PHP进阶学习

[一 数组 2](#_Toc5704)

[1.数组定义 2](#_Toc11303)

[2.PHP数组 2](#_Toc25067)

[1.索引数组 2](#_Toc13162)

[2.索引数组赋值 3](#_Toc7099)

[3.关联数组 3](#_Toc27855)

[4.关联数组初始化 3](#_Toc14890)

[5.关联数组赋值 4](#_Toc13377)

[二 函数 4](#_Toc29623)

[1.自定义函数步骤 4](#_Toc28566)

[2.函数返回值 4](#_Toc17125)

[3.可变函数 4](#_Toc21285)

[4.内置函数 5](#_Toc9993)

[三 类和对象 5](#_Toc29669)

[1.简介 5](#_Toc28778)

[2.创建一个对象 5](#_Toc23421)

[3.访问控制的关键字 5](#_Toc6796)

[4.定义类的方法 6](#_Toc13816)

[5.构造函数和析构函数 6](#_Toc3481)

[6.static静态关键字详解 7](#_Toc11769)

[7.访问控制详解 7](#_Toc10306)

[7.重载 8](#_Toc29597)

[四 字符串函数 8](#_Toc15768)

[1.去除空格 8](#_Toc2933)

[2.计算字符串长度 8](#_Toc20131)

[3.截取字符串 8](#_Toc11832)

[4.查找字符串 8](#_Toc17864)

[5.替换字符串 8](#_Toc1824)

[6.格式化字符串 9](#_Toc27189)

[7.合并/分隔字符串 9](#_Toc19979)

[8.特殊字符串转义 9](#_Toc13303)

[五 正则表达式 10](#_Toc25565)

[1.正则表达式介绍 10](#_Toc11613)

[六 会话控制 10](#_Toc2740)

[1.cookie简介 10](#_Toc13527)

[2.设置cookie 10](#_Toc24809)

[3.cookie的删除和过期时间 11](#_Toc19061)

[4.cookie的有效路径 11](#_Toc15607)

[5.session与cookie的异同 11](#_Toc11950)

[6.session的使用 12](#_Toc14782)

[7.删除和销毁session 12](#_Toc1054)

[8.使用session存储登录信息 13](#_Toc32190)

[七 文件操作系统 13](#_Toc29776)

[1.读取文件内容 13](#_Toc23864)

[2.判断文件是否存在 14](#_Toc6104)

[3.获取文件的属性——001 14](#_Toc13918)

[4.获取文件的属性——002 15](#_Toc9291)

[八 日期和时间 15](#_Toc8183)

[1.获取当前时间Unix时间戳 15](#_Toc14032)

[2.取得当前的日期 16](#_Toc1564)

[3.取得日期的时间戳 16](#_Toc2145)

[4.将格式化的日期字符串转换为Unix时间戳 16](#_Toc28164)

[5.格式化格林威治（GMT）标准时间 16](#_Toc248)

[九 GD库——图形操作 17](#_Toc8432)

[1. GD库简介 17](#_Toc2363)

[2. 绘制线条 17](#_Toc16311)

[3.图形中绘制文字 18](#_Toc13917)

[4.输出图像文件 18](#_Toc2953)

[5.生成图像验证码 19](#_Toc2651)

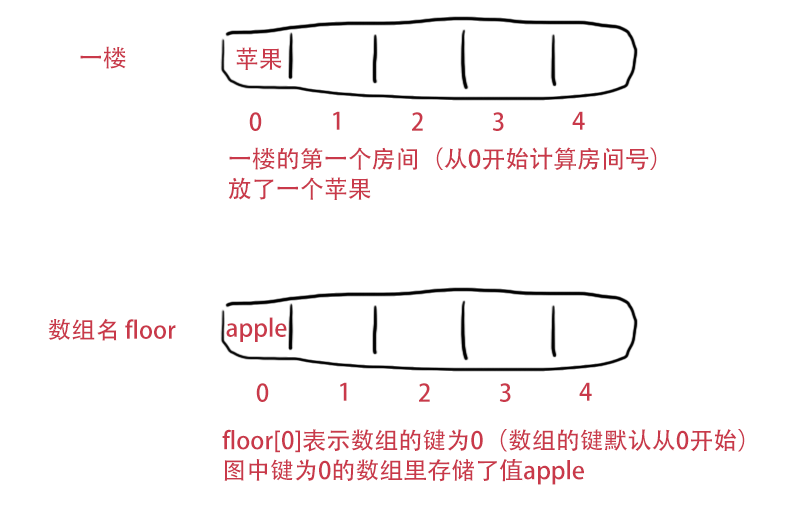
[6.图片加水印 19](#_Toc21976)

一 数组

1.数组定义

数组就是一个键值对组成的语言结构，键类似于酒店的房间号，值类似于酒店房间里存储的东西。

$arr = array();表示创建一个空数组，并把创建的空数组赋值给变量$arr。



2.PHP数组

PHP有两种数组：索引数组、关联数组。

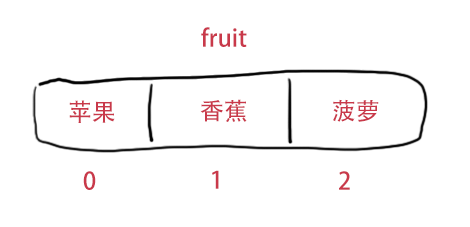
1.索引数组

···

$fruit = array(‘苹果’,’香蕉’,’菠萝’);

Print\_r($fruit);

···



2.索引数组赋值

1.第一种：用数组变量的名字后面跟一个中括号的方式赋值，当然，索引数组中，中括号内的键一定是整数。比如，$arr[0]='苹果';

2.第二种：用array()创建一个空数组，使用=>符号来分隔键和值，左侧表示键，右侧表示值。当然，索引数组中，键一定是整数。比如，array('0'=>'苹果');

3.第三种：用array()创建一个空数组，直接在数组里用英文的单引号'或者英文的双引号"赋值，数组会默认建立从0开始的整数的键。比如array('苹果');这个数组相当于array('0'=>'苹果');

3.关联数组

关联数组是指数组的键是字符串的数组



4.关联数组初始化

···

$fruit = array(

'apple'=>"苹果",

'banana'=>"香蕉",

'pineapple'=>"菠萝"

);

···

5.关联数组赋值

1.第一种：用数组变量的名字后面跟一个中括号的方式赋值，当然，关联数组中，中括号内的键一定是字符串。比如，$arr['apple']='苹果';

2.第二种：用array()创建一个空数组，使用=>符号来分隔键和值，左侧表示键，右侧表示值。当然，关联数组中，键一定是字符串。比如，array('apple'=>'苹果');

二 函数

1.自定义函数步骤

1.使用关键字“function”开始

2.函数名可以是字母或下划线开头：function name()

3.在大括号中编写函数体：

···

Function name(){

Echo ‘MyName’;

}

···

2.函数返回值

使用return关键字可以使函数返回值，可以返回包括数组和对象的任意类型，如果省略了 return，则默认返回值为 NULL。

返回语句会立即中止函数的运行，并且将控制权交回调用该函数的代码行。

函数不能返回多个值，但可以通过返回一个数组来得到类似的效果。

···

function add($a) {

return $a+1;

}

$b = add(1);

function add($a) {

return $a+1;

$a = 10;

return $a+20;

}

$b = add(1);

//返回结果一样，因为return会结束函数体

···

3.可变函数

1. 先定义一个函数

2. 定义一个变量，把值设置等于函数名（不加括号）

3. 调用这个变量+括号

···

function name() {

echo 'jobs';

}

$func = 'name';

$func(); //调用可变函数

···

4.内置函数

三 类和对象

1.简介

类是面向对象程序设计的基本概念，通俗的理解类就是对现实中某一个种类的东西的抽象，比如汽车可以抽象为一个类，汽车拥有名字、轮胎、速度、重量等属性，可以有换挡、前进、后退等操作方法。

···

class car{

Var $name = ‘汽车’;

Function getName(){

Return $this->name;

}

}

···

类是一类东西的结构描述，而对象则是一类东西的一个具体实例，例如汽车这个名词可以理解为汽车的总类，但这辆汽车则是一个具体的汽车对象。

···

$car = new car();

Echo $car->getName();

···

类与对象看起来比较相似，但实际上有本质的区别，类是抽象的概念，对象是具体的实例。类可以使程序具有可重用性。

2.创建一个对象

类通过关键字class开头，然后是类名与花括号，在花括号中定义类的属性与方法。类名必须是字母或下划线开头，后面紧跟若干个字母、数字或下划线，类名最好能够表意，可以采用名词或者英文单词。

3.访问控制的关键字

在类中定义的变量称之为属性，通常属性跟数据库中的字段有一定的关联，因此也可以称作“字段”。属性声明是由关键字 public，protected 或者 private 开头，后面跟一个普通的变量声明来组成。属性的变量可以设置初始化的默认值，默认值必须是常量

1.public：公开

2.protected：受保护

3.private：私有的

默认都为public，外部可以访问。一般通过->对象操作符来访问对象的属性或者方法，对于静态属性则使用::双冒号进行访问。当在类成员方法内部调用的时候，可以使用$this伪变量调用当前对象的属性。

受保护的属性与私有属性不允许外部调用，在类的成员方法内部是可以调用的。

···

class Car {

//定义公共属性

public $name = '汽车';

//定义受保护的属性

protected $corlor = '白色';

//定义私有属性

private $price = '100000';

public function getPrice() {

return $this->price; //内部访问私有属性

}

}

$car = new Car();

echo $car->name; //调用对象的属性

echo $car->color; //错误 受保护的属性不允许外部调用

echo $car->price; //错误 私有属性不允许外部调用

···

4.定义类的方法

方法就是在类中的function，很多时候我们分不清方法与函数有什么差别，在面向过程的程序设计中function叫做函数，在面向对象中function则被称之为方法。

访问控制的关键字代表的意义为：

1.public：公开的

2.protected：受保护的

3.private：私有的

4.static：静态的

使用关键字static修饰的，称之为静态方法，静态方法不需要实例化对象，可以通过类名直接调用，操作符为双冒号::。

5.构造函数和析构函数

PHP5可以在类中使用\_\_construct()定义一个构造函数，具有构造函数的类，会在每次对象创建的时候调用该函数，因此常用来在对象创建的时候进行一些初始化工作。

···

class Car {

function \_\_construct() {

print "构造函数被调用\n";

}

}

$car = new Car(); //实例化的时候 会自动调用构造函数\_\_construct，这里会输出一个字符串

···

同样，PHP5支持析构函数，使用\_\_destruct()进行定义，析构函数指的是当某个对象的所有引用被删除，或者对象被显式的销毁时会执行的函数。

···

class Car {

function \_\_construct() {

print "构造函数被调用 \n";

}

function \_\_destruct() {

print "析构函数被调用 \n";

}

}

$car = new Car(); //实例化时会调用构造函数

echo '使用后，准备销毁car对象 \n';

unset($car); //销毁时会调用析构函数

···

6.static静态关键字详解

1.静态属性与方法可以在不实例化类的情况下调用，直接使用类名::方法名的方式进行调用。静态属性不允许对象使用->操作符调用。

2.静态方法中，$this伪变量不允许使用。可以使用self，parent，static在内部调用静态方法与属性。

···

class Car {

private static $speed = 10;

public static function getSpeed() {

return self::$speed;

}

public static function speedUp() {

return self::$speed+=10;

}

}

class BigCar extends Car {

public static function start() {

parent::speedUp();

}

}

BigCar::start();

echo BigCar::getSpeed();

···

7.访问控制详解

定义为公有的类成员可以在任何地方被访问。被定义为受保护的类成员则可以被其自身以及其子类和父类访问。被定义为私有的类成员则只能被其定义所在的类访问。

如果构造函数定义成了私有方法，则不允许直接实例化对象了，这时候一般通过静态方法进行实例化，在设计模式中会经常使用这样的方法来控制对象的创建，比如单例模式只允许有一个全局唯一的对象。

···

class Car {

private function \_\_construct() {

echo 'object create';

}

private static $\_object = null;

public static function getInstance() {

if (empty(self::$\_object)) {

self::$\_object = new Car();

//内部方法可以调用私有方法，因此这里可以创建对象

}

return self::$\_object;

}

}

//$car = new Car(); //这里不允许直接实例化对象

$car = Car::getInstance(); //通过静态方法来获得一个实例

···

7.重载

PHP中的重载指的是动态的创建属性与方法，是通过魔术方法来实现的。属性的重载通过\_\_set，\_\_get，\_\_isset，\_\_unset来分别实现对不存在属性的赋值、读取、判断属性是否设置、销毁属性。

四 字符串函数

1.去除空格

1.Trim():去除两边空格

2.ltrim():去除左边空格

3.rtrim():去除右边空格

2.计算字符串长度

1.strlen(str)：计算英文长度准确

2.mb\_strlen(str，code):计算中文长度

···

Echo mb\_strlen（‘我是谁’，‘UTF8’）；

···

3.截取字符串

1.substr(字符串变量,开始截取的位置，截取个数):英文截取

2.mb\_substr(字符串变量,开始截取的位置，截取个数,code):中文截取

4.查找字符串

1.strpos(要处理的字符串, 要定位的字符串, 定位的起始位置[可选])

···

$str = 'I want to study at imooc';

$pos = strpos($str, 'imooc');

echo $pos;//结果显示19，表示从位置0开始，imooc在第19个位置开始出现

···

5.替换字符串

1.str\_replace(要查找的字符串, 要替换的字符串, 被搜索的字符串, 替换进行计数[可选])

备注：查找不区分大小写

···

$str = 'I want to learn js';

$replace = str\_replace('js', 'php', $str);

echo $replace;//结果显示I want to learn php

···

6.格式化字符串

1.sprintf(格式, 要转化的字符串)

···

$str = '99.9';

$result = sprintf('%01.2f', $str);

echo $result;//结果显示99.90

1、这个 % 符号是开始的意思，写在最前面表示指定格式开始了。 也就是 "起始字符", 直到出现 "转换字符" 为止，就算格式终止。

2、跟在 % 符号后面的是 0， 是 "填空字元" ，表示如果位置空着就用0来填满。

3、在 0 后面的是1，这个 1 是规定整个所有的字符串占位要有1位以上(小数点也算一个占位)。如果把 1 改成 6，则 $result的值将为 099.90因为，在小数点后面必须是两位，99.90一共5个占位，现在需要6个占位，所以用0来填满。

4、在 %01 后面的 .2 （点2） 就很好理解了，它的意思是，小数点后的数字必须占2位。 如果这时候，$str 的值为9.234,则 $result的值将为9.23.为什么4 不见了呢？ 因为在小数点后面，按照上面的规定，必须且仅能占2位。 可是 $str 的值中，小数点后面占了3位，所以，尾数4被去掉了，只剩下 23。

5、最后，以 f "转换字符" 结尾。

···

7.合并/分隔字符串

1.implode(分隔符[可选], 数组)

返回值：把数组元素组合为一个字符串

2.explode(分隔符[可选], 字符串)

返回值：函数返回由字符串组成的数组

···

$arr = array('Hello', 'World!');

$result = implode('', $arr);

print\_r($result);//结果显示Hello World!

$str = 'apple,banana';

$result = explode(',', $str);

print\_r($result);//结果显示array('apple','banana')

···

8.特殊字符串转义

1.addslashes()函数说明：用于对特殊字符加上转义字符，返回一个字符串

返回值：一个经过转义后的字符串

···

$str = "what's your name?";

echo addslashes($str);//输出：what\'s your name?

···

五 正则表达式

1.正则表达式介绍

正则表达式是对字符串进行操作的一种逻辑公式，就是用一些特定的字符组合成一个规则字符串，称之为正则匹配模式。

···

$p = '/apple/';

$str = "apple banna";

if (preg\_match($p, $str)) {

echo 'matched';

}

其中字符串'/apple/'就是一个正则表达式，他用来匹配源字符串中是否存在apple字符串。

PHP中使用PCRE库函数进行正则匹配，比如上例中的preg\_match用于执行一个正则匹配，常用来判断一类字符模式是否存在。

···

六 会话控制

1.cookie简介

Cookie是存储在客户端浏览器中的数据，我们通过Cookie来跟踪与存储用户数据。一般情况下，Cookie通过HTTP headers从服务端返回到客户端。多数web程序都支持Cookie的操作，因为Cookie是存在于HTTP的标头之中，所以必须在其他信息输出以前进行设置，类似于header函数的使用限制。

PHP通过setcookie函数进行Cookie的设置，任何从浏览器发回的Cookie，PHP都会自动的将他存储在$\_COOKIE的全局变量之中，因此我们可以通过$\_COOKIE['key']的形式来读取某个Cookie值。

PHP中的Cookie具有非常广泛的使用，经常用来存储用户的登录信息，购物车等，且在使用会话Session时通常使用Cookie来存储会话id来识别用户，Cookie具备有效期，当有效期结束之后，Cookie会自动的从客户端删除。同时为了进行安全控制，Cookie还可以设置域跟路径，我们会在稍后的章节中详细的讲解他们。

2.设置cookie

PHP设置Cookie最常用的方法就是使用setcookie函数，setcookie具有7个可选参数，我们常用到的为前5个：

1. name(cookie名):可以通过$\_COOKIE[‘name’]进行访问
2. value(cookie值)
3. Expire(过期时间):Unix时间戳格式，默认为0，表示浏览器关闭即失效
4. Path(有效路径):如果路径设置为‘/’，表示整个网站有效
5. Domain(有效域):默认整个域名有效，如果设置了'www.imooc.com',则只在www子域中有效

···

$value = 'test';

setcookie("TestCookie", $value);

setcookie("TestCookie", $value, time()+3600); //有效期一小时

setcookie("TestCookie", $value, time()+3600, "/path/", "imooc.com"); //设置路径与域

···

PHP中还有一个设置Cookie的函数setrawcookie，setrawcookie跟setcookie基本一样，唯一的不同就是value值不会自动的进行urlencode，因此在需要的时候要手动的进行urlencode。

···

setrawcookie('cookie\_name', rawurlencode($value), time()+60\*60\*24\*365);

···

因为Cookie是通过HTTP标头进行设置的，所以也可以直接使用header方法进行设置。

···

header("Set-Cookie:cookie\_name=value");

···

3.cookie的删除和过期时间

在PHP中删除cookie也是采用setcookie函数来实现。

···

Setcookie(‘test’，‘’，time()-1)；

···

可以看到将cookie的过期时间设置到当前时间之前，则该cookie会自动失效，也就达到了删除cookie的目的。之所以这么设计是因为cookie是通过HTTP的标头来传递的，客户端根据服务端返回的Set-Cookie段来进行cookie的设置，如果删除cookie需要使用新的Del-Cookie来实现，则HTTP头就会变得复杂，实际上仅通过Set-Cookie就可以简单明了的实现Cookie的设置、更新与删除。

···

header("Set-Cookie:test=1393832059; expires=".gmdate('D, d M Y H:i:s \G\M\T', time()-1));

···

这里用到了gmdate，用来生成格林威治标准时间，以便排除时差的影响。

4.cookie的有效路径

cookie中的路径用来控制设置的cookie在哪个路径下有效，默认为'/'，在所有路径下都有，当设定了其他路径之后，则只在设定的路径以及子路径下有效

···

setcookie('test', time(), 0, '/path');

上面的设置会使test在/path以及子路径/path/abc下都有效，但是在根目录下就读取不到test的cookie值。

···

5.session与cookie的异同

cookie将数据存储在客户端，建立起用户与服务器之间的联系，通常可以解决很多问题，但是cookie仍然具有一些局限：

cookie相对不是太安全，容易被盗用导致cookie欺骗

单个cookie的值最大只能存储4k

每次请求都要进行网络传输，占用带宽

session是将用户的会话数据存储在服务端，没有大小限制，通过一个session\_id进行用户识别，PHP默认情况下session id是通过cookie来保存的，因此从某种程度上来说，seesion依赖于cookie。但这不是绝对的，session id也可以通过参数来实现，只要能将session id传递到服务端进行识别的机制都可以使用session。

···

<?php

//开始使用session

session\_start();

//设置一个session

$\_SESSION['test'] = time();

//显示当前的session\_id

echo "session\_id:".session\_id();

echo "<br>";

//读取session值

echo $\_SESSION['test'];

//销毁一个session

// unset($\_SESSION['test']);

echo "<br>";

var\_dump($\_SESSION);

···

6.session的使用

在PHP中使用session非常简单，先执行session\_start方法开启session，然后通过全局变量$\_SESSION进行session的读写。

···

Session\_start();

$\_SESSION[‘test’] = time();

Var\_dump($\_SESSION[‘test’]);

···

session会自动的对要设置的值进行encode与decode，因此session可以支持任意数据类型，包括数据与对象等

···

session\_start();

$\_SESSION['ary'] = array('name' => 'jobs');

$\_SESSION['obj'] = new stdClass();

var\_dump($\_SESSION);

···

7.删除和销毁session

删除某个session值可以使用PHP的unset函数，删除后就会从全局变量$\_SESSION中去除，无法访问。

···

session\_start();

$\_SESSION['name'] = 'jobs';

unset($\_SESSION['name']);

echo $\_SESSION['name']; //提示name不存在

···

如果要删除所有的session，可以使用session\_destroy函数销毁当前session，session\_destroy会删除所有数据，但是session\_id仍然存在。

···

session\_start();

$\_SESSION['name'] = 'jobs';

$\_SESSION['time'] = time();

session\_destroy();

···

值得注意的是，session\_destroy并不会立即的销毁全局变量$\_SESSION中的值，只有当下次再访问的时候，$\_SESSION才为空，因此如果需要立即销毁$\_SESSION，可以使用unset函数。

···

session\_start();

$\_SESSION['name'] = 'jobs';

$\_SESSION['time'] = time();

unset($\_SESSION);

session\_destroy();

var\_dump($\_SESSION); //此时已为空

···

如果需要同时销毁cookie中的session\_id，通常在用户退出的时候可能会用到，则还需要显式的调用setcookie方法删除session\_id的cookie值。

8.使用session存储登录信息

session可以用来存储多种类型的数据，因此具有很多的用途，常用来存储用户的登录信息，购物车数据，或者一些临时使用的暂存数据等。

用户在登录成功以后，通常可以将用户的信息存储在session中，一般的会单独的将一些重要的字段单独存储，然后所有的用户信息独立存储。

一般来说，登录信息既可以存储在sessioin中，也可以存储在cookie中，他们之间的差别在于session可以方便的存取多种数据类型，而cookie只支持字符串类型，同时对于一些安全性比较高的数据，cookie需要进行格式化与加密存储，而session存储在服务端则安全性较高。

七 文件操作系统

1.读取文件内容

最简单的读取文件的函数为file\_get\_contents，可以将整个文件全部读取到一个字符串中

···

$content = file\_get\_contents('./test.txt', null, null, 100, 500);

···

2.判断文件是否存在

PHP中常用来判断文件存在的函数有两个is\_file与file\_exists.

···

$filename = './test.txt';

if (file\_exists($filename)) {

echo file\_get\_contents($filename);

}

···

如果只是判断文件存在，使用file\_exists就行，file\_exists不仅可以判断文件是否存在，同时也可以判断目录是否存在，从函数名可以看出，is\_file是确切的判断给定的路径是否是一个文件。

···

$filename = './test.txt';

if (is\_file($filename)) {

echo file\_get\_contents($filename);

}

···

更加精确的可以使用is\_readable与is\_writeable在文件是否存在的基础上，判断文件是否可读与可写。

···

$filename = './test.txt';

if (is\_writeable($filename)) {

file\_put\_contents($filename, 'test');

}

if (is\_readable($filename)) {

echo file\_get\_contents($filename);

}

···

3.获取文件的属性——001

文件有很多元属性，包括：文件的所有者、创建时间、修改时间、最后的访问时间等。

···

fileowner：获得文件的所有者

filectime：获取文件的创建时间

filemtime：获取文件的修改时间

fileatime：获取文件的访问时间

$filename = '/data/webroot/usercode/code/resource/test.txt';

echo '所有者：'.fileowner($filename).'<br>';

echo '创建时间：'.filectime($filename).'<br>';

echo '修改时间：'.filemtime($filename).'<br>';

echo '最后访问时间：'.fileatime($filename).'<br>';

//给$mtime赋值为文件的修改时间

$mtime = filemtime($filename);

//通过计算时间差 来判断文件内容是否有效

if (time() - $mtime > 3600) {

echo '<br>缓存已过期';

} else {

echo file\_get\_contents($filename);

}

···

4.获取文件的属性——002

通过filesize函数可以取得文件的大小，文件大小是以字节数表示的。

···

$filename = '/data/webroot/usercode/resource/test.txt';

$size = filesize($filename);

···

如果要转换文件大小的单位，可以自己定义函数来实现。

···

function getsize($size, $format = 'kb') {

$p = 0;

if ($format == 'kb') {

$p = 1;

} elseif ($format == 'mb') {

$p = 2;

} elseif ($format == 'gb') {

$p = 3;

}

$size /= pow(1024, $p);

return number\_format($size, 3);

}

$filename = '/data/webroot/usercode/code/resource/test.txt';

$size = filesize($filename);

$size = getsize($size, 'kb'); //进行单位转换

echo $size.'kb';

···

值得注意的是，没法通过简单的函数来取得目录的大小，目录的大小是该目录下所有子目录以及文件大小的总和，因此需要通过递归的方法来循环计算目录的大小。

八 日期和时间

1.获取当前时间Unix时间戳

UNIX 时间戳（英文叫做：timestamp）是 PHP 中关于时间与日期的一个很重要的概念，它表示从 1970年1月1日 00:00:00 到当前时间的秒数之和。

PHP提供了内置函数 time() 来取得服务器当前时间的时间戳

···

$time = time();

echo $time;//1396193923，这个数字表示从1970年1月1日 00:00:00 到我输出这个脚本时经历了1396193923秒

···

2.取得当前的日期

函数说明：date(时间戳的格式, 规定时间戳【默认是当前的日期和时间，可选】)

···

//设置默认的时区

date\_default\_timezone\_set('Asia/Shanghai');

//date函数，第二个参数取默认值的情况

echo date("Y-m-d");//2014-03-30

//date函数，第二个参数有值的情况

echo date("Y-m-d",'1396193923');//2014-03-30,1396193923表示2014-03-30的unix时间戳

echo date('Y-m-d H:i:s',time());

···

3.取得日期的时间戳

PHP提供了内置函数strtotime实现功能：获取某个日期的时间戳，或获取某个时间的时间戳。

···

echo strtotime('2014-04-29');//1398700800

echo strtotime('2014-04-29 00:00:01');//1398700801

···

4.将格式化的日期字符串转换为Unix时间戳

函数说明：strtotime(要解析的时间字符串, 计算返回值的时间戳【默认是当前的时间，可选】)

···

echo strtotime("now");//相当于将英文单词now直接等于现在的日期和时间，并把这个日期时间转化为unix时间戳。这个效果跟echo time();一样。

echo strtotime("+1 seconds");//相当于将现在的日期和时间加上了1秒，并把这个日期时间转化为unix时间戳。这个效果跟echo time()+1;一样。

echo strtotime("+1 day");//相当于将现在的日期和时间加上了1天。

echo strtotime("+1 week");//相当于将现在的日期和时间加上了1周。

echo strtotime("+1 week 3 days 7 hours 5 seconds");//相当于将现在的日期和时间加上了1周3天7小时5秒。

···

5.格式化格林威治（GMT）标准时间

gmdate 函数能格式化一个GMT的日期和时间，返回的是格林威治标准时（GMT）

···

举个例子，我们现在所在的中国时区是东八区，领先格林威治时间8个小时，有时候也叫GMT+8，那么服务器运行以下脚本返回的时间应该是这样的：

当前时间假定是2014-05-01 15:15:22

echo date('Y-m-d H:i:s', time()); //输出为：2014-05-01 15:15:22

echo gmdate('Y-m-d H:i:s', time()); //输出为：2014-05-01 07:15:22 因为格林威治时间是现在中国时区的时间减去8个小时，所以相对于现在时间要少8个小时

···

九 GD库——图形操作

1. GD库简介

GD指的是Graphic Device，PHP的GD库是用来处理图形的扩展库，通过GD库提供的一系列API，可以对图像进行处理或者直接生成新的图片。

PHP除了能进行文本处理以外，通过GD库，可以对JPG、PNG、GIF、SWF等图片进行处理。GD库常用在图片加水印，验证码生成等方面。

PHP默认已经集成了GD库，只需要在安装的时候开启就行。

···

header("content-type: image/png");

$img=imagecreatetruecolor(100, 100); //绘制画布

$red=imagecolorallocate($img, 0xFF, 0x00, 0x00);

//imagefill() 在 image 图像的坐标 x，y（图像左上角为 0, 0）处用 color 颜色执行区域填充（即与 x, y 点颜色相同且相邻的点都会被填充）。

imagefill($img, 0, 0, $red);

imagepng($img);

imagedestroy($img);

···

2. 绘制线条

要对图形进行操作，首先要新建一个画布，通过imagecreatetruecolor函数可以创建一个真彩色的空白图片：

···

$img = imagecreatetruecolor(100, 100);

···

GD库中对于画笔所用的颜色，需要通过imagecolorallocate函数进行分配，通过参数设定RGB的颜色值来确定画笔的颜色：

···

$red = imagecolorallocate($img, 0xFF, 0x00, 0x00);

···

然后我们通过调用绘制线段函数imageline进行线条的绘制，通过指定起点跟终点来最终得到线条。

···

imageline($img, 0, 0, 100, 100, $red);

···

线条绘制好以后，通过header与imagepng进行图像的输出。

···

header("content-type: image/png");

imagepng($img);

···

最后可以调用imagedestroy释放该图片占用的内存。

···

imagedestroy($img);

···

通过上面的步骤，可以发现PHP绘制图形非常的简单，但很多时候我们不只是需要输出图片，可能我们还需要得到一个图片文件，可以通过imagepng函数指定文件名将绘制后的图像保存到文件中。

···

imagepng($img, 'img.png');

···

3.图形中绘制文字

**绘制画布**

**确定颜色**

**绘制样式**

**输出格式**

**释放内存**

跟绘制线条类似，首先需要新建一个图片与初始化颜色。

然后使用imagestring函数来进行文字的绘制，这个函数的参数很多：imagestring ( resource $image , int $font , int $x , int $y , string $s , int $col )，可以通过$font来设置字体的大小，x,y设置文字显示的位置，$s是要绘制的文字,$col是文字的颜色。

···

$img = imagecreatetruecolor(100, 100);

$red = imagecolorallocate($img, 0xFF, 0x00, 0x00);

imagestring($img, 5, 0, 0, "Hello world", $red);

header("content-type: image/png");

imagepng($img);

imagedestroy($img);

···

4.输出图像文件

通过imagepng可以直接输出图像到浏览器，但是很多时候，我们希望将处理好的图像保存到文件，以便可以多次使用。通过指定路径参数将图像保存到文件中。

使用imagepng可以将图像保存成png格式，如果要保存成其他格式需要使用不同的函数，使用imagejpeg将图片保存成jpeg格式，imagegif将图片保存成gif格式，需要说明的是，imagejpeg会对图片进行压缩，因此还可以设置一个质量参数。

···

<?php

header('content-type:text/html;charset=utf-8');

$img = imagecreatetruecolor(100, 100);

$red = imagecolorallocate($img, 0xFF, 0x00, 0x00);

imagestring($img, 5, 0, 0, "Hello world", $red);

$filename = 'img.png';

//在这里将图片保存到$filename文件中

imagepng($img,'img.png');

imagedestroy($img);

if (file\_exists($filename)) {

echo '文件保存成功';

}

···

5.生成图像验证码

简单的验证码其实就是在图片中输出了几个字符，通过我们前面章节讲到的imagestring函数就能实现。

但是在处理上，为了使验证码更加的安全，防止其他程序自动识别，因此常常需要对验证码进行一些干扰处理，通常会采用绘制一些噪点，干扰线段，对输出的字符进行倾斜、扭曲等操作。

可以使用imagesetpixel绘制点来实现噪点干扰，但是只绘制一个点的作用不大，因此这里常常会使用循环进行随机绘制。

···

<?php

$img = imagecreatetruecolor(100, 40);

$black = imagecolorallocate($img, 0x00, 0x00, 0x00);

$green = imagecolorallocate($img, 0x00, 0xFF, 0x00);

$white = imagecolorallocate($img, 0xFF, 0xFF, 0xFF);

imagefill($img,0,0,$white);

//生成随机的验证码

$code = '';

for($i = 0; $i < 4; $i++) {

$code .= rand(0, 9);

}

imagestring($img, 5, 10, 10, $code, $black);

//加入噪点干扰

for($i=0;$i<50;$i++) {

imagesetpixel($img, rand(0, 100) , rand(0, 100) , $black);

imagesetpixel($img, rand(0, 100) , rand(0, 100) , $green);

}

//输出验证码

header("content-type: image/png");

imagepng($img);

imagedestroy($img);

···

6.图片加水印

给图片添加水印的方法一般有两种，一种是在图片上面加上一个字符串，另一种是在图片上加上一个logo或者其他的图片。

因为这里处理的是已经存在的图片，所以可以直接从已存在的图片建立画布，通过imagecreatefromjpeg可以直接从图片文件创建图像。

···

$im = imagecreatefromjpeg($filename);

···

创建图像对象以后，我们就可以通过前面的GD函数，绘制字符串到图像上。如果要加的水印是一个logo图片，那么就需要再建立一个图像对象，然后通过GD函数imagecopy将logo的图像复制到源图像中。

···

$logo = imagecreatefrompng($filename);

imagecopy($im, $logo, 15, 15, 0, 0, $width, $height);

···

当将logo图片复制到原图片上以后，将加水印后的图片输出保存就完成了加水印处理。

···

imagejpeg($im, $filename);

···

十 异常处理

1. 抛出一个异常

从PHP5开始，PHP支持异常处理，异常处理是面向对象一个重要特性，PHP代码中的异常通过throw抛出，异常抛出之后，后面的代码将不会再被执行。

既然抛出异常会中断程序执行，那么为什么还需要使用异常处理？

异常抛出被用于在遇到未知错误，或者不符合预先设定的条件时，通知客户程序，以便进行其他相关处理，不至于使程序直接报错中断。

当代码中使用了try catch的时候，抛出的异常会在catch中捕获，否则会直接中断。

1、基本语法

···

try{

//可能出现错误或异常的代码

//catch表示捕获，Exception是php已定义好的异常类

} catch(Exception $e){

//对异常处理，方法：

//1、自己处理

//2、不处理，将其再次抛出

}

···

2、处理处理程序应当包括：

Try - 使用异常的函数应该位于 "try" 代码块内。如果没有触发异常，则代码将照常继续执行。但是如果异常被触发，会抛出一个异常。

Throw - 这里规定如何触发异常。注意：每一个 "throw" 必须对应至少一个 "catch"，当然可以对应多个"catch"

Catch - "catch" 代码块会捕获异常，并创建一个包含异常信息的对象。

···

//创建可抛出一个异常的函数

function checkNum($number){

if($number>1){

throw new Exception("异常提示-数字必须小于等于1");

}

return true;

}

//在 "try" 代码块中触发异常

try{

checkNum(2);

//如果异常被抛出，那么下面一行代码将不会被输出

echo '如果能看到这个提示，说明你的数字小于等于1';

}catch(Exception $e){

//捕获异常

echo '捕获异常: ' .$e->getMessage();

}

上面代码将获得类似这样一个错误：

捕获异常:: 异常提示-数字必须小于等于1

例子解释：

上面的代码抛出了一个异常，并捕获了它：

创建 checkNum() 函数。它检测数字是否大于 1。如果是，则抛出一个异常。

在 "try" 代码块中调用 checkNum() 函数。

checkNum() 函数中的异常被抛出

"catch" 代码块接收到该异常，并创建一个包含异常信息的对象 ($e)。

通过从这个 exception 对象调用 $e->getMessage()，输出来自该异常的错误消息

···

2. 异常处理类

PHP具有很多异常处理类，其中Exception是所有异常处理的基类。

Exception具有几个基本属性与方法，其中包括了：

message 异常消息内容

code 异常代码

file 抛出异常的文件名

line 抛出异常在该文件的行数

其中常用的方法有：

getTrace 获取异常追踪信息

getTraceAsString 获取异常追踪信息的字符串

getMessage 获取出错信息

如果必要的话，可以通过继承Exception类来建立自定义的异常处理类。

···

//自定义的异常类，继承了PHP的异常基类Exception

class MyException extends Exception {

function getInfo() {

return '自定义错误信息';

}

}

try {

//使用异常的函数应该位于 "try" 代码块内。如果没有触发异常，则代码将照常继续执行。但是如果异常被触发，会抛出一个异常。

throw new MyException('error');//这里规定如何触发异常。注意：每一个 "throw" 必须对应至少一个 "catch"，当然可以对应多个"catch"

} catch(Exception $e) {//"catch" 代码块会捕获异常，并创建一个包含异常信息的对象

echo $e->getInfo();//获取自定义的异常信息

echo $e->getMessage();//获取继承自基类的getMessage信息

}

···

3.捕获异常信息

在了解了异常处理的基本原理之后，我们可以通过try catch来捕获异常，我们将执行的代码放在try代码块中，一旦其中的代码抛出异常，就能在catch中捕获。

这里我们只是通过案例来了解try catch的机制以及异常捕获的方法，在实际应用中，不会轻易的抛出异常，只有在极端情况或者非常重要的情况下，才会抛出异常，抛出异常，可以保障程序的正确性与安全，避免导致不可预知的bug。

一般的异常处理流程代码为：

···

try {

throw new Exception('wrong');

} catch(Exception $ex) {

echo 'Error:'.$ex->getMessage().'<br>';

echo $ex->getTraceAsString().'<br>';

}

echo '异常处理后，继续执行其他代码';

<?php

try {

throw new Exception('wrong');

} catch(Exception $ex) {

$msg = 'Error:'.$ex->getMessage()."\n";

$msg.= $ex->getTraceAsString()."\n";

$msg.= '异常行号：'.$ex->getLine()."\n";

$msg.= '所在文件：'.$ex->getFile()."\n";

//将异常信息记录到日志中PHP异常处理之 file\_put\_contents('error.log', $msg);

}

···