# 数据库抽象层PDO

PDO(php data object)扩展类库为php访问数据库定义了轻量级的、一致性的接口，它提供了一个数据库访问抽象层，这样，无论你使用什么数据库，都可以通过一致的函数执行查询和获取数据，大大简化了数据库的捉拿和，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，是将来php在数据库处理方面的主要发展方向，它可以支持mysql,postgresql,oracle,mssql等多种数据库

### PDO的安装

编辑php.ini文件:

extension=php\_pdo.dll

extension=php\_pdo\_mysql.dll

2.重启apache服务:

httpd –k restart

3.打开phpinfo.php查看是否有pdo

# 2.创建PDO对象

/\*连接如果失败，使用异常处理模式进行捕获 \*/

try{

$dbh = new PDO("OCI:dbname=accounts;charset=UTF-8", "scott", "tiger");

}catch(PDOException $e) {

echo "数据库连接失败： " .$e->getMessage();

}

$dsn = 'mysql:dbname=testdb;host=127.0.0.1'; //连接MySQL数据库的DSN

$user = 'dbuser'; //MySQL数据库的用户名

$password = 'dbpass'; //MySQL数据库的密码

try {

$dbh = new PDO($dsn, $user, $password);

} catch (PDOException $e) {

echo '数据库连接失败： ' . $e->getMessage();

}

Try{}cache{}不懂的话就去看php错误提示。

try {

$dbh = new PDO('uri:file:///usr/local/dbconnect', 'webuser', 'password');

} catch (PDOException $e) {

echo '连接失败： ' . $e->getMessage();

}

try {

//使用php.ini文件中的oraclepdo别名

$dbh = new PDO("oraclepdo", "scott", "tiger");

} catch (PDOException $e) {

echo "数据库连接失败： " .$e->getMessage();

}

//设置持久连接的选项数组作为最后一个参数,可以一起设置多个元素

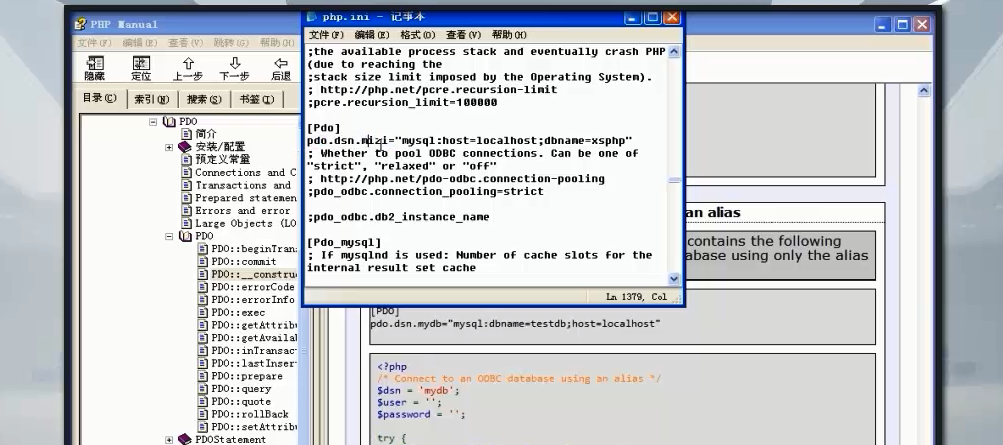
$opt = array(PDO::ATTR\_PERSISTENT => true);

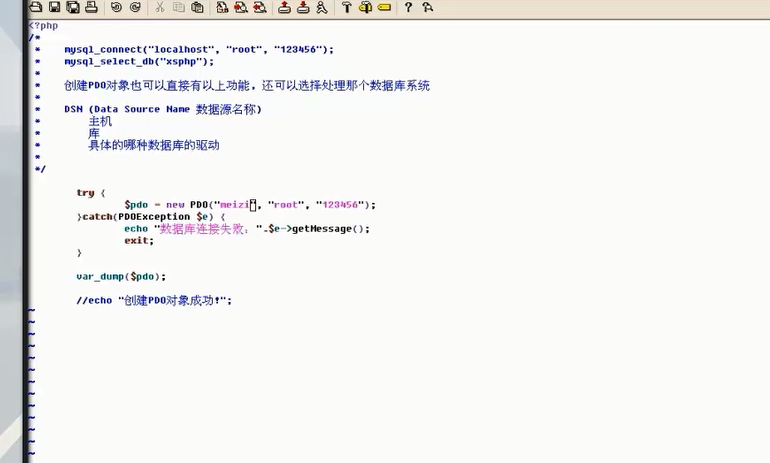
try {

$db = new PDO('mysql:host=localhost;dbname=test'，'dbuser'，'passwrod',$opt);

} catch (PDOException $e) {

echo "数据库连接失败： " .$e->getMessage();





# PDO与连接有关的选项

PDO::ATTR\_ERRMODE

1.PDO::ERRMODE\_SILENT 0

2.PDO::ERRMODE\_WARNING 1

3.PDO::ERRMODE\_EXCEPTION 2

PDO::ATTR\_AUTOCOMMIT

1.0 //关闭自动提交

2.1 //开启自动提交

PDO::ATTR\_DEFAULT\_FETCH\_MODE

1.PDO::FETCH\_ASSOC 2

2.PDO::FETCH\_NUM 3

3.PDO::FETCH\_BOTH 4

4.PDO::FETCH\_OBJ 5

$pdo->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

//$pdo->setAttribute(3,2);

$pdo->setAttribute(PDO::ATTR\_AUTOCOMMIT,0);//$pdo->setAttribute(0,0);

$pdo->setAttribute(PDO::ATTR\_DEFAULT\_FETCH\_MODE, PDO::FETCH\_ASSOC);

//$pdo->setAttribute(19,2);

echo "\nPDO是否关闭自动提交功能：". $pdo->getAttribute(PDO::ATTR\_AUTOCOMMIT);

echo "\n当前PDO的错误处理的模式：". $pdo->getAttribute(PDO::ATTR\_ERRMODE);

echo "\n表字段字符的大小写转换： ". $pdo->getAttribute(PDO::ATTR\_CASE);

echo "\n与连接状态相关特有信息： ". $pdo->getAttribute(PDO::ATTR\_CONNECTION\_STATUS);

echo "\n空字符串转换为SQL的null：". $pdo->getAttribute(PDO::ATTR\_ORACLE\_NULLS);

echo "\n应用程序提前获取数据大小：".$pdo->getAttribute(PDO::ATTR\_PERSISTENT);

echo "\n与数据库特有的服务器信息：".$pdo->getAttribute(PDO::ATTR\_SERVER\_INFO);

echo "\n数据库服务器版本号信息：". $pdo->getAttribute(PDO::ATTR\_SERVER\_VERSION);

echo "\n数据库客户端版本号信息：". $pdo->getAttribute(PDO::ATTR\_CLIENT\_VERSION);

# PDO的错误处理模式

PDO::ATTR\_ERRMODE=>错误处理模式:(3)

PDO::ERRMODE\_SILENT:不报错误（忽略）(0)

PDO::ERRMODE\_WARNING:以警告的方式报错(1)

PDO::ERRMODE\_EXCEPTION：以异常的方式报错(推荐使用)(2)

设置方式：2种

在构造时初始化错误模式

通过pdo对象的setAttribute（）方法设置。

$pdo->setAttribute(PDO::ATTR\_ERRMODE,PDO::ERRMODE\_EXCEPTION);

面向过程

$con->errorCode()：SQL语句错误代码

$con->errorInfo()：错误信息

面向对象

捕获PDOException异常对象

<?php

/\*

\* 1. 默认的错误模式（不提示, 我们看到到问题. 被忽视!）

\*

\*

\*/

try {

$pdo = new PDO("mysql:host=localhost;dbname=xsphp", "root", "123456");

}catch(PDOException $e) {

echo "数据库连接失败：".$e->getMessage();

exit;

}

//更改模式

$pdo -> setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_WARNING);

echo $pdo->getAttribute(PDO::ATTR\_ERRMODE)."<br>";

$affected\_rows = @$pdo -> exec("delete from helloworld");

if(!$affected\_rows) {

echo $pdo->errorCode();

echo '<br>';

print\_r($pdo->errorinfo());

exit;

}

echo "OK!";

?>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

下面的这个模式是常用的。异常模式。ERRMODE\_EXCEPTION

<?php

/\*

\* setAttribute();

\* PDO::ATTR\_ERRMODE

\*

\* 1. 默认的错误模式（不提示, 我们看到到问题. 被忽视!）

\* 2. 警告模式 PDO::ERRMODE\_WARNING

\* 3. 异常的模式 PDO::ERRMODE\_EXCEPTION

\*

\*/

try {

//创建对象

$pdo = new PDO("mysql:host=localhost;dbname=xsphp", "root", "123456");

//设置错误使用异常的模式

$pdo -> setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

}catch(PDOException $e) {

echo "数据库连接失败：".$e->getMessage();

exit;

}

try {

//使用PDO中的方法执行语句

}catch(PDOException $e) {

echo "错误：".$e->getMessage();

}

?>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<?php

/\*

\* setAttribute();

\* PDO::ATTR\_ERRMODE

\* 1. 默认的错误模式（不提示, 我们看到到问题. 被忽视!）

\* 2. 警告模式 PDO::ERRMODE\_WARNING

\* 3. 异常的模式 PDO::ERRMODE\_EXCEPTION

\*

\*/

try {

$pdo = new PDO("mysql:host=localhost;dbname=xsphp", "root", "123456");

}catch(PDOException $e) {

echo "数据库连接失败：".$e->getMessage();

exit;

}

$pdo -> setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

echo $pdo->getAttribute(PDO::ATTR\_ERRMODE)."<br>";

try {

$affected\_rows = $pdo -> exec("delete from helloworld");

}catch(PDOException $e) {

echo "错误原因：".$e->getMessage();

}

echo "OK!";

6.使用PDO执行SQL语句

<?php

try{

$dbh = new PDO('mysql:dbname=testdb;host=localhost', 'mysql\_user', 'mysql\_pwd');

}catch(PDOException $e){

exit('数据库连接失败：'.$e->getMessage());

}

$query = “UPDATE contactInfo SET phone=‘15801680168’ where name=‘高某某’”;  
 //使用exec()方法可以执行INSERT、UPDATE和DELETE等。 exec()是影响行数的函数

$affected = $dbh->exec($query);

if($affected){

echo '数据表contactInfo中受影响的行数为：'.$affected;

}else{

print\_r($dbh->errorInfo());

}

\*\*\*\*\*\*\*\*\*

<?php

$dbh = new PDO('mysql:dbname=testdb;host=localhost', 'mysql\_user', 'mysql\_pwd');

$dbh->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$query =

"SELECT name, phone, email FROM contactInfo WHERE departmentId='D01'";

try {

//执行SELECT查询，并返回PDOstatement对象

$pdostatement = $dbh->query($query);

echo "一共从表中获取到".$pdostatement->rowCount()."条记录:\n";

foreach ($pdostatement as $row) {　　 //从PDOstatement对象中遍历结果

echo $row['name'] . "\t"; //输出从表中获取到的联系人的名字

echo $row['phone'] . "\t";　//输出从表中获取到的联系人的电话

echo $row['email'] . "\n";　　　　　//输出从表中获取到的联系人的电子邮件

}

} catch (PDOException $e) {

echo $e->getMessage();

}

# MySQL的事务处理

事务：将多条sql操作（增删改）作为一个操作单元，要么都成功，要么都失败。

MySQL对事务的支持:

被操作的表必须是innoDB类型的表（支持事务）

MySQL常用的表类型：MyISAM(非事务)增删改速度快、InnodB（事务型）安全性高

更改表的类型为innoDB类型

mysql> alter table stu engine=innodb;

Query OK, 29 rows affected (0.34 sec)

Records: 29 Duplicates: 0 Warnings: 0

mysql> show create table stu\G; //查看表结构

**MySQL的事务处理**

Mysql> set autocommit = 0;//设置自动提交

mysql>start transaction;//

Mysql>commit;

Mysql>roolback;

# 构建事务处理的应用程序

开启一次事务:

$pdo->beginTransaction();

提交一次事务:

$pdo->commit();

回滚一次事务:

$pdo->rollback();

注意如下设置:

1.$pdo->setAttribute(PDO::ATTR\_AUTOCOMMIT,0);

2.$pdo->setAttribute(PDO::ATTR\_AUTOCOMMIT,1);

### 代码部分

<?php

/\*

\* PDO中执行SQL语句的方法有两个主要的:

\*

\* 1. exec() 用来处理非结果集的 insert update delete create ....

\*

\* 返回影响的函数

\*

\* 2. query() 用来处理有结果集的语句 select desc show

\* set names utf8;

\*

\* $pdo -> query("set names utf8");

\* $pdo -> exec("set names utf8");

\*/

try {

//创建对象

$pdo = new PDO("mysql:host=localhost;dbname=xsphp", "root", "123456");

//设置错误使用异常的模式

$pdo -> setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

}catch(PDOException $e) {

echo "数据库连接失败：".$e->getMessage();

exit;

}

try {

//使用PDO中的方法执行语句

$affected\_rows = $pdo -> exec("insert into users(name, pass, sex, age, email) values('aaa', 'bbb', 'nn', '11', 'aaa@b.com')");

echo $affected\_rows."<br>";

echo $pdo->lastinsertid();

}catch(PDOException $e) {

echo "错误：".$e->getMessage();

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<?php

/\*

\* PDO中执行SQL语句的方法有两个主要的:

\* 1. exec() 用来处理非结果集的 insert update delete create ....

\* 返回影响的函数

\* 如果是插入语句可以使用lastinsertid()方法获取最后自动插入id

\* 2. query() 用来处理有结果集的语句 select desc show

\* 返回来的是 PDOStatement类的对象， 再通过这个类的方法，获取结果。 也可以直接foreach遍历获取结果（但不常用）

\*

\* set names utf8;

\*

\* $pdo -> query("set names utf8");

\* $pdo -> exec("set names utf8");

\*/

try {

//创建对象

$pdo = new PDO("mysql:host=localhost;dbname=xsphp",

"root", "123456");

//设置错误使用异常的模式

$pdo -> setAttribute(PDO::ATTR\_ERRMODE,

PDO::ERRMODE\_EXCEPTION);

//关闭自动提交

$pdo-> setAttribute(PDO::ATTR\_AUTOCOMMIT, 0);

}catch(PDOException $e) {

echo "数据库连接失败：".$e->getMessage();

exit;

}

try {

//开启一个事务

$pdo -> beginTransaction();

$price = 50;

//妹子转出50元

$affected\_rows = $pdo-> exec("update demo set

ye=ye-{$price} where id=1");

if($affected\_rows > 0) {

echo "妹子转出{$price}元成功!<br>";

} else {

throw new PDOException("妹子转出失败！<br>");

}

//峰哥会收到50元

$affected\_rows = $pdo-> exec("update demo set ye=ye

+{$price} where id=3");

if($affected\_rows) {

echo "峰哥收到{$price}元成功!";

}else {

throw new PDOException("峰哥收入失败！

<br>");

}

echo "交易成功!<br>";

//提交以上的操作

$pdo->commit();

}catch(PDOException $e) {

echo "错误：".$e->getMessage();

echo "交易失败!<br>";

//撤销所有操作

$pdo -> rollback();

}

//运行完成以后， 最后开启自动提交

$pdo-> setAttribute(PDO::ATTR\_AUTOCOMMIT, 1);

# \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# PDO对预处理语句的支持

PDOStatement对象的方法

准备语句

绑定参数

执行预处理方式

预处理查询

# 使用PDO准备语句并执行语句

得到pdo预处理对象的方法:

$sql=“select \* from user order by id”;

$sth=$pdo->prepare($sql);

以上代码中的$sth即为预处理对象

在PDO中参数式的SQL语句有两种(预处理sql)：

1.insert into stu(id,name) value(?,?); //?号式(适合参数少的)

2.insert into stu(id,name) value(:id,:name);//别名式(适合参数多的)

### 代码

<?php

/\*

\* PHP的预处理语句 (做到)

\*

\* 效率要提高

\*

\* 安全性要好

\*

\* 建议: 使用这种方式去执行SQL语句

\*

\*/

try {

//创建对象

$pdo = new PDO("mysql:host=localhost;dbname=xsphp", "root", "123456");

//设置错误使用异常的模式

$pdo -> setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$pdo -> query("set names utf8");

}catch(PDOException $e) {

echo "数据库连接失败：".$e->getMessage();

exit;

}

try {

//给数据库管理系统并直接执行

//$pdo -> query("select \* from users");

//只是将这个语句放到服务器上（数据库管理系统）上, 编写后等待，没有执行

$stmt = $pdo -> prepare("insert into users(name, pass, age, sex, email) values(?, ?, ?, ?, ?)");

//绑定参数（？），将问号和一个变量关联起来

$stmt -> bindParam(1, $name);

$stmt -> bindParam(2, $pass);

$stmt -> bindParam(3, $age);

$stmt -> bindParam(4, $sex);

$stmt -> bindParam(5, $email);

//给变量一个值， 就会给准好的语句中的对应?一个值

$name = "admin1";

$pass = "hello1";

$age = 1001;

$sex = "nnn1";

$email = "aaa@bbb.com1";

//执行上面在数据库系统中准备好的语句

$stmt -> execute();

//给变量一个值， 就会给准好的语句中的对应?一个值

$name = "admin2";

$pass = "hello2";

$age = 1002;

$sex = "nnn2";

$email = "aaa@bbb.com2";

//执行上面在数据库系统中准备好的语句

$stmt -> execute();

}catch(PDOException $e) {

echo "错误：".$e->getMessage();

}

# 使用PDO准备语句并执行语句

1.

<?php

/\*

\* PHP的预处理语句 (做到)

\*

\* 效率要提高

\*

\* 安全性要好

\*

\* 建议: 使用这种方式去执行SQL语句

\*

\*/

try {

//创建对象

$pdo = new PDO("mysql:host=localhost;dbname=xsphp", "root", "123456");

//设置错误使用异常的模式

$pdo -> setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$pdo -> query("set names utf8");

}catch(PDOException $e) {

echo "数据库连接失败：".$e->getMessage();

exit;

}

try {

//给数据库管理系统并直接执行

//$pdo -> query("select \* from users");

//只是将这个语句放到服务器上（数据库管理系统）上, 编写后等待，没有执行

$stmt = $pdo -> prepare("insert into users(name, pass, age, sex, email) values(:name, :pass, :age, :sex, :email)");

//绑定参数（？），将问号和一个变量关联起来

$stmt -> bindParam("name", $name, PDO::PARAM\_STR);

$stmt -> bindParam("pass", $pass, PDO::PARAM\_STR);

$stmt -> bindParam("age", $age, PDO::PARAM\_INT);

$stmt -> bindParam("sex", $sex, PDO::PARAM\_STR);

$stmt -> bindParam("email", $email, PDO::PARAM\_STR);

//给变量一个值， 就会给准好的语句中的对应?一个值

$name = "admin1";

$pass = "hello1";

$age = 1001;

$sex = "nnn1";

$email = "aaa@bbb.com1";

//执行上面在数据库系统中准备好的语句

$stmt -> execute();

//给变量一个值， 就会给准好的语句中的对应?一个值

$name = "admin2";

$pass = "hello2";

$age = 1002;

$sex = "nnn2";

$email = "aaa@bbb.com2";

//执行上面在数据库系统中准备好的语句

$stmt -> execute();

}catch(PDOException $e) {

echo "错误：".$e->getMessage();

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2.

<?php

try {

//创建对象

$pdo = new PDO("mysql:host=localhost;dbname=xsphp", "root", "123456");

//设置错误使用异常的模式

$pdo -> setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$pdo -> query("set names utf8");

}catch(PDOException $e) {

echo "数据库连接失败：".$e->getMessage();

exit;

}

try {

//给数据库管理系统并直接执行

//$pdo -> query("select \* from users");

//只是将这个语句放到服务器上（数据库管理系统）上, 编写后等待，没有执行

// $stmt = $pdo -> prepare("insert into users(name, pass, age, sex, email) values(:name, :pass, :age, :sex, :email)");

$stmt = $pdo -> prepare("insert into users(name, pass, age, sex, email) values(?, ? ,?, ?, ?)");

$stmt -> execute(array("meizi", '123456', 18, 'nv', 'aa@bb.com'));

$stmt -> execute(array("meizi2", '123456', 18, 'nv', 'aa@bb.com'));

$stmt -> execute(array("meizi22", '123456', 18, 'nv', 'aa@bb.com'));

}catch(PDOException $e) {

echo "错误：".$e->getMessage();

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<?php

try {

//创建对象

$pdo = new PDO("mysql:host=localhost;dbname=xsphp", "root", "123456");

//设置错误使用异常的模式

$pdo -> setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$pdo -> query("set names utf8");

}catch(PDOException $e) {

echo "数据库连接失败：".$e->getMessage();

exit;

}

try {

//给数据库管理系统并直接执行

//$pdo -> query("select \* from users");

//只是将这个语句放到服务器上（数据库管理系统）上, 编写后等待，没有执行

$stmt = $pdo -> prepare("insert into users(name, pass, age, sex, email) values(:name, :pass, :age, :sex, :email)");

// $stmt = $pdo -> prepare("insert into users(name, pass, age, sex, email) values(?, ? ,?, ?, ?)");

$stmt -> execute(array("name"=>"feng", "pass"=>"abc123#", "age"=>28, "sex"=>"nan", "email"=>"aaa@bbb.com"));

$stmt -> execute(array("name"=>"fenga", "pass"=>"abc123#", "age"=>28, "sex"=>"nan", "email"=>"aaa@bbb.com"));

$stmt -> execute(array("name"=>"fengb", "pass"=>"abc123#", "age"=>28, "sex"=>"nan", "email"=>"aaa@bbb.com"));

}catch(PDOException $e) {

echo "错误：".$e->getMessage();

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<?php

try {

//创建对象

$pdo = new PDO("mysql:host=localhost;dbname=xsphp", "root", "123456");

//设置错误使用异常的模式

$pdo -> setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

$pdo -> query("set names utf8");

}catch(PDOException $e) {

echo "数据库连接失败：".$e->getMessage();

exit;

}

try {

//给数据库管理系统并直接执行

//$pdo -> query("select \* from users");

//只是将这个语句放到服务器上（数据库管理系统）上, 编写后等待，没有执行

$stmt = $pdo -> prepare("insert into users(name, pass, age, sex, email) values(:name, :pass, :age, :sex, :email)");

// $stmt = $pdo -> prepare("insert into users(name, pass, age, sex, email) values(?, ? ,?, ?, ?)");

$stmt -> execute($\_GET);

}catch(PDOException $e) {

echo "错误：".$e->getMessage();

}

# PDO的预处理查询

//声明一个SELECT查询，从表contactInfo中获取D01部门的四个字段的信息 $query = "SELECT uid, name, phone, email FROM contactInfo WHERE departmentId='D01'";

try {

$stmt = $dbh->prepare($query);　　　//准备声明好的一个查询

$stmt->execute();　　　　　　　　　 //执行准备好的查询

$stmt->bindColumn(1, $uid);　　　　 //通过列位置偏移数绑定变量

$uid $stmt->bindColumn(2, $name); //通过列位置偏移数绑定变量$name

$stmt->bindColumn('phone', $phone); //绑定列名称到变量$phone上

$stmt->bindColumn('email', $email);　//绑定列名称到变量$email上

while ($stmt->fetch(PDO::FETCH\_BOUND)) {

//fetch()方法传入特定的参数遍历

echo $uid."\t".$name."\t".$phone."\t".$email."\n";

//输出自动将列值赋给对应变量的值

}

} catch (PDOException $e) {

echo $e->getMessage();

}

# 11用PDO存取大数据对象

$dbh = new PDO('mysql:dbname=testdb;host=localhost', 'mysql\_user', 'mysql\_pwd');

$stmt = $dbh->prepare("INSERT INTO images(contenttype, imagedata) VALUES (?, ?)");

$fp = fopen($\_FILES['file']['tmp\_name'], 'rb'); //使用fopen()函数打开上传的文件

$stmt->bindParam(1, $\_FILES['file']['type']);

//将上传文件的MIME类型绑定到第一个参数中

$stmt->bindParam(2, $fp, PDO\_PARAM\_LOB);

//将上传文件的二进制数据和第二个参数绑定

$stmt->execute();

$dbh = new PDO('mysql:dbname=testdb;host=localhost', 'mysql\_user', 'mysql\_pwd');

$stmt = $dbh->prepare("SELECT contenttype, imagedata FROM images WHERE id=?");

$stmt->execute(array($\_GET['id']));

//通过表单中输入的ID值和参数绑定，并执行查询

list($type, $lob) = $stmt->fetch(); //获取结果集中的大数据类型和文件指针

header("Content-Type: $type");

//将从表中读取的大文件类型作为合适的报头发送

fpassthru($lob);