

NodeJS REST APIs

Introduction



Different Kind of Response is Needed

Representational State Transfer

Transfer Data instead of User Interfaces

Important: Only the response (and the request data) changes, NOT the general server-side logic!



Data Formats

HTML	Plain Text	XML	JSON
<code><p>Node.js</p></code>	<code>Node.js</code>	<code><name>Node.js</name></code>	<code>{"title": "Node.js"}</code>
Data + Structure	Data	Data	Data
Contains User Interface	No UI Assumptions	No UI Assumptions	No UI Assumptions
Unnecessarily difficult to parse if you just need the data	Unnecessarily difficult to parse, no clear data structure	Machine-readable but relatively verbose; XML-parser needed	Machine-readable and concise; Can easily be converted to JavaScript



Http Methods (Http Verbs)

More than just GET & POST

GET	POST	PUT
Get a Resource from the Server	Post a Resource to the Server (i.e. create or append Resource)	Put a Resource onto the Server (i.e. create or overwrite a Resource)
PATCH	DELETE	OPTIONS
Update parts of an existing Resource on the Server	Delete a Resource on the Server	Determine whether follow-up Request is allowed (sent automatically)



REST Principles

Uniform Interface	Stateless Interactions		
Clearly defined API endpoints with clearly defined request + response data structure	Server and client don't store any connection history, every request is handled separately		
Cacheable	Client-Server	Layered System	Code on Demand
Servers may set caching headers to allow the client to cache responses	Server and client are separated, client is not concerned with persistent data storage	Server may forward requests to other APIs	Executable code may be transferred from server to client

Create a basic REST API

Step 1. Create a new folder

Step 2. Autogenerate npm package

npm init

Step 3. Install Express framework

npm install --save express

Step 4. Create “app.js” file in the root folder

```
const express = require('express');
const app = express();
app.listen(8080);
```

Step 5. Automatically restart the app on code changes

Install nodemon package:

npm install --save-dev nodemon

In the package.json file inside the “scripts” object add the “start” line:

```
"start": "nodemon app.js"
```

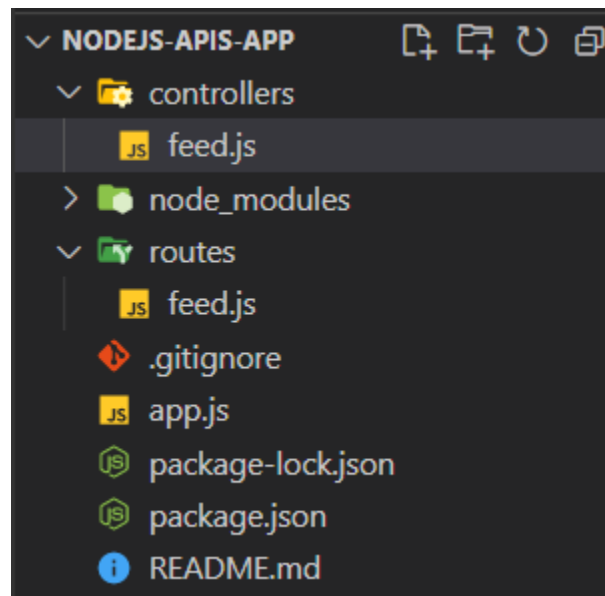
Step 6. Run npm package with all commands specified there

npm start

To stop the command press CTRL + C

Now, for REST APIs we need to create two folders:

- “**controllers**” folder with “feed.js” file
- “**routes**” folder with “feed.js” file



Step 7. Create a new “**controllers**” folder and add “**feed.js**” file here:

```
// implementation of GET /feed/posts-kali
exports.getPostsKali = (req, res, next) => {
  const response = {
```

```

    posts: [
      {
        title: 'List of posts',
        content: 'Here you go - the list of posts for you!'
      }
    ]
  };

  // HTTP 200 OK success status response code indicates that the request has
succeeded
  res.status(200).json(response);
};

// implementation of POST /feed/post-kali
exports.createPostKali = (req, res, next) => {
  const title = req.body.title;
  const content = req.body.content;
  const response = {
    message: 'Your post has been created successfully!',
    post: {
      id: new Date().toISOString(),
      title: title,
      content: content
    }
  };
};

// HTTP 201 CREATED success status response code indicates that the request
has succeeded and has led to the creation of a resource (e..g in db)
  res.status(201).json(response);
};

```

Step 8. Create a new “**routes**” folder and add “**feed.js**” file here:

```

const express = require('express');
const feedController = require('../controllers/feed');
const router = express.Router();

// GET /feed/posts-kali
router.get('/posts-kali', feedController.getPostsKali);

// POST /feed/post-kali
router.post('/post-kali', feedController.createPostKali);

```

```
module.exports = router;
```

Step 9. Install body-parser

It helps to parse incoming request bodies in a middleware before your handlers (available under the req.body property)

npm install --save body-parser

Step 10. Add the code to "app.js" file

```
const express = require('express');
const bodyParser = require('body-parser');
const feedRoutes = require('./routes/feed');
const app = express();

// app.use(bodyParser.urlencoded()); // x-www-form-urlencoded <form>
app.use(bodyParser.json()); // application/json

// configure CORS
app.use((req, res, next) => {
  res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Methods', 'OPTIONS, GET, POST, PUT,
PATCH, DELETE');
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type,
Authorization');
  next();
});

app.use('/feed', feedRoutes);

app.listen(8080);
```

Step 11. Run the app

npm start

Open Postman

a) Call GET <http://localhost:8080/feed/posts-kali> API:

GET GET /posts-kali X POST POST /post-kali + ... No Environment

▶ GET /posts-kali Examples (0)

GET http://localhost:8080/feed/posts-kali Send Save

Params Authorization Headers Body Pre-request Script Tests Cookies Code Comments (0)

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (10) Test Results Status: 200 OK Time: 43 ms Size: 486 B Save Download

Pretty Raw Preview JSON

```

1 {
2   "posts": [
3     {
4       "title": "List of posts",
5       "content": "Here you go - the list of posts for you!"
6     }
7   ]
8 }

```

b) Call POST <http://localhost:8080/feed/post-kali> method

GET GET /posts-kali POST POST /post-kali X + ... No Environment

▶ POST /post-kali Examples (0)

POST http://localhost:8080/feed/post-kali Send Save

Params Authorization Headers (1) Body Pre-request Script Tests Cookies Code Comments (0)

none form-data x-www-form-urlencoded raw binary JSON (application/json) Beautify

```

1 {
2   "title": "My new post",
3   "content": "Run Forrest run!"
4 }

```

Body Cookies Headers (10) Test Results Status: 201 Created Time: 8 ms Size: 548 B Save Download

Pretty Raw Preview JSON

```

1 {
2   "message": "Your post has been created successfully!",
3   "post": {
4     "id": "2022-07-27T00:52:40.048Z",
5     "title": "My new post",
6     "content": "Run Forrest run!"
7   }
8 }

```