
| backup | 备份目录 |
| server | PHP、Apache、Nginx、MySQL等组件目录 |
| wwwlogs | 网站访问日志 |
| wwwroot | 放置网站程序的目录 |

### 运行环境

我们在安装CentOS虚拟机，然后在虚拟机里面安装宝塔面板和Git服务软件。

通过Git服务可以拉取和提交在Windows中编写的代码，并通过宝塔面板来运行PHP代码。

当前，我们**宝塔面板**安装的相关软件为：

|  |  |
| --- | --- |
| 软件名称 | 版本号 |
| Nginx | 1.16.1 |
| MySQL | 8.0.20 |
| PHP | 7.4 |

我们需要在服务器中创建名称为PHPStudy的Git空仓库，然后拉取到Windows本地的工作目录，然后在使用IntelliJ IDEA来打开该项目，并通过IntelliJ IDEA来编写和提交代码文件到PHPStudy仓库。

### 开发工具

我们在Windows系统中使用IntelliJ IDEA做为PHP开发环境。

IntelliJ IDEA下载地址为：<https://www.jetbrains.com/idea/download/>

### 搭建总结

之所以采用这种分开方式，是因为我们最终的代码仍然是要运行在CentOS服务器上的。而分开方式可以保持程序和运行环境的一致性。

## PHPUnit单元测试组件

### PHPUnit相关网址

PHPUnit官网：<https://phpunit.de/>

中文官网：<http://www.phpunit.cn/>

在线中文手册：<http://www.phpunit.cn/documentation.html>

中文入门文档 ：<http://www.phpunit.cn/getting-started.html>

### PHPUnit组件安装

打开网站根目录下的composer.json文件，在文件中增加如下代码：

|  |
| --- |
| "require-dev": {  "phpunit/phpunit": "\*"  }, |

然后在网站根目录下执行如下命令：

|  |
| --- |
| composer update |

这样composer就会在网站中安装PHPUnit组件了。

然后在网站根目录下添加phpunit.xml文件，在文件中添加如下代码：

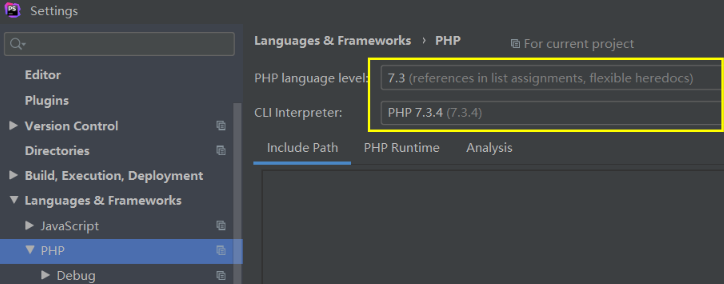
|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <phpunit backupGlobals="false"  backupStaticAttributes="false"  bootstrap="vendor/autoload.php"  colors="true"  convertErrorsToExceptions="true"  convertNoticesToExceptions="true"  convertWarningsToExceptions="true"  processIsolation="false"  stopOnFailure="false">  <testsuites>  <testsuite name="Unit">  <directory suffix="Test.php">./tests/Unit</directory>  </testsuite>  <testsuite name="Feature">  <directory suffix="Test.php">./tests/Feature</directory>  </testsuite>  </testsuites>  <filter>  <whitelist processUncoveredFilesFromWhitelist="true">  <directory suffix=".php">./app</directory>  </whitelist>  </filter>  <php>  <server name="APP\_ENV" value="testing"/>  <server name="BCRYPT\_ROUNDS" value="4"/>  <server name="CACHE\_DRIVER" value="array"/>  <server name="MAIL\_DRIVER" value="array"/>  <server name="QUEUE\_CONNECTION" value="sync"/>  <server name="SESSION\_DRIVER" value="array"/>  </php>  </phpunit> |

配置文件的代码基本上不需要做修改。

### PHPStorm 配置 PHPUnit 组件

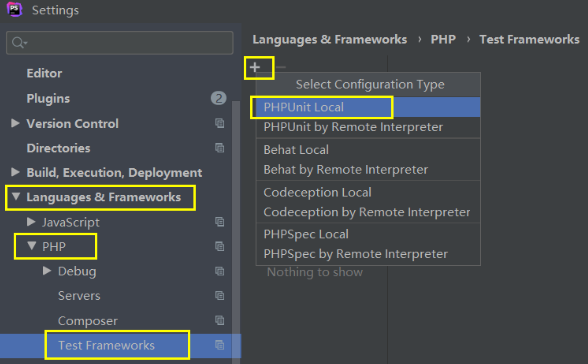
#### 配置 PHP 路径

PHPStorm → File → Settings... → Languages & Frameworks → PHP：

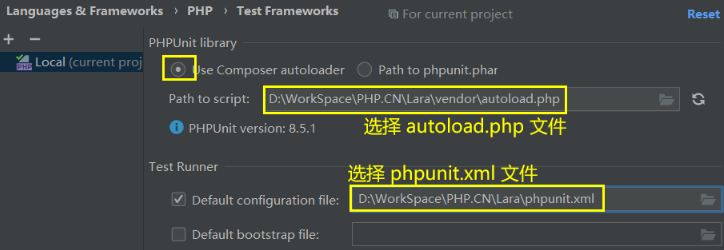


#### 配置 Test Frameworks

PHPStorm → File → Settings... → Languages & Frameworks → PHP → Test Frameworks：



配置项填写如下图所示：



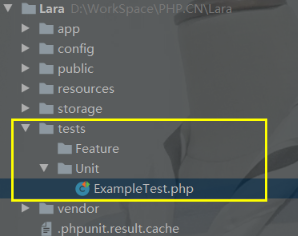
然后 → Apply → OK 即可。

#### 编写单元测试案例

在项目中添加如下目录结构：

|  |
| --- |
| tests\Unit  tests\Feature |

如下图所示：



然后在 Unit 目录下创建文件 ExampleTest.php，代码如下：

|  |
| --- |
| <?php  # 命名空间  namespace Tests\Unit;  # TestCase  use PHPUnit\Framework\TestCase;  class ExampleTest extends TestCase  {  public function testBasicExample()  {  $this->assertTrue(true);  }  public function testPushAndPop()  {  $stack = [];  $this->assertEquals(0, count($stack));  array\_push($stack, 'foo');  $this->assertEquals('foo', $stack[count($stack) - 1]);  $this->assertEquals(1, count($stack));  $this->assertEquals('foo', array\_pop($stack));  $this->assertEquals(0, count($stack));  }  } |

然后就可以执行单元测试代码了。

## PHP调试工具 Xdebug

PHP调试环境的配置过程。这里我们采用如下组合来实现：

|  |
| --- |
| PHP7.3.4 + IntelliJ IDEA + Xdebug + Chrome |

### 公共配置

#### 不使用集成环境的 php\_xdebug 插件

在这里使用的是小皮 phpstudy 集成环境软件，不要在该集成环境中开启 php 扩展的 php\_xdebug ，也就是不要 √ 上；同时，对应 PHP 版本的 php.in 中，注释掉 php\_xdebug 插件，代码如下：

|  |
| --- |
| ;extension=php\_xdebug |

因为下面我们要使用自己下载和安装的 php\_xdebug 扩展。

#### 分析 Xdebug 插件的版本和安装步骤

1、打开 https://xdebug.org/wizard 网站；

2、本地页面执行 phpinfo() 函数；

3、本地页面选择全部并复制，然后粘贴到 https://xdebug.org/wizard 网站的输入框；

4、点击 Analyse my phpinfo() output 按钮

5、然后就可以看到本地 PHP 环境对应的版本信息以及需要安装的 Xdebug 具体版本和步骤；

#### 在 PHP 环境中安装 Xdebug 插件

1、下载 Xdebug 插件 php\_xdebug-2.9.0-7.3-vc15-nts-x86\_64.dll；

2、把下载来的 Xdebug 插件移动到 C:\PHPStudy\Extensions\php\php7.3.4nts\ext 目录中；

3、编辑 C:\PHPStudy\Extensions\php\php7.3.4nts\php.ini 文件，添加如下代码：

|  |
| --- |
| ; Local Variables:  ; tab-width: 4  ; End:  [xdebug]  zend\_extension = C:\PHPStudy\Extensions\php\php7.3.4nts\ext\php\_xdebug-2.9.0-7.3-vc15-nts-x86\_64.dll  xdebug.remote\_autostart = 1  xdebug.default\_enable = 1  ;配置端口和监听的域名  xdebug.remote\_port = 9000  xdebug.remote\_host = 127.0.0.1  xdebug.remote\_connect\_back = 1  xdebug.remote\_enable = 1  xdebug.idekey=PHPSTORM  ;指定性能分析文件的存放目录  xdebug.profiler\_output\_dir = "D:/php/tmp"  xdebug.profiler\_enable\_trigger = 1  xdebug.profiler\_output\_name = cachegrind.out.%t.%p |

#### 在 PHPStorm 中进行 Xdebug 的配置

配置 PHP 版本：

File → Settings... → Languages & Frameworks → PHP：

PHP language level: 7.3

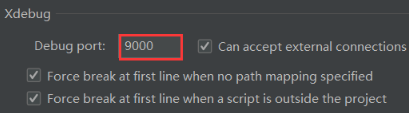
CLI Interpreter: PHP 7.3.4

本配置项，本质上不影响使用 Xdebug 功能，所以可以不配置。

配置 Debug：

File → Settings... → Languages & Frameworks → PHP → Debug：

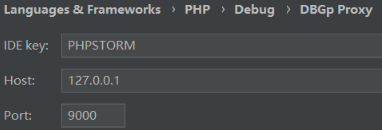
配置 Xdebug 信息



注意：Debug port 的值需要与 php.in 中的 xdebug.remote\_port 的值保持一致。

配置 DBGp Proxy：

File → Settings... → Languages & Frameworks → PHP → Debug → DBGp Proxy，配置如下：



注意：IDE key、Host、Port 与 php.in 中的 xdebug.idekey、xdebug.remote\_host、xdebug.remote\_port 保持一致。

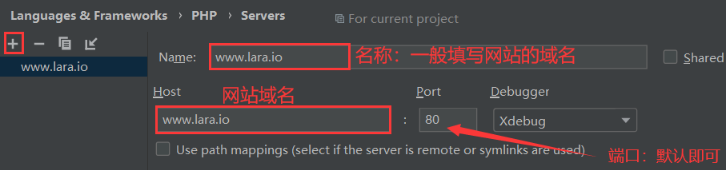
### 个性配置

这里的个性配置是指，除了以上的公共配置之外，需要给每个独立的网站做配置。当然，每个网站的配置项是一样的，只是配置的值不同而已。

这里，我们以 www.lara.io 网站为例子来学习配置每个网站使用 Xdebug 功能的过程。

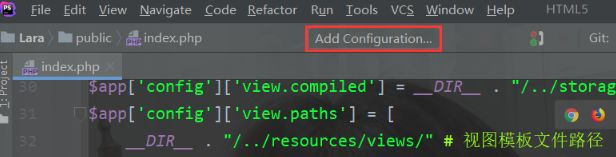
#### 网站配置 Servers

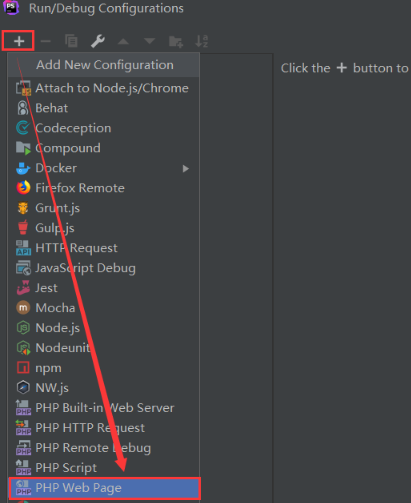
File → Settings... → Languages & Frameworks → PHP → Servers：



#### 网站配置 Add Configuration...

菜单栏 → Run → Edit Configurations... 或者直接通过如下截图的 Add Configuration... 进入配置。

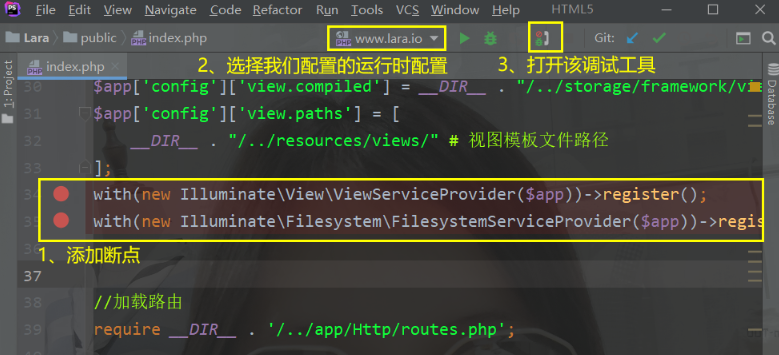




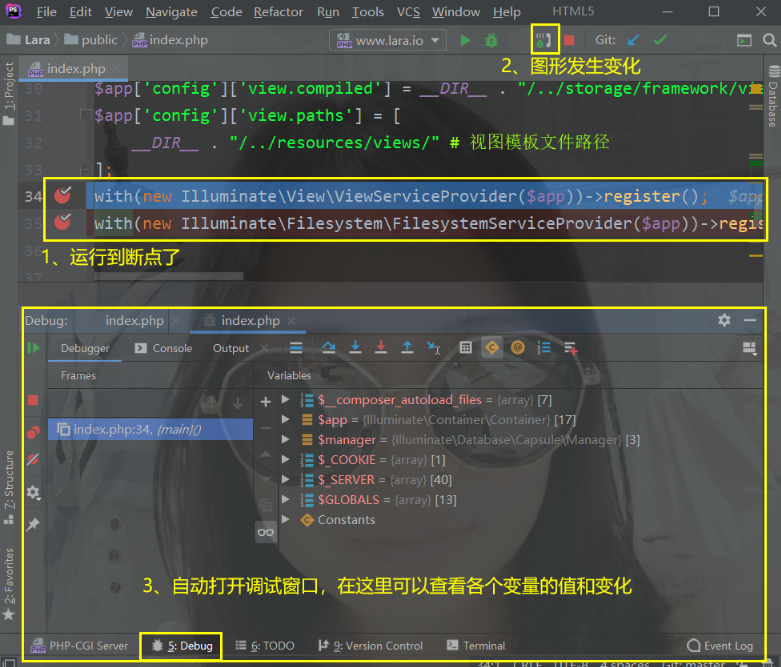
在页面中，按照如下方式一一填写并保存即可。



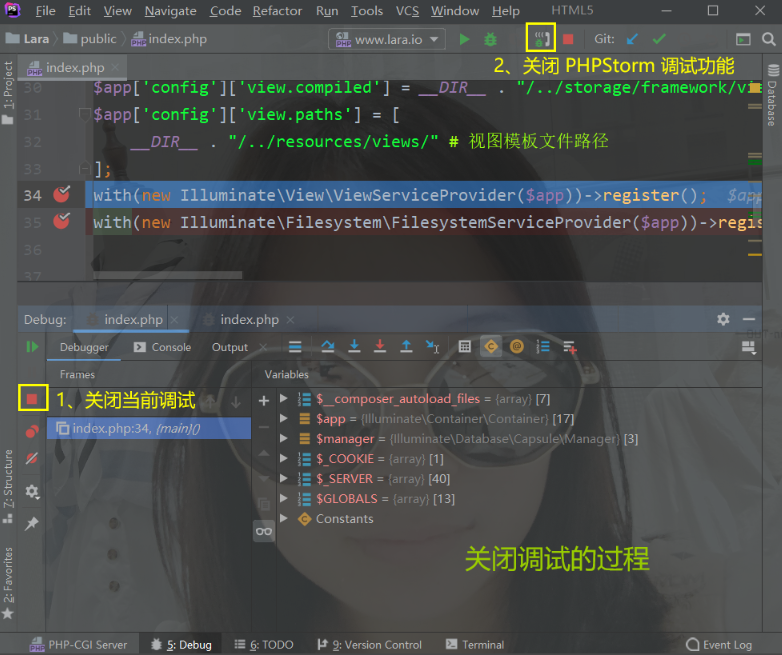
#### 代码添加断点



在浏览器输入：http://www.lara.io/ 此时 PHPStorm 发生如下变化：请求 url 的时候就会自动捕捉到请求。



关闭调试：



### Chrome 安装 Xdebug help 插件

下载地址:

<https://chrome.google.com/webstore/detail/xdebug-helper/eadndfjplgieldjbigjakmdgkmoaaaoc>

或者通过扩展迷进行下载:

<https://www.extfans.com/chrome-app/eadndfjplgieldjbigjakmdgkmoaaaoc/>

右键选择 Xdebug → 选项 → IDE key → PhpStorm → Save

参考资料:

<https://blog.csdn.net/qq_42249896/article/details/88056260>

## PHP文件结构

一般情况下，PHP 以成对的开始和结束标签来包裹 PHP 代码，如下所示：

|  |
| --- |
| <?php  // 此时可以包含 HTML 代码  ?> |

如果没有结束标签呢？则在代码中不能包含 HTML 代码：

|  |
| --- |
| <?php  // 此时不可以包含 HTML 代码，  // 必须是纯 PHP 代码，才可以不带结束标签 |

## PHP标记格式

* XML风格

|  |
| --- |
| <?php  echo "这是XML风格的标记。";  ?> |

推荐使用的标记风格。

* 脚本风格

|  |
| --- |
| <script language="php">  echo "这个是脚本风格的标记。";  </script> |

* 简短风格

|  |
| --- |
| <? echo "这个是简短风格的标记";?> |

* ASP风格

|  |
| --- |
| <%  echo "这是ASP风格的标记";  %> |

如果要使用简短风格和ASP风格，需要在php.ini配置文件中，把short\_open\_tag和asp\_tags设置为on值，并重启Nginx服务器。

## PHP注释语句

单行注释（//）

|  |
| --- |
| <?php  // 这是写在PHP语句上方的单行注释  echo "使用C++风格的单行注释";  echo "使用C++风格的单行注释"; // 这是写在PHP语句后方的单行注释  ?> |

多行注释（/\*…\*/）

|  |
| --- |
| <?php  /\*  多行注释第一行  多行注释第二行  多行注释第三行  \*/  echo "PHP的多行注释";  ?> |

文档注释

|  |
| --- |
| <?php  /\*  \* 类说明……  \* 作者……  \*/  class MyUtil  {  /\*  \* 方法说明……  \* 参数……  \* 返回值……  \*/  function addPreFix()  {  }  }  ?> |

注意，多行注释和文档注释是不允许嵌套的。

## PHP数据类型

### 数据类型种类

PHP中的数据类型包括8种。

|  |  |  |
| --- | --- | --- |
| 标量类型 | boolean（布尔型） | 包含true（真）和false（假） |
| integer（整型） | 包含正整数、0、负整数 |
| float/double（浮点型） | 包含小数点的小数 |
| string（字符串型） | 连续的字符序列 |
| 复合类型 | array（数组） | 用来保存具有相同数据类型的多个数据项 |
| object（对象） | 保存类的实例 |
| 特殊类型 | resource（资源） | 用来保存到外部资源的一个引用。 比如：打开文件、数据库连接、图形画布区域等。 |
| NULL（空） | 没有赋值的变量； 被重置的变量； 赋值为NULL的变量； |

由于在定义变量时，不需要指定变量的数据类型，因此，PHP的变量的数据类型是由赋给

该变量的值的类型决定的。

示例代码：

|  |
| --- |
| <?php  $name = "Orange"; //字符串  $gender = "男"; //字符串  $age = 18;//整型  $height = 180.56;//浮点型  $weight = 50.35;//浮点型  echo "姓名：" . $name . "<br/>";  echo "性别：" . $gender . "<br/>";  echo "年龄：" . $age . "<br/>";  echo "身高：" . $height . "厘米<br/>";  echo "体重：" . $weight . "公斤<br/>"; |

### 数据类型转换

#### 强制类型转换

在PHP中，数据类型转换非常简单，只需要在变量前面加上（），然后在（）里面写上数据类型即可。

|  |  |  |
| --- | --- | --- |
| **PHP中允许转换的数据类型** | | |
| **转换操作** | **转换类型** | **代码示例** |
| (int), (integer) | 转换成整型 | (int)$bool,(integer)$str |
| (bool),(boolean) | 转换成布尔 | (bool)$num,(boolean)$str |
| (string) | 转换成字符串 | (string)$bool |
| (array) | 转换成数组 | (array)$str |
| (float), (double), (real) | 转换成浮点型 | (float)$str,(double)$str |
| (object) | 转换成对象 | (object)$str |
| (unset) | 转换为NULL | (unset)$str |

在数据转换成boolean布尔型时，会发生以下情况：

|  |  |
| --- | --- |
| NULL | false |
| 0 | false |
| 未赋值的变量 | false |
| 未赋值的数组 | false |

其它转换都是true。

在数据转换成int整型时，会发生以下情况：

|  |  |
| --- | --- |
| false | 0 |
| true | 1 |
| 浮点型 | 舍去小数部分 |
| 字符串 | 如果以数字开头，则截取到非数字位，否则输出0 |

#### 使用转换函数

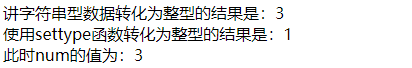
我们还可以通过settype()函数实现数据类型转换。该函数定义如下：

|  |
| --- |
| bool settype(mixed &$var, string $type) |
| 参数var为指定的变量  参数type为指定的类型 |
| type的可选值有7个：boolean、integer、float、string、array、object、null。  也就是说在8种数据类型当中，除了resource资源外，其它7种都可以。 |
| 转换成功则返回true，转换失败则返回false。 |

**字符串转换成整型或浮点型时，直接取到非数字位为止**。如下示例：

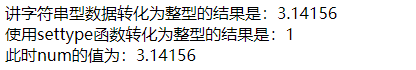
|  |
| --- |
| 字符串转换成整型 |
| <?php  $num = "3.14156后面跟上字符串";  echo "讲字符串型数据转化为整型的结果是：";  echo (integer)$num;  echo "<br/>";  $result = settype($num, "integer");  echo "使用settype函数转化为整型的结果是：" . $result;  echo "<br/>";  echo "此时num的值为：" . $num; |

结果如下图：



|  |
| --- |
| 字符串转换成浮点型 |
| <?php  $num = "3.14156后面跟上字符串";  echo "讲字符串型数据转化为整型的结果是：";  echo (double)$num;  echo "<br/>";  $result = settype($num, "double");  echo "使用settype函数转化为整型的结果是：" . $result;  echo "<br/>";  echo "此时num的值为：" . $num; |

结果如下图：



通过观察可以知道，在使用强制数据类型转换时，原数据不发生变化，而使用settype进行数据类型转换时，原数据发生了变化。

既然有设置变量数据类型的函数，自然也就有对应的获取变量数据类型的函数了，该函数就是gettype()。

#### 四舍五入操作

如果是**浮点型要转换成整型**，除了以上的强制类型转换、使用settype函数转换之外，还可以通过四舍五入相关函数来实现转换。

|  |  |  |
| --- | --- | --- |
| intval() | 直接取整，舍弃小数保留整数 | 直接取整 |
| round() | 四舍五入取整，也可以设置要保留的小数位数 | 四舍五入取整 |
| ceil() | 有小数，则在整数的基础上进行 +1 操作 | 进一取整 |
| floor() | 有小数，则舍去小数 | 舍去取整 |

### 数据类型检测

除了不能检测是否为resource资源之外，其它都可以。

|  |  |
| --- | --- |
| **函数** | **检测类型** |
| is\_bool() | 是否为布尔型 |
| is\_integer/is\_int | 是否为整型 |
| is\_float/is\_double | 是否为浮点型 |
| is\_string | 是否为字符串类型 |
| is\_array | 是否为数组 |
| is\_object | 是否为对象 |
| is\_null | 是否为空 |
| is\_numeric | 是否为数字或数字组成的字符串 |

示例代码：

|  |
| --- |
| <?php  $is\_true = true;  $name = 'orange';  $age = 25;  $height = 26.9;  $width = '29.4';  $mixnum = '123abc';  echo '判断 true 是否是布尔类型' . '<br/>';  echo '判断结果为: ' . is\_bool($is\_true);  echo '<br/>';  echo '判断 25 是否为字符串类型' . '<br/>';  echo '判断结果为: ' . is\_string($age);  echo '<br/>';  echo "判断 '29.4' 是否为浮点类型" . '<br/>';  echo '判断结果为: ' . is\_float($width);  echo '<br/>';  echo "判断 '29.4' 是否为数字或数字字符串类型" . '<br/>';  echo '判断结果为: ' . is\_numeric($width); |

运行结果如下所示：

|  |
| --- |
| 判断 true 是否是布尔类型  判断结果为: 1  判断 25 是否为字符串类型  判断结果为:  判断 '29.4' 是否为浮点类型  判断结果为:  判断 '29.4' 是否为数字或数字字符串类型  判断结果为: 1 |

## PHP常量

常量标识符一般总是大写的。

常量定义与变量不同，定义变量时，需要以$开头，而定义常量时则不需要$符号。

### 自定义常量

#### 使用define()函数

在PHP中使用define()函数来自定义常量。

|  |
| --- |
| function define ($name, $value, $case\_insensitive = false) {} |
| $name 必选参数，常量名。  $value 必选参数，常量值。  $case\_insensitive 可选参数，是否大小写敏感，默认false表示敏感。 |

定义完常量之后，使用常量名可以直接获取常量值，不需要加$符号。

示例代码：

|  |
| --- |
| <?php  define('NAME', '我是一个PHPer程序员');  echo NAME; |
| 我是一个PHPer程序员 |

#### 使用const定义

示例代码：

|  |
| --- |
| <?php  define('NAME', '我是一个 define() 函数定义的常量，定义和使用时都不需要加 $ 符号');  echo '<p>' . NAME . '</p>';  const name = '我是一个 const 定义的常量，定义和使用时都不需要加 $ 符号';  echo '<p>' . name . '</p>'; |
| 我是一个 define() 函数定义的常量，定义和使用时都不需要加 $ 符号  我是一个 const 定义的常量，定义和使用时都不需要加 $ 符号 |

#### 两种方式的区别

### 预定义常量

|  |  |
| --- | --- |
| PHP常用的预定义常量 | |
| PHP\_OS | PHP运行所在的操作系统 |
| PHP\_VERSION | PHP版本号 |
| PHP\_INT\_MIN | PHP整数的最小值 |
| PHP\_INT\_MAX | PHP整数的最大值 |
| TRUE | 该常量表示一个真值true |
| FALSE | 该常量表示一个假值false |
| M\_PI | 圆周率 |
| NULL | 一个null值 |

### 魔术常量

|  |  |
| --- | --- |
| PHP中的魔术常量 | |
| \_\_NAMESPACE\_\_ | 当前命名空间的名称 |
| \_\_CLASS\_\_ | 当前类的名称 |
| \_\_TRAIT\_\_ | 当前Trait的名称 |
| \_\_METHOD\_\_ | 类的方法名 |
| \_\_FUNCTION\_\_ | 函数名称 |
| \_\_DIR\_\_ | 文件所在目录 |
| \_\_FILE\_\_ | 文件的完整路径和文件名 |
| \_\_LINE\_\_ | 文件中的当前行号 |

示例代码：

|  |
| --- |
| <?php  echo "当前文件的命名空间为：" . \_\_NAMESPACE\_\_;  echo "<br/>";  echo "当前文件的类名为：" . \_\_CLASS\_\_;  echo "<br/>";  echo "当前文件的 Trait 为：" . \_\_TRAIT\_\_;  echo "<br/>";  echo "当前文件的方法为：" . \_\_METHOD\_\_;  echo "<br/>";  echo "当前文件的函数为" . \_\_FUNCTION\_\_;  echo "<br/>";  echo "当前文件的目录为：" . \_\_DIR\_\_;  echo "<br/>";  echo "当前文件的文件名为：" . \_\_FILE\_\_;  echo "<br/>";  echo "当前所在行数为：" . \_\_LINE\_\_;  echo "<br/>";  echo "当前PHP版本为：" . PHP\_VERSION;  echo "<br/>";  echo "当前PHP所在操作系统为：" . PHP\_OS;  echo "<br/>"; |
| 输出结果如下： |
| 当前文件的命名空间为：  当前文件的类名为：  当前文件的 Trait 为：  当前文件的方法为：  当前文件的函数为  当前文件的目录为：D:\WorkSpace\PHP\PHPLearn  当前文件的文件名为：D:\WorkSpace\PHP\PHPLearn\check\_data\_type.php  当前所在行数为：16  当前PHP版本为：7.3.4  当前PHP所在操作系统为：WINNT |

### 判断常量是否存在

判断常量是否存在时，使用defined()函数，就是在define()多了一个d字母。

## PHP变量

### 变量定义

变量的定义必须以**$**开头，后面跟上**数字**、**字母**、**下划线**，但是**不能以数字**开头。

|  |
| --- |
| 示例 |
| <?php  $122 = "hell"; // × 错误  $\_name = "world"; // √ 正确  $age = 25; // √ 正确 |

### 变量赋值

变量格式如下：

|  |
| --- |
| $变量名 = 变量值 |

$不是变量名的一部分，从以下示例可以看出其中的区别。

|  |
| --- |
| 示例 |
| <?php  $name = "orange"; |
| 说明：  $name 为变量  name 为变量名  orange为变量值 |

#### 普通赋值

#### 引用赋值

引用赋值是值用不同的名字来访问同一个变量地址的内容，当改版其中一个变量的值时，另一个也跟着发生变化。使用“&”符号来表示引用。

|  |
| --- |
| <?php  $i = "orange";  echo $i;  echo "<br/>";  /\*  \* 把$i的地址保存到$j中，因此$j引用了$i的值  \* 因此，修改$j的值就是修改$i的值  \*/  $j =& $i;  $j = "zheng";  echo $i;  echo "<br/>";  echo $j; |
| 输出结果 |
| orange  zheng  zheng |

### 变量特点

在 PHP 中，变量属于**弱类型**，同一个变量可以保存不同数据类型的变量值，比如：

|  |
| --- |
| 示例 |
| <?php  $age = 25; // √ 正确  print $age;  $age = "orange"; // √ 正确  print $age;  $age = true; // √ 正确  print $age;  $age = 45.89; // √ 正确  print $age; |

如上例子所示，可以给变量 age 定义不同数据类型的值。

### 预定义变量

|  |  |
| --- | --- |
| PHP中的预定义变量 | |
| $GLOBALS | Global变量 |
| $\_SERVER | 服务器变量 |
| $\_COOKIE | Cookies变量 |
| $\_SESSION | Session变量 |
| $\_FILES | 文件上传变量 |
| $\_REQUEST | REQUEST变量 |
| $\_GET | GET 变量 |
| $\_POST | POST变量 |
| $\_ENV | 环境变量 |
| $HTTP\_RAW\_POST\_DATA |  |

## PHP运算符

### 算术运算符

### 字符串操作符

### 赋值运算符

### 自增自减运算符

### 逻辑运算符

### 比较运算符

### 三元运算符

### 位移运算符

### 运算符优先级

## PHP表达式

## PHP流程控制

### 条件语句

if语句

|  |
| --- |
| 格式 |
| <?php  if(){  ……  } |
| 示例 |
|  |

if……else……

|  |
| --- |
| 格式 |
| <?php  if(){  ……  }else{  ……  } |
| 示例 |
|  |

if……elseif……else

|  |
| --- |
| 格式 |
| <?php  if(){    }elseif (){    }else{    } |
| 示例 |
|  |

switch……case语句

|  |
| --- |
| 格式 |
| <?php  switch (表达式){  case "1":  "";  break;  case "2":  "";  break;  default:  "";  } |
| 说明 |
| 表达式的值一般为字符串或者整数。 |
| 示例 |
|  |

### 循环语句

while循环

|  |
| --- |
| 格式 |
| <?php  while (true){    } |
| 示例 |
|  |

do……while循环

|  |
| --- |
| 格式 |
| <?php  do{  ……  }while(true); |
| 示例 |
|  |

for循环

|  |
| --- |
| 格式 |
| <?php  for($i=1;$i<100;$i++){  ……  } |
| 示例 |
|  |

foreach循环

|  |
| --- |
| 格式 |
| <?php  foreach($arr as $key=>$value){  ……  } |
|  |
|  |

### 跳转语句

break语句

作用：用于循环语句和 switch 开关语句，表示跳出当前所在的循环体。

break 一般情况下用于跳出当前循环，也可以指定要跳出几重循环，格式如下：

|  |
| --- |
| <?php  for(){  while(){  ……  break 2;  }  }  echo '跳出 2 重循环'; |

在例子中 break 2 表示跳出 2 重循环，所以当跳出来之后，接着执行 echo 语句了。

continue语句

作用：continue 用于跳过本次循环的剩余语句，并进入下一次循环。continue 也可以指定跳过几重循环。

return语句

作用：返回语句，用于函数中，返回调用函数的地方。

## PHP输出语句

echo：

|  |
| --- |
| echo "Hello World"; |
|  |

var\_dump：用于输出数组。

|  |
| --- |
| $my\_arr = [1,2,3,4,5,6];  var\_dump($my\_arr); |
|  |

print\_r：

|  |
| --- |
| <?php  $my\_arr = [1, 2, 3, 4, 5, 6];  print\_r($my\_arr, true);// true 表示返回值 |
|  |

printf：用于格式化输出。

dd:

|  |
| --- |
| <?php  $my\_arr = [1, 2, 3, 4, 5, 6];  dd($my\_arr); |
|  |

## PHP编码规范

## 字符串和正则

## PHP数组

## PHP函数

### 自定义函数

### 内置函数

构造函数

析构函数

对象的引用

对象的克隆（clone）

对象访问权限（public、protected、private）

接口（interface）

抽象类

魔术方法

静态成员

### 函数参数

#### 形式参数

#### 实际参数

#### 变长形参

#### 变长实参

|  |
| --- |
| 示例 |
| <?php  # -\*- coding:utf-8 -\*-  # @ProjectName: DeDeCMS  # @Email : 276517382@qq.com  # @FileName : ${NAME}.py  # @DATETime : 2019/11/26 13:45  # @Author : 笑看风云  //变长形参和变长实参  function sum($a, $b, ...$c)  {  print\_r($a);  echo '<br>';  print\_r($b);  echo '<br>';  print\_r($c);  echo '<br>';  }  $arr = [1, true, 6, 'orange', 8, 9.99, 11];  sum(1, 2, 3, 4, 5, ...$arr); |
| 结果 |
| 1<br>2<br>Array  (  [0] => 3  [1] => 4  [2] => 5  [3] => 1  [4] => 1  [5] => 6  [6] => orange  [7] => 8  [8] => 9.99  [9] => 11  )  <br> |

### 匿名函数

### 闭包函数

https://www.jianshu.com/p/6f949ce649d7

https://www.jianshu.com/p/73d4fa0d3e9b

https://www.php.cn/php-ask-429876.html

https://www.cnblogs.com/WuNaiHuaLuo/p/4928524.html

匿名函数：

https://www.cnblogs.com/liliuguang/p/7832472.html

依赖注入：

https://www.cnblogs.com/liliuguang/p/7825078.html

回调函数：

https://www.cnblogs.com/zzl-21086595/p/4547519.html

命名空间：

https://ask.csdn.net/questions/823382

# 核心技术

## PHP面向对象

## 数据库操作

### MySQL操作

### PDO数据库操作

# 高级应用

## PHP与Ajax

## 会话控制

### Cookie

### Session

## 文件操作

### 文件包含

|  |  |
| --- | --- |
| require | include |
|  |  |

|  |  |
| --- | --- |
| require\_once | include\_once |
|  |  |

### 文件上传

### 文件读写

## 图形和图像处理

# 项目实战

# PHP手册

<https://www.php.net/manual/zh/>

# 技术拓展

## PHP 如何拦截图片请求并进行处理？

我们以文章内容为例说明，如下是一篇包含图片的文章正文内容：

|  |
| --- |
| <p>  <center>  <img alt="" height="418" src="/uploads/allimg/aHR0cHM6Ly9uLnNpbmFpbWcuY24vdGVjaC90cmF.jpg?token=uc2Zvcm0vMjQ4L3c2MzBoNDE4LzIwMjAwNTA2L2NhNWEtaXRleWZ3djYwNDc2NDIuanBn" width="630" /></center>  </p>  <p>  新浪科技讯 北京时间5月6日午间消息，据外媒报道，三星电子副董事长李在镕将为集团先前的不当行为&mdash;&mdash;帮助他&ldquo;继承&rdquo;父亲李健熙的三星集团董事长一职&mdash;&mdash;向公众道歉。</p>  <p>  <center>  <img alt="" height="418" src="/uploads/allimg/aHR0cHM6Ly9waWNzNS5iYWlkdS5jb20vZmVlZC9hNjg2YzkxNzdmM2U.jpg?token=2NzA5ZDNiYzUyZTdkODFkMDEzYmY5ZGM1NTRlLmpwZWc/dG9rZW49MDRlODkzZTkzYzM2OThiOTUwMThlYTUwNWJmNGExZjA=" width="630" /></center>  </p>  <p>  周一，三星高层消息人士与韩国媒体透露，他们预计，副董事长李在镕，将在集团外部合规监督委员会设定的5月11日最后日期之前，做公开道歉。</p>  <p>  一名匿名消息人士说：&ldquo;说实话，很多情况仍在持续审查中。但一个可行的选择是让副董事长阅读准备好的声明向公众道歉。道歉的时机很重要，而且如果副董事长在最后期限之前这么做的话，道歉应该会有一定影响。严肃地说，我认为效果将非常积极。&rdquo;</p>  <p>  <center>  <img alt="" height="418" src="/uploads/allimg/aHR0cHM6Ly9waWNzNy5iYWlkdS5jb20vZmVlZC9hMmNjN2NkOThkMTAwMWU5Nzd.jpg?token=jMzkxYmQ2N2Q0ZTVlYTU2ZTc5N2UyLmpwZWc/dG9rZW49NmE5YmE4N2RlZmUzYmU1NmU3MmFlNmE0NGNlMjgxZDU=" width="630" /></center>  </p> |

我们把正常图片的地址进行 base64 编码，然后拆成两个部分，一部分是图片的文件名，另一部分则是 token 参数的值。

然后我们在 .htaccess 文件中对图片进行拦截，如下代码：

|  |
| --- |
| RewriteRule ^(uploads/allimg/)(.\*\.)(jpg|gif|png|bmp|swf|jpeg)$ /doImages.php [L] |

接下来，我们在网站的根目录下创建 `doImages.php` 文件，并在文件中编写如下代码：

|  |
| --- |
| <?php    $req\_uri = $\_SERVER["REQUEST\_URI"];  $chazhao = "/uploads/allimg/";  $start = stripos($req\_uri,$chazhao) + strlen($chazhao);  $end = stripos($req\_uri,".");  $length = $end - $start;  $canshu\_1 = substr($req\_uri, $start, $length);  $query = $\_SERVER["QUERY\_STRING"];  $canshu\_2 = str\_replace("token=", "", $query);  $img\_base64\_url = $canshu\_1 . $canshu\_2;  $img\_url = base64\_decode($img\_base64\_url);    header("HTTP/1.1 301 Moved Permanently");    Header("Location: ".$img\_url); |

这样就达到了 PHP 拦截图片请求的目标了。

注意其中，字符串 `/uploads/allimg/` 和 `token=` 是固定不变的。

其它方式的代码:

|  |
| --- |
| <?php  /\*  URL 编码加密问题:  1、base64 编码 A~Z a~z 0~9 + / =  2、https://blog.csdn.net/qq\_39494838/article/details/75649678  在 .htaccess 文件中增加如下代码:  #带参数的图片  RewriteRule .\*.(gif|jpg|jpeg|bmp|png) /image.php [L]  或者  RewriteRule .\*.(gif|jpg|jpeg|bmp|png) /image.php [R=301,L]  http://www.uploadimage.com/jianfeifangfa/3135.html  http://www.uploadimage.com/uploads/allimg/2019/4/9/b96f077889ba990b721c6732051333d1.jpg?auth\_key=aHR0cDovLzViMDk4OGU1OTUyMjUuY2RuLnNvaHVjcy5jb20vaW1hZ2VzLzIwMTkwNDE1LzgwNTA0MDM5MTBmZTRhOGZiZjE4ZGNmMzQyZDBmZjgyLmpwZw==  http://5b0988e595225.cdn.sohucs.com/images/20190415/8050403910fe4a8fbf18dcf342d0ff82.jpg  \*/  # 参考 http://www.php.cn/php-weizijiaocheng-382897.html  function get\_remote\_image($imgsrc){  /\*  $ch = curl\_init();  $headers = array (  "Host" => "p.nanrenwo.net",  "Connection" => "keep-alive",  'User-Agent" => "Mozilla/5.0 (compatible; Baiduspider/2.0; +http://www.baidu.com/search/spider.html)'  );  curl\_setopt($ch, CURLOPT\_URL, $imgsrc); # 文件请求地址  curl\_setopt($ch, CURLOPT\_HTTPHEADER, $headers); # 构造请求头  curl\_setopt($ch, CURLOPT\_RETURNTRANSFER, 1); # 获取的信息以文件流的方式返回  $data = curl\_exec($ch); # 执行 curl  curl\_close($ch); # 关闭 curl  if($data){  return $data;  }else{  echo "no pictures";  }  \*/  $ch = curl\_init();  //随机生成IP  $ip = rand(0,255).'.'.rand(0,255).'.'.rand(0,255).'.'.rand(0,255) ; // 百度 蜘蛛  $timeout = 15;  curl\_setopt($ch,CURLOPT\_URL,$imgsrc);  curl\_setopt($ch,CURLOPT\_TIMEOUT,0);  //伪造百度 蜘蛛IP  curl\_setopt($ch,CURLOPT\_HTTPHEADER,array('X-FORWARDED-FOR:'.$ip.'','CLIENT-IP:'.$ip.''));  //伪造百度 蜘蛛头部  curl\_setopt($ch,CURLOPT\_USERAGENT,"Mozilla/5.0 (compatible; Baiduspider/2.0; +http://www.baidu.com/search/spider.html)");  curl\_setopt($ch,CURLOPT\_RETURNTRANSFER,1);  curl\_setopt($ch,CURLOPT\_HEADER,0);  curl\_setopt($ch,CURLOPT\_CONNECTTIMEOUT,$timeout);  curl\_setopt($ch,CURLOPT\_SSL\_VERIFYPEER,false);  curl\_setopt($ch, CURLOPT\_IPRESOLVE, CURL\_IPRESOLVE\_V4);  $content = curl\_exec($ch);  return $content;  }  $auth\_key = $\_GET['auth\_key'];  $imgsrc = base64\_decode($auth\_key);  $img = get\_remote\_image($imgsrc);  # extension=php\_openssl.dll C:\PHPStudy\PHPTutorial\php\php-5.4.45\php.ini  @header('Content-Type: image/png');  echo $img;  exit;  ?> |

# PHP7特性

## trait特性

通过trait实现类似于其它语言的多继承功能。

|  |
| --- |
| <?php  trait MyTrait  {  public function getInfo()  {  echo '我是 Trait 中的方法';  echo '<hr>';  }  }  class Father  {  use MyTrait;  public function getInfo()  {  echo '我是 Father 中的 getInfo 方法';  echo '<hr>';  }  }  class Child extends Father  {  use MyTrait;  public function getInfo()  {  echo '我是 Child 中的 getInfo 方法';  echo '<hr>';  }  }  $obj = new Child;  $obj->getInfo();  /\*  \* 如果 父类 子类 和 trait 定义了相同的方法，那么调用的优先级为：  \* 子类 > trait > 父类  \* 为什么呢？  \* 1、子类 extends 父类时，继承了 getInfo() 方法；  \* 2、子类 use MyTrait 时，继承了 MyTrait 中的 getInfo() 方法，从而覆盖了父类的 getInfo() 方法；  \* 3、子类自定义了 getInfo() ，从而覆盖了 MyTrait 中的 getInfo() 方法  \*/ |

# 常用函数列表



# Composer依赖管理工具

## Composer介绍

Composer 的官网地址为：<https://getcomposer.org/>

Composer 的使用文档为：<https://getcomposer.org/doc/00-intro.md>

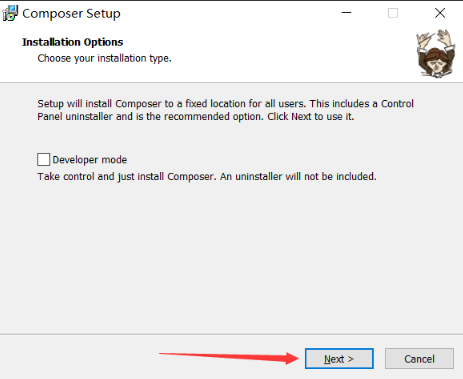
## Composer安装

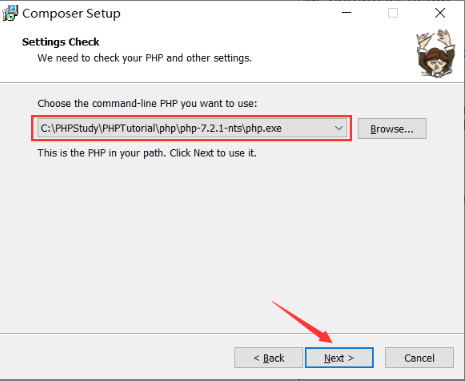
### Windows系统下安装Composer

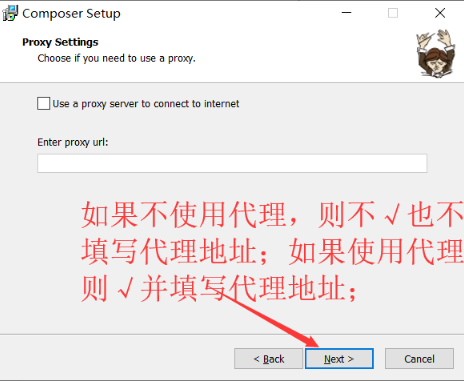
#### Composer安装

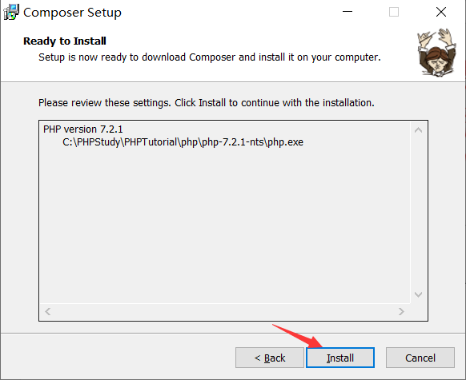
在安装 Composer 之前，要求系统必须有 PHP 5.3.2+ 以上环境，我这里使用的是 PHPStudy 套件中的 PHP 7.2 版本。

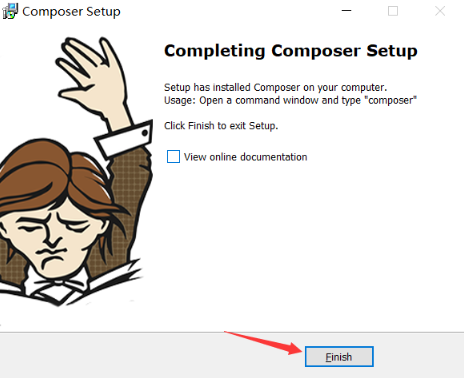
然后，打开 <https://getcomposer.org/Composer-Setup.exe> 进行下载，下载完成之后，双击打开进行安装，如下图所示：











点击 Install 直到安装完成。

右键单击“电脑” → 属性 → 高级系统设置 → 环境变量 → 系统变量 → Path → 编辑 → 其中有一个为：

|  |
| --- |
| C:\ProgramData\ComposerSetup\bin |

的环境变量，说明 Composer 安装完成之后，自动设置了环境变量，如果没有则自己手工配置。

#### Composer验证

在命令行输入如下命令：

|  |
| --- |
| composer --version |

如下过程：

|  |
| --- |
| C:\Users\Administrator\Desktop  λ composer --version  Composer version 1.9.1 2019-11-01 17:20:17  C:\Users\Administrator\Desktop  λ |

说明 Composer 安装成功。

### CentOS7 系统下安装 Composer

#### Composer安装

在安装 Composer 之前，要求系统必须有 PHP 5.3.2+ 以上环境，我这里使用的 PHP 版本为 7.2 版本。

然后执行如下操作进行安装：

|  |
| --- |
| curl -sS https://getcomposer.org/installer | php  mv composer.phar /usr/local/bin/composer  或者  cp composer.phar /usr/local/bin/composer |

#### 配置国内镜像

由于众所周知的原因，国外的网站连接速度很慢。因此安装的时间可能会比较长，我们建议使用国内镜像（比如下面的阿里云）。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://mirrors.aliyun.com/composer/ |
| 阿里云Composer全量镜像的地址为：  <https://developer.aliyun.com/composer> |

在CentOS中配置Composer全局镜像时，出现如下警告：

|  |
| --- |
| Warning: putenv() has been disabled for security reasons in phar |

解决步骤：

|  |
| --- |
| [root@localhost ~]# cd /www/server/php/74/etc/  [root@localhost etc]# gedit php.ini |

然后查找disable\_functions然后把值里面的putenv删除，然后通过宝塔面板重启PHP服务。在安装其他框架时，需要把proc\_open也删除。

然后再重新配置上面的国内镜像。

取消全局配置：

|  |
| --- |
| composer config -g --unset repos.packagist |

Composer 各大厂商镜像地址：

|  |
| --- |
| 阿里云：  composer config -g repo.packagist composer https://mirrors.aliyun.com/composer/ |
| 华为云：  composer config -g repo.packagist composer https://mirrors.huaweicloud.com/repository/php/ |
| 腾讯云：  composer config -g repos.packagist composer https://mirrors.cloud.tencent.com/composer/ |
| 交通大学：  composer config -g repos.packagist composer https://packagist.mirrors.sjtug.sjtu.edu.cn |
| phpcomposer：  composer config -g repo.packagist composer https://packagist.phpcomposer.com |
| cnpkg：  composer config -g repos.packagist composer https://php.cnpkg.org |

#### Composer更新

在终端输入如下代码进行 Composer 的更新：

|  |
| --- |
| composer selfupdate //终端执行该命令将 composer 更新到最新版 |

结果如下所示：

|  |
| --- |
| C:\Users\Administrator\Desktop  λ composer selfupdate  You are already using composer version 1.9.0 (stable channel).  C:\Users\Administrator\Desktop  λ |