Building Restful Service with Express

|  |
| --- |
| // console.log('\*\*\*\*\*\*\*\*\*\*Built in module: Http\*\*\*\*\*\*\*\*\*\*\*');  const http = require('http');  const server = http.createServer((req,res)=>{  if(req.url==='/'){  res.write('Hello World');  res.end();  }  if(req.url==='/api/courses'){  res.write(JSON.stringify([1,2,3,4]));  res.end();  }  });  // server.on('connection', (socket) => {  // console.log('New connection...');  // });  server.listen(3000);  console.log('listing on port 3000...'); |

Express is fast and light weight framework for building web application.

Next we will look at RESTful Services.

# Restful Services

A picture containing drawing

Description automatically generated

A picture containing clock

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

A picture containing device

Description automatically generated

<https://www.npmjs.com/>

<http://expressjs.com/>

create an directory

$ mkdir express-demo

$cd express-demo

express-demo$ npm init –yes

express-demo$ npm i express

create a index.js file inside express-demo folder

|  |
| --- |
| const express = require('express');  const app = express();  app.get('/', (req, res) => {  //callback function called route handler  res.send('Hello World');  });  app.get('/api/courses', (req, res) => {  res.send([1, 2, 3, 4]);  });  app.listen(3000, () => console.log('Listning on port 3000...'));  // app.post();  // app.put();  // app.delete(); |

Run the file

$ node index.js

Listening on port 3000...

Open the browser : http://localhost:300

A screenshot of a cell phone

Description automatically generated

Open the browser: <http://localhost:3000/api/courses>

A screenshot of a cell phone

Description automatically generated

# Nodemon – for node monitor

$ sudo npm i -g nodemon

A screenshot of a cell phone

Description automatically generated

$ nodemon index.js

A picture containing screen, phone, holding

Description automatically generated

Make a simple change in index.js

const express = require('express');

const app = express();

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

//callback function called route handler

res.send('Hello World !!!');

});

app.get('/api/courses', (req, res) => {

res.send([1, 2, 3, 4]);

});

app.listen(3000, () => console.log('Listning on port 3000...'));

// app.post();

// app.put();

//app.delete();

Go to terminal

A picture containing table, green

Description automatically generated

Go and refresh the page

A screenshot of a cell phone

Description automatically generated

# Environment variable

//Creating a enviornment variable

//PORT

const port = process.env.PORT || 3000;

app.listen(port, () => console.log(`Listning on port ${port}...`));

|  |
| --- |
| const express = require('express');  const app = express();  app.get('/', (req, res) => {  //callback function called route handler  res.send('Hello World !!!');  });  app.get('/api/courses', (req, res) => {  res.send([1, 2, 3, 4]);  });  //Creating a enviornment variable  //PORT  const port = process.env.PORT || 3000;  app.listen(port, () => console.log(`Listning on port ${port}...`));  // app.post();  // app.put();  // app.delete(); |

A picture containing orange, meter, sitting, clock

Description automatically generated

Setting the environment variable:

$ export PORT=5000

$ nodemon index.js

Screen of a cell phone

Description automatically generated

# Route Parameters

app.get('/api/courses/:id', (req, res) => {

res.send('The id parameter ' + req.params.id);

});

A screenshot of a cell phone

Description automatically generated

app.get('/api/courses/:year/:month', (req, res) => {

res.send(req.params);

});

A screenshot of a cell phone

Description automatically generated

Query Parameter

app.get('/api/posts/:year/:month', (req, res) => {

//res.send(req.params);

res.send(req.query);

});

<http://localhost:5000/api/posts/2018/1?sortBy=name>

A screenshot of a cell phone

Description automatically generated

# Handling HTTP GET Requests

const express = require('express');

const app = express();

const courses = [

{ id: 1, name: 'course1' },

{ id: 2, name: 'course2' },

{ id: 3, name: 'course3' }

];

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

//callback function called route handler

res.send('Hello World !!!');

});

app.get('/api/courses', (req, res) => {

res.send(courses);

});

app.get('/api/courses/:id', (req, res) => {

const course = courses.find((c) => c.id === parseInt(req.params.id));

if (!course) res.status(404).send('The course with a given Id was not found');

res.send(course);

});

app.get('/api/posts/:year/:month', (req, res) => {

//res.send(req.params);

res.send(req.query);

});

//app.post();

// app.put();

// app.delete();

//Creating a enviornment variable

//PORT

const port = process.env.PORT || 3000;

app.listen(port, () => console.log(`Listning on port ${port}...`));

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# Handling HTTP POST Requests

app.post('/api/courses', (req, res) => {

let course = {

id: courses.length + 1,

name: req.body.name,

};

courses.push(course);

res.send(course);

});

Using Rest client

Request.http file

POST http://localhost:3000/api/courses/

Content-Type: application/json

{

"name": "course5"

}

A screenshot of a cell phone

Description automatically generated

Input validations

app.post('/api/courses', (req, res) => {

if (!req.body.name || req.body.name.length < 3) {

//400 BAD request

res.status(400).send('Name is required and should be minimum 3 charcters');

return;

}

let course = {

id: courses.length + 1,

name: req.body.name,

};

courses.push(course);

res.send(course);

});

POST http://localhost:3000/api/courses/

Content-Type: application/json

{

"name": "co"

}

A screenshot of a cell phone

Description automatically generated

JOI

const Joi = require('joi');

app.post('/api/courses', (req, res) => {

const schema = {

name: Joi.string().min(3).required()

};

const result = Joi.validate(req.body, schema);

// console.log(result);

// if (!req.body.name || req.body.name.length < 3) {

// //400 BAD request

// res.status(400).send('Name is required and should be minimum 3 charcters');

// return;

// }

if (result.error) {

//400 BAD request

//res.status(400).send(result.error);

res.status(400).send(result.error.details[0].message);

return;

}

let course = {

id: courses.length + 1,

name: req.body.name,

};

courses.push(course);

res.send(course);

});

POST http://localhost:3000/api/courses/

Content-Type: application/json

{

"name": "new course"

}

A screenshot of a cell phone

Description automatically generated

POST http://localhost:3000/api/courses/

Content-Type: application/json

{

"name": "co"

}

A screenshot of a cell phone

Description automatically generated

POST http://localhost:3000/api/courses/

Content-Type: application/json

{

"name": ""

}

A screenshot of a cell phone

Description automatically generated

# Handling HTTP PUT Requests

app.put('/api/courses/:id', (req, res) => {

//look up the course

//if, not existing , return 404

const course = courses.find((c) => c.id === parseInt(req.params.id));

if (!course) res.status(404).send('The course with a given Id was not found');

//validate

//if invalid, return 400 Bad Request

//const result = validateCourse(req.body);

// if (result.error) {

// //400 BAD request

// //res.status(400).send(result.error);

// res.status(400).send(result.error.details[0].message);

// return;

// }

//Object destruction

const { error } = validateCourse(req.body);

if (error) {

//400 BAD request

//res.status(400).send(error);

res.status(400).send(error.details[0].message);

return;

}

//Update course

course.name = req.body.name;

//Return the updated course

res.send(course);

});

function validateCourse(course) {

//validate

//if invalid, return 400 Bad Request

const schema = {

name: Joi.string().min(3).required()

};

return Joi.validate(course, schema);

}

PUT http://localhost:3000/api/courses/1

Content-Type: application/json

{

"name": "new-name"

}

A screenshot of a cell phone

Description automatically generated

GET http://localhost:3000/api/courses/

A screenshot of a cell phone

Description automatically generated

Id doesn’t exist

PUT http://localhost:3000/api/courses/10

Content-Type: application/json

{

"name": "new-name"

}

A screenshot of a cell phone

Description automatically generated

BAD Request

PUT http://localhost:3000/api/courses/1

Content-Type: application/json

{

"name": ""

}

A screenshot of a cell phone

Description automatically generated

# Handling HTTP Delete Requests

app.delete('/api/courses/:id', (req, res) => {

//look up the course

//if, not existing , return 404

const course = courses.find((c) => c.id === parseInt(req.params.id));

if (!course) res.status(404).send('The course with a given Id was not found');

//delete

const index = courses.indexOf(course);

courses.splice(index, 1);

//Return the same course

res.send(course);

});

DELETE http://localhost:3000/api/courses/4

A screenshot of a cell phone

Description automatically generated

DELETE http://localhost:3000/api/courses/40

A screenshot of a cell phone

Description automatically generated