第11章　使用PHP从Web访问MySQL数据库

在我们前面使用PHP的过程中，使用了普通文件来存储与检索数据。第2章中介绍这种文件时，我们提到了在Web应用中使用关系数据系统可以使得这些存储和检索操作变得更容易、更安全、更有效。现在，在已经使用了MySQL创建数据库后，我们可以开始通过基于Web的前台来连接该数据库。

在本章中，我们将介绍如何使用PHP从Web访问Book-O-Rama数据库。我们将学习到如何从数据库读取数据和将数据写入数据库，以及如何过滤潜在的、可能造成麻烦的输入数据。

在本章中，我们将主要介绍以下内容：

- Web数据库架构的工作原理
- 通过Web查询数据库的基本步骤
- 建立数据库连接
- 获取关于可用数据库的信息
- 选择要使用的数据库
- 查询数据库
- 检索查询结果
- 与数据库断开连接
- 在数据库中插入新信息
- 使用prepared语句
- 使用PHP与数据库交互的其他接口
- 使用常规的数据库接口：PEAR MDB2

11.1　Web数据库架构的工作原理

在第8章中，我们简单地介绍了数据库架构的工作原理，作为一个简单的回顾，在这里，我们给出如下所示的基本步骤：

1) 一个用户的浏览器发出一个HTTP请求，请求特定的Web页面。例如，用户中能使用一个HTML表单请求搜索Book-O-Rama数据库中所有由Michael Morgan编写的书籍。该搜索结果页面为results.php。

2) Web服务器接收到对results.php页面的请求后，检索该文件，并将其传递给PHP引擎处理。

3) PHP引擎开始解析脚本。脚本主要包括了连接到数据库和执行查询的命令（执行对书籍的搜索）。PHP启动了对MySQL服务器的连接并向该服务器发送适当的查询。

4) MySQL服务器接收到数据库查询的请求，开始处理这个查询，并将查询结果（一个书籍的列表）返回给PHP引擎。

5) PHP引擎完成了脚本的运行后（其中包括以HTML格式表示经过处理后的查询结果），然后将该HTML返回给Web服务器。

6) Web服务器再将HTML返回给客户端浏览器，用户就可以看到所要求查询的书籍。现在，我们已经有了一个MySQL数据库，因此就可以编写PHP代码来执行上述步骤。我们从搜索表单开始。这个搜索表单是一个普通的HTML表单。代码如程序清单11-1所示。

程序清单11-1　search.html——Book-O-Rama的数据库搜索页

```
<html>
<head>
  <title>Book-O-Rama Catalog Search</title>
</head>

<body>
  <h1>Book-O-Rama Catalog Search</h1>

  <form action="results.php" method="post">
    Choose Search Type:<br />
    <select name="searchtype">
      <option value="author">Author</option>
      <option value="title">Title</option>
      <option value="isbn">ISBN</option>
    </select>
    <br />
    Enter Search Term:<br />
    <input name="searchterm" type="text" size="40" />
    <br />
    <input type="submit" name="submit" value="Search" />
  </form>

</body>
</html>
```

这是一个非常直观的HTML表单。其输出结果如图11-1所示。

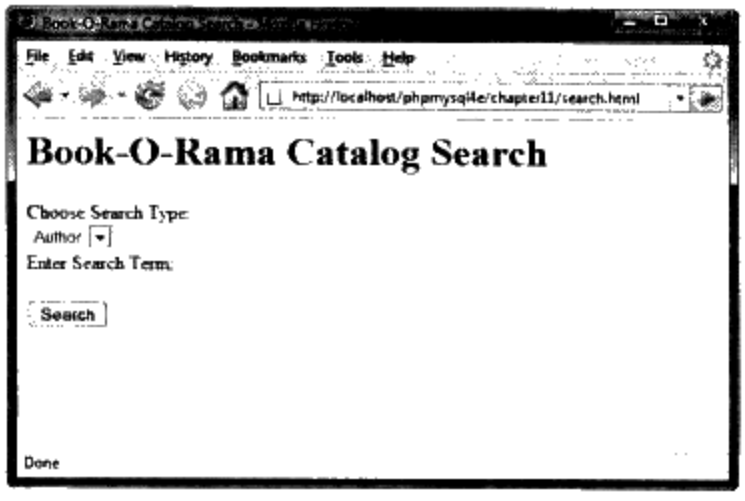

图11-1　搜索表单非常简单，可以根据书籍的标题、作者或ISBN号进行搜索

当点击“Search”（搜索）按钮时，将调用results.php脚本。程序清单11-2给出完整代码。在本章的后续内容中，我们将讨论该脚本的功能及其工作原理。

程序清单11-2 results.php——从MySQL数据库获取并格式化搜索结果，以便显示结果

```
<html>
<head>
  <title>Book-O-Rama Search Results</title>
</head>
<body>
<h1>Book-O-Rama Search Results</h1>
<?php
  // create short variable names
  $searchtype=$_POST['searchtype'];
  $searchterm=trim($_POST['searchterm']);
  if (!$searchtype || !$searchterm) {
    echo 'You have not entered search details. Please go back and try again.';
    exit;
  }

  if (!get_magic_quotes_gpc()){
    $searchtype = addslashes($searchtype);
    $searchterm = addslashes($searchterm);
  }

@ $db = new mysqli('localhost', 'bookorama', 'bookorama123', 'books');

if (mysqli_connect_errno()) {
  echo 'Error: Could not connect to database. Please try again later.';
  exit;
}

$query = "select * from books where ".$searchtype." like '%".$searchterm."%'";
$result = $db->query($query);

$num_results = $result->num_rows;

echo "<p>Number of books found: ".$num_results."</p>";

for ($i=0; $i <$num_results; $i++) {
  $row = $result->fetch_assoc();
  echo "<p><strong>".($i+1).". Title: ";
  echo htmlspecialchars(stripslashes($row['title']));
  echo "</strong><br />Author: ";
  echo stripslashes($row['author']);
  echo "<br />ISBN: ";
  echo stripslashes($row['isbn']);
  echo "<br />Price: ";
```

```
        echo stripslashes($row['price']);
        echo "</p>";
    }

    $result->free();
    $db->close();

?>
</body>
</html>
```

请注意，以上脚本允许输入MySQL通配符%和_。这个功能对用户来说是非常有用的。如果它会给应用程序带来问题，你就必须对字符转义。

图11-2给出了该脚本执行搜索操作以后的结果。

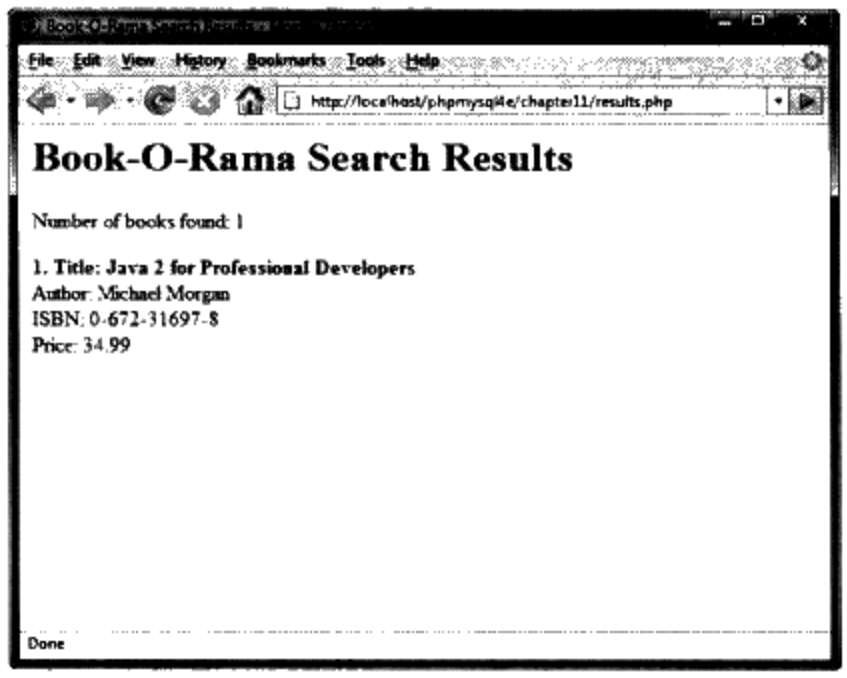

图11-2　在Web页面中显示了运行result.php脚本搜索到的关于Java的书籍

11.2　从Web查询数据库的基本步骤

在任何用于从Web访问数据库的脚本中，都应该遵循以下这些基本步骤：

1) 检查并过滤来自用户的数据。
2) 建立一个到适当数据库的连接。
3) 查询数据库。
4) 获取查询结果。
5) 将结果显示给用户。

这些是我们在脚本results.php中已经遵循的步骤，在接下来的内容中，我们将逐个详细介绍。

11.2.1　检查与过滤用户输入数据

首先，我们将过滤用户可能在其搜索条件的起始或结束位置不小心输入的空白字符。过滤操

作是通过对$_POST['searchterm']变量定义简短名称变量的值应用trim()函数来实现的。

```
$searchterm=trim($_POST['searchterm']);
```

接下来，我们将验证用户已经输入搜索条件并选择了搜索类型。请注意，我们是在过滤了$searchterm两端的空白字符之后再检查用户是否已经输入一个查询条件。如果将这几行代码的顺序颠倒，我们将遇到用户的搜索条件不为空时的情况，因此这不会产生一个错误信息，但是如果都是空白字符，trim()函数将删除它们：

```
if (!$searchtype || !$searchterm) {
   echo 'You have not entered search details. Please go back and try again.';
   exit;
}
```

我们已经检查了$searchtype变量，尽管在这个例子中，它来自一个HTML SELECT。你可能要问，为什么我们要这么麻烦来检查这些必须由用户输入的数据。请记住，可能有多个接口可以连接到数据库。例如，Amazon（亚马逊）的许多会员会都使用他们自己的界面。同样，因为用户从不同的界面进入，这样就可能会导致安全问题。

当准备使用用户输入的任何数据时，也要适当地过滤一些控制字符。回顾前面所介绍的，在第4章中，我们介绍了addslashes()函数、stripslashes()函数和get_magic_quotes_gpc()函数。当将用户输入数据提交到一个数据库（例如MySQL）时，必须转义数据。

在这个例子中，我们检查了get_magic_quotes_gpc()函数的返回值。它告诉我们是否已经自动完成了引号。如果还没有，可以使用addslashes()函数来过滤数据：

```
if (!get_magic_quotes_gpc()) {
  $searchtype = addslashes($searchtype);
  $searchterm = addslashes($searchterm);
}
```

我们也可以对来自数据库的数据调用stripslashes()。如果魔术引号特性没有打开，从数据库出来的数据将包含反斜杠，因此必须过滤这些反斜杠。

我们使用函数htmlspecialchars()对HTML中的特殊意义字符进行编码。我们当前的测试不包含下列任何符号：与号&、小号（<）、大于号（>）或双引号（"），但是许多美观的图书标题包含&号。通过使用该函数，可以清除将来可能发生的错误。

11.2.2　建立一个连接

PHP为连接MySQL提供了函数库。这个函数库是mysqli（*i*表示改进）。当在PHP中使用mysqli函数库时，你可以使用面向对象或面向过程的语法。

在脚本中，我们使用如下语句连接MySQL服务器：

```
@ $db = new mysqli('localhost', 'bookorama', 'bookorama123', 'books');
```

以上代码实例化了mysqli类并且创建了到主机localhost的连接，该连接使用的用户名和密码分别是：bookorama和bookorama123。该连接被设置成使用books数据库。

使用这种面向对象的方法，可以调用这个对象的方法来访问数据库。如果你喜欢过程方法，

mysqli也提供了这个选项。要以这种面向过程的方式连接，可以使用如下语句：

```
@ $db = mysqli_connect('localhost', 'bookorama', 'bookorama123', 'books');
```

这个函数将返回一个资源，而不是一个对象。这个资源表示到数据库的连接，而且如果使用过程方法，必须将这个资源传递到mysqli的所有其他函数。这与文件处理函数非常类似，例如fopen()的工作方式。

mysqli的大多数函数都有面向对象接口和过程接口。通常，二者的差异在于过程版本的函数名称是以mysqli_开始的，同时要求传入通过mysqli_connect()函数获得的资源句柄。对这个规则来说，数据库连接是一个异常，因为它是由mysqli对象的构造函数来创建的。

尝试连接的结果需要进行检查，因为其他代码都无法在没有有效的数据库连接的情况下工作。可以使用如下所示的代码：

```
if (mysqli_connect_errno()) {
  echo 'Error: Could not connect to database. Please try again later.';
  exit;
}
```

(以上代码在面向对象版本和过程版本中相同。) mysqli_connect_errno()函数将在出现连接错误时返回一个错误号，如果成功，则返回0。

请注意，当连接到数据库时，我们通常会以错误抑制操作符@作为第一行代码。这样，可以很巧妙地处理任何错误。(这也可以通过异常来处理，我们只是没有在这个简单的例子中使用。)

请记住，MySQL对同时连接数据库的连接数量有一定的限制。MySQL参数max_connections决定了同时连接的个数，该参数和相关的Apache参数MaxClients的作用是，告诉服务器拒绝新的连接请求，从而确保系统资源不会在系统忙碌的时候，或软件瘫痪的时候被请求和使用。

可以通过修改配置文件来改变这两个参数的默认值。要设置Apache中的MaxClients参数，可以编辑系统中的httpd.conf文件。要为MySQL设置max_connections参数，可以编辑文件my.conf。

11.2.3　选择使用的数据库

请记住，当我们通过命令行界面使用MySQL的时候，需要告诉它要使用哪个数据库，命令如下所示：

```
use books;
```

当从Web连接数据库的时候，我们也需要这么做。在PHP中，可以调用mysqli_select_db()函数来实现，在这个例子中，我们调用了如下函数：

```
$db->select_db(dbname)
```

或

```
mysqli_select_db(db_resource, db_name)
```

这里，可以看到以上代码与我们前面介绍的函数类似：过程版本的函数名称以mysqli_开始，

需要额外的数据库句柄参数。

11.2.4　查询数据库

要执行数据库查询，可以使用mysqli_query()函数。但是，在使用之前，最好建立要运行的查询：

```
$query = "select * from books where ".$searchtype." like '%".$searchterm."%'";
```

在这个例子中，我们将在用户指定字段（$searchtype）中搜索用户输入值($searchterm)。注意我们使用了相似（like）逻辑用于匹配而不是相等逻辑——在数据库搜索的时候条件要更宽松，这是需要注意的。

提示　请记住，发送给MySQL的查询不需要在后面加一个分号，这与在MySQL监视程序输入查询有所不同。

现在，我们可以运行如下查询：

```
$result = $db->query($query);
```

或者，如果希望使用面向过程版本的函数，可以使用：

```
$result = mysqli_query($db, $query);
```

将所要运行的查询传给它，在过程版本的接口中，它是数据库连接（在这个例子中，为$db)。

面向对象版本将返回一个结果对象；过程版本将返回一个结果资源。(这与连接函数的工作原理类似）无论何种方法，都会将结果保存在一个变量（$result）中，以供以后使用。这个函数执行失败时，将返回false。

11.2.5　检索查询结果

我们可以使用不同的函数以不同的方式将查询结果从结果对象或标识符中取出来。结果对象或标识符是访问查询返回行的关键。

在这个例子中，我们统计了所返回记录行的行数，并且使用了mysqli_fetch_assoc()函数。

当使用面向对象方法时，返回的行数保存在结果对象的num_rows成员变量中，可以通过以下形式访问它：

```
$num_results = $result->num_rows;
```

当使用一个过程方法时，函数mysqli_num_rows()给出了查询返回的行数。你应该把它传给结果标识符，如下所示：

```
$num_results = mysqli_num_rows($result);
```

了解这一点是有用的——如果计划处理或显示该结果，现在可以知道这些结果有多少，并且通过一个循环就可以完成：

```
for ($i=0; $i <$num_results; $i++) {
  // process results
```

```
}
```

在每轮循环中，我们都将调用$result->fetch_assoc()函数（或mysqli_fetch_assoc()函数）。

如果没有返回行，该循环将停止执行。该函数接受结果集合中每一行并以一个相关数组返回该行，每个关键词为一个属性名，每个值为数组中相应的值：

```
$row = $result->fetch_assoc();
```

或者可以使用过程方法：

```
$row = mysqli_fetch_assoc($result);
```

给定相关数组$row，我们可以遍历每个字段并适当地显示它们，例如：

```
echo '<br />ISBN: ';
echo stripslashes($row['isbn']);
```

如前所述，我们调用stripslashes()函数以便在显示前整理其值。

从结果标识符中获取查询结果有几种不同的方法。不使用相关数组，可以使用函数mysqli_fetch_row()将结果取回到一个列举数组中，如下所示：

```
$row = $result->fetch_row($result);
```

或者

```
$row = mysqli_fetch_row($result);
```

这里，属性值将在每个数组值$row[0]、$row[1]等等的里面列出（mysqli_fetch_array()函数允许获取一行，作为两种数组之一，或作为两种数组）。

也可以使用mysqli_fetch_object()函数将一行取回到一个对象中：

```
$row = $result->fetch_object();
```

或者

```
$row = mysqli_fetch_object($result);
```

然后，通过$row->title、$row->author等访问每个属性。

11.2.6　从数据库断开连接

通过调用如下语句，可以释放结果集：

```
$result->free();
```

或者

```
mysqli_free_result($result);
```

然后可以使用：

```
$db->close();
```

或者

```
mysqli_close($db);
```

来关闭一个数据库连接。严格地说，这并不是必要的，因为脚本执行完毕的时候它们将被自动关闭。

11.3 将新信息放入数据库

显然，将新数据插入到数据库与从数据库中取回数据是相似的。我们可以遵循同样的基本步骤——建立一个连接、发送查询，最后检查结果。在这种情况中，发送的查询是INSERT而不是SELECT。

尽管这些处理过程非常类似，但是通过一个例子来了解二者的区别是非常有意义的。在图11-3中，可以看到一个基本HTML表单，它可以用来在数据库中输入新的图书。

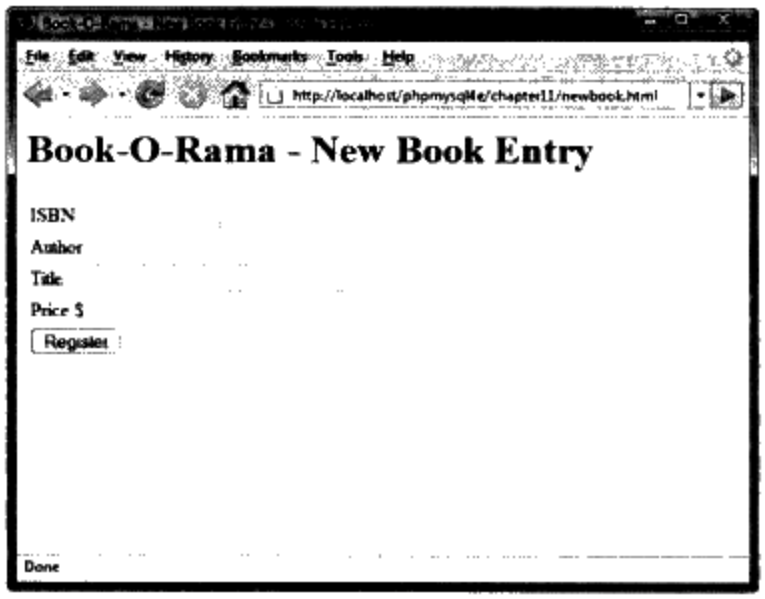

图11-3　输入新图书到数据库的界面，可供Book-O-Rama职员使用

该页面的HTML源代码如程序清单11-3所示。

该表单的结果将传递给insert_book.php，此脚本接收图书细节，执行一些小的验证，并尝试将数据写入到数据库中。其代码如程序清单11-4所示。

程序清单11-3　newbook.html——图书输入页的HTML

```
<html>
<head>
  <title>Book-O-Rama - New Book Entry</title>
</head>

<body>
  <h1>Book-O-Rama - New Book Entry</h1>

  <form action="insert_book.php" method="post">
    <table border="0">
      <tr>
        <td>ISBN</td>
        <td><input type="text" name="isbn" maxlength="13" size="13"></td>
      </tr>
      <tr>
        <td>Author</td>
        <td> <input type="text" name="author" maxlength="30" size="30"></td>
      </tr>
      <tr>
        <td>Title</td>
        <td> <input type="text" name="title" maxlength="60" size="30"></td>
      </tr>
```

```
      <tr>
        <td>Price $</td>
        <td><input type="text" name="price" maxlength="7" size="7"></td>
      </tr>
      <tr>
        <td colspan='2'><input type="submit" value="Register"></td>
      </tr>
    </table>
  </form>
</body>
</html>
```

程序清单11-4　insert_book.php——该脚本将新的图书写入到数据库

```
<html>
<head>
  <title>Book-O-Rama Book Entry Results</title>
</head>
<body>
<h1>Book-O-Rama Book Entry Results</h1>
<?php
  // create short variable names
  $isbn=$_POST['isbn'];
  $author=$_POST['author'];
  $title=$_POST['title'];
  $price=$_POST['price'];

  if (!$isbn || !$author || !$title || !$price) {
    echo "You have not entered all the required details.<br />"
        ."Please go back and try again.";
    exit;
  }

  if (!get_magic_quotes_gpc()) {
    $isbn = addslashes($isbn);
    $author = addslashes($author);
    $title = addslashes($title);
    $price = doubleval($price);
  }

  @ $db = new mysqli('localhost', 'bookorama', 'bookorama123', 'books');

  if (mysqli_connect_errno()) {
    echo "Error: Could not connect to database. Please try again later.";
    exit;
  }
  $query = "insert into books values
           ('".$isbn."', '".$author."', '".$title."', '".$price."')";
```

```
    $result = $db->query($query);

    if ($result) {
        echo $db->affected_rows." book inserted into database.";
    } else {
            echo "An error has occurred. The item was not added.";
    }

    $db->close();
?>
</body>
</html>
```

成功插入一本书的结果如图11-4所示。

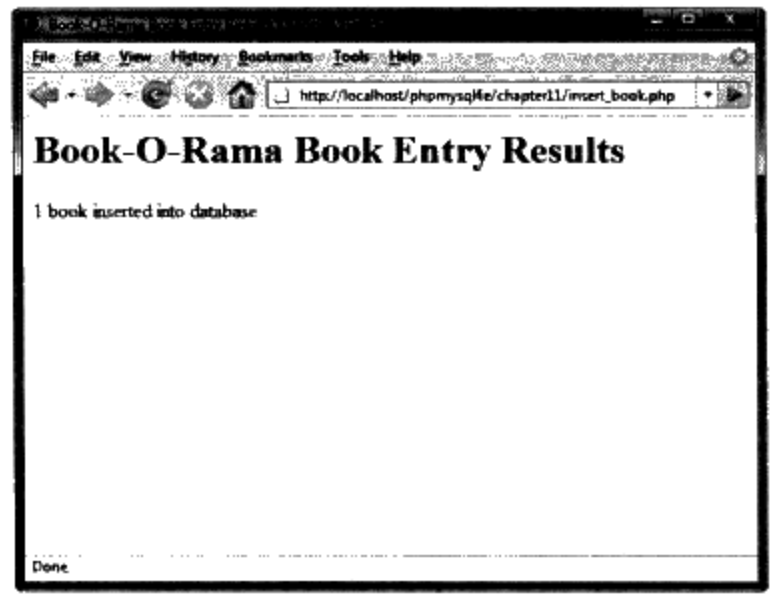

图11-4　脚本完成任务并报告图书已经被添加到数据库中

如果看看insert_book.php的代码，可以发现其中许多代码都与从数据库中取回数据的脚本相似。我们已经验证所有表格字段都已经填满，并调用addslashes()函数（如果是必需的）正确地将数据格式化，以便插入数据库：

```
if (!get_magic_quotes_gpc()) {
    $isbn = addslashes($isbn);
    $author = addslashes($author);
    $title = addslashes($title);
    $price = doubleval($price);
}
```

当价格以浮点数的形式保存在数据库时，我们不希望在小数点的前后插入斜杠。通过调用doubleval()函数，可以对该数字字段进行过滤，从而去除所有临时字符。该函数在第1章中讨论过。也要注意用户可能输入的任何货币符号。

在这里，我们再次通过实例化mysqli对象来连接数据库，而且设置了一个发送给数据库的查询。在这个例子中，该查询是一个SQL INSERT操作：

```
$query = "insert into books values
          ('".$isbn."', '".$author."', '".$title."', '".$price."')";
$result = $db->query($query);
```

通过调用$db->query()（或者mysqli_query()，如果希望使用面向过程风格的方法），该查询将以常见的方式在数据库上执行。

使用INSERT和SELECT的一个显著不同之处在于对mysqli_affected_rows()的使用，这是一个过程式版本的函数或者面向对象版本中的一个类成员变量：

```
echo $db->affected_rows." book inserted into database.";
```

在前面的脚本中，我们使用mysqli_num_rows()来确定SELECT操作可以返回多少行记录。

当编写一个修改数据库的查询时，例如INSERT、DELETE和UPDATE，应该使用

mysqli_affected_rows()函数。

到目前为止，我们已经介绍了通过PHP使用MySQL数据库的基础知识。

11.4 使用Prepared语句

mysqli函数库支持prepared语句的使用。它们对于在执行大量具有不同数据的相同查询时，可以提高执行速度。它们也可以免受SQL注射风格（injection-style）的攻击。

Prepared语句的基本思想是可以向MySQL发送一个需要执行的查询模板，然后再单独发送数据。我们可以向相同的Prepared语句发送大量的相同数据；这个特性对批处理的插入操作来说是非常有用的。

在insert_book.php脚本中，可以使用prepared语句，如下所示：

```
$query = "insert into books values(?, ?, ?, ?)";
$stmt = $db->prepare($query);
$stmt->bind_param("sssd", $isbn, $author, $title, $price);
$stmt->execute();
echo $stmt->affected_rows.' book inserted into database.';
$stmt->close();
```

下面，我们逐行分析以上代码。

当设置查询时，不是替换前面已经生成的变量，而是在每一段数据的位置设置问号。在这些问号的周围，不能再设置问号或其他分界符号。

第二行是调用$db->prepare()，在过程版本中，是通过mysqli_stmt_prepare()函数实现的。这一行将构建一个语句对象或需要用来完成实际处理的资源。

语句对象有一个bind_param()方法。（在过程版本中，是mysqli_stmt_bind_param()函数）。bind_param()的用途是告诉PHP哪些变量应该被问号所替换。第一个参数是一个格式化字符串，与printf()使用的格式化字符串不同。在这里，所传递的值意味着4个参数分别是字符串、字符串、字符串和双精度。格式化字符串中的其他可能字符还有：i表示整数，b表示blob。在这个参数之后，必须列出与语句中的问号数量相同的变量。它们将依次被替换。

调用$stmt->execute()函数（在过程版本中是mysqli_stmt_execute()函数）将真正运行这个查询。我们可以访问受影响的行数并关闭这个语句。

那么Prepared语句的作用如何呢？这里，一个优点是可以改变这4个绑定变量的值，并且在不用准备的情况下重新执行这个语句。这个功能对于循环执行批量插入操作来说是非常有用的。

与绑定参数一样，也可以绑定结果。对于SELECT类型查询，可以使用$stmt->bind_result()函数（或mysqli_stmt_bind_result()函数）提供希望填充结果列的变量列表。

每次调用$stmt->fetch()函数（或者mysqli_stmt_fetch()函数）时，结果集下一行的列值将被填充到这些绑定变量中。例如，在前面介绍的图书搜索脚本中，可以使用：

```
$stmt->bind_result($isbn, $author, $title, $price);
```

将这4个变量绑定到将通过查询返回的4列。在调用如下语句后：

```
$stmt->execute();
```

可以在循环中调用：

```
$stmt->fetch();
```

每当该语句被调用时，它将获得下一个结果行，并填充到4个绑定变量中。
也可以在相同的脚本中使用mysqli_stmt_bind_param()函数和mysqli_stmt_ bind_result()函数。

11.5 使用PHP与数据库交互的其他接口

PHP支持连接到许多不同数据库的函数，包括Oracle、Microsoft SQL Server和PostgreSQL。

通常，连接和查询这些数据库的基本原理是相同的。个别函数名称可能会有所不同，而且不同的数据库具有不同的功能，但是如果可以连接到MySQL，就应该能够很容易应用MySQL中的知识连接到其他数据库。

如果希望使用PHP还没有提供支持的特殊数据库，可以使用常规的ODBC函数。ODBC表示开放的数据库连接，它是连接数据库的标准。由于各种明显的原因，ODBC只具有任何函数集的有限功能。如果要求必须兼容所有数据库，就不能使用任何数据库的特殊功能。

除了PHP附带的函数库以外，一些可供使用的数据库抽象类，例如MDB2，允许为不同的数据库类型使用相同的函数名称。

使用常规的数据库接口：PEAR MDB2

接下来，我们将简要地介绍使用PEAR MDB2抽象层的例子。这是PEAR所有组件中使用最为广泛的组件之一。关于MDB2抽象层的安装，请参阅附录A中“PEAR的安装”一节的详细介绍。

为了便于比较，我们将介绍如何使用MDB2来编写搜索结果的脚本（见程序清单11-5）。

程序清单11-5　results_generic.php——从MySQL数据库检索结果并且格式化以供显示

```
<html>
<head>
  <title>Book-O-Rama Search Results</title>
</head>
<body>
<h1>Book-O-Rama Search Results</h1>
<?php
  // create short variable names
  $searchtype=$_POST['searchtype'];
  $searchterm=trim($_POST['searchterm']);

  if (!$searchtype || !$searchterm) {
     echo "You have not entered search details. Please go back and try again.";
     exit;
```

```
  }

  if (!get_magic_quotes_gpc()) {
    $searchtype = addslashes($searchtype);
    $searchterm = addslashes($searchterm);
}

  // set up for using PEAR MDB2
  require_once('MDB2.php');
  $user = 'bookorama';
  $pass = 'bookorama123';
  $host = 'localhost';
  $db_name = 'books';

  // set up universal connection string or DSN
  $dsn = 'mysqli://'.$user.':'.$pass.'@'.$host.'/'.$db_name;

  // connect to database
  $db = &MDB2::connect($dsn);

  // check if connection worked
  if (MDB2::isError($db)) {
    echo $db->getMessage();
    exit;
  }

  // perform query
  $query = "select * from books where ".$searchtype." like '%".$searchterm."%'";

  $result = $db->query($query);

  // check that result was ok
  if (MDB2::isError($result)) {
    echo $db->getMessage();
    exit;
  }

  // get number of returned rows
  $num_results = $result->numRows();

  // display each returned row
  for ($i=0; $i <$num_results; $i++) {
     $row = $result->fetchRow(MDB2_FETCHMODE_ASSOC);
     echo "<p><strong>'.($i+1).'. Title: ";
     echo htmlspecialchars(stripslashes($row['title']));
```

```
        echo "</strong><br />Author: ";
        echo stripslashes($row['author']);
        echo "<br />ISBN: ";
        echo stripslashes($row['isbn']);
        echo '<br />Price: ";
        echo stripslashes($row['price']);
        echo "</p>";
    }

    // disconnect from database
    $db->disconnect();
?>
</body>
</html>
```

下面，让我们看看以上代码与前面的代码有什么差异。

要连接数据库，我们使用了如下语句：

```
$db = MDB2::connect($dsn);
```

这个函数接收一个通用的连接字符串，该字符串包含了连接一个数据库所必需的所有参数。如果查看连接字符串的格式，可以看到这些参数：

```
$dsn = 'mysqli://".$user.":".$pass.'@'.$host.'/'.$db_name;
```

在完成数据库连接后，我们将使用isError()检查该连接是否成功，如果失败，将打印错误信息并退出。

```
if (MDB2::isError($db)) {
    echo $db->getMessage();
    exit;
}
```

假设所有操作都正确完成，我们将设置并执行一个查询，如下所示：

```
$result = $db->query($query);
```

我们可以检查返回的记录行数：

```
$num_results = $result->numRows();
```

按照以下代码取回每一行：

```
$row = $result->fetchRow(DB_FETCHMODE_ASSOC);
```

通用方法fetchRow()可以以许多格式提取结果集中的一行；MDB2_FETCHMODE_ASSOC参数表示我们希望以相关数组方式返回结果行。

在输出查询结果行后，我们将关闭数据库连接：

```
$db->disconnect();
```

可以看到，这个例子非常类似于我们的第一个脚本。

使用MDB2的优点是只要记住一种数据库函数集，如果要改变数据库软件，只要对代码进

行少量的修改就可以了。

既然这是一本介绍MySQL的图书，出于灵活性和速度的考虑，我们还将使用MySQL自带的本地函数库。但是，在项目中，当使用抽象层时，MDB2包所提供的抽象层特性将非常有用，因此我们可能会希望使用MDB2包。

11.6 进一步学习

要了解更多关于连接MySQL和使用PHP的信息，请参阅PHP和MySQL手册的相关部分。

要了解更多关于ODBC的信息，请访问：http://www.webopedia.com/TERM/O/ODBC.html

11.7 下一章

在下一章中，我们将更详细地探讨关于MySQL管理的细节并讨论如何优化数据库。