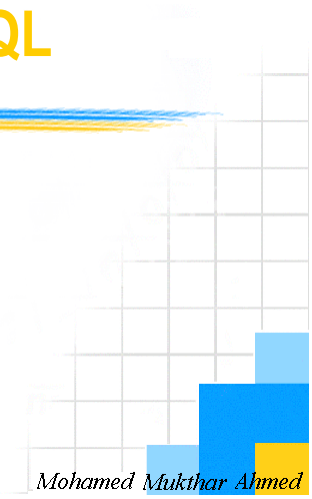
Node.js MySQL

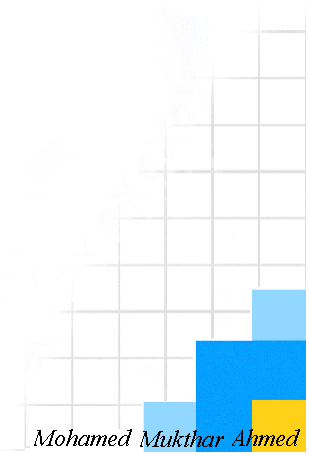


38

Node.js - MySQL

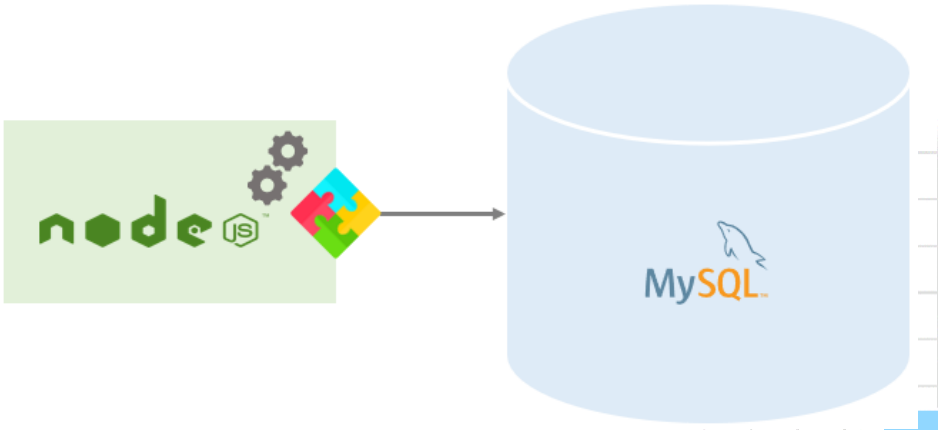
Outline

- Install mysql Module
- Connecting to MySQL
- DB Connection Pool
- Creating a Table
- Performing CRUD Operations
  - Insert a Row
  - Querying Data
    - Passing Data to Query
  - Update Data
  - Delete Row(s)
- Calling Stored Procedures



39

Node.js - MySQL



40

Install mysql Module

- To Interact with **MySQL** from Node.js applications using the **mysql** module.

```
npm install mysql
```

- Connecting to **MySQL** with the following steps:

1. Import the **mysql** module
2. Create a connection object to the **MySQL** database by using **createConnection()**
3. Using the connection object, invoke the **'connect'** method
4. To close a DB connection gracefully, invoke the **end()** method on the connection object.

41

Database Connection Pool

- The **MySQL** driver for node.js module provides you with a built-in connection pooling feature.

```
var pool = mysql.createPool({
    connectionLimit: 5,
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'todoapp'
});
```

- To get a connection from the pool, you use the **getConnection()** method:

```
pool.getConnection(function(err, connection) {
  // execute query
  // ...
  connnection.release();
});
```

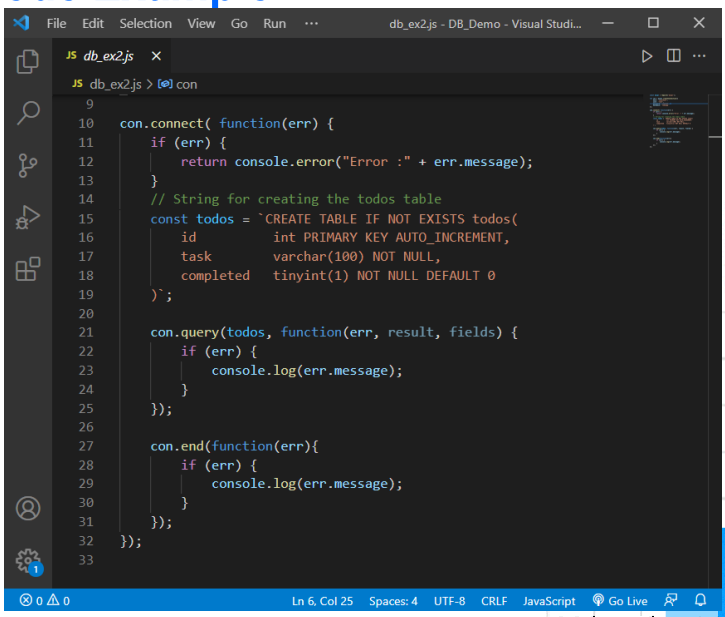

42

Creating a Table

- To create a table from node.js, you use these steps:
- Connect to the **MySQL** database server.
- Call the **query()** method on the connection object to execute a **CREATE TABLE** statement.
- Close the database connection.
- The **query()** method accepts an SQL statement and a callback.
- The callback function takes three arguments:
  - error: stores the detailed error if an error occurred during the execution of the statement
  - results: contains the results of the query
  - fields: contains results fields information if any

43

Code Example



Insert a Row

- To insert a new row into a table, you follow these steps:
  - Connect to the **MySQL** database..
  - Execute an **INSERT** statement by calling the **query()** method on a connection object.
  - Close the database connection.
- Look into the code below:

```
let sql = `INSERT INTO todos(task, completed)
          VALUES('Attend Node.js Training', true)`;

con.query( sql );
con.end();
```

Insert a Row

- To pass data to an **SQL** statement, you use the **question marks** (?) as the **placeholders**.
- Look into the code below:

```
let sql = `INSERT INTO todos(task, completed)
          VALUES(?,?)`;

theTask = ['Complete Node.js Assignment', false];

con.query( sql, theTask, (err, results, fields) => {
    if (err) {
        console.error(err);
    }
    // get last inserted ID
    console.log("Todo ID:" + results.insertId);
});
```


46

Querying Data

- To query data in the **MySQL** database, you follow these steps:
  - Connect to the **MySQL** database..
  - Execute a **SELECT** statement and process the **result set**.
  - Close the database connection.
- Look into the code below:

```
let sql = `SELECT * FROM todos`;

con.query( sql, (err, results, fields) => {
    if (err) {
        console.error(err);
    }
    console.log(results);
} );
```


47

Passing Data to Query

- To pass data to the query, use **placeholders**:
- Look into the code below:

```
let sql = `SELECT * FROM todos WHERE completed = ?`;

con.query( sql, [false], (err, results, fields) => {
    if (err) {
        console.error(err);
    }
    console.log(results);
} );
```


48

Updating Data

- To update data from Node.js application into the **MySQL** database, you follow these steps:
  - Connect to the **MySQL** database..
  - Execute an **UPDATE** statement by calling the **query()** method on a Connection object.
  - Close the database connection.
- Look into the code below:

```
let sql = `UPDATE todos
SET completed = ?
WHERE id = ?`;

con.query( sql, [false, 1], (err, results, fields) => {
    if (err) {
        console.error(err);
    }
    console.log("Rows effected :" + results.affectedRows);
} );
```


49

Delete Row(s)

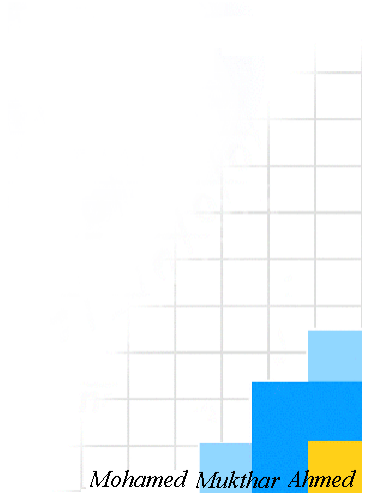
- To delete row(s) from Node.js application into the **MySQL** database table, you follow these steps:
  - Connect to the **MySQL** database..
  - Execute a **DELETE** statement by calling the **query()** method on a Connection object.
  - Close the database connection.
- Look into the code below:

```
let sql = `DELETE FROM todos WHERE id = ?`;

con.query( sql, 1, (err, results, fields) => {
    if (err) {
        console.error(err);
    }
    console.log("Delete Row(s) :" + results.affectedRows);
} );
```


Stored Procedure

- The steps for calling a stored procedure are similar to the steps for executing a query:
  - Connect to the **MySQL** database..
  - Call the stored procedure by executing the **CALL** <**spName**> statement..
  - Close the database connection.
- Look into the code below:


Mohamed Mukthar Ahmed