# PHP基础语法

## 1，变量/常量等

|  |  |  |  |
| --- | --- | --- | --- |
| PHP错误控制符：  PHP中提供了一个错误控制运算符“@”，对于一些可能会在运行过程中出错的表达式时，我们不希望出错的时候给客户显示错误信息，可以将@放置在一个PHP表达式之前，该表达式可能产生的任何错误信息都被忽略掉；  如果激活了track\_error（在php.ini中设置）特性，表达式所产生的任何错误信息都被存放在变量$php\_errormsg中，此变量在每次出错时都会被覆盖  需要注意的是：错误控制前缀“@”不会屏蔽解析错误的信息，不能把它放在函数或类的定义之前，也不能用于条件结构例如if和foreach等。 | 变量：  $var\_name = “苹果”;  var\_dump($string); 输出数据类型    字符串型可以用三种方法定义：**单引号形式**、**双引号形式**和**Heredoc结构形式（<<<GOD字符串GOD;)**  **当双引号中包含变量时，变量会与双引号中的内容连接在一起；**  **当单引号中包含变量时，变量会被当做字符串输出。** | 资源（resource）：资源是由专门的函数来建立和使用的，例如打开文件、数据连接、图形画布。我们可以对资源进行操作（创建、使用和释放）。任何资源，在不需要的时候应该被及时释放。如果我们忘记了释放资源，系统自动启用垃圾回收机制，在页面执行完毕后回收资源，以避免内存被消耗殆尽。  常量：define()完成定义  bool define(string $constant\_name,  mixed $value[, $case\_sensitive = true])  define("PI",3.14);  define($pi,3.14);  defined()函数可以帮助我们判断一个常量是否已经定义，其语法格式为：  bool defined(string constants\_name) true/false | **引用其他php文件**：  require\_once("./db.php); |
| 系统变量：  （1）\_\_FILE\_\_ :php程序文件名。  （2）\_\_LINE\_\_ :PHP程序文件行数。  （3）PHP\_VERSION:当前解析器的版本号。。  （4）PHP\_OS：执行当前PHP版本的操作系统名称。 | PHP的赋值运算符有两种，分别是：  (1)“=”：把右边表达式的值赋给左边的运算数。它将右边表达式值复制一份，交给左边的运算数。换而言之，首先给左边的运算数申请了一块内存，然后把复制的值放到这个内存中。  (2)“&”：引用赋值，意味着两个变量都指向同一个数据。它将使两个变量共享一块内存，如果这个内存存储的数据变了，那么两个变量的值都会发生变化。 | C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(11).png | C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(12).png |

## 2，语句

|  |  |  |
| --- | --- | --- |
| echo是PHP中的输出语句，可以把字符串输出（字符串用双引号括起来）。  如下代码：  <?php echo "Hello world!";?>  <?php echo 'Hi,'.'imooc!';?> 连接符 | foreach循环：  在PHP中foreach循环语句，常用于遍历数组，一般有两种使用方式:不取下标、取下标。  （1）只取值，不取下标   foreach (数组 as 值){  //执行的任务} | （2）同时取下标和值  foreach (数组 as 下标 => 值){   //执行的任务} |

## 3，数组和函数

|  |  |  |  |
| --- | --- | --- | --- |
| PHP有两种数组：索引数组、关联数组  定义索引数组：  $arr = array();表示创建一个空数组，并把创建的空数组赋值给变量$arr  $fruit = array("苹果","香蕉","菠萝"); | 定义关联数组：  $fruit = array(      'apple'=>"苹果",      'banana'=>"香蕉",      'pineapple'=>"菠萝"  );  print\_r($fruit);语句输出数组键及对应的值。 | 函数：  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。  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。  php中可以直接使用**$变量名=函数名**，作为指针使用 | function numbers() {     return array(1, 2, 3); }  list ($one, $two, $three) = numbers();  判断函数是否存在：  function\_exists('函数名');  类似的还有class\_exists,file\_exists等 |

## 4，字符串

|  |  |  |  |
| --- | --- | --- | --- |
| **去掉字符串的空格**  trim去除一个字符串两端空格。 rtrim是去除一个字符串右部空格，其中的r是right的缩写。  ltrim是去除一个字符串左部空格，其中的l是left的缩写。 | **查找字符串：**  strpos(要处理的字符串, 要定位的字符串, 定位的起始位置[可选])    **获取字符串长度**  $len = strlen($str);获取长度  echo mb\_strlen($str,"UTF8");获取中文长度 | **截取字符串**：  1、英文字符串的截取函数substr()  函数说明：substr(字符串变量,开始截取的位置，截取长度）  2、中文字符串的截取函数mb\_substr()  函数说明：mb\_substr(字符串变量,开始截取的位置，截取长度, 编码） | **替换字符串**  **str\_replace**(要查找的字符串, 要替换的字符串, 被搜索的字符串, 替换计数[可选])  **格式化字符串**：  函数说明：sprintf(格式, 要转化的字符串)  返回：格式化好的字符串  $result = sprintf('%01.2f', $str); |
| **字符串合并**：  函数说明：implode(分隔符[可选], 数组)  返回值：把数组元素组合为一个字符串  **字符串分隔**：  函数说明：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 | **字符串转义**：  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? |  |  |

## 5，类

|  |  |  |
| --- | --- | --- |
| 类定义：  class Car extends 父类**(单继承）**{  function **\_\_construct()** {  $a = new Car(); $str = serialize($a); //对象序列化成字符串 echo $str.'<br>';  $b = unserialize($str); //反序列化为对象  var\_dump($b);         parent::\_\_construct();         print "构造函数被调用\n";     }  function **\_\_destruct()** {        print "析构函数被调用 \n";     }      //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 $price = '100000';      定义function类似属性(**$this->name**)    public **static**$speed = 10;  } | **生成对象**：  $car = new Car(); 创建对象，指针接收  echo $car->getName();  echo $car->name;  echo Car::speed;  **对象比较**：  当同一个类的两个实例的所有属性都相等时，可以使用比较运算符==进行判断，当需要判断两个变量是否为同一个对象的引用时，可以使用全等运算符===进行判断。  **对象复制**：  在一些特殊情况下，可以通过关键字clone来复制一个对象，这时\_\_clone方法会被调用，通过这个魔术方法来设置属性的值。  $b = **clone** $a;  **对象序列化**：  $a = new Car(); $str = serialize($a); //对象序列化成字符串 echo $str.'<br>'; $b = unserialize($str); //反序列化为对象  var\_dump($b);    **类接口**：（同java）  interface 接口名{方法定义}  class 类名 implements 接口名{方法实现} | **方法重载**：  方法的重载通过\_\_call来实现，当调用不存在的方法的时候，将会转为参数调用\_\_call方法，当调用不存在的静态方法时会使用\_\_callStatic重载。  class Car {     public $speed = 0;         public function \_\_call($name, $args) {         if ($name == 'speedUp') {             $this->speed += 10;         }     } } $car = new Car(); $car->speedUp(); //调用不存在的方法会使用重载  echo $car->speed;  **类多态**：（同C++）   1，定义抽象类 aabstract class 类名{抽象方法定义;普通方法定义}   2，子类继承抽象类，实现抽象方法   3，定义抽象类指向子类对象，调用抽象方法就会调用某个子类对象的实现，调用普通方法则会调用父类实现  **假多态** |
| C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(13).png | **类魔术方法**：  1，\_\_totring(): 对象当作字符串使用时  2，\_\_invoke($arg):对象当作方法使用时  3，\_\_call&\_\_callStatic：方法重载，见上面解释  4，\_\_clone：调用clone关键字深拷贝时调用  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(14).png |  |

## 6，正则表达式

|  |  |  |
| --- | --- | --- |
| PCRE库函数中，正则匹配模式使用分隔符与元字符组成，分隔符可以是非数字、非反斜线、非空格的任意字符。经常使用的分隔符是正斜线(/)、hash符号(#) 以及取反符号(~)，例如：  /foo bar/ #^[^0-9]$#  ~php~  1，分隔符后面可以使用模式修饰符，模式修饰符包括：i, m, s, x等  2，如果模式中包含分隔符，则分隔符需要使用反斜杠（\）进行转义 | PHP中使用PCRE库函数进行正则表达式匹配：  **preg\_match**用来执行一个匹配，可以简单的用来判断模式是否匹配成功，或者取得一个匹配结果，他的返回值是匹配成功的次数0或者1，在匹配到1次以后就会停止搜索。  $subject = "abcdef"; $pattern = '/def/'; preg\_match($pattern, $subject, $matches); print\_r($matches); //结果为：Array ( [0] => def ) | **preg\_match\_all**可以循环获取一个列表的匹配结果数组，参数和preg\_match一样  **preg\_replace**：根据匹配的内容按照规则进行替换  $string = 'April 15, 2014'; **$pattern = '/(\w+) (\d+), (\d+)/i'; $replacement = '$3, ${1} $2';**  echo preg\_replace($pattern, $replacement, $string); //结果为：2014, April 15 |

## 7，文件操作

|  |  |  |  |
| --- | --- | --- | --- |
| **文件内容读取**：  **类似于C语言操作文件**：  $fp = fopen('./text.txt',   'rb'); while(!feof($fp)) {  //读取一行  echo fgets($fp);  //一次读取4096个字符    $contents .= fread($fp, 4096);  }  fclose($fp);  **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');  也可以通过参数控制读取内容的开始点以及长度。  $content = file\_get\_contents  ('./test.txt', null, null,  100, 500); | **判断文件是否存在/可读写**  **file\_exists(path**): 判断一个文件或目录是否存在；  **is\_file(path)**: 判断是否是文件；  **is\_readable(file**): 文件是否可读  **is\_writeable(file)**: 文件是否可写  =============================  **文件内容写入**    **类似C语言操作**：  $fp = fopen('./test.txt', 'w'); fwrite($fp, 'hello'); fwrite($fp, 'world'); fclose($fp);  **file\_put\_contents**($filename,  $data);当$data是数组的时候，会自动的将数组连接起来，相当于$data=implode('', $data); | **获取文件各时间属性**：  fileowner($file) ：获得文件的所有者  filectime($file) ：获取文件的创建时间  filemtime($file) ：获取文件的修改时间  fileatime($file) ：获取文件的访问时间  **获取文件大小：**  **filesize**($file):  获取文件大小，返回字节数。  封装返回各种单位的大小（很实用）  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);  } | **删除文件**：  PHP使用unlink函数进行文件删除。  **unlink**($filename);  删除文件夹使用rmdir函数，文件夹必须为空，如果不为空或者没有权限则会提示失败。  **rmdir**($dir);  如果文件夹中存在文件，可以先循环删除目录中的所有文件，然后再删除该目录，循环删除可以使用glob函数遍历所有文件。  foreach (**glob("\*")** as $filename)  {    unlink($filename);  } |

### 文件上传

 原理：客户端上传文件，服务器端得到临时文件，保存到指定目录即为上传成功。

|  |  |  |
| --- | --- | --- |
| **客户端网页配置**：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(15).png  <form action="fileUpload.php" **method="post" enctype="multipart/form-data"**>       请选择要上传的文件：       <input **type="file"** name="myFile"></input><br>       <input type="submit" value="上传文件"></input>   </form>  **服务器端接收上传文件**：  //通过$\_FILES变量获取上传文件的相关信息  $file=**$\_FILES**['myFile'];  $filename=$file["name"];  $type=$file["type"];  $tmpPath=$file["tmp\_name"];  $size=$file["size"];  //将临时文件保存到指定目录下  //可以使用copy方法或move\_uploaded\_file方法  if(**move\_uploaded\_file**(  $tmpPath,"upfiles/".$filename))  {      echo "上传成功";  }else{      echo "上传失败";  } | **文件上传信息**：  通过$\_FILES得到的文件信息如下：  Array  (      [name] => copyright   客户端文件名      [type] => application/octet-stream    文件mine类型      [tmp\_name] => D:\PhpComeOn\tmp\php8C6E.tmp   服务器临时文件路径      [error] => 0  //错误号      [size] => 2171    文件大小  )  错误号取值：  0 上传成功；1 文件大小超过php配置的最大大小；2 文件大小超过表单限制大小；3 文件部分上传；4 没有选择上传文件；6 没有找到临时目录； 7 系统错误，写入失败；8 系统错误，文件被php扩展截断  **文件上传配置**：  php.ini:  ;;;;;;;;;;;;;;;;  ; File Uploads ;  ;;;;;;;;;;;;;;;;  file\_uploads = On   是否开启上传功能  upload\_tmp\_dir = "\PhpComeOn\tmp"  临时文件路径  upload\_max\_filesize = 2M   单个文件最大大小  max\_file\_uploads = 20   一起请求最多上传文件数  另外 Reousrce limits下的配置项也可以对文件上传做出限制 | **客户端限制上传文件**：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(16).png  第一个配置最大值必须配置，不然文件大小超过form最大值的时候，到达服务器端的$error仍然是0。  **服务器端限制上传文件**：   * 限制文件大小 * 限制上传文件类型 * 检测是否是真实图片 * 检测是否为POST方式上传   //判断上传文件的大小      if($fileInfo['size']>$maxSize){          exit('上传文件过大');      }  $ext=pathinfo($fileInfo['name'],PATHINFO\_EXTENSION);  //文件扩展名      if(!in\_array($ext,$allowExt)){  //$allowExt为允许扩展名的数组          exit('非法文件类型');      }   //判断文件是否是通过HTTP POST方式上传来的      if(!is\_uploaded\_file($fileInfo['tmp\_name'])){          exit('文件不是通过HTTP POST方式上传来的');      }  //检测是否为真实的图片类型  if(!getimagesize($fileInfo['tmp\_name'])){              exit('不是真正图片类型');          } |

### 文件下载

|  |  |  |
| --- | --- | --- |
| 1，通过超链接的方式  <a href="文件路径">超链接名</a>  该方式必须是浏览器支持的文件才可以直接使用 | 2，php程序：  <a href="fileDownload.php?filename=silvie.jpg">下载美女</a>  $filename="downloads/".$\_GET["filename"];//获取文件名 | //设置文件为附件形式   header("content-disposition:attachment; filename=".basename($filename));   header("content-length:".filesize($filename)); //文件大小   readfile($filename);//读取文件 |

## 8，日期/时间操作

|  |  |  |
| --- | --- | --- |
| **获取当前时间**：  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  **获取当前日期**：  php内置了date()函数，来取得当前的日期。  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】)；返回值：函数日期和时间  echo date("Y-m-d");//2014-03-30  echo date("Y-m-d",'1396193923'); | **获取某个日期时间对应的时间戳**：  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')  **strtotime**函数还可以接受一个英语字符串转换为时间戳：  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。 | **格式化日期时间**：  **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）;  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 |

## 9，异常处理

|  |  |
| --- | --- |
| **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 $e){       //对异常处理，方法：    1,echo '捕获异常: ' .$e->getMessage();        //2、不处理，将其再次抛出  }  抛出异常：  throw new Exception("异常提示-数字必须小于等于1"); | **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  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  getLine  异常所在行  getFile  异常所在文件 |

# PHP网站知识

## 1，cookie和session

|  |  |  |  |
| --- | --- | --- | --- |
| Cookie：（支持字符串）  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。且在使用会话Session时通常使用Cookie来存储会话id来识别用户        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。        在PHP中删除cookie也是采用setcookie函数来实现。  setcookie('test', '', time()-1); | setcookie具有7个可选参数，我们常用到的为前5个：  **name**（ Cookie名）：可以通过$\_COOKIE['name'] 进行访问  **value**（Cookie的值） **expire**（过期时间）：Unix时间戳格式，默认为0，表示浏览器关闭即失效  **path**（有效路径） ：如果路径设置为'/'，则整个网站都有效； 当设定了其他路径之后，则只在设定的路径以及子路径下有效  **domain**（有效域）：默认整个域名都有效，如果设置了'www.imooc.com',则只在www子域中有效 | **cookie**将数据存储在客户端，建立起用户与服务器之间的联系，通常可以解决很多问题，但是cookie仍然具有一些局限：  cookie相对不是太安全，容易被盗用导致cookie欺骗 单个cookie的值最大只能存储4k 每次请求都要进行网络传输，占用带宽  **session**是将用户的会话数据存储在服务端，没有大小限制，通过一个session\_id进行用户识别，PHP默认情况下session id是通过cookie来保存的，因此从某种程度上来说，seesion依赖于cookie。但这不是绝对的，session id也可以通过参数来实现，只要能将session id传递到服务端进行识别的机制都可以使用session。 | Session：（支持多种类型）  在PHP中使用session非常简单，先执行session\_start方法开启session，然后通过全局变量$\_SESSION进行session的读写。  session\_start();  $\_SESSION['test'] = time();  var\_dump($\_SESSION);  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。 |
| 删除Session项：  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$\_SESSION中去除，无法访问。  unset($\_SESSION['name']);  unset($\_SESSION); |  |  |  |

## 2，数据库操作

### Mysql

|  |  |  |  |
| --- | --- | --- | --- |
| **数据库扩展**：  PHP中一个数据库可能有一个或者多个扩展，其中既有官方的，也有第三方提供的。像Mysql常用的扩展有原生的mysql库，也可以使用增强版的mysqli扩展，还可以使用PDO进行连接与操作  **mysql扩展**：  $link = mysql\_connect('mysql\_host',  'mysql\_user', 'mysql\_password')  **or die**('数据库连接失败');  **mysqli扩展**：  $link = mysqli\_connect('mysql\_host',  'mysql\_user', 'mysql\_password')  or die('数据库连接失败');  **PDO扩展**：  $dsn = 'mysql:dbname=testdb;  host=127.0.0.1';  $user = 'dbuser'; $password = 'dbpass';  $dbh = new PDO($dsn, $user,   $password)  or die('数据库连接失败'); | **mysql扩展使用**：  连接数据库：见左列  选择要操作的数据库：  mysql\_select\_db('code1');  设置连接使用的字符编码：  mysql\_query("set names 'utf8'");  执行sql查询：  $res = mysql\_query('select \* from   user limit 1');  对于查询类的语句会返回一个资源句柄（resource），可以通过该资源获取查询结果集中的数据。  $row = **mysql\_fetch\_array**($res);  或**mysql\_fetch\_row** var\_dump($row);  执行插入操作：  $sql = "insert into user(name, age,   class) values('李四', 18, '高三一班')";  mysql\_query($sql); //执行插入语句  在mysql中，执行插入语句以后，可以得到自增的主键id,通过PHP的mysql\_insert\_id函数可以获取该id。  $uid = **mysql\_insert\_id**(); | 获取查询结果:  **mysql\_fetch\_array**，可以通过设定参数来更改行数据的下标，默认的会包含数字索引的下标以及字段名的关联索引下标。可以通过设定参数**MYSQL\_NUM**只获取数字索引数组，等同于**mysql\_fetch\_row**函数，如果设定参数为MYSQL\_ASSOC则只获取关联索引数组，等同于**mysql\_fetch\_assoc**函数。  如果要获取数据集中的所有数据，我们通过循环来遍历整个结果集。  $data = array(); while ($row = mysql\_fetch\_array  ($result)) {     $data[] = $row;  }  分页数据  通过mysql的limit可以很容易的实现分页，limit m,n表示从m行后取n行数据，在PHP中我们需要构造m与n来实现获取某一页的所有数据。  假定当前页为$page，每页显示$n条数据，那么m为当前页前面所有的数据，既$m = ($page-1) \* $n  $sql = "select \* from user **limit $m, $n**"; | 更新/删除数据:  数据的更新与删除相对比较简单，只需要构建好相应的sql语句，然后调用mysql\_query执行就能完成相应的更新与删除操作。  对于删除与更新操作，可以通过mysql\_affected\_rows函数来获取更新过的数据行数，如果数据没有变化，则结果为0  if (mysql\_query($sql)) {     echo **mysql\_affected\_rows**();  }  关闭数据库连接：  mysql\_close();  mysql\_close($link); |

### MySqli

|  |  |  |
| --- | --- | --- |
| **MySQLi扩展**：   mysql扩展不支持数据库的新的特性，因此推荐使用mysqli和PDO操作数据库。mysqli的优势：  1，面向对象，面向过程  2，支持预处理语句  3，支持事务  **mysqli配置**：  1，php.ini，开启php\_mysqli扩展  2，配置extension\_dir，扩展所在路径  3，重启服务器 | **面向对象方式使用mysqli流程**：  //1,建立mysql数据库连接  $mysqli=@new mysqli("localhost","barret","1230");//@取消默认警告  if($mysqli->connect\_errno){//错误编号      die("连接数据库错误".$mysqli->connect\_error);//错误信息  }  echo "客户端信息".$mysqli->client\_info."<br/>";  echo "客户端版本".$mysqli->client\_version."<br/>";  echo "服务器端信息".$mysqli->server\_info."<br/>";  echo "服务器端版本".$mysqli->server\_version."<br/>";  //2，选择数据库  $mysqli->select\_db("test");  //3，设置字符集  $mysqli->set\_charset("utf-8");  //4，执行sql查询语句  $sql="select \* from barret";  $result=$mysqli->query($sql);  //5 ，关闭连接  $mysqli->close(); | **插入记录**：  //4，插入记录sql语句  $sql="insert barret(name,age) values('shidandan',23);";  $result=$mysqli->query($sql);  if($result){      //执行成功      echo "插入一条记录成功，id为".$mysqli->insert\_id."<br/>";      echo "受影响的记录数：".$mysqli->affected\_rows."<br/>";  }else{      echo "插入记录错误，error".$mysqli->error."<br/>";  }  其他操作均为sql语句的不同，调用方法一样  **解析查询结果**：  if($result && $result->num\_rows >0){      $row=$result->fetch\_row();//得到一条记录，索引数组      $row=$result->fetch\_assoc();//得到一条记录，关联数组      $row=$result->fetch\_array();//得到一条记录，关联&索引数组      $row=$result->fetch\_object();//得到一条记录，对象格式      echo $row->age;      $result->data\_seek(0);//移动指针     $result->close();//释放结果集       } |
| 显示查询结果到网页表格中：  <h2 align="center">用户列表</h2>     <table align="center" border="1" cellspacing="0" cellpadding="0" width="80%" bgcolor="#ABCDE">         <tr>             <td>编号</td>             <td>用户名</td>             <td>年龄</td>         </tr>         <?php while($row=$result->fetch\_object()){?>         <tr>             <td><?php echo $row->id ?></td>             <td><?php echo $row->name ?></td>             <td><?php echo $row->age ?></td>         </tr>         <?php } $result->close(); ?>     </table> | **多条SQL语句同时执行**：  返回值仅表示第一条语句是否成功，前面的执行失败后面的不会执行  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(17).png  多查询时结果集的解析：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(18).png | **预处理语句**：  $sql="insert barret(name,age) values(?,?);";//占位符  //准备预处理语句  $stmt=$mysqli->prepare($sql);  //绑定参数  $stmt->bind\_param("si",$name,$age);//si表示参数类型  //执行预处理语句  if($stmt->execute()){      echo $stmt->insert\_id;  }else{      echo $stmt->error;  }  /释放结果集  $stmt->free\_result();  //关闭预处理语句  $stmt->close(); |
| **预处理语句解析结果集**：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(19).png | **mysqli事务处理**：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(20).png |  |

### PDO

|  |  |  |
| --- | --- | --- |
| **PDO数据库扩展**：      又称数据库访问抽象层，是最流行的数据库操作方式。特点：  1，编码一致性  2，灵活，性能高  3，面向对象，操作方便  **PDO使用配置**：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(21).png | **PDO连接数据库**：   try{       //1，通过参数连接数据库       $dsn="mysql:host=localhost;dname=test"; //连接字符串       $username="barret";       $password="1230";       $pdo=new PDO($dsn,$username,$password);       var\_dump($pdo);       //2，通过URL连接数据库       $dsn2="uri:file://G:\PhpSpace\\xampp  \htdocs\\PDO\dsn.txt";       $pdo=new PDO($dsn2,$username,$password);       var\_dump($pdo);       //3，配置文件连接数据库，php.ini       //pdo.dsn.数据源名即为$dsn参数   }catch(PDOException $e){       echo $e->getMessage();   } | **PDO主要方法**：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(22).png  $mt=$pdo->query($sql);//查询          foreach (**$mt as $row**) {              echo "编号".$row["id"]."<br/>";              echo "姓名".$row["name"]."<br/>";              echo "年龄".$row["age"]."<br/>";          } |
| **PDOstatement对象的方法**：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(23).png |  |  |

## 3，GD库图形图像

|  |  |  |
| --- | --- | --- |
| **GD库简单使用**：  1，新建一个画布，通过imagecreatetruecolor函数可以创建一个真色彩的空白图片：$img = **imagecreatetruecolor**(100, 100);  2，画笔颜色（RGB）  $red = **imagecolorallocate**($img, 0xFF,  0x00, 0x00);  3，绘制直线  **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,[文件路径]); 输出文件并保存  4, 释放图片内存  imagedestroy($img); | 5，图像中绘制文字  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.  imagestring($img, 5, 0, 0,  "Hello world", $red);  6，生成图像验证码  简单的验证码其实就是在图片中输出了几个字符，通过我们前面章节讲到的imagestring函数就能实现。  可以使用**imagesetpixel**绘制点来实现噪点干扰，但是只绘制一个点的作用不大，因此这里常常会使用循环进行随机绘制。  for($i=0;$i<50;$i++) {   imagesetpixel($im, rand(0, 100) ,    rand(0, 100) , $black);    imagesetpixel($im, rand(0, 100) ,  rand(0, 100) , $green);  } | 7，添加图片水印  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。  通过imagecreatefromjpeg可以直接从图片文件创建图像。  $im = **imagecreatefromjpeg**($filename);  添加图片水印  $logo = imagecreatefrompng($filename);  **imagecopy**($im, $logo, 15, 15, 0,   0, $width, $height);  imagejpeg($im, $filename);保存 |

### 代码实例：

|  |  |  |
| --- | --- | --- |
| **图片添加文字水印的代码**：  //1, 打开图片  $img="res/silvie.jpg"; //图片路径  $info=getimagesize($img); //获取图片信息，具体参考手  $type=image\_type\_to\_extension($info[2],false);//获取图片类型 索引2为类型成员  //2，根据图片创建画布  $fun="imagecreatefrom{$type}";//根据不同图片类型调用相的create函数  $canvas=$fun($img);   //创建画布  //3，添加文字水印  $font="res/simhei.ttf";  //选择水印字体  $content="love you，Silvie"; //水印内容  //设置画笔颜色透明度等  $pen=imagecolorallocatealpha($canvas, 255,  255, 255, 50);  //写入文字，并设置字体  **imagettftext($canvas,200,0,20,300,**  **$pen,$font,$content);**  //4，输出图片或保存图片  //输出到浏览器  header("content-type:{$info['mime']}");  $func="image{$type}";  $func($canvas);  //保存图片  $func($canvas,"res/newimg.$type");  //5，销毁图片  imagedestroy($canvas); | **图片添加图片水印的代码**：  //1, 打开图片  $img="res/silvie.jpg"; //图片路径  $info=getimagesize($img); //获取图片信息，具体参考手册  $type=image\_type\_to\_extension($info[2],false); //获取图片类型 索引2为类型成员  //2，根据图片创建画布  $fun="imagecreatefrom{$type}"; //根据不同图片类型调用相应的create函数  $canvas=$fun($img);   //创建画布  //3，添加文字水印  $icon=imagecreatefrompng("res/icon.png");//获取水印图片  $iconInfo=getimagesize("res/icon.png");  //添加图片水印，合并图片  **imagecopymerge($canvas, $icon, 20, 30, 0, 0, $iconInfo[0], $iconInfo[1],50);**  //4，输出图片或保存图片  //输出到浏览器  header("content-type:{$info['mime']}");  $func="image{$type}";  $func($canvas);  //保存图片  $func($canvas,"res/newicon.$type");  //5，销毁图片  imagedestroy($canvas); | **图片缩略图（压缩）的代码：**   //1, 打开图片  $img="res/silvie.jpg"; //图片路径  $info=getimagesize($img); //获取图片信息，具体参考手册  $type=image\_type\_to\_extension($info[2],false); //获取图片类型 索引2为类型成员  //2，根据图片创建画布  $fun="imagecreatefrom{$type}"; //根据不同图片类型调用相应的create函数  $canvas=$fun($img);   //创建画布  //3，创建一个真色彩空白画布  $dst=imagecreatetruecolor(200,300);  //4，将原图压缩到空白画布上  **imagecopyresampled($dst, $canvas, 0, 0, 0, 0, 200, 300, $info[0], $info[1]);**  //5，输出图片或保存图片  //输出到浏览器  header("content-type:{$info['mime']}");  $func="image{$type}";  $func($dst);  //保存图片  $func($dst,"res/newsmall.$type");  //6，销毁图片  imagedestroy($canvas);  imagedestroy($dst); |
| **数字验证码的代码实现**：  //1创建一个白色的画布区域  //画布,默认黑色   $img=imagecreatetruecolor(100,30);   //设置背景为白色   $bgcolor=**imagecolorallocate**($img,255,255,255);   //填充背景   imagefill($img,0,0,$bgcolor);  //2，底图画布上添加数字验证码   for($i=0;$i<4;$i++){       $size=6;  $color=imagecolorallocate($img,rand(0,120),  rand(0,120),rand(0,120));// 随机颜色       $content=rand(0,9);//0-9随机数字       //绘制位置，防止重合       $x=($i\*100/4)+rand(5,10);       $y=rand(5,10);       //绘制字符串到底图  **imagestring**($img,$size,$x,$y,$content,$color);   }  //3，添加点干扰元素   for($i=0;$i<200;$i++) {   $color=imagecolorallocate($img,rand(50,120),  rand(50,120),rand(50,120));// 随机颜色  **imagesetpixel**($img, rand(0, 100) , rand(0, 100),  $color);  }  //4，添加线干扰元素  for($i=0;$i<3;$i++) {   $color=imagecolorallocate($img,rand(80,220),  rand(80,220),rand(80,220));// 随机色  **imageline**($img,rand(1,99),rand(1,29),rand(1,99),  rand(1,29),$color)  }  //5，显示到网页  header("content-type: image/png");  imagepng($img);  imagedestroy($img); | **字母+数字验证码的代码实现（仅生成部分）**:  //2，底图画布上添加数字+字母 验证码   for($i=0;$i<4;$i++){       $size=6;  $color=imagecolorallocate($img,rand(0,120),  rand(0,120),rand(0,120));// 随机颜色    **$data="abcdefghijklmnopqrtuvwxyz1234567890";**  **$content=substr($data,rand(0,strlen($data)-1),1)**;//字母+数字       //绘制位置，防止重合       $x=($i\*100/4)+rand(5,10);       $y=rand(5,10);       //绘制字符串到底图       imagestring($img,$size,$x,$y,$content,$color);   }  **验证码验证原理**：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(2).png  1，服务器端生成验证码时保存验证码到session中：  session\_start();//启动session会话，必须在脚本第一句  $code="";// 用于保存验证码  $data="abcdefghijklmnopqrtuvwxyz1234567890";   $content=substr($data,rand(0,strlen($data)-1),1);//字母+数字   $code.=$content;  2，html网页显示验证码图片，并把用户输入交给后台php判断:  <p>验证码图片：<img **src="imageSession.php?r=<php echo rand();?>"** border="1" width="100" height="30">  <a href="javascript:void(0)" onclick="changeImg()">看不清</a>  </p>  <p>请输入图片的内容：<input type="text" name="authcode" value=""/></p> | changeImg用于动态更换验证码图片：  function changeImg(){          var img=document.getElementById("img");          img.src="imageSession.php?r="+Math.random();      }  3，后台php比较用户输入和session中保存的验证码：  session\_start();   $autocode=strtolower($\_POST["autocode"]);//获取用户的输入，全部变成小写   if($autocode == $\_SESSION["autocode"]){       echo "输入正确";   }else{       echo "输入失败";   } |

## 4，页面静态化

|  |  |  |
| --- | --- | --- |
| **优化页面响应时间**：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(24).png  动态页面静态化就是最终生成html文件，直接被浏览器解析。不适合内容经常变动的网页。 | **静态化介绍**：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image.png | **伪静态：**动态url地址设置为静态形式  通过服务器配置的，映射到静态url，称为伪静态  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(25).png  **注意：局部静态化：又称Ajax技术** |
| **伪静态的配置**：  1，php处理伪静态url  （1）apache中配置开启PATH\_INFO：      directory节点下：AcceptPathInfo On  （2）php文件中处理url   if(preg\_match("/^\/(\d+).html/", $\_SERVER["PATH\_INFO"],$arr)){   $type=$arr[1];   header("Location:http://localhost/BarretPHP/  FalseStatic/{$type}.html"); //重定向网页  2，服务器配置：(见lamp配置）  (1)虚拟域名  (2)httpd\_vhost.conf文件配置 |  | **PHP实现纯静态化**：  1，file\_put\_contents()函数：将内容数据写到文件  2，使用php内置缓存机制：  C:\Users\BARRET~1\AppData\Local\Temp\enhtmlclip\Image(1).png |