目录

[01、初学者的学习路线 2](#_Toc513541717)

[02、PHP简介 2](#_Toc513541718)

[03、PHP的基本语法 2](#_Toc513541719)

[04、PHP变量 3](#_Toc513541720)

[05、echo和print语句 3](#_Toc513541721)

[06、EOF使用说明 3](#_Toc513541722)

[07、数据类型 4](#_Toc513541723)

[08、常量 4](#_Toc513541724)

[09、字符串变量 4](#_Toc513541725)

[10、PHP运算符 4](#_Toc513541726)

[11、PHP条件语句 6](#_Toc513541727)

[12、PHP数组 6](#_Toc513541728)

[13、超级全局变量 7](#_Toc513541729)

[14、PHP循环 8](#_Toc513541730)

[15、PHP函数 8](#_Toc513541731)

[16、PHP魔术变量（不区分大小写） 9](#_Toc513541732)

[17、PHP命名空间 9](#_Toc513541733)

[18、面向对象 9](#_Toc513541734)

# 01、初学者的学习路线

(1) 熟悉HTML/CSS/JS等网页基本元素，完成阶段可自行制作简单的网页，对元素属性相对熟悉。

(2) 理解动态语言的概念和运做机制，熟悉基本的PHP语法。

(3) 学习如何将PHP与HTML结合起来，完成简单的动态页面。

(4) 接触学习MySQL，开始设计数据库。

(5) 不断巩固PHP语法，熟悉大部分的PHP常用函数，理解面向对象编程，MySQL优化，以及一些模板和框架。

(6) 最终完成一个功能齐全的动态站点。

# 02、PHP简介

* PHP（全称：Hypertext Preprocessor，超文本预处理器）是一种通用开源脚本语言
* PHP脚本在服务器上执行
* PHP文件可以包含文本、HTML、JavaScript代码和PHP代码。
* php服务器组建：使用WampServer，它集成了php、Apache、Mysql。
* PHP能做什么？

1、PHP可以生成动态页面内容

2、PHP可以创建、打开、读取、写入、关闭服务器上的文件

3、PHP可以收集表单数据

4、PHP可以发送和接受cookies

5、PHP 可以添加、删除、修改您的数据库中的数据

6、PHP 可以限制用户访问网站上的一些页面

7、PHP 可以加密数据

8、通过 PHP，不再限于输出 HTML，可以输出图像、PDF 文件，甚至 Flash 电影，还可以输出任意的文本，比如 XHTML 和 XML。

# 03、PHP的基本语法

* 开始 <?php

PHP\_BODY;

?> 结尾

* 每个句尾都要加；号
* // 单行注释号
* /\*……\*/多行注释号

# 04、PHP变量

* 以$开始，后面跟变量名称
* 变量名区分大小写
* local：变量局部作用域关键字（在函数内部声明的变量为局部变量，只能在函数内部使用）
* global：变量全局作用域关键字（所有在函数外部定义的变量为全局变量，全局变量可

以被脚本中的任何部分访问，但访问时要加上global关键字。）

在函数内部定义一个全局变量示例：

<?php

$x = 10;

$y = 10;

function sum(){

global $x,$y;

$GLOBALS['z'] = $x + $y; //用$GLOBALS[‘z’]在函数内部定义一个全局变量z

}

sum();

echo $z;

?>

输出：20

* static：当一个函数运行完时，所有的变量都会被删除，当一个变量值需要记忆时，用static关键字标注变量时，变量会保留前一次被调用时的值。
* parameter：参数作用域。

# 05、echo和print语句

* echo：可以输出一个或多个字符串（字符串可以包含HTML标签且能够翻译出来，变量不可以，会报错），各字符串之间用逗号隔开。

例：<?php

$first = 1;

echo "hello world<br>",$first; //<br>换行符

?>

输出：hello world  
1

* print：只能输出一个字符串（字符串可以包含HTML标签）。

注意：echo没有返回值，print有返回值1，返回值一般用于判断输出是否成功。

# 06、EOF使用说明

* PHP EOF(heredoc)是一种在命令行shell（如sh、csh、ksh、bash、PowerShell和zsh）和程序语言（像Perl、PHP、Python和Ruby）里定义一个字串的方法。
* 注意：1、以<<<EOF开头，以EOF结尾，结尾标记必须占一行且顶头写。

2、位于标记中的变量可以被正常解析，但是函数不能。

例：<?php

echo <<<EOF //标记不仅限于EOF，但最好是大写。

<h1>我的第一个标题</h1>

<p>我的第一个段落。</p>

EOF;// 结束需要独立一行且前后不能空格

?>

输出：我的第一个标题

我的第一个段落。

# 07、数据类型

* 字符串（string）：在单引号和双引号之间的部分。
* 整形（integer）：没有小数的数字。
* 浮点型（float）：带有小数的数字。
* 布尔型（Boolean）：TURE和FALSE
* 数组（array）：由多个值组成的变量。
* 对象（object）：
* 空值（NULL）：NULL
* var\_dump（）：函数返回变量的数据类型和数值。

# 08、常量

概述：一个简单值的标识符，在值在脚本中不能被改变。

* 常量用关键字const定义，定义和使用的时候不用$符号，例如： const x=1；
* 常量是默认是全局的，可以在脚本中任何地方访问，且不用标注global关键字。
* 设置常量：define(name, value [, bool $case\_insensitive=false or ture])

name：常量的名称。注意：当name是字符型时，应加引号

value：常量的值。

bool $case\_insensitive：常量是否区分大小写，默认区分，设置时直接写false 或true，bool $case\_insensitive只是一个描述。

# 09、字符串变量

* . （点）并置运算符：连接两个字符串。
* strlen( )：获得字符串长度。
* strpos(string1，string2)：在字符串string1中查找字符串string2的第一个匹配的位置，

# 10、PHP运算符

* PHP算术运算符

+（加）、-（减）、\*（乘）、/（除）、%（除法求余）、.（并置）

* PHP赋值运算符

=（赋值）、+=（加）、-=（减）、\*=（乘）、/=（除）、%=（除法取余）、.=（并置）

* PHP递增递减运算符

++x（预递增，x加1后返回x）

x++（后递增，返回x后x加1）

--x（预递减）

x--（后递减）

* PHP比较运算符

|  |  |  |
| --- | --- | --- |
| 运算符 | 名称 | 描述 |
| x == y | 等于 | 如果 x 等于 y，则返回 true |
| x === y | 绝对等于 | 如果 x 等于 y，且它们类型相同，则返回 true |
| x != y | 不等于 | 如果 x 不等于 y，则返回 true |
| x <> y | 不等于 | 如果 x 不等于 y，则返回 true |
| x !== y | 绝对不等于 | 如果 x 不等于 y，或它们类型不相同，则返回 true |
| x > y | 大于 | 如果 x 大于 y，则返回 true |
| x < y | 小于 | 如果 x 小于 y，则返回 true |
| x >= y | 大于等于 | 如果 x 大于或者等于 y，则返回 true |
| x <= y | 小于等于 | 如果 x 小于或者等于 y，则返回 true |

* PHP逻辑运算符

|  |  |  |
| --- | --- | --- |
| 运算符 | 名称 | 描述 |
| x and y | 与 | 如果 x 和 y 都为 true，则返回 true |
| x or y | 或 | 如果 x 和 y 至少有一个为 true，则返回 true |
| x xor y | 异或 | 如果 x 和 y 有且仅有一个为 true，则返回 true |
| x && y | 与 | 如果 x 和 y 都为 true，则返回 true |
| x || y | 或 | 如果 x 和 y 至少有一个为 true，则返回 true |
| ! x | 非 | 如果 x 不为 true，则返回 true |

* PHP数组运算符

|  |  |  |
| --- | --- | --- |
| 运算符 | 名称 | 描述 |
| x + y | 集合 | x 和 y 的集合 |
| x == y | 相等 | 如果 x 和 y 具有相同的键/值对，则返回 true |
| x === y | 恒等 | 如果 x 和 y 具有相同的键/值对，且顺序相同类型相同，则返回 true |
| x != y | 不相等 | 如果 x 不等于 y，则返回 true |
| x <> y | 不相等 | 如果 x 不等于 y，则返回 true |
| x !== y | 不恒等 | 如果 x 不等于 y，则返回 true |

* 三元运算符：(expr1) ? (expr2) : (expr3)

expr1 求值为 TRUE 时的值为 expr2，在 expr1 求值为 FALSE 时的值为 expr3

# 11、PHP条件语句

* if（条件）{

条件成立时执行代码代码；

}

* if（条件）{

条件成立时执行的代码；

}

else{

条件不成立时执行的代码；

}

* if（条件）{

条件成立时执行的代码；

}

elseif（条件）{

条件成立时执行的代码；

}

else{

条件不成立时执行的代码；

}

* switch语句

<?php

switch (n) {

case label1: 如果 n=label1，此处代码将执行并跳出循环;

break;

case label2: 如果 n=label2，此处代码将执行并跳出循环;

break;

default: 如果 n 既不等于 label1 也不等于 label2，此处代码将执行;

}

?>

工作原理：首先对一个简单的表达式 n（通常是变量）进行一次计算。将表达式的值与结构中每个 case 的值进行比较。如果存在匹配，则执行与 case 关联的代码。代码执行后，使用 break 来阻止代码跳入下一个 case 中继续执行。default 语句用于不存在匹配（即没有 case 为真）时执行。

# 12、PHP数组

* 数值数组：array(value，value)默认数字ID（key）键的数组
* 关联数组：array(”key”=>”value”, ”key”=>”value”)ID（key）键的数组
* 多维数组：数组里面包括数组的情况。
* 数组排序：

sort（）：对数组进行升序排列

rsort（）：对数组进行降序排列

asort（）：根据关联数组的值，对数组进行升序排列

ksort（）：根据关系数组的键，对数组进行升序排列

arsort（）：根据关联数组的值，对数组进行降序排列

krsort（）：根据关联数组的键，对数组进行降序排列

# 13、超级全局变量

概述：超全局变量是在全部作用域中始终可用的内置变量级。

* $GLOBALS：是PHP的一个超级全局变量组，在一个PHP脚本的全部作用域中都可以访问；是一个包含了全部变量的全局组合数组，变量的名字就是数组的键。

例：<?php

$x = 75; $y = 25;

function addition() {

$GLOBALS['z'] = $GLOBALS['x'] + $GLOBALS['y'];

}

addition();

echo $z;

?>

输出：100

* $\_SERVER：是一个包含了如头信息（header）、路径（path）、以及脚本位置（script locations）等等信息的数组。这个变量还有许多参数，暂略。
* $\_REQUEST：用于收集HTM表单提交的数据。

注意：打开HTML文件时应该用本地链接（http://localhost/my%20php/form.html），用

（file:///C:/wamp/www/my%20php/form.html）不行。

例：创建名为form.html的HTML文件

<html>

<body>

//form是HTML表单标志，action属性的值是处理程序的程序名。method属性用来定义处理程序从表单中获得信息的方式，取值为POST或GET，GET方式是处理程序从当前HTML文档中获取数据，数据量限制在1KB一下；POST与GET相反，它是当前的HTML文档把数据传送给处理程序，传送数据量可以很大。

<form action="<?php "welcome.php" method="post" >

//input定义一个输入区，type属性是控件类型，text创建一个文本输入框，submit提交到服务器按钮；name属性是控件名称，用来提交表单时传递表单的值，PHP脚本中要用的变量名要和name的名字相同，这样才可以将表单中输入的值传到PHP脚本需要的地方去。

Name: <input type="text" name="fname">

<input type="submit" value=”take up”>

</form>

</body></html>

创建名为welcom.php的PHP文件

//$\_REQUEST用于收集刚才输入表单的值，并用echo打印出来。

<?php

echo $\_POST[“fname”], “’s first PHP and HTML”

?>

</body>

</html>

* $\_POST：收集表单数据，在HTML标签指定该属性：method=“post“。
* $\_GET：收集表单数据，在HTML标签指定该属性：method=“get“。
* $\_FILES
* $\_ENV
* $\_COOKIE
* $\_SESSION

# 14、PHP循环

* while循环：循环重复执行代码块，直到条件不成立

while（条件）{

要执行的代码；

}

* do……while语句：先执行do中代码，再执行判断条件，条件成立继续执行do中代码，不成立终止，此时执行了一次代码。

do{

要执行的代码；

}

while（条件）；

* for循环

for（初始值；条件（可以有多个，用都好隔开）；增量）{

要执行的代码；

}

* foreach循环：用于遍历数组

foreach（$array as $value）{

要执行的代码；

}

# 15、PHP函数

* 内建函数
* 创建函数：

function function\_name( ){

function\_body

}

* return语句：返回函数值，

注：因为函数内部定义的变量是局部变量，函数外部不能访问，用return可以返回到函数外部。

# 16、PHP魔术变量（不区分大小写）

* \_\_LINE\_\_：显示当前的行号
* \_\_FILE\_\_：显示当前文件的完整路径和文件名
* \_\_DIR\_\_：文件所在的目录，如果用在被包括的文件中，则返回被包括的文件所在的目录（不返回文件的名字）
* \_\_FUNCTION\_\_：返回函数名称
* \_\_CLASS\_\_：返回该类被定义时的名字
* \_\_TRAIT\_\_：这个方法有问题，先略过
* \_\_METHOD\_\_：返回该方法被定义时的名字
* \_\_NAMESPACE\_\_：返回当前命名空间的名称

# 17、PHP命名空间

概述：命名空间指为变量名创建一个空间，变量名和命名空间的关系就像文件夹和文件夹内文件的关系类似。

* 命名空间通过关键字namespace来声明，且必须在所有的代码之前。
* 可以在同一个文件中创建不同的命名空间。
* 将全局的非命名空间中的代码与命名空间中的代码组合在一起，只能使用大括号形式的语法，全局代码必须用一个不带名称的namespace语句加上大括号括起来，例如：

<?php

namespace MyProject { const

CONNECT\_OK = 1;

class Connection { /\* ... \*/ }

function connect() { /\* ... \*/ }

}

namespace { // 全局代码

session\_start();

$a = MyProject\connect();

echo MyProject\Connection::start();

}

?>

* 子命名空间：

例：namespace MyProject\sub; //定义了MyProject的子空间sub。

# 18、面向对象

* 创建类语法：

class class\_name{

成员变量（或属性）（类的对象由属性而不同）（类的变量名前面要加var）

方法（类的对象有同样的方法）

}

* 用new运算符实例化一个对象。例：$a=new class\_name；
* 类：定义了一件事物的抽象特点。
* 对象：类的实例
* 成员变量：定义在类内部的变量，该变量的值对外不可见，但可以通过成员函数访问，在类被实例化为对象后，该变量即可成为对象的数据。
* 继承（extends）：继承性是子类自动共享父类数据结构和方法的机制，是类之间的一种关系。在定义和实现一个类的时候，可以在一个已经存在的类基础之上来进行，把这个已经存在的类所定义的内容作为自己的内容，并加入若干新的内容。

语法：class child extends parent{

代码，一般是基于父类的扩展功能；

]

* 父类：一个类被其他类继承，可将该类称为父类、基类或超类。
* 子类：一个类继承其他类称为子类或派生类。
* 多态：多态性指相同的函数或方法可作用于多种类型的对象上并获得不同的结果。不同的对象，收到同一消息可以产生不同的结果，这种现象称为多态。
* 重载：函数或者方法有相同的名称，但参数列表不相同，这种同名不同参数的函数或者方法之间，互相称之为重载函数或重载方法。
* 抽象性：抽象性指将具有一致的数据结构（属性）和行为（操作）的对象抽象成类。
* 封装：指将现实世界中存在的某个客体的属性与行为绑在一起，并放置在一个逻辑单元内。
* 构造函数（construct）：主要用来在创建对象是初始化对象，即为对象成员变量赋初始值，总与new运算符一起使用在创建对象的语句中。

语法：function \_\_construct(需要初始化的变量名，这个变量名是类的成员变量){

$this->需要初始化的变量名=需要初始化的变量名;

//$this是伪变量，$this->需要初始化的变量名表示需要初始化的变量名

}

* 调用父类的构造方法：PHP 不会在子类的构造方法中自动的调用父类的构造方法。要执行父类的构造方法，需要在子类的构造方法中调用 parent：：\_\_construct()
* 析构函数（destructor）：析构函数与构造函数相反，当对象结束其生命周期时，系统自动执行析构函数，析构函数往往用来做清理工作，释放占用的工作空间，析构函数没有参数。

语法：function \_\_destruct(){

自动清除对象所占空间；

}

* 访问控制：PHP对属性或方法的访问控制，通过添加关键字来实现的，默认为公有。

public（公有）：公有的类成员可以在任何地方被访问。

protected（受保护）：受保护的类成员可以被其自身以及其子类和父类访问。

private（私有）：私有的类成员只能被其定义所在的类访问。

* 抽象类：任何一个类，如果里面有方法被声明为抽象的，那么这个类就必须声明为抽象的类，定义为抽象的类不能被实例化，但能被继承，但是继承的子类必须定义父类中的所有抽象方法，其中子类实例父类抽象类的时候，类的参数可以多于父类的抽象类，但参数必须要有默认值。

语法：abstract class Abstractfather{

//子类必须要定义的方法（抽象方法）

abstract protected function first（）；

abstract protected function second（）；

//普通方法（非抽象方法）

public function printout（）{

echo “我是非抽象方法”；

}

* 静态方法：静态方法不属于任何一个对象，不需要对象也可以调用；

语法：static 变量（静态属性） 使用方式：类名：：静态属性名

static 方法（静态方法） 使用方式：类名：：静态方法名

静态方法调用静态方法：self：：静态方法名

* final关键字：如果父类中的方法被声明为 final，则子类无法覆盖该方法。如果一个类被声明为 final，则不能被继承。