Node.js + MySQL

COEN 161

Getting Started

- To access a MySQL database with Node.js, you need a MySQL driver
- We can use an NPM library as a driver, the `mysql` package!

```
$   npm install mysql
```

- Now you can import the `mysql` module into Node.js

```
const mysql = require('mysql');
```

Query a Database

- The first step in working with a database, is connecting to it

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
});
```

Query a Database

- Once connected, we can begin to operate on the database
- This is also called *querying* the database

```
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Result: " + result);
  });
});
```

- The query method takes an sql statement as a parameter and passes the result to a callback function

Create a Database

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  con.query("CREATE DATABASE mydb", function (err, result) {
    if (err) throw err;
    console.log("Database created");
  });
});
```

Create a Table

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "CREATE TABLE customers (name VARCHAR(255), address VARCHAR(255))";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table created");
  });
});
```

Insert Into Table

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "INSERT INTO customers (name, address) VALUES ('Company Inc', 'Highway 37')";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted");
  });
});
```

Insert Multiple Records

```
con.connect(function(err) {
  if (err) throw err;
  console.log("Connected!");
  var sql = "INSERT INTO customers (name, address) VALUES ?";
  var values = [
    ['John', 'Highway 71'],
    ['Peter', 'Lowstreet 4'],
    ['Amy', 'Apple st 652'],
    ['Hannah', 'Mountain 21'],
    ['Michael', 'Valley 345'],
    ['Sandy', 'Ocean blvd 2'],
    ['Viola', 'Sideway 1633']
  ];
  con.query(sql, [values], function (err, result) {
    if (err) throw err;
    console.log("Number of records inserted: " + result.affectedRows);
  });
});
```

The Result Object

- When executing a query, a result object is returned containing information about how the query affected the table

```
/*
{
  fieldCount: 0,
  affectedRows: 14,
  insertId: 0,
  serverStatus: 2,
  warningCount: 0,
  message: '\'Records:14  Duplicated: 0  Warnings: 0',
  protocol41: true,
  changedRows: 0
}
*/
console.log(result.affectedRows) // 14
```

Get Inserted ID

- For tables with an auto increment id field, you can get the id of the row you just inserted through the result object
- This only works when inserting one record

```
con.connect(function(err) {
  if (err) throw err;
  var sql = "INSERT INTO customers (name, address) VALUES ('Michelle', 'Blue Village 1')";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("1 record inserted, ID: " + result.insertId);
  });
});
```

Selecting From a Table

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM customers", function (err, result, fields) {
    if (err) throw err;
    console.log(result);
  });
});
```

Selecting Columns

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT name, address FROM customers", function (err, result, fields) {
    if (err) throw err;
    console.log(result);
  });
});
```

The Fields Object

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT name, address FROM customers", function (err, result, fields) {
    if (err) throw err;
    console.log(fields);
  });
});
```

Select With a Filter

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM customers WHERE address = 'Park Lane 38'", function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Wildcard Characters

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM customers WHERE address LIKE 'S%'", function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Escaping Query Values

- When query values are variables provided by the user, you should escape the values
- This is to prevent SQL injections, which is a common web hacking technique to destroy or misuse your database
- The MySQL module has methods to escape query values

```
var adr = 'Mountain 21';
var sql = 'SELECT * FROM customers WHERE address = ' + mysql.escape(adr);
con.query(sql, function (err, result) {
  if (err) throw err;
  console.log(result);
});
```

Escaping Query Values

- You can also use a ? as a placeholder for the values you want to escape
- The values are automatically escaped

```
var name = 'Amy';
var adr = 'Mountain 21';
var sql = 'SELECT * FROM customers WHERE name = ? OR address = ?';
con.query(sql, [name, adr], function (err, result) {
  if (err) throw err;
  console.log(result);
});
```

Sort the Result

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM customers ORDER BY name", function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

ORDER BY DESC

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  con.query("SELECT * FROM customers ORDER BY name DESC", function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Update Table

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  var sql = "UPDATE customers SET address = 'Canyon 123' WHERE address = 'Valley 345'";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result.affectedRows + " record(s) updated");
  });
});
```

Delete Record

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  var sql = "DELETE FROM customers WHERE address = 'Mountain 21'";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Number of records deleted: " + result.affectedRows);
  });
});
```

The Result Object

- The result object contains information about how the query affected the table

```
{
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 34,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0
}
```

Delete a Table

```
var mysql = require('mysql');

var con = mysql.createConnection({
  host: "localhost",
  user: "yourusername",
  password: "yourpassword",
  database: "mydb"
});

con.connect(function(err) {
  if (err) throw err;
  var sql = "DROP TABLE customers";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log("Table deleted");
  });
});
```

Drop Only if Exist

- If the the table you want to delete is already deleted, or for any other reason does not exist, you can use the IF EXISTS keyword to avoid getting an error

```
con.connect(function(err) {
  if (err) throw err;
  var sql = "DROP TABLE IF EXISTS customers";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Drop Only if Exist - The Result Object

- If the table exists

```
{
  fieldCount: 0,
  affectedRows: 0,
  insertId: 0,
  serverstatus: 2,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0
}
```

- If the table does **not** exist

```
{
  fieldCount: 0,
  affectedRows: 0,
  insertId: 0,
  serverstatus: 2,
  warningCount: 1,
  message: '',
  protocol41: true,
  changedRows: 0
}
```

The Result Object

- The result object contains information about how the query affected the table

```
{
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 34,
  warningCount: 0,
  message: '(Rows matched: 1 Changed: 1 Warnings: 0',
  protocol41: true,
  changedRows: 1
}
```

Limit the Result

- You can limit the number of records returned from the query, by using the "LIMIT" statement

```
con.connect(function(err) {
  if (err) throw err;
  var sql = "SELECT * FROM customers LIMIT 5";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Start From Another Position

- If you want to return five records, starting from the third record, you can use the "OFFSET" keyword

```
con.connect(function(err) {
  if (err) throw err;
  var sql = "SELECT * FROM customers LIMIT 5 OFFSET 2";
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Shorter Syntax

- You can also use write your SQL statement like this "LIMIT 2, 5" which returns the same as the offset example

```
con.connect(function(err) {
  if (err) throw err;
  var sql = "SELECT * FROM customers LIMIT 2, 5"; // numbers are reversed
  con.query(sql, function (err, result) {
    if (err) throw err;
    console.log(result);
  });
});
```

Resources

https://www.w3schools.com/nodejs/nodejs_mysql.asp

https://www.npmjs.com/package/mysql