

the Master Course

{C0DENATION}

Backend Development

Introduction to Express



Learning Objectives

To know what a Route is.

To understand the basics of Client/Server requests and responses

To know what a database is and what it does.

To be able to make an Express static server.

ROUTES

Imagine visiting a bookshop...

1. INTERACT

I ask the shopkeeper for a copy of Lord of the Rings.

(I interact with a web page e.g. button click)



ROUTES

2. Make A Request

The shopkeeper goes to the stockroom.

(A request is made by the client to the server)



ROUTES

3. Request reaches the Server

The Shopkeeper asks Stockroom person to get a copy of LotR.

(the client request reaches the server)



ROUTES

4. Route is Followed

Stockroom person walks to the shelf where LotR is kept.

(an API route is followed to a controller, where a database operation is performed)



ROUTES

5. Database Response

Stockroom person returns with LotR and gives it to the Shopkeeper.

(the DB responds to the API with the correct data)



ROUTES

6. Client Receives Data

Shopkeeper passes LotR to you.

(the API responds to the client with the requested data and the data is displayed for the user)

ROUTES

The bookshop is an analogy for how the **internet** works, and how we interact with **websites/web applications**.

ROUTES

What is a User?

A user is a person interacting with a website or web app.

ROUTES

What is a Client?

It is any internet connected device that **requests** access to a service provided by a **server**.

What is a Server?

It is a computer system that provides resources or **data to clients** over a network.

ROUTES

What is a Route?

The path that data travels on a network.



Database

What is a Database?

An organised collection of related data that is stored in such a way that it can be easily accessed, retrieved, updated or deleted.



Database

A database is accessed with what are called **CRUD** operations:

Create

Read

Uppdate

Delete

What is Express.js?

As an **NPM Library**, Express gives us a variety of methods to build a **web server** in Node.



Express.js

Express.js

Express is a back-end web application framework for building APIs with Node.js.

We can create a web server, set it to listen for requests, create routes, perform CRUD operations on a database and return responses.



Express.js

Express.js

It is written in **Node** (the backend version of JavaScript) and it allows an app to transfer data over over **HTTP Requests**.

Express.js

Express allows us to listen on certain **paths** and deal with a **HTTP request** before sending a **HTTP response**.



Express.js

Installing Express

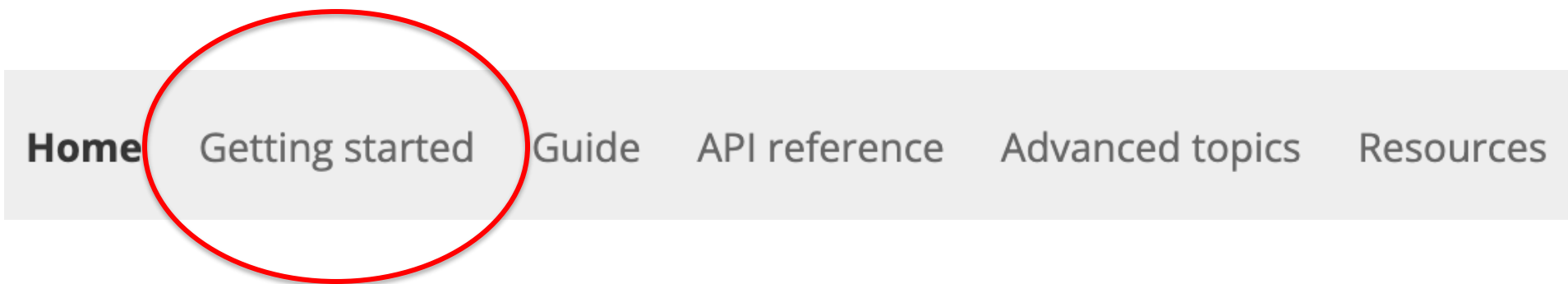
Visit the central hub for node packages:

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

... and search for 'express'.

Let's take a look at the documentation...

<https://expressjs.com>



express()

Methods

express.json()

express.raw()

express.Router()

express.static()

express.text()

express.urlencoded()

Application

Request

Response

Router

What about the reference section...?

Getting started

Guide

API reference

Advanced topics

Resources

Express.js

express()

Application

Request

Properties

req.app

req.baseUrl

req.body

req.cookies

req.fresh

req.hostname

req.ip

req.ips

req.method

Request is the best!

Req.params

Req.route

Req.body

express()

Application

Request

Response

Properties

res.app

res.headersSent

res.locals

Methods

res.append()

res.attachment()

res.cookie()

res.clearCookie()

Response renaissance!

Res.get

Res.json

Res.send



Express.js



Let's install Express.js!

Create a new 'Week7' folder.

In Terminal:

```
npm init -y
```

```
{ } package.json ?
```



Express.js

express()

Application

Request

Response

Router

Methods

router.all()

router.METHOD()

router.param()

router.route()

router.use()

Router for the future!

Router.all

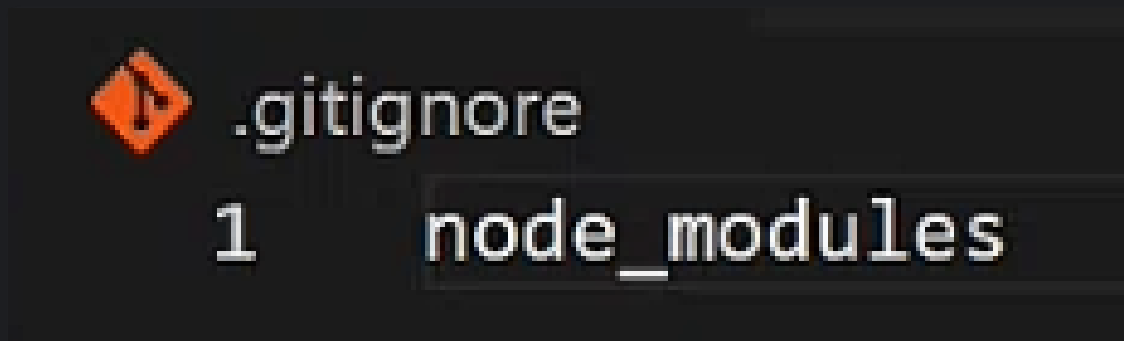
Router.param

Router.use



Express.js

- Create a .gitignore file.
- Add node_modules to it.



```
.gitignore
1  node_modules
```





Express.js

In Terminal:

```
npm install express
```





Express.js

Check the dependencies...

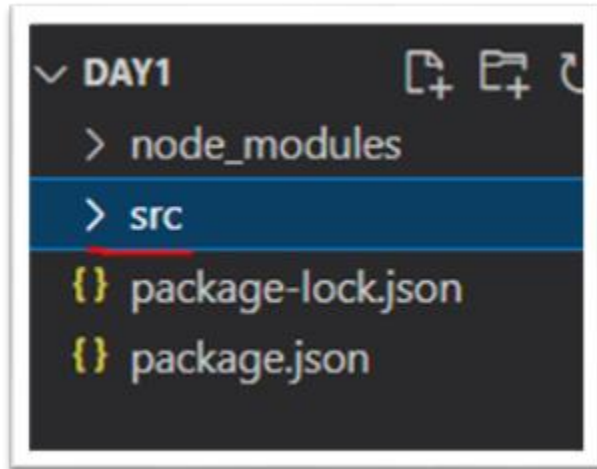
```
},  
"keywords": [],  
"author": "",  
"license": "ISC",  
"dependencies": {  
  "express": "^4.18.2"  
}  
}
```

Congratulations!



Express.js

Ready to create our first Express Server?

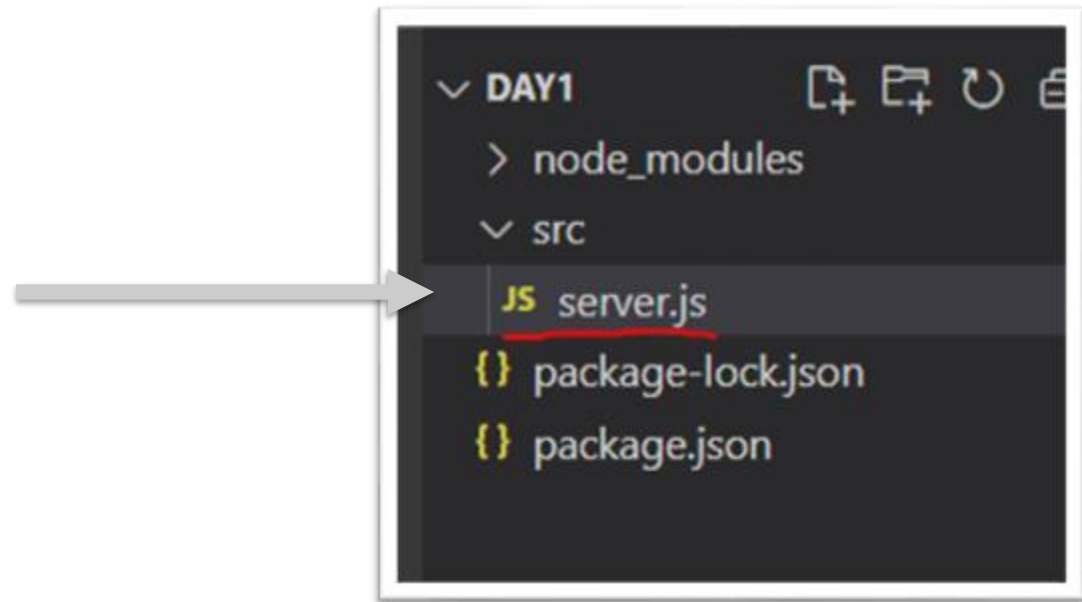


Remember the **src** folder in React?
Let's create one in our folder.

Express.js

Our first Express Server

In the **src** folder...
add a new file: **server.js**



Express.js

Our first Express Server

We need to import the express library into our project



```
JS server.js •
Day1 > src > JS server.js > ...
1  const express = require('express');
2  const app = express();
```

Now to setup some Routes ...

```
src > JS server.js > ...  
1  const express = require('express');  
2  const app = express();  
3  
4  app.use('/aboutme', express.static('about'));
```

Declare the route.

Add the method for creating static webpages, e.g., **about**

Tell Express to listen on Ports

A port is like an extra extension to an address.

Each server has thousands of potential ports and there are standards on what those ports are. Listeners will ignore traffic that doesn't go through this port.

Tell Express to listen on **Ports**

A server can be doing **multiple** things at any time, e.g., acting as a web server on port 80 whilst acting as a SQL Database on port 336.



It allows one type of information to come in on a particular port.

<https://www.hostpapa.com/knowledgebase/commonly-used-ports/>

Express Ports

We need to use the **correct port** when setting up a server so that we are not conflicting with other ports used for different things.

```
src > JS server.js > ...  
1  const express = require('express');  
2  const app = express();  
3  
4  app.use('/aboutme', express.static('about'));  
5  
6  app.listen(5001, () => console.log('Server is running on Port 5001'));
```



Let's tell the server to listen on **port 5001** and to add a helpful **console log** (to make sure that it's working).

Express.js

Express **Ports** working?

In Terminal:

```
node src/server.js
```



Express.js

All working?

```
Server is running on Port 5001
```

Good job!





Express.js

We now have a **working server** – but we don't have anything to serve because the **books directory** is empty!

It is expecting some HTML - so, let's give it some...

Let's give our server something to serve...

Add an **about** folder.

Add a new **index.html** file, inside.

```

  ✓ DAY1
    ✓ about
      <> index.html
    > node_modules
    ✓ src
      JS server.js
      .gitignore
      {} package-lock.json
      {} package.json
  about > <> index.html > html > body > h1
    1  <!DOCTYPE html>
    2  <html lang="en">
    3  <head>
    4    <meta charset="UTF-8">
    5    <meta name="viewport" content="width=device-width, initial-scale=1">
    6    <title>About Me</title>
    7  </head>
    8  <body>
    9    <h1>Welcome to my About Me page</h1>
   10  </body>
   11  </html>
```

Don't forget to use the html **boilerplate**. Add a little **heading**, too.

A working web server?

Open a fresh web browser and we will access the **about** webpage.

Enter this into the URL:

<http://localhost:5001/aboutme/>



Learning Objectives

To know what a Route is.

To understand the basics of Client/Server requests and responses

To know what a database is and what it does.

To be able to make an Express static server.

Activity

Create **multiple** webpages in your server, about yourself with a simple navigation system for the pages. (favourites, hobbies, etc)

Stretch

Research the common **HTTP response** codes.