Pluralsight Node Js

Contents

1. What is REST
2. Getting Data
3. Posting Data
4. Updating Data
5. Testing
6. HATEOAS

what is REST

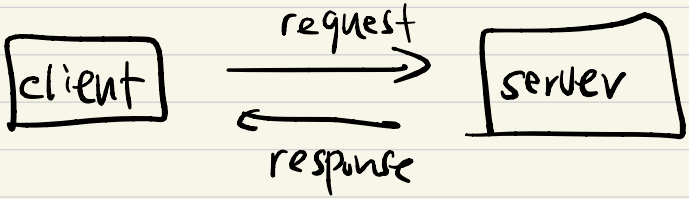
simple API →

goal: build RESTful API with Node.Js + Express

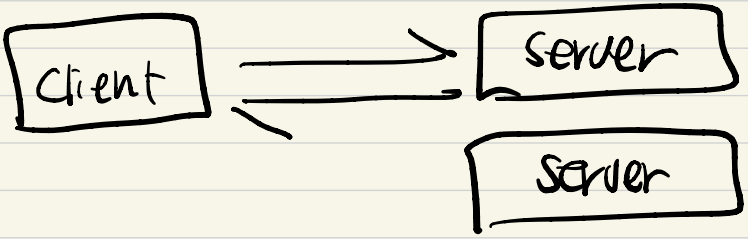
- REST Verbs.
- MongoDB.

① REST

Series of rules of your server



stateless server



Cache ?

3 constraints

cache.

Uniform Interface.

① Resources → (n) book, author
/book /author

② Http verb

Get, Post, Delete

Put/patch ⇒ update

whole — only update part

HATEOAS → request → hyperlink

package.json:

all the info about project
dependencies all listed.

npm install xx --save
can skipped

express @ 4.16.4
for specific version

<starting express server>

```
Var express = require('express') //import
var app = express() // use

Var port = process.env.PORT || 3000
```

alternative

tool for pass in this port to our app

function

```
app.get('/', (req, res) => {
    res.send("Hello World")
})
```

every time get request at /

response

request + response

```
app.listen(port, () => {
    console.log("blah")
})
```

listening at port.

Dev Dependency?

tools.

① Node도 Java가 필요한가?

① npm i eslint -D

② Script area → "lint": "eslint"

③ npm run lint -- --init

② ES Lint

linting.

popular styleguide

---

① npm install nodemon → script part "start": "nodemon app.js"

Airbnb styleguide.

★ eslint → format.

Node JS + Express

REST, resource based. (noun)
HTTP (verb)

+.

nodemon config

+

★ npm start

(2) Getting Data from API

Implement the HTTP GET verb

- get a list of item
  a item or search
- mongoDB + mongoose.
- search by ID, name?

Build a RESTful API with Node.js and Express

- Implement the HTTP GET verb
- Get a list of items or just one
- Wire up to MongoDB
- Search for items

① Router

capsulate to router

```
const bookRouter = express.Router();
const port = process.env.PORT || 3000;

bookRouter.route('/books')
  .get((req, res) => {
    const response = { hello: 'This is my API' };
    res.json(response);
});
app.use('/api', bookRouter)
```

→ books route

Json

→ wiring

you can do
. post
. delete
:

(req, res) => {
const response = { hello: 'xy' }
}

res. send
render

MongoDB.

① Connect to mongo DB

② inject Data.

- model/bookModel.JS and export

Model

Schema

{Schema}

```
const mongoose = require("mongoose");

let app = express();
//connect to mongo db
const db = mongoose.connect("mongodb://localhost/bookAPI");
// rtourer
let bookRouter = express.Router();
// create port   but we dont have so go to 3000?
var port = process.env.PORT || 3000;
const Book = require("./models/bookModel");

bookRouter.route("/books").get((req, res) => {
  Book.find((err, books) => {
    if (err) {
      return res.send(err);
    }
    return res.json(books);
  });
});

app.use("/bookshelf", bookRouter);
```

from model

-> error function

Find One item.

mongoose built in find.

① const {query} = req;

```
bookRouter.route("/books").get((req,
res) => {
  const { query } = req;
  Book.find(query, (err, books) => {
    if (err) {
      return res.send(err);
    }
    return res.json(books);
  });
});
```

localhost:4000/bookshelf/books?genre=Fantasy

What ever you type in the URL split into a javaScript object passing into mongoDB

→ If you type wrong one ⇒ you got [ ]

```
const query = {};
if (req.query.genre) {
  query.genre = req.query.genre;
}
```

if it has genre → empty object

then we that

→ this will ignore the wrong get.

Getting single item

take the id
and only take the item

* params?

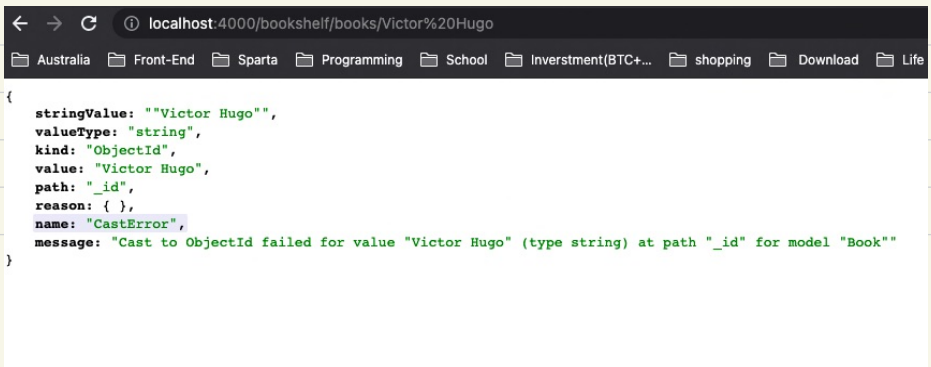
* how to find by Author

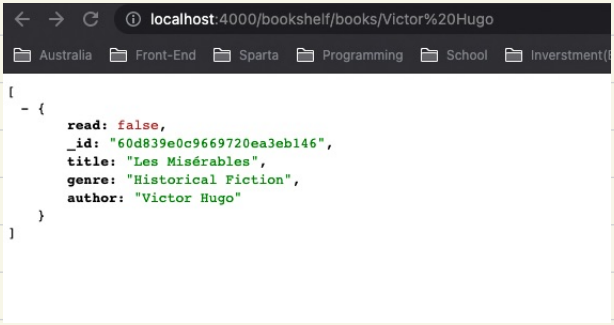
```
bookRouter.route("/books/:bookId").get((req, res) =>
{
  Book.findById(req.params.bookId, (err, book) => {
    if (err) {
      return res.send(err);
    }
    return res.json(book);
  });
});
```

id 는 찾으면 바로 나오는데?

:/ author 로 하면 다른게 나옴

```
bookRouter.route("/books/:author").get((req, res) => {
  Book.findByAuthor(req.params.author, (err, book) => {
    if (err) {
      return res.send(err);
    }
    return res.json(book);
  });
});
```






```
bookRouter.route("/books/:author").get((req, res) => {
  let author = req.params;
  Book.find(author, (err, book) => {
    if (err) {
      return res.send(err);
    }
    return res.json(book);
  });
});
```

원래 /books/:id 를 comment out 해두고

findById 는 mongoose inbuild function

find 로 Author를 찾았다

↓

해결완료

1번을

/book/author/:author 로 바꿨다.

/book/:bookid

Posting Data.

Build a RESTful API with Node.js and Express

- Implement the HTTP POST verb
- Add a new item to the list
- Test with Postman

allow us to add new data in dB

help us post / verb

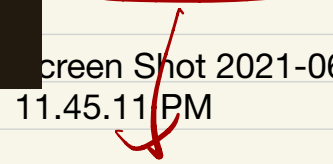
t

post

```
bookRouter
  .route("/books")
  .post((req, res) => {
    const book = new Book(req.body);
    book.save();
    res.status(201).json(book);
  })
```

what ever post this route ↓ create new mongoose

req.body = ?



← const bodyParser = require(body/parse)

Install bodyparse

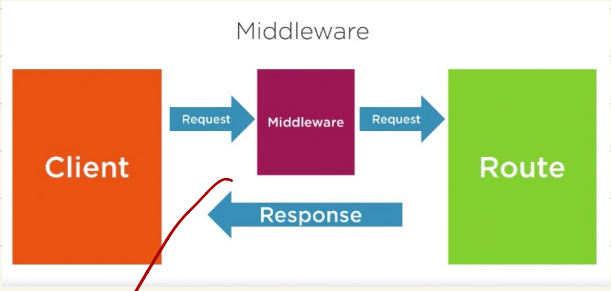
app.use (body parser. urlencoder extend = true)

app.use (bodyParser. Json)

Updating Data.

Build a RESTful API with Node.js and Express (update)

- Implement the HTTP PUT and PATCH verbs
- PUT replaces an item (full replace)
- PATCH only changes a piece (only part)
- Middleware
- Implement Delete.



originally it was in the route repeated so just made as middleware

middleware

Middleware

keep repeated so we make it as middleware

```
bookRouter.use("/books/:bookId", (req, res, next) => {
  Book.findById(req.params.bookId, (err, book) => {
    if (err) {
      return res.send(err);
    }
    if (book) {
      req.book = book;
      return next();
    }
    return res.send(404);
  });
});
```

sendStatus

since this is from middleware

```
bookRouter
  .route("/books/:bookId")
  .get((req, res) => res.json(req.book))
  .put((req, res) => {
    const { book } = req;
    book.title = req.body.title;
    book.author = req.body.author;
    book.genre = req.body.genre;
    book.read = req.body.read;
    req.book.save((err) => {
      if (err) {
        return res.send(err);
      }
    });
  })
```

any

.Put ((req, res) => {

Book.findById (req.Paraqms.bookId, (err, book) =>

{ if (err){

return res.send(err)

}

return res.send(404)

})

book.title = req.body.title

book.author

genre

read

book.save()

patch

```
.patch((req, res) => {
  const { book } = req;
  // eslint-disable-next-line no-underscore-dangle
  if (req.body._id) {
    // eslint-disable-next-line no-underscore-dangle
    delete req.body._id;
  }
  Object.entries(req.body).forEach((item) => {
    const key = item[0];
    const value = item[1];
    book[key] = value;
  });
  req.book.save((err) => {
    if (err) {
      return res.send(err);
    }
    return res.json(book);
  });
})
```

→ We don't want to change id, If id => delete

delete

```
.delete((req, res) => {
  req.book.remove((err) => {
    if (err) {
      return res.send(err);
    }
    return res.sendStatus(204);
  });
});
```