Overview of PHP MySQL connectors

Using the three mysqlnd driver interfaces

Justin Swanhart

mysqlnd vs libmysqlclient

- Compile time decision
  - mysqlnd is the **M**ySQL **N**ative **D**river for PHP
    - Introduced in PHP 5.3
    - Default in PHP 5.4
  - libmysqlclient
    - requires C client installation and headers

Libmysqlclient warning messages

- If you choose libmysqlclient the headers should match the compiled version:
  - PHP Warning: mysql_connect(): Headers and client library minor version mismatch. Headers:50537 Library:50622 in /home/justin/t.php on line 15

PHP API choices

- There are three MySQL API choices
  - All three work with mysqlnd and libmysqlclient
  - Choices are
    - Legacy mysql functions
    - mysqli extension
    - PDO_MySQL

Available API choices

- Legacy mysql API
  - This is the original MySQL API for PHP
  - Many older applications written in this API
  - Not recommended few new development (this API is officially deprecated)

Why not use a deprecated interface?

- Will be removed in the future
- Does not support all features of mysqlnd
- No object-oriented interface

Example mysql API program


```
<?php

$sql = <<<EOF
select
FlightDate, UniqueCarrier, TailNum, FlightNum, OriginAirportID, DestAirportID
from demo.ontime
limit 2;
EOF;

$conn =mysql_connect('localhost','root','password');
if(!$conn) {
  echo mysql_error() . "\n";
  exit(-1);
}

$stmt = mysql_query($sql, $conn);
if(!$stmt) {
  echo mysql_error($conn) . "\n";
}

while($row = mysql_fetch_assoc($stmt)) {
  print_r($row);
}
```

Output

```
Array
(
    [FlightDate] => 2010-10-14
    [UniqueCarrier] => DL
    [TailNum] => N921DL
    [FlightNum] => 1
    [OriginAirportID] => 11193
    [DestAirportID] => 12478
)
Array
(
    [FlightDate] => 2010-10-14
    [UniqueCarrier] => DL
    [TailNum] => N698DL
    [FlightNum] => 3
    [OriginAirportID] => 14869
    [DestAirportID] => 12478
)
```

Useful functions

- mysql_close
- mysql_insert_id
- mysql_fetch_array
- mysql_real_escape_string
- mysql_select_db
- mysql_unbuffered_query

mysqli extension

- Dual interface
  - Procedural (function) based interface
  - Object-oriented interface

Procedural interface


```
$conn = mysqli_connect('localhost','root','password','demo');
if (mysqli_connect_errno($conn)) {
    echo "Failed to connect to MySQL: " . mysqli_connect_error();
    exit(-1);
}
$stmt = mysqli_query($conn, $sql);
if(!$stmt) {
  echo mysqli_error($conn) . "\n";
  exit(-1);
}
while($row = mysqli_fetch_assoc($stmt)) {
  print_r($row);
}
```

mysqli object-oriented interface


```
$conn = new mysqli('localhost','root','password','demo');
if($conn->connect_errno) {
  echo $mysqli->connect_error . "\n";
  exit(-1);
}

$stmt = $conn->query($sql);
if(!$stmt) {
  echo $conn->error . "\n";
  exit(-1);
}

while($row = $stmt->fetch_assoc()) {
  print_r($row);
}
```

Prepared statements


```
if (!($stmt = $mysqli->prepare("INSERT INTO tbl(id) VALUES (?)"))) {
    echo $mysqli->error . "\n";
    exit(-1);
}
$id = 9;
if (!$stmt->bind_param("i", $id)) {
    echo $stmt->error . "\n";
    exit(-1);
}
if (!$stmt->execute()) {
    echo "Execute failed: (" . $stmt->errno . ") " . $stmt->error;
}
```

Useful functions/members

- mysqli::set_charset
- mysqli::real_escape_string
- mysqli::$insert_id
- mysqli::begin_transaction
- mysql::commit
- mysql::select_db

PDO / PDO_MySQL

- PDO (**P**HP **D**ata **O**bjects) is a consistent object-oriented database abstraction layer to multiple database drivers
- Does not provide missing SQL features, you must use cross-platform SQL
- PDO_MySQL is the MySQL interface for PDO

Connecting with PDO

Uses a DSN:

```
$dsn = 'mysql:dbname=testdb;host=127.0.0.1';
try {
    $conn = new PDO($dsn, 'root'], 'password',
array(PDO::MYSQL_ATTR_LOCAL_INFILE => 1));
}
catch (PDOException $e) {
  echo 'Connection failed: ' . $e->getMessage();
  return false;
}
```

Using buffered queries


```
$conn-
>setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY,
true);
try{
$stmt = $conn->query($sql);
}
catch {
…
}
```

Using unbuffered queries


```
$conn->setAttribute(PDO::MYSQL_ATTR_USE_BUFFERED_QUERY,
false);
try{
   $stmt = $conn->query($sql);
}
catch {
   …
}
```

Fetching results

- As array with column names
  - `$r = $stmt->fetch(PDO::FETCH_ASSOC);`
- Numerically indexed
  - `$r = $stmt->fetch(PDO::FETCH_NUM);`

Example wrapper for PDO

- The Shard-Query DAL (Simple-DAL) wraps around PDO, using legacy mysql like functions but also providing OO interface support
- One single object for all actions, unlike PDO or mysqli

Simple-DAL


```
$server=array('host'=>localhost,'user'=>'root','password'=>'pass','db'=>'demo');
$conn = SimpleDAL::factory($server);
if($conn->my_error()){
  die($conn->my_error() . "\n");
}

$stmt = $conn->my_query($sql);
while($row = $conn->my_fetch_assoc($stmt)) {
 …
}
or

$conn->my_query($sql);

while($row = $conn->my_fetch_assoc()) {
 …
}
```

https://github.com/greenlion/swanhart-tools/tree/master/shard-query/include/DAL