MySQL

Using Node and PDO to connect and use the MySQL database system

MySQL Workbench

MySQL Workbench does everything you need, and it is free.

Text Editors – IDE's

VI, emacs, TextMate, …

phpMyAdmin

phpMyAdmin is widely used, web-based application.

You can use phpMyAdmin to administer, develop and manage your MySQL instances.

http://www.phpmyadmin.net/home_page/index.php

PDO: PHP Data Objects

An abstraction over various database systems (Oracle, DB2, MySQL, …)

Why use PDO

- **Same interface** for use with different database systems.

- **Flexibility**: You simply switch the driver name when you move to another database system (MySQL->Oracle)

- **Object-oriented**: It is more efficient to code with Objects than straight procedural coding

Connection Example

```
<?php

$host = "127.0.0.1";
$dbname = "metcs";
$user = ”some_user";
$pass = ”some_pswd";

try {
  $db = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
  $affectedRowCount =
        $db->exec("insert into users(email, password)
                        values (aes_encrypt(’asheehan@bu.edu', 'key'),
                                aes_encrypt('test', 'key'));");

        echo "Affected Row Count: " . $affectedRowCount;
}
catch(PDOException $e) {
   echo $e->getMessage();
}
?>
```

Exception Handling

```
<?php
    try {
          // do something that may raise an exception
   } catch (Exception $exception) {
         // handle the exception...
         include 'general_sys_error.php';
         exit();
  }
?>
```

Transactions

```
 <?php
  $host = "127.0.0.1";
  $dbname = "metcs";
  $user = ”some_user";
  $pass = ”some_pswd";

  $db = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
?>
```

```
<?php
    // autocommit is OFF
    $db -> beginTransaction();
    $pdoStmt = $db->exec("delete from users where
            aes_decrypt(email, 'key') = 'andrew.asheehan@gmail.com'");
    $db -> rollBack();

    // Now, autocommit is back on (default)
?>
```

Prepared Statements (named placeholders)

Write a SQL statement once, then use it N times.

```
$db = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
 $statement = $db->prepare("insert into……”);
 $email = "test@test.edu";
 $pswd = "test";

 $statement->bindParam(":email", $email, PDO::PARAM_STR);
 $statement->bindParam(":pswd", $pswd, PDO::PARAM_STR);
 $affectedCount = $statement->execute();

 $email = ”foo@beer.com”;
 $pswd = ”beer";

 $statement->bindParam(":email", $email, PDO::PARAM_STR);
 $statement->bindParam(":pswd", $pswd, PDO::PARAM_STR);

 $affected = $statement->execute();
```

Prepared Statements (positional placeholders)

```
<?php
$stmt = $dbh->prepare("insert into users (name, email)
      values (aes_encrypt(?, 'key’), aes_encrypt(?, 'key'))");

$stmt->bindParam(1, $name);
$stmt->bindParam(2, $value);

// insert one row
$name = 'one';
$value = 'a@b.com';
$stmt->execute();

$name = 'two';
$value = 'c@d.com’;
$stmt->execute();
?>
```

bindValue() vs. bindParam()

bindParam() is by reference. bindValue() is not.

Example:

bindValue()

```
    $stmt->bindValue(”:name", ”Andrew");
```

bindParam()

```
    $firstName = “Andrew”;
    $stmt->bindParam(“:name”, $firstName);
```

lastInsertId()

Returns the ID value of the last inserted row.

If you use a transaction, you should use lastInsertId() BEFORE you commit otherwise it will return 0

Example

```
<?php
$host = "127.0.0.1";
$dbname = "metcs";
$user = "user";
$pass = "pswd";

try {
  $db = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
  $stmt = $db->prepare("insert into class(name, professor) values (:name, :prof)");

  $stmt->bindValue(':name', 'MET CS601', PDO::PARAM_STR);
  $stmt->bindValue(':prof', 'SHEEHAN', PDO::PARAM_STR);

  // Notice: its not called from $stmt, but from $db
  echo "Last Inserted Value (primary key):   " . $db->lastInsertId();
}
catch(PDOException $pdoError) {
   echo "error: " + $pdoError->getMessage();
}
?>
```

query()

```
<?php
$host = "127.0.0.1";
$dbname = "metcs";
$user = "root";
$pass = "joe81";

try {
  $db = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
  $records = $db->query("select * from class");

  echo '<table>';
  echo '<tr><th>Class ID</th><th>Name</th><th>Professor</th></tr>';
  foreach ($records as $rec) {
    echo '<tr><td>' . $rec['class_id'] . '</td>';
    echo '<td>' . $rec['name'] . '</td><td>' . $rec['professor'] . '</td></tr>';
  }
  echo '</table>';
} catch(PDOException $e) {
    echo "error: " + $e->getMessage();
}
?>
```

fetch() vs fetchAll

The difference between:

fetchAll() performance could suffer when the number of results in your set is large

(*more memory being used*.. per client…)

fetchAll()

```
$stmt = $db->prepare("select class_id,name,professor from class");
$stmt->execute();

$records = $stmt->fetchAll();
$stmt->closeCursor();

echo '<table>’;
echo '<tr><th>Class ID</th><th>Name</th><th>Professor</th></tr>';

foreach ($records as $rec) {
   echo '<tr><td>’   .  $rec['class_id']      .  '</td>';
   echo '<td>'        .  $rec['name']          .  '</td>';
   echo '<td>'        .  $rec['professor']     .  '</td></tr>';
 }
 echo '</table>’;
```

fetch()

```
$stmt = $db->prepare("select class_id,name,professor from class");
$stmt->execute();

$record = $stmt->fetch();  // get the first row

echo '<table>’;
echo '<tr><th>Class ID</th><th>Name</th><th>Professor</th></tr>';

while( $record != null ) {
   echo '<tr><td>’   .   $record['class_id']      .   '</td>';
   echo '<td>'        .   $record['name']          .   '</td>';
   echo '<td>'        .   $record['professor']     .   '</td></tr>';

   $record = $stmt->fetch();  // get the next rows…
}

echo '</table>’;

$stmt->closeCursor();
```

MySql with Node

```
var mysql = require('mysql');
var connection = mysql.createConnection({
 host : 'mydbserver',
 user : 'user',
 password : 'password',
 database : 'schemaname'
});

connection.connect();

connection.query('SELECT MAX(order_num) AS result, function (error, results, fields) {
 if (error) throw error;
 console.log('The result is: ', results[0].result);
});

connection.end();
```

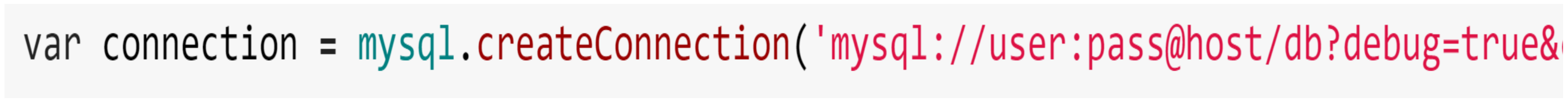
Using the callback

```
var mysql      = require('mysql');
var connection = mysql.createConnection({
  host     : 'example.org',
  user     : 'bob',
  password : 'secret'
});

connection.connect(function(err) {
  if (err) {
    console.error('error connecting: ' + err.stack);
    return;
  }

  console.log('connected as id ' + connection.threadId);
});
```

Can still use connections strings in the URL



```
var connection = mysql.createConnection('mysql://user:pass@host/db?debug=true&
```

Ending your connections: using the callback

```
connection.end( err => {
  // do something (logging…) when
  // the connection is terminated.
});
```