第 5 章　PHP 的缓存与静态化应用

前面已经介绍了许多 PHP 应用的例子，这些例子无一例外是通过访问 PHP 脚本文件来完成的。用户每次从浏览器端的访问都是对 PHP 脚本的执行过程，程序根据用户的需要返回信息。对于一些大型系统，访问量往往很大，频繁的动态操作与数据库操作大大加重了服务器的负担。

在这种情况下，通常使用缓存技术或者静态化操作。也就是将 PHP 脚本的页面结果放到缓存中，或者放到静态 HTML 文件中。这样，访问者在访问页面时，不需要重新执行动态 PHP 代码即可获得结果，大大缓解了服务器的负担。本章将介绍缓存机制与静态化的原理，以及使用 PHP 实现静态化的方法。

5.1　为什么要静态化

在实际应用中，静态化所得到页面结果与通过直接访问 PHP 页面所得到的页面结果一般完全相同。也就是说，查看静态化后的页面并不会对页面中的内容、布局造成影响。静态化的主要目的就是提高页面访问的性能。除此之外，将页面静态化可以方便的储存信息，在本地查看页面的时候不需要架设服务器、数据库等。

那么，静态化对于提高性能有什么好处呢？这里首先介绍一个用于测试性能的工具——Apache Benchmarking Tool。该工具是 Apache 服务器提供的一个模拟多用户访问的命令行测量工具，通过对访问地址页面的访问计算出用户的响应时间。Apache Benchmarking Tool 的可执行文件在 Apache 的安装目录下的 bin 文件夹，文件名为 ab.exe，语法格式如下所示。

```
ab [options] [http[s]://]hostname[:port]/path
```

其中，option 指的是参数。常用的参数有以下两种。

- -n：执行访问的次数。
- -c：同时并发用户的数目。

对于更多的参数信息，可以通过直接执行 ab.exe 来获得。

hostname 是主机的地址，port 是端口号，path 是访问路径。

以下代码通过一个循环结构来比较 PHP 脚本与 HTML 静态页面的性能。

```
<?php
for($i=0;$i<100;$i++)
{
      echo "$i";
}
?>
```

静态 HTML 代码如下所示。

```
0123456789101112131415161718192021222324252627282930313233343536373839404142434445464748495051525354555657585960616263646566676869707172737475767778798081828384858687888990919293949596979899
```

可以看出，这两个文件在浏览器上的输出结果是相同的。然后在命令行上执行 ab 命令分别测试这

两个页面的性能。测试参数采用对页面访问 10000 次，并且有 10 个并发用户同时访问。

对于 PHP 脚本文件的测试结果如下所示。

```
C:\apache\bin>ab -n 10000 -c 10 http://localhost/test/test.p
hp
This is ApacheBench, Version 2.0.40-dev <$Revision: 1.146 $> apache-2.0
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Copyright 1997-2005 The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Finished 10000 requests


Server Software:        Apache/2.2.0
Server Hostname:        localhost
Server Port:            80

Document Path:          /test/test.php
Document Length:        501 bytes

Concurrency Level:      10
Time taken for tests:   285.400386 seconds
Complete requests:      10000
Failed requests:        9997
   (Connect: 0, Length: 9997, Exceptions: 0)
Write errors:           0
Total transferred:      2634055 bytes
HTML transferred:       434055 bytes
Requests per second:    35.04 [#/sec] (mean)
Time per request:       285.400 [ms] (mean)
Time per request:       28.540 [ms] (mean, across all concurrent requests)
Transfer rate:          9.01 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    2  42.4      0    4216
Processing:    60  277 122.5    270    4075
Waiting:       60  270 108.1    260    3595
Total:         70  280 141.3    270    8261

Percentage of the requests served within a certain time (ms)
  50%    270
```

```
  66%      280
  75%      280
  80%      290
  90%      300
  95%      310
  98%      330
  99%      350
 100%     8261 (longest request)
```

对于静态 HTML 文件的测试结果如下所示。

```
C:\apache\bin>ab -n 10000 -c 10 http://localhost/test/test.h
tm
This is ApacheBench, Version 2.0.40-dev <$Revision: 1.146 $> apache-2.0
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Copyright 1997-2005 The Apache Software Foundation, http://www.apache.org/

Benchmarking localhost (be patient)
Completed 1000 requests
Completed 2000 requests
Completed 3000 requests
Completed 4000 requests
Completed 5000 requests
Completed 6000 requests
Completed 7000 requests
Completed 8000 requests
Completed 9000 requests
Finished 10000 requests


Server Software:        Apache/2.2.0
Server Hostname:        localhost
Server Port:            80

Document Path:          /test/test.htm
Document Length:        190 bytes

Concurrency Level:      10
Time taken for tests:   48.329495 seconds
Complete requests:      10000
Failed requests:        0
Write errors:           0
Total transferred:      5010000 bytes
HTML transferred:       1900000 bytes
Requests per second:    206.91 [#/sec] (mean)
Time per request:       48.329 [ms] (mean)
Time per request:       4.833 [ms] (mean, across all concurrent requests)
Transfer rate:          101.22 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    3   5.0      0      60
```

```
Processing:    10   43   8.9     40     160
Waiting:        0   24  14.9     20     140
Total:         20   47   9.7     50     160

Percentage of the requests served within a certain time (ms)
  50%     50
  66%     50
  75%     50
  80%     50
  90%     50
  95%     60
  98%     60
  99%     90
 100%    160 (longest request)
```

以上两个运行结果的比较如表 5-1 所示。

表 5-1　PHP脚本与HTML静态页面的性能比较实例

指标	PHP脚本	HTML静态页面
文件大小	190字节	190字节
平均每秒执行的访问次数	88.92次	206.91次
平均每次访问的响应时间	11.246毫秒	4.833毫秒
最长等待时间	570毫秒	160毫秒

从表 5-1 可以看出，仅对于一个简单的循环结构来说，使用静态 HTML 页面的执行效率与相应的 PHP 脚本相比就提高了 57%。

5.2　大型应用的缓存机制与静态化

一般来说，一个小型网站或者一个小型应用仅有几页组成，可能仅有少量的动态操作，访问的用户也不是很多。对于这样的网站或者应用来说，使用缓存机制或者静态化的方法作用并不明显。因为用户的访问并不会给服务器造成很大的负担，并发的用户访问数也不会很多。而对于用户来说，获得一个完整页面用了 10 毫秒和 100 毫秒区别并不是很大。因此，缓存机制与静态化应用在小型系统中很少见。

而对于大型系统来说，访问量很大，动态的脚本文件和静态 HTML 的性能差别就很明显了。本节将简要介绍如何在大型应用上使用缓存机制和静态化技术。

5.2.1　缓存机制

缓存是一种“以空间换时间”的策略，是一种提高性能的常用方法。缓存机制通常用于缓解大访问量下的数据库和磁盘负担，有效的应用缓存机制可以很大的减少对数据库和磁盘的操作次数。在 PHP 中，目前已经有多种缓存的解决方案，例如 PEAR Cache、Zend Cache、Alternative PHP Cache、和 Afterburner Cache 等。

对于 PHP 来说，上述缓存模块会在代码被第一次访问的时候将 PHP 的中间代码保存到服务器的内存中。中间代码指的是编译、解析过的 PHP 代码。当第一次访问后，接下来的访问都是通过代码直接从服务器中的内存获取数据的过程。由于数据库和磁盘操作被大大的减少了，访问速度大大提高。

缓存模块的另一个特性是能够监视 PHP 代码和数据的变化。这个特性能够有效的保证在 PHP 脚本

代码被修改或者数据库中的数据发生变化以后，访问者可以及时看到更新，而不是获取到内存中的旧的页面。缓存机制在大型系统中非常常见，主要用于访问用户过多的情况。

5.2.2　静态化机制

静态化即将动态的脚本文件生成静态 HTML 页面。由于静态 HTML 页面没有对数据库、磁盘的操作，甚至没有任何逻辑计算，访问速度极快。对于一些大型应用，将一些变化不大的页面以静态 HTML 的形式存储可以有效的提高页面的访问性能。

例如，将新闻网站的新闻页面制作成静态 HTML 页面。由于新闻页面中的新闻内容一旦生成就不再改变，交互性不强，很适合以静态 HTML 的形式供用户访问。

5.3　PHP 如何实现静态化

上一节介绍了静态化的机制，本节将以几个在设计静态化网站时常用到的一些功能为例说明 PHP 如何对页面实现静态化。

5.3.1　根据模版生成静态页面

模版是没有内容的 HTML 页面，也就是要生成的静态页面的版式。与前面介绍过的 PEAR HTML_Template_IT 类库类似，根据模版生成静态页面的方法是根据对模版文件中的 HTML 代码读取，然后将关键字进行内容替换并写入一个新的 HTML 静态页面，由此实现根据模版生成静态页面。

以下代码实现了一个简单的根据模版生成静态页面的实例。该实例读取模版文件 template.htm，然后将其中的关键字用程序中变量的值替换，并生成静态页面文件 new.htm。其中的关键字用一对百分号“%%”括起来表示。

模版文件 template.htm 如下所示。

```
<html>
    <head>
        <title>%title%</title>
    </head>
    <body>
        <H1>%title%</H1>
        <hr>
        <pre>%body%</pre>
    </body>
</html>
```

用于从 template.htm 模版文件生成新 HTML 页面的 PHP 代码如下所示。

```
<?php
//Replace 函数用于将从模版文件中读取的内容中的关键字替换成变量中的内容
function Replace($row)
{
    //定义用来替换的变量
    $title = "文章标题";
    $body   = "这里是文章主体";
```

```
        //替换参数中的关键字
        $row     = str_replace("%title%", $title, $row);
        $row     = str_replace("%body%", $body, $row);
        //返回替换后的结果
        return $row;
}
//模版文件指针
$f_tem = fopen("template.htm","r");
//生成的文件指针
$f_new = fopen("new.htm","w");
//循环读取模版文件，每次读取一行
while(!feof($f_tem))
{
      $row = fgets($f_tem);
      $row = Replace($row);                    //替换读入内容中的关键字
      fwrite($f_new, $row);                 //将替换后的内容写入生成的 HTML 文件
}
//关闭文件指针
fclose($f_new);
fclose($f_tem);
?>
```

生成页面从浏览器中得到的结果如图 5-1 所示。

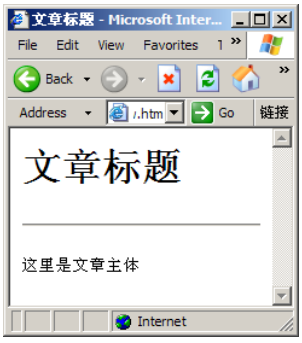

图 5-1　new.htm

查看其源代码，如下所示。

```
<html>
      <head>
            <title>文章标题</title>
      </head>
      <body>
            <H1>文章标题</H1>
            <hr>
            <pre>这里是文章主体</pre>
      </body>
</html>
```

可以看出，模版中的所有用“%%”括起来的关键字都被替换成变量的值了。

5.3.2　数据库与静态页的关系

静态页的应用通常与数据库的应用联系起来。一般来说，在实际应用中，静态 HTML 页面的生成是在系统向数据库中插入数据的时候。例如，一个新闻系统在生成新的静态新闻页面时是在添加一篇新新闻的时候。为了防止数据被破坏，以及方便后期的数据维护，在生成静态 HTML 页面的同时往往向数据库中同步插入一条记录。

以下代码是一个插入记录并生成静态页的例子。在这个例子中，从一个表单内输入要插入的内容，然后 PHP 程序会将用户的输入保存到数据库中，并同时生成一个静态 HTML 页面。这里使用的模版与上面的例子相同。

存放供用户输入数据的表单文件的代码如下。

```
<html>
<head>
<title>添加一条新记录</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
</head>

<body>
 <p><h1>添加一条新记录</h1></p>
 <form name="form1" method="post" action="insert.php">
    <table width="500" border="0" cellspacing="0" cellpadding="0">
      <tr>
        <td>标题</td>
        <td><input name="title" type="text" id="title"></td>
      </tr>
      <tr>
        <td>内容</td>
        <td><textarea name="body" cols="35" rows="5" id="body"></textarea></td>
      </tr>
      <tr>
        <td colspan="2"><input type="submit" name="Submit" value="Submit">
        <input type="reset" name="Submit2" value="Reset"></td>
      </tr>
    </table>
</form>
</body>
</html>
```

用于处理用户输入的 PHP 代码如下所示。

```
<?php
//Replace 函数用于将从模版文件中读取的内容中的关键字替换成变量中的内容
function Replace($row, $title='', $body='')
{
      //替换参数中的关键字
      $row     = str_replace("%title%", $title, $row);
      $row     = str_replace("%body%", $body, $row);
      //返回替换后的结果
      return $row;
}
//主程序开始
```

```
@mysql_connect("localhost", "root")                    //选择数据库之前需要先连接数据库服务器
or die("数据库服务器连接失败");
@mysql_select_db("mydb")                               //选择数据库 mydb
or die("数据库不存在或不可用");
//将表单中的数据通过$_POST 方式获取然后存储在相应的变量中
$title = $_POST['title'];
$body = $_POST['body'];
//生成文件名
$filename = 'S'.date("YmdHis").'.htm';
//执行 SQL 语句
$query = mysql_query("insert into news values('$title', '$body', '$filename')");
//根据 SQL 执行语句返回的 bool 型变量判断是否插入成功
if($query)
{
      $f_tem = fopen("template.htm","r");              //模版文件指针
      $f_new = fopen($filename,"w");                   //生成的文件指针
      while(!feof($f_tem))                             //循环读取模版文件，每次读取一行
      {
          $row = fgets($f_tem);
          $row = Replace($row, $title, $body);         //替换读入内容中的关键字
          fwrite($f_new, $row);                        //将替换后的内容写入生成的 HTML 文件
      }
      //关闭文件指针
      fclose($f_new);
      fclose($f_tem);
      echo "数据插入成功";                             //提示
}
else
     echo "数据插入失败";
mysql_close();                                         //关闭与数据库服务器的连接
?>
```

在数据库中创建表 news 如下所示。

```
mysql> create table news(title char(50), body text, filename char(20));
```

在浏览器上访问存放用户表单的页面并输入一些内容，如图 5-2 所示。

图 5-2　index.htm

提交后，在数据库中可以看到，这条记录也被插入到数据库中了，如下所示。

```
mysql> select * from news;
+---------------+----------------------+--------------------+
| title         | body                 | filename           |
+---------------+----------------------+--------------------+
| This is a test | Welcome to PHP world!! | S20060904150821.htm |
+---------------+----------------------+--------------------+
1 row in set (0.00 sec)
```

在浏览器上访问生成的 HTML 文件如图 5-3 所示。

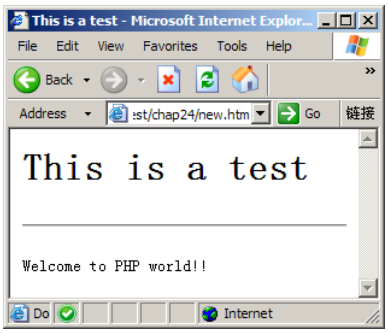

图 5-3　new.htm

这种同时向数据库中插入数据的生成静态 HTML 页面的方法在实际应用中非常常见。

5.3.3　静态页内容的修改

前面在介绍数据库与静态页面关系的时候介绍了在插入一条新数据时应该如何处理。在更新一条数据的时候方法与插入一条新数据时类似。其方法是在更新数据库的同时，重新生成一个新的静态 HTML 页面。需要注意的是在对内容进行修改的时候，并不是直接修改原有的静态页，而是根据用户的修改重新创建。

下面来考虑如何对上面例子中的数据进行更新操作。需要注意的是代码中对 filename 的操作，由于仍然需要使用原文件名作为文件的存储位置，所以在更新时需要获取数据库中的 filename，而不是像前面例子那样根据时间来确认文件名。

一个简单的方法是在生成用户表单时将 filename 作为一个 hidden 域放在表单中。这样，在处理用户的表单时，PHP 程序就可以很容易的获取原来的文件名了，实现的代码如下所示。

```
<?php
@mysql_connect("localhost", "root")                    //选择数据库之前需要先连接数据库服务器
or die("数据库服务器连接失败");
@mysql_select_db("mydb")                               //选择数据库 mydb
or die("数据库不存在或不可用");
$title = $_GET['title'];                               //读取地址栏上的参数 title
$query = mysql_query("select * from news where title = $title");
$row = mysql_fetch_array($query);                      //读取数据库中的数据并放置在数组$row 中
?>
<html>
<head>
<title>修改一条新记录</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
```

```
</head>

<body>
 <p><h1>修改一条新记录</h1></p>
 <form name="form1" method="post" action="">
   <table width="500" border="0" cellspacing="0" cellpadding="0">
     <tr>
       <td>标题</td>
       <td><input name="title" type="text" id="title" value="<?php echo $row['title']?>"></td>
     </tr>
     <tr>
       <td>内容</td>
       <td><textarea name="body" cols="35" rows="5" id="body"
                value="<?php echo $row['body']?>"></textarea>
       <input name="filename" type="hidden" value="<?php echo $row['filename']?>"></td>
     </tr>
     <tr>
       <td colspan="2"><input type="submit" name="Submit" value="Submit">
       <input type="reset" name="Submit2" value="Reset"></td>
     </tr>
   </table>
</form>
</body>
</html>
```

这样做，不仅没有影响页面美观，也很容易得达到了传递参数的目的。

5.3.4　模版的替换

对于一个动态页面实现的网站或者应用系统，如果需要修改页面的样式，只需要将页面中的 HTML 代码进行相应的修改即可。但是对于使用静态 HTML 页面进行内容显示的网站或者应用系统来说，由于每增加一条新记录，就会生成一个新的 HTML 页面。如果要对所有的 HTML 页面逐一进行修改，工作量非常大，而且容易出错。因此，这种方法是行不通的。

一般来说，如果需要修改静态 HTML 页面的模版，通常的做法是将所有的已经生成的 HTML 页面删除，然后重新创建新的 HTML 页面。以下代码实现了对 news 表中的所有记录根据 template.htm 模版生成静态 HTML 页面的功能。

```
<?php
//Replace 函数用于将从模版文件中读取的内容中的关键字替换成变量中的内容
function Replace($row, $title='', $body='')
{
      //替换参数中的关键字
      $row = str_replace("%title%", $title, $row);
      $row = str_replace("%body%", $body, $row);
      //返回替换后的结果
      return $row;
}
@mysql_connect("localhost", "root")                    //选择数据库之前需要先连接数据库服务器
or die("数据库服务器连接失败");
@mysql_select_db("mydb")                               //选择数据库 mydb
```

```
or die("数据库不存在或不可用");
$query = @mysql_query("select * from news")  //执行 SQL 语句
or die("SQL 语句执行失败");
//通过循环的方式处理从第 0 行到最大的一行的所有记录
while($record = mysql_fetch_array($query))
{
    //模版文件指针
    $f_tem = fopen("template.htm","r");
    //生成的文件指针
    $f_new = fopen($record['filename'],"w");
    //循环读取模版文件，每次读取一行
    while(!feof($f_tem))
    {
        $row = fgets($f_tem);
        $row = Replace($row, $record['title'], $record['body']);      //替换读入内容中的关键字
        fwrite($f_new, $row);                                         //将替换后的内容写入生成的 HTML 文件
    }
    //关闭文件指针
    fclose($f_new);
    fclose($f_tem);
}
mysql_close();                                                        //关闭与数据库服务器的连接
?>
```

上面的代码逐一读取数据库表中的记录，对于每一条记录分别生成一个静态 HTML 页面。由于使用的文件名与原文件名相同，并且使用“w”方式打开文件，原有文件会被自动覆盖。

5.3.5　静态页上的动态操作

有些时候，在创建的静态 HTML 页上面也需要进行一些动态操作。例如，新闻系统中的每篇新闻要统计点击率。对于这种情况，一个简单的解决办法是将新创建的页面的扩展名由“.htm”改成“.php”，并在模版文件中写入相应的 PHP 代码。这样，一些需要的动态操作就可以在该文件被访问时执行了。

这种方法易于实现，并且有一定灵活性。但是这种方法从某种程度上讲有些失去了“静态”的意义，并且创建的文件不能在本地访问。因此，这里介绍另一种方法，即通过一个宽和高都为 0 像素的图像控件来隐藏的调用以下代码实现了在一个静态 HTML 页面上调用一个 PHP 文件来实现页面计数器的功能。用于存放计数器的数据表如下所示。

```
mysql> select * from counter;
+-------+--------+
| count | fileid |
+-------+--------+
|     0 | S001   |
+-------+--------+
1 row in set (0.01 sec)
```

静态 HTML 页面如下所示。

```
<html>
    <head>
        <title>News</title>
    </head>
```

```
    <body>
        <H1>News</H1>
        <hr>
        <pre>This is a test!!</pre>
        <img width='0' height='0' src='counter.php?fileid=S001'>
    </body>
</html>
```

用于更新计数器的 PHP 代码如下所示。

```
<?php
mysql_connect("localhost", "root");                    //选择数据库之前需要先连接数据库服务器
mysql_select_db("mydb");                               //选择数据库 mydb
$fileid = $_GET['fileid'];
//执行 SQL 语句
$query = mysql_query("update counter set count=1 where fileid='$fileid'");
?>
```

在浏览器中访问静态 HTML 页面如图 5-4 所示。

图 5-4　静态 HTML 页面

从图 26-4 可以看出，<img>图像控件的加入并没有影响页面的整体构架。查看数据库中的 counter 表如下所示。

```
mysql> select * from counter;
+-------+--------+
| count | fileid |
+-------+--------+
|     1 | S001   |
+-------+--------+
1 row in set (0.01 sec)
```

可以看出，虽然执行的是静态页面，由于隐藏的调用了 PHP 代码，动态的更新数据库的功能被实现了。

5.3.6　静态页面文件的目录

对于动态页面来说，数据库中记录的列表可以很容易的通过读取数据库来完成。对于生成静态页面的系统来说，记录的列表也可以通过读取数据库来完成，只是将链接直接指向静态 HTML 文件即可。

但是，通常对于使用静态页面的系统来说，往往将记录的目录也生成静态 HTML 文件供访问者浏览。对于 HTML 文件中的记录列表，可以通过读取数据库中的记录然后逐条写入文件中来完成。需要

注意的是因为每增加或者减少一条记录都会对记录列表受到影响，因此，每次对记录进行添加和删除时都需要执行此操作。以下代码实现了一个简单的生成记录列表的功能。

模版文件如下所示。

```
<html>
    <head>
        <title>目录</title>
    </head>
    <body>
        <H1>目录</H1>
        <hr>
        <pre>%menu%</pre>
    </body>
</html>
```

PHP 代码如下所示。

```
<?php
@mysql_connect("localhost", "root")                    //选择数据库之前需要先连接数据库服务器
or die("数据库服务器连接失败");
@mysql_select_db("mydb")                               //选择数据库 mydb
or die("数据库不存在或不可用");
$query = @mysql_query("select * from news")            //执行 SQL 语句
or die("SQL 语句执行失败");
$menu = '';                                            //初始化$menu 变量

while($record = mysql_fetch_array($query))             //通过循环的方式处理从第 0 行到最大的一行的所有记录
{
    //读取记录中的 title 和 filename
    $title = $record['title'];
    $filename = $record['filename'];
    //生成链接的 HTML 代码
    $link = "<a href='$filename'>$title</a><br>";
    //将链接放到$menu 中
    $menu .= $link;
}

$f_tem = fopen("template.htm","r");                    //模版文件指针
$f_new = fopen("menu.htm","w");                        //生成的文件指针

while(!feof($f_tem))                                   //循环读取模版文件，每次读取一行
{
    $row = fgets($f_tem);
    $row = str_replace("%menu%", $menu, $row);         //替换读入内容中的关键字
    fwrite($f_new, $row);                              //将替换后的内容写入生成的 HTML 文件
}
//关闭文件指针
fclose($f_new);
fclose($f_tem);
mysql_close();                                         //关闭与数据库服务器的连接
?>
```

代码运行后，从浏览器中访问生成的 HTML 文件，如图 5-5 所示。

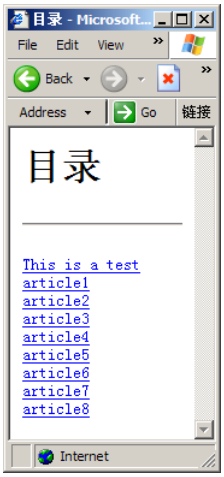

图 5-5　目录页面

在第 11 章介绍过记录列表的分页的设计。对于静态的记录列表页面，分页的设计可以通过创建多个静态 HTML 页面文件来完成。以下代码对上面的例子进行了修改，实现了每页显示 5 条记录的功能。

模版文件如下所示。

```
<html>
    <head>
        <title>目录</title>
    </head>
    <body>
        <H1>目录</H1>
        <hr>
        <pre>%menu%</pre>
        <a href='%prev%'>上一页</a> | <a href='%next%'>下一页</a>
    </body>
</html>
```

PHP 代码如下所示。

```
<?php
@mysql_connect("localhost", "root")                    //选择数据库之前需要先连接数据库服务器
or die("数据库服务器连接失败");
@mysql_select_db("mydb")                               //选择数据库 mydb
or die("数据库不存在或不可用");
$query = @mysql_query("select * from news")            //执行 SQL 语句
or die("SQL 语句执行失败");
$pagesize = 5;                                         //每页显示记录数
$i = 0;                                                //初始化计数器
$p = 0;                                                //初始化页码
$menu = '';                                            //初始化$menu 变量
$sum = mysql_numrows($query);                          //计算总记录数

while($record = mysql_fetch_array($query))             //通过循环的方式处理从第 0 行到最大的一行的所有记录
{
    //读取记录中的 title 和 filename
    $title = $record['title'];
    $filename = $record['filename'];
    //生成链接的 HTML 代码
```

```
        $link = "<a href='$filename'>$title</a><br>";
        //将链接放到$menu 中
        $menu .= $link;
        //计数
        $i++;
        //当计数器$i 能够整除$pagesize 时或者所有的记录都被读取时输出文件
        if($i%$pagesize == 0 OR $i == $sum)
        {
            $p++;                                                    //增加页码
            $f_tem = fopen("template.htm","r");                      //模版文件指针
            $f_new = fopen("menu$p.htm","w");                        //生成的文件指针
            if($p>1)                                                 //计算上一页页码
            {
                $prev = $p - 1;
            }
            else
            {
                $prev = $p;
            }
            if($i<$sum)                                              //计算下一页页码
            {
                $next = $p + 1;
            }
            else
            {
                $next = $p;
            }

            while(!feof($f_tem))                                     //循环读取模版文件，每次读取一行
            {
                $row = fgets($f_tem);
                $row = str_replace("%menu%", $menu, $row);           //替换读入内容中的关键字
                $row = str_replace("%prev%", "menu$prev.htm", $row); //替换读入内容中的关键字
                $row = str_replace("%next%", "menu$next.htm", $row); //替换读入内容中的关键字
                fwrite($f_new, $row);                                //将替换后的内容写入生成的 HTML 文件
            }
            fclose($f_new);                                          //关闭文件指针
            fclose($f_tem);
            $menu = '';                                              //初始化$menu
        }
}
mysql_close();                                                       //关闭与数据库服务器的连接
?>
```

代码运行后，从浏览器中访问生成的 HTML 文件，如图 5-6 所示。

图 5-6　支持分页的目录页面

5.4　小结

本章介绍了如何使用 PHP 对网站页面实现静态化。在大型系统中，静态化的应用非常广泛。因此，掌握如何实现静态化的方法在建设大型网站时非常有用。本章中介绍的一些实际中常用的静态化方法一些普遍使用的方法。读者可以对这些方法仔细研究，在实际项目中根据实际情况设计出自己的静态化方法。