echo 打印 | var\_dump 打印详细信息

一、

1、数据类型

gettype($name); ==>返回数据类型

settype($age,’string’); ==>永久转换数据类型

is\_string($name); ==>判断一个数据是否为字符串

is\_number($name); ==>判断一个数据是否为数字

1. 数字方法

Intval(123.6); ==>123 取整

Round(123.6); ==>124 四舍五入

Floatval(123.123); ==>123.123 保留浮点数

比较两个浮点数是否相等时不能直接比较，应先定一个精度值0.1||0.01||0.0000001，

两数相减的绝对值abs()在和精度值比较，小于精度值则相等。

1. 字符串

$net = ‘官网\’www.victorg.club\’’; ==>官网’www.victorg.club’ \为转义符

$net = “官网\”www.victorg.club\”的”; ==>官网”www.victorg.club” \为转义符

适用于html结构的字符串定义方法 $heredoc、$nowdoc

$heredoc1 = <<<"L"

<ul>

<li>{$str}</li>

<li>"$str"</li>

<li>'$str'</li>

<li>$str</li>

<li>4</li>

</ul>

L;

$heredoc2 = <<<'L'

<ul>

<li>{$str2}</li>

<li>$str2</li>

<li>"$str2"</li>

<li>'$str2'</li>

<li>4</li>

</ul>

L;

注：以<<<开头L为自定义名称，如果L为双引号，则内部变量可以解析，单引号不能解析，以字符串形式渲染。

1. 数组

索引数组和关联数组

$arr =[q,w,e]; $arr2=[‘name’=>’gss’,’age’=>18];

索引相同时会被覆盖 $arr3=[‘id’:1,’id’:2,’num’:3] ==>2,3

添加数据 $arr[]=’gss’; 自动按索引添加

$arr[‘add’] = ‘add’; 按键名添加

$arr[‘add’] = ‘addd’; 键名相同时表更新

删除数组中某值 unset($arr[‘键名或索引’])

清空数组（非删除）用循环

foreach($arr as $key => $value){

Unset($arr[$key]);

}

删除数组

Unset($arr);

重建索引，针对索引数组中间某项被删后，重新按顺序建立索引

array\_values($arr);

计算数组中元素个数

count($arr);

1. 对象

定义：

Class Student{

Public $name=’gss’;

Public function getInfo(){

Return $this->name;

}

}

$s1 = new Studen;

$s1->getInfo();

标准类：stdClass

$s2 = new stdClass();

$s2->name = ‘gss;

$s2->fn = function(){

Return ‘返回值’;

}

调用：$s2->name;

Call\_user\_func($s2->fn);

1. 变量、作用域

局部变量、全局变量（$GLOBALS）类似于js里的window

$name = ‘gss’;

$GLOBALS[‘name’]; //gss

funtion getName(){

$name =’ggg’;

retufn $name;

}

getName(); //ggg

$name2 = $name;

$name2 = ‘lll’;

//name==>gss,name2==>lll

引用：相当于一个值有两个名字，改其中一个都跟着改

$name3 = &$name;

$name3 = ‘kkk’;

//name==>kkk,name2==>lll,name3==>kkk

静态变量static，供重复累加使用

Function sum(){

static $count =10;

return $count +=10;

}

Sum(),sum() //20,30

1. $GLOBALS中的$\_COOKIE,$\_SESSION,$\_GET,$\_POST

$\_GET获取get请求的参数数组

$\_GET[‘name’];获取某一项

$\_POST同上获取post数组

$\_COOKIE同上

Setcookie(name,value,time()+3600);参数：键、值、时间（time()为当前时间戳）,

删除某cookie将时间-3600即设置已经过了的时间即可。

Isset(val);判断是否存在

Header(Location:1.php);跳转到1.php

session

Session的值是数组。

session\_start ();

先开启session会话，开启后每台电脑即生成唯一cookie (一个sessionid)，同时服务端生成对应session文件（C:\wamp64\tmp）。

session\_id();获取sessionid。

设置值：

$\_SESSION[‘name’]=’gss’;

$\_SESSION[‘domain’]=’www.victorg.club’;

读取：$\_SESSION[‘name’];

删除：数组删除方法unset；

unset($\_SESSION[‘name’]);

全部删除：session\_destroy();

1. if、switch、for、while、do{}while()、foreach

While($i<$num){

....;

$i++;

}

do{ //先执行一次

....;

$i++;

}while($i<$num);

foreach($arr as $key=>$value){

echo $key;

Echo $value;

}

1. 运算符:不等于: 30<>’30’ ==>false 30!==’30’ ==>true
2. 函数

形参、实参

func\_get\_args();获取实参形成数组;

func\_get\_arg(2);获取索引为2的实参；

函数内操作对全局变量不产生影响：

$num =10;

Function add($num){

return $num+=10;

}

add($num); //20

$num; //10

引用参数的方法可改变全局变量：

$num =10;

Function add(&$num){

return $num+=10;

}

add($num); //20

$num; //20

可变函数：

function add($n,$n2){};

function div($n,$n2){};

$fn=’add’;

$fn($n);

$fn = ‘div’;

$fn($n);

可变函数调用用：call\_user\_func\_array($fn,[$n,$n2]);

回调函数:

Function cl($fnName,$n1.$n2){

//return $fnName($n1,$n2);

return call\_user\_func\_array($fnName,[$n1,$n2]);

}

cl((‘add’,$n,$n2);

回调方式调用类或对象中的方法：

class Student{

static function f1($s1,$s2){return $s1.$s2}; //静态方法

public function f2($s1,$s2){return $s1.$s2}; //普通方法

};

call\_user\_func\_array( [ ‘Student’ , ’f1’ ], [ ‘我叫’ , ’谷守帅’ ] ); //调用静态方法直接在类里找到方法即可调用。

call\_user\_func\_array( [ new Student , ‘f2’ ] ,[ ‘我叫’ , ‘谷守帅’ ] ); //调用普通方法需通过new实例再找到相应方法调用。

1. 引入文件的两种方式require和include

require:再文件头引入 include：一般再程序运行中引入

被引入文件与正常php文件无区别，不需export暴漏方法

1. 闭包：在一个函数内调用匿名函数并将返回值返回。

function closer ($d){

$s=’我是函数内部的数据，匿名函数可以用use()调用’;

$f1 = function( $data )use( $s ){return $data.$s};

Return $f1($d);

};

closer(‘我是唯一的实参，$s不算函数参数’);

二、

1. array深入
   1. 数组指针(默认从索引为0开始)：key() current() next() prev() end() reset()
      1. $arr = [‘name’=>’gss’ , ‘sex’=>’man’ , ‘age’=> 18];

Key($arr);==>name; //当前指针键名

Current($arr);==>gss; //当前指针值

Next($arr); //指针下移

end($arr); //指向最后一个指针

reset($arr); //重置指针

指针超出数组长度时，键值为空

* 1. 数组循环
     1. For while foreach each

$arr = range(2,10,2); //生成2~10间距为2的整数；

For ($i=0; $i<count($arr);$i++){ echo key($arr).current($arr); nexy($arr) };

Foreach( $arr as $key=>$value){};

each:逐个遍历数组，将每个元素分为0=>key,1=>value,key=>key,value=>value的数组

each($arr); ==> [0=>’name’,1=>’gss’,’key’=>’name’,’value’=>’gss’];

each($arr); ==> [0=>sex,1=>man,’key’=>sex,’value’=>man];

list()同echo，都是语言结构，可放等号左边。只针对索引数组

$arr =[1,2,3];

List($a, ,$c) =$arr;

$a==>1, $c==>3;

* 1. array\_values()、array\_keys();
     1. Array\_values($arr); //取出数组的值重新构成有序的索引数组
     2. Array\_keys($arr); //取出数组的键重新构成有序的索引数组
  2. 判断数组中有无某值、某键、查询键名
     1. Array\_key\_exists(‘name’,$arr); //返回布尔值（没有严格模式）；
     2. In\_array(‘gss’,$arr,true); //判断某值是否在数组中(区分大小写)，true表示开启严格模式（区分数据类型），返回布尔默认false
     3. Array\_search($value,$arr,true); //查询某值在数组中的键名；返回键名
  3. 数组元素操作
     1. Count($arr); //数组元素个数
     2. Count($arr,1); //开启多维数组检测，包括多维数组中所有元素的个数
     3. Array\_count\_values($arr); //数组中每个值出现的次数，并返回以值为键，次数为值的数组。由于此新数组以原数组值为键，所以原数组的值需是整数或字符串，否则回报错。（不区分数据类型）
     4. Array\_unique($arr); //去掉数组中重复的值返回新数组（不区分数据类型）
  4. Array\_filter($arr,clb);过滤数组
     1. Clb回调函数接受数组中每一个元素，判断为true的返回用新数组接受
     2. $arr =[1,2,3,4,5];
     3. Function clb($n){

Return $n & 1; //位运算，基数返回1；

}

* + 1. $arr\_odd = array\_filter($arr,’clb’);
  1. Array\_walk($arr,clb); 遍历更新数组
     1. Clb可接受两或三个参数（&$v,$k,$suffix）
     2. &$v:引用方式的数组的值（这样才能更新）
     3. $k:键
     4. $suffix:后缀名（可忽略）
     5. $arr =[‘name’=>’gss’,’age’=>18];
     6. Function clb(&$v,$k.$s){

$v .=’更新’;

$v .=$s; //添加后缀

};

Array\_walk($arr,’clb’,’.txt’); //添加后缀直接添加实参

* 1. Array\_map(clb,$arr,$arr1); 可遍历多个数组，取值拼接形成新的索引数组
     1. $arr =[1,2,3]; $arr2=[‘唐山’,’北京’,’秦皇岛’];
     2. Function clb($v1,$v2){

Return ‘第’.$v1.’座城市是’.$v2;

}

$arr3 = array\_map(‘clb’,$arr,$arr2);

* 1. 数组升序降序
     1. Sort rsort 按值升降序形成索引数组
     2. kSort lrsort 按键升降序
     3. Asort arsort 按值升降序但保留键
  2. Array\_multisort() 给多个或多维数组排序
     1. Array\_multisort($arr,SORT\_DESC,$arr2,SORT\_ASC);
        1. SORT\_DESC降序；SORT\_ASC:升序
        2. 升序降序针对第一个数组，其余数组按照原有对应关系排序，再考虑升序降序
  3. Array\_push ,array\_pop || array\_unshift ,array\_shift 从尾部添加删除||从头部添加删除
     1. Array\_push($arr,’a’,’b’);
     2. Array\_pop($arr);
     3. Array\_unshift($arr,’a’);
     4. Array\_shift($arr);

1. Obj
   1. 声明
      1. class Demo{

Public $name=’gss’; //公共属性，可供外部调用

Static $age = 18; //静态属性，只能内部调用

protected $sex =’man’;

Public function getName(){

return $this->name; //$this调用属性、方法

}

public function getSelf(){

Return new self();

}

Public function getStatic(){

Return new static();

}

}

Class Demo2 extends Demo{ //Demo2继承demo的子类,能调用Demo中的方法属性。

public functin getParent(){

Return new parent();

}

}

* 1. 实例化
     1. $obj = new Demo();
     2. $obj2 = $obj; //对象赋值是引用行，不会产生新对象
        1. Get\_class($obj2)==get\_class($obj)==Demo //获取实例的类名
     3. (New Demo)->getSelf(); //指向Demo
     4. (new Demo2)->getParent(); //指向Demo
     5. (new Demo)->getStatic(); //指向Demo
     6. (new Demo2)->getStatic(); //指向Demo2
  2. 类常量
     1. const name=’dd’; //注意没有$符
     2. const html =<<<’html’

<div>支持nowdoc语法</div>

Html;

在类的方法中不能再用$this->name的方式调用，需用self::name

在类外调用直接用类名即可 Demo::name

* 1. 自动引用外部文件
     1. \_\_autoload()函数，如果发现为定义的类被调用，会自动执行\_\_autoload()方法,根据类名找相应php文件及文件内的类
        1. Function \_\_autoload($className){

$path = $className.’.php’;

If( file\_exists ($path) ){

Require\_once( $path );

}else{

echo ‘无此php文件或文件中未定义此类’;

}

}

(new Test) -> name;

//注：Test为当前未定义的类。此时会调用\_\_autoload( Test );在查找Test.php文件时不区分大小写，即test.php文件也可以。找到文件后执行文件内名为Test类。

* + 1. spl\_autoload\_register() 执行自定义的autoload
       1. class Loader{

public function autoLoader($className){

$path = $className.’.php’;

If( file\_exists ($path) ){

Require\_once( $path );

}else{

echo ‘无此php文件或文件中未定义此类’;

}

}

}

spl\_autoload\_register( [(new Loader), ‘autoLoader’ ]); //调用即可自动执行，类中public方法需实例化后调用。

* 1. \_\_construct和\_\_destruct都是类中的自有方法前者在实例话调用时执行，后者在对象销毁时执行
     1. class Demo {

Public $name;

Public $age;

Public function \_\_construct($name,$age){

$this->name = $name;

$this->age = $age;

}

Public function \_\_destruct(){

echo ‘对象被销毁了，或者此次调用结束了，最后执行’;

}

}

$obj = new Demo(‘谷守帅’,18); //初始化实例传参直接在类上传

* 1. private 私有属性 只能在内部访问
     1. \_\_get \_\_set \_\_isset \_\_unset 通过这些自有方法可实现外部操作
     2. class Demo{

Private $name;

Private $age;

Public function \_\_construct($name ,$age){

$this->name = $name;

$this->age = $age;

}

Public function \_\_get($name){

return $this->$name;

}

public function \_\_set($name ,$value){

return $this->$name = $value;

}

public function \_\_isset($name){

return isset($this->$name);

}

public function \_\_unset($name){

Unset($this->$name);

}

}

$obj = new Demo(‘gss’,18);

$obj->name;

$obj->age=20;

isset($obj->name);

unset($obj->age);

* 1. protected 受保护的属性，内部和子类访问
     1. class Father{

protected $name;

protected function show(){ //受保护的方法不能被外界调用

return $this->name;

}

public function show2(){

Return $this->name;

}

public function \_\_construct($name){

$this->name = $name;

}

}

class Son extends Father{

protected $sex;

public function \_\_construct($name,$sex){

parent::\_\_construct($name); //子类继承并添加参数

$this->sex=$sex;

}

//重写show方法，注权限不能低于父类

public function show(){

return parent::show().’添加sex的展示’.$this->sex;

}

}

$obj = new Father(‘gss’);

$obj2= new Son(‘谷守帅’,’男’);

$obj2->show(); //因权限变了，所以能访问

* 1. static 成员状态符，静态符
     1. 访问限制符：
        1. public 公共属性，内外子都能调用
        2. private 私有属性，只有内部调用
        3. protected 受保护属性，内部和子类调用
     2. 成员状态符：
        1. 静态符：
           1. self ：指向所在类，对象初始化时调用
           2. parent ：指向父类
           3. static ：指向所在类，运行中绑定调用（延迟绑定）
        2. 非静态：$this(伪变量)
     3. 访问静态资源
        1. Demo::$name; Demo::getName(); 类直接调用
        2. $obj::$name; $obj::getName(); $obj->getName(); 对象调用静态属性只能用::形式，方法能用两种。
        3. class Father{

Public static $name;

Private static $age;

Function \_\_construct($name,$age){

Self::$name = $name;

Self::$age = $age;

}

Static function show(){

Echo self::$name.self::$age;

}

Static Function getSex(){

Echo static::$sex;

}

}

Class Son extends Father{

Static $sex=’man’;

Static function getName(){

Echo parent::$name;

}

}

Son::getSex(); //父类用子类属性。通过static 延迟绑定实现。

* 1. clone :同new Demo();会产生新对象
     1. class Demo {

$name = ‘gss’;

}

$obj = new Demo();

$obj2=$obj

$obj->name=’lss’;

$obj3 = clone $obj;

$obj1->name=’wss’;

//$obj->name =====wss

//$obj2 =========wss

//$obj3 =========lss

* 1. \_\_call \_\_callStatic
     1. class Demo{

Function \_\_call($method,$agrs){

//调用不存在的方法时自动调用改方法

}

Function \_\_callStatic ($method,$agrs){

//调用不存在的静态方法时调用

}

}

* 1. 序列化和反序列化serialize、unserialize 都是针对对象
     1. Serialize 是将类中的属性序列化成字符串
     2. Unserialize 将被序列化的类转成新的对象
        1. Class Demo{

$name = ‘gss’;

$age =18;

Function \_\_sleep(){

return [‘name’]; //序列化过滤，将要序列化的属性写在数组中

}

Function \_\_wakeup(){

$this->name=’谷守帅’; //反序列化时更改属性值

}

}

$obj = new Demo();

$objStr = serialize($obj);

$obj2 = unserialize($objStr);

* 1. 抽象类 abstract
     1. Abstract class Demo{

$name = ‘gss’;

function show(); //抽象方法

function show2(){

//可以有普通方法

}

}

class Son extends Demo{

Function show(){ //子类中必须实现父类中所有抽象方法

Echo $this->name;

}

}

* 1. 接口类 interface ,和抽象类很像
     1. 属性必须是常量 方法必须是public的抽象方法
     2. Interface Demo{

const SITENAME =’网站名称’;

function show();

}

Interface Demo2{

Const WWW = ‘www.victorg.club’;

function show2();

}

//继承用implements,可继承多个接口类

class Son implements Demo,Demo2{

funtion show(){//子类中实现抽象方法};

funtion show2(){//子类中实现抽象方法}

}

* 1. trait类：放私有代码的类，不能实例话，不能有常量，用use导入
     1. trait test1{

public $name = ‘gss’;

Function show(){

//同名方法

}

}

trait test2{

public $sex= ‘gss’;

Function show(){

//同名方法

}

}

trait test3{

use test2{

Test2::show as show2; //引入test2类，并重命名重名方法

}

public $age= ‘gss’;

Function show(){

//同名方法

}

}

class Demo{

Use test1,test3{

Test1::show insteadOf test3; //用test1中的show方法替换其他类中的show

test3::show as show3;

}

}

* 1. 命名空间namespace
     1. namespace part1;
        1. Const NAME= ‘谷守帅’;
        2. Class Demo1{

Public $age = 18;

}

* + 1. namespace part2;
       1. Use part1\Demo1 as pDemo1; //引用其他命名空间的类并重命名
       2. const NAME = ‘lss’;
       3. Class Demo1{

Public $age = 25;

}

* + - 1. Echo NAME; //lss
      2. Echo \part1\NAME; //谷守帅
  1. 遍历对象 foreach()
     1. 由于对象中的属性包含private、protected和static。不能在外部访问，所以须在类中定义遍历方法然后外部调用。Static静态成员不能遍历得到。
     2. Foreach( $this as $key=>$value){}