



福州大学至诚学院
FUZHOU UNIVERSITY ZHICHENG COLLEGE

服务端开发技术

杨雄



7. PHP与MySQL

01

PHP对MySQL的支持

02

PHP与MySQL的连接

03

PHP与MySQL的交互

04

使用PDO与MySQL交互

05

应用实例

Part.1

7.1 对MySQL的支持

7.1 PHP对MySQL的支持

1、PHP对数据库的支持

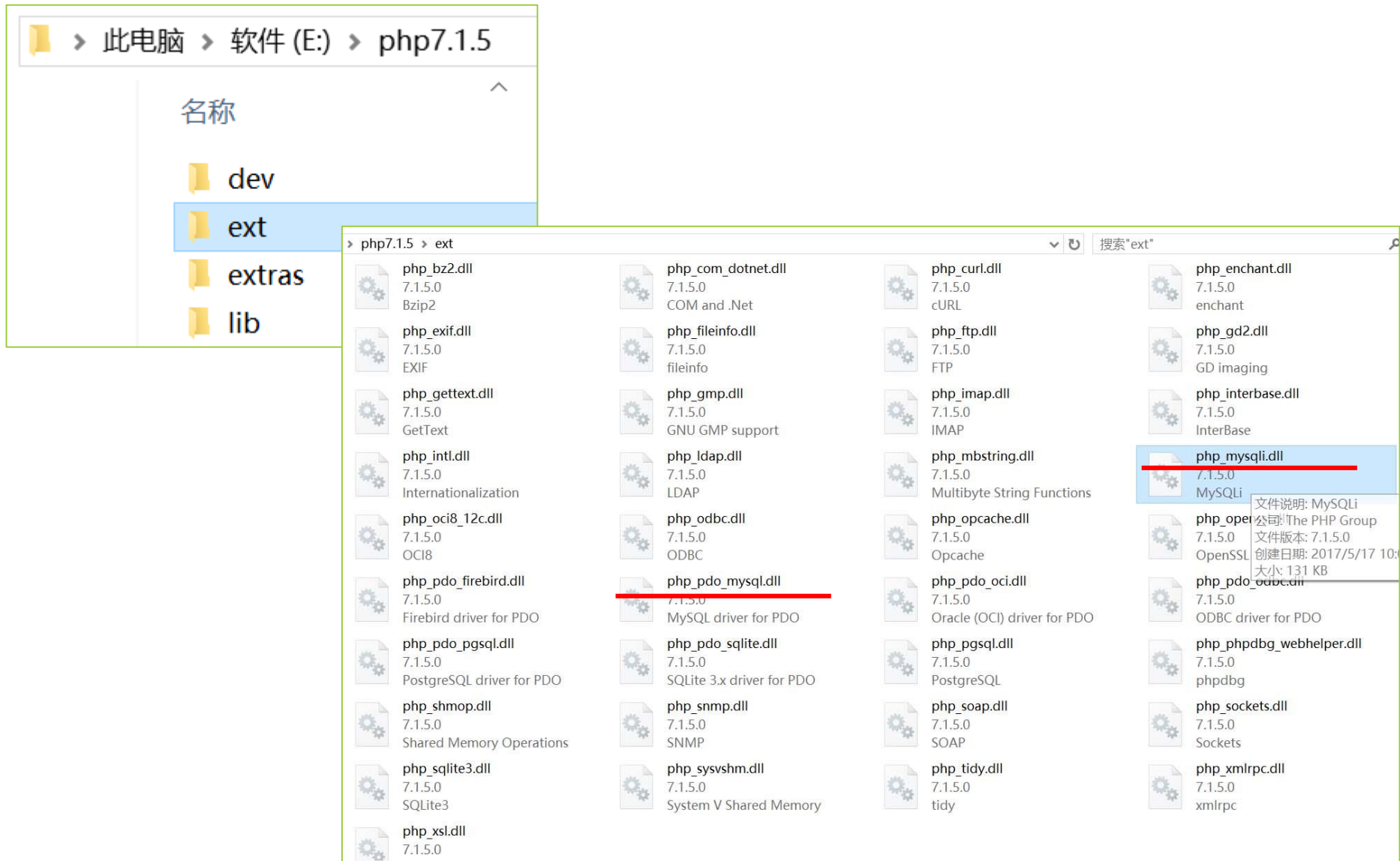
PHP支持对多种数据库的操作，且提供了相关的数据库连接函数和操作函数。

PHP支持目前几乎所有的主流数据库，如，Adabas D、InterBase、PostgreSQL、dBase、FrontBase、**SQLite**、Empress、**mSQL**、Sybase、Hyperwave、**MySQL**、Velocis、**IBM DB2**、ODBC、Unix dbm、informix、**Oracle** (OCI7 和 OCI8)、Ingres、Ovrimos等。

PHP对数据库的支持，是通过安装相应的**扩展库**来实现的。

PHP的扩展库位于**PHP根目录的ext子目录**中，如图所示。

7.1 PHP对MySQL的支持



7.1 PHP对MySQL的支持

2、PHP的MySQL扩展

在PHP中访问数据库通常有**2种方法**，一种是使用数据库**特定的扩展**，另一种是使用不受数据库约束的**PDO(PHP数据对象)扩展**。

对于**MySQL数据库**来说，常用的**MySQL与MySQLi扩展**都是属于PHP的**MySQL数据库特定扩展**。

7.1 PHP对MySQL的支持

(1) MySQL特定扩展

PHP对MySQL数据库的特定扩展有2种。旧版本中的**MySQL标准扩展**。该扩展能够用于所有版本的PHP和MySQL中，标准MySQL扩展函数都以“mysql_”开头。

MySQLi扩展，即**改进的MySQL扩展**。该扩展在PHP 5中添加，并能够用于MySQL 4.1或更高的版本中。这些函数都以“mysqli_”开头，并使用了MySQL中的一些新增功能。

PHP的MySQLi扩展使用了面向对象技术，所以，其中的扩展函数也可以以面向对象的方式来使用。

7.1 PHP对MySQL的支持

(2) PDO扩展

PHP的**PDO扩展**，就是为PHP访问数据库定义的一个轻量级的**一致接口**。通过这个接口，PHP可以**与各种不同的数据库**进行交互。

但要注意的是，利用PDO扩展自身并不能实现任何数据库功能，必须**使用一个具体数据库PDO驱动**来完成相关的数据库服务。

(3) 扩展的启用

要加载某个扩展库，需要在PHP的配置文件php.ini中启用该扩展。

7.1 PHP对MySQL的支持

配置扩展库主目录

PHP的扩展是以动态链接库的形式存在的，在启用某个扩展之前，首先要查看安装的PHP版本中有没有该扩展的动态链接库；然后，打开PHP的配置文件将extension_dir配置项指向扩展库的存放目录。如下所示。

```
extension_dir = "e:/php7.1.5/ext"
```

启用mysqli及pdo_mysql扩展

启用扩展，就是在启动PHP应用服务器时，加载相应的扩展动态链接库。在PHP的配置文件中打开或添加如下语句：

```
extension = php_mysqli.dll
```

```
extension = php_pdo_mysql.dll
```

这里，开启PHP的MySQLi扩展与PDO扩展。

Part.2

7.2 与MySQL的连接

7.2 PHP与MySQL的连接

1、连接服务器

要实现PHP与MySQL的交互，首先必须建立PHP应用服务器与MySQL数据库服务器的连接，让它们能够互相识别与通信，也就是能够进行双向的数据传递。

PHP与MySQL服务器的连接，可以使用2种方法来完成，一种是面向过程方法，另一种是面向对象方法。

(1) 面向过程方法

```
mysqli_connect ([ string $host = ini_get("mysqli.default_host") [,  
string $username = ini_get("mysqli.default_user") [, string $passwd =  
ini_get("mysqli.default_pw") [, string $dbname = "" [, int $port =  
ini_get("mysqli.default_port") [, string $socket =  
ini_get("mysqli.default_socket") ]]]]] )
```

7.2 PHP与MySQL的连接

(2) 面向对象方法

使用面向对象方法连接MySQL，需要**实例化mysql扩展中的mysql类**，该类的**对象表示了PHP和MySQL数据库之间的一个连接**。

使用构造方法

其语法格式为：

```
$link = new mysqli($host = null, $username = null, $passwd = null,  
$dbname = null, $port = null, $socket = null));
```

7.2 PHP与MySQL的连接

使用成员方法

其语法格式为：

```
$objLink = new mysqli();
```

```
$link = $objLink->connect($host = null, $user = null, $password = null,  
$database = null, $port = null, $socket = null);
```

从上述面向对象的2种格式可以看出，不管是使用构造方法、还是使用connect()成员方法，**需要的参数都是一样的**，这些参数也都与**mysqli_connect()函数参数相同**。

其实，mysqli_connect()只是mysqli类的connect对象方法的**别名**而已。（严格来说，它是mysqli::__construct()构造方法的别名）

7.2 PHP与MySQL的连接

【例】 连接MySQL数据库服务器。

The image shows a code editor window titled 'example9_1.php' and a web browser window titled 'Internal Web Browser'. The code editor contains PHP code for connecting to MySQL using three different methods: `mysql_connect()`, `mysqli_connect()`, and `new mysqli()`. The web browser displays the output of the code, which is '连接MySQL'. Red horizontal lines are drawn under the code lines in the editor, and pink arrows point from these lines to the corresponding output text in the browser window.

```
9 <?php
10 $link1 = mysqli_connect();
11 echo '函数方法, 默认参数: '.mysqli_get_server_info($link1);
12 echo '<br/>';
13 $link2 = mysqli_connect("localhost","root","wwp");
14 echo '函数方法, 自带参数: '.mysqli_get_server_info($link2);
15 echo '<br/><br/>';
16 $link3 = new mysqli("localhost","root","wwp");
17 echo '对象方法, 构造函数: '.$link3->get_server_info();
18 echo '<br/>';
19 $link4 = new mysqli();
20 $link4->connect("localhost","root","wwp");
21 echo '对象方法, 成员函数【自带参数】: ';
22 echo $link4->get_server_info();
23 echo '<br/>';
24 $link5 = new mysqli();
25 $link5->connect();
26 echo '对象方法, 成员函数【默认参数】: ';
27 echo $link5->get server info().'<br/><br/>';
28 ?></p>
```

Internal Web Browser

plee/chapter09/example9_1.php

连接MySQL

函数方法, 默认参数: 5.7.17
函数方法, 自带参数: 5.7.17

对象方法, 构造函数: 5.7.17
对象方法, 成员函数【自带参数】: 5.7.17
对象方法, 成员函数【默认参数】: 5.7.17

7.2 PHP与MySQL的连接

2、连接错误的处理

在上述例中，演示了**PHP与MySQL连接的5种代码形式**，从浏览器输出的结果可以看出，这几种方式的连接都是成功的。

但是在实际开发过程中，由于种种原因，可能会导致连接失败，因此，在进行数据库连接操作时，一定要**注意监视连接错误并做出相应的反应**。

(1) 获取错误信息

一般情况下，**错误信息**包括**错误码**与**错误消息**2种类型。

这2种错误信息的获取在采用不同的连接代码形式时，稍有差别。

7.2 PHP与MySQL的连接

若采用**面向过程的方式**与MySQL建立连接，通常使用

mysqli_connect_errno()

函数来**获取错误码**。该函数返回**上一次连接**错误的代码值，如果没有发生连接错误，则返回整数**0**。

另外，使用函数

mysqli_connect_error()

来**获取错误描述信息**。该函数返回上一次连接错误的错误描述字符串，如果没有错误发生，则返回 **NULL**。

若采用**面向对象的代码形式**来连接MySQL数据库，则可以通过**mysqli对象**，直接获取其**成员属性****connect_errno**与**connect_error**的值。

7.2 PHP与MySQL的连接

(2) 处理连接错误

【例】处理MySQL连接错误。

example9_3.php

```
9      <?php
10      $link = @mysqli_connect("127.0.0.1","root","wwp");
11      $errno = mysqli_connect_errno();
12      $error = mysqli_connect_error();
13      if ($errno) {
14          echo '连接MySQL失败! ['. $error. ']' ;
15          exit();
16      }else {
17          try {
18              $link = @new mysqli("127.0.0.1","root","123");
19              if($link->connect_errno)
20                  throw new mysqli_sql_exception();
21          } catch (mysqli_sql_exception $e) {
22              echo '连接MySQL失败! ' . $e . '<br/>';
23          }
24          echo '继续执行后续代码...';
25      }
26      ?></p>
```

Internal Web Browser

http://localhost/example9_3.php

处理MySQL连接错误

连接MySQL失败!
mysqli_sql_exception in E:\Apache24
\htdocs\examplelee\chapter09
\example9_3.php:20 Stack trace: #0
{main}

继续执行后续代码...

7.2 PHP与MySQL的连接

3、断开服务器

MySQL服务器的连接占用了数据库服务器以及Web服务器的大量资源，PHP程序与MySQL服务器交互完成后，应尽早断开连接，以提高设备性能以及Web应用程序的执行效率。

使用**mysqli_close()**函数可以关闭MySQL服务器连接。其语法格式为：

```
bool mysqli_close ([ resource $link ])
```

若使用面向对象方法进行连接，直接使用mysqli对象调用其**成员函数close()**即可。

需要说明的是，在PHP与MySQL数据库的交互结束以后，如果没有手动断开连接，连接也会在**文件执行完毕后**，**由PHP自动关闭**，所以不必担心它会永久存在。

7.2 PHP与MySQL的连接

4、连接文件

如果Web应用的每个页面都需要获取数据库中的数据，那么，就必须**在每个页面中**编写数据库连接代码，建立与数据库服务器的连接。

在每个Web页面中编写相同的代码，不仅**增加了出错的概率**，也**不利用后期的扩展与维护**。另外，所有的连接参数都是以明文的形式存储在文件中的，包括像密码这样的**敏感数据**。

在实际开发过程，为了安全起见，**常常将数据库连接代码存放在一个单独的PHP文件中**，这个文件作者称之为**连接文件**。

```
<?php
$link = @mysqli_connect("localhost","root","wwp")
or die('数据库服务器连接失败！系统错误信息为：'.mysqli_connect_error());
@mysqli_select_db($link, "test_students")
or die('打开数据库失败！系统错误信息为：'.mysqli_error($link));
mysqli_query($link, "set names utf8");
?>
```

Part.3

7.3 与MySQL的交互

7.3 PHP与MySQL的交互

1、执行SQL语句

在PHP编程中，SQL语句的执行，是通过直接调用函数mysql_query()、或者是调用mysqli类的成员函数query()来实现的。

(1) 使用mysql_query()函数

该函数语法格式如下：

```
mixed mysql_query(resource $link, string $query, [, int $resultmode =  
MYSQLI_STORE_RESULT]);
```

7.3 PHP与MySQL的交互

The image shows a PHP script and its output in a web browser. The script is named `example9_4.php` and is located in the directory `ter09/example9_4.php`. The script connects to a MySQL database and queries for tables. The output shows an array of table names: `course`, `score`, `student`, `teacher`, and `testtable`.

PHP Script (example9_4.php):

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8" >
5 <title>例9.4 使用MySQL连接文件</title>
6 </head>
7 <body>
8 <h4>使用MySQL连接文件</h4><hr /><p>
9 <?php
10     require_once 'example9_4_connect.php';
11     $query = "show tables";
12     $result = mysqli_query($link, $query);
13     $rows = mysqli_fetch_all($result);
14     echo '<pre>';
15     print_r($rows);
16     echo '</pre>';
17     mysqli_free_result($result);
18     mysqli_close($link);
19 ?></p>
20 </body>
21 </html>
```

Annotations:

- 连接** (Connection): Points to the `require_once 'example9_4_connect.php';` line.
- 查询字符串** (Query String): Points to the `$query = "show tables";` line.

Internal Web Browser Output:

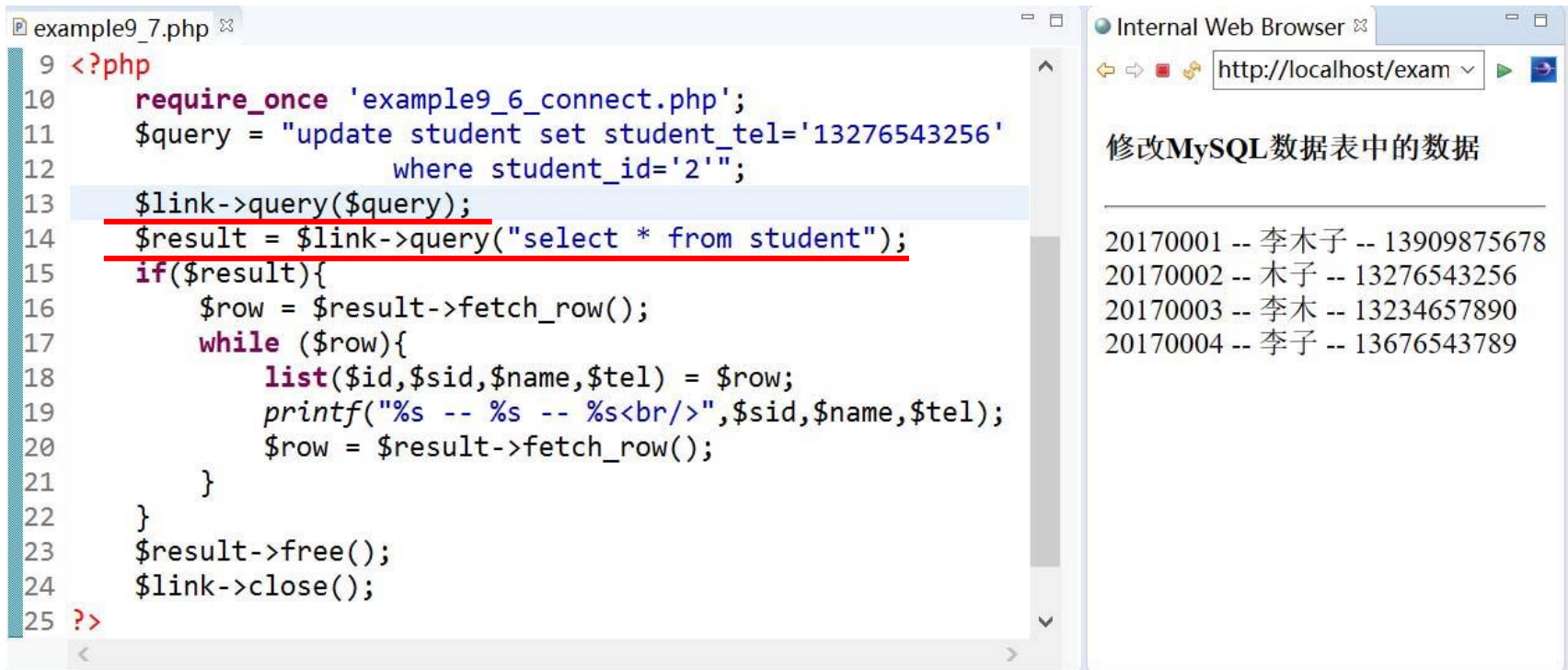
```
Array
(
    [0] => Array
        (
            [0] => course
        )
    [1] => Array
        (
            [0] => score
        )
    [2] => Array
        (
            [0] => student
        )
    [3] => Array
        (
            [0] => teacher
        )
    [4] => Array
        (
            [0] => testtable
        )
)
```

7.3 PHP与MySQL的交互

(2) 使用mysqli类成员函数

mixed **mysqli::query** (string \$query [, int \$resultmode = MYSQLI_STORE_RESULT])

其中，各参数含义与mysqli_query()函数中同名参数含义相同。



The screenshot shows a PHP script named `example9_7.php` and its output in an internal web browser. The script performs an update and a query on a MySQL database.

```
9 <?php
10 require_once 'example9_6_connect.php';
11 $query = "update student set student_tel='13276543256'
12         where student_id='2'";
13 $link->query($query);
14 $result = $link->query("select * from student");
15 if($result){
16     $row = $result->fetch_row();
17     while ($row){
18         list($id,$sid,$name,$tel) = $row;
19         printf("%s -- %s -- %s<br/>", $sid, $name, $tel);
20         $row = $result->fetch_row();
21     }
22 }
23 $result->free();
24 $link->close();
25 ?>
```

The web browser displays the output of the script, titled "修改MySQL数据表中的数据". The output shows a list of student records:

Student ID	Student Name	Student Phone Number
20170001	李木子	13909875678
20170002	木子	13276543256
20170003	李木	13234657890
20170004	李子	13676543789

7.3 PHP与MySQL的交互

(3) 获取数据

【例】 获取MySQL数据库中的数据。

```
example9_5.php
9 <?php
10 require_once 'example9_4 connect.php';
11 $query = "select * from student";
12 $result = mysqli_query($link, $query);
13 $rows = mysqli_fetch_all($result);
14 ?>
15 <table border="1">
16     <caption>学生信息</caption>
17     <tr><td>学号</td><td>姓名</td><td>联系方式</td></tr>
18     <?php foreach ($rows as $row){?>
19     <tr>
20         <td><?php echo $row[1]?></td>
21         <td><?php echo $row[2]?></td>
22         <td><?php echo $row[3]?></td>
23     </tr>
24     <?php }?>
25 </table>
26 <?php
27     mysqli_free_result($result);
28     mysqli_close($link);
29 ?>
```

结果集

Internal Web Browser

http://localhost/example/

获取MySQL中的数据

学生信息

学号	姓名	联系方式
20170001	李木子	13909875678
20170002	木子	
20170003	李木	13234657890

7.3 PHP与MySQL的交互

(4) 插入数据

【例】向MySQL数据表中插入数据。

```
example9_6.php
9 <?php
10 require_once 'example9_6_connect.php';
11 $query = "insert into student values
12         (null,'20170004','李子','13676543789)";
13 mysqli_query($link, $query);
14 $query = "select * from student";
15 $result = mysqli_query($link, $query);
16 $rows = mysqli_fetch_all($result);
17 ?>
18 <table border="1">
19     <caption>学生信息</caption>
20     <tr><td>学号</td><td>姓名</td><td>联系方式</td></tr>
21     <?php foreach ($rows as $row){?>
22         <tr>
23             <td><?php echo $row[1]?></td>
24             <td><?php echo $row[2]?></td>
25             <td><?php echo $row[3]?></td>
26         </tr>
27     <?php }?>
28 </table>
29 <?php
30     mysqli_free_result($result);
31     mysqli_close($link);
32 ?>
```

Internal Web Browser

http://localhost/exam

向MySQL数据表中插入数据

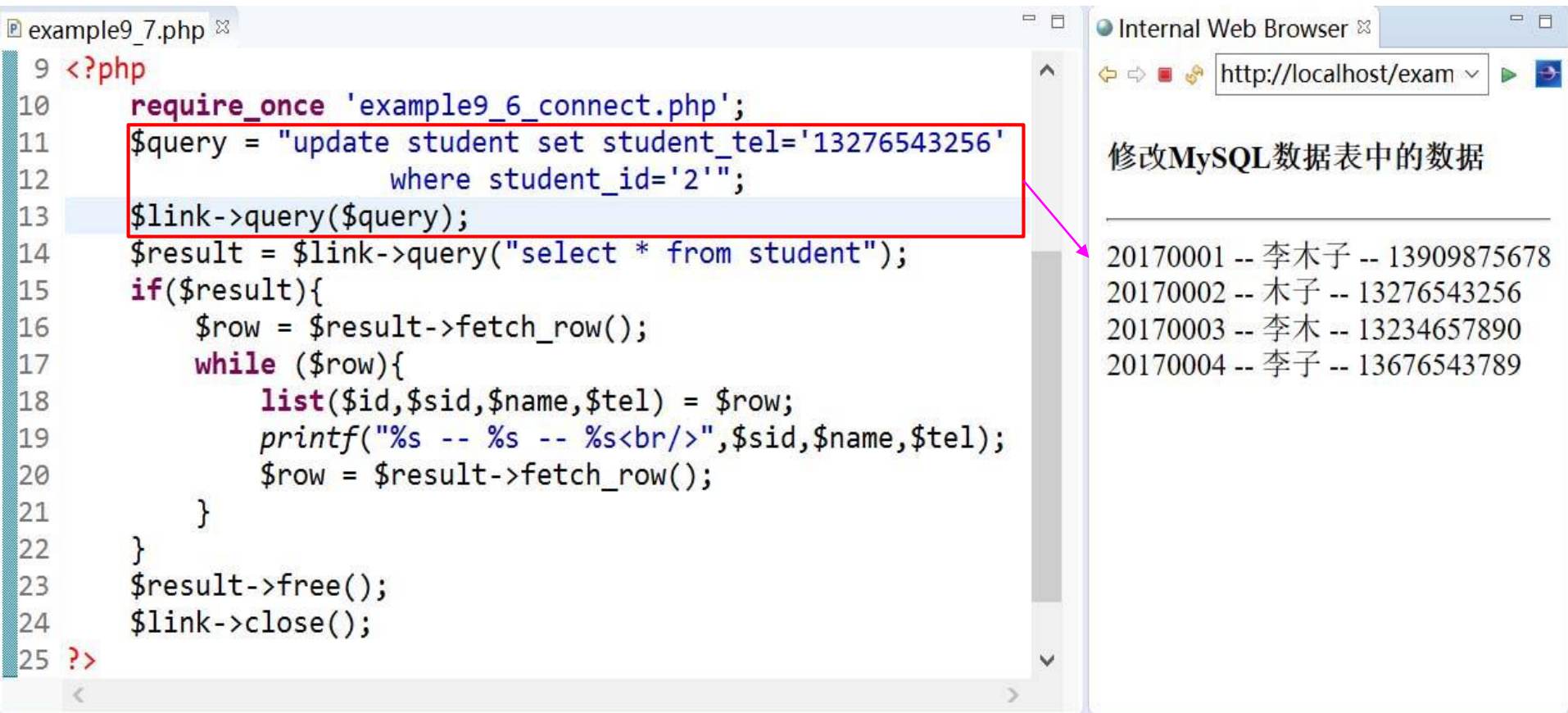
学生信息

学号	姓名	联系方式
20170001	李木子	13909875678
20170002	木子	
20170003	李木	13234657890
20170004	李子	13676543789

7.3 PHP与MySQL的交互

(5) 更新数据

【例】修改MySQL数据表中的数据。



The image shows a development environment with a code editor on the left and a web browser on the right. The code editor displays a PHP script named `example9_7.php`. A red box highlights the SQL update query and the execution line. A pink arrow points from this box to the web browser. The browser shows the output of the script, which is a list of student records with their IDs, names, and phone numbers.

```
example9_7.php
9  <?php
10  require_once 'example9_6_connect.php';
11  $query = "update student set student_tel='13276543256'
12          where student_id='2'";
13  $link->query($query);
14  $result = $link->query("select * from student");
15  if($result){
16      $row = $result->fetch_row();
17      while ($row){
18          list($id,$sid,$name,$tel) = $row;
19          printf("%s -- %s -- %s<br/>", $sid, $name, $tel);
20          $row = $result->fetch_row();
21      }
22  }
23  $result->free();
24  $link->close();
25  ?>
```

Internal Web Browser
http://localhost/exam

修改MySQL数据表中的数据

20170001 -- 李木子 -- 13909875678
20170002 -- 木子 -- 13276543256
20170003 -- 李木 -- 13234657890
20170004 -- 李子 -- 13676543789

7.3 PHP与MySQL的交互

(6) 删除数据

【例】删除MySQL数据表中的数据。

```
9 <?php
10 require_once 'example9_6_connect.php';
11 $query = "delete from student where student_name='李子'";
12 $del = @$link->query($query);
13 if(($del !== false) && ($link->affected_rows != 0)){
14     $result = $link->query("select * from student");
15     if($result){
16         $row = $result->fetch_row();
17         while ($row){
18             echo $row[1]. '    ';
19             echo $row[2]. '    ';
20             echo $row[3]. '<br/>';
21             $row = $result->fetch_row();
22         }
23     }
24     $result->free();
25 }else{
26     echo '<p style="color:red">删除数据失败! </p>';
27 }
28 $link->close();
29 ?>
```

Internal Web Browser

http://localhost/exar

删除MySQL数据表中的数据

20170001	李木子	13909875678
20170002	木子	13276543256
20170003	李木	13234657890

7.3 PHP与MySQL的交互

2、解析查询结果

一旦**执行**了查询并**获取**到了结果集，那么接下来的工作就是对结果集进行**解析**，从中**提取**需要的数据信息。

在PHP中，可以采用**多种方法**来**获取结果集中的数据**，具体选择哪一种，主要取决于个人的习惯与喜好，因为这些方法**只是引用数据的方法有所不同而已**。

(1) 数组方式

用**数组**的方式接收来自于**查询结果集的数据**。使用的函数主要有：

`mysqli_fetch_array()`、`mysqli_fetch_row()`、`mysqli_fetch_assoc()`、`mysqli_fetch_all()`

7.3 PHP与MySQL的交互

mysqli_fetch_array()

该函数从结果集中取得**一行**作为**关联数组**、或**数字索引数组**、或**二者兼有**。

example9_9.php

```
7 <body>
8 <h4>使用数组方式接收MySQL查询数据</h4><hr /><p>
9 <?php
10     require_once 'example9_4_connect.php';
11     $query = "select * from student";
12     $result = mysqli_query($link, $query);
13     $row = mysqli_fetch_array($result);
14     //$row = mysqli_fetch_array($result,MYSQLI_ASSOC);
15     //$row = mysqli_fetch_array($result,MYSQLI_NUM);
16     //$row = mysqli_fetch_array($result,MYSQLI_BOTH);
17     echo '<pre>';print_r($row);echo '</pre>';
18     mysqli_free_result($result);
19     mysqli_close($link);
20 ?></p>
21 </body>
22 </html>
```

默认为
MYSQLI_BOTH

Internal Web Browser

http://localhost/example/cl

使用数组方式接收MySQL查询数据

```
Array
(
    [0] => 1
    [student_id] => 1
    [1] => 20170001
    [student_no] => 20170001
    [2] => 李木子
    [student_name] => 李木子
    [3] => 13909875678
    [student_tel] => 13909875678
)
```

7.3 PHP与MySQL的交互

`mysqli_fetch_row()`

该函数从结果集中取得**一行**作为**数字索引数组**。

`mysqli_fetch_assoc()`

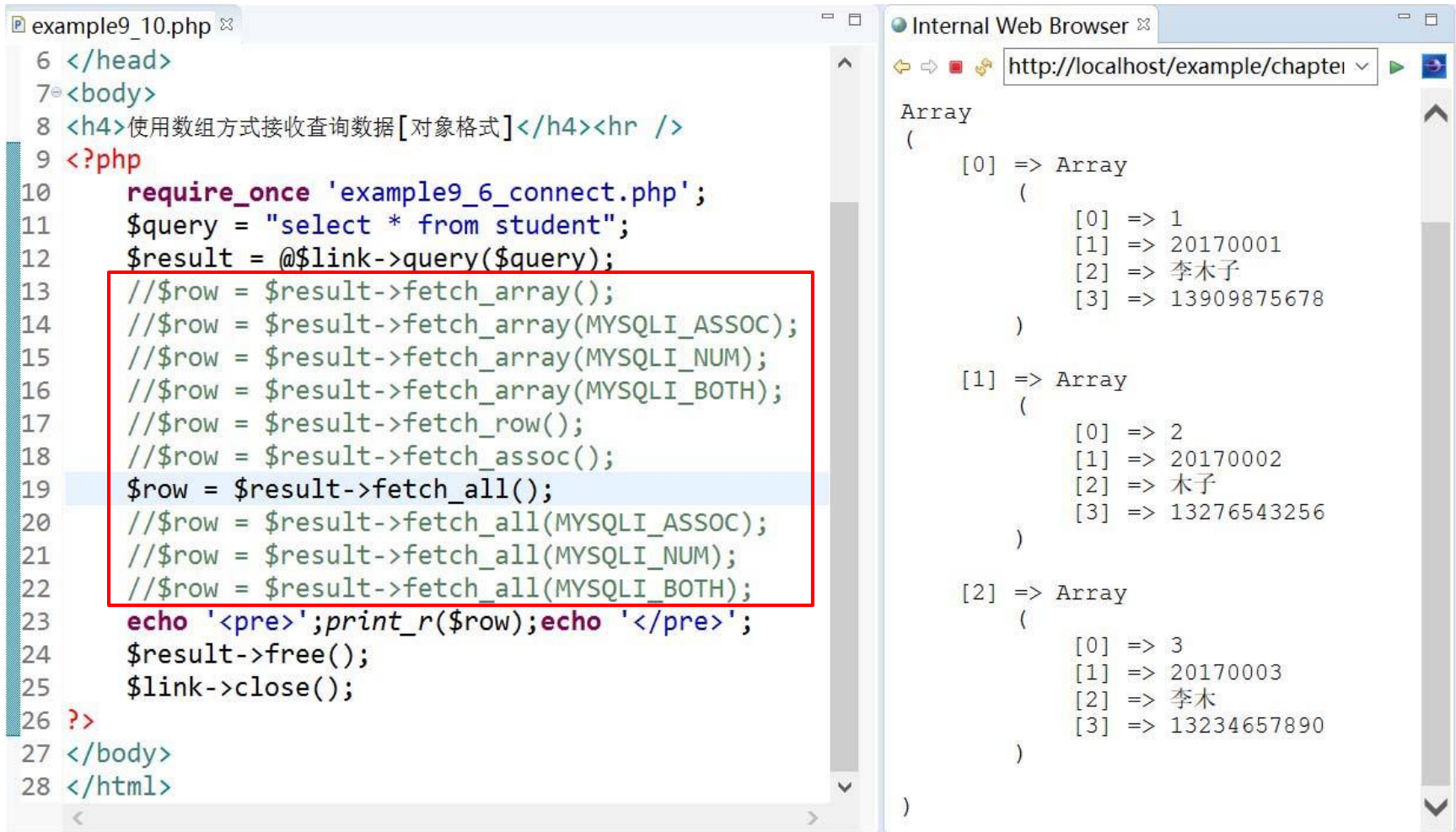
该函数从结果集中取得**一行**作为**关联数组**。

`mysqli_fetch_all()`

该函数从结果集中取得**所有行**作为**关联数组**、**或数字索引数组**、**或二者兼有**。

7.3 PHP与MySQL的交互

【例】 使用数组接收MySQL查询数据，用面向对象格式编写代码。



The image shows a code editor on the left and a web browser on the right. The code editor displays a PHP script named `example9_10.php`. The script connects to a MySQL database, executes a query `select * from student`, and fetches all results using `$result->fetch_all()`. The fetched data is printed using `print_r($row)`. A red box highlights the `fetch_all` method calls for different result formats. The web browser on the right shows the output of the script, which is a JSON array of three student records. Each record is an array with four elements: ID, name, and phone number.

```
6 </head>
7 <body>
8 <h4>使用数组方式接收查询数据[对象格式]</h4><hr />
9 <?php
10     require_once 'example9_6_connect.php';
11     $query = "select * from student";
12     $result = @$link->query($query);
13     //$row = $result->fetch_array();
14     //$row = $result->fetch_array(MYSQLI_ASSOC);
15     //$row = $result->fetch_array(MYSQLI_NUM);
16     //$row = $result->fetch_array(MYSQLI_BOTH);
17     //$row = $result->fetch_row();
18     //$row = $result->fetch_assoc();
19     $row = $result->fetch_all();
20     //$row = $result->fetch_all(MYSQLI_ASSOC);
21     //$row = $result->fetch_all(MYSQLI_NUM);
22     //$row = $result->fetch_all(MYSQLI_BOTH);
23     echo '<pre>'; print_r($row); echo '</pre>';
24     $result->free();
25     $link->close();
26 ?>
27 </body>
28 </html>
```

Internal Web Browser
http://localhost/example/chapter1
Array
(
 [0] => Array
 (
 [0] => 1
 [1] => 20170001
 [2] => 李木子
 [3] => 13909875678
)
 [1] => Array
 (
 [0] => 2
 [1] => 20170002
 [2] => 木子
 [3] => 13276543256
)
 [2] => Array
 (
 [0] => 3
 [1] => 20170003
 [2] => 李木
 [3] => 13234657890
)
)

7.3 PHP与MySQL的交互

(2) 对象方式

【例】使用对象接收MySQL查询数据。

用对象方式接收来自于查询结果集的数据，使用的是`mysqli_fetch_object()`函数。该函数从结果集中取得当前行，并作为对象返回。

The image shows a PHP script in a text editor and its output in a web browser. The PHP script, named `example9_11.php`, connects to a MySQL database and queries a table named `student`. It uses `mysqli_fetch_object()` to fetch the first row of the result set as an object. The script then echoes the values of `student_no`, `student_name`, and `student_tel` from the object. A green callout box points to the `mysqli_fetch_object()` function call, labeling it as "mysqli_result类的对象". The web browser output shows the title "使用对象方式接收查询数据" and the data "20170001 李木子 13909875678". A red arrow points from the `student_tel` property access in the PHP code to the corresponding value in the browser output.

```
example9_11.php
9 <?php
10 require_once 'example9_4_connect.php';
11 $query = "select * from student";
12 $result = mysqli_query($link, $query);
13
14 $row = mysqli_fetch_object($result);
15 // $row = $result->fetch_object();
16 echo $row->student_no. '    ';
17 echo $row->student_name. '    ';
18 echo $row->student_tel. '    ';
19
20 mysqli_free_result($result);
21 mysqli_close($link);
22 ?>
```

mysqli_result类的对象

Internal Web Browser
http://localhost/example

使用对象方式接收查询数据

20170001 李木子 13909875678

7.3 PHP与MySQL的交互

3、处理其他查询结果

在上述小节中，处理了返回记录的查询结果集，但并非所有查询返回的都是记录，比如数据的插入、更改与删除查询，返回的就是整数值。显然，这样的返回结果是不能用数组及对象来处理的。

(1) 确定返回的记录数

执行select查询以后，若要知道返回的记录数，可以调用mysql_num_rows()函数或访问mysqli_result类对象的num_rows属性。

(2) 确定受影响的行数

使用mysql_num_rows()函数获取结果集中行的数目，仅对select是有效的。如果要取得被insert、update、delete查询所影响到的行的数目，需要使用mysql_affected_rows()函数，或者访问mysqli类对象的affected_rows属性。

7.3 PHP与MySQL的交互

【例】 获取结果集中记录总数及执行查询后受影响的行数。

```
example9_12.php
10 require_once 'example9_6_connect.php';
11 $query = "select * from student";
12 $result = $link->query($query);
13 // $num = mysqli_num_rows($result);
14 $num = $result->num_rows;
15 // $num = mysqli_affected_rows($link);
16 // $num = $link->affected_rows;
17 echo '<table border="1" width="100%" rules="all">'.
18     '<caption>学生信息</caption>'. '<tr>'. '<th>学号</th>'.
19     '<th>姓名</th>'. '<th>联系方式</th>'. '</tr>';
20 while ($num){
21     $row = $result->fetch_object();
22     echo '<tr><td>'. $row->student_no. '</td>';
23     echo '<td>'. $row->student_name. '</td>';
24     echo '<td>'. $row->student_tel. '</td></tr>';
25     $num--;
26 }
27 echo '</table>';
28 $sql = "update student set student_tel='123456'".
29     " where student_tel is null";
30 $res = $link->query($sql);
31 if($res) echo '成功修改'. $link->affected_rows. '条学生信息!';
32 $result->free();
33 $link->close();
34 ?>
```

Internal Web Browser

http://localhost/example/chapter08

从结果集中获取记录数及受影响的行数

学生信息

学号	姓名	联系方式
20170001	李木子	13909875678
20170002	木子	02786543266
20170003	李木	13234657890
20170004	李子木	
20170005	木子李	

成功修改2条学生信息!

7.3 PHP与MySQL的交互

4、处理准备语句

在PHP与MySQL的交互过程中，通常会**重复执行一个查询**，可以使用PHP对MySQL数据库**准备(Prepared)语句**的扩展支持。

MySQL数据库准备语句的**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**。因此，可以向一个相同的准备语句发送大量的数据，大大提高了查询执行速度。

这个特性常用于**批量数据的插入操作**。

7.3 PHP与MySQL的交互

【例】使用准备语句插入多条数据。

```
example9_13.php
10  require_once 'example9_6_connect.php';
11  $query = "insert into student values (null,?,?,?)";
12  $stmt = $link->stmt_init();
13  $stmt->prepare($query);
14  $sno = null;
15  $stmt->bind_param('sss', $sno, $sname, $stel);
16  $sno = '20170010'; $sname = '梦林'; $stel = '12345678';
17  $stmt->execute();
18  $sno = '20170011'; $sname = '梦夕'; $stel = '123456789';
19  $stmt->execute();
20  $query = "select * from student";
21  $stmt = $link->prepare($query);
22  $stmt->execute();
23  $sid = null;
24  $stmt->bind_result($sid, $sno, $sname, $stel);
25  echo '<table border="1" width="100%" rules="all">';
26  echo '<caption>学生信息</caption>'. '<tr>'. '<th>学号</th>'.
27  echo '<th>姓名</th>'. '<th>联系方式</th>'. '</tr>';
28  while($stmt->fetch()){
29  echo '<tr><td>'. $sno. '</td>';
30  echo '<td>'. $sname. '</td>';
31  echo '<td>'. $stel. '</td></tr>';}
32  echo '</table>';
33  $stmt->close();
34  $link->close();
35  ?>
```

Internal Web Browser
http://localhost/example/chapter08/e/

使用MySQL准备语句

学生信息		
学号	姓名	联系方式
20170001	李木子	13909875678
20170002	木子	02786543266
20170003	李木	13234657890
20170004	李子木	123456
20170005	木子李	123456
20170010	梦林	12345678
20170011	梦夕	123456789

7.3 PHP与MySQL的交互

(1) 准备查询语句

准备语句中使用的**查询字符串**是一个**模板**，如第11行代码所示，其中**可能变化的数据位置**用**占位符(?)**来表示，在这些问号周围，不能再设置问号或其他分界符号。

(2) 创建准备对象

要使用准备语句进行MySQL数据库的查询操作，首先需要创建一个准备语句类的对象，即**mysqli_stmt**类的对象。如第12行代码所示。

stmt_init()函数是mysqli类的成员函数，其语法格式为：

```
mysqli_stmt mysqli::stmt_init(void)
```

该函数**返回**一个**mysqli_stmt**类的对象。

7.3 PHP与MySQL的交互

(3) 将查询语句放入准备对象

调用**mysqli_stmt**类的**prepare()**成员函数，完成要执行的查询语句的准备。
如第13行代码所示。

(4) 绑定参数

调用**mysqli_stmt**类的**bind_param()**成员函数，绑定查询语句中的参数。

其语法格式为：

```
bool mysqli_stmt::bind_param(string $type,mixed &$v1[,mixed &$v2 ...])
```

其中，参数**type**表示其后各个变量(由**&\$v1**、**&\$v2**、**...**、**&\$vn**表示)的**数据类型**，该参数为必须的，以确保向数据库服务器发送时能最有效地实现数据编码。

7.3 PHP与MySQL的交互

目前，支持**4种类型码**。

i: 所有INTEGER类型

d: DOUBLE和FLOAT类型

b: BLOB类型

s: 所有其他类型，包括字符串

参数`&$v1`、`&$v2`、...、`&$vn`是**与查询语句中“?”占位符相对应的变量**，**其数量与占位符数量相同**。

针对代码中的第15行，给type参数值为“sss”，表示后面的3个变量均为字符串；后面的变量`$sno`、`$sname`、`$stel`分别对应student数据表的`student_no`、`student_name`和`student_tel`字段。

7.3 PHP与MySQL的交互

(5) 执行准备语句

准备语句的执行是通过调用**mysqli_stmt**类的**execute()**成员函数来实现的。

如第17、19、22行所示。其语法格式为：

```
bool mysqli_stmt::execute ( void )
```

该函数返回TRUE或FALSE。

(6) 释放准备语句资源

一旦准备语句使用结束之后，它所占有的资源可以通过**mysqli_stmt**类的**close()**成员函数来释放。如第33行代码所示。

Part.4

7.4 PDO与MySQL的交互

7.4 PDO与MySQL的交互

所谓**PDO**，就是指**PHP的数据对象**，即**PHP Data Object**。

PDO扩展类库**为PHP访问数据库**定义了一个轻量级的、**一致性的接口**，它提供了一个**数据访问抽象层**，这样，**无论使用什么样的数据库**，都可以通过**一致的函数**执行查询和获取数据。

大大简化了PHP对数据库的操作，并能够屏蔽不同数据库之间的差异。使用PDO可以很方便地进行**跨数据库程序的开发**，以及**不同数据库间的移植**。

1、PDO扩展的启用

开启PHP配置文件php.ini中的如下配置项：

extension = php_pdo_mysql.dll

7.4 PDO与MySQL的交互

2、PDO对象的创建

使用PDO与不同数据库之间交互时，使用的操作函数都是相同的，都是PDO对象中的成员方法，所以在使用PDO与数据库交互之前，首先要创建一个PDO对象。

PDO类的构造方法原型如下：

```
PDO::__construct ( string $dsn [, string $username [, string $password [,  
array $driver_options ]]] )
```

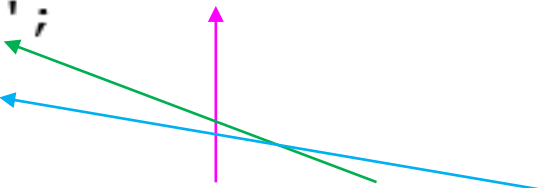
其中，参数dsn是必选项，表示数据源名称，它定义了一个确定的数据库和必须要用到的驱动程序。DSN的PDO命名惯例为PDO驱动程序的名称，后面为一个冒号，再后面是可选的驱动程序的数据库连接变量信息，如主机名、端口号和数据库名。

如连接MySQL服务器的DSN:mysql:host=localhost;dbname=test_students

7.4 PDO与MySQL的交互

【例】创建PDO对象。

```
<?php
    $dsn = "mysql:dbname=test_students;host=localhost";
    $user = 'root';
    $pwd = 'wvp';
    try {
        $pdo = new PDO($dsn, $user, $pwd);
    } catch (PDOException $e) {
        echo '数据库连接失败: ' . $e->getMessage();
        exit();
    }
?>
```



7.4 PDO与MySQL的交互

3、通过PDO执行查询

在PHP中，通过PDO执行SQL查询与数据库进行交互，可以使用**PDO类**或**PDOStatement类**的**成员函数**来实现。

(1) 使用PDO类的exec()方法

当执行**INSERT**、**UPDATE**和**DELETE**等**不返回结果集的查询**时，使用PDO对象中的**exec()**方法来实现。

该方法成功执行后，将**返回**执行查询操作而**受影响**的数据表中**数据记录**的**行数**。

7.4 PDO与MySQL的交互

【例】 使用PDO类的exec()方法执行查询。

The image shows a development environment with a code editor and a web browser. The code editor displays a PHP script named `example9_15.php` that uses PDO to execute an SQL insert query. The web browser shows the output of the script, which is a confirmation message. A table of student data is also visible in the bottom right corner.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8" >
5 <title>例9.15 使用PDO类的exec()方法执行查询</title>
6 </head>
7 <body>
8 <h4>使用PDO::exec()方法执行查询</h4><hr />
9 <?php
10     require_once 'example9_14_pdo.php';
11     $query = "insert into student value ".
12             "(null, '20170015', '王一', '15723645789')";
13     $insert = $pdo->exec($query);
14     if ($insert) {
15         echo '成功插入' . $insert . '条记录!';
16     } else {
17         echo '数据插入失败: ';
18         print_r($pdo->errorInfo());
19     }
20 ?>
21 </body>
22 </html>
```

Internal Web Browser
http://localhost/example/chapter
使用PDO::exec()方法执行查询
成功插入1条记录!

student_id	student_no	student_name	student_tel
1	20170001	李木子	13909875678
2	20170002	木子	02786543266
3	20170003	李木	13234657890
14	20170004	李子木	123456
15	20170005	木子李	123456
16	20170010	梦林	12345678
17	20170011	梦夕	123456789
26	20170015	王一	15723645789
*	(Auto)	(NULL)	(NULL)

7.4 PDO与MySQL的交互

(2) 使用PDO类的query()方法

当执行SELECT等返回结果集的查询时，使用PDO对象中的query()方法

```
9 <?php
10 require_once 'example9_14_pdo.php';
11 $query = "select * from student";
12 try {
13     $result = $pdo->query($query);
14     foreach ($result as $row) {
15         echo $row['student_no'].'&nbsp;&nbsp;&nbsp;';
16         echo $row['student_name'].'&nbsp;&nbsp;&nbsp;';
17         echo $row['student_tel'].'<br/>';
18     }
19 } catch (PDOException $e) {
20     echo $e->getMessage();
21 }
22 echo '<p>数据库中共有'.$result->rowCount().'条学生信息! </p>';
23 ?>
```

Internal Web Browser

http://localhost/exa...

使用PDO::query()方法执行查询

20170001	李木子	13909875678
20170002	木子	02786543266
20170003	李木	13234657890
20170004	李子木	123456
20170005	木子李	123456
20170010	梦林	12345678
20170011	梦夕	123456789
20170015	王一	15723645789

数据库中共有8条学生信息!

7.4 PDO与MySQL的交互

(3) 使用PDOStatement类的execute()方法

与mysqli扩展相同，PDO扩展也支持MySQL数据库的准备语句。

使用PDO类的prepare()方法以及PDOStatement类的execute()方法可以执行数据库的查询操作。

7.4 PDO与MySQL的交互

4、PDO对准备语句的支持

PDO扩展中对数据库准备语句的支持，是**通过PDOStatement类的对象来实现的**。所以，首先必须**创建PDOStatement类的对象**，然后通过对象**调用其成员方法**，**实现查询模板的导入、参数的绑定、查询的执行以及对结果集的处理等**。

(1) 创建PDOStatement对象

与其他类的实例化不同，PDOStatement对象**不是通过new运行符创建**，而是需要**调用PDO类的prepare()方法**。其语法格式为：

```
public PDOStatement PDO::prepare ( string $statement [, array  
$driver_options = array() ] )
```

7.4 PDO与MySQL的交互

(2) 绑定参数

如果查询语句中**使用了占位符**，就需要在每次执行查询前**将其替换成数据**。

通过**PDOStatement对象**中的**bindParam()方法**，把存储数据的变量绑定到准备好的占位符上。

其语法格式如下：

```
bool PDOStatement::bindParam ( mixed $parameter , mixed &$variable [,  
int $data_type = PDO::PARAM_STR [, int $length [, mixed  
$driver_options ]]] )
```

7.4 PDO与MySQL的交互

(3) 执行查询

【例】使用PDOStatement类的execute ()方法执行查询。

```
example9_17.php
9 <?php
10 require_once 'example9_14_pdo.php';
11 $query = "insert into student values (null,:sno,:sname,:stel)";
12 // $query = "insert into student values (null,?,?,?)";
13 $stmt = $pdo->prepare($query);
14 $stmt->bindParam(':sno', $sno);
15 $stmt->bindParam(':sname', $sname);
16 $stmt->bindParam(':stel', $stel);
17 /* $stmt->bindParam(1, $sno);
18 $stmt->bindParam(2, $sname);
19 $stmt->bindParam(3, $stel); */
20 $sno = '20160001';
21 $sname = '赵四';
22 $stel = '13123456789';
23 $stmt->execute();
24 $sno = '20160002';
25 $sname = '赵五';
26 $stel = '13123456788';
27 $stmt->execute();
28 ?>
```

Internal Web Brow...
http://loca...
使用
PDOStatement::execute
()方法执行查询

1 信息 2 表数据 3 信息

	student_id	student_no	student_name	student_tel
<input type="checkbox"/>		1 20170001	李木子	13909875678
<input type="checkbox"/>		2 20170002	木子	02786543266
<input type="checkbox"/>		3 20170003	李木	13234657890
<input type="checkbox"/>		14 20170004	李子木	123456
<input type="checkbox"/>		15 20170005	木子李	123456
<input type="checkbox"/>		16 20170010	梦林	12345678
<input type="checkbox"/>		17 20170011	梦夕	123456789
<input type="checkbox"/>		26 20170015	王一	15723645789
<input type="checkbox"/>		28 20160001	赵四	13123456789
<input type="checkbox"/>		29 20160002	赵五	13123456788
*	(Auto)	(NULL)	(NULL)	(NULL)

7.4 PDO与MySQL的交互

【例】使用数组方式输入查询数据。

example9_18.php

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8" >
5 <title>例8.18 使用数组方式输入查询数据</title>
6 </head>
7 <body>
8 <h4>使用数组方式输入查询数据</h4><hr />
9 <?php
10     require_once 'example9_14_pdo.php';
11     //$query = "insert into student values (null,:sno,:sname,:stel)";
12     $query = "insert into student values (null,?,?,?)";
13     $stmt = $pdo->prepare($query);
14     //$stmt->execute(array(":sno"=>'20150001',":sname"=>'刘一',":stel"=>'02187654377'));
15     //$stmt->execute(array(":sno"=>'20150002',":sname"=>'刘二',":stel"=>'02187654378'));
16     $stmt->execute(array('201500011','刘一','02187654377'));
17     $stmt->execute(array('201500021','刘二','02187654378'));
18 ?>
19 </body>
20 </html>
```


7.4 PDO与MySQL的交互

(4) 获取数据

PDO的数据获取方法与前面介绍的mysqli数据库扩展中使用的方法非常相似，只是获取数据的函数都来自于**PDOStatement类的成员**，比如**fetch()**方法、**fetchAll()**方法等。

fetch()

该方法可以将结果集中**当前行的记录**以某种方式返回，并将**结果集指针移到下一行**，当**到达结果集末尾时返回FALSE**。语法格式为：

```
mixed PDOStatement::fetch ([ int $fetch_style [, int $cursor_orientation =  
PDO::FETCH_ORI_NEXT [, int $cursor_offset = 0 ]]] )
```


7.4 PDO与MySQL的交互

【例】使用fetch()方法获取查询数据。

```
example9_19.php
2 <html>
3 <head>
4 <meta charset="UTF-8" >
5 <title>例9.19 使用fetch()方法获取查询数据</title>
6 </head>
7 <body>
8 <h4>使用fetch()方法获取查询数据</h4><hr />
9 <?php
10     require_once 'example9_14_pdo.php';
11     $query = "select * from student";
12     $stmt = $pdo->query($query);
13     echo '<table border="1" width="100%" rules="all">'.
14         '<caption>学生信息</caption>'.<tr>'.<th>学号</th>'.
15         '<th>姓名</th>'.<th>联系方式</th>'.</tr>';
16     $row = $stmt->fetch(PDO::FETCH_NUM);
17     while($row){
18         list($sid, $sno, $sname, $stel) = $row;
19         echo '<tr><td>'. $sno. '</td>';
20         echo '<td>'. $sname. '</td>';
21         echo '<td>'. $stel. '</td></tr>';
22         $row = $stmt->fetch(PDO::FETCH_NUM);
23     }
24     echo '</table>';
25 ?>
```

Internal Web Browser

http://localhost/example/cl

使用fetch()方法获取查询数据

学生信息

学号	姓名	联系方式
20170001	李木子	13909875678
20170002	木子	02786543266
20170003	李木	13234657890
20170004	李子木	123456
20170005	木子李	123456
20170010	梦林	12345678
20170011	梦夕	123456789
20170015	王一	15723645789
20160001	赵 四	13123456789
20160002	赵五	13123456788
20150001	刘一	02187654377
20150002	刘二	02187654378
201500011	刘一1	021876543771
201500021	刘二1	021876543781

7.4 PDO与MySQL的交互

fetchAll()

此方法与上述的fetch()方法类似，唯一不同的是，该方法**只需调用一次就可以获取结果集中的所有行的数据，并返回一个二维数组**。其语法格式为：

```
array PDOStatement::fetchAll ([ int $fetch_style [, mixed $fetch_argument [,  
array $ctor_args = array() ]]] )
```

其中，参数fetch_style是可选项，其值及含义与fetch()方法中同名参数相同。

若要**返回单独一列所有的数据**，可以指定参数fetch_style的值为

PDO::FETCH_COLUMN。

7.4 PDO与MySQL的交互

【例】使用fetchAll()方法获取查询数据。

```
example9_20.php
2 <html>
3 <head>
4 <meta charset="UTF-8" >
5 <title>例9.20 使用fetchAll()方法获取查询数据</title>
6 </head>
7 <body>
8 <h4>使用fetchAll()方法获取查询数据</h4><hr />
9 <?php
10     require_once 'example9_14_pdo.php';
11     $query = "select * from student";
12     $stmt = $pdo->query($query);
13     echo '<table border="1" width="100%" rules="all">'.
14         '<caption>学生信息</caption>'. '<tr>'. '<th>学号</th>'.
15         '<th>姓名</th>'. '<th>联系方式</th>'. '</tr>';
16     $rows = $stmt->fetchAll(PDO::FETCH_ASSOC);
17     foreach ($rows as $row) {
18         echo '<tr><td>'. $row['student_no']. '</td>';
19         echo '<td>'. $row['student_name']. '</td>';
20         echo '<td>'. $row['student_tel']. '</td></tr>';
21     }
22     echo '</table>';
23 ?>
24 </body>
25 </html>
```

Internal Web Browser

http://localhost/example/ct

使用fetchAll()方法获取查询数据

学生信息

学号	姓名	联系方式
20170001	李木子	13909875678
20170002	木子	02786543266
20170003	李木	13234657890
20170004	李子木	123456
20170005	木子李	123456
20170010	梦林	12345678
20170011	梦夕	123456789
20170015	王一	15723645789
20160001	赵四	13123456789
20160002	赵五	13123456788
20150001	刘一	02187654377
20150002	刘二	02187654378
201500011	刘一1	021876543771
201500021	刘二1	021876543781

Part.5

7.5 应用实例

7.5 应用实例

1、数据库设计

The screenshot shows the phpMyAdmin web interface in a browser. The address bar indicates the URL is localhost/phpmyadmin/db_structure.php?server=1&db=db_phpweb. The interface displays the database structure for 'db_phpweb'. On the left, a tree view shows the database hierarchy, including 'db', 'db_books', 'db_phpweb', and 'test_userinfo'. The main panel shows a table list with columns: 表 (Table), 操作 (Operations), 行数 (Rows), 类型 (Type), 排序规则 (Collation), 大小 (Size), and 多余 (Extra). The table list includes 'test_focus', 'test_user', and 'test_userinfo'. Below the table list, there is a '新建数据表' (New Table) section with input fields for '名字' (Name) and '字段数' (Number of fields), and an '执行' (Execute) button.

表	操作	行数	类型	排序规则	大小	多余
<input type="checkbox"/> test_focus	★ 浏览 结构 搜索 插入 清空 删除	10	InnoDB	utf8_general_ci	16 KB	-
<input type="checkbox"/> test_user	★ 浏览 结构 搜索 插入 清空 删除	2	InnoDB	utf8_general_ci	16 KB	-
<input type="checkbox"/> test_userinfo	★ 浏览 结构 搜索 插入 清空 删除	0	InnoDB	utf8_general_ci	16 KB	-
3 张表	总计	12	InnoDB	utf8_general_ci	48 KB	0 字节

新建数据表

名字: 字段数: 4

执行

7.5 应用实例

localhost / localhost / db_php X

localhost/phpmyadmin/tbl_structure.php?db=db_phpweb&table=test_focus

phpMyAdmin

服务器: localhost » 数据库: db_phpweb » 表: test_focus

浏览 结构 SQL 搜索 插入 导出 导入 权限 操作 触发器

表结构 关联视图

#	名字	类型	排序规则	属性	空	默认	注释	额外	操作
1	id	int(10)		UNSIGNED	否	无		AUTO_INCREMENT	修改 删除 主键 唯一 索引 空间 更多
2	title	varchar(100)	utf8_general_ci		否	无			修改 删除 主键 唯一 索引 空间 更多
3	author	char(30)	utf8_general_ci						索引 空间 更多
4	pub	timestamp							索引 空间 更多
5	visits	int(11)							索引 空间 更多

正在显示第 0 - 9 行 (共 10 行, 查询花费 0.0003 秒。)

SELECT * FROM 'test_focus'

性能分析 [编辑内]

显示全部 | 行数: 25 | 过滤行: 在表中搜索 | 按索引排序: 无

添加 1 个字段 于 visits 之后

索引

操作	键名	类型	唯一	紧凑
编辑 删除	PRIMARY	BTREE	是	否

在 1 个字段创建索引 执行

	id	title	author	pub	visits
<input type="checkbox"/> 编辑 复制 删除	1	近期关注测试数据0001	微梦	2018-04-05 09:23:48	68
<input type="checkbox"/> 编辑 复制 删除	2	近期关注测试数据0002	微梦	2018-04-05 10:13:27	67
<input type="checkbox"/> 编辑 复制 删除	3	近期关注测试数据0003	微梦	2018-04-05 10:23:37	66
<input type="checkbox"/> 编辑 复制 删除	4	近期关注测试数据0004	微梦	2018-04-05 12:23:48	67
<input type="checkbox"/> 编辑 复制 删除	5	近期关注测试数据0005	微梦	2018-04-05 13:13:27	66
<input type="checkbox"/> 编辑 复制 删除	6	近期关注测试数据0006	微梦	2018-04-05 12:30:22	66
<input type="checkbox"/> 编辑 复制 删除	7	近期关注测试数据0007	微梦	2018-04-05 12:23:48	66
<input type="checkbox"/> 编辑 复制 删除	8	近期关注测试数据0008	微梦	2018-04-05 06:29:27	66
<input type="checkbox"/> 编辑 复制 删除	9	近期关注测试数据0009	微梦	2018-04-05 05:23:37	66
<input type="checkbox"/> 编辑 复制 删除	10	近期关注测试数据0010	微梦	2018-04-05 16:23:48	66

7.5 应用实例

2、数据库操作基类设计

- 创建名为config.php的配置文件;
- 创建名为DBMySQL.class.php的PHP文件, 用来定义用于数据库连接的类。

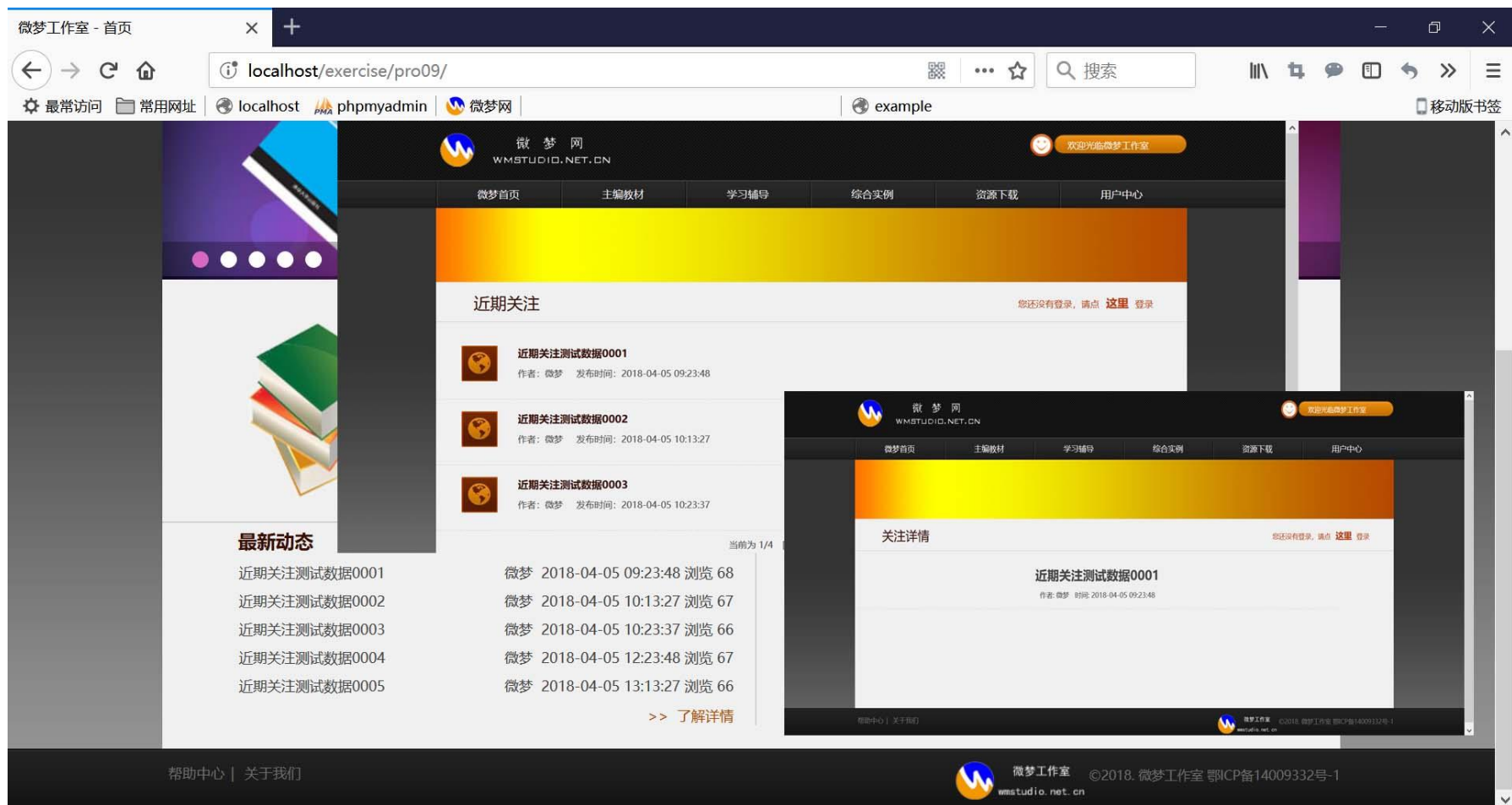
7.5 应用实例

3、模型类设计

- 打开项目system/library目录下的模型基类文件Model.class.php，修改其代码。
- 打开项目module/default/model目录下的模型文件FocusModel.class.php，修改代码。

7.5 应用实例

4、运行测试



总结

总结

★PHP与MySQL的连接

★PHP与MySQL的交互

★PDO与MySQL的交互