|  |
| --- |
| **PHP语言和Discuz!X3入门教程**  针对Java开发人员的PHP语言和Discuz!X3的模板、插件和二次开发的教程。  Php只是简要介绍与Java不同的地方，详情请查看php手册。  Discuz!X3同样只是简要介绍，详情查看源码或官方论坛。 |



**目录**

[**PHP语言和Discuz!X3入门教程** 1](#_Toc355871467)

[**目录** 2](#_Toc355871468)

[**第一节 、Php与Java语法不同之处** 3](#_Toc355871469)

[**1.php标记(从HTML中分离)** 3](#_Toc355871470)

[**2.变量的命名（变量名前必须加“$”）** 3](#_Toc355871471)

[**3.变量定义** 3](#_Toc355871472)

[**4.常量定义** 4](#_Toc355871473)

[**5.函数定义与调用** 4](#_Toc355871474)

[**1.展示函数用途的伪代码** 4](#_Toc355871475)

[**2.条件函数(Conditional functions)** 4](#_Toc355871476)

[**3.函数中的函数** 5](#_Toc355871477)

[**4.递归函数** 5](#_Toc355871478)

[**5.函数的动态调用的两种方式** 5](#_Toc355871479)

[**6.类与对象** 5](#_Toc355871480)

[**1.自动加载对象** 5](#_Toc355871481)

[**2.构造函数和析构函数** 6](#_Toc355871482)

[**3.魔术方法** 7](#_Toc355871483)

[**7.其他常用** 7](#_Toc355871484)

[**第二节、PHP常用函数与类库** 8](#_Toc355871485)

[**1. 函数** 8](#_Toc355871486)

[**2. 类** 8](#_Toc355871487)

[**第三节、DX3模板开发** 9](#_Toc355871488)

[**第四节、DX3插件开发** 10](#_Toc355871489)

[**第五节、DX附录** 11](#_Toc355871490)

1. **、Php与Java语法不同之处**

**1.php标记(从HTML中分离)**

长标记和短标记，标记可以插在php页面的任何位置

长标记：

<?php echo '<p>Hello World</p>'; ?>

短标记(php.ini配置short\_open\_tag=On)：

<? echo '<p>Hello World</p>',date(‘Y-m-d H:i:s’); ?>

<?='<p>Hello World</p>' ?>

<?=gmdate('Y-m-d H:i:s')?>

**2.变量的命名（变量名前必须加“$”）**

可以包括“\_”、“A-Z”、“a-z”和“0-9”，必须以“\_”、“A-Z”或“a-z”开头，变量名是区分大小写的。

如：$\_var1,和$var2

~~$1var和var2~~

**预定义变量：**

$GLOBALS — 引用全局作用域中可用的全部变量

$\_SERVER — 服务器和执行环境信息

$\_GET/$HTTP\_GET\_VARS — HTTP GET 变量

$\_POST/$HTTP\_POST\_VARS — HTTP POST 变量

$\_FILES — HTTP 文件上传变量

$\_REQUEST — HTTP Request 变量

$\_SESSION — Session 变量session\_start()

$\_ENV — 环境变量

$\_COOKIE — HTTP Cookies

$php\_errormsg — 前一个错误信息

$HTTP\_RAW\_POST\_DATA — 原生POST数据

$http\_response\_header — HTTP 响应头

$argc — 传递给脚本的参数数目

$argv — 传递给脚本的参数数组

**可变变量：**

<?php  
 $a = 'hello';

$$a = 'world';

echo "$a ${$a}";

echo "$a $hello";

?>

第一个echo与第二个echo输出是一样的结果。

**3.变量定义**

由于php是弱类型语言，所以变量定义时候不需要定义变量类型，赋值后变量类型为其值的类型。

如：$var1 = 1;

$var2 = array(‘kk’,’rr’);

$var3 = “kkkkkk”;

$var4 = false;

从未赋值过的变量值为NULL

**4.常量定义**

以字母或下划线开始，后面跟着任何字母，数字或下划线。用正则表达式是这样表达的：*[a-zA-Z\_\x7f-\xff][a-zA-Z0-9\_\x7f-\xff]\**

define(‘WEB\_ROOT\_PATH’,true)

使用defined判断常量是否已定义

**5.static关键字**

**函数内**static $var;可以记住最后修改的值

**类的属性** MyClass::$var

**类的方法** MyClass::myMethod()

**6.函数定义与调用**

函数名和 PHP 中的其它标识符命名规则相同。有效的函数名以字母或下划线打头，后面跟字母，数字或下划线。可以用正则表达式表示为：[a-zA-Z\_\x7f-\xff][a-zA-Z0-9\_\x7f-\xff]\*。

**1.展示函数用途的伪代码**

<?php  
function foo($arg\_1, $arg\_2, ..., $arg\_n)  
{  
 echo "Example function.\n";  
 return $retval;  
}  
?>

***除非***是下面两个例子中的情况，函数必须在其调用之前定义。

**2.条件函数(Conditional functions)**

<?php  
  
$makefoo = true;  
  
/\* 我们不能在处调用foo()函数，  
 因为它还不存在，但可以调用bar()函数。\*/  
  
bar();  
  
if ($makefoo) {  
 function foo()  
 {  
 echo "I don't exist until program execution reaches me.\n";  
 }  
}  
  
/\* 现在我们可以安全调用函数 foo()了，  
 因为 $makefoo 值为真 \*/  
  
if ($makefoo) foo();  
  
function bar()  
{  
 echo "I exist immediately upon program start.\n";  
}  
  
?>

**3.函数中的函数**

<?php  
function foo()  
{  
 function bar()  
 {  
 echo "I don't exist until foo() is called.\n";  
 }  
}  
  
/\* 现在还不能调用bar()函数，因为它还不存在 \*/  
  
foo();  
  
/\* 现在可以调用bar()函数了，因为foo()函数  
 的执行使得bar()函数变为已定义的函数 \*/  
  
bar();  
  
?>

**4.递归函数**

<?php  
function recursion($a)  
{  
 if ($a < 20) {  
 echo "$a\n";  
 recursion($a + 1);  
 }  
}  
?>

**5.函数的动态调用的两种方式**

第一种：$func($args)

先求得$func的字符值，再调用与值同名的函数并传递参数$args

第二种：使用系统函数

call\_user\_func,call\_user\_func\_array,call\_user\_method,call\_user\_method\_array

**7.类与对象**

**1.自动加载对象**

<?php  
function \_\_autoload($class\_name) {  
    require\_once $class\_name . '.php';  
}  
  
$obj  = new MyClass1();  
$obj2 = new MyClass2();  
?>

\_\_autoload是一个魔术函数，当某类还没有加载时，会调用此函数进行自动加载。

**2.构造函数和析构函数**

* 构造函数

void **\_\_construct** ([ [mixed](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/language.pseudo-types.html#language.types.mixed) $args [, $... ]] )

**Note**: 如果子类中定义了构造函数则不会暗中调用其父类的构造函数。要执行父类的构造函数，需要在子类的构造函数中调用 **parent::\_\_construct()**。

**Example #1 使用新标准的构造函数**

<?php  
class BaseClass {  
   function \_\_construct() {  
       print "In BaseClass constructor\n";  
   }  
}  
  
class SubClass extends BaseClass {  
   function \_\_construct() {  
       parent::\_\_construct();  
       print "In SubClass constructor\n";  
   }  
}  
  
$obj = new BaseClass();  
$obj = new SubClass();  
?>

为了实现向后兼容性，如果 PHP 5 在类中找不到 **\_\_construct()** 函数，它就会尝试寻找旧式的构造函数，也就是和类同名的函数。因此唯一会产生兼容性问题的情况是：类中已有一个名为 **\_\_construct()** 的方法，但它却又不是构造函数。

* 析构函数

void **\_\_destruct** ( void )

PHP 5 引入了析构函数的概念，这类似于其它面向对象的语言，如 C++。析构函数会在到某个对象的所有引用都被删除或者当对象被显式销毁时执行。

**Example #2 析构函数示例**

<?php  
class MyDestructableClass {  
   function \_\_construct() {  
       print "In constructor\n";  
       $this->name = "MyDestructableClass";  
   }  
  
   function \_\_destruct() {  
       print "Destroying " . $this->name . "\n";  
   }  
}  
  
$obj = new MyDestructableClass();  
?>

和构造函数一样，父类的析构函数不会被引擎暗中调用。要执行父类的析构函数，必须在子类的析构函数体中显式调用 **parent::\_\_destruct()**。

**Note:**

析构函数在脚本关闭时调用，此时所有的头信息已经发出。

试图在析构函数中抛出一个异常会导致致命错误。

**3.魔术方法**

*\_\_construct*, *\_\_destruct* (参看 [构造方法和析构方法](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/language.oop5.decon.html)), *\_\_call*, *\_\_callStatic*, *\_\_get*, *\_\_set*, *\_\_isset*, *\_\_unset* (参看 [重载](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/language.oop5.overloading.html)), *\_\_sleep*, *\_\_wakeup*, *\_\_toString*, *\_\_set\_state* 和 [\_\_clone](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/language.oop5.cloning.html) 等方法在PHP中被称为“魔术方法”（Magic methods）。 你在命名自己的类方法时不能使用这些方法名。

**Caution**：

PHP把所有以\_\_（两个下划线）开头的类方法当成魔术方法。所以你定义自己的类方法时，不要以 \_\_为前缀。

**4.成员作用域**

public protected private

**8.其他常用**

1. 变量的引用
2. 函数和方法的参数的引用传递
3. 取消引用（unset函数）：unset函数一般只是把变量的引用计数减一。  
   **注：**php的数据类型使用zval结构保存，一般refcount\_\_gc成员的值为0时才会回收内存
4. 遍历数组和对象属性（foreach）
5. include/require/include\_once/require\_once等区别  
   include与require这两种结构除了在如何处理失败之外完全一样。**include()** 产生一个[警告](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/errorfunc.constants.html#errorfunc.constants.errorlevels.e-warning)而 [require()](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/function.require.html) 则导致一个[致命错误](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/errorfunc.constants.html#errorfunc.constants.errorlevels.e-error)。换句话说，如果想在遇到丢失文件时停止处理页面就用 [require()](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/function.require.html)。**include()** 就不是这样，脚本会继续运行。后缀\_once就表示只包含（运行）一次。
6. 异常处理
7. 预定义变量、异常和接口
8. 命名空间：不常用/不像Java/.net/Flash那样方便，与C++比较相似
9. 类型转换：  
   (int), (integer) - 转换为 整型(integer)   
   (bool), (boolean) - 转换为 布尔型(boolean)   
   (float), (double), (real) - 转换为 浮点型(float)   
   (string) - 转换为 字符串(string)   
   (binary) - 转换为二进制 字符串(string) (PHP 6)   
   (array) - 转换为 数组(array)   
   (object) - 转换为 对象(object)   
   ~~(unset) - 转换为~~ [~~NULL~~](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/language.types.null.html) ~~(PHP 5)~~
10. 伪类型：php手册-》语言参考-》类型-》本文档中使用

**第二节、PHP常见关键字与扩展**

函数名、类名和接口名不区分大小写（只有变量名和常量名区分大小写）

1. **作用域**

Static

Private

Public

Abstract

Interface

Final

1. **语言构造器**

clone new echo print empty var extends

更多请查看php手册

1. **魔术方法与魔术函数**

*\_\_construct*, *\_\_destruct* (参看 [构造方法和析构方法](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/language.oop5.decon.html)), *\_\_call*, *\_\_callStatic*, *\_\_get*, *\_\_set*, *\_\_isset*, *\_\_unset* (参看 [重载](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/language.oop5.overloading.html)), *\_\_sleep*, *\_\_wakeup*, *\_\_toString*, *\_\_set\_state* 和 [\_\_clone](mk:@MSITStore:D:\ZAMPS\books\php_manual_zh.chm::/res/language.oop5.cloning.html)

1. **常用扩展/函数参考**

数学扩展 abs pow

文本处理

PCRE（preg\_开头的正则表达式函数）

字符串

与变量和类型有关的扩展

数组

Classes/Objects

Function Handling

Variable handling

Web Services

SOAP

XML-RPC

XML解析器

Session扩展

其他服务

cURL

FTP

HTTP

Java(需要的话可以看一下)

Memcache

Sockets

SSH2

SVN

其它基本扩展

JSON

Misc.(相当常用)

URLs

图像生成和处理

GD

Exif

国际化与字符编码支持

Iconv

Multibye String

文件系统相关扩展

Directories

Filesystem

数据库扩展

MySQL

MySQLi

日历和事件相关扩展（Date/Time）

Gmdate

Date

Microtime

Time

影响php行为的扩展(Output Control)

flush ob\_clean ob\_end\_clean ob\_end\_flush ob\_flash ob\_get\_clean ob\_get\_contents ob\_start

1. **保留字列表（查看手册的附录中）**

关键词列表

预定义类

预定义常量

**第三节、DX3模板开发**

**嵌入模板：**{subtemplate tplname}

**包含模板：**{template tplname}等于<? include template(‘tplname’); ?>

**变量打印：**{变量}、{常量}，如：{$arr[key]},{$hw},{\_\_FILE\_\_}

**模板语言：**{lang varkey}

**数据调用：**{block/id} {blockdata/id}官方未实现

**广告：**{ad/参数}或{ad 变量名/参数}

有“变量名”结果存到变量中，否则直接输出

“参数”格式：广告位名/样式引用class属性值

**日期时间：**{date($timestamp, $format = 'dt', $timeoffset = '9999', $uformat = '')}

**头像调用：**{avatar $uid, $size, $returnsrc, $real, $static, $ucenterurl}

**执行php代码：**{eval phpcode}或{eval}phpcode{/eval}

**加载系统自动生成的CSS压缩文件：**

Css源文件包括：

模板目录：

common/common.css

common/module.css

common/extend\_common.css

common/extend\_module.css

插件目录：

template/extend\_common.css

template/ extend\_module.css

格式为：{csstemplate}

**换行：**{LF}

**钩子：**{hook/钩子名 参数}

**输出：**{echo 参数}

**判断：**

{if 条件1}

…

{elseif 条件2}

…

{else}

…

{/if}

**循环：**

{loop $array $value}

…

{/loop}

{loop $array $key $value}

…

{/loop}

**块（将块内内容存入变量）：**

{block $varname}

…

{/block}

**Tplname有两种格式：**

插件名:模板文件名

模板文件名

**第四节、DX3插件开发**



**第五节、DX附录**

