Date:30/05/2018

Database connectivity in express

-------------------------------------------

So for that you need to install xampp server in which my sql Is pre installed then

Create a basic application folder in express js and run the command .

To install the “mysql” through node.

“npm install mysql –save”

Then configure the database file in your project .

Create one file inside the project and import it into your app.js file

Like your database file name is “dbcon.js”

Write the code in “dbcon.js” file –

var mysql = require('mysql');

var con = mysql.createConnection({

host : 'localhost',

user : 'root',

password : '',

database : 'exp\_test'

});

// connnect

con.connect((error) => {

if(error){

throw error;

}

console.log("database is connected");

})

“app.js” file

var createError = require('http-errors');

var express = require('express');

var mysql = require('mysql');

var path = require('path');

var cookieParser = require('cookie-parser');

var logger = require('morgan');

// define the route variable

var indexRouter = require('./routes/index');

var usersRouter = require('./routes/users');

var demoRouter = require('./routes/demo');

var app = express();

// import database connection

var dbcon = require('./dbcon');

// view engine setup

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'jade');

app.use(logger('dev'));

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

// use the route variable

app.use('/', indexRouter);

app.use('/users', usersRouter);

app.use('/demo' , demoRouter);

// catch 404 and forward to error handler

app.use(function(req, res, next) {

next(createError(404));

});

// error handler

app.use(function(err, req, res, next) {

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

// render the error page

res.status(err.status || 500);

res.render('error');

});

module.exports = app;

now run your APPLICATION

you will get one message in terminal that your database is connected.

So for that you have to first create one database which you have mentioned in the connection field and you xammp server should run on that time, otherwise your connection will net setup it would terminate.

We will perform **CRUD operation in mysql with express js** for that we will create route for each.

To create Database:

Create one route name as – “created.js”

The syntax is look like as –

// include express module

var express = require('express');

var router = express.Router();

var db = require('../dbcon');

router.get('/',(req, res) => {

// res.send("response from the create db routes");

// create a sql query string

let sql ="CREATE DATABASE `nodemysql`";

db.query(sql, (err,result) => {

if(err) throw err;

res.send('database created');

console.log(result);

});

});

// export the module to use in another module

module.exports = router;

now “dbcon.js” code –

const mysql = require('mysql');

const db = mysql.createConnection({

host : 'localhost',

user : 'root',

password : ''

});

// connnect

db.connect((error) => {

if(error){

throw error;

}

console.log("database is connected");

});

module.exports = db;

“app.js” code-

var createError = require('http-errors');

var express = require('express');

var mysql = require('mysql');

var path = require('path');

var cookieParser = require('cookie-parser');

var logger = require('morgan');

// define the route variable

var indexRouter = require('./routes/index');

var usersRouter = require('./routes/users');

var demoRouter = require('./routes/demo');

var createDbRouter = require('./routes/createdb');

var app = express();

// import database connection

var dbcon = require('./dbcon');

// view engine setup

app.set('views', path.join(\_\_dirname, 'views'));

app.set('view engine', 'jade');

app.use(logger('dev'));

app.use(express.json());

app.use(express.urlencoded({ extended: false }));

app.use(cookieParser());

app.use(express.static(path.join(\_\_dirname, 'public')));

// use the route variable

app.use('/', indexRouter);

app.use('/users', usersRouter);

app.use('/demo' , demoRouter);

app.use('/createdb',createDbRouter);

// catch 404 and forward to error handler

app.use(function(req, res, next) {

next(createError(404));

});

// error handler

app.use(function(err, req, res, next) {

// set locals, only providing error in development

res.locals.message = err.message;

res.locals.error = req.app.get('env') === 'development' ? err : {};

// render the error page

res.status(err.status || 500);

res.render('error');

});

module.exports = app;

Create a Table –

For this make a seprate route name as “createtable”

The syntax is like-

// load express module

const express = require('express');

const router = express.Router();

// import the datbase connection

var con = require('../dbcon');

router.get('', (req, res)=> {

// res.send("response from the create table routes");

var sql = "CREATE TABLE `nodemysql`.`post` ( `id` INT(10) NOT NULL AUTO\_INCREMENT , `post\_title` VARCHAR(255) NOT NULL , `post\_author` VARCHAR(255) NOT NULL , `post\_descp` VARCHAR(500) NOT NULL , PRIMARY KEY (`id`)) ENGINE = InnoDB;";

con.query(sql, (err,result) => {

if(err) throw err;

res.send("table is created");

console.log(result);

});

});

// export this module to use outside

module.exports = router;

Insert the data into the table :

“insertdata.js”

// load express module

const express = require('express');

const router = express.Router();

// import the database connection

var con = require('../dbcon');

router.get('/', (req, res) =>{

// res.send("response from the insert router");

let sql = "INSERT INTO `post` (`id`, `post\_title`, `post\_author`, `post\_descp`) VALUES (NULL, 'node\_my\_sql\_tutorial', 'wishvanath', 'this tutorial contains connectivity of database with express')";

con.query(sql, (err,result) =>{

if(err) throw err;

res.send("data is inserted successfully");

console.log(result);

});

});

// export the router to use it outside

module.exports = router;

show the data from the backend of the mysql in express

“todolist.js”

// load the express module

const express = require('express');

const router = express.Router();

// import the database connection

var con = require('../dbcon');

router.get('/', (req, res) =>{

// res.send("response send from the todolist router");

let sql = "SELECT \* FROM `todolist`";

con.query(sql,(err, result) => {

let data = result;

console.log(result);

return res.json(result);

});

});

// export to use in another module

module.exports = router;

client >package.json

{

"name": "todoapp",

"version": "0.1.0",

"private": true,

"dependencies": {

"axios": "^0.18.0",

"react": "^16.3.2",

"react-dom": "^16.3.2",

"react-scripts": "1.1.4"

},

"scripts": {

"start": "react-scripts start",

"build": "react-scripts build",

"test": "react-scripts test --env=jsdom",

"eject": "react-scripts eject"

},

"proxy": "http://localhost:3001/"

}

Read the data from the backend of express and my sql

Now we have created the “todolist.js” routes in express in which they have all the data from the backend so now through the react we are going to access it

In react “app.js” file –

import React from 'react';

class App extends React.Component {

state = {

data:[]

}

componentDidMount(){

fetch('/todolist')

.then(res => res.json())

.then(data => this.setState({data},()=>(console.log(data))))

}

render() {

let data = this.state.data

return (

<div>

<Header />

<h1>Your Data from backend is :</h1>

{JSON.stringify(data)}

{data.map((data => <List key={data.id} data={data}/>))}

</div>

);

}

}

// list componet

class List extends React.Component{

render(){

return(

<div>

<li>{this.props.data.items}</li>

</div>

);

}

}

class Header extends React.Component{

render(){

return(

<div><h2>TO do Items from database</h2></div>

);

}

}

export default App;

now deploy your project

remember need to run three server

one for node application(express server)

second for react application

and third for mysql(apache)