

后盾网 人人做后盾

www.houdunwang.com

PDO抽象层

后盾网 2011-2017 v5.0

- PDO扩展为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，无论使用什么数据库，都可以通过一致的函数执行查询和获取数据。PDO随PHP5.1发行
- PHP通过PDO扩展动态加载相应的数据库驱动程序来完成直接操作各种数据库，所以在使用PDO时要明确告之所要使用的数据库驱动

PDO

LINUX开启PDO

- 如果你正在使用PHP5.1版本，PDO和PDO SQLITE已经包含在了此发行版中
- `yum -y install php-pdo`

Window开启PDO

- Window开启PDO需要修改php.ini文件，先将`extension=php_pdo.dll`前面的分号去掉
- 将需要的数据库驱动开启，如将`;extension=php_pdo_mysql.dll`前的分号去掉，就开启了MYSQL数据库驱动，可以使用PDO抽象层操作MYSQL了，其他数据库操作也是按这种方式操作

开启PDO库

PDO::__construct() (\$dsn , \$username , \$password)

- Dsn代表数据源，包括数据库类型，主机地址，数据库名
- username代码连接数据库的用户名
- password代码连接数据库的密码

PHP使用PDO

1. `$dsn = "mysql:host=localhost;dbname=demo";`
2. `$username="root";`
3. `$password = "admin888";`
4. `try{`
5. `$pdo = new PDO($dsn,$username,$password);`
6. `}catch(PDOException $e) {`
7. `die($e->getMessage());`
8. `}`

如果参数正确，通过以上代码已经成功连接mysql

通过PDO连接MYSQL

1. `$result = $pdo->query("select * from stu");`

2. `$row = $result->fetchAll();`

3. `print_r($row);`

通过`fetch()`函数一次得到一条记录

```
$result = $pdo->query("select * from hd_user limit 2");
```

```
print_r($result->fetchAll());
```

通过`fetchAll()`函数一次得到所有记录

发送查询

以上操作方式得到的数据包含关联和索引方式，这样数组量会很大

```
$result->fetchAll(PDO::FETCH_ASSOC)
```

- 只得到关联方式的数据

```
$result->fetchAll(PDO::FETCH_NUM)
```

- 得到索引表示记录

```
$result->fetchAll(PDO::FETCH_BOTH)
```

- 得到关联与索引表示的数据

```
$result->fetchAll(PDO::FETCH_OBJ)
```

- 得到以对象形式表示的数据

返回数据

PDO::ATTR_ERRMODE 错误提示.

- PDO::ERRMODE_SILENT: 不显示错误信息
- PDO::ERRMODE_WARNING: 显示警告错误
- PDO::ERRMODE_EXCEPTION: 抛出异常

示例

1. `$dsn = "mysql:host=127.0.0.1;dbname=demo;charset=utf8";`
 2. `$pdo = new PDO($dsn,"root","");`
 3. `$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING);`
 4. `$pdo->query("set names utf8");`
 5. `$result = $db->query("select * from hd_user");`
- # 由于设置了警告提示, 当发生错误时会自动弹出警告

PDO错误处理

`mixed PDOStatement::fetch (int $fetch_style)`

- 每次从结果集中取出一条记录

`array PDOStatement::fetchAll ([int $fetch_style])`

- 一次将结果集中的所有记录全部取出

得到数据结果

int PDO::exec (string \$statement)

exec方法用于执行没有结果集的数据库操作

返回值为受影响的条数

返回值可能为false，比如更新时字段不存在

1. \$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
2. try {
3. \$sql = "delete from stu where sid>2";
4. \$affected_rows = \$pdo->exec(\$sql);
5. echo "共删除成功{\$affected_rows}条记录";
6. } catch (PDOException \$e) {
7. echo \$e->getMessage();
8. }

发送SQL

PDOStatement PDO::query (string \$statement)

query方法用于执行有结果集的select查询使用

1. \$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
2. try {
3. \$sql = "select * from hd_user LIMIT 1";
4. \$result = \$pdo->query(\$sql);
5. \$rows = \$result->fetchAll(PDO::FETCH_ASSOC);
6. } catch (PDOException \$e) {
7. die(\$pdo->errorCode());
8. }

注：可以将结果集直接用foreach遍历

发送SQL

- 预准备语句是SQL语句编译后形成的模板，也就是说语句只解析一次，以后只要传替不同的参数就可以了
- 每一条SQL的执行过程包括分析、编译、优化查询等操作，如果只是SQL的参数不同而其他项一样，那么可以想象，数据库做了很多次重复的分析、编译、优化操作，显然这种执行方式会降低效率。如果使用预准备方式操作，这种分析、编译、优化的操作只需要执行一次，以后只是参数的不同，这样效率会更快
- 由于SQL是先进行解析处理后传替参数，所以可以想相对的减少SQL注入的风险

PDO实现预准备

我们已经知道预准备的思想，首先要预先编译一下sql，同时要会后期传递变量预留位置，PDO可以方便的这样的操作

1.\$sth = \$pdo->prepare('SELECT sname FROM stu WHERE sname = ? AND age >=?');

2.\$sname='houdunwang';

3.\$age=22;

4.\$sth->bindParam(1, \$sname, PDO::PARAM_STR);

5.\$sth->bindParam(2, \$age, PDO::PARAM_INT);

6.\$sth->execute();

7.\$row = \$sth->fetchAll();

预准备查询

```
1. try {  
2.     $sql = "insert into stu(sname) values(?)";  
3.     $state = $pdo->prepare($sql);  
4.     $sname = "baidu";  
5.     $state->bindParam(1,$sname,PDO::PARAM_STR);  
6.     $row = $state->execute();  
7.     print_r($row);  
8. } catch (PDOException $e) {  
9.     $e->getMessage();  
10.}
```

预准备语句
