Expressjs

# Definition

Expressjs is a nodejs web framework

# Usage

It’s using for both server render apps and Api microservices

# Express basic syntax

*//bring express using the common j/s module syntax*

*//notes: we can't use import or es6 modules with node by default. if you do want to use it you'll have to use something*

*//like babel to compile it*

const express = require('express');

*//initialise express*

const app = express();

*//create your endpoints / |route| handlers*

*//here we create a get request to the index route which is the slash*

*//actually we can add a html code*

app.get('/' , function(*req* , *res*){

*//app.get() , app.post() , app.put() , app.delete()*

*//access to params , query string , url parts*

*//express has a router so we can store routes in separate files and export*

    res.send('hello world');

}) ;

*//listen on the port*

app.listen(5000);

# express middleware functions

these are functions that have access to the request and response object express has built in middleware but middleware also comes from 3rd party packages as well as custom middleware

* Execute any code (res.send)
* Make change to the request / response objects (res.json(JSON)
* End response cycle
* Call next middleware in the stack (next())

Every time we make request the middleware gonna run

app.use((req, res, next) => {

console.log('Time:', Date.now())

next()

})

# Routing

Routing refers to how an application’s endpoints (URL) respond to client requests.

These routing methods can have more than one callback function (sometimes called “handler functions”) called when the application receives a request to the specific route and HTTP method. In other words, the application “listens” for request that match the specific routes, and methods and when it detects a match, it calls the specified callback function.

**Express.Router()**

Use the express.Router class to create modular, mountable route handlers. A Router instance is a complete middleware and routing system; for this reason, it is often referred to as a “mini-app”.

The following example creates a router as a module, loads a middleware function in it, defines some routes, and mounts the router module on a path in the main app.

Const express = require(‘express’);

Const router = express.Router();

router.use((req, res, next) => {

console.log('Time: ', Date.now())

next()

})

// define the home page route

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

res.send('Birds home page')

})

// define the about route

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

res.send('About birds')

})

module.exports = router

Then, load the router module in the app:

const birds = require('./birds')

// ...

app.use('/birds', birds)

The app will now be able to handle requests to /birds and /birds/about, as well as call the timeLog middleware function that is specific to the route.

# Response method

|  |  |
| --- | --- |
| method | description |
| Res.downloads() | Prompt a file to be downloaded |
| Res.end() | End the response process |
| Res.json() | Send a json response |
| Res.jsonp() | Send a json response with jsonp support |
| Res.redirect() | Redirect a request |
| Res.render() | Render a view template |
| Res.send() | Send a response to various type |
| Res.sedFile() | Send a file as an octet stream |
| Res.sendStatus() | Set the response status code and send its string representation as the response body. |

# Get started with express

Step 1:

Install the requirement 🡪 nodejs

## Step 2: npm installation

Create the package.json file 🡪 npm init

Instlall express package 🡪 npm I express

Nodemon to rebuild automatically 🡪 npm I -d nodemon

## Step 3: create the basic syntax

Create the port const

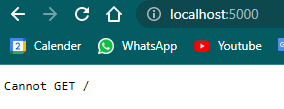
const PORT = process.env.PORT || 5000

app.listen() fuction take the port number and an arrow function as parameter

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

then the server is working

testing it: localhost:PORT



This bet / said that he can’t find a route handler and that take us to the next step

## Step 4:

Enter the route functions

### Get all values

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

*res*.json(members)

})

### Get single value

*//get single*

app.get('/members/:id' , (*req* , *res*) =>{

*//some is a method return bollean that check if there is similar data or no*

    const found = members.some(*member* => *member*.id === parseInt(*req*.params.id));

    if(found){

*res*.json(members.filter(*member*=> *member*.id === parseInt(*req*.params.id)))

    }else{

*res*.status(400).json({msg : `there are not a member with id ${*req*.params.id}`})

    }

})

### Post value

Express.js

*//body parser middleware*

*//express.urlencoded() Returns middleware that only parses urlencoded bodies and only looks at requests where the Content-Type header matches the type option.*

*//This parser accepts only UTF-8 encoding of the body and supports automatic inflation of gzip and deflate encodings.*

app.use(express.json());

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

*member.js*

*//post member*

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

    const newMember = {

        id : counter,

        name : *req*.body.name,

        email : *req*.body.email

    }

    if(!newMember.email || !newMember.name) *res*.status(400).json({ msg:'enter the complete information'}).end();

    members.push(newMember);

*res*.send('member added successfuly')

})

### Update member

*//update member*

router.put('/:id' , (*req* , *res*) =>{

*//some is a method return bollean that check if there is similar data or no*

    const found = members.some(*member* => *member*.id === parseInt(*req*.params.id));

    if(!found) *res*.status(400).json({msg : `there are not a member with id ${*req*.params.id}`}).end();

    const updMember = *req*.body;

    members.forEach(*member* =>{

        if(*member*.id === parseInt(*req*.params.id)){

*member*.name = updMember.name ? updMember.name : *member*.name

*res*.json({ msg : "updated successfully" , members} )

        }

    })

});