PDO

一、PDO 的作用

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

简单理解：用户 PHP 面向对象操作数据库中的数据。

选择 PDO 的原因：跨数据库，带预处理（防 SQL 注入），支持事务操作

二、PDO 的安装

1、PHP 扩展库 ext 中存在 php_pdo_mysql.dll，php_pdo.dll 文件

2、编辑 php.ini 文件:

extension=php_pdo.dll

extension=php_pdo_mysql.dll

3、重启 apache 服务:

httpd －k restart

4、打开 phpinfo.php 查看是否有 pdo

三、创建 PDO 对象

```
<?php
//数据库连接配置信息
define("DSN","mysql:host=localhost;dbname=lamp94");
define("USER","root");
define("PASS","");
?>
<?php
header('Content-type:text/html;charset=utf-8');
//创建 PDO 对象，实现数据库的连接
require("dbconfig.php");

//1. 创建 PDO 对象，实现数据库的连接
try{                //"POD 驱动名:主机名;数据库,数据库用户库,数据库密码"
    //$pdo = new PDO("mysql:host=localhost;dbname=lamp94","root","");
    $pdo = new PDO(DSN,USER,PASS);
    $pdo->query("set names utf8");//设置字符集
}catch(PDOException $pe){//PDO 报错是异常方式,需要异常方式处理
    die('数据库连接失败！原因:'.$pe->getMessage());
}
//2.查询 user 表信息
$sql = "select * from user";
```

注意：处理返回的是 PDOStatement 对象（预处理对象）赋给$stmt 变量

```
$stmt = $pdo->query($sql);

//3.解析并输出数据结果。
while($ob = $stmt->fetch(PDO::FETCH_ASSOC)){
    echo $ob['id'].":".$ob['username']."<br/>    ";
}
//4.释放资源
//$pdo=null; //释放资源
```

四、在 PDO 操作中使用的类

PDO 类

PDOStatement 预处理对象类

PODException 异常处理类

五、SQL 注入----原本是值，却提交的是 SQL 语句（说白了就值中带有 SQL 语句）

```
$username = “zhangsan”;
$username = “;drop database lamp94;”;   ‘ or 1=1 or ‘’=”    SQL 注入

$pass = “123”;
$sql = “selct * from user where username=’{$username} ‘and pass=’{$pass}’”;

$sql = “selct * from user where username=’;drop database lamp94;‘and pass=’{$pass}’”;
SQL 注入
mysql_query($sql);
```

六、PDO::__construct — 创建一个表示数据库连接的 PDO 实例

```
    driver_options: 配置选项:
    如:  PDO::ATTR_PERSISTENT=>true,是否开启持久链接
     *PDO::ATTR_ERRMODE=>错误处理模式: (可以是以下三个)(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)

$pdo =  new PDO("mysql:host=localhost;dbname=lamp36db","root","root");
$pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
其他方法:
------------------------------------------------------------
1. query($sql); 用于执行查询SQL语句。返回PDOStatement对象
2. exec($sql);  用于执行增、删、改操作，返回影响行数;
3. getAttribute(); 获取一个"数据库连接对象"属性。
4. setAttribute(); 设置一个"数据库连接对象"属性。
5. beginTransaction 开启一个事物（做一个回滚点）
6. commit   提交事务
7. rollBack 事务回滚操作。
8. errorCode    获取错误码
9. errorInfo    获取错误信息
10.lastInsertId  获取刚刚添加的主键值。
11.prepare  创建SQL的预处理，返回PDOStatement对象
12.quote    为sql字串添加单引号。
```

操作实例：

```php
<?php
//创建PDO对象，实现数据库的连接
require("dbconfig.php");

//1. 创建PDO对象，实现数据库的连接
try{
    //$pdo = new PDO("mysql:host=localhost;dbname=lamp94","root","");
    //构造方法：初始化
    $driver = array(
        //PDO::ATTR_PERSISTENT=>true,    //开启持久连接
        //PDO::ATTR_ERRMODE=>PDO::ERRMODE_SILENT,    //不报错模式(0)
        //PDO::ATTR_ERRMODE=>PDO::ERRMODE_WARING,    //警告模式(1)
        PDO::ATTR_ERRMODE=>PDO::ERRMODE_EXCEPTION,  //异常模式(2) 推荐
    );

    $pdo = new PDO(DSN,USER,PASS,$driver); //通过构造时来设置pdo的相关信息
    $pdo->query("set names utf8");

    //前期不指定错误模式可以通过setAttribute()方法设置
    $pod -> setAttribute();
}catch(PDOException $pe){ $pdo ->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_WARNING);
    die('数据库连接失败！原因:'.$pe->getMessage());
}
//2.查询stu表信息
$sql = "select * from stu12";
$stmt = $pdo->query($sql);

//3.解析并输出数据结果。
while($ob = $stmt->fetch(PDO::FETCH_ASSOC)){
    echo $ob['name'].":".$ob['sex']."<br/>  ";
}

//$pdo=null; //释放资源
```

七、PDO 的其他方法

1、使用 PDO 实现数据的增，删，改

```php
<?php
header('Content-type:text/html;charset=utf-8');
//使用PDO实现数据的增，删，改，查
require("dbconfig.php");
//1. 创建PDO对象，实现数据库的连接
try{

    $pdo = new PDO(DSN,USER,PASS);
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(异常模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    //$pdo->setAttribute(3,2);//此方法同上。

    $pdo->query("set names utf8");
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}
```

```php
//2.定义一条添加SQL语句并执行操作
//$sql = "insert into users values (null,'uuuu','1')";
//执行成功并返回受影响的行数
//$rows = $pdo->exec($sql);

//echo "添加成功{$rows}条数据<br>";
//echo "添加成功返回数据的自增ID".$pdo -> lastInsertId();
//============================================================
//2.定义一条修改SQL语句并执行操作
//$sql = "update users set name='oxxoxoxo',sex='1' where id in(2,3,4)";
//执行成功并返回受影响的行数
//$rows = $pdo->exec($sql);

//echo "修改成功{$rows}条数据<br>";

//============================================================
//2.定义一条删除SQL语句并执行操作
$sql = "delete from user where id > 70";
//执行成功并返回受影响的行数
$rows = $pdo->exec($sql);

echo "删除成功{$rows}条数据<br>";

//$pdo=null; //释放资源
```

2、PDOStatement 对象的方法

```
PDOstatement对象的方法：
----------------------------------------------------------------------
1、fetch() 返回结果集的下一行，结果指针下移，到头返回false 。
    参数:    PDO::FETCH_BOTH (default)、: 索引加关联数组模式
             PDO::FETCH_ASSOC、       : 关联数组模式
             PDO::FETCH_NUM、           : 索引数组模式
             PDO::FETCH_OBJ、           : 对象模式
             PDO::FETCH_LAZY          : 所有模式（SQL语句和对象）

2、fetchAll() 通过一次调用返回所有结果，结果是以数组形式保存
        参数: PDO::FETCH_BOTH (default)、
        PDO::FETCH_ASSOC、
        PDO::FETCH_NUM、
        PDO::FETCH_OBJ、
        PDO::FETCH_COLUMN表示取指定某一列，
        如: $rslist = $stmt->fetchAll(PDO::FETCH_COLUMN,2);取第三列
3、execute()    负责执行一个准备好了的预处理语句
4. fetchColumn()返回结果集中下一行某个列的值
5. setFetchMode()设置需要结果集合的类型
6. rowCount()   返回使用增、删、改、查操作语句后受影响的行总数
7. setAttribute()为一个预处理语句设置属性
8. getAttribute()获取一个声明的属性
9. errorCode()  获取错误码
10. errorInfo() 获取错误信息
11. bindParam() 将参数绑定到相应的查询占位符上
```

12. bindColumn() 用来匹配列名和一个指定的变量名，这样每次获取各行记录时，会自动将相应
13. bindValue() 将一值绑定到对应的一个参数中
14. nextRowset() 检查下一行集
15. columnCount() 在结果集中返回列的数目
16. getColumnMeta() 在结果集中返回某一列的属性信息
17. closeCursor() 关闭游标，使该声明再次执行

3、实例：PDOStatement 对象的方法

```php
<?php
//创建PDO对象，实现数据库的连接后数据查询
header("Content-type:text/html;charset=utf-8");
//1. 导入配置文件
require("dbconfig.php");

//2. 创建PDO对象，实现数据库的连接，并做属性设置
try{

    $pdo = new PDO(DSN,USER,PASS);//获取连接
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(异常模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    $pdo->query("set names utf8");//设置编码
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}

try{
    //3. 执行查询,并获取结果处理对象PDOStatment
    $sql = "select * from users";
    $stmt = $pdo->query($sql);

    //4. 此时stmt变量就是PDOStatment的对象，然后开始解析里面的结果
    //var_dump($stmt); //PDOStatement
    /*
    //第一种遍历数据结果方式
    foreach($stmt as $ob){
        echo $ob['id'].":".$ob['name'].":".$ob['sex']."<br/>";
    }
    */

    /*
    //第二中遍历
    $list = $stmt->fetchAll(PDO::FETCH_OBJ); //以对象方式解析出所有数据，返回给list
    foreach($list as $ob){
        echo $ob->id.":".$ob->name."<br/>";
    }
    */
    //第三种方式遍历
    while($ob = $stmt->fetch(PDO::FETCH_OBJ)){
        echo $ob->id.":".$ob->name.":".$ob->sex."<br/>";
    }

    echo "共计".$stmt->rowCount()."条数据";

    }catch(PDOException $pe){
        echo "执行失败！原因：".$pe->getMessage();

    }
```

八、预处理 ---- 将 SQL 语句和数据分离发送

1、SQL 语句中用占位符代替

2、每次都要校验 SQL 语句是否正确，但是预处理仅一次校验（固定格式保存在缓存中）

3、预处理时 SQL 语句和值是分开的，在值中存在 SQL 都当成字符串处理

4、SQL 语句的格式缓存在数据库中（先校验）最后再给值

PDO 中参数式的 SQL 语句有两种

```
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)
```

九、应用预处理绑定参数-----6 种方式

```
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)
在PDO中为参数式SQL语句赋值有三种:
   1.使用数组
     $stmt->execute(array("lamp1404","qq2"));
     $stmt->execute(array("id"=>"lamp1404","name"=>"qq2"));
   2.使用方法单个赋值
     $stmt->bindValue(1,"lamp1901");
     $stmt->bindValue(2,"qq2");
     $stmt->execute();

     $stmt->bindValue(":id","lamp1901",PDO::PARAM_STR);  //带指定类型
     $stmt->bindValue(":name","qq2",PDO::PARAM_STR);
     $stmt->execute();

   3. 使用方法绑定变量
     $stmt->bindParam(":id",$id);
```

第一种方式： ？ 绑定值 ---bindValue

```php
<?php
//1. 使用PDO的预处理实现了数据的添加
//示例方式：预处理sql语句方式是：？问号式sql语句，采用bindValue的方式绑定值

//1. 导入配置文件
require("dbconfig.php");

//2. 创建PDO对象，实现数据库的连接，并做属性设置
try{

    $pdo = new PDO(DSN,USER,PASS);//获取连接
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(异常模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    $pdo->query("set names utf8");//设置编码
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}
```

```php
try{
    //3. 定义并预处理sql语句，返回处理对象PDOStatment
    $sql = "insert into users values(null,?,?)";
    $stmt = $pdo->prepare($sql);

    //4. 对参数式sql语句中的参数？直接绑定值(前面1,2表示第几个?所对应的值)
    $stmt->bindValue(1,"tt001");//第一个?
    $stmt->bindValue(2,0,PDO::PARAM_INT);//第二个?，第三个可选参数表示类型

    //5. 执行添加
    $stmt->execute();

    //6. 输出结果
    echo "成功添加".$stmt->rowCount()."条数据<br/>";
    echo "添加数据的自增id值：".$pdo->lastInsertId()."<br/>";

}catch(PDOException $pe){
    echo "执行失败！原因：".$pe->getMessage();
}
```

第二种：？绑定值 ---bindParam

```php
<?php
//2、使用PDO的预处理实现了数据的添加
//示例方式：预处理sql语句方式是：？问号式sql语句，采用bindParam的方式绑定参数
//1. 导入配置文件
require("dbconfig.php");

//2. 创建PDO对象，实现数据库的连接，并做属性设置
try{

    $pdo = new PDO(DSN,USER,PASS);//获取连接
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(异常模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    $pdo->query("set names utf8");//设置编码
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}

try{
    //3. 定义并预处理sql(?问号方式的参数sql语句)语句，返回处理对象PDOStatment
    $sql = "insert into users values(null,?,?)";
    $stmt = $pdo->prepare($sql);

    //4. 对参数式sql语句中的参数？直接绑定参数
    $stmt->bindParam(1,$name);//前面的1表示第一个问号,第二个参数必须是变量
    $stmt->bindParam(2,$sex,PDO::PARAM_INT); //第三个可选参数表示类型

    //赋值(注意:赋值时可以写在绑定之前也可以写在绑定之后同时连续赋值但是必须在执行之前赋值)
    $name = "zhangsang";
    $sex = 1;
    $stmt->execute();
    $m='';
    $m+= $stmt -> rowCount();
    $name = "lisi";
    $sex = 3;
    $stmt->execute();
    $m+= $stmt -> rowCount();
    $name = "wangwuxoxo";
    $sex = 0;
```

```
    //5. 执行添加
    $stmt->execute();
    $m+= $stmt -> rowCount();

    //6. 输出结果
    //echo "成功添加".$stmt->rowCount()."条数据<br/>";
    echo "成功添加".$m."条数据<br/>";
    echo "添加数据的自增id值：".$pdo->lastInsertId()."<br/>";

}catch(PDOException $pe){
    echo "执行失败！原因：".$pe->getMessage();
}
```

第三种 ？ 问号式　参数绑定批量添加

```
<?php
header("Content-type:text/html;charset=utf-8");
//3. 使用PDO的预处理实现了数据的添加
//示例方式（推荐）：预处理sql语句方式是：？问号式sql语句，采用执行时execute的直接绑定。
//可以实现批量添加

//1. 导入配置文件
require("dbconfig.php");

//2. 创建PDO对象，实现数据库的连接，并做属性设置
try{

    $pdo = new PDO(DSN,USER,PASS);//获取连接
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(异常模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    $pdo->query("set names utf8");//设置编码
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}

try{
    //3. 定义并预处理sql语句，返回处理对象PDOStatment
    $sql = "insert into users values(null,?,?)";
    $stmt = $pdo->prepare($sql);

    //4. 绑定参数并执行(遍历数组)--批量操作
    $list = array(
        array("tt003",1),//索引数组
        array("tt004",0),
        array("tt005",1),
        array("tt006",2)

    );
    //遍历数组直接放值(数组)
    $m = '';
    foreach($list as $row){
        $stmt ->execute($row);
        $m+=$stmt->rowCount();//获取影响的行数
    }

    //$stmt->execute(array("tt003",0));//单条添加

    //6. 输出结果
    //echo "成功添加".$stmt->rowCount()."条数据<br/>";
    echo "成功添加".$m."条数据<br/>";
    echo "添加数据的自增id值：".$pdo->lastInsertId()."<br/>";

}catch(PDOException $pe){
    echo "执行失败！原因：".$pe->getMessage();
}
```

第四种、别名式 bindValue 直接绑定值

```php
<?php
header("Content-type:text/html;charset=utf-8");
//4. 使用PDO的预处理实现了数据的添加
//示例方式：预处理sql语句方式是：别名式的sql语句，采用bindValue的方式绑定值

//1. 导入配置文件
require("dbconfig.php");

//2. 创建PDO对象，实现数据库的连接，并做属性设置
try{

    $pdo = new PDO(DSN,USER,PASS);//获取连接
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(异常模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    $pdo->query("set names utf8");//设置编码
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}
//======================以上可以直接放在配置文件中=======================
try{
    //3. 定义并预处理sql(别名式的参数sql语句)语句，返回处理对象PDOStatment
    $sql = "insert into users values(null,:n,:s)";//别名自定义(参数过多使用别名方式)
    $stmt = $pdo->prepare($sql);

    //4. 对参数式sql语句中的参数直接绑定值
    $stmt->bindValue("n","tt004");//前面的冒号带不带都可以
    $stmt->bindValue("s",1,PDO::PARAM_INT);//第三个可选参数表示类型

    //5. 执行添加
    $stmt->execute();

    //6. 输出结果
    echo "成功添加".$stmt->rowCount()."条数据<br/>";
    echo "添加数据的自增id值：".$pdo->lastInsertId()."<br/>";

}catch(PDOException $pe){
    echo "执行失败！原因：".$pe->getMessage();
}
```

第五种、别名式 bindParam 绑定参数再指定值

```php
<?php
header("Content-type:text/html;charset=utf-8");
//5. 使用PDO的预处理实现了数据的添加
//示例方式：预处理sql语句方式是：别名式的sql语句，采用bindParam的方式绑定参数

//1. 导入配置文件
require("dbconfig.php");

//2. 创建PDO对象，实现数据库的连接，并做属性设置
try{

    $pdo = new PDO(DSN,USER,PASS);//获取连接
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(异常模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    $pdo->query("set names utf8");//设置编码
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}
```

```php
try{
    //3. 定义并预处理sql(别名式的参数sql语句)语句，返回处理对象PDOStatment
    $sql = "insert into users values(null,:n,:s)";//带上冒号表示别名
    $stmt = $pdo->prepare($sql);

    //4. 对参数式sql语句中的参数别名方式直接绑定参数
    $stmt->bindParam("n",$name,PDO::PARAM_STR); //第三个参数表示类型
    $stmt->bindParam(":s",$sex,PDO::PARAM_INT);

    //参数赋值(在执行有值就OK不分前后顺序)
    $name="tt005";
    $sex=1;

    //5. 执行添加
    $stmt->execute();

    //6. 输出结果
    echo "成功添加".$stmt->rowCount()."条数据<br/>";
    echo "添加数据的自增id值：".$pdo->lastInsertId()."<br/>";

}catch(PDOException $pe){
    echo "执行失败！原因：".$pe->getMessage();
}
```

第六种、执行时 execute 的直接绑定--批量添加

```php
<?php
header("Content-type:text/html;charset=utf-8");
//6. 使用PDO的预处理实现了数据的添加
//示例方式（推荐）：预处理sql语句方式是：别名式的sql语句，采用执行时execute的直接绑定。

//1. 导入配置文件
require("dbconfig.php");

//2. 创建PDO对象，实现数据库的连接，并做属性设置
try{

    $pdo = new PDO(DSN,USER,PASS);//获取连接
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(异常模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    $pdo->query("set names utf8");//设置编码
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}

try{
    //3. 定义并预处理sql(别名式的参数sql语句)语句，返回处理对象PDOStatment
    $sql = "insert into users values(null,:n,:s)";
    $stmt = $pdo->prepare($sql);

    $list=array(
        array("n"=>"t001","s"=>1),  //关联数组
        array(":s"=>0,"n"=>"t002"),//不分先后顺序
        array("n"=>"t003","s"=>0),
        array(":n"=>"t004","s"=>1),//可以带冒号

    );
```

```php
    $m='';
    foreach($list as $value){
        //4. 绑定参数并执行
        $stmt->execute($value);
        //$m+=$stmt->rowCount();
        $m++;
    }

    //6. 输出结果
    echo "成功添加".$m."条数据<br/>";
    echo "添加数据的自增id值：".$pdo->lastInsertId()."<br/>";
}catch(PDOException $pe){
    echo "执行失败！原因：".$pe->getMessage();
}
```

第十、PDO 连接属性设置实例

```php
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
//$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);
```

第十一、PDO 连接设置字符集

设置 php 连接 mysql 时的客户端字符串和连接字符串集为:

$pdo->exec(“set names utf8”);

或者:

$pdo->query(“set names utf8”);

第十二、PDO 对象中的成员方法

1. query($sql); 用于执行查询 SQL 语句。返回 PDOStatement 对象
2. exec($sql); 用于执行增、删、改操作，返回影响行数；
3. getAttribute(); 获取一个"数据库连接对象"属性。
4. setAttribute(); 设置一个"数据库连接对象"属性。
5. beginTransaction 开启一个事物（做一个回滚点）
6. commit 提交事务
7. rollBack 事务回滚操作。

8. errorCode　获取错误码
9. errorInfo　获取错误信息
10.lastInsertId　获取刚刚添加的主键值。
11.prepare　创建 SQL 的预处理，返回 PDOStatement 对象
12.quote 为 sql 字串添加单引号。

第十三、pdo::exec()方法 和 pdo::query()方法

pdo::exec()方法

当执行 insert,update,delete 没有结果集的查询时，使用 pdo 对象中的 exec()方法去执行，该方法成功执行时，将返回受影响的行数，注意，该方法不能用于 select 查询，

pdo::query()方法

当执行返回结果集的 select 查询时，或者所影响的行数无关紧要时，应当使用 pdo 对象中的 query()方法，如果该方法成功执行指定的查询，则返回一个 pdostatement 对象，如果使用了 query()方法，并想了解获取数据行总数，可以使用 pdostament 对象中的 rowCount()方法获取，

PDO 对预处理语句的支持

一、PDO 预处理

1 PDOStatement 对象的方法
2 准备语句
3 绑定参数
4 执行预处理方式
5 预处理查询
6 大数据对象的存取

二、PDOStatement 对象的方法

1、fetch() 返回结果集的下一行，结果指针下移，到头返回 false 。
参数：　PDO::FETCH_BOTH (default)、：索引加关联数组模式
PDO::FETCH_ASSOC、　：关联数组模式
PDO::FETCH_NUM、　：索引数组模式
PDO::FETCH_OBJ、　：对象模式
PDO::FETCH_LAZY　：所有模式（SQL 语句和对象）

2、fetchAll() 通过一次调用返回所有结果，结果是以数组形式保存
参数：　PDO::FETCH_BOTH (default)、
PDO::FETCH_ASSOC、
PDO::FETCH_NUM、
PDO::FETCH_OBJ、
PDO::FETCH_COLUMN 表示取指定某一列
如：$rslist = $stmt->fetchAll(PDO::FETCH_COLUMN,2);取第三列

3. execute() 负责执行一个准备好了的预处理语句
4. fetchColumn()返回结果集中下一行某个列的值

5. setFetchMode()设置需要结果集合的类型
6. rowCount()　返回使用增、删、改、查操作语句后受影响的行总数
7. bindParam() 将参数绑定到相应的查询占位符上

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

其中：

$parameter：占位符名或索引偏移量

&$variable:参数的值，需要按引用传递也就是必须放一个变量

$data_type:数据类型

PDO::PARAM_BOOL　　PDO::PARAM_NULL

PDO::PARAM_INT　　PDO::PARAM_STR

PDO::PARAM_LOB　　PDO::PARAM_STMT

PDO::PARAM_INPUT_OUTPUT

$length：指数据类型的长度

$driver_options：驱动选项。

8. setAttribute()为一个预处理语句设置属性
9. getAttribute()获取一个声明的属性
10. errorCode()　获取错误码
11. errorInfo() 获取错误信息
12. bindColumn() 用来匹配列名和一个指定的变量名，这样每次获取各行记录时，会自动将相应的值赋给变量。
13. bindValue() 将一值绑定到对应的一个参数中
14. nextRowset() 检查下一行集
15. columnCount() 在结果集中返回列的数目
16. getColumnMeta() 在结果集中返回某一列的属性信息
17. closeCursor() 关闭游标，使该声明再次执行

三、准备语句

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)

四、绑定参数　----前面已经介绍 6 种方式

$stmt->bindParam

$stmt->bindValue

五、执行预处理方式方法

$stmt->execute();

六、预处理查询

1、使用 PDO 的预处理实现了数据的查询

```php
<?php
//使用PDO的预处理实现了数据的查询

//1. 导入配置文件
require("dbconfig.php");

//2. 创建PDO对象，实现数据库的连接，并做属性设置
try{
    $pdo = new PDO(DSN,USER,PASS);//获取连接
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(异常模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    $pdo->query("set names utf8");//设置编码
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}

try{
    //3. 定义查询sql语句，并执行预处理
    $sql= "select * from users where id>? and sex=?";
    $stmt = $pdo->prepare($sql);
    |
    //4.绑定参数值并执行
    $stmt->execute(array(20,1));
    |
    //解析结果
    foreach($stmt as $row){
    |    echo $row['id'].":".$row['sex'].":".$row['name']."<br/>";
    }
    |
}catch(PDOException $pe){
    echo "执行失败！原因：".$pe->getMessage();
}
```

2、使用 PDO 的预处理实现了数据的查询+绑定输出结果

```php
<?php
//使用PDO的预处理实现了数据的查询+绑定输出结果
header("Content-type:text/html;charset=utf-8");
//1. 导入配置文件
require("dbconfig.php");

//2. 创建PDO对象，实现数据库的连接，并做属性设置
try{
    $pdo = new PDO(DSN,USER,PASS);//获取连接
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(异常模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    $pdo->query("set names utf8");//设置编码
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}
```

```php
try{
    //3. 定义查询sql语句，并执行预处理指明查询的列信息
    $sql= "select id,name,sex from users where id>? and sex=?";
    $stmt = $pdo->prepare($sql);

    //4.绑定参数值并执行返回PDOStatement对象的结果集
    $stmt->execute(array(18,1));

    //5.对输出的结果进行绑定
    $stmt->bindColumn(1,$id);/*  通过列号绑定  */
    $stmt->bindColumn(2,$name);
    $stmt->bindColumn("sex",$sex);/*  通过列名绑定  */
    //6.开始迭代输出
    while($stmt->fetch(PDO::FETCH_BOUND)){
        echo $id.":".$name.":".$sex."<br/>";
    }

}catch(PDOException $pe){
    echo "执行失败！原因：".$pe->getMessage();
}
```

PDO 的事务处理

一、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
```

二、构建事务处理的应用程序

事务处理是处理多条 SQL 语句来完成一件事的全部过程，自增 ID 不会回滚

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);

使用异常处理解决事务处理

```php
<?php
//使用PDO的事务支持实现多数据添加。（采用异常处理来实现的事务过程）
//被操作的表必须是innoDB类型的表（支持事务）
header('Content-type:text/html;charset=utf-8');
//1. 导入配置文件
require("dbconfig.php");

//2. 创建PDO对象，实现数据库的连接，并做属性设置
try{
    $pdo = new PDO(DSN,USER,PASS);//获取连接
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(异常模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
    $pdo->query("set names utf8");//设置编码
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}

try{
    $pdo->beginTransaction(); //开启事务
    //3. 定义查询sql语句，并执行预处理
    $sql= "insert into users values(null,:name,:sex)";
    $stmt = $pdo->prepare($sql);

    //批量添加数据
    $stulist = array(
        array("name"=>"tt013","sex"=>1),
        array("name"=>"tt014","sex"=>2),
        array("name"=>"tt015","sex"=>0),
        array("name"=>"tt016","sex"=>2),
    );

    $m=0;
    foreach($stulist as $stu){
        //绑定参数值并执行
        $stmt->execute($stu);
        $m+=$stmt->rowCount();
    }
    echo "成功添加{$m}条数据";
    $pdo->commit(); //执行提交
}catch(PDOException $pe){
    echo "执行失败！原因：".$pe->getMessage();
    $pdo->rollBack(); //执行事务回滚
}
```

不采用异常处理方式

```php
<?php
//使用PDO的事务支持实现多数据添加。（不采用异常处理来实现的事务过程）
//被操作的表必须是innoDB类型的表（支持事务）
$stulist = array(
        array("name"=>"tt024","sex"=>"w","age"=>22,"classid"=>"lamp71"),
        array("name"=>"tt025","sex"=>"m","age"=>23,"classid"=>"lamp71"),
        array("name"=>"tt027","sex"=>"m","age"=>22,"classid"=>"lamp71"),
        array("name"=>"tt026","sex"=>"w","age"=>25,"classid"=>"lamp71"),
    );
```

```php
//1. 导入配置文件
require("dbconfig.php");

//2. 创建PDO对象，实现数据库的连接，并做属性设置
try{
    $pdo = new PDO(DSN,USER,PASS);//获取连接
    //通过setAttribute方法来设置PDO中的属性信息（报错模式(忽略模式)）
    $pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_SILENT);
    $pdo->query("set names utf8");//设置编码
}catch(PDOException $pe){
    die('数据库连接失败！原因:'.$pe->getMessage());
}


    $pdo->beginTransaction(); //开启事务
    //3. 定义查询sql语句，并执行预处理
    $sql= "insert into stu values(null,:name,:sex,:age,:classid)";
    $stmt = $pdo->prepare($sql);

    //批量添加数据
    $m=0;
    foreach($stulist as $stu){
        //绑定参数值并执行
        $stmt->execute($stu);
        if($stmt->errorCode()>0){
            $m=0;
            break;
        }
        $m+=$stmt->rowCount();
    }

    //判断是否提交
    if($m>0){
        $pdo->commit(); //执行提交
    }else{
        $pdo->rollBack(); //执行事务回滚
    }

    echo "成功添加{$m}条数据";
    /*
    通过错误号累加来判断是否中间出了错误
    $code=0;
    foreach($list as $v){
        $stmt -> execute($v);
        $m+=$stmt -> rowCount();
        $code+=$stmt -> errorCode();
    }

    if($code == 0){
        $pdo -> commit():
    }else{
        $pdo -> rollback();
        $m=0;
    }
    */
```

```php
try{
    $dsn = "mysql:host=127.0.0.1;dbname=class94";
    $name = "root";
    $pwd = "";
    $pdo = new PDO($dsn,$name,$pwd);
    $pdo -> setAttribute(3, 2);
}catch(PDOException $e){
    echo $e -> getMessage();
}


try{
    /**
     * 确保数据库的引擎是支持事务的
     * 1、开启事务
     * 2、关闭自动提交
     * 3、执行流程：
     *      如果流程失败就是抛出一个异常,并执行事务回滚
     *      （回滚数据的区间。开启事务到异常）
     *      如果流程成功，就提交事务。
     * 4、不管失败或成功都要开启自动提交
     *
     */
    $pdo -> beginTransaction();
    $pdo -> setAttribute(PDO::ATTR_AUTOCOMMIT,0);
    $sql = "UPDATE cash SET money = money - 50 WHERE name = 'liming'";
    $affectedNum = $pdo -> exec($sql);
    if(!$affectedNum){
        throw new PDOException("转出失败！");
    }
    $sql ="UPDATE cash SET money = money + 50 WHERE name = 'lichao'";
    $affectedNum = $pdo -> exec($sql);
    if(!$affectedNum){
        throw new PDOException("转入失败");
    }

    $pdo -> commit();
    echo "交易成功！";
}catch(PDOException $e){
    $pdo -> rollback();
    echo $e -> getMessage();

}

$pdo -> setAttribute(PDO::ATTR_AUTOCOMMIT,1);
```