# **The Express\_Server is a REST-Backend for Angular with Node.js & Express**

This is how to install and configure it.

First install node.js by following this guide:

<https://nodejs.org/en/>

On your PC create a folder called: Express\_Server.

Navigate to this folder with the command prompt or Visual Studio and type:

**npm init**

This will generate a package.json file.

Replace the scripts configuration in this package.json with this:

"scripts": {

"start": "nodemon src/index.js",

"test": "echo \"Error: no test specified\" && exit 1"

},

We’ll use nodemon to handel our server rendering. Nodemon will be activated and used when starting up our server with **npm start**.

”nodemon is a tool that helps develop node.js based applications by automatically restarting the node application when file changes in the directory are detected.”

In comand prompt run these commands.

**npm i -g nodemon** // install nodemo

**npm install express** //install express, node\_modules

**npm install cors** //install cors

”CORS - cross-origin-resource sharing” - Will get errors if not used; further configuration is already present in the index.js file.”

This is how the package.json file should look like now

{  
 "name": "express\_server",  
 "version": "1.0.0",  
 "description": "", "main": "index.js",  
 "scripts": {  
 "start": "nodemon src/index.js",  
 "test": "echo \"Error: no test specified\" && exit 1"  
 },  
 "author": "",  
 "license": "ISC",  
 "dependencies": {  
 "cors": "^2.8.5",  
 "express": "^4.17.1"  
 }  
}

… and how the Express\_Server file folder looks like.



Let’s create index.js file with our server configurations in this folder.

Add this script into that file. Or just copy index.js from gitHub folder :  
<https://github.com/keno1se/REST_Project/tree/master/Express_Server>

const express = require('express'); // load express module

const homes = express(); // call express funct

/\*----------\*/

/\*-- CORS --\*/

/\*----------\*/

const cors = require('cors');

const corsOptions = {

  origin: '\*',

  optionsSuccessStatus: 200 // some legacy browsers (IE11, various SmartTVs) choke on 204

}

homes.use(cors(corsOptions))

/\*------------------------------\*/

/\* Application data - hus1/hus2 \*/

/\*------------------------------\*/

const hus1 = {

rooms: [

{

name: 'vardagsrum',

temperature: '21',

humidity: '0.1'

},

{

name: 'kök',

temperature: '27',

humidity: '0.77 '

}

]};

const hus2 = {

rooms: [

{

name: 'vardagsrum',

temperature: '22',

humidity: '0.2'

},

{

name: 'kök',

temperature: '28',

humidity: '0.88 '

}

]};

/\*----------------\*/

/\* Route 1 - hus1 \*/

/\*----------------\*/

homes.get('/homes/hus1/rooms', (req, res) => {

const hus1AllRooms = hus1.rooms;

res.send(hus1AllRooms); // \*see output print below

});

// url

// http://127.0.0.1:3000/homes/hus1/rooms

// \* output print

// [{"name":"vardagsrum","temperature":"21","humidity":"0.1"},

// {"name":"kök","temperature":"27","humidity":"0.77 "}]

/\*----------------\*/

/\* Route 2 - hus2 \*/

/\*----------------\*/

homes.get('/homes/hus2/rooms', (req, res) => {

const hus2AllRooms = hus2.rooms;

res.send(hus2AllRooms); // \*see output print below

});

// url

// http://127.0.0.1:3000/homes/hus2/rooms

// \* output print

// [{"name":"vardagsrum","temperature":"22","humidity":"0.2"},

// {"name":"kök","temperature":"28","humidity":"0.88 "}]

/\*------------------------\*/

/\* port - start up server \*/

/\*------------------------\*/

const port = process.env.PORT || 3000; // if port not set then use 3000,

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

// start server with: npm start

// Server will start listen on port 3000

// In any browser type either of these two URL's to render the application data routes.

// hus1 data - http://127.0.0.1:3000/homes/hus1/rooms

// hus2 data - http://127.0.0.1:3000/homes/hus2/rooms

// Changing any data in the above Application data should be reflected in the browser automatically

The set up of the Exprss\_Server is now finished. Start the server with: **npm Start**. Copy hus1 or hus2 URL to any browser and it will render the data objects. Any changes of the data in the application data, name, temperature, etc…, will be automatically reflected in the browser.

The server responds emmediately on any changes to the applicaton data.

The idea is to connect external sensors to this api to set the application data on a continued input flow.