一、基本语法

1、以<?php 开始，以?>结束；用;做分隔符；注释用//和/\*   \*/

2、输出文本的基础指令：print 和echo

3、变量都是以$开头，$var\_name = value;  PHP是一门松散类型的语言，设置变量前无需声明，

4、并置运算符：.

      "hello" . " " . "world!"

5、strlen():计算字符串长度，strlen("hello world!") --->12；

     strpos():在字符串内检索一段字符串或一个字符, strpos("Hello world!", "world") ---> 6；

     date("D"):输出某一天，比如“Fri”

6、if...else..

     if...elseif..else...

     switch (expression)

     {

     case label1:

        ....;

        break;

     case label2:

        ....;

        break;

     default:

         ......

     }

7、数组：

    （1）数值数组

             1)$names = array("Peter", "Quagmire", "Joe");    (这种情况会自动分配ID键)；

             2)$names[0] = "Peter";

                $names[1] = "Quagmire";

                $names[2] = "Joe";                                        (人工分配ID键)

    （2）关联数组:每个ID键都关联一个值

             1)$ages = array("Peter" => 32, "Quagmire" =>30, "Joe" =>34);

             2)$ages['Peter'] = 32;

                $ages['Quagmire'] = 30;

                $ages['Joe'] = 34;

    （3）多维数组

            $families = array(

                "Griffin" => array("Peter", "Lois", "Megin"),

                "Quagmire" => array("Glegan"),

                "Joe" => array("CleveLand", "Loretta","Junior")

            );

            输出某个值：

            $families['Grifin'][2]

8、循环

     while

     do...while

     for

     foreach

     foreach($arr as $value)

9、函数（大约700多个内建函数）

     function functionName()

     {

          ......

     }

     function functionName($var)

     {

          ......

     }

      function functionName($var1, $var2)

     {

          ......

          return $total;

     }

10、表单和用户输入

    $\_GET和$\_POST用于检索表单中的值，比如用户输入；

            eg:

            POST:

            <html>

                <body>

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

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

                Age: <input type="text" name="age" />

                <input type="submit" />

                </form>

                </body>

                </html>

               welcome.php

               <html>

               <body>

               Welcome <?php echo $\_POST["name"]; ?>.<br />

               You are <?php echo $\_POST["age"]; ?> years old.

               </body>

               </html>

              GET:

              <form action="welcome.php" method="get">

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

              Age: <input type="text" name="age" />

             <input type="submit" />

             </form>

             [http://www.w3school.com.cn/welcome.php?name=Peter&age=37](http://www.w3school.com.cn/welcome.php?name=Peter&amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;amp;age=37)

         Welcome <?php echo $\_GET["name"]; ?>.<br />

            You are <?php echo $\_GET["age"]; ?> years old!

               表单验证：应在任何可能的时候对用户输入进行验证，客户端验证更快且可以减轻服务器压力，任何流量很高以至于不得不担心服务器资源的站点也有必要担心站点的安全性，若表单访问的是数据库就有必要进行服务端的验证，服务器验证表单的好处是把表单传给自己而不是跳转到不同的页面，这样用户就可以在同一张表单页面得到错误信息，用户也就更容易发现错误了。

            $\_GET变量是一个数组，内容是由http get方法发送的变量名称和值，用于收集来自method="get"的表单的值，对任何人是可见的，对发送信息量也有限制（最多100个字符），在发送密码或其他敏感信息时，不应该使用这个方法。不过，正因为变量显示在 URL 中，因此可以在收藏夹中收藏该页面。在某些情况下，这是很有用的；

            $\_POST 变量用于收集来自 method="post" 的表单中的值，$\_POST 变量是一个数组，内容是由 HTTP POST 方法发送的变量名称和值，变量长度无限制，无法把页面加入书签；

            $\_REQUEST 变量包含了 $\_GET, $\_POST 以及 $\_COOKIE 的内容。

11、Date()

    用于格式化时间或日期,可把时间戳格式化为可读性更好的日期和时间。

        date(format,timestamp)

           format 必需。规定时间戳的格式。

          timestamp 可选。规定时间戳。默认是当前的日期和时间

        eg:date("Y/m/d"),date("Y.m.d");

                     mktime() 函数可为指定的日期返回 Unix 时间戳: mktime(hour,minute,second,month,day,year,is\_dst)

                     eg:

                     <?php

                      $tomorrow = mktime(0,0,0,date("m"),date("d")+1,date("Y"));

                     echo "Tomorrow is ".date("Y/m/d", $tomorrow);

                      ?>

12、PHP引用文件

    include() 或 require() 函数：可以在服务器执行 PHP 文件之前在该文件中插入一个

文件的内容。除了它们处理错误的方式不同之外，这两个函数在其他方面都是相同的。

include() 函数会生成一个警告（但是脚本会继续执行），而 require() 函数会生成一个致命错

误（fatal error）（在错误发生后脚本会停止执行）。

<html>

<body>

<?php include("header.php"); ?>

<h1>Welcome to my home page</h1>

<p>Some text</p>

</body>

</html>

13、文件处理：

(1)

fopen()用于打开文件：

r 只读。在文件的开头开始。

r+ 读/写。在文件的开头开始。

w 只写。打开并清空文件的内容；如果文件不存在，则创建新文件。

w+ 读/写。打开并清空文件的内容；如果文件不存在，则创建新文件。

a 追加。打开并向文件文件的末端进行写操作，如果文件不存在，则创建新文件。

a+ 读/追加。通过向文件末端写内容，来保持文件内容。

x 只写。创建新文件。如果文件以存在，则返回 FALSE。

x+ 读/写。创建新文件。如果文件已存在，则返回 FALSE 和一个错误。

注释：如果 fopen() 无法打开指定文件，则返回 0 (false)。

注释：在 w 、a 以及 x 模式，您无法读取打开的文件！

eg:

<html>

<body>

<?php

$file=fopen("welcome.txt","r");

....

fclose($file);

?>

</body>

</html>

如果 fopen() 不能打开指定的文件，下面的例子会生成一段消息：

<html>

<body>

<?php

$file=fopen("welcome.txt","r") or exit("Unable to open file!");

?>

</body>

</html>

(2)

feof() 函数检测是否已达到文件的末端 (EOF)。

if (feof($file)) echo "End of file";

fgets() 函数用于从文件中逐行读取文件

注释：在调用该函数之后，文件指针会移动到下一行

 eg:

<?php

$file = fopen("welcome.txt", "r") or exit("Unable to open file!");

//Output a line of the file until the end is reached

while(!feof($file))

{

echo fgets($file). "<br />";

}

fclose($file);

?>

fgetc() 函数用于从文件逐字符地读取文件

eg:

<?php

$file=fopen("welcome.txt","r") or exit("Unable to open file!");

while (!feof($file))

{

echo fgetc($file);

}

fclose($file);

?>

(3)文件上传：

<html>

<body>

<form action="upload\_file.php" method="post"

enctype="multipart/form-data">

<label for="file">Filename:</label>

<input type="file" name="file" id="file" />

<br />

<input type="submit" name="submit" value="Submit" />

</form>

</body>

</html>

upload\_file.php:

<?php

if ($\_FILES["file"]["error"] > 0)

{

echo "Error: " . $\_FILES["file"]["error"] . "<br />";

}

else

{

echo "Upload: " . $\_FILES["file"]["name"] . "<br />";

echo "Type: " . $\_FILES["file"]["type"] . "<br />";

echo "Size: " . ($\_FILES["file"]["size"] / 1024) . " Kb<br />";

echo "Stored in: " . $\_FILES["file"]["tmp\_name"];

}

?>

enctype 属性规定了在提交表单时要使用哪种内容类型。在表单需要二进制数据时，比如文件内容，请使用 "multipart/form-data"；

type="file" 属性规定了应该把输入作为文件来处理。举例来说，当在浏览器中预览时，会看到输入框旁边有一个浏览按钮；

$\_FILES全局数组 （$\_FILES["file"]["tmp\_name"] - 存储在服务器的文件的临时副本的名称）

加限制：

if ((($\_FILES["file"]["type"] == "image/gif")

|| ($\_FILES["file"]["type"] == "image/jpeg")

|| ($\_FILES["file"]["type"] == "image/pjpeg"))

&& ($\_FILES["file"]["size"] < 20000))

注释：对于 IE，识别 jpg 文件的类型必须是 pjpeg，对于 FireFox，必须是 jpeg

保存文件：上面的例子在服务器的 PHP 临时文件夹创建了一个被上传文件的临时副本，这个临时的复制文件会在脚本结束时消失。

<?php

if ((($\_FILES["file"]["type"] == "image/gif")

|| ($\_FILES["file"]["type"] == "image/jpeg")

|| ($\_FILES["file"]["type"] == "image/pjpeg"))

&& ($\_FILES["file"]["size"] < 20000))

{

if ($\_FILES["file"]["error"] > 0)

{

echo "Return Code: " . $\_FILES["file"]["error"] . "<br />";

}

else

{

echo "Upload: " . $\_FILES["file"]["name"] . "<br />";

echo "Type: " . $\_FILES["file"]["type"] . "<br />";

echo "Size: " . ($\_FILES["file"]["size"] / 1024) . " Kb<br />";

echo "Temp file: " . $\_FILES["file"]["tmp\_name"] . "<br />";

if (file\_exists("upload/" . $\_FILES["file"]["name"]))

{

echo $\_FILES["file"]["name"] . " already exists. ";

}

else

{

move\_uploaded\_file($\_FILES["file"]["tmp\_name"],

"upload/" . $\_FILES["file"]["name"]);

echo "Stored in: " . "upload/" . $\_FILES["file"]["name"];

}

}

}

else

{

echo "Invalid file";

}

?>

14、cookies

(1)setcookie()设置cookie

setcookie(name, value, expire, path, domain);

注释：setcookie() 函数必须位于 <html> 标签之前

注释：在发送 cookie 时，cookie 的值会自动进行 URL 编码，在取回时进行自动解码（为防止 URL 编码，请使用 setrawcookie() 取而代之）

eg:

<?php

setcookie("user", "Alex Porter", time()+3600);

?>

<html>

<body>

</body>

</html>

(2)取回cookie值

<?php

// Print a cookie

echo $\_COOKIE["user"];

// A way to view all cookies

print\_r($\_COOKIE);

?>

<html>

<body>

<?php

if (isset($\_COOKIE["user"]))

echo "Welcome " . $\_COOKIE["user"] . "!<br />";

else

echo "Welcome guest!<br />";

?>

</body>

</html>

(3)删除cookie

<?php

// set the expiration date to one hour ago

setcookie("user", "", time()-3600);

?>

15、sessions

PHP session 变量用于存储有关用户会话的信息，或更改用户会话的设置。 Session 变量保存的信息是单一用户的，并且可供应用程序中的所有页面使用.

Session 的工作机制是：为每个访问者创建一个唯一的 id (UID)，并基于这个 UID 来存储变量。UID 存储在 cookie 中，亦或通过 URL 进行传导。

在您把用户信息存储到 PHP session 中之前，首先必须启动会话:

<?php session\_start(); ?>

<html>

<body>

</body>

</html>

上面的代码会向服务器注册用户的会话，以便您可以开始保存用户信息，同时会为用户会话分配一个 UID。

注释：session\_start() 函数必须位于 <html> 标签之前

<?php

session\_start();

// store session data

$\_SESSION['views']=1;

?>

<html>

<body>

<?php

//retrieve session data

echo "Pageviews=". $\_SESSION['views'];

?>

</body>

</html>

<?php

session\_start();

if(isset($\_SESSION['views']))

$\_SESSION['views']=$\_SESSION['views']+1;

else

$\_SESSION['views']=1;

echo "Views=". $\_SESSION['views'];

?>

如果您希望删除某些 session 数据，可以使用 unset() 或 session\_destroy() 函数:

unset() 函数用于释放指定的 session 变量：

<?php

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

?>

您也可以通过 session\_destroy() 函数彻底终结 session：

<?php

session\_destroy();

?>

注释：session\_destroy() 将重置 session，您将失去所有已存储的 session 数据。

16、mail函数

mail(to,subject,message,headers,parameters)

to 必需。规定 email 接收者。

subject 必需。规定 email 的主题。注释：该参数不能包含任何新行字符。

message 必需。定义要发送的消息。应使用 LF (\n) 来分隔各行。

headers可选。规定附加的标题，比如 From、Cc 以及 Bcc。应当使用 CRLF (\r\n) 分隔附加的标题。

parameters 可选。对邮件发送程序规定额外的参数。

eg:

<?php

$to = "someone@example.com";

$subject = "Test mail";

$message = "Hello! This is a simple email message.";

$from = "someonelse@example.com";

$headers = "From: $from";

mail($to,$subject,$message,$headers);

echo "Mail Sent.";

?>

email注入：

假如用户在表单中的输入框内加入这些文本，会出现什么情况呢？

someone@example.com%0ACc:person2@example.com

%0ABcc:person3@example.com,person3@example.com,

anotherperson4@example.com,person5@example.com

%0ABTo:person6@example.com

与往常一样，mail() 函数把上面的文本放入邮件头部，那么现在头部有了额外的 Cc:, Bcc:

以及 To: 字段。当用户点击提交按钮时，这封 e-mail 会被发送到上面所有的地址

预防：

$field=filter\_var($field, FILTER\_SANITIZE\_EMAIL);

if(filter\_var($field, FILTER\_VALIDATE\_EMAIL))

。。。。。

FILTER\_SANITIZE\_EMAIL 从字符串中删除电子邮件的非法字符

FILTER\_VALIDATE\_EMAIL 验证电子邮件地址

16、PHP错误处理

        在 **PHP** 中，默认的错误处理很简单。一条消息会被发送到浏览器，这条消息带有文件名、行号以及一条描述错误的消息。

        创建自定义错误处理器：error\_function(error\_level,error\_message,error\_file,error\_line,error\_context)

           error\_level    必需。为用户定义的错误规定错误报告级别。必须是一个值数；

        error\_message 必需。为用户定义的错误规定错误消息；

        error\_file 可选。规定错误在其中发生的文件名；

        error\_line 可选。规定错误发生的行号；

        error\_context 可选。规定一个数组，包含了当错误发生时在用的每个变量以及它们的值。

error\_level：

      2    E\_WARNING       非致命的 run-time 错误。不暂停脚本执行;

       8      E\_NOTICE             脚本发现可能有错误发生，但也可能在脚本正常运行时发生

       256  E\_USER\_ERROR   致命的用户生成的错误

       512  E\_USER\_WARNING 非致命的用户生成的警告

       1024 E\_USER\_NOTICE    用户生成的通知

       4096 E\_RECOVERABLE\_ERROR  可捕获的致命错误

       8191 E\_ALL    所有错误和警告，除级别 E\_STRICT 以外

（1）设置默认错误处理程序：set\_error\_handler("customError");

eg:

<?php

//error handler function

function customError($errno, $errstr)

{

echo "<b>Error:</b> [$errno] $errstr";

}

//set error handler

set\_error\_handler("customError");

//trigger error

echo($test);

?>

（2）触发错误：trigger\_error()

    E\_USER\_ERROR - 致命的用户生成的 run-time 错误。错误无法恢复。脚本执行被中断；

    E\_USER\_WARNING - 非致命的用户生成的 run-time 警告。脚本执行不被中断；

    E\_USER\_NOTICE - 默认。用户生成的 run-time 通知。脚本发现了可能的错误，也有可能在脚本运行正常时发生。

eg:

<?php

//error handler function

function customError($errno, $errstr)

{

echo "<b>Error:</b> [$errno] $errstr<br />";

echo "Ending Script";

die();

}

//set error handler

set\_error\_handler("customError",E\_USER\_WARNING);

//trigger error

$test=2;

if ($test>1)

{

trigger\_error("Value must be 1 or below",E\_USER\_WARNING);

}

?>

（3）错误记录：根据在 php.ini 中的 error\_log 配置，PHP 向服务器的错误记录系统或文件发送错误记录。通过使用 error\_log() 函数，您可以向指定的文件或远程目的地发送错误记录

eg:

<?php

//error handler function

function customError($errno, $errstr)

{

echo "<b>Error:</b> [$errno] $errstr<br />";

echo "Webmaster has been notified";

error\_log("Error: [$errno] $errstr",1,

"someone@example.com","From: webmaster@example.com");

}

//set error handler

set\_error\_handler("customError",E\_USER\_WARNING);

//trigger error

$test=2;

if ($test>1)

{

trigger\_error("Value must be 1 or below",E\_USER\_WARNING);

}

?>

17、PHP异常处理

<?php

//创建可抛出一个异常的函数

function checkNum($number)

{

if($number>1)

{

throw new Exception("Value must be 1 or below");

}

return true;

}

//在 "try" 代码块中触发异常

try

{

checkNum(2);

//If the exception is thrown, this text will not be shown

echo 'If you see this, the number is 1 or below';

}

//捕获异常

catch(Exception $e)

{

echo 'Message: ' .$e->getMessage();

}

?>

自定义Exception：

<?php

class customException extends Exception

{

public function errorMessage()

{

//error message

$errorMsg = 'Error on line '.$this->getLine().' in '.$this->getFile()

.': <b>'.$this->getMessage().'</b> is not a valid E-Mail address';

return $errorMsg;

}

}

$email = "someone@example...com";

try

{

//check if

if(filter\_var($email, FILTER\_VALIDATE\_EMAIL) === FALSE)

{

//throw exception if email is not valid

   throw new customException($email);

}

}

catch (customException $e)

{

//display custom message

echo $e->errorMessage();

}

?>

多个异常：

可以使用多个 if..else 代码块，或一个 switch 代码块，或者嵌套多个异常,这些异常能够使用不同的 exception 类，并返回不同的错误消息

设置顶层异常处理器 （ **Top Level Exception Handler**）:

set\_exception\_handler() 函数可设置处理所有未捕获异常的用户定义函数

<?php

function myException($exception)

{

echo "<b>Exception:</b> " , $exception->getMessage();

}

set\_exception\_handler('myException');

throw new Exception('Uncaught Exception occurred');

?>

输出：Exception: Uncaught Exception occurred

异常的规则：

需要进行异常处理的代码应该放入 try 代码块内，以便捕获潜在的异常；

每个 try 或 throw 代码块必须至少拥有一个对应的 catch 代码块；

使用多个 catch 代码块可以捕获不同种类的异常；

可以在 try 代码块内的 catch 代码块中再次抛出（re-thrown）异常。

简而言之：如果抛出了异常，就必须捕获它。

17、PHP过滤器Filter

        **PHP** 过滤器用于验证和过滤来自非安全来源的数据，比如用户的输入

（1）

函数和过滤器：

filter\_var() - 通过一个指定的过滤器来过滤单一的变量；

filter\_var\_array() - 通过相同的或不同的过滤器来过滤多个变量；

filter\_input - 获取一个输入变量，并对它进行过滤；

filter\_input\_array - 获取多个输入变量，并通过相同的或不同的过滤器对它们进行过滤

eg:

<?php

$int = 123;

if(!filter\_var($int, FILTER\_VALIDATE\_INT))

{

echo("Integer is not valid");

}

else

{

echo("Integer is valid");

}

?>

过滤器：FILTER\_VALIDATE\_INT

（2）**Validating** 和 **Sanitizing**

Validating过滤器：

      验证用户输入；

      严格的格式规则（比如 URL 或 E-Mail 验证）；

      返回若成功预期的类型，否则返回 FALSE。

Sanitizing过滤器：

       用于允许或禁止字符串中指定的字符；

       无数据格式规则；

       始终返回字符串。

选项和标志：

选项和标志用于向指定的过滤器添加额外的过滤选项。

eg:

<?php

$var=300;

$int\_options = array(

"options"=>array

(

"min\_range"=>0,

"max\_range"=>256

)

);

if(!filter\_var($var, FILTER\_VALIDATE\_INT, $int\_options))

{

echo("Integer is not valid");

}

else

{

echo("Integer is valid");

}

?>

验证输入：

<?php

if(!filter\_has\_var(INPUT\_GET, "email"))

{

echo("Input type does not exist");

}

else

{

if (!filter\_input(INPUT\_GET, "email", FILTER\_VALIDATE\_EMAIL))

{

echo "E-Mail is not valid";

}

else

{

echo "E-Mail is valid";

}

}

?>

净化输入："http://www.W3#$%S^%$#ool.com.cn/"  ----->  http://www.W3School.com.cn/

<?php

if(!filter\_has\_var(INPUT\_POST, "url"))

{

echo("Input type does not exist");

}

else

{

$url = filter\_input(INPUT\_POST,

"url", FILTER\_SANITIZE\_URL);

}

?>

过滤多个输入：

filter\_var\_array(), filter\_input\_array() 避免对filter\_var和filter\_input的重复调用

18、Mysql

（1）连接到一个mysql库：

           mysql\_connect(servername, username, password);

          eg:

          <?php

          $con = mysql\_connect("localhost","peter","abc123");

         if (!$con)

        {

            die('Could not connect: ' . mysql\_error());

        }

        // some code

        mysql\_close($con);

        ?>

（2）mysql\_query()

        eg:

        if (mysql\_query("CREATE DATABASE my\_db",$con))

       {

           echo "Database created";

        }

 （3）mysql\_select\_db()选取使用哪个数据库

        mysql\_connect();

        mysql\_select\_db();

        mysql\_query();

        mysql\_close();

 （4）sql类型：

       int(size)

       smallint(size)

       tinyint(size)

       mediumint(size)

       bigint(size)

       仅支持整数，size规定数字的最大值；

       decimal(size, d)

       double(size, d)

       float(size, d)

       支持带有小数的数字，size规定最大值，d规定小数点右侧的数字最大值；

      文本类型：

      char(size)    固定长度的字符串；

      varchar(size)  可变长度的字符串；

      tinytext  可变长度字符串，最大长度255个字符；

             text

             blob

             可变长度字符串，最大65535个字符串

              mediumtext

              mediumblob

              可变长度字符串，最大长度16777215个字符；

               longtext

               longblob

               可变长度字符串，最大长度4294967295个字符；

       日期数据类型：

       date(yyyy-mm-dd)

       datetime(yyyy-mm-dd hh:mm:ss)

       timestamp(yyyymmddhhmmss)

       time(hh:mm:ss)

       杂项数据类型：

       enum(value1, value2, etc) 括号里最多可放65535个值；

       set  与enum相似但是set可拥有最多64个列表项目，并可存放不止一个choice

（5）插入数据insert into(在php中需要使用mysql\_query函数)

          insert into table\_name

          values(value1, value2, ......)

         eg:

         $sql="INSERT INTO person (FirstName, LastName, Age)

         VALUES

          ('$\_POST[firstname]','$\_POST[lastname]','$\_POST[age]')";

          if (!mysql\_query($sql,$con))

         {

              die('Error: ' . mysql\_error());

          }

         echo "1 record added";

（6）查询数据 select

         select colum\_name(s) from table\_name;

         eg:

         <?php

         $con = mysql\_connect("localhost","peter","abc123");

         if (!$con)

         {

             die('Could not connect: ' . mysql\_error());

         }

         mysql\_select\_db("my\_db", $con);

         $result = mysql\_query("SELECT \* FROM person");

         while($row = mysql\_fetch\_array($result))

        {

             echo $row['FirstName'] . " " . $row['LastName'];

             echo "<br />";

         }

         mysql\_close($con);

         ?>

（7）where

select colum from table where colum opertator value;

          order by

select colum\_name(s) from table\_name order by column\_name;   默认是升序，若要降序要加DESC；

根据两列排序：select colum\_name(s) from table\_name order by column\_name1, column\_name2;  只要第一列相同时才使用第二列；

        update

update table\_name set column\_name = new\_value where column\_name = some\_value;

        delete from

delete from table\_name where column\_name = some\_value;

（8）PHP database ODBC（Application Programming Interface,API）   使我们有能力连接到某一个数据源

           odbc\_connect() 连接到ODBC，四个参数：数据源名、用户名、密码、可选的指针类型参数；

           odbc\_exec()  用于执行sql语句；

           odbc\_fetch\_row()  从结果集中返回记录，如果能返回行则返回true，否则返回false；两个参数：odbc结果标示符和可选行号；

           odbc\_result()  从记录中读取字段，两个参数：ODBC结果标示符和字段编号或名称；eg：$compname=odbc\_result($rs,1);

                                                                                                                                                                                   $compname=odbc\_result($rs,"CompanyName");

           odbc\_close($conn);  关闭ODBC连接；

eg:

<html>

<body>

<?php

$conn=odbc\_connect('northwind','','');

if (!$conn)

{exit("Connection Failed: " . $conn);}

$sql="SELECT \* FROM customers";

$rs=odbc\_exec($conn,$sql);

if (!$rs)

{exit("Error in SQL");}

echo "<table><tr>";

echo "<th>Companyname</th>";

echo "<th>Contactname</th></tr>";

while (odbc\_fetch\_row($rs))

{

$compname=odbc\_result($rs,"CompanyName");

$conname=odbc\_result($rs,"ContactName");

echo "<tr><td>$compname</td>";

echo "<td>$conname</td></tr>";

}

odbc\_close($conn);

echo "</table>";

?>

</body>

</html>

19、XML解析

         两种基本的XML解析器：基于树的解析器、基于事件的解析器；

         基于树的：把XML转化为树型结构，它分析整篇文档，并提供了API来访问树中的元素（例如文档对象模型DOM）；

         基于事件的：将XML文档视为一系列的事件，当某个具体的事件发生时，解析器会调用函数来处理；（集中在XML文档的内容而不是它们的结果，所以基于事件的解析器能更快的访问数据）

（1）expat解析（内建的解析器，基于事件的）

         eg:(不过这个例子是无效的XML，因为没有与它关联的文档类型声明DTD，也没有内嵌的DTD，不过在使用expat解析器时无关系它不检查有效性忽略任何DTD)

         <from>John</from>

         三个事件：开始元素<from>;    开始CDATA部分，值：John；   关闭元素</from>

eg:

<?php

//Initialize the XML parser

$parser=xml\_parser\_create();

//Function to use at the start of an element

function start($parser,$element\_name,$element\_attrs)

{

switch($element\_name)

{

case "NOTE":

echo "-- Note --<br />";

break;

case "TO":

echo "To: ";

break;

case "FROM":

echo "From: ";

break;

case "HEADING":

echo "Heading: ";

break;

case "BODY":

echo "Message: ";

}

}

//Function to use at the end of an element

function stop($parser,$element\_name)

{

echo "<br />";

}

//Function to use when finding character data

function char($parser,$data)

{

echo $data;

}

//Specify element handler

xml\_set\_element\_handler($parser,"start","stop");

//Specify data handler

xml\_set\_character\_data\_handler($parser,"char");

//Open XML file

$fp=fopen("test.xml","r");

//Read data

while ($data=fread($fp,4096))

{

xml\_parse($parser,$data,feof($fp)) or

die (sprintf("XML Error: %s at line %d",

xml\_error\_string(xml\_get\_error\_code($parser)),

xml\_get\_current\_line\_number($parser)));

}

//Free the XML parser

xml\_parser\_free($parser);

?>

工作原理解释：

· 通过 xml\_parser\_create() 函数初始化 XML 解析器

· 创建配合不同事件处理程序的的函数

· 添加 xml\_set\_element\_handler() 函数来定义，当解析器遇到开始和结束标签时执行哪

个函数

· 添加 xml\_set\_character\_data\_handler() 函数来定义，当解析器遇到字符数据时执行哪

个函数

· 通过 xml\_parse() 函数来解析文件 "test.xml"

· 万一有错误的话，添加 xml\_error\_string() 函数把 XML 错误转换为文本说明

· 调用 xml\_parser\_free() 函数来释放分配给 xml\_parser\_create() 函数的内存

（2）DOM解析：基于树的解析。（DOM：W3C DOM提供了针对HTML和xml文档的标准对象集，以及用于访问和操作这些文档的标准接口（Core，XML，Html））

（3）PHP SimpleXML

          从 PHP 5.0 开始，SimpleXML 函数是 PHP 核心的组成部分。无需安装就可以使用这些函数

<?xml version="1.0" encoding="ISO-8859-1"?>

<note>

<to>George</to>

<from>John</from>

<heading>Reminder</heading>

<body>Don't forget the meeting!</body>

</note>

eg:

<?php

$xml = simplexml\_load\_file("test.xml");

echo $xml->getName() . "<br />";

foreach($xml->children() as $child)

{

echo $child->getName() . ": " . $child . "<br />";

}

?>

以上代码的输出：

note

to: George

from: John

heading: Reminder

body: Don't forget the meeting!

dd

20、AJAX简介（Asynchronous JavaScript And XML  异步JavaScript 及 XML）

（1）XMLHttpRequest

IE 用ActiveXObject，其他浏览器用XMLHttpRequest

eg:

var XMLHttp=null

if (window.XMLHttpRequest)

{

XMLHttp=new XMLHttpRequest()

}

else if (window.ActiveXObject)

{

XMLHttp=new ActiveXObject("Microsoft.XMLHTTP")

}

21、PHP和AJAX MySQL

AJAX可用来与数据库进行交互式通信；

22、PHP和AJAX responseXML实例

AJAX可用于以XML返回数据库信息；

23、PHP和AJAX Live Search

AJAX可为用户提供更友好、交互性更强的搜索体验；

24、PHP和AJAX RSS阅读器

RSS阅读器用于阅读RSS Feed

RSS允许对新闻和更新进行快速浏览；

25、PHP和AJAX投票

26、PHP函数

（1）Array函数（PHP支持单维、多维数组，同时提供了用数据库查询结果来构造数组的函数）

array()    创建数组；

array\_change\_key\_case()    返回其键均为大写或小写的数组；

array\_chunk()    把数组分割为新的数组块；

常量：

CASE\_LOWER

（2）Calendar函数

cal\_days\_in\_month()    针对指定的年份和日历，返回一个月中的天数；

常量：

（3）Date/Time函数（日期/时间函数的行为受到 php.ini 中设置的影响）

（4）Directory函数

chdir()    改变当前的目录；

（5）Error和Logging函数

（6）Filesystem函数

basename()    返回路径中的文件名部分；

chgrp()     改变文件组；

（7）Filter函数（用于对来自非安全来源的数据（比如用户输入）进行验证和过滤）

 filter\_has\_var()     检查是否存在指定输入类型的变量；

（8）Ftp函数

ftp\_alloc()    为要上传到FTP服务器的文件分配空间；

（9）Http函数

header()    向客户端发送原始的http报头；

（10）libxml函数（libxml 函数和常量与 SimpleXML, XSLT 以及 DOM 一起使用）

libxml\_clear\_errors() 清空 libxml 错误缓冲

（11）Mail函数（HTTP 函数允许您从脚本中直接发送电子邮件）

（12）Math函数（数学 (Math) 函数能处理 integer 和 float 范围内的值）

abs（）   绝对值

acos()    反余弦

（13）MySQL函数

mysql\_connect()

mysql\_pconnect()

（14）SimpleXML函数（SimpleXML 函数允许您把 XML 转换为对象）

（15）String函数

addcslashes()     在指定的字符前添加反斜杠；

（16）XML Parser函数（XML 函数允许我们解析 XML 文档，但无法对其进行验证）

（17）Zip File （压缩文件函数允许我们读取压缩文件）

zip\_open() 打开 ZIP 文件

zip\_read() 读取 ZIP 文件中的下一个项目

zip\_close() 关闭 ZIP 文件

（18）PHP杂项函数（不属于其他类别的函数归纳到这，杂项函数函数的行为受到 php.ini 中设置的影响）

connection\_aborted() 检查是否断开客户机

connection\_status() 返回当前的连接状态

二、Morden PHP

1、namespace

eg:

<?php

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | /\* |
|  | \* This file is part of the Symfony package. |
|  | \* |
|  | \* (c) Fabien Potencier <fabien@symfony.com> |
|  | \* |
|  | \* For the full copyright and license information, please view the LICENSE |
|  | \* file that was distributed with this source code. |
|  | \*/ |
|  |  |
|  | namespace Symfony\Component\HttpFoundation; |
|  |  |
|  | use Symfony\Component\HttpFoundation\Session\SessionInterface; |
|  |  |
|  | /\*\* |
|  | \* Request represents an HTTP request. |
|  | \* |
|  | \* The methods dealing with URL accept / return a raw path (% encoded): |
|  | \* \* getBasePath |
|  | \* \* getBaseUrl |
|  | \* \* getPathInfo |
|  | \* \* getRequestUri |
|  | \* \* getUri |
|  | \* \* getUriForPath |
|  | \* |
|  | \* @author Fabien Potencier <fabien@symfony.com> |
|  | \*/ |
|  | class Request |
|  | {  .................  } |
| } |  |
|  |  |

使用：

use Symfony\Component\HttpFoundation\Request;

use Symfony\Component\HttpFoundation\Response;

$request = Request::createFromGlobals();echo $request->getPathInfo();

2、import and alias

<?php

**use** Symfony\Component\HttpFoundation\Response;

$response = **new** Response('Oops' , 400);

$response->send ();

<?php

**use** Symfony\Component\HttpFoundation\Response **as** Res;

$r = **new** Res('Oops' , 400);

$r->send ();

<?php

**use** func **Namespace** \functionName;

functionName();

<?php

**use** constant **Namespace** \CONST\_NAME;

**echo** CONST\_NAME;

multiple namespaces:

<?php

**namespace** Foo {

*// Declare classes, interfaces, functions, and constants here*

}

**namespace** Bar {

*// Declare classes, interfaces, functions, and constants here*

}

3、traits

定义：

<?php

trait Geocodable{

    protected $address;

    protected $geocoder;

    protected $geocoderResult;

    public function setGeocoder(\Geocoder\GeocoderInterface $geocoder)

    {

        $this->geocoder = $geocoder;

     }

     .....

}

使用：

<?php

class RetailStore

{

    use Geocodable;

    .....

}

<?php

$geocoderAdapter = new \Geocoder\HttpAdapter\CurlHttpAdapter();

$geocoderProvider = new \Geocoder\Provider\GoogleMapsProvider($geocoderAdapter);

$geocoder = new \Geocoder\Geocoder($geocoderProvider);

$store = new RetailStore();

$store->setAddress('420 9th Avenue, New York, NY 10001 USA');

$store->setGeocoder($geocoder);

.....

4、generators

在顺序遍历时可以节省内存（只用少量内存），但是回溯、快进搜索等不能用；

(1)

<?php

function myGenerator(){

    yield 'value1';

    yield 'value2';

    yield 'value3';

}

<?php

foreach(myGenerator() as $yieldValue){

    echo $yieldValue, PHP\_EOL;

}

(2)

<?php

function makeRange($length){

    for($i=0; $i<length; $i++){

        yield $i;

    }

}

foreach(makeRange(1000000) ad $i){

    echo $i, PHP\_EOL;

}

(3)big file

<?php

function getRows($file){

    $handle = fopen($file, 'rb');

    if($handle === false){

        throw new Exception();

    }

    while(feof($handle) === false){

        yield fgetcsv($handle);

    }

    fclose($handle);

}

foreach (getRows('data.csv') as $row){

    print\_r($row);

}

5、closure（Anonymous function）

(1)<?php

$closure = function($name){

    return sprintf('Hello %s', $name);

};

echo $closure("josh");

(2)

<?php

$numbersPlusOne = array\_map(function($number){

    return $number + 1;

},[1,2,3]);

print\_r($numbersPlusOne);

或者：

<?php

function incrementNumber($number){

    return $number + 1;

}

$numbersPlusOne = array\_map('incrementNumber', [1,2,3]);

print\_r($numbersPlusOne);

（3）

<?php

**function** enclosePerson( $name) {

**return function** ($doCommand ) **use** ($name) {

**return** sprintf( '%s, %s', $name , $doCommand);

};

}

*// Enclose "Clay" string in closure*

$clay = enclosePerson ('Clay');

*// Invoke closure with command*

**echo** $clay('get me sweet tea!');

*// Outputs --> "Clay, get me sweet tea!"*