第 3 章

PHP 常用内置函数

为便于各类应用的开发，PHP 提供了极为丰富的各种内置函数。适当使用 PHP 的内置函数，既可简化 PHP 程序代码的编写，又可方便地实现所需要的有关功能。本章主要对 PHP Web 应用开发中常用的一些内置函数进行简要介绍，包括数学函数、字符串处理函数、日期和时间函数、文件操作函数、检测函数等。

3.1 数学函数


视频讲解

数学函数主要用于实现各种数学运算，或对有关的数值进行相应的处理。PHP 所提供的数学函数是十分全面的，常用的有 abs()、floor()、ceil()、round()、rand()、pow()与 sqrt()等。

1. abs()函数

abs()为绝对值函数，其语法格式为

```
number abs(mixed $value)
```

参数：value 用于指定要处理的数值，其类型为 mixed(即可以接受多种不同的类型)。

返回值：value 的绝对值，其类型为 number(即 integer 或 float)。若参数 value 的类型为 float，则返回值的类型也为 float，否则为 integer。

【例 3.1】 abs()函数应用示例(math01.php)。

```
<?php
echo "abs(1)=".abs(1)."<br>";
echo "abs(0)=".abs(0)."<br>";
echo "abs(-1)=".abs(-1)."<br>";
echo "abs(-1.5)=".abs(-1.5)."<br>";
?>
```

该示例的运行结果如图 3.1 所示。

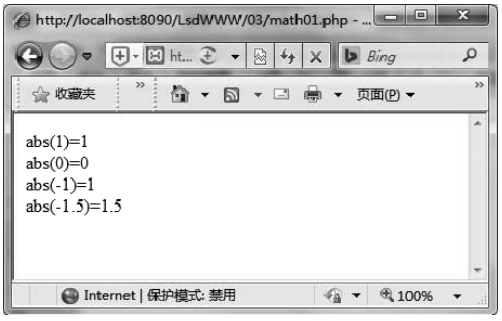

图 3.1　程序 math01.php 的运行结果

2. floor()与 ceil()函数

floor()与 ceil()均为取整函数,其语法格式为

```
float floor(float $value)
float ceil(float $value)
```

参数：value 用于指定要处理的数值,其类型为 float。

返回值：value 经过取整后的值,其类型为 float 型。其中,floor()函数采用舍去法取整,返回不大于 value 的最接近的整数;ceil()函数则采用进一法取整,返回不小于 value 的下一个整数。

【例 3.2】 floor()与 ceil()函数应用示例(math02.php)。

```
<?php
echo "floor(3.14)=".floor(3.14)."<br>";
echo "floor(9.99)=".floor(9.99)."<br>";
echo "floor(-3.14)=".floor(-3.14)."<br>";
echo "ceil(3.14)=".ceil(3.14)."<br>";
echo "ceil(9.99)=".ceil(9.99)."<br>";
echo "ceil(-3.14)=".ceil(-3.14)."<br>";
?>
```

该示例的运行结果如图 3.2 所示。

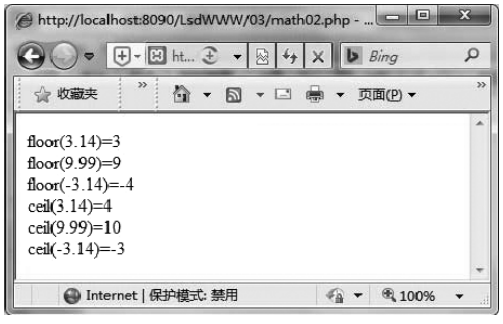

图 3.2　程序 math02.php 的运行结果

3. round()函数

round()为四舍五入函数，其语法格式为

```
float round(float $value[,int $precision=0])
```

参数：value 用于指定要处理的数值；precision(可选)用于指定四舍五入的精度(即十进制小数点后数字的数目)，可以是正数、负数或零(默认值)。

返回值：value 按精度 precision 四舍五入后的值，其类型为 float。

【例 3.3】 round()函数应用示例(math03.php)。

```
<?php
echo "round(125.456,2)=".round(125.456,2)."<br>";
echo "round(125.456,1)=".round(125.456,1)."<br>";
echo "round(125.456,0)=".round(125.456,0)."<br>";
echo "round(125.456,-1)=".round(125.456,-1)."<br>";
echo "round(125.456,-2)=".round(125.456,-2)."<br>";
echo "round(125.456)=".round(125.456)."<br>";
?>
```

该示例的运行结果如图 3.3 所示。

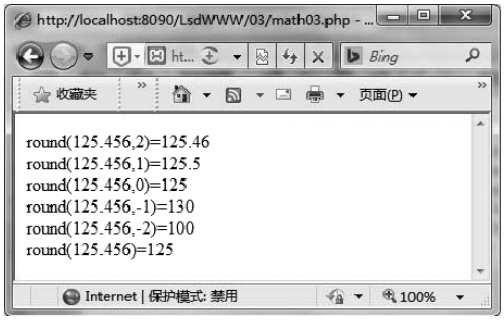

图 3.3　程序 math03.php 的运行结果

4. rand()函数

rand()为随机整数函数，其语法格式为

```
int rand([int $min,int $max])
```

参数：min(可选)用于指定最小值，默认为 0；max(可选)用于指定最大值，默认为 getrandmax()，即随机整数最大的可能值。

返回值：返回 min 与 max 之间(包括 min 与 max)的伪随机整数。未指定 min 与 max 时，则返回 0 与 getrandmax()之间的伪随机整数。

【例 3.4】 rand()函数应用示例(math04.php)。

```
<?php
echo "rand()=".rand()."<br>";
echo "rand()=".rand()."<br>";
echo "rand()=".rand()."<br>";
```

```
echo "rand(1,100)=".rand(1,100)."<br>";
echo "rand(1,100)=".rand(1,100)."<br>";
echo "rand(1,100)=".rand(1,100)."<br>";
?>
```

该示例的运行结果如图 3.4 所示。

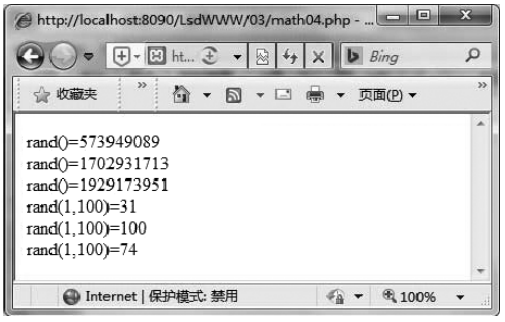

图 3.4　程序 math04.php 的运行结果

5. pow()函数

pow()函数为乘方函数,其语法格式为

```
number pow(number $base, number $exp)
```

参数：base 用于指定底数,exp 用于指定指数。

返回值：base 的 exp 次幂。

【例 3.5】 pow()函数应用示例(math05.php)。

```
<?php
echo "pow(0,0)=".pow(0,0)."<br>";
echo "pow(2,0)=".pow(2,0)."<br>";
echo "pow(2,3)=".pow(2,3)."<br>";
echo "pow(2,-3)=".pow(2,-3)."<br>";
echo "pow(-2,3)=".pow(-2,3)."<br>";
echo "pow(-2,-3)=".pow(-2,-3)."<br>";
?>
```

该示例的运行结果如图 3.5 所示。

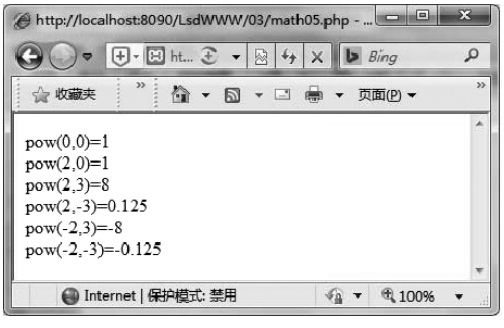

图 3.5　程序 math05.php 的运行结果

6. sqrt()函数

sqrt()函数为平方根函数，其语法格式为

```
float sqrt(float $value)
```

参数：value 用于指定要处理的数值。

返回值：value 的平方根。若 value 为负数，则返回 NAN。

【例 3.6】 sqrt()函数应用示例(math06.php)。

```
<?php
echo "sqrt(0)=".sqrt(0)."<br>";
echo "sqrt(100)=".sqrt(100)."<br>";
echo "sqrt(-100)=".sqrt(-100)."<br>";
echo "sqrt(101)=".sqrt(101)."<br>";
echo "sqrt(2.25)=".sqrt(2.25)."<br>";
?>
```

该示例的运行结果如图 3.6 所示。

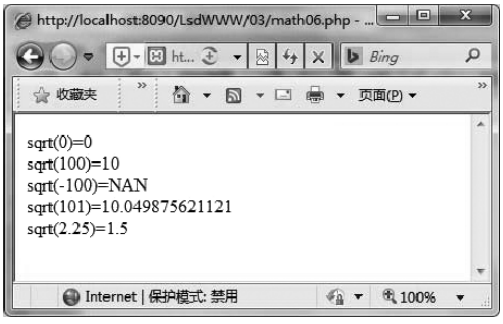

图 3.6　程序 math06.php 的运行结果

3.2　字符串处理函数


视频讲解

字符串处理函数主要用于对字符串进行相应的处理。PHP 所提供的字符串处理函数是相当丰富的，常用的有 strlen()、trim()、substr()、strtoupper()、strtolower()、strcmp()与 str_replace()等。

1. strlen()函数

strlen()函数用于获取字符串的长度(即字符串中所包含的字符的个数)，其语法格式为

```
int strlen(string str)
```

参数：str 用于指定需要计算其长度的字符串。

返回值：字符串 str 的长度。若 str 为空，则返回 0。

【例 3.7】 strlen()函数应用示例(string01.php)。

```
<?php
$str="Hello,World!";
echo strlen($str)."<br>";
$str="Hello, World!";
echo strlen($str)."<br>";
?>
```

该示例的运行结果如图 3.7 所示。

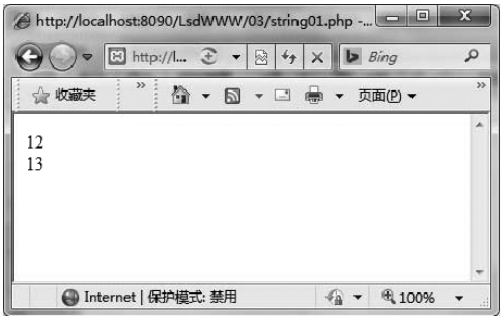

图 3.7 程序 string01.php 的运行结果

2. trim()函数

trim()函数用于删除字符串首尾处的空白字符(或指定字符),其语法格式为

```
string trim( string str[, string charlist])
```

参数:str 用于指定需要处理的字符串;charlist(可选)用于指定需要删除的字符,既可逐一列出,也可使用".."列出一个字符范围。若不指定 charlist 参数,则表示要删除空白字符,包括普通空格符(" ")、制表符("\t")、换行符("\n")、回车符("\r")、空字节符("\0")与垂直制表符("\x0B")。

返回值:删除了首尾处空白字符(或指定字符)的字符串。

说明:如果只要删除字符串开头或末尾的空白字符(或指定字符),那么可分别使用 ltrim()或 rtrim()函数。这两个函数的语法格式与使用方法类似于 trim()函数。

【例 3.8】 trim()函数应用示例(string02.php)。

```
<?php
$str="\t#Hello,World!...   ";
echo strlen($str)."<br>";
$str=trim($str);
echo strlen($str)."<br>";
echo $str."<br>";
$str=trim($str,"#.");
echo $str."<br>";
?>
```

该示例的运行结果如图 3.8 所示。

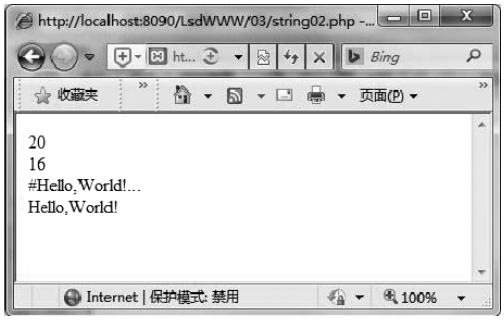

图 3.8　程序 string02.php 的运行结果

3. substr()函数

substr()函数用于获取字符串的子串,其语法格式为

```
string substr(string str, int start[, int length])
```

参数：str 用于指定需要从中获取子串的字符串;start 用于指定子串开始的位置(从 0 开始计算);length(可选)用于指定子串的长度。未指定 length 参数时,表示子串从位置 start 处开始直到字符串结尾。

返回值：成功时返回字符串 str 从位置 start 处开始最大长度为 length 的子串,失败时返回 FALSE。特别地,若 length 值为 0、FALSE 或 NULL,则返回一个空字符串。

说明：若参数 start 为负数,则表示从倒数第 start 个字符开始;若参数 length 为负数,则表示忽略字符串末尾的 length 个字符(即提取到倒数第 length 个字符止)。

【例 3.9】 substr() 函数应用示例(string03.php)。

```
<?php
$str="Hello,World!";
echo substr($str,0,1)."<br>";
echo substr($str,6,5)."<br>";
echo substr($str,6)."<br>";
echo substr($str,-6)."<br>";
echo substr($str,-6,5)."<br>";
echo substr($str,-6,-5)."<br>";
?>
```

该示例的运行结果如图 3.9 所示。

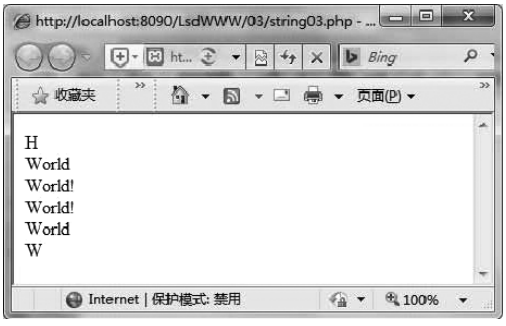

图 3.9　程序 string03.php 的运行结果

4. strtoupper()与strtolower()函数

strtoupper()与strtolower()函数用于将字符串中的所有字母转换为大写或小写,其语法格式为

```
string strtolower(string $str)
string strtoupper(string $str)
```

参数:str用于指定需要处理的字符串。

返回值:strtoupper()函数返回转换后的大写字符串,strtolower()函数返回转换后的小写字符串。

【例3.10】 strtoupper()与strtolower()函数应用示例(string04.php)。

```
<?php
$str="Hello,Abc123!";
$str1=strtoupper($str);
$str2=strtolower($str);
echo $str."<br>";
echo $str1."<br>";
echo $str2."<br>";
?>
```

该示例的运行结果如图3.10所示。

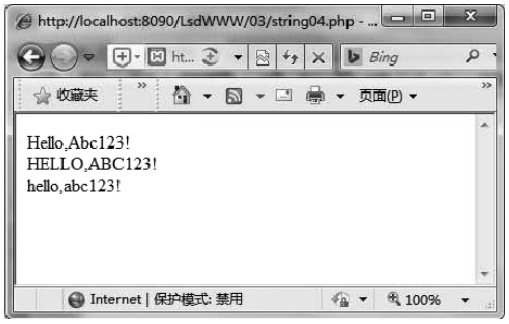

图3.10 程序string04.php的运行结果

5. strcmp()函数

strcmp()函数用于比较两个字符串的大小,其语法格式为

```
int strcmp(string str1, string str2)
```

参数:str1用于指定第一个字符串;str2用于指定第二个字符串。

返回值:若str1大于str2,则返回值大于0;若str1小于str2,则返回值小于0;若str1等于str2,则返回值为0。

【例3.11】 strcmp()函数应用示例(string05.php)。

```
<?php
$str1="Hello";
$str2="hello";
```

```
if (strcmp($str1,$str2)>0)
    echo "$str1>$str2";
elseif (strcmp($str1,$str2)==0)
    echo "$str1=$str2";
else
    echo "$str1<$str2";
?>
```

该示例的运行结果如图 3.11 所示。

6. str_replace()函数

str_replace()函数用于实现字符串子串的替换，其基本的语法格式为

```
string str_replace(string search, string replace, string str)
```

参数：search 用于指定需要替换的子串，replace 用于指定替换后的子串，str 用于指定需要处理的字符串。

返回值：替换后的字符串。

【例 3.12】 str_replace()函数应用示例(string06.php)。

```
<?php
$str="我是一位学生!";
echo "替换前:<br>";
echo $str."<br>";
$str=str_replace("学生","教师",$str);
echo "替换后:<br>";
echo $str."<br>";
?>
```

该示例的运行结果如图 3.12 所示。

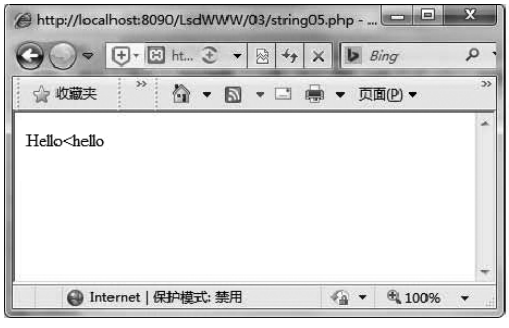

图 3.11　程序 string05.php 的运行结果

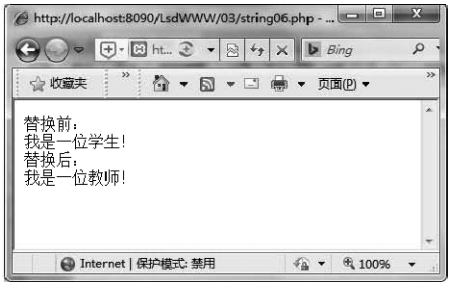

图 3.12　程序 string06.php 的运行结果

3.3　日期和时间处理函数

视频讲解

日期和时间处理函数主要用于获取相应的日期与时间，或对日期与时间进行相应的处理(如格式化等)。PHP 提供了一系列日期与时间处理函数，常用的有 time()、date()、

getdate()、mktime()与 checkdate()等。

1. time()函数

time()函数用于获取当前的 UNIX 时间戳,其语法格式为

```
int time()
```

返回值:当前的 UNIX 时间戳,即从 UNIX 纪元(格林尼治时间 1970 年 1 月 1 日 00:00:00)到当前时间的秒数。

2. date()函数

date()函数用于对时间/日期进行格式化,其语法格式为

```
string date(string format[, int timestamp])
```

参数:format 用于指定所使用的格式字符串(其中可包含的常用格式字符如表 3.1 所示);timestamp(可选)用于指定需要进行格式化的 UNIX 时间戳。未指定 timestamp 参数时,则默认为当前的本地时间,即 time()函数的返回值。

表 3.1 format 参数中的常用格式字符

格式字符	说明
a	小写的上午和下午值(am 或 pm)
A	大写的上午和下午值(AM 或 PM)
d	月份中的第几天,有前导零的两位数字(01～31)
D	星期中的第几天,文本表示,三个字母(Mon～Sun)
F	月份,完整的文本格式(January～December)
g	小时,12 小时格式,没有前导零(1～12)
G	小时,24 小时格式,没有前导零(0～23)
h	小时,12 小时格式,有前导零(01～12)
H	小时,24 小时格式,有前导零(00～23)
i	有前导零的分钟数(00～59)
j	月份中的第几天,没有前导零(1～31)
l	星期几,完整的文本格式(Sunday～Saturday)
m	数字表示的月份,有前导零(01～12)
M	三个字母缩写表示的月份(Jan～Dec)
n	数字表示的月份,没有前导零(1～12)
N	数字表示的星期中的第几天(PHP 5.1.0 新增,1～7,其中,1 表示星期一,7 表示星期日)
s	秒数,有前导零(00～59)
S	每月天数后面的英文后缀,两个字符(st、nd、rd 或 th)
t	指定月份所应有的天数(28～31)

续表

格式字符	说　明
T	本机所在的时区，如 EST、MDT 等
w	星期中的第几天，数字表示（0～6，其中，0 表示星期天，6 表示星期六）
y	年份，两位数字（如 96、06 等）
Y	年份，4 位数字（如 1996、2006 等）
z	年份中的第几天（0～366）

返回值：格式化后的日期时间字符串。若 timestamp 参数不是一个有效数值，则返回 FALSE。

【例 3.13】 time()与 date()函数应用示例（datetime01.php）。

```
<?php
$now=time();
echo $now."<br>";
echo date("Y-m-d",$now)."<br>";
echo date("H:i:s",$now)."<br>";
echo date("Y-m-d H:i:s",$now)."<br>";
echo date("Y 年 m 月 d 日 H 时 i 分 s 秒",$now)."<br>";
echo date("l",$now)."<br>";
?>
```

该示例的运行结果如图 3.13 所示。

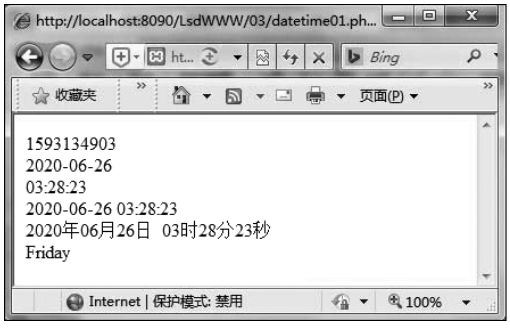

图 3.13　程序 datetime01.php 的运行结果

3. getdate()函数

getdate()函数用于获取日期/时间的信息，其语法格式为

```
array getdate([int timestamp])
```

参数：timestamp（可选）用于指定需要获取其信息的 UNIX 时间戳。未指定 timestamp 参数时，则默认为当前的本地时间，即 time()函数的返回值。

返回值：一个包含日期/时间信息的关联数组（其键名如表 3.2 所示）。

表 3.2　getdate()函数返回的关联数组的键名

键　　名	说　　明
seconds	秒的数字表示(0～59)
minutes	分钟的数字表示(0～59)
hours	小时的数字表示(0～23)
mday	月份中第几天的数字表示(1～31)
wday	星期中第几天的数字表示(0～6,其中,0 表示星期日,6 表示星期六)
mon	月份的数字表示(1～12)
year	年份的数字表示(4 位数字,如 1996、2006 等)
yday	一年中第几天的数字表示(0～365)
weekday	星期几的完整文本表示(Sunday～Saturday)
month	月份的完整文本表示(January～December)
0	自 UNIX 纪元开始至今的秒数(系统相关,典型值为－2 147 483 648～2 147 483 647)

【例 3.14】 getdate()函数应用示例(datetime02.php)。

```
<?php
$now=time();
$dtstr=date("Y-m-d l H:i:s",$now);
$dtinfo=getdate($now);
echo $now."<br>";
echo $dtstr."<br>";
print_r($dtinfo);
?>
```

该示例的运行结果如图 3.14 所示。

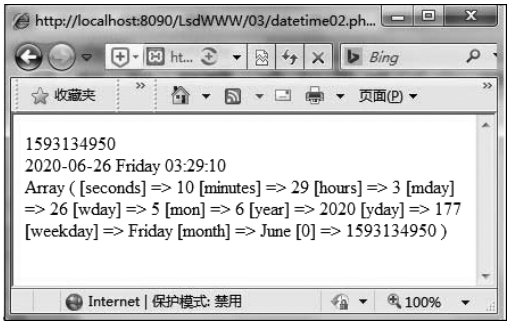

图 3.14　程序 datetime02.php 的运行结果

4. mktime()函数

mktime()函数用于获取一个日期/时间的 UNIX 时间戳,其语法格式为

```
int mktime([int hour[, int minute[, int second[, int month[, int day[, int year]]]]]])
```

参数：hour、minute、second、month、day 与 year 均为可选参数，分别用于指定小时数、分钟数、秒数(一分钟之内)、月份数、天数与年份数(两位或四位数字，其中，0～69 对应于 2000—2069，70～100 对应于 1970—2000)。参数可以从右向左省略，任何省略的参数会被设置成本地日期和时间的当前值。

返回值：与参数所指定的日期和时间相对应的 UNIX 时间戳。若参数非法，则返回 FALSE(在 PHP 5.1 之前则返回－1)。

【例 3.15】 mktime()函数应用示例(datetime03.php)。

```
<?php
$now=mktime(21,30,28,6,26,1996);
$dtstr=date("Y-m-d l H:i:s",$now);
$dtinfo=getdate($now);
echo $now."<br>";
echo $dtstr."<br>";
print_r($dtinfo);
?>
```

该示例的运行结果如图 3.15 所示。

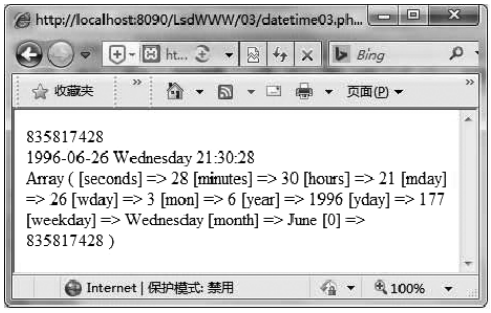

图 3.15　程序 datetime03.php 的运行结果

5. checkdate()函数

checkdate()函数用于检测一个日期的有效性(或合法性)，其语法格式为

```
bool checkdate(int month, int day, int year)
```

参数：month 用于指定月份数(1～12)，day 用于指定天数(1～31)，year 用于指定年份数(1～32 767)。

返回值：若参数所指定的日期有效(year 值为 1～32 767，month 值为 1～12，day 值在相应年份与月份所应具有的天数范围之内)，则返回 TRUE，否则返回 FALSE。

【例 3.16】 checkdate()函数应用示例(datetime04.php)。

```
<?php
$year=2020;
$month=2;
$day1=29;
$day2=30;
```

```
if (checkdate($month,$day1,$year))
    echo $year."-".$month."-".$day1."是一个有效日期.<br>";
if (!checkdate($month,$day2,$year))
    echo $year."-".$month."-".$day2."是一个无效日期.<br>";
?>
```

该示例的运行结果如图 3.16 所示。

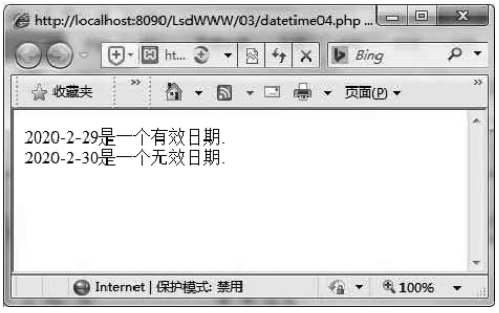

图 3.16　程序 datetime04.php 的运行结果


视频讲解

3.4　文件操作函数

文件操作函数主要用于对文件进行相应的操作，如文件的创建、读取、写入、复制、重命名与删除等。PHP 提供了一系列的文件操作函数，常用的有 fopen()、fclose()、fwrite()、fgetc()、feof()、fgets()、file()、filesize()、fread()、copy()、rename()与 unlink()等。

1. fopen()函数

fopen()函数主要用于打开一个文件，其基本的语法格式为

```
resource fopen(string filename, string mode)
```

参数：filename 用于指定文件名，mode 用于指定文件的打开方式(常用的文件打开方式如表 3.3 所示)。

表 3.3　常用的文件打开方式

打开方式	说　明
r	以只读方式打开，将文件指针指向文件头
r+	以读写方式打开，将文件指针指向文件头
w	以写入方式打开，将文件指针指向文件头并将文件大小截为零。若文件不存在，则尝试创建
w+	以读写方式打开，将文件指针指向文件头并将文件大小截为零。若文件不存在，则尝试创建
a	以写入方式打开，将文件指针指向文件末尾。若文件不存在，则尝试创建

续表

打开方式	说　明
a+	以读写方式打开,将文件指针指向文件末尾。若文件不存在,则尝试创建
x	创建并以写入方式打开,将文件指针指向文件头。若文件已存在则打开失败,若文件不存在,则尝试创建
x+	创建并以读写方式打开,将文件指针指向文件头。若文件已存在,则打开失败,若文件不存在,则尝试创建
rb	以只读方式打开(二进制模式),将文件指针指向文件头
rb+	以读写方式打开(二进制模式),将文件指针指向文件头
wb	以写入方式打开(二进制模式),将文件指针指向文件头并将文件大小截为零。若文件不存在,则尝试创建
wb+	以读写方式打开(二进制模式),将文件指针指向文件头并将文件大小截为零。若文件不存在,则尝试创建
ab	以写入方式打开(二进制模式),将文件指针指向文件末尾。若文件不存在,则尝试创建
ab+	以读写方式打开(二进制模式),将文件指针指向文件末尾。若文件不存在,则尝试创建
xb	创建并以写入方式打开(二进制模式),将文件指针指向文件头。若文件已存在,则打开失败,若文件不存在,则尝试创建
xb+	创建并以读写方式打开(二进制模式),将文件指针指向文件头。若文件已存在,则打开失败,若文件不存在,则尝试创建

返回值:成功时返回相应的文件指针,失败时返回 FALSE。

2. fclose()函数

fclose()函数用于关闭一个已打开的文件,其语法格式为

```
bool fclose(resource handle)
```

参数:handle 为文件指针(该文件指针指向欲关闭的文件)。

返回值:成功时返回 TRUE,失败时返回 FALSE。

【例 3.17】 fopen()与 fclose()函数应用示例(file01.php)。

```
<?php
$fn="test.txt";
$fp=fopen($fn, "w");
if (!$fp){
    echo $fn."文件打开失败!<br>";
    die();
}
echo $fn."文件打开成功!<BR>";
if(!fclose($fp))
{
    echo $fn."文件关闭失败!<br>";
    die();
}
```

```
echo $fn."文件关闭成功!<BR>";
?>
```

该示例的运行结果如图 3.17 所示。

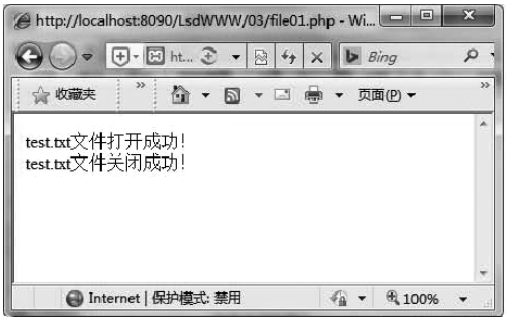

图 3.17　程序 file01.php 的运行结果

3. fwrite()函数

fwrite()函数用于向文件写入内容,其语法格式为

```
int fwrite(resource handle, string str[, int length])
```

参数：handle 为文件指针(该文件指针指向要向其写入内容的文件);str 用于指定需要写入的内容,length(可选)用于指定写入内容的长度(或字节数)。

返回值：成功时返回写入内容的字节数(写入过程在已写入了 length 字节或者已写完了 str 时停止),失败时返回 FALSE。

【例 3.18】 fwrite()函数应用示例(file02.php)。

```
<?php
$fn="test.txt";
$fp=fopen($fn, "w");
if (!$fp){
    echo "文件打开失败!<br>";
    die();
}
$str="Hello,World!\n";
if (($n=fwrite($fp,$str))!=FALSE)
    echo "写入了".$n."字节的内容.<br>";
else
    echo "写入失败!<br>";
$str="你好,世界!\n";
if (($n=fwrite($fp,$str))!=FALSE)
    echo "写入了".$n."字节的内容.<br>";
else
    echo "写入失败!<br>";
fclose($fp);
?>
```

该示例的运行结果如图 3.18 所示。

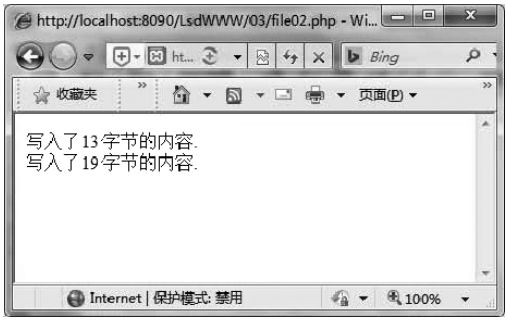

图 3.18　程序 file02.php 的运行结果

4. fgetc()函数

fgetc()函数用于从文件中读取一个字符,其语法格式为

```
string fgetc(resource handle)
```

参数：handle 为文件指针(该文件指针指向要从其读取内容的文件)。
返回值：包含所读取到的一个字符的字符串。若已到达 EOF,则返回 FALSE。

【例 3.19】 fgetc()函数应用示例(file03.php)。

```
<?php
$fn="test.txt";
$fp=fopen($fn, "r");
if (!$fp){
    echo "文件打开失败!<br>";
    die();
}
while (($str=fgetc($fp))!=FALSE){
    if ($str!="\n")
        echo $str;
    else
        echo "<br>";
}
fclose($fp);
?>
```

该示例的运行结果如图 3.19 所示。

5. feof()函数

feof()函数用于检测文件指针是否已到达文件末尾(EOF),其语法格式为

```
bool feof(resource handle)
```

参数：handle 为文件指针。
返回值：文件指针已到达 EOF 时返回 TRUE,否则返回 FALSE。

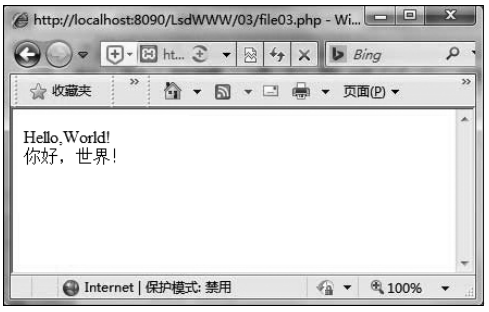

图 3.19　程序 file03.php 的运行结果

【例 3.20】 feof()函数应用示例(file04.php)。

```
<?php
$fn="test.txt";
$fp=fopen($fn, "r");
if (!$fp){
    echo "文件打开失败!<br>";
    die();
}
while (!feof($fp)){
    $str=fgetc($fp);
    if ($str!="\n")
        echo $str;
    else
        echo "<br>";
}
fclose($fp);
?>
```

该示例的运行结果如图 3.20 所示。

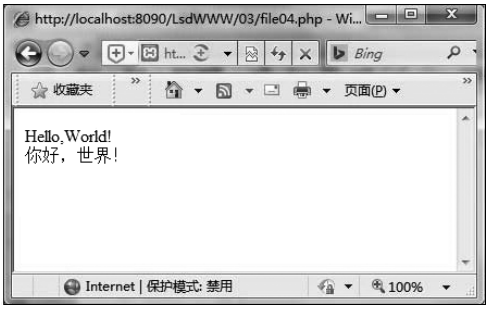

图 3.20　程序 file04.php 的运行结果

6. fgets()函数

fgets()函数用于从文件中读取一行,其语法格式为

```
string fgets(resource handle[, int length])
```

参数：handle 为文件指针(该文件指针指向要从其读取内容的文件)；length(可选)用于指定长度(或字节数)。未指定 length 参数时，则默认为 1KB 或 1024B(从 PHP 4.3 开始，则忽略掉此假定，而改为至行末结束)。

返回值：成功时返回所读取到的最多包含 length－1 个字符的字符串(读取过程在碰到换行符、到达 EOF 或者已读取了 length－1 字节时停止)，失败时返回 FALSE。

【例 3.21】 fgets()函数应用示例(file05.php)。

```
<?php
$fn="test.txt";
$fp=fopen($fn, "r");
if (!$fp){
    echo "文件打开失败!<br>";
    die();
}
while (!feof($fp)){
    $str=fgets($fp);
    echo $str."<br>";
}
fclose($fp);
?>
```

该示例的运行结果如图 3.21 所示。

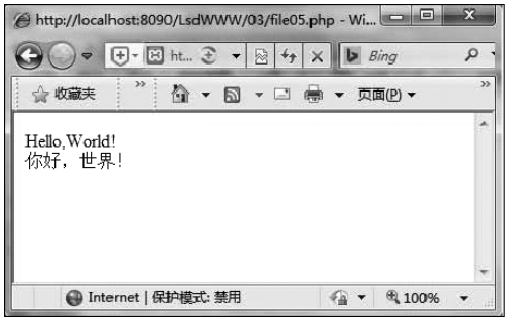

图 3.21　程序 file05.php 的运行结果

7. file()函数

file()函数用于将整个文件读取到一个数组中，其基本的语法格式为

```
array file(string filename)
```

参数：filename 用于指定文件名。

返回值：成功时返回包含整个文件内容的数组(每个元素存放文件的一行，包括换行符在内)，失败时返回 FALSE。

【例 3.22】 file()函数应用示例(file06.php)。

```
<?php
$fn="test.txt";
```

```
$lines=file($fn);
foreach ($lines as $n=>$line){
    echo $n.": ".$line."<br>";
}
?>
```

该示例的运行结果如图 3.22 所示。

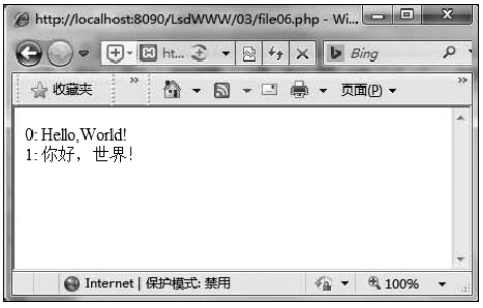

图 3.22　程序 file06.php 的运行结果

8. filesize()函数

filesize()函数用于获取文件的大小(即字节数),其语法格式为

```
int filesize(string filename)
```

参数：filename 用于指定文件名。

返回值：成功时返回指定文件的字节数,失败时返回 FALSE。

9. fread()函数

fread()函数用于读取文件的内容,其语法格式为

```
string fread(resource handle, int length)
```

参数：handle 为文件指针(该文件指针指向要从其读取内容的文件);length 用于指定长度(或字节数)。

返回值：成功时返回所读取到的最多包含 length 个字符的字符串(读取过程在到达 EOF 或者已读取了 length 字节时停止),失败时返回 FALSE。

【例 3.23】 filesize()与 fread()函数应用示例(file07.php)。

```
<?php
$fn="test.txt";
$fp=fopen($fn, "r");
if (!$fp){
    echo "文件打开失败!<br>";
    die();
}
$str=fread($fp,filesize($fn));
echo str_replace("\n","<br>",$str);
fclose($fp);
?>
```