

LAB 4

```
You, 3 hours ago | 2 authors (bbachi and others)
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 ./index.js localhost 4000"
10  },
```

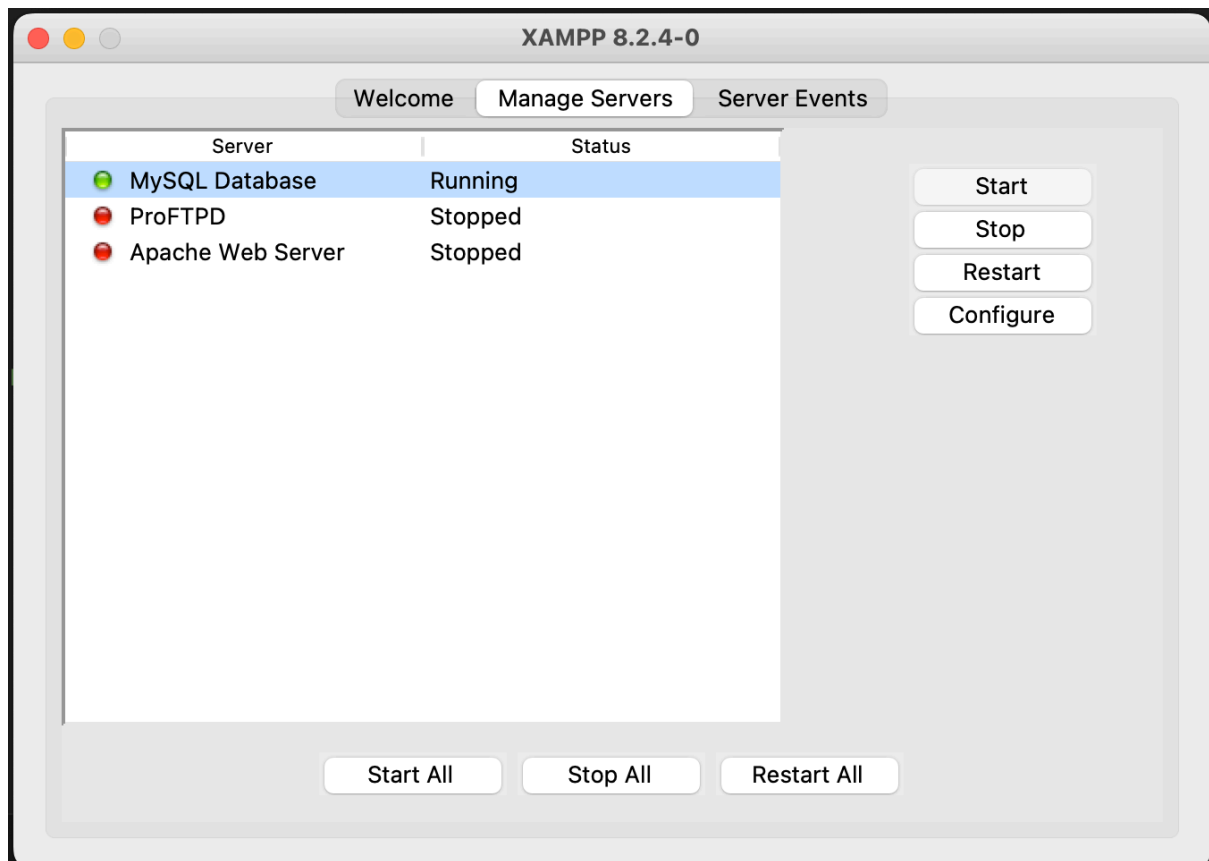
```
JS server.js > ...
You, 3 hours ago | 2 authors (bbachi and others)
1  const express = require('express');
2  const path = require('path');
3  const app = express(),
4    bodyParser = require("body-parser"); 483.9k (gzipped: 211.2k)
5    port = 4000;
6
```

เปลี่ยน Port ของ backend เป็น 4000 หน้า package.json , server.js

```
<  →  localhost:4000/api/users
[{"firstName":"first1","lastName":"last1","email":"abc@gmail.com"}, {"firstName":"first2","lastName":"last2","email":"abc@gmail.com"}, {"firstName":"first3","lastName":"last3","email":"abc@gmail.com"}]
```

<http://localhost:4000/api/users>

ทดสอบ ลองรันดู



เปิด MySQL ใน Xampp

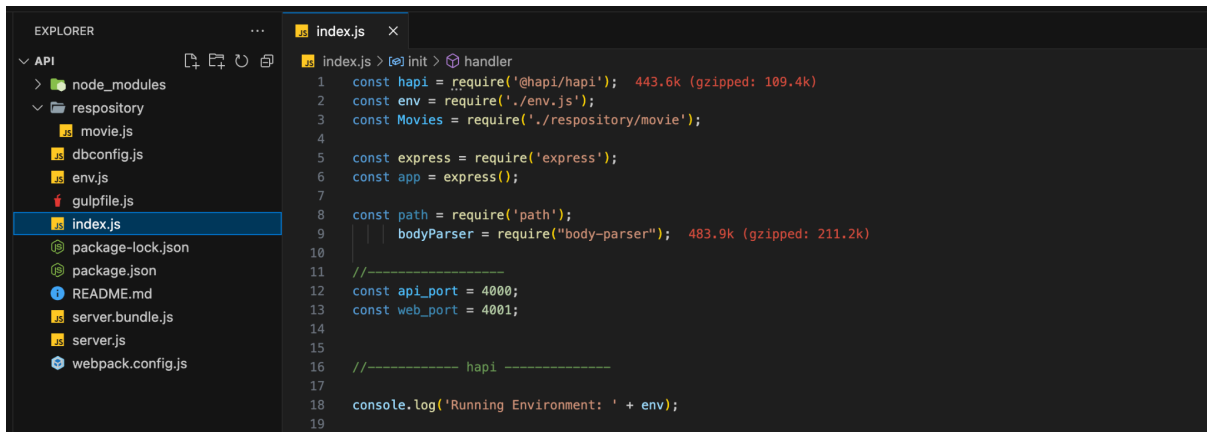


สร้าง database , เพิ่มตารางและเพิ่มข้อมูลลงในตาราง

title	genre	director	release_year
Forrest Gump	Comedy	Todd Phillips	2019
Joker	psychological thriller	Todd Phillips	2019

SELECT * FROM movies;

แสดงข้อมูล



สร้างหน้า index.js มาใน backend และใส่โค้ด

```
const hapi = require('@hapi/hapi');
const env = require('./env.js');
const Movies = require('./respository/movie');

const express = require('express');
const app = express();

const path = require('path');
    bodyParser = require("body-parser");

//-----
const api_port = 4000;
const web_port = 4001;

//----- hapi -----

console.log('Running Environment: ' + env);

const init = async () => {

    const server = hapi.Server({
        port: api_port,
        host: '0.0.0.0',
        routes: {
            cors: true
        }
    });

    //-----
    await server.register(require('@hapi/inert'));
    server.route({
```

```

    method: "GET",
    path: "/",
    handler: () => {
        return '<h3> Welcome to API Back-end Ver. 1.0.0</h3>';
    }
});

//API: http://localhost:3001/api/movie/all
server.route({
    method: 'GET',
    path: '/api/movie/all',
    config: {
        cors: {
            origin: ['*'],
            additionalHeaders: ['cache-control', 'x-requested-width']
        }
    },
    handler: async function (request, reply) {
        //var param = request.query;
        //const category_code = param.category_code;
        try {
            const responsedata = await Movies.MovieRepo.getMovieList();
            if (responsedata.error) {
                return responsedata.errMessage;
            } else {
                return responsedata;
            }
        } catch (err) {
            server.log(["error", "home"], err);
            return err;
        }
    }
});

server.route({
    method: 'GET',
    path: '/api/movie/search',
    config: {
        cors: {
            origin: ['*'],
            additionalHeaders: ['cache-control', 'x-requested-width']
        }
    },
    handler: async function (request, reply) {
        var param = request.query;

```

```

        const search_text = param.search_text;
        //const title = param.title;
        try {
            const respondedata = await
Movies.MovieRepo.getMovieSearch(search_text);
            if (respondedata.error) {
                return respondedata.errMessage;
            } else {
                return respondedata;
            }
        } catch (err) {
            server.log(["error", "home"], err);
            return err;
        }
    }
});

server.route({
    method: 'POST',
    path: '/api/movie/insert',
    config: {
        payload: {
            multipart: true,
        },
        cors: {
            origin: ['*'],
            additionalHeaders: ['cache-control', 'x-requested-width']
        }
    },
    handler: async function (request, reply) {
        const {
            title,
            genre,
            director,
            release_year
        } = request.payload;
        //const title = request.payload.title;
        //const genre = request.payload.genre;
        try {
            const respondedata = await Movies.MovieRepo.postMovie(title, genre,
director,release_year);
            if (respondedata.error) {
                return respondedata.errMessage;
            } else {
                return respondedata;
            }
        }
    }
});

```

```

    }
  } catch (err) {
    server.log(["error", "home"], err);
    return err;
  }
}

});

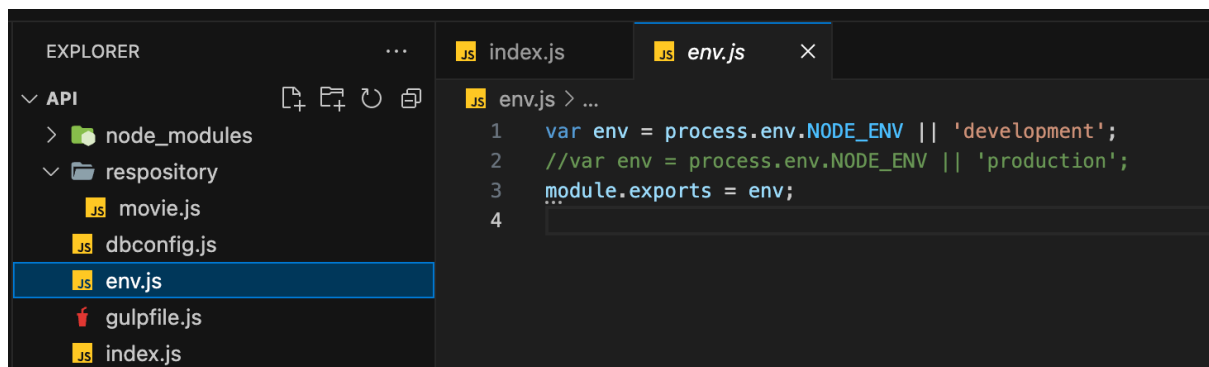
await server.start();
console.log('API Server running on %s', server.info.uri);
//-----
};

process.on('unhandledRejection', (err) => {
  console.log(err);
  process.exit(1);
});

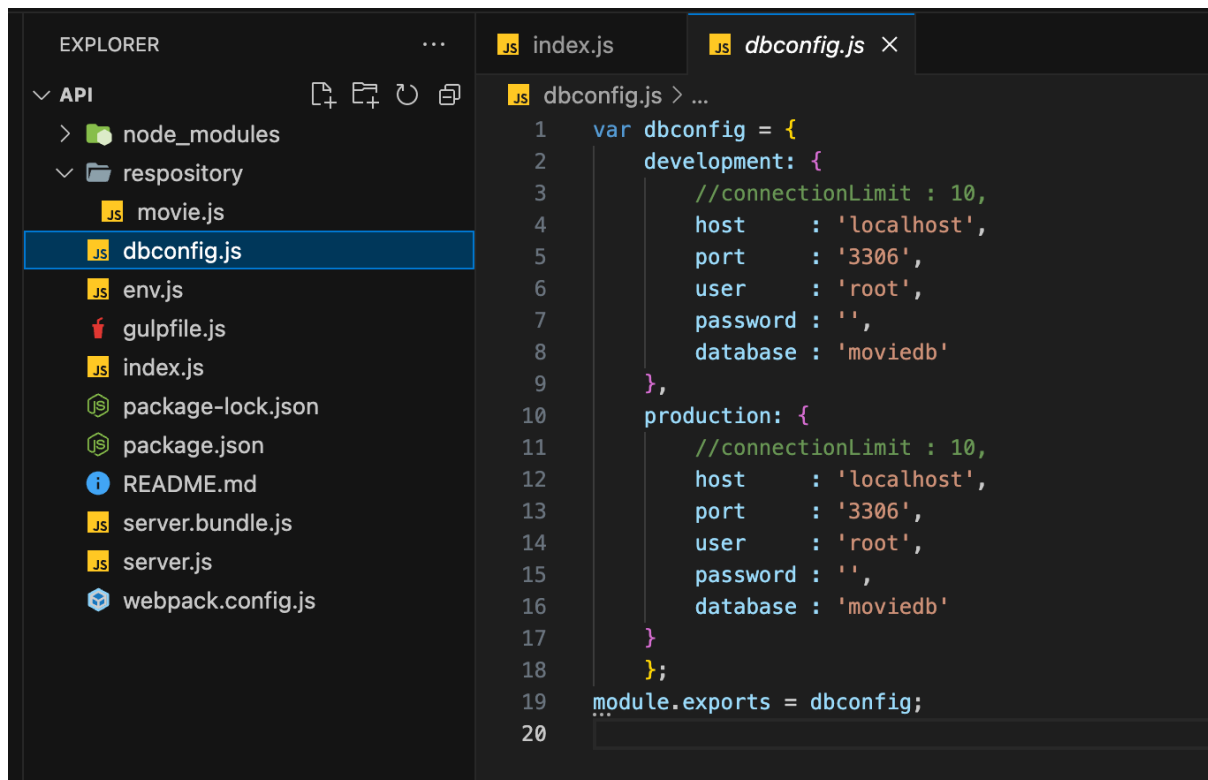
init();

```

โค้ดหน้า index.js



สร้างหน้า env.js เพื่อเก็บ config ว่าทำงานอยู่ในสภาพแวดล้อมใด



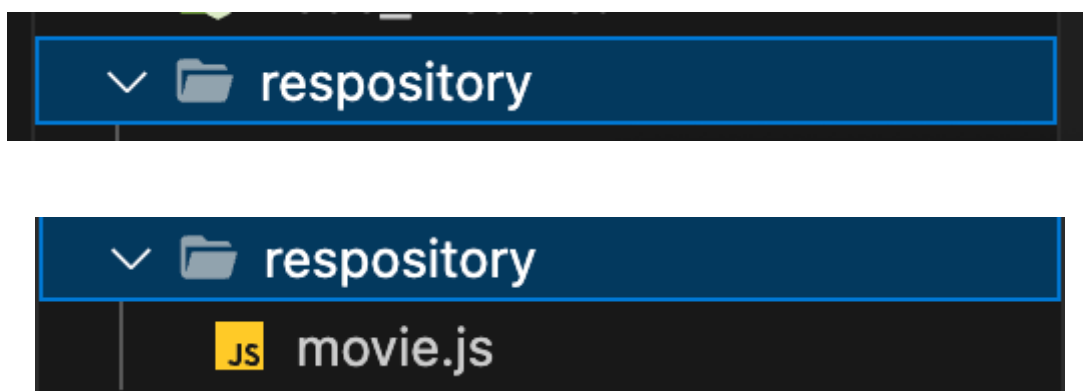
The screenshot shows the VS Code interface. On the left, the Explorer sidebar displays a file tree with the following structure:

- API
 - node_modules
 - repository
 - movie.js
 - dbconfig.js** (selected)
 - env.js
 - gulpfile.js
 - index.js
 - package-lock.json
 - package.json
 - README.md
 - server.bundle.js
 - server.js
 - webpack.config.js


The main editor window shows 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
```

สร้างหน้า dbconfig.js



สร้างโฟลเดอร์ repository และสร้างไฟล์ movie.js ด้วย



The screenshot shows the content of `movie.js`:

```
var mysql = require("mysql");
const env = require("../env.js");
const config = require("../dbconfig.js")[env];

/*
async function getMovieList() {

  var Query;
  var pool  = mysql.createPool(config);
```

```

    return new Promise((resolve, reject) => {
        //Query = `SELECT * FROM movies WHERE warehouse_status = 1 ORDER BY CONVERT(
warehouse_name USING tis620 ) ASC `;
        Query = `SELECT * FROM movies`;
        pool.query(Query, function (error, results, fields) {
            if (error) throw error;

            if (results.length > 0) {
                pool.end();
                return resolve({
                    statusCode: 200,
                    returnCode: 1,
                    data: results,
                });
            } else {
                pool.end();
                return resolve({
                    statusCode: 404,
                    returnCode: 11,
                    message: 'No movie found',
                });
            }

        });
    });
});

}
*/

async function getMovieList() {
    var Query;
    var pool = mysql.createPool(config);

    return new Promise((resolve, reject) => {
        //Query = `SELECT * FROM movies WHERE warehouse_status = 1 ORDER BY CONVERT(
warehouse_name USING tis620 ) ASC `;
        Query = `SELECT * FROM movies`;

        pool.query(Query, function (error, results, fields) {
            if (error) throw error;

            if (results.length > 0) {
                pool.end();
            }
        });
    });
}

```



```

        return resolve(results);
    } else {
        pool.end();
        return resolve({
            statusCode: 404,
            returnCode: 11,
            message: "No movie found",
        });
    }
});
});
}

async function getMovieSearch(search_text) {
    var Query;
    var pool = mysql.createPool(config);

    return new Promise((resolve, reject) => {
        Query = `SELECT * FROM movies WHERE title LIKE '%${search_text}%'`;

        pool.query(Query, function (error, results, fields) {
            if (error) throw error;

            if (results.length > 0) {
                pool.end();
                return resolve({
                    statusCode: 200,
                    returnCode: 1,
                    data: results,
                });
            } else {
                pool.end();
                return resolve({
                    statusCode: 404,
                    returnCode: 11,
                    message: "No movie found",
                });
            }
        });
    });
}

async function postMovie(p_title, p_genre, p_director, p_release_year) {
    var Query;
    var pool = mysql.createPool(config);

```

```

return new Promise((resolve, reject) => {
  //Query = `SELECT * FROM movies WHERE title LIKE '%${search_text}%'`;

  var post = {
    title: p_title,
    genre: p_genre,
    director: p_director,
    release_year: p_release_year,
  };

  console.log("post is: ", post);

  Query = "INSERT INTO movies SET ?";
  pool.query(Query, post, function (error, results, fields) {
    //pool.query(Query, function (error, results, fields) {

    // if (error) throw error;

    console.log("error_msg: ", error.code + ":" + error.sqlMessage);
    if (error) {
      pool.end();
      return resolve({
        statusCode: 300,
        returnCode: 6,
        message: error.code + ": " + error.sqlMessage,
      });
    } else {
      console.log("result: ", results);
      if (results.affectedRows > 0) {
        pool.end();
        return resolve({
          statusCode: 200,
          returnCode: 1,
          message: "Movie list was inserted",
        });
      }
    }
  });
});

module.exports.MovieRepo = {
  getMovieList: getMovieList,
  getMovieSearch: getMovieSearch,

```

```
postMovie: postMovie,  
};
```

ในบรรทัดที่ 132 ถึง 141 แก่โค้ดใหม่ดังนี้

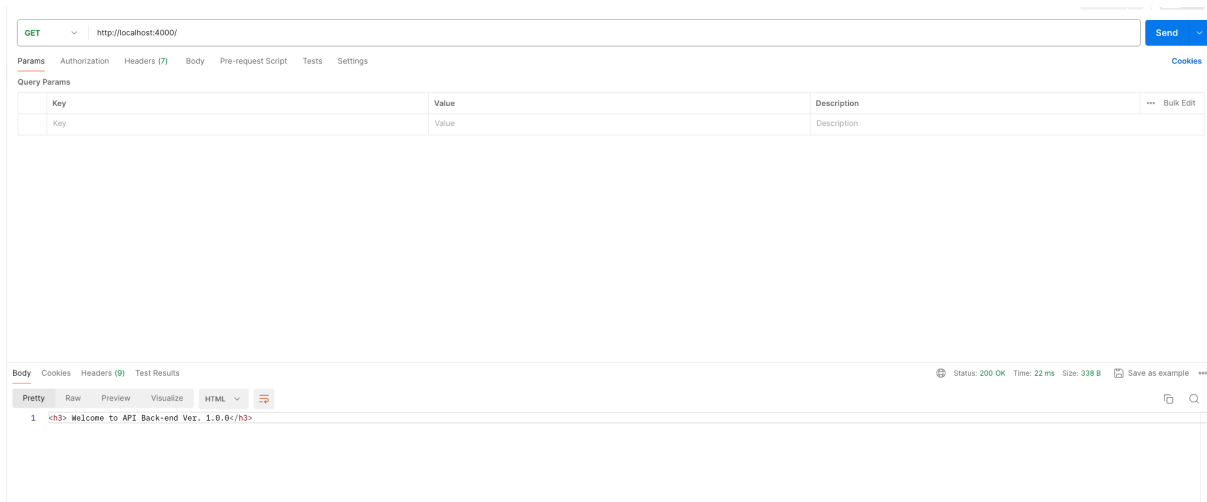
```
        if (error) throw error;  
  
        if (results.length > 0) {  
            pool.end();  
            return resolve({  
                statusCode: 200,  
                returnCode: 1,  
                message: 'Movie list was inserted',  
            });  
        }  
    });  
});  
}
```

โค้ดเดิม

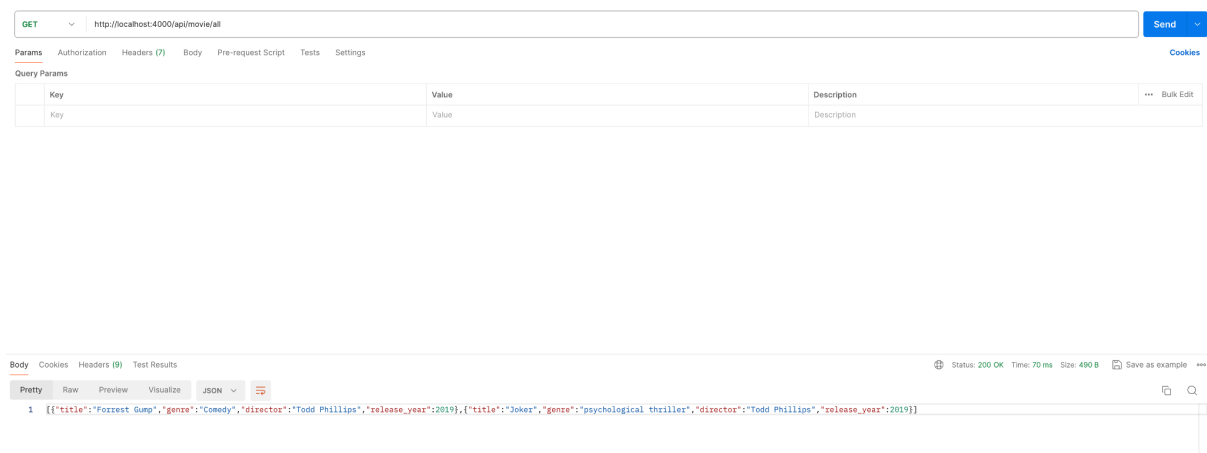
```
console.log("error_msg: ", error.code + ":" + error.sqlMessage);
if (error) {
  pool.end();
  return resolve({
    statusCode: 300,
    returnCode: 6,
    message: error.code + ":" + error.sqlMessage,
  });
} else {
  console.log("result: ", results);
  if (results.affectedRows > 0) {
    pool.end();
    return resolve({
      statusCode: 200,
      returnCode: 1,
      message: "Movie list was inserted",
    });
  }
}
```

โค้ดใหม่

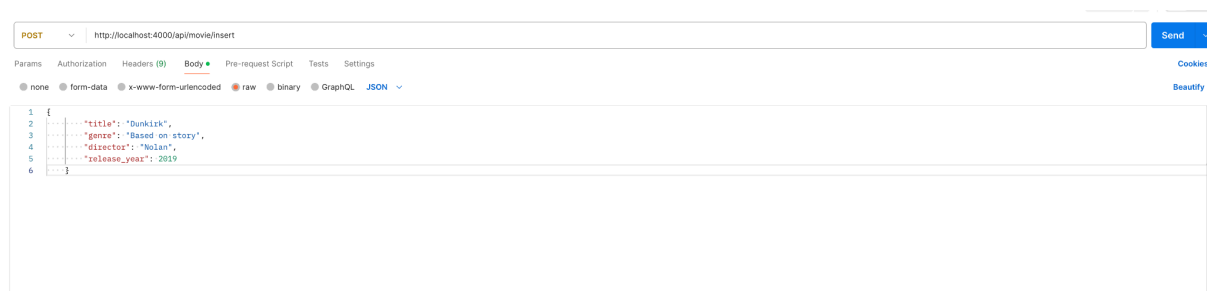
ทดสอบการทำงานใน Postman



<http://localhost:4000/>

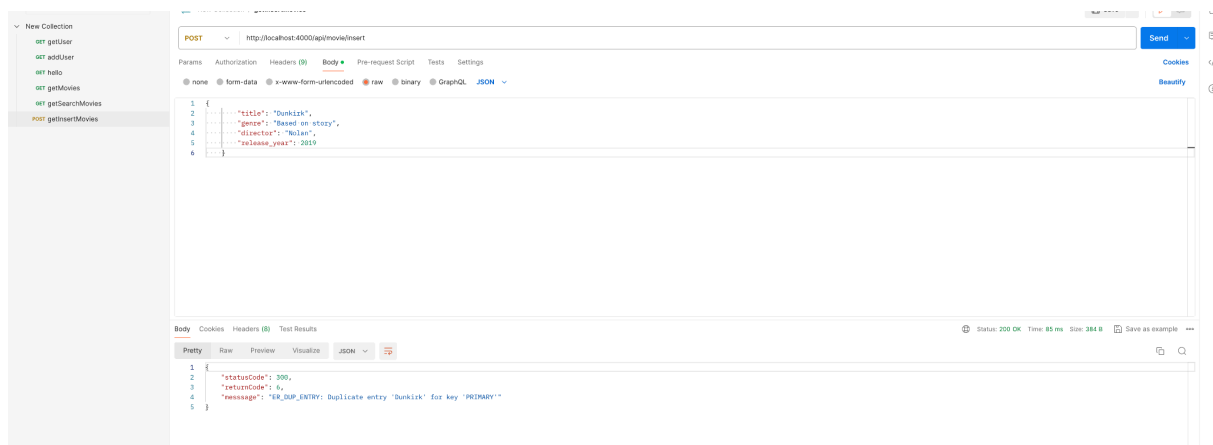


<http://localhost:4000/api/movie/all>



<http://localhost:4000/api/movie/insert>

```
Running Environment: development
API Server running on http://0.0.0.0:4000
post is: {
  title: 'Dunkirk',
  genre: 'Based on story',
  director: 'Nolan',
  release_year: 2019
}
```



กรณี insert ข้อมูลซ้ำ <http://localhost:4000/api/movie/insert>

```
> react-nodejs-example@1.0.0 dev
> nodemon ./index.js localhost 4000

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node ./index.js localhost 4000`
Running Environment: development
API Server running on http://0.0.0.0:4000
post is: {
  title: 'Dunkirk',
  genre: 'Based on story',
  director: 'Nolan',
  release_year: 2019
}
error_msg: ER_DUP_ENTRY: Duplicate entry 'Dunkirk' for key 'PRIMARY'
[]
```

กรณี insert ข้อมูลซ้ำ <http://localhost:4000/api/movie/insert>

[illegible]