

第 18 章 PHP 操作数据库

在实际的应用中，通常使用数据库存储数据信息。虽然可以直接对数据库进行操作，但是对于 Web 系统而言，更多是使用程序对数据库进行相关操作。例如，读取数据库信息、从数据库查询信息等。本章主要内容如下。

- ❑ PHP 操作与管理 MySQL 数据库。
- ❑ PHP 操作 SQL Server 数据库。
- ❑ PHP 操作 Access 数据库。

通过对本章的学习，读者能够使用 PHP 对常用的数据库进行相关操作，并能够根据实际需要进行相关管理系统的开发。

18.1 PHP 操作 MySQL 数据库

在 PHP 中，支持对多种数据库的操作，且提供了相关的数据库连接函数或操作函数。特别是与 MySQL 数据库的组合，PHP 提供了强大的数据库操作函数，读者可直接在 PHP 中使用这些函数进行数据库的操作。操作数据库，首先需要进行数据库的连接，然后选择需要进行操作的数据库，再执行相关的数据库操作，最后需要关闭所建立的数据库连接。这一节将对在 PHP 中如何进行数据库的操作进行详细地讲解。

18.1.1 连接 MySQL 数据库

在 PHP 中，要对数据库进行操作，首先需要连接数据库。连接数据库可使用 `mysql_connect()` 函数，其语法格式如下所示。

```
resource mysql_connect ([ string $server [, string $username [, string $password [, bool $new_link [, int $client_flags ]]]]] )
```

其中，参数 `server` 为要连接的数据库服务器的名称或 IP；参数 `username` 为连接数据库的用户名，若没有设置该参数，则默认为服务器进程所有者的用户名；参数 `password` 为连接数据库的密码，如果未设置该参数，则默认为空。

若每一次都使用一样的参数进行连接，在 PHP 中将不会进行新的数据库连接，而是直接返回已打开的数据库连接标识。参数 `new_link` 则改变了此行为，若设置该参数为布尔值 `true`，则将在每一次使用 `mysql_connect` 函数进行数据库连接时打开新的数据库连接，甚至在之前曾使用同样的参数进行过数据库的连接。

参数 `client_flags` 为设置客户端信息，它可以是以下常用的组合。

- ❑ `MYSQL_CLIENT_COMPRESS`: 在客户端使用压缩的通讯协议。
- ❑ `MYSQL_CLIENT_IGNORE_SPACE`: 允许在函数名后留空格位。
- ❑ `MYSQL_CLIENT_INTERACTIVE`: 允许设置断开连接之前所空闲等候的 `interactive_timeout` 时间。
- ❑ `MYSQL_CLIENT_SSL`: 使用 SSL 协议进行加密。

该函数尝试打开或重复使用一个已打开的 MySQL 数据库服务器的连接。若成功连接，MySQL 数据库服务器，返回一个 MySQL 连接标识，否则返回布尔值 `false`。

【示例 18-1】如何在 PHP 脚本中进行 MySQL 数据库服务器的连接。代码如下所示。

```
<?php
mysql_connect("localhost","root","root");    //连接至本地 MySQL 服务器，用户名和密码均为 root
?>
```

分析：在上述代码中使用 `mysql_connect` 函数连接本地 MySQL 数据库服务器，连接数据库的用户名和密码均为 `root`。

注意：若数据库服务器不可用，或连接数据库的用户名或密码错误，则可能会引起一条 PHP 警告信息，如下所示。

```
Warning: mysql_connect() [function.mysql-connect]: Access denied for user 'root'@'localhost' (using
password: YES) in D:\xampplite\htdocs\book\source\18\18.1.php on line 2
```

在该警告信息中，提示使用用户名 `root` 无法连接本地 MySQL 数据库，并且该警告信息并不能停止脚本的继续执行。同时也将暴露数据库连接的敏感信息，很不利于数据库的数据安全。为此通常在进行数据库连接时，在连接函数前使用 “@” 符号抑制错误信息的输出，然后在连接函数后使用 `DIE` 函数指定错误信息并停止脚本的执行。

【示例 18-2】在 PHP 脚本中如何安全地连接 MySQL 数据库服务器。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
or die("数据库连接失败！");    //连接本地数据库服务器
?>
```

分析：在上述程序中，使用 “@” 符号抑制连接数据库服务器的错误信息的输出，并使用 `die()` 函数抛出定制的错误信息，终止整个脚本的执行。

注意：包含创建数据库连接的脚本一结束，与服务器的连接就被关闭，除非之前已经明确调用 `mysql_close()` 函数关闭了数据库连接。

在实际的应用中，通常在多个脚本文件中都需要进行数据库的连接，此时为了维护方便和节省代码，可将数据库连接放在一个单独的文件中，在需要使用数据库连接的脚本中使用 `include()` 函数或 `require()` 函数引用该文件。也可将数据库连接作为一个单例，在每一个需要使用的脚本中调用该单例。

18.1.2 断开与 MySQL 的连接

通常在完成数据库的使用后，需要断开与 MySQL 数据库服务器的连接。断开与 MySQL

数据库服务器的连接通常使用函数 `mysql_close()` 函数，其语法格式如下所示。

```
bool mysql_close ( [ resource $link_identifier ] )
```

其中，参数 `link_identifier` 为数据库连接标识。若没有该参数，则关闭上一个已打开的非持久数据库连接。函数关闭指定的连接标识所关联的非持久数据库连接。

注意：`mysql_close()` 函数不会关闭由 `mysql_pconnect()` 函数所建立的持久连接。

【示例 18-3】在 PHP 脚本中关闭一个由 `mysql_connect()` 函数所建立的数据库连接。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
or die("数据库连接失败！");           //连接数据库服务器
mysql_close();                         //关闭数据服务器连接
?>
```

分析：在上述代码中，使用 `mysql_close()` 函数关闭一个已创建的非持久数据连接。

虽然在前面已讲到，创建数据库连接的脚本一结束，其数据库连接自动关闭。但是从节省服务器资源层面上讲，在使用完数据库连接后，使用 `mysql_close()` 函数关闭数据库连接能更有效地节省服务器资源。

18.1.3 选择和使用 MySQL 数据库

在进行数据库的连接后，需要在 PHP 脚本中选择需要进行操作的 MySQL 数据库。选择数据库可使用 `mysql_select_db()` 函数，其语法格式如下所示。

```
bool mysql_select_db ( string $database_name [, resource $ link_identifier ] )
```

其中，参数 `database_name` 为要进行操作的数据库，参数 `link_identifier` 为创建的数据库连接标识。如果没有指定数据库连接标识，则使用上一个已打开的数据库连接标识；若没有已打开的数据库连接标识，函数将尝试使用无参数调用 `mysql_connect()` 创建数据库连接并使用。函数尝试设定与指定的连接标识符所关联的 MySQL 数据库服务器上的当前激活数据库。若操作成功，则返回布尔值 `true`，否则返回 `false`。

【示例 18-4】在 PHP 脚本中如何选择 MySQL 数据库服务器上的数据库。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
or die("数据库连接失败！");
@mysql_select_db("mydb")
or die("选择的数据库不存在或不可用！");
mysql_close();
?>
```

分析：在上述程序中，使用当前数据库连接选择 `mydb` 数据库作为当前活动数据库，对数据库的所有操作都将作用于该活动数据库。读者可以看出，这时选择数据作为当前活动数据库，其实就相当于在 MySQL 命令行所使用的 `use` 命令。

18.1.4 执行 MySQL 指令

进行数据库的操作,在 PHP 中需要使用一函数来执行 MySQL 指令,这就是 `mysql_query()` 函数。其语法格式如下所示。

```
resource mysql_query ( string $query [, resource $link_identifier ])
```

其中,参数 `query` 为要执行的 MySQL 语句,参数 `link_identifier` 为打开的数据库连接。若没有设置参数 `link_identifier`,函数则使用上一个已打开的数据库连接;如果没有已打开的数据库连接,该函数则尝试以无参数方式调用 `mysql_connect()` 函数创建一个数据库连接并使用。

函数向参数 `link_identifier` 所关联的数据库服务器中的当前活动数据库发送一条查询。函数仅对 `select`、`show`、`explain` 和 `describe` 语句返回一个资源标识符用于存储 SQL 语句的执行结果,若执行不成功则返回 `false`。对于其他类型的 SQL 语句,函数在执行成功时返回 `true`,出错时将返回 `false`。因此,任何非 `false` 的返回值意味着此次查询合法并能够被 SQL 服务器成功执行,但并不意味着影响或者返回的行数,因为有可能此次 SQL 语句成功执行,但是并没有影响到或返回任何一行。

【示例 18-5】在 PHP 脚本中如何执行 SQL 语句。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
or die("数据库连接失败!");           //连接数据库服务器
@mysql_select_db("mydb")
or die("选择的数据库不存在或不可用!"); //选择数据库
$myquery = @mysql_query("select * from userinfo")
or die("SQL 语句执行失败!");         //执行 SQL 语句
mysql_close();                         //关闭数据库连接
?>
```

分析:在上述程序中,进行了一次完整的数据连接操作。首先连接数据库服务器,然后选择数据库,执行 SQL 语句,最后关闭数据库连接。使用了 `mysql_query()` 函数执行 SQL 语句。

除了 `mysql_query()` 函数能够执行 SQL 语句外,PHP 还提供了另一函数 `mysql_db_query()`。该函数与 `mysql_query()` 函数具有相同的功能,其区别在于 `mysql_db_query()` 函数在执行 SQL 语句时可以同时选择数据库。其语法格式如下所示。

```
resource mysql_db_query ( string $database , string $query [, resource $ link_identifier ] )
```

其中,参数 `database` 为要执行 SQL 语句的数据库,参数 `query` 为要执行的 SQL 语句,参数 `link_identifier` 为数据库连接标识符。若没有设置该参数,函数将打开上一个已创建的数据库连接;若找不到已打开的数据库连接,则尝试以无参数方式调用 `mysql_connect()` 函数创建一个数据库连接。

函数在执行成功时根据查询结果返回一个 MySQL 结果资源号,出错时将返回 `false`。函数会对 `INSERT`、`UPDATE` 和 `DELETE` 语句返回 `true` 或 `false` 来指示执行成功或失败。

注意:在使用该函数后,不会自动切换回先前使用 `mysql_select_db()` 函数连接的数据库。若想再次使用先前连接的数据库,需要再次手动指定,因此建议在查询时使用 `database.table` 的格式来替换该函数。

【示例 18-6】采用 `mysql_db_query()` 函数重写【示例 18-5】。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
or die("数据库连接失败！");           //连接数据库服务器
$myquery = @mysql_db_query("mydb","select * from userinfo")
or die("SQL 语句执行失败!");           //采用 mysql_db_query 进行查询
mysql_close();                           //关闭数据库连接
?>
```

分析：在上述程序中，使用 `mysql_db_query()` 函数进行数据库的查询。因为可以在该函数内指定要查询的数据库，所以在以上脚本中不再指定要查询的数据库。

18.1.5 分析结果集

在每一次成功执行 SQL 语句后，`mysql_query()` 函数总是会返回一个结果集。要对结果集进行分析，首先需要获取所执行的 SQL 语句影响的行数。

1. 获取影响的行数

对于该结果集中包含的记录数，可使用 `mysql_num_rows()` 函数获取。其语法格式如下所示。

```
int mysql_num_rows ( resource $result )
```

其中，参数 `result` 为函数 `mysql_query()` 所返回的结果集。函数返回结果集中的记录数。

【示例 18-7】如何在 PHP 脚本中获取结果集中的记录数。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
or die("数据库连接失败！");           //连接数据库服务器
@mysql_select_db("mydb")
or die("选择的数据库不存在或不可用!"); //选择数据库
$myquery = @mysql_query("select * from userinfo")
or die("SQL 语句执行失败!");           //执行 SQL 语句
echo "结果集中的行数为：" . mysql_num_rows($myquery); //获取结果集的记录数
mysql_close();                           //关闭数据库连接
?>
```

分析：在上述程序中，使用 `mysql_num_rows` 函数获取结果集中的记录数。

若在 `mysql_query()` 函数中使用 `INSERT`、`UPDATE` 和 `DELETE` 语句，应使用 `mysql_affected_rows()` 函数获取所影响到的记录数，其语法格式如下所示。

```
int mysql_affected_rows ([ resource $link_identifier ] )
```

其中，参数 `link_identifier` 为已打开的数据库连接标识符。若未设置该参数，函数默认使用上一次所打开的数据库连接；若未找到该连接，函数将尝试以无参数方式调用 `mysql_connect()` 函数建立数据库连接并使用；若发生意外，如找不到数据库连接或创建数据库连接失败时，将产生一条警告信息。

函数返回由参数 `link_identifier` 所关联的数据库连接进行的 `INSERT`、`UPDATE` 和

DELETE 查询所影响到的行数。函数执行成功则返回最近一次操作所影响的行数，若最近一次查询失败，则返回-1。在使用 UPDATE 语句时，MySQL 不会将原值与新值一样的列进行更新，因此该函数所返回的值不一定就是使用 mysql_query() 函数所影响到的行数，此时只是返回真正被更新的行数。

2. 获取结果集中的数据

要显示对于使用 mysql_query() 函数所返回的结果集，首先须获取结果集中的数据。获取结果集中的某一条数据可使用 mysql_result() 函数，其语法格式如下所示。

```
mixed mysql_result ( resource $result , int $row [, mixed $field ] )
```

其中，参数 result 为函数 mysql_query() 所返回的结果集，参数 row 为指定要返回的结果集中的某一行的行号，参数 field 为要返回的列名或列的偏移量。若未设置该参数，默认返回第一列的值。函数返回结果集中一个单元的数据。

【示例 18-8】在 PHP 脚本中显示某一行记录。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
or die("数据库连接失败！");           //连接数据库服务器
@mysql_select_db("mydb")
or die("选择的数据库不存在或不可用!"); //选择数据库
mysql_query("set names gb2312");        //设置输出编码
$myquery = @mysql_query("select * from userinfo")
or die("SQL 语句执行失败!");           //执行 SQL 语句
echo "id:" . mysql_result($myquery, 1, 0) . "<br>"; //显示结果集第一行第一列数据
echo "姓名:" . mysql_result($myquery, 1, 1) . "<br>"; //显示结果集第一行第二列数据
echo "性别:" . mysql_result($myquery, 1, 2) . "<br>"; //显示结果集第一行第三列数据
echo "地址:" . mysql_result($myquery, 1, 3) . "<br>"; //显示结果集第一行第四列数据
echo "邮件:" . mysql_result($myquery, 1, 4) . "<br>"; //显示结果集第一行第五列数据
mysql_close();                          //关闭数据库连接
?>
```

分析：在上述程序中，使用 mysql_result() 函数获取结果集中第一行的数据信息。

注意：因 MySQL 服务器中存储数据的编码与浏览器显示结果的编码有可能不一致所以在程序中，使用了 set names gb2312 直接指定其内容输出的编码。若未指定输出字符编码，则可能出现乱码。

以上实现了获取结果集中的某一行数据。其实通过与 mysql_num_rows() 函数所返回的行数，可以输出结果集中的所有数据。

【示例 18-9】如何显示 mysql_query() 所返回的结果集中的所有信息。代码如下所示。

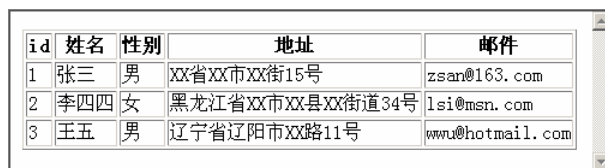
```
<?php
@mysql_connect("localhost","root","root")
or die("数据库连接失败！");           //连接数据库服务器
@mysql_select_db("mydb")
or die("选择的数据库不存在或不可用!"); //选择数据库
mysql_query("set names gb2312");        //设置输出字符编码
$myquery = @mysql_query("select * from userinfo")
or die("SQL 语句执行失败!");           //执行 SQL 语句
```

```

$rowscnt = mysql_num_rows($myquery);           //获取结果集行数
echo "<table border='1'><tr><th>id</th><th> 姓名 </th><th> 性别 </th><th> 地 址 </th><th> 邮 件
</th></tr>";
for($i=0; $i < $rowscnt; $i++){                //对结果集进行循环
echo "<tr><td>" . mysql_result($myquery, $i, 0) . "</td>";
echo "<td>" . mysql_result($myquery, $i, 1) . "</td>";
echo "<td>" . mysql_result($myquery, $i, 2) . "</td>";
echo "<td>" . mysql_result($myquery, $i, 3) . "</td>";
echo "<td>" . mysql_result($myquery, $i, 4) . "</td></tr>";
}
echo "</table>";
mysql_close();                                 //关闭数据库连接
?>

```

分析：在上述程序中，通过对结果集进行循环，输出结果集中的所有数据信息。其输出结果如图 18-1 所示。



id	姓名	性别	地址	邮件
1	张三	男	XX省XX市XX街15号	zsar@163.com
2	李四四	女	黑龙江省XX市XX县XX街道34号	lsi@msn.com
3	王五	男	辽宁省辽阳市XX路11号	www@hotmail.com

图 18-1 输出结果

3. 逐行获取结果集中的记录

当需要在结果集中获取记录时，使用 `mysql_result()` 函数就显得有些复杂，并且处理效率也差了很多。为此，PHP 提供了从结果集中获取整行记录的函数 `mysql_fetch_row()`，其语法格式如下所示。

```
array mysql_fetch_row ( resource $result )
```

其中，参数 `result` 为 `mysql_query()` 函数所返回的结果集。该函数从指定的结果集中取得一行数据并以数组的形式返回。数组以列的顺序依次进行分配。依次调用该函数将返回结果中的下一行，若没有更多行时则返回 `false`。

【示例 18-10】采用 `mysql_result` 函数获取结果集中的所有数据。代码如下所示。

```

<?php
@mysql_connect("localhost","root","root")
or die("数据库连接失败！");                    //连接数据库服务器
@mysql_select_db("mydb")
or die("选择的数据库不存在或不可用!");          //选择数据库
mysql_query("set names gb2312");                 //设置输出字符编码
$myquery = @mysql_query("select * from userinfo")
or die("SQL 语句执行失败!");                    //执行 SQL 语句
echo "<table border='1'><tr><th>id</th><th> 姓名 </th><th> 性别 </th><th> 地 址 </th><th> 邮 件
</th></tr>";
while($row = mysql_fetch_row($myquery)){        //循环获取结果集中的每一行
echo "<tr><td>" . $row[0] . "</td>";
echo "<td>" . $row[1] . "</td>";
echo "<td>" . $row[2] . "</td>";
echo "<td>" . $row[3] . "</td>";
echo "<td>" . $row[4] . "</td></tr>";
}

```

```

}
echo "</table>";
mysql_close();                                //关闭数据库连接
?>

```

分析：在上述程序中，使用 `mysql_fetch_row()` 函数循环获取结果集中的每一行。读者可以发现，其执行的结果与前例结果完全相同。

在函数 `mysql_fetch_row()` 所返回的每一行结果数组中，该行的第一列在数组中是以 0 开始，所以在实际的应用中很不方便。PHP 提供了另一功能强大的函数 `mysql_fetch_array()`，其语法格式如下所示。

```
array mysql_fetch_array ( resource $result [, int $ result_type ] )
```

其中，参数 `result` 为使用 `mysql_query()` 函数所返回的结果集，参数 `result_type` 为下列取值的一种。

- ❑ **MYSQL_ASSOC**：返回根据结果集中的某一行所生成的关联数组。关联数组以数据库中的列名作为数组键名，其返回数组与函数 `mysql_fetch_assoc()` 返回结果一样。
- ❑ **MYSQL_NUM**：返回根据结果集中的某一行所生成的索引数组。索引数组中的键名以 0 开始依次顺序分配，其返回结果数组与函数 `mysql_fetch_row()` 返回的结果一样。
- ❑ **MYSQL_BOTH**：返回根据结果集中的某一行所生成的关联和索引数组，在未指定结果类型时默认为该值。

函数返回根据从结果集中取得的一行所生成的数组，若没有更多行时返回 `false`。

注意：该函数所返回的数组中列名是区分大小写的。

【示例 18-11】在 PHP 中使用 `array mysql_fetch_array` 函数获取结果集中的数据。代码如下所示。

```

<?php
@mysql_connect("localhost","root","root")
or die("数据库连接失败！");                                //连接数据库服务器
@mysql_select_db("mydb")
or die("选择的数据库不存在或不可用!");                    //选择数据库
mysql_query("set names gb2312");                            //设置输出字符编码
$myquery = @mysql_query("select * from userinfo")
or die("SQL 语句执行失败!");                                //执行 SQL 语句
echo "<table border='1'><tr><th>id</th><th>姓名</th><th>性别</th><th>地址</th><th>邮件</th></tr>";
while($row = mysql_fetch_array($myquery, MYSQL_BOTH)){      //循环获取结果集中的数据
echo "<tr><td>" . $row[0] . "</td>";
echo "<td>" . $row[1] . "</td>";
echo "<td>" . $row["sex"] . "</td>";
echo "<td>" . $row[3] . "</td>";
echo "<td>" . $row["email"] . "</td></tr>";
}
echo "</table>";
mysql_close();                                              //关闭数据库连接
?>

```

分析：在上述程序中，使用 `mysql_fetch_array()` 函数循环获取结果集。读者可以发现，程序输出结果与前例结果完全一样。因该函数具有很强的灵活性，在实际应用中通常使用该函数。

当结果集很大时,使用上述的方式会使页面很长,不美观,并且用户查询信息也很麻烦,因此需要对结果集进行分页显示。

【示例 18-12】分页显示结果集中的数据。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")           //连接数据库服务器
or die("数据库连接失败!");
@mysql_select_db("mydb")                             //选择数据库
or die("选择的数据库不存在或不可用!");
mysql_query("set names gb2312");                     //设置输出字符编码
$myquery = @mysql_query("select * from userinfo")    //执行 SQL 语句
or die("SQL 语句执行失败!");
$page_size = 10;                                     //每页显示记录数
$num_cnt = mysql_num_rows($myquery);                 //获取记录总数
$page_cnt = ceil($num_cnt / $page_size);             //计算总页数

if(isset($_GET['p'])){                                //设置第一页还是其他页
    $page = $_GET['p'];
}else{
    $page = 1;
}
$query_start = ($page - 1) * $page_size;             //计算每页开始的记录号
$querysql = "select * from userinfo limit $query_start, $page_size"; //使用 limit 获取记录
$queryset = mysql_query($querysql);                  //执行 SQL 语句
echo "<table border='1'><tr><th>id</th><th>姓名</th><th>性别</th><th>地址</th><th>邮件</th></tr>";
while($row = mysql_fetch_array($queryset, MYSQL_BOTH)){ //循环获取结果集
    echo "<tr><td>" . $row[0] . "</td>";
    echo "<td>" . $row[1] . "</td>";
    echo "<td>" . $row["sex"] . "</td>";
    echo "<td>" . $row[3] . "</td>";
    echo "<td>" . $row["email"] . "</td></tr>";
}
echo "</table><br>";
$pager = "共 $page_cnt 页 跳转至第 ";              //显示分布
if($page_cnt > 1){                                  //页面总数大于是则显示分布
    for($i=1; $i <= $page_cnt; $i++){
        if($page == $i){
            $pager .= "<a href='?p=$i ' ><b>$i</b></a> ";
        }else{
            $pager .= "<a href='?p=$i ' >$i</a> ";
        }
    }
    echo $pager . " 页";
}
mysql_close();
?>
```

分析:在上述程序中,分页显示结果集,每一页显示 10 条记录。首先获取结果集的总数,然后根据每一页显示的记录数计算总的页数,再由当前页数和记录数计算每一页的起止记录数,并使用 SELECT 语句的 LIMIT 子句实现,最后加粗显示当前页码。

18.2 管理 MySQL 数据库中的数据

在 Web 系统中,常常需要用户在浏览器上通过表单对数据库中的数据进行操作,如添加数据记录、更新数据记录和删除数据记录等。这一节将对用户在 HTML 表单上对数据进行操作,然后提交至服务器并使用 `mysql_query()` 函数执行 SQL 语句的方式操作数据进行详细讲解。

18.2.1 添加数据

在实际的应用中,用户常常直接在浏览器表单中输入相关数据,然后提交表单。服务器站在接收到用户提交的数据后采用 `mysql_query()` 函数执行相应的 `INSERT` 语句将用户输入的数据添加至数据库。

【示例 18-13】用户输入数据的 HTML 页面代码。代码如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>添加数据</title>
</head>

<body>
<form id="form1" name="form1" method="post" action="18.14.php">
  <table width="512" border="1">
    <tr>
      <td width="63">姓名: </td>
      <td width="433"><input name="username" type="text" id="username" size="10" /></td>
    </tr>
    <tr>
      <td>性别: </td>
      <td><input name="sex" type="text" id="sex" size="5" /></td>
    </tr>
    <tr>
      <td>地址: </td>
      <td><input type="text" name="address" id="address" size="50" /></td>
    </tr>
    <tr>
      <td>邮件: </td>
      <td><input name="email" type="text" id="email" size="20" /></td>
    </tr>
    <tr>
      <td>&nbsp;</td>
      <td><input type="submit" name="submit" id="submit" value="提交" /></td>
    </tr>
  </table>
</form>
</body>
</html>
```

分析：在上述代码中，设置了一个用户输入数据的表单。在用户单击【提交】按钮后，将表单提交至服务器进行处理。

【示例 18-14】服务器在接收到用户所提交的数据后，使用 `mysql_query()` 函数将用户所提交的数据添加至数据库。代码如下所示。

```
<?php
$username = $_POST['username'];           //获取表单数据
$sex = $_POST['sex'];
$address = $_POST['address'];
$email = $_POST['email'];
$ins_sql = "insert into userinfo('username', 'sex', 'address', 'email') values('$username', '$sex', '$address', '$email')"; //组成 SQL 语句
@mysql_connect("localhost","root","root") //连接数据库服务器
    or die("数据库连接失败！");
@mysql_select_db("mydb")
    or die("选择的数据库不存在或不可用!"); //选择数据库
$myquery = mysql_query($ins_sql);          //执行 SQL 语句
if($myquery){
    echo "插入数据成功！";
}else{
    echo "插入数据失败！";
}
mysql_close();                             //关闭数据库连接
?>
```

分析：在上述程序中，首先获取表单数据，再将表单数据组成 SQL 语句，然后使用 `mysql_query()` 函数执行该 SQL 语句，并根据返回结果输入不同的提示信息。

18.2.2 更新数据

在实际的应用中，用户常常需要对选择的数据进行修改。

【示例 18-15】如何在页面中浏览数据。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root") //连接数据库服务器
or die("数据库连接失败！");
@mysql_select_db("mydb")
or die("选择的数据库不存在或不可用!"); //选择数据库
mysql_query("set names gb2312");          //设置字符编码
$myquery = @mysql_query("select * from userinfo") //执行 SQL 语句
or die("SQL 语句执行失败!");
$page_size = 10;
$num_cnt = mysql_num_rows($myquery);      //获取所有记录
$page_cnt = ceil($num_cnt / $page_size);  //计算所有页数

if(isset($_GET['p'])){
    $page = $_GET['p'];
}else{
    $page = 1;
}
```

```

$query_start = ($page - 1) * $page_size;           //计算每页开始记录号
$querysql = "select * from userinfo limit $query_start, $page_size"; //组成 SQL 语句
$queryset = mysql_query($querysql);                //执行 SQL 语句
echo "<table border='1'><tr><th>id</th><th>姓名</th><th>性别</th><th>地址</th><th>邮件</th><th>操作</th></tr>";
while($row = mysql_fetch_array($queryset, MYSQL_BOTH)){ //逐行从数据集获取数据
echo "<tr><td>" . $row[0] . "</td>";
echo "<td>" . $row[1] . "</td>";
echo "<td>" . $row["sex"] . "</td>";
echo "<td>" . $row[3] . "</td>";
echo "<td>" . $row["email"] . "</td><td><a href='18.16.php?id=$row[0]'>修改</a></td></tr>";
}
echo "</table><br>";
$pager = "共 $page_cnt 页 跳转至第";
if($page_cnt > 1){
    for($i=1; $i <= $page_cnt; $i++){
        if($page == $i){
            $pager .= "<a href='?p=$i'><b>$i</b></a> ";
        }else{
            $pager .= "<a href='?p=$i'>$i</a> ";
        }
    }
    echo $pager . " 页";           //显示分页
}
mysql_close();                   //关闭连接
?>

```

分析：在上述程序中，以每页显示 10 行记录的方式进行浏览。用户单击每一行后的【修改】链接时，将跳转至修改页面。

【示例 18-16】根据 ID 号调用其信息并显示在 HTML 表单。代码如下所示。

```

<?php
if(isset($_GET['id'])){           //若没有参数 ID，则显示出错信息
    $id = $_GET['id'];
    @mysql_connect("localhost","root","root")
    or die("数据库连接失败！");   //连接数据库服务器
    @mysql_select_db("mydb")
    or die("选择的数据库不存在或不可用!"); //选择数据库
    mysql_query("set names gb2312"); //设置字符编码
    $sql = "select * from userinfo where userid ='$id'"; //组成 SQL 语句
    $myquery = @mysql_query($sql)
    or die("SQL 语句执行失败!"); //执行 SQL 语句
    $row = mysql_fetch_array($myquery, MYSQL_BOTH); //获取结果集
echo <<<Eof
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/
xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
<title>修改数据</title>
</head>
<form action="18.17.php" method="post" name="upinfo">
<table width="200" border="1">
<tr>

```

```

        <td>ID: </td>
        <td>{$row[0]}</td>
    </tr>
    <tr>
        <td>姓名: </td>
        <td><input name="username" type="text" value="{$row[1]}" size="10" /></td>
    </tr>
    <tr>
        <td>性别: </td>
        <td><input name="sex" type="text" value="{$row[2]}" size="5" /></td>
    </tr>
    <tr>
        <td>地址: </td>
        <td><input name="address" type="text" value="{$row[3]}" size="50" /></td>
    </tr>
    <tr>
        <td>邮件: </td>
        <td><input name="email" type="text" value="{$row[4]}" size="30" /></td>
    </tr>
    <tr>
        <td></td>
        <td><input name="submit" type="submit" value="提交" /></td>
    </tr>
</table>

</form>
</body>
</html>
EOF;
    mysql_close();
}
else{
    echo "ID 号错误, 请<a href='18.15.php'>浏览</a>";
}
?>

```

分析：在上述程序中，根据用户提交的 ID，从数据库获取信息，并显示在 HTML 表单上。用户可以自行修改其中信息，然后提交表单。

【示例 18-17】根据用户提交的表单中的信息进行修改。代码如下所示。

```

<?php
$userid = $_POST['userid'];           //获取表单 USERID
$username = $_POST['username'];       //获取表单姓名
$sex = $_POST['sex'];                 //获取表单性别
$address = $_POST['address'];         //获取表单地址
$email = $_POST['email'];             //获取表单邮件地址
$upd_sql = "update userinfo set username = '$username', sex = '$sex', address = '$address', email = '$email' where userid = '$userid'"; //组成 SQL 语句
@mysql_connect("localhost","root","root") //连接数据库服务器
    or die("数据库连接失败！");
@mysql_select_db("mydb") //选择数据库服务器
    or die("选择的数据库不存在或不可用!");
mysql_query("set names gb2312"); //设置字符编码
$myquery = mysql_query($upd_sql);     //执行 SQL 语句

```

```

if($myquery){
    echo "更新数据成功! ";
}
else{
    echo "更新数据失败! ";
}
echo "<a href='18.15.php'> 浏览</a>";
mysql_close();
?>

```

//根据返回值给出不同提示信息
//关闭数据库连接

分析：在上述程序中，根据用户提交的表单数据，组成 SQL 语句，再使用 `mysql_query()` 函数执行该语句完成资料的修改。

18.2.3 删除数据

在实际应用中，常常需要提供删除功能。通常使用的方式是让用户自行选择要删除的资料，再给出提示框，让用户确认是否真的删除该信息。

【示例 18-18】用户在选择要删除的资料时弹出相应的窗口。代码如下所示。

```

<script language="javascript">
function chk(id){
    if(confirm("确定要删除该资料? ")){
        window.location="18.19.php?id="+id;
    }
    else{
        return false;
    }
}
</script>
<?php
@mysql_connect("localhost","root","root")
or die("数据库连接失败! ");
@mysql_select_db("mydb")
or die("选择的数据库不存在或不可用!");
mysql_query("set names gb2312");
$myquery = @mysql_query("select * from userinfo");
$page_size = 10;
$num_cnt = mysql_num_rows($myquery);
$page_cnt = ceil($num_cnt / $page_size);

if(isset($_GET['p'])){
    $page = $_GET['p'];
}else{
    $page = 1;
}

$query_start = ($page - 1) * $page_size;
$querysql = "select * from userinfo limit $query_start, $page_size";
$queryset = mysql_query($querysql);
echo "<table border='1'><tr><th>id</th><th>姓名</th><th>性别</th><th>地址</th><th>邮件</th><th>操作</th></tr>";
while($row = mysql_fetch_array($queryset, MYSQL_BOTH)){
    echo "<tr><td>". $row[0] . "</td>";
}

```

//确认删除函数
//连接数据库服务器
//选择数据库
//设置字符编码
//执行 SQL 语句
//设置每页显示记录数
//获取总的记录数
//计算总的页数
//计算每页开始的记录号
//执行 SQL 语句

```

echo "<td>" . $row[1] . "</td>";
echo "<td>" . $row["sex"] . "</td>";
echo "<td>" . $row[3] . "</td>";
echo "<td>" . $row["email"] . "</td><td><a href='18.16.php?id=$row[0]'>修改</a>
<a href='javascript:void(0);' onclick='chk($row[0]);'>删除</a> </td></tr>";
}
echo "</table><br>";
$pager = "共 $page_cnt 页 跳转至第";
if($page_cnt > 1){
    for($i=1; $i <= $page_cnt; $i++){
        if($page == $i){
            $pager .= "<a href='?p=$i'><b>$i</b></a> ";
        }else{
            $pager .= "<a href='?p=$i'>$i</a> ";
        }
    }
    echo $pager . " 页";                                     //显示分页
}
mysql_close();                                             //关闭数据库连接
?>

```



分析：在上述程序中，用户在单击【删除】链接时，将弹出如图 18-2 所示的删除窗口，用户确认需要删除该数据，才跳转至删除页面进行删除。

图 18-2 删除窗口

【示例 18-19】采用 mysql_query 函数执行删除语句。代码如下所示。

```

<?php
$userid = $_GET['id'];                                     //获取要删除的 ID
$upd_sql = "delete from userinfo where userid='$userid'"; //组成 SQL 语句
@mysql_connect("localhost","root","root")                 //连接数据库服务器
    or die("数据库连接失败！");
@mysql_select_db("mydb")                                   //选择数据库
    or die("选择的数据库不存在或不可用!");
mysql_query("set names gb2312");                           //设置编码
$query = mysql_query($upd_sql);                             //执行 SQL 语句
if($query){                                                //根据删除结果显示不同信息
    echo "删除数据成功！";
}else{
    echo "删除数据失败！";
}
echo "<a href='18.18.php'> 浏览</a>";
mysql_close();                                             //关闭数据库连接
?>

```

分析：在上述程序中，首先获取需要删除的记录 ID，然后组成 SQL 语句，再使用 mysql_query() 函数执行 SQL 语句，根据返回的结果显示不同的提示信息。

18.3 获取数据库信息

在进行数据库操作时，有时可能因程序的需求需要获取数据库的相关信息。例如，数据

库列表、数据库中数据表的列表、数据表中某一列的类型等信息。本节将对在 PHP 中如何获取 MySQL 服务器的相关信息进行详细讲解。

18.3.1 获取数据库的信息

获取 MySQL 服务器的数据库列表信息，可使用 `mysql_list_dbs()` 函数。其语法格式如下所示。

```
resource mysql_list_dbs ([ resource $link_identifier ] )
```

其中，参数 `link_identifier` 为数据库连接标识符。若未设置该标识符，函数将使用上一个已打开的数据库连接；若没有可用的数据库连接，函数将尝试以无参数方式调用 `mysql_connect()` 函数建立数据库连接。函数返回一个结果指针，包含当前 MySQL 进程中所有可用的数据库。可使用 `mysql_fetch_array()` 类似的函数获取其结果集。

【示例 18-20】如何使用 `mysql_list_dbs` 函数列出 MySQL 服务器上的所有可用数据库。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
    or die("数据库连接失败！");           //连接数据库服务器
$dbms = mysql_list_dbs();                 //获取数据库列表
while($dbrow = mysql_fetch_array($dbms, MYSQL_BOTH)){ //获取结果集
    echo $dbrow[0]."<br>";
}
mysql_close();                             //关闭数据库连接
?>
```

分析：在上述程序中，首先连接数据库服务器，再使用 `mysql_list_dbs()` 函数获取服务器数据库列表，最后使用 `mysql_fetch_array()` 函数获取结果集，并将结果集显示出来。读者可以发现，使用该方式与在 MySQL 命令行使用“`show databases;`”命令的结果是一样的。

18.3.2 获取数据表的信息

对于显示某个数据库中的数据表的信息，首先需要选择数据库，再使用 `mysql_list_tables()` 函数获取数据表列表。其语法格式如下所示。

```
resource mysql_list_tables ( string $database [, resource $link_identifier ] )
```

其中，参数 `database` 为要获取数据表列表的数据库名，参数 `link_identifier` 为数据库连接标识符。若未设置该参数，函数将使用上一个已打开的数据库连接。若没有可用的数据库连接，函数将尝试以无参数方式调用 `mysql_connect()` 函数创建数据库连接；函数返回一个包含指定数据库的数据表列表的结果集，可使用类似 `mysql_fetch_array()` 函数获取该结果集，若失败则返回 `false`。

【示例 18-21】在 PHP 中使用 `mysql_list_tables` 函数获取指定数据库中的数据表列表。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
```



```

        or die("数据库连接失败！");           //连接数据库服务器
$tables = mysql_list_tables("mydb");           //获取数据库 mydb 的数据表列表
if($tables){                                   //根据结果显示信息
    while($tabrow = mysql_fetch_array($tables)){ //获取结果集
        echo $tabrow[0] . "<br>";
    }
}else{
    echo "获取数据表失败！";
}
mysql_close();                                //关闭数据库连接
?>

```

分析：在上述程序中，首先连接数据库服务器，再使用 `mysql_list_tables()` 函数获取指定数据库中的数据表列表。

读者可以发现，该方式与在 MySQL 命令行选择数据库后使用 “show tables;” 命令的结果一样。

18.3.3 获取数据表中列的信息

在应用中，有时需要获取数据表的列的信息，如数据表的列的数目、名称、长度等。

1. 获取列的数目

获取数据表的列的数目，可使用 `mysql_num_fields()` 函数。其语法格式如下所示。

```
int mysql_num_fields ( resource $result )
```

其中，参数 `result` 为使用 `mysql_query()` 函数所返回的结果集。函数返回结果集中列的数目。

注意：函数返回的是结果集中的列的数目。因此，若想获取数据表的列的数目，在使用 `mysql_query()` 函数执行 SQL 语句时，使用的查询语句应使用 “*” 代表所有的列。

【示例 18-22】在 PHP 中如何获取数据表中的列的总数。代码如下所示。

```

<?php
@mysql_connect("localhost","root","root")
    or die("数据库连接失败！");           //连接数据库服务器
@mysql_select_db("mydb")
    or die("数据库选择失败！");
$myquery = @mysql_query("select * from userinfo")
    or die("SQL 语句执行失败！");
echo "数据表 userinfo 列的数目为： " . mysql_num_fields($myquery);
mysql_close();
?>

```

分析：在上述程序中，首先连接数据库服务器，然后使用 `mysql_query()` 函数执行一个包含所有列的 SQL 语句，再根据返回的结果集使用 `mysql_num_fields()` 函数获取结果集中的列的数目。

2. 获取列的名称

获取数据表的列的名称，其实在前面使用 `mysql_fetch_array()` 函数获取结果集时，若采

用参数 `MYSQL_ASSOC` 或 `MYSQL_BOTH` 获取的结果集数组中键名就是列的名称。除此以外，PHP 还提供了一个专门获取列名称的函数 `mysql_field_name()`，其语法格式如下所示。

```
string mysql_field_name ( resource $result , int $field_index )
```

其中，参数 `result` 为使用 `mysql_query()` 函数执行 SQL 语句所返回的结果集，参数 `field_index` 为列的偏移量。函数返回结果集中指定列的列名。

注意：偏移量是从 0 开始，并且所返回的列名与使用 `mysql_fetch_array()` 函数一样，区分大小写。

【示例 18-23】如何获取数据表所有列的名称。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
    or die("数据库连接失败！");           //连接数据库服务器
@mysql_select_db("mydb")
    or die("数据库选择失败！");           //选择数据库
$myquery = @mysql_query("select * from userinfo")
    or die("SQL 语句执行失败！");         //执行 SQL 语句
$fieldcnt = mysql_num_fields($myquery);    //获取列的总数
for($i=0; $i<$fieldcnt; $i++){
    echo mysql_field_name($myquery, $i). "<br>"; //显示列的名称
}
mysql_close();                             //关闭数据库连接
?>
```

分析：在上述程序中，首先获取返回结果集中列的总数，再使用 `mysql_field_name()` 函数循环获取每一列的名称。

3. 获取列的长度

获取数据表中列的长度，可使用 `mysql_field_len()` 函数，其语法格式如下所示。

```
int mysql_field_len ( resource $result , int $field_offset )
```

其中，参数 `result` 为使用 `mysql_query()` 函数执行 SQL 语句所返回的结果集，参数 `field_index` 为列的偏移量。函数返回结果集中指定列的长度。

【示例 18-24】如何获取数据表中所有列的长度。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
    or die("数据库连接失败！");           //连接数据库服务器
@mysql_select_db("mydb")
    or die("数据库选择失败！");           //选择数据库
$myquery = @mysql_query("select * from userinfo")
    or die("SQL 语句执行失败！");         //执行 SQL 语句
$fieldcnt = mysql_num_fields($myquery);    //获取列的总数
for($i=0; $i<$fieldcnt; $i++){
    echo mysql_field_len($myquery, $i). "<br>"; //显示列的长度
}
mysql_close();                             //关闭数据库连接
?>
```

分析：在上述程序中，首先获取返回结果集中列的总数，再使用 `mysql_field_len()` 函数

循环获取每一列的长度。

4. 获取列的类型

获取数据表的类型可使用 `mysql_field_type()` 函数，其语法格式如下所示。

```
string mysql_field_type ( resource $result , int $field_offset )
```

其中，参数 `result` 为使用 `mysql_query()` 函数执行 SQL 语句所返回的结果集，参数 `field_index` 为列的偏移量。函数返回结果集中指定列的类型。

【示例 18-25】如何获取数据表中所有列的类型。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
    or die("数据库连接失败！");           //连接数据库服务器
@mysql_select_db("mydb")
    or die("数据库选择失败！");           //选择数据库
$myquery = @mysql_query("select * from userinfo")
    or die("SQL 语句执行失败！");         //执行 SQL 语句
$fieldcnt = mysql_num_fields($myquery);    //获取列的总数
for($i=0; $i<$fieldcnt; $i++){
    echo mysql_field_type($myquery, $i)."<br>"; //显示列的长度
}
mysql_close();                             //关闭数据库连接
?>
```

分析：在上述程序中，首先获取返回结果集中列的总数，再使用 `mysql_field_len()` 函数循环获取每一列的类型。

5. 获取数据表完整结构信息

前面介绍了使用各种获取列信息的函数获取单个的信息，有时需要直接获取整个数据表的所有信息。综合使用以上几种函数即可实现获取数据表的所有表结构信息。

【示例 18-26】在应用中如何获取整个数据表的所有结构信息。代码如下所示。

```
<?php
@mysql_connect("localhost","root","root")
    or die("数据库连接失败！");           //连接数据库服务器
@mysql_select_db("mydb")
    or die("数据库选择失败！");           //选择数据库
$myquery = @mysql_query("select * from userinfo")
    or die("SQL 语句执行失败！");         //执行 SQL 语句
$fieldcnt = mysql_num_fields($myquery);    //获取数据表列的总数
echo "<table border=1><tr><th>列名称</th><th>长度</th><th>类型</th><th>标志</th></tr>";
for($i=0; $i<$fieldcnt; $i++){
    $name = mysql_field_name($myquery, $i); //获取列的名称
    $len = mysql_field_len($myquery, $i);   //获取列的长度
    $type = mysql_field_type($myquery, $i); //获取列的类型
    $flags = mysql_field_flags($myquery, $i); //获取表的标志
    echo "<tr><td>$name</td><td>$len</td><td>$type</td><td>$flags</td></tr>";
}
echo "</table>";
mysql_close();                             //关闭数据库连接
?>
```

分析：在上述程序中，首先获取数据表的结果集，然后根据结果集中列的数目循环获取列的名称、长度、类型、标志等信息。列的标志就是该列是否为主键、是否为空等信息。

18.4 PHP 操作 SQL Server 数据库

在实际的应用中，有时需要使用 SQL Server 数据库存储数据。这时就需要对 SQL Server 数据库进行操作，如连接关闭数据库、执行 SQL 语句等。实际上 PHP 所提供的操作 SQL Server 数据库的函数用法大多与连接 MySQL 数据库的函数类似。这一节将对在 PHP 如何对 SQL Server 数据库进行相关操作进行简要讲解。

18.4.1 连接和关闭 SQL Server 数据库

要能够使用 PHP 连接 SQL Server 数据库，首先需要确认 Apache 服务器安装有 SQL Server 数据库的 mssql 扩展，找到当前 Apache 服务器所加载的 PHP 配置文件 php.ini，打开该文件，找如下内容。

```
;extension=php_mssql.dll
```

将其前面的“;”去掉，然后重新启动 Apache 服务器。重新启动 Apache 服务器后使用以下语句查看状态信息。

```
<?php
phpinfo();
?>
```

若看到类似如图 18-3 所示的信息，说明 SQL Server 的 mssql 扩展安装成功，就可以进行数据库的操作了。

使用 PHP 连接 SQL Server 数据库，需要使用 mssql_connect() 函数。其语法格式如下所示。

```
resource mssql_connect ([ string $servername [, string $username [, string $password [, bool $new_link ]]])
```

其中，参数 servername 为要连接的 SQL Server 数据库的地址，参数 username 为用于连接数据库服务器的用户名，参数 password 为连接数据库服务器所使用的密码，参数 new_link 用于标识是否每一次使用该函数总是打开新的连接。函数返回一个连接数据库的资源对象。

【示例 18-27】如何在 PHP 中连接 SQL Server 数据库服务器。代码如下所示。

```
<?php
$conn = @mssql_connect("localhost","sa","sa")
        or die("无法连接 SQL Server 数据库服务器!");           //连接本地数据库服务器
?>
```

分析：在上述程序中，使用 mssql_connect() 函数连接本地 SQL Server 数据库服务器。若未能连接数据库服务器，将显示错误信息。

关闭与 SQL Server 数据库服务器的连接可直接使用 mssql_close() 函数，其语法格式如下所示。

```
bool mssql_close ( [ resource $link_identifier ] )
```

其中，参数 `link_identifier` 为数据库连接标识符。通常在使用数据库连接的脚本文件执行结束，会自动关闭数据库连接。但是为了更好地节省资源，应在程序结束使用数据库连接时，手动使用该函数关闭数据库连接。

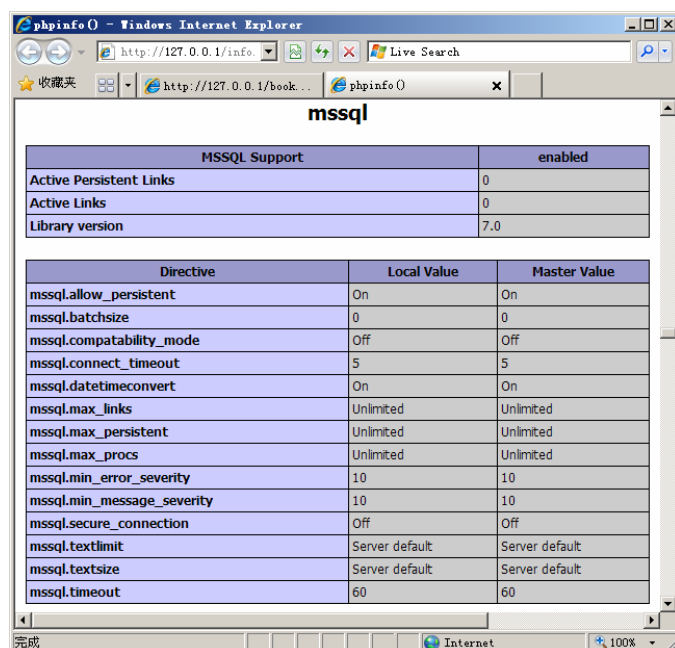


图 18-3 mssql 扩展信息

【示例 18-28】在 PHP 脚本中手动使用 `mssql_close` 函数关闭数据库连接。代码如下所示。

```
<?php
$conn = @mssql_connect()
    or die("无法连接 SQL Server 数据库服务器!");           //连接本地数据库服务器
mssql_close($conn);                                         //关闭数据库连接
```

分析：在上述程序中，直接在 PHP 脚本中使用 `mssql_close()` 函数关闭与数据库的连接。

18.4.2 执行 SQL 语句

在 PHP 中，对 SQL Server 数据库进行相关操作，都是采用 `mssql_query()` 函数执行相关 SQL 语句进行的。其语法格式如下所示。

```
mixed mssql_query ( string $query [, resource $link_identifier [, int $batch_size ]] )
```

其中，参数 `query` 为要执行的 SQL 语句，参数 `link_identifier` 为数据库连接标识符。若没有设置该参数，函数将使用上一个打开的数据库连接；若没有已打开的数据库连接，该函数将尝试调用 `mssql_connect()` 函数建立数据库连接并使用该连接。参数 `batch_size` 为返回的记录缓冲区大小。函数执行 SQL 语句成功返回一个包含结果集的资源对象，执行不成功则返回 `false`。

在执行 SQL 语句前也像连接 MySQL 数据库一样，需要选择进行操作的数据库，只是选择数据库的函数为 `mssql_select_db()`。其用法与 `mysql_select_db()` 函数用法一样。

【示例 18-29】在 PHP 中如何执行 SQL 语句操作 SQL Server 数据库。代码如下所示。

```
<?php
$conn = @mssql_connect("localhost","sa","sa")
        or die("无法连接 SQL Server 数据库服务器!");           //连接数据库服务器
$dbname = mssql_select_db("mydb", $conn)
        or die("选择数据库失败!");                               //选择数据库
$query = mssql_query("select * from userinfo")
        or die("执行 SQL 语句失败!");                             //执行 SQL 语句
mssql_close($conn);                                              //关闭数据库连接
?>
```

分析：在上述程序中，首先连接 SQL Server 数据库服务器，然后选择要进行操作的数据，再执行 SQL 语句，最后关闭与 SQL Server 数据库服务器的连接。

18.4.3 获取结果集

在使用 `mssql_query()` 函数执行了 SQL 查询语句后，该函数返回一个结果集。对于该结果集，可使用类似 `mssql_fetch_array()` 函数获取。其语法格式如下所示。

```
array mssql_fetch_array ( resource $result [, int $result_type ] )
```

其中，参数 `result` 为使用 `mssql_query()` 函数执行 SQL 语句所返回的结果集。参数 `result_type` 的取值有以下几种。

- ❑ `MSSQL_ASSOC`：返回的结果数组以列名作为数组的键名。
- ❑ `MSSQL_NUM`：返回的结果数组以偏移量作为数组的键名，偏移量从 0 开始。
- ❑ `MSSQL_BOTH`：返回的结果数组中的键名包含前面两种方式。在未设置该参数时默认为该值。

函数返回一个获取的数组，若没有更多行则返回 `false`。

【示例 18-30】使用 `mssql_query` 函数获取结果集中的值，并使用其他相关函数列出数据表的结构信息。代码如下所示。

```
<?php
$conn = @mssql_connect("localhost","sa","sa")
        or die("数据库连接失败！");                               //连接数据库服务器
@mssql_select_db("mydb", $conn)
        or die("选择的数据库不存在或不可用!");                   //选择数据库
$query = @mssql_query("select * from userinfo", $conn)
        or die("SQL 语句执行失败!");                             //执行 SQL 语句
echo "<table border='1'><tr><th>id</th><th>姓名</th><th>性别</th><th>地址</th><th>邮件</th></tr>";
while($row = mssql_fetch_array($query, MSSQL_BOTH)){              //获取结果集的每一行
    echo "<tr><td>" . $row[0] . "</td>";
    echo "<td>" . $row[1] . "</td>";
    echo "<td>" . $row[2] . "</td>";
    echo "<td>" . $row[3] . "</td>";
```

```

        echo "<td>" . $row[4] . "</td></tr>";
    }
    echo "</table><br>";
    mssql_close(); //关闭数据库连接
?>

```

分析：在上述程序中，首先获取数据集中总的记录数，再使用 `mssql_fetch_array()` 函数循环获取结果集中的行，并将其显示出来，同时使用其他函数获取数据表的结构信息。

18.5 PHP 操作 Access 数据库

在实际的应用中，有时需要使用 Microsoft 的 Access 数据库。在 PHP 中使用该数据库通常有 3 种方式：创建系统数据源，再采用 ODBC 方式进行连接；直接使用 PHP 的 ODBC 函数进行操作；使用微软所提供的 ADODB 的方式进行操作。这一节将对直接使用 ODBC 函数操作 Access 数据库进行讲解。

18.5.1 连接和关闭 Access

在 PHP 中操作 Access 数据库，首先需要连接该数据库。连接 Access 数据库使用 `odbc_connect()` 函数，其语法格式如下所示。

```
resource odbc_connect ( string $dsn , string $user , string $password [, int $cursor_type ] )
```

其中，参数 `dsn` 为连接 Access 数据库的数据源字符串，其中包含驱动程序及 Access 数据库文件的位置；参数 `user` 为连接 Access 数据库的用户名；参数 `password` 为连接 Access 数据库的密码；参数 `cursor_type` 设置数据库游标，通常可省略，其取值有以下几种。

- ❑ `SQL_CUR_USE_IF_NEEDED`。
- ❑ `SQL_CUR_USE_ODBC`。
- ❑ `SQL_CUR_USE_DRIVER`。
- ❑ `SQL_CUR_DEFAULT`。

在程序中，当使用复杂的资料存取时可能会出现类似以下错误信息的字符串，此时若将参数 `cursor_type` 的值设为 `SQL_CUR_USE_ODBC` 就可以避免该错误。

```
Cannot open a cursor on a stored procedure that has anything other than a single select statement in it
```

函数连接成功返回一个 ODBC 连接资源 ID，失败则返回 `false`。

【示例 18-31】在 PHP 脚本中连接本地的 Access 数据库。代码如下所示。

```

<?php
$dsn = "DRIVER=Microsoft Access Driver (*.mdb);DBQ=".realpath("mydb.mdb"); //数据源
$conn = @odbc_connect($dsn, "", "", SQL_CUR_USE_ODBC)
        or die("连接 Access 数据库失败!"); //连接数据库
?>

```

分析：在上述程序中，采用 `odbc_connect()` 函数连接当前目录下的 Access 数据库

mydb.mdb。读者在进行操作时需确认在当前目录下是否有一个名为 mydb.mdb 的 Access 数据库。

使用完数据库连接后，应使用 `odbc_close()` 函数将数据库连接关闭。其语法格式如下所示。

```
void odbc_close ( resource $connection_id )
```

其中，参数 `connect_id` 为采用 `odbc_connect()` 函数连接数据库所返回的资源 ID。通常在使用数据库连接的脚本文件结束执行时，自动关闭数据库连接。但为了节省资源，推荐在每次使用完数据库连接后，手动使用该函数关闭数据库连接。

注意：在使用事务处理时，无法使用该函数关闭 ODBC 数据库连接。

【示例 18-32】在 PHP 脚本中如何关闭数据库连接。代码如下所示。

```
<?php
$dsn = "DRIVER=Microsoft Access Driver (*.mdb);DBQ=".realpath("mydb.mdb"); //设置 DSN
$conn = @odbc_connect($dsn, "", "", SQL_CUR_USE_ODBC)
        or die("连接 Access 数据库失败!"); //连接数据库
odbc_close($conn); //关闭数据库连接
?>
```

分析：在上述程序中，首先通过 `odbc_connect()` 函数建立数据库连接，在使用结束后，手动使用 `odbc_close()` 函数关闭该数据库连接。

18.5.2 执行 SQL 语句

在实际的应用中，需要对 Access 数据库中的数据进行操作，这些操作大多都是以执行 SQL 语句的方式实现的。执行 SQL 语句使用 `odbc_exec()` 函数，其语法格式如下所示。

```
resource odbc_exec ( resource $connection_id , string $query_string )
```

其中，参数 `connection_id` 为数据库连接函数 `odbc_connect()` 所返回的连接资源 ID，参数 `query_string` 为要执行的 SQL 语句。函数如果执行 SQL 语句成功则返回 ODBC 结果集标识符，失败则返回 `false`。

【示例 18-33】在 PHP 脚本中如何执行 SQL 语句。代码如下所示。

```
<?php
$dsn = "DRIVER=Microsoft Access Driver (*.mdb);DBQ=".realpath("mydb.mdb");//设置 DSN
$conn = @odbc_connect($dsn, "", "", SQL_CUR_USE_ODBC)
        or die("连接 Access 数据库失败!"); //连接数据库
$sql = "select * from userinfo"; //设置 SQL 语句
$query = @odbc_exec($conn, $sql)
        or die("执行 SQL 语句失败!"); //执行 SQL 语句
odbc_close($conn); //关闭数据库连接
?>
```

分析：在上述程序中，首先通过 ODBC 连接数据库，然后执行 SQL 语句，最后手动关闭数据库连接。

18.5.3 获取结果集

对数据库进行操作，最常用的就是从数据库读取数据。读取数据使用 `odbc_exec()` 函数执行 SQL 查询语句，该函数返回包含查询结果的结果集。可使用 `odbc_fetch_array()` 等函数获取结果集中的内容，其语法格式如下所示。

```
array odbc_fetch_array ( resource $result [, int $rownumber ] )
```

其中，参数 `result` 为函数 `odbc_exec()` 执行 SQL 查询语句后所返回的结果集，参数 `rownumber` 为指定获取哪一行数据。函数返回一个包含结果集的关联数组。

【示例 18-34】在 PHP 脚本中如何获取结果集中的数据。代码如下所示。

```
<?php
$dsn = "DRIVER=Microsoft Access Driver (*.mdb);DBQ=".realpath("mydb.mdb"); //设置 DSN
$conn = @odbc_connect($dsn, "", "", SQL_CUR_USE_ODBC) //连接数据库
        or die("连接 Access 数据库失败!");
$sql = "select * from userinfo"; //设置 SQL 语句
$query = @odbc_exec($conn, $sql) //执行 SQL 语句
        or die("执行 SQL 语句失败!");
echo "<table border='1'><tr><th>id</th><th>姓名</th><th>性别</th><th>地址</th><th>邮件</th></tr>";
while($row = odbc_fetch_array($query)){ //获取结果集
    echo "<tr><td>". $row["userid"] . "</td>";
    echo "<td>". $row["username"] . "</td>";
    echo "<td>". $row["sex"] . "</td>";
    echo "<td>". $row["address"] . "</td>";
    echo "<td>". $row["email"] . "</td></tr>";
}
echo "</table>";
odbc_close($conn); //关闭数据库连接
?>
```

分析：在上述程序中，首先通过 ODBC 连接 Access 数据库，然后执行 SQL 查询语句，再使用 `odbc_fetch_array()` 函数获取结果集中的每一行，并将其显示出来，最后关闭数据库连接。

18.6 本章实例

在 PHP 应用中，常常需要对数据库进行操作，如从数据库获取数据、更新、删除数据库等，通常将所有与数据库进行的操作封装成一个数据库类。

【示例 18-35】一个数据库类。代码如下所示。

```
<?php
$db_config["hostname"] = "127.0.0.1"; //服务器地址
$db_config["username"] = "root"; //数据库用户名
$db_config["password"] = "root"; //数据库密码
$db_config["database"] = "cms"; //数据库名称
```

```

$db_config["charset"] = "gb2312"; //编码

$db = new db(); //实例数据库类
$row = $db->row_select('user', 'type=user');

class db {
    public $connection_id = "";
    public $pconnect = 0;
    public $shutdown_queries = array();
    public $queries = array();
    public $query_id = "";
    public $query_count = 0;
    public $record_row = array();
    public $failed = 0;
    public $halt = "";
    public $query_log = array();

    function __construct($db_config){ //构造函数
        if ($this->pconnect){
            $this->connection_id = mysql_pconnect($db_config["hostname"], $db_config["username"],
$db_config["password"]);
        }else{
            $this->connection_id = mysql_connect($db_config["hostname"], $db_config["username"],
$db_config["password"]);
        }
        if ( ! $this->connection_id ){
            $this->halt("Can not connect MySQL Server");
        }
        if ( ! @mysql_select_db($db_config["database"], $this->connection_id) ){
            $this->halt("Can not connect MySQL Database");
        }
        if ($db_config["charset"]) {
            @mysql_unbuffered_query("SET NAMES ".$db_config["charset"]."" );
        }
        return true;
    }

    function query($query_id, $query_type='mysql_query'){ //发送 SQL 查询，并返回结果集
        $this->query_id = $query_type($query_id, $this->connection_id);
        $this->queries[ ] = $query_id;
        if ( ! $this->query_id ) {
            $this->halt("查询失败:\n$query_id");
        }
        $this->query_count++;
        $this->query_log[ ] = $str;
        return $this->query_id;
    }

    function fetch_array($sql = ""){ //从结果集中取得一行作为关联数组
        if ($sql == "") $sql = $this->query_id;
        $this->record_row = @mysql_fetch_array($sql, MYSQL_ASSOC);
        return $this->record_row;
    }

    function affected_rows() { //取得结果集中行的数目
        return @mysql_affected_rows($this->connection_id);
    }
}

```

```

function num_rows($query_id="") { //取得结果集中行的数目，仅对 SELECT 语句有效
    if ($query_id == "") $query_id = $this->query_id;
    return @mysql_num_rows($query_id);
}

function get_errno(){ //返回上一个 MySQL 操作中的错误信息的数字编码
    $this->errno = @mysql_errno($this->connection_id);
    return $this->errno;
}

function free_result($query_id=""){ //释放结果内存
    if ($query_id == "") $query_id = $this->query_id;
    @mysql_free_result($query_id);
}

//关闭 MySQL 连接
function close_db(){
    if ( $this->connection_id ) return @mysql_close( $this->connection_id );
}

//从结果集中取得列信息并作为对象返回，取得所有字段
function get_result_fields($query_id=""){
    if ($query_id == "") $query_id = $this->query_id;
    while ($field = mysql_fetch_field($query_id)) {
        $fields[ ] = $field;
    }
    return $fields;
}

//错误提示
function halt($the_error=""){
    $message = $the_error."<br/>\r\n";
    $message.= $this->get_errno() . "<br/>\r\n";
    $sql = "INSERT INTO 'db_error' (pagename, errstr, timer) VALUES('".$_SERVER["PHP_SELF"]."',
    "'.addslashes($message)."', ".time().")";
    @mysql_unbuffered_query($sql);
    if (DEBUG==true){
        echo "<html><head><title>MySQL 数据库错误</title>";
        echo "<style type='text/css'><!--.error { font: 11px tahoma, verdana, arial, sans-serif,
simsum; }--></style></head>\r\n";
        echo "<body>\r\n";
        echo "<blockquote>\r\n";
        echo "<textarea class='error' rows='15' cols='100' wrap='on' > " . htmlspecialchars
($message) . "</textarea>\r\n";
        echo "</blockquote>\r\n</body></html>";
        exit;
    }
}

function sql_select($tname, $where="", $limit=0, $fields="*", $orderby="id", $sort="DESC"){
    $sql = "SELECT ".$fields." FROM ".$tname." ".$($where?" WHERE ".$where:".") .
    "ORDER BY ".$orderby." ".$sort."($limit ? " limit ".$limit:".")";
    return $sql;
}

```

```

//插入数据
function sql_insert($tbname, $row){
    foreach ($row as $key=>$value) {
        $sqlfield .= $key." ";
        $sqlvalue .= "".$value." ";
    }
    return "INSERT INTO ".$tbname." (" .substr($sqlfield, 0, -1).") VALUES (" .substr($sqlvalue,
0, -1).")";
}

//更新数据
function sql_update($tbname, $row, $where){
    foreach ($row as $key=>$value) {
        $sqlud .= $key."= "".$value." ";
    }
    return "UPDATE ".$tbname." SET ".substr($sqlud, 0, -1)." WHERE ".$where;
}

//删除数据
function sql_delete($tbname, $where){
    return "DELETE FROM ".$tbname." WHERE ".$where;
}

//新增加一条记录
function row_insert($tbname, $row){
    $sql = $this->sql_insert($tbname, $row);
    return $this->query_unbuffered($sql);
}

//更新指定记录
function row_update($tbname, $row, $where){
    $sql = $this->sql_update($tbname, $row, $where);
    return $this->query_unbuffered($sql);
}

//删除满足条件的记录
function row_delete($tbname, $where){
    $sql = $this->sql_delete($tbname, $where);
    return $this->query_unbuffered($sql);
}

/* 根据条件查询, 返回所有记录
* $tbname 表名, $where 查询条件, $limit 返回记录, $fields 返回字段
*/
function row_select($tbname, $where="", $limit=0, $fields="", $orderby="id", $sort="DESC"){
    $sql = $this->sql_select($tbname, $where, $limit, $fields, $orderby, $sort);
    return $this->row_query($sql);
}

//详细显示一条记录
function row_select_one($tbname, $where, $fields="", $orderby="id"){
    $sql = $this->sql_select($tbname, $where, 1, $fields, $orderby);
    return $this->row_query_one($sql);
}

function row_query($sql){
    $rs = $this->query($sql);

```

```
$rs_num = $this->num_rows($rs);  
$rows = array();  
for($i=0; $i<$rs_num; $i++){  
    $rows[ ] = $this->fetch_array($rs);  
}  
$this->free_result($rs);  
return $rows;  
}  
  
function row_query_one($sql){  
    $rs = $this->query($sql);  
    $row = $this->fetch_array($rs);  
    $this->free_result($rs);  
    return $row;  
}  
?>
```

分析：在上述程序中，将与 MySQL 数据库进行的基本操作封装在一个数据库类中。在使用时，只需要实例化该类，然后调用该类所提供的数据库操作方法即可实现与数据库的存储。

18.7 小结

本章主要介绍了 PHP 与几种常用的数据库之间连接、选择、取数据表数据等操作。在实际的应用中，PHP 多与 MySQL 结合使用。对于一些特殊的情况，有可能会使用到其他几种常用的数据库，读者可根据实际的需要选择使用的数据库。