


## LAB4

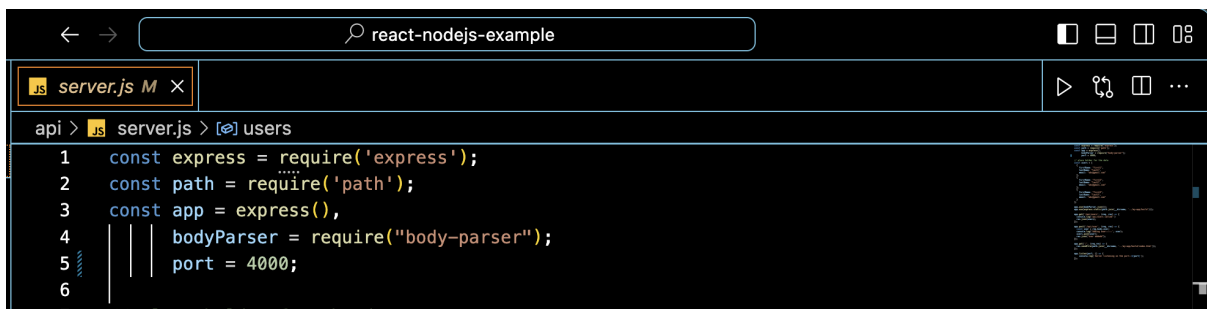
เปลี่ยน Port ของ backend เป็น 4000 หน้า package.json



The screenshot shows the VS Code editor with the file `package.json` open. The content of the file is as follows:

```
1 {
2   "name": "react-nodejs-example",
3   "version": "1.0.0",
4   "description": "example project react with nodejs",
5   "main": "server.js",
6   "scripts": {
7     "start": "node server.bundle.js",
8     "build": "webpack",
9     "dev": "nodemon ./server.js localhost 4000"
10  },
11 }
```

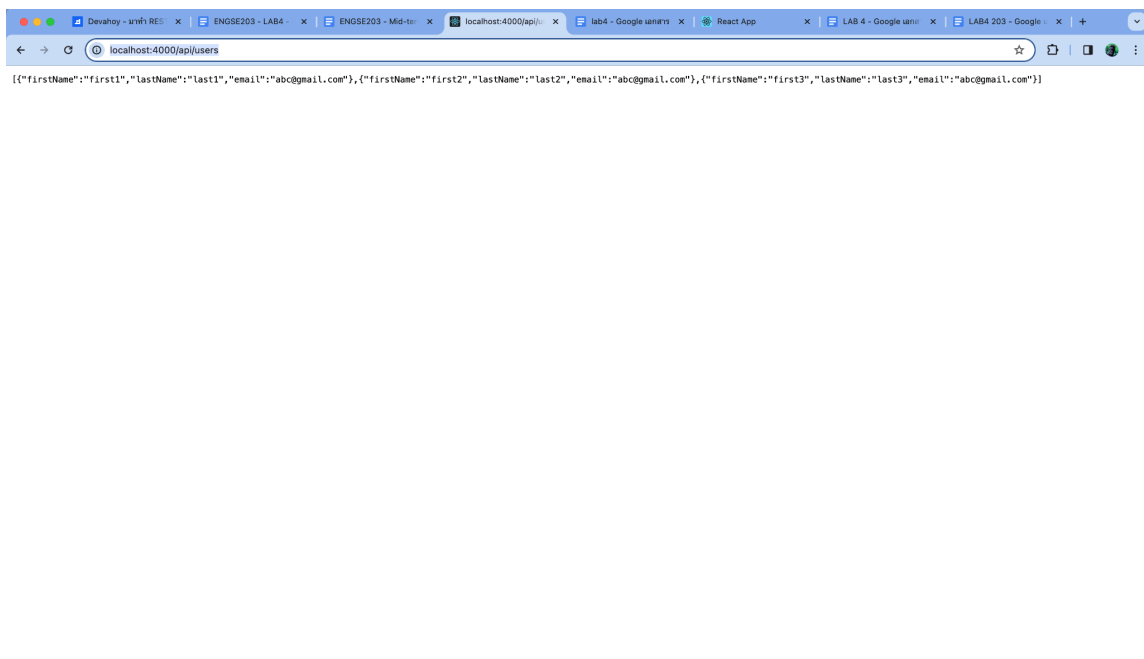
เปลี่ยน Port ของ backend เป็น 4000 หน้า server.js



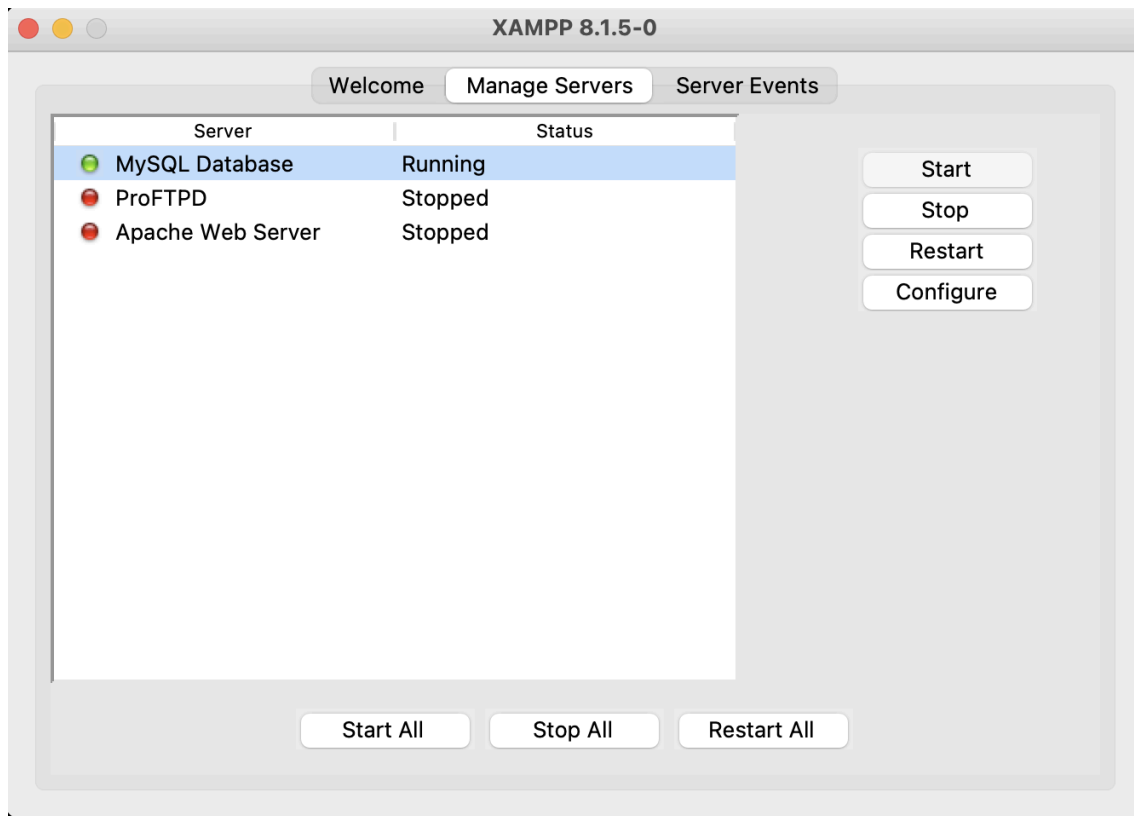
The screenshot shows the VS Code editor with the file `server.js` open. The content of the file is as follows:

```
1 const express = require('express');
2 const path = require('path');
3 const app = express(),
4       bodyParser = require("body-parser");
5 app.use(bodyParser.json());
6 app.use(bodyParser.urlencoded({ extended: false }));
7 app.use(express.static(path.resolve(__dirname, 'public')));
8 app.get('/', (req, res) => res.send('Hello World!'));
9 app.post('/users', (req, res) => {
10   const { firstName, lastName, email } = req.body;
11   if (!firstName || !lastName || !email) {
12     return res.status(400).send('Invalid input');
13   }
14   const user = { firstName, lastName, email };
15   users.push(user);
16   res.status(201).send(user);
17 });
18 app.listen(4000, () => console.log('Server is running on port 4000'));
```

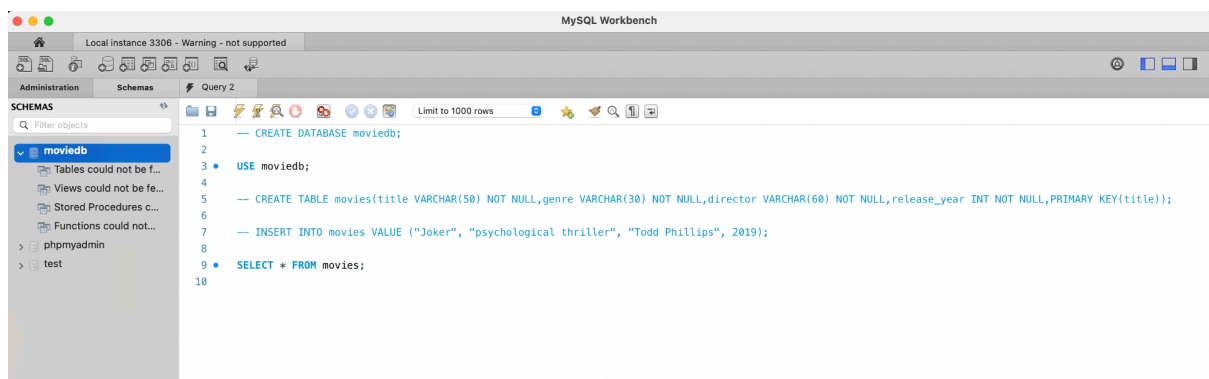
ทดสอบ run (<http://localhost:4000/api/users>)



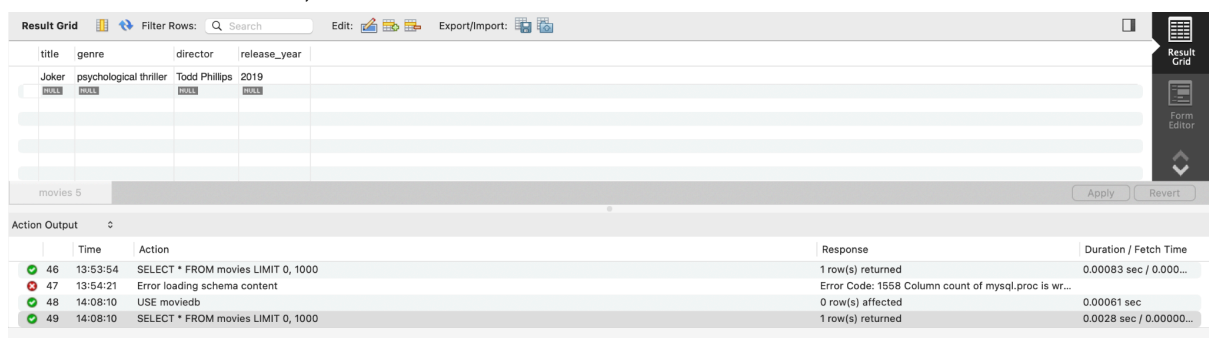
## เปิด MySQL ใน Xampp



## สร้าง database , เพิ่มตารางและเพิ่มข้อมูลลงในตาราง



## SELECT \* FROM movies;



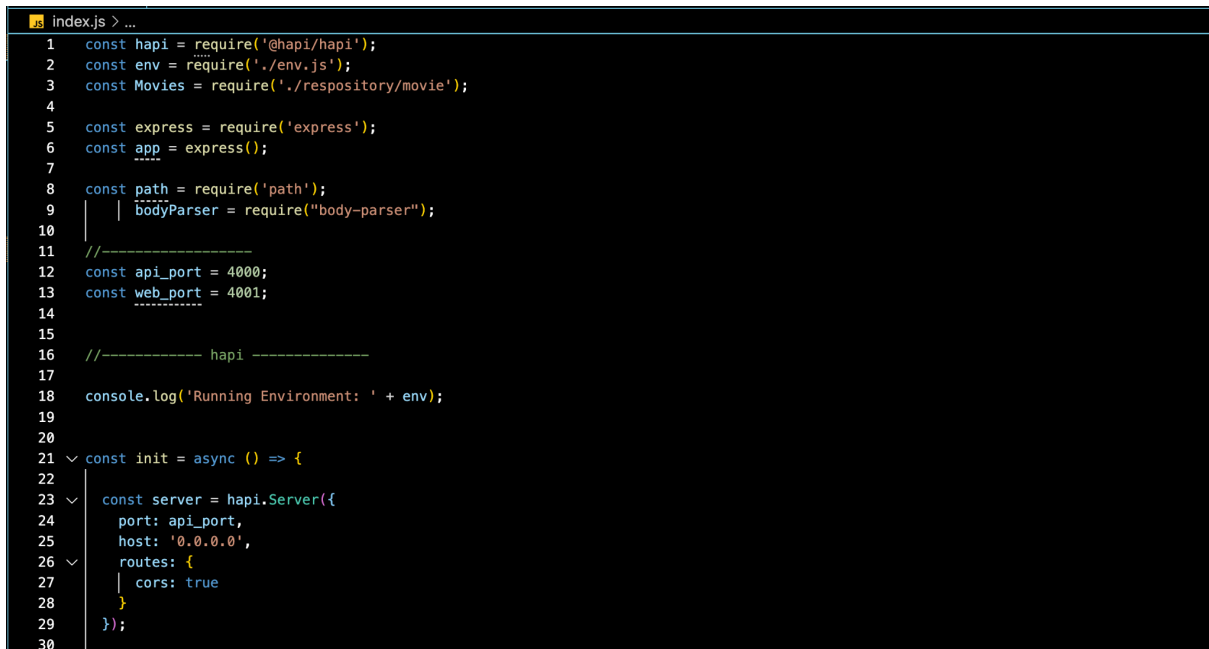
## แสดงข้อมูล

สร้างหน้า index.js มาใน backend และใส่โค้ด



```
1 const hapi = require('@hapi/hapi');
2 const env = require('./env.js');
3 const Movies = require('./repository/movie');
4
5 const express = require('express');
6 const app = express();
7
8 const path = require('path');
9 | bodyParser = require("body-parser");
10
11 //-----
12 const api_port = 4000;
13 const web_port = 4001;
14
15
16 //----- hapi -----
17
18 console.log('Running Environment: ' + env);
```

โค้ดหน้า index.js



```
1 const hapi = require('@hapi/hapi');
2 const env = require('./env.js');
3 const Movies = require('./repository/movie');
4
5 const express = require('express');
6 const app = express();
7
8 const path = require('path');
9 | bodyParser = require("body-parser");
10
11 //-----
12 const api_port = 4000;
13 const web_port = 4001;
14
15
16 //----- hapi -----
17
18 console.log('Running Environment: ' + env);
19
20
21 const init = async () => {
22
23   const server = hapi.Server({
24     port: api_port,
25     host: '0.0.0.0',
26     routes: {
27       cors: true
28     }
29   });
30 }
```

```

31 //-----
32
33 await server.register(require('@hapi/inert'));
34
35 server.route({
36   method: "GET",
37   path: "/",
38   handler: () => {
39     return '<h3> Welcome to API Back-end Ver. 1.0.0</h3>';
40   }
41 });
42
43
44 //API: http://localhost:3001/api/movie/all
45 server.route({
46   method: 'GET',
47   path: '/api/movie/all',
48   config: {
49     cors: {
50       origin: ['*'],
51       additionalHeaders: ['cache-control', 'x-requested-width']
52     }
53   },
54   handler: async function (request, reply) {
55     //var param = request.query;
56     //const category_code = param.category_code;
57
58     try {
59
60       const respondedata = await Movies.MovieRepo.getMovieList();
61       if (respondedata.error) {
62         return respondedata.errMessage;
63       } else {
64         return respondedata;
65       }
66     } catch (err) {
67       server.log(["error", "home"], err);
68       return err;
69     }
70   }
71 });
72
73
74 server.route({
75   method: 'GET',
76   path: '/api/movie/search',
77   config: {
78     cors: {
79       origin: ['*'],
80       additionalHeaders: ['cache-control', 'x-requested-width']
81     }
82   },
83   handler: async function (request, reply) {
84     var param = request.query;
85     const search_text = param.search_text;
86     //const title = param.title;
87
88     try {
89
90       const respondedata = await Movies.MovieRepo.getMovieSearch(search_text);
91       if (respondedata.error) {
92         return respondedata.errMessage;
93       } else {
94         return respondedata;
95       }
96     } catch (err) {
97       server.log(["error", "home"], err);
98       return err;

```

```

99     }
100   }
101 }
102 });
103
104
105 server.route({
106   method: 'POST',
107   path: '/api/movie/insert',
108   config: {
109     payload: {
110       multipart: true,
111     },
112     cors: {
113       origin: ['*'],
114       additionalHeaders: ['cache-control', 'x-requested-width']
115     }
116   },
117   handler: async function (request, reply) {
118
119     const {
120       title,
121       genre,
122       director,
123       release_year
124     } = request.payload;
125
126     //const title = request.payload.title;
127     //const genre = request.payload.genre;
128
129     try {
130       const respondedata = await Movies.MovieRepo.postMovie(title, genre, director, release_year);
131       if (respondedata.error) {
132
133         return respondedata.errMessage;
134       } else {
135         return respondedata;
136       }
137     } catch (err) {
138       server.log(["error", "home"], err);
139       return err;
140     }
141   }
142 });
143
144
145
146
147
148 await server.start();
149 console.log('API Server running on %s', server.info.uri);
150
151 //-----
152 };
153
154
155 process.on('unhandledRejection', (err) => {
156
157   console.log(err);
158   process.exit(1);
159 });
160
161 init();

```

## สร้างหน้า env.js เพื่อเก็บ config

The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a file tree with the following structure:

- EXPLORED
- OPEN EDITORS
  - index.js
  - env.js
- API
  - node\_modules
  - respository
    - movie.js
    - dbconfig.js
    - env.js

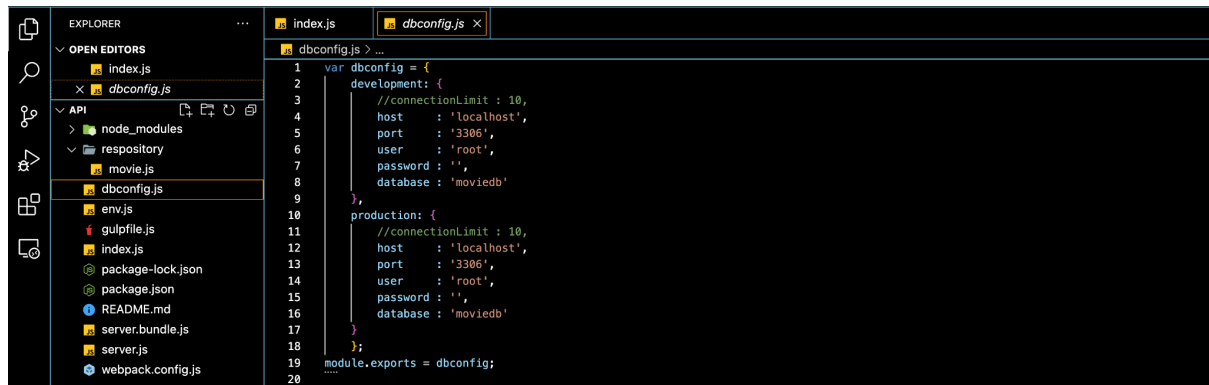
The main editor area shows the content of the selected file, `env.js`:

```

1 var env = process.env.NODE_ENV || 'development';
2 //var env = process.env.NODE_ENV || 'production';
3 module.exports = env;
4 .....
5

```

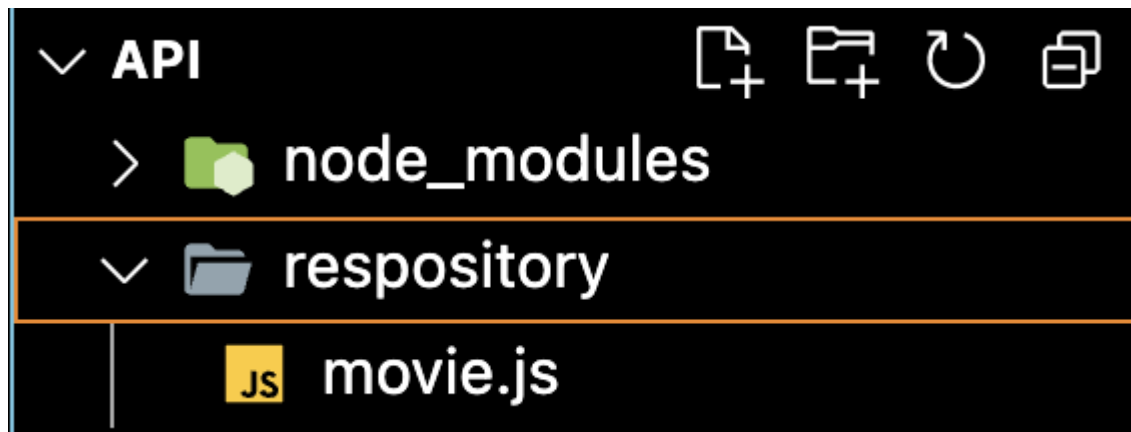
## สร้างหน้า dbconfig.js



The screenshot shows the VS Code interface. In the Explorer view on the left, the file `dbconfig.js` is selected under the `repository` folder. The Editor view on the right displays the content of `dbconfig.js`:

```
1 var dbconfig = {  
2   development: {  
3     //connectionLimit : 10,  
4     host      : 'localhost',  
5     port      : '3306',  
6     user       : 'root',  
7     password   : '',  
8     database   : 'moviedb'  
9   },  
10  production: {  
11    //connectionLimit : 10,  
12    host      : 'localhost',  
13    port      : '3306',  
14    user       : 'root',  
15    password   : '',  
16    database   : 'moviedb'  
17  }  
18 };  
19 module.exports = dbconfig;  
20
```

## สร้างโฟลเดอร์ repository และสร้างไฟล์ movie.js ใน repository



## ใส่โค้ดเข้าไปใน movie.js พร้อมแก้ไขโค้ดบางส่วน

```
respository > movie.js > ...
1  var mysql = require("mysql");
2  const env = require("../env.js");
3  const config = require("../dbconfig.js")[env];
4
5  /*
6
7  async function getMovieList() {
8
9      var Query;
10     var pool = mysql.createPool(config);
11
12     return new Promise((resolve, reject) => {
13
14         //Query = `SELECT * FROM movies WHERE warehouse_status = 1 ORDER BY CONVERT( warehouse_name USING tis620 ) ASC `;
15         Query = `SELECT * FROM movies`;
16
17         pool.query(Query, function (error, results, fields) {
18             if (error) throw error;
19
20             if (results.length > 0) {
21                 pool.end();
22                 return resolve({
23                     statusCode: 200,
24                     returnCode: 1,
25                     data: results,
26                 });
27             } else {
28                 pool.end();
29                 return resolve({
30                     statusCode: 404,
31                     returnCode: 11,
32                     message: 'No movie found',
33                 });
34             }
35         });
36     });
37 }
38
39
40
41
42 */
43
44 async function getMovieList() {
45     var Query;
46     var pool = mysql.createPool(config);
47
48     return new Promise((resolve, reject) => {
49         //Query = `SELECT * FROM movies WHERE warehouse_status = 1 ORDER BY CONVERT( warehouse_name USING tis620 ) ASC `;
50         Query = `SELECT * FROM movies`;
51
52         pool.query(Query, function (error, results, fields) {
53             if (error) throw error;
54
55             if (results.length > 0) {
56                 pool.end();
57                 return resolve(results);
58             } else {
59                 pool.end();
60                 return resolve({
61                     statusCode: 404,
62                     returnCode: 11,
63                     message: "No movie found",
64                 });
65             }
66         });
67     });
68 }
```

```

69
70 async function getMovieSearch(search_text) {
71   var Query;
72   var pool = mysql.createPool(config);
73
74   return new Promise((resolve, reject) => {
75     Query = `SELECT * FROM movies WHERE title LIKE '${search_text}%'`;
76
77     pool.query(Query, function (error, results, fields) {
78       if (error) throw error;
79
80       if (results.length > 0) {
81         pool.end();
82         return resolve({
83           statusCode: 200,
84           returnCode: 1,
85           data: results,
86         });
87       } else {
88         pool.end();
89         return resolve({
90           statusCode: 404,
91           returnCode: 11,
92           message: "No movie found",
93         });
94       }
95     });
96   });
97 }
98
99 async function postMovie(p_title, p_genre, p_director, p_release_year) {
100   var Query;
101   var pool = mysql.createPool(config);
102

```

```

103   return new Promise((resolve, reject) => {
104     //Query = `SELECT * FROM movies WHERE title LIKE '${search_text}%'`;
105
106     var post = {
107       title: p_title,
108       genre: p_genre,
109       director: p_director,
110       release_year: p_release_year,
111     };
112
113     console.log("post is: ", post);
114
115     Query = "INSERT INTO movies SET ?";
116     pool.query(Query, post, function (error, results, fields) {
117       //pool.query(Query, function (error, results, fields) {
118
119       //if (error) throw error;
120       if (error) {
121         console.log("error_code_msg ", error.code + ":" + error.sqlMessage);
122         pool.end();
123         return resolve({
124           statusCode: 405,
125           returnCode: 9,
126           message: error.code + "+" + error.sqlMessage,
127         });
128       } else {
129         console.log("results: ", results);
130         if (results.affectedRows > 0) {
131           pool.end();
132           return resolve({
133             statusCode: 200,
134             returnCode: 1,
135             message: "Movie list was inserted",
136           });

```

```

137         }
138       }
139     });
140   });
141 }
142
143 module.exports.MovieRepo = {
144   getMovieList: getMovieList,
145   getMovieSearch: getMovieSearch,
146   postMovie: postMovie,
147 };

```



## ทดสอบการทำงานใน Postman