第 3 章　ADOdb 的应用

在本书第二篇中，介绍了 PHP 与 MySQL 等几种常用数据库的结合应用。PHP 提供了多种数据库访问组件来访问不同类型的数据库。PHP 的这种机制为 PHP 程序员制造了很多麻烦，为了访问不同中的数据库，不得不记住多种多样的数据库存取函数。而本章将要介绍的 ADOdb 类库有效地解决了这个问题。ADOdb 类库通过对不同数据库的差异封装，有效的使 PHP 应用可以轻松的在不同的数据库之间进行切换。

3.1　ADOdb 介绍

ADOdb 是“Active Data Object DataBase”的缩写，提供了与 Microsoft 的 ADO 类似的功能。其主要作用就是将多样化的 PHP 数据库操作函数统一起来，以提高 PHP 数据库操作能力的可移植性。目前，ADOdb 已经可以支持 MySQL、PostgreSQL、Interbase、Firebird、Informix、Oracle、MS SQL、Foxpro，Access、ADO、Sybase、FrontBase、DB2、SAP DB、SQLite 和 ODBC 等多种数据库。

3.1.1　ADOdb 的优势

ADOdb 的优势主要表现在以下几点。

- 可以很方便的连接多种数据库。如果在一个应用中需要连接并操作多种数据库，使用 ADOdb 类库可以很容易的实现。
- 可以很容易的实现数据库移植。在实际应用中有时候可能需要从更换数据库，如果使用传统的数据库连接方式，这种更换将非常麻烦。例如，连接 MySQL 数据库使用 mysql_connect 函数来实现，而连接 Microsoft SQL Server 则使用 mssql_connect 函数来实现。如果在这里使用 ADOdb 类库，则只需要进行简单的修改即可。
- 可通用的数据类型。使用 ADOdb 类库，可以使用通用的数据类型，而不用根据数据库的数据类型本身。
- 完善的侦错机制。在使用 ADOdb 类库时，可以方便的在 PHP 代码中设置是否显示侦错信息。侦错机制可以输出一切需要的信息，方便程序的调试。

3.1.2　ADOdb 的适用场合

根据 ADOdb 的优势，ADOdb 主要适用于以下场合。

- 可能需要数据库移植的应用系统。如果使用 PHP 开发的应用可能会根据需要更换数据库的类型，则使用 ADOdb 开发会大大减少因为更换数据库带来的麻烦。
- 需要支持多种数据库的通用系统。如果使用 PHP 开发的应用需要支持多种数据库以供用户使用，则使用 ADOdb 开发会大大提高开发效率，缩短开发周期。

- 各种 CMS 系统及 Web 站点。对于一般的 CMS 系统或 Web 站点，也可以使用 ADOdb 开发。

3.2　ADOdb 类库的安装与配置

ADOdb 类库是一个外部类库，并不包含在 PHP 核心中。如果需要使用 ADOdb 类库，则需要先下载方能使用。

3.2.1　ADOdb 的下载与安装

ADOdb 的下载可以通过访问 http://adodb.sourceforge.net/来得到。ADOdb 是一个源代码开放的类库，并且根据 ADOdb 的使用协议，开发者可以在商业产品中使用 ADOdb 类库。

ADOdb 类库的安装只需要将下载的文件直接解压到某一文件夹即可。为了方便起见，一般可以将下载的文件解压到 PEAR 目录或者网站的根目录下以方便使用。

3.2.2　ADOdb 的第一个测试程序

在编写 ADOdb 的第一个测试程序之前，在本地 MySQL 数据库服务器的 mydb 数据库下建有 mytable 表，如下所示。

```
mysql> select * from mytable;
+-----------+---------+-----+------------+---------+---------+
| serial_no | name    | age | birthday   | salary  | bonus   |
+-----------+---------+-----+------------+---------+---------+
|    100001 | Simon   |  24 | 1982-11-06 | 5000.00 | 1000.00 |
|    100002 | Elaine  |  24 | 1982-01-01 | 5000.00 | 1000.00 |
|    100003 | Susan   |  31 | 1975-01-01 | 9000.00 | 2000.00 |
+-----------+---------+-----+------------+---------+---------+
3 rows in set (0.00 sec)
```

以下代码实现了使用 ADOdb 连接 MySQL 数据库。其中，下载的 ADOdb 类库的代码放在文件所在目录的 adodb 文件夹下。

```
<?php
include("adodb/adodb.inc.php");                    //包含 adodb 类库文件
$conn = NewADOConnection('mysql');                 //创建 adodb 对象，声明数据库类型为 MySQL
$conn->debug = true;                               //显示侦错讯息
$conn->Connect("localhost", "root", "", "mydb");   //连接数据库，其中 mydb 为数据库名
$rs = $conn->Execute("SELECT * FROM mytable");     //执行 SQL 语句
print_r($rs->GetRows());                           //输出 SQL 语句的执行结果
?>
```

运行结果如下所示。

```
<hr />
(mysql): SELECT * FROM mytable  
<hr />
Array
(
    [0] => Array
```

```
        (
            [0] => 100001
            [serial_no] => 100001
            [1] => Simon
            [name] => Simon
            [2] => 24
            [age] => 24
            [3] => 1982-11-06
            [birthday] => 1982-11-06
            [4] => 5000.00
            [salary] => 5000.00
            [5] => 1000.00
            [bonus] => 1000.00
        )

    [1] => Array
        (
            [0] => 100002
            [serial_no] => 100002
            [1] => Elaine
            [name] => Elaine
            [2] => 24
            [age] => 24
            [3] => 1982-01-01
            [birthday] => 1982-01-01
            [4] => 5000.00
            [salary] => 5000.00
            [5] => 1000.00
            [bonus] => 1000.00
        )

    [2] => Array
        (
            [0] => 100003
            [serial_no] => 100003
            [1] => Susan
            [name] => Susan
            [2] => 31
            [age] => 31
            [3] => 1975-01-01
            [birthday] => 1975-01-01
            [4] => 9000.00
            [salary] => 9000.00
            [5] => 2000.00
            [bonus] => 2000.00
        )

)
```

需要注意的是代码输出结果的第一行并不是在上述 PHP 代码通过输出语句输出的，而是因为开启了 ADOdb 的侦错机制，由 ADOdb 输出的侦错信息。由此可见，ADOdb 的侦错机制不光在程序出错时

会输出相应的信息，在程序正常运行时也会尽可能多的输出调试信息。

对于 ADOdb 的具体使用将在下一节中详细介绍。

3.3　ADOdb 的常用数据库操作

上一节简要介绍了 ADOdb 的安装，并通过一个简单的例子实现了对 MySQL 数据库中数据的读取。本节将结合 MySQL 数据库详细介绍 ADOdb 的常用操作。

3.3.1　连接数据库

由前面的例子可以看到，连接数据库的方法是 NewADOConnection 函数，该函数返回一个 Connection 对象，其语法格式如下所示。

```
object NewADOConnection($dsn);
```

这里，$dsn 是用于数据库连接的引擎，该引擎可以是一个数据库系统的名称。如以下代码所示，用于声明连接一个 MySQL 服务器。

```
<?php
include("adodb/adodb.inc.php");                    //包含 adodb 类库文件
$conn = NewADOConnection('mysql');                 //创建数据库连接对象
?>
```

使用 NewADOConnection 函数声明数据库连接后，需要使用 Connect 函数对其进行连接，其语法格式如下所示。

```
Connect($hostname,[$username],[$password],[$database])
```

其中，hostname 是数据库服务器所在的主机地址，username 是用于连接数据库的用户名，password 是相应的密码。以下代码用于连接一个 MySQL 服务器。

```
<?php
include("adodb/adodb.inc.php");                    //包含 adodb 类库文件
$conn = NewADOConnection('mysql');                 //创建数据库连接对象
$conn -> Connect('localhost', 'root', 'pass', 'mydb');    //连接数据库
?>
```

上面的例子用于连接一个存放在 localhost 上的 MySQL 数据库服务器，使用用户名 root、密码 pass 登录，连接到 mydb 服务器。

对于前面的 NewADOConnection 函数，其参数$dsn 也可以是一个用于表示数据源的字符串（DSN）。此时，NewADOConnection 函数将隐性调用 Connect 函数来完成连接。数据源的语法格式如下所示。

```
driver://username:password@hostname/database
```

其中，driver 是数据库系统的名称，username 是用于连接数据库的用户名，password 是相应的密码，hostname 是数据库服务器所在的主机地址。以下代码用于连接一个 MySQL 服务器。

```
<?php
include("adodb/adodb.inc.php");                                    //包含 adodb 类库文件
$conn = NewADOConnection('mysql://root:pass@localhost/mydb');      //连接数据库
?>
```

上面传入 NewADOConnection 函数的 DSN 参数用于表示一个存放在 localhost 上的 MySQL 数据库服务器，使用用户名 root、密码 pass 登录，连接到 mydb 服务器。

函数 NewADOConnection 还有另一个函数名 ADONewConnection，其用法与 NewADOConnection

相同。

ADOdb 支持多种数据库的连接，下面将简要介绍几种常见的数据库连接方法。

1．MySQL

对于 MySQL 的连接，就像前面介绍过的那样，有两种不同的方法。

其一是使用标准的连接字符串来连接，如以下代码所示。

```
<?php
include("adodb/adodb.inc.php");                    //包含 adodb 类库文件
$conn = NewADOConnection('mysql');
$conn->Connect('localhost', 'root', 'pass', 'mydb');    //连接 MySQL 数据库
?>
```

这里，localhost、root、pass 和 mydb 分别表示要连接的数据库服务其所在地址、用户名、密码和数据库名称。

其二是采用数据源名称（DSN）的方式进行连接，如以下代码所示。

```
<?php
include("adodb/adodb.inc.php");                                    //包含 adodb 类库文件
$conn = NewADOConnection('mysql://root:pass@localhost/mydb');      //连接 MySQL 数据库
?>
```

实现了与上面相同的功能。

2．PostgreSQL

对于 PostgreSQL 的连接，就像前面介绍过的那样，有三种不同的方法。

其一是使用标准的连接字符串来连接，如以下代码所示。

```
<?php
include("adodb/adodb.inc.php");                    //包含 adodb 类库文件
$conn = NewADOConnection('postgres');
$conn->Connect('localhost', 'root', 'pass', 'mydb');    //连接 PostgreSQL 数据库
?>
```

这里，localhost、root、pass 和 mydb 分别表示要连接的数据库服务其所在地址、用户名、密码和数据库名称。

其二是采用一个 PostgreSQL 特定的字符串的方式进行连接，如以下代码所示。

```
<?php
include("adodb/adodb.inc.php");                          //包含 adodb 类库文件
$conn = NewADOConnection(' postgres ');
$conn->Connect('host=localhost port=5432 dbname=mydb');  //连接 PostgreSQL 数据库
?>
```

这里，localhost、5432 和 mydb 分别表示要连接的数据库服务其所在地址、端口号和数据库名称。

其三是采用 DSN 的方式进行连接，如以下代码所示。

```
<?php
include("adodb/adodb.inc.php");                              //包含 adodb 类库文件
$conn = NewADOConnection('postgres://root:pass@localhost/mydb');   //连接 PostgreSQL 数据库
?>
```

这里，localhost、root、pass 和 mydb 分别表示要连接的数据库服务其所在地址、用户名、密码和数据库名称。

3．Microsoft Access：

对于 Access 数据库，通常使用 ODBC 的连接方法。以下代码演示了 ADOdb 连接 Access 的方法。

```
<?php
include("adodb/adodb.inc.php");                          //包含 adodb 类库文件
$conn = NewADOConnection('access');                      //连接 Access 数据库
$conn->Connect("Driver={Microsoft Access Driver (*.mdb)};Dbq=d:\\mydb.mdb;Uid=Admin;Pwd=;");
?>
```

其中，d:\\mydb.mdb 表示数据库所在文件名。

4．Microsoft SQL Server

对于 SQL Server，ADOdb 提供两种连接方法。

其一是使用 ODBC 的连接方法，如以下代码所示。

```
<?php
include("adodb/adodb.inc.php");                          //包含 adodb 类库文件
$conn=NewADOConnection('odbc_mssql');                    //连接 SQL Server 数据库
$conn->Connect("Driver={SQL Server};Server=localhost;Database=mydb;",'username','password');
?>
```

这里，localhost、username、password 和 mydb 分别表示要连接的数据库服务其所在地址、用户名、密码和数据库名称。

其二是使用 MSSQL 扩展，如以下代码所示。

```
<?php
include("adodb/adodb.inc.php");                          //包含 adodb 类库文件
$conn=NewADOConnection('mssql');                         //连接 SQL Server 数据库
$conn->Connect("localhost', 'username', 'password', 'mydb');
?>
```

这里，localhost、username、password 和 mydb 分别表示要连接的数据库服务其所在地址、用户名、密码和数据库名称。

5．IBM DB2

对于 DB2 数据库，通常使用 ODBC 的连接方法。以下代码演示了 ADOdb 连接 DB2 的方法。

```
<?php
include("adodb/adodb.inc.php");                          //包含 adodb 类库文件
$conn=NewADOConnection('db2');                           //连接 DB2 数据库
$conn->Connect("driver={IBM db2 odbc DRIVER};database=mydb;hostname=localhost;port=50000;
 protocol=TCPIP;uid=root; pwd=pass");
?>
```

这里，localhost、5000、TCPIP、root、pass 和 mydb 分别表示要连接的数据库服务其所在地址、端口号、连接协议、用户名、密码和数据库名称。

3.3.2　数据的插入、更新与删除

在 ADOdb 中，数据的插入、更新和删除均可以通过 Connection 对象的 Execute 方法来执行一个 SQL 语句实现，其语法格式如下所示。

```
$conn->Execute($sql);
```

其中$conn 为 Connection 对象，$sql 为 SQL 语句，如以下代码所示，执行了一条 insert 语句。

```
<?php
include("adodb/adodb.inc.php");                          //包含 adodb 类库文件
$conn = NewADOConnection('mysql');                       //连接数据库
$conn -> Connect('localhost', 'root', 'pass', 'mydb');
```

```
$conn -> Execute(“insert into mytable values(100001, ‘Simon’)”);　　//执行 SQL
?>
```

使用 Execute 方法执行 SQL 语句时，如果 SQL 语句存在错误，Execute 语句将不会有任何错误提示，只是返回值为 false。在 ADOdb 中，使用 ErrorMsg 方法来获取执行 SQL 语句时的错误信息，其语法格式如下所示。

```
$conn->ErrorMsg();
```

其中$conn 为 Connection 对象，如以下代码所示，得到了前面执行 insert 语句时的错误信息。

```
<?php
include("adodb/adodb.inc.php");　　//包含 adodb 类库文件
$conn = NewADOConnection('mysql');　　//创建 adodb 对象，声明数据库类型为 MySQL
$conn->Connect("localhost", "root", "", "mydb");　　//连接数据库，其中 mydb 为数据库名
$sqle = $conn->Execute("insert into mytable values(100001, 'Simon')");　　//使用$sqle 来确定 SQL 是否执行成功
if($sqle)　　//如果执行成功，则输出执行成功的信息
{
　　echo "SQL 执行成功";
}
else　　//如果执行失败，则输出错误信息
{
　　echo $conn->ErrorMsg();
}
?>
```

上面的代码通过变量$sqle 来保存 Execute 函数的返回值，当$sqle 为 true 的时候，输出 SQL 执行成功的信息，当$sqle 为 false 的时候，输出相应的错误信息。

3.3.3　数据查询

进行数据查询的方法与前面执行数据插入、更新和删除的方法基本相同。所不同的是这次的 SQL 语句是一条 SELECT 语句。如果语句执行成功，则返回一个结果集。如果语句执行失败，则返回 false。一个结果集可以看作存储 SELECT 语句执行结果的对象。以下代码执行了一个 SELECT 语句并将执行结果保存在结果集对象$rs 中。

```
<?php
include("adodb/adodb.inc.php");　　//包含 adodb 类库文件
$conn = NewADOConnection('mysql');　　//创建 adodb 对象，声明数据库类型为 MySQL
$conn->Connect("localhost", "root", "", "mydb");　　//连接数据库，其中 mydb 为数据库名
$rs = $conn->Execute("SELECT * FROM mytable");　　//执行 SQL 语句，将结果保存在结果集中
if($rs)　　//如果执行成功，则输出语句成功执行的信息
{
　　echo “语句执行成功”;
}
else　　//如果执行失败，则输出错误信息
{
　　echo $conn->ErrorMsg();
}
?>
```

可以看出，上面的程序与 1.3.2 节中的区别并无太大的区别。两者都是使用 Execute 方法来执行 SQL 语句，使用 ErrorMsg 方法来获取 SQL 语句的错误信息。

在读取结果集对象时，程序是逐行读取的，每次读取一行。对于$rs 结果集对象，常用的属性和方

法有以下四个。

- $rs->EOF：用于检测$rs 是否为空，或者指针是否已经移过最后一行。
- $rs->fields[]：ADOdb 将当前读取的行存储在$rs->fields[]数组中，该数组以列从左到右依次排列从 0 开始作为键名，也可以以列名为键名。
- $rs->MoveNext：MoveNext 方法用于将读取结果集的指针向下移动一行。
- $rs->RecordCount：RecordCount 方法用于获取结果集的记录数，当结果集的记录行数无法获得时，返回-1。

以下代码在前面的基础上进行了一些扩充，循环读取了结果集中的所有行。

```
<?php
include("adodb/adodb.inc.php");                          //包含 adodb 类库文件
$conn = NewADOConnection('mysql');                       //创建 adodb 对象，声明数据库类型为 MySQL
$conn->Connect("localhost", "root", "", "mydb");         //连接数据库，其中 mydb 为数据库名
$rs = $conn->Execute("SELECT * FROM mytable");           //执行 SQL 语句，将结果保存在结果集中
if($rs)                                                  //如果执行成功，则循环读取结果集
{
    while (!$rs->EOF)                                    //循环读取$rs 中的所有记录
    {
        echo $rs->fields[0].' '.$rs->fields[1].'<BR>';   //输出当前行
        $rs->MoveNext();                                 //将指针移到下一条记录
    }
}
else                                                     //如果执行失败，则输出错误信息
{
    echo $conn->ErrorMsg();
}
?>
```

上面的例子使用 while 语句对结果集中的所有记录进行循环，当指针移到最后一条记录以后时结束循环。在循环体内，输出每个记录的第一列和第二列。运行结果如下所示。

```
100001 Simon
100002 Elaine
100003 Susan
```

对于$rs->fields 数组，一个更有意义的写法是使用列名来代替数字，如以下代码所示。

```
<?php
include("adodb/adodb.inc.php");                          //包含 adodb 类库文件
$conn = NewADOConnection('mysql');                       //创建 adodb 对象，声明数据库类型为 MySQL
$conn->Connect("localhost", "root", "", "mydb");         //连接数据库，其中 mydb 为数据库名
$rs = $conn->Execute("SELECT * FROM mytable");           //执行 SQL 语句，将结果保存在结果集中
if($rs)                                                  //如果执行成功，则循环读取结果集
{
    while (!$rs->EOF)                                    //循环读取$rs 中的所有记录
    {
        echo $rs->fields['serial_no'].' '.$rs->fields['name'].'<BR>';    //输出当前行
        $rs->MoveNext();                                                 //将指针移到下一条记录
    }
}
else                                                     //如果执行失败，则输出错误信息
{
    echo $conn->ErrorMsg();
```

```
}
?>
```

其运行结果与前面相同。

3.3.4　记录分页的实现

ADOdb 提供了一个非常好的分页类库，使用户能够非常容易的使用 ADOdb 创建记录分页页面。文件 adodb-pager.inc.php 用于存储分页类库。如果需要使用 ADOdb 的分页类库创建分页页面，则需要在 PHP 代码中包含这个文件，如以下代码所示。

```
<?php
include("adodb/adodb.inc.php");                    //包含 adodb 类库文件
include("adodb/adodb-pager.inc.php");              //包含 adodb-pager 类库文件
?>
```

分页对象的创建方法如下所示。

```
$pager = new ADODB_Pager($conn, $sql);
```

其中，$pager 为分页对象，$conn 为使用 NewADOConnection 创建的连接对象，$sql 为一个 SQL 语句字符串。

分页对象的 Render 方法用于输出分页页面，其语法格式如下所示。

```
$pager -> Render($rows_per_page);
```

其中，$rows_per_page 为每页显示的记录数。

以下代码是一个使用分页类库创建分页记录页面的例子。

```
<?php
include("adodb/adodb.inc.php");                    //包含 adodb 类库文件
include("adodb/adodb-pager.inc.php");              //包含 adodb-pager 类库文件
$conn = NewADOConnection('mysql');                 //创建 adodb 对象，声明数据库类型为 MySQL
$conn->Connect("localhost", "root", "", "mydb");   //连接数据库，其中 mydb 为数据库名
$sql = "select * from mytable";                    //定义要执行的 SQL 语句

$pager = new ADODB_Pager($conn, $sql);             //根据连接对象和 SQL 语句创建分页对象
$pager->Render(2);                                 //输出分页的页面，每页 2 条记录
?>
```

运行结果如图 3-1 所示。

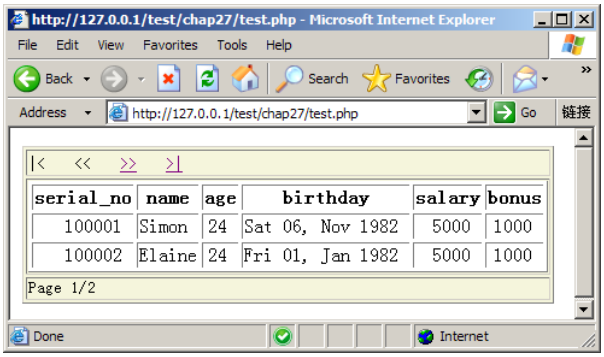

图 3-1　记录分页页面

在使用 ADOdb 的分页类库创建分页页面的实际应用中，一个最常用的操作是修改列名。例如，在上面的例子中使用的数据表中的列名都为英文单词，在页面的显示时，可能希望将其替换成中文。更改

列名的方法可以通过修改 SQL 语句，即在 SQL 语句中指定一个列的别名，如以下代码所示。

```
select serial_no as '序号', name as '姓名', age as '年龄', birthday as '生日', salary as '工资', bonus as '奖金'
from mytable
```

上面的 SQL 语句通过使用 AS 对每个列指定一个别名。以下代码使用分页类库为上面的 SQL 语句创建了分页记录页面。

```
<?php
include("adodb/adodb.inc.php");                    //包含 adodb 类库文件
include("adodb/adodb-pager.inc.php");              //包含 adodb-pager 类库文件
$conn = NewADOConnection('mysql');                 //创建 adodb 对象，声明数据库类型为 MySQL
$conn->Connect("localhost", "root", "", "mydb");   //连接数据库，其中 mydb 为数据库名
$sql = "select serial_no as '序号', name as '姓名', age as '年龄', birthday as '生日', salary as '工资', bonus as '奖金'
from mytable";                                     //定义要执行的 SQL 语句

$pager = new ADODB_Pager($conn, $sql);             //根据连接对象和 SQL 语句创建分页对象
$pager->Render(2);                                 //输出分页的页面，每页 2 条记录
?>
```

运行结果如图 3-2 所示。

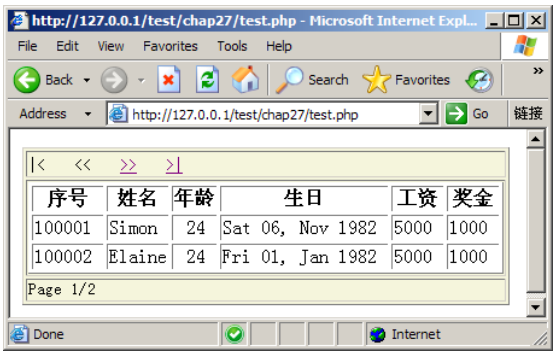

图 3-2　更改了列名的记录分页页面

在实际应用中，往往还会根据网站整体布局需要修改分页页面。这种情况下，就需要修改 adodb-pager.inc.php 文件中的 ADODB_Pager 类。adodb-pager.inc.php 文件已经提供了大量的注释以方便用户修改。ADODB_Pager 类的属性中也包含了一些供定制的文字。

以下代码修改了 adodb-pager.inc.php 文件，将翻页的链接修改成了中文的链接，修改这些链接是通过修改$first、$prev、$next 和$last 的值来实现的，如下所示。

```
var $first = '<code>|&lt;</code>';
var $prev = '<code>&lt;&lt;</code>';
var $next = '<code>>></code>';
var $last = '<code>>>|</code>';
```

修改后的代码如下所示。

```
var $first = '<code>首页</code>';
var $prev = '<code>上一页</code>';
var $next = '<code>下一页</code>';
var $last = '<code>尾页</code>';
```

修改后代码的运行结果如图 3-3 所示。

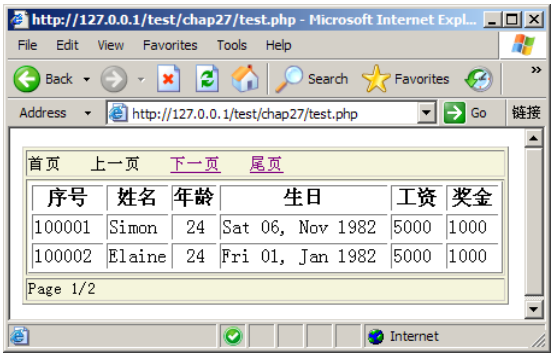

图 3-3　定制分页链接

3.3.5　多个数据库的连接方法

由于 ADOdb 可以方便的连接到多种不同的数据库，在实际应用中可以很容易的在一段 PHP 脚本中对两种不同的数据库进行操作。以下代码通过读取 MySQL 数据库将其中的 mytable 表中的全部记录存放在一个 Access 数据库文件中。

```
<?php
include("adodb/adodb.inc.php");                                    //包含 adodb 类库文件

$conn_mysql = NewADOConnection('mysql');                           //创建 adodb 对象，声明数据库类型为 MySQL
$conn_mysql->Connect("localhost", "root", "", "mydb");             //连接数据库，其中 mydb 为数据库名

$conn_access = NewADOConnection('access');                         //创建 adodb 对象，声明数据库类型为 Access
$conn_access->Connect("Driver={Microsoft Access Driver (*.mdb)};Dbq=d:\\db1.mdb;");
                                                                   //连接数据库，其中 d:\\db1.mdb 为数据库文件

$rs = $conn_mysql->Execute("select * from mytable");               //执行 SQL 语句选择 MySQL 数据库中的记录

if($rs)                                                            //如果有结果集返回则逐条读取记录
{
    while(!$rs->EOF)                                               //循环读取结果集
    {
        $serial_no = $rs->fields[0];                               //获取当前行的每一个字段
        $name = $rs->fields[1];
        $age = $rs->fields[2];
        $birthday = $rs->fields[3];
        $salary = $rs->fields[4];
        $bonus = $rs->fields[5];

        //对每条数据均向 Access 数据库执行一个插入操作
        $insertsql = "insert into mytable values($serial_no, '$name', $age, '$birthday', $salary, $bonus)";
        $sqle = $conn_access->Execute();
        if(!$sqle)                                                 //如果 SQL 执行出错则输出错误信息
        {
            echo conn_access->ErrorMsg();
        }
        $rs->MoveNext();                                           //读取下一条记录
```

```
    }

}
else                                                    //如果没有数据返回则输出错误信息
{
    echo "没有获得结果集，错误信息：".conn_mysql->ErrorMsg();
}
```

执行后，如果程序运行无误，在页面上将什么都不输出。但是访问相应的 Access 数据库文件，可以看到如图 3-4 所示的记录。

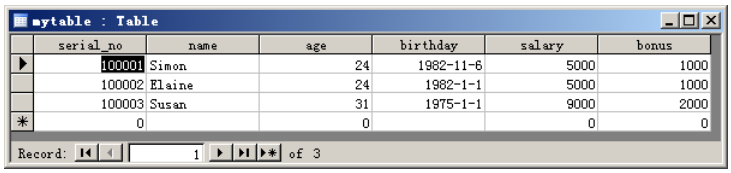

图 3-4　Access 的结果

可以看到，在程序运行后，Access 数据库获得了与 MySQL 数据库中同样的结果，如下所示。

```
mysql> select * from mytable;
+-----------+---------+-----+------------+---------+---------+
| serial_no | name    | age | birthday   | salary  | bonus   |
+-----------+---------+-----+------------+---------+---------+
|    100001 | Simon   |  24 | 1982-11-06 | 5000.00 | 1000.00 |
|    100002 | Elaine  |  24 | 1982-01-01 | 5000.00 | 1000.00 |
|    100003 | Susan   |  31 | 1975-01-01 | 9000.00 | 2000.00 |
+-----------+---------+-----+------------+---------+---------+
7 rows in set (0.00 sec)
```

3.4　ADOdb 的程序调试

在 ADOdb 中，由于其封装性，很多运行时的错误信息无法被显示出来。这对于程序的调试非常不方便。为此，ADOdb 提供了在 PHP 代码中增加调试信息的方法。

为 ADOdb 增加调试信息是通过设置 Connection 对象的 debug 属性来实现的。将 debug 属性设置成 true 以后将在页面输出的同时输出代码调试信息，其语法格式如下所示。

```
$conn->debug=true;
```

通过设置，可以使 PHP 代码在运行前显示 SQL 语句，并显示 SQL 语句运行时的错误信息。下面的代码在 1.3.5 节的代码前将 debug 属性设置为 true，如下所示。

```
<?php
include("adodb/adodb.inc.php");                         //包含 adodb 类库文件

$conn_mysql = NewADOConnection('mysql');                //创建 adodb 对象，声明数据库类型为 MySQL
$conn_mysql->Connect("localhost", "root", "", "mydb");  //连接数据库，其中 mydb 为数据库名

$conn_mysql->debug = true;                              //设置 conn_mysql 对象的 debug 属性为 true

$conn_access = NewADOConnection('access');              //创建 adodb 对象，声明数据库类型为 Access
$conn_access->Connect("Driver={Microsoft Access Driver (*.mdb)};Dbq=d:\\db1.mdb;");
```

```
                                                                    //连接数据库，其中 d:\\db1.mdb 为数据库文件
$conn_access->debug = true;                                         //设置 conn_access 对象的 debug 属性为 true

$rs = $conn_mysql->Execute("select * from mytable");               //执行 SQL 语句选择 MySQL 数据库中的记录

if($rs)                                                             //如果有结果集返回则逐条读取记录
{
    while(!$rs->EOF)                                                //循环读取结果集
    {
        $serial_no = $rs->fields[0];                                //获取当前行的每一个字段
        $name = $rs->fields[1];
        $age = $rs->fields[2];
        $birthday = $rs->fields[3];
        $salary = $rs->fields[4];
        $bonus = $rs->fields[5];

        //对每条数据均向 Access 数据库执行一个插入操作
        $insertsql = "insert into mytable values($serial_no, '$name', $age, '$birthday', $salary, $bonus)";
        $sqle = $conn_access->Execute();
        if(!$sqle)                                                  //如果 SQL 执行出错则输出错误信息
        {
            echo conn_access->ErrorMsg();
        }
        $rs->MoveNext();                                            //读取下一条记录
    }

}
else                                                                //如果没有数据返回则输出错误信息
{
    echo "没有获得结果集，错误信息：".conn_mysql->ErrorMsg();
}
```

运行结果如下所示。

```
---------------------------------------------------------------------------
(mysql): select * from mytable
---------------------------------------------------------------------------


---------------------------------------------------------------------------
(access): insert into mytable values(100001, 'Simon', 24, '1982-11-06', 5000.00, 1000.00)
---------------------------------------------------------------------------


---------------------------------------------------------------------------
(access): insert into mytable values(100002, 'Elaine', 24, '1982-01-01', 5000.00, 1000.00)
---------------------------------------------------------------------------


---------------------------------------------------------------------------
(access): insert into mytable values(100003, 'Susan', 31, '1975-01-01', 9000.00, 2000.00)
---------------------------------------------------------------------------
```

可以看出，代码在运行时的 SQL 语句均被显示出来。如果在 SQL 语句的运行过程中存在任何错误，均可以通过调试信息检测出来。

3.5　小结

本章介绍了 PHP 的 ADOdb 类库。ADOdb 类库为 Windows 下的程序员使用 PHP 操作数据库提供了很大的方便。编程时，可以很容易的是用类似于 ASP 的方法来进行 PHP 数据库程序的编写。

对于一些小型的数据库程序，使用 ADOdb 类库进行编写可以有效的提高效率。但是，由于 ADOdb 类库的功能限制，在很多功能的实现方面不够灵活。因此，读者应根据实际情况选择是否使用 ADOdb 类库来进行数据库操作。并且，在适当时候可以通过修改 ADOdb 类库的代码来实现需要的功能。