json-server DT

0.17.3 • Public • Published 4 months ago

Readme

Code Beta

20 Dependencies

314 Dependents

127 Versions

JSON Server Node.js CI passing

Get a full fake REST API with **zero coding** in **less than 30 seconds** (seriously)

Created with <3 for front-end developers who need a quick back-end for prototyping and mocking.

- Egghead.io free video tutorial - Creating demo APIs with json-server
- JSONPlaceholder - Live running version
- **My JSON Server** - no installation required, use your own data

See also:

- husky - Git hooks made easy
- lowdb - local JSON database
- ✅ xv - a beautifully simple and capable test runner

Gold sponsors

Silver sponsors

Become a sponsor and have your company logo here

Sponsor

Please help me build OSS 👉 GitHub Sponsors ❤️

Table of contents

Getting started

Install JSON Server

```
npm install -g json-server
```

Create a `db.json` file with some data

```
{
  "posts": [
    { "id": 1, "title": "json-server", "author": "typicode" }
  ],
  "comments": [
    { "id": 1, "body": "some comment", "postId": 1 }
  ],
  "profile": { "name": "typicode" }
}
```

Start JSON Server

```
json-server --watch db.json
```

Now if you go to **http://localhost:3000/posts/1**, you'll get

```
{ "id": 1, "title": "json-server", "author": "typicode" }
```

Also when doing requests, it's good to know that:

- If you make POST, PUT, PATCH or DELETE requests, changes will be automatically and safely saved to `db.json` using lowdb.
- Your request body JSON should be object enclosed, just like the GET output. (for example `{"name": "Foobar"}`)
- Id values are not mutable. Any `id` value in the body of your PUT or PATCH request will be ignored. Only a value set in a POST request will be respected, but only if not already taken.
- A POST, PUT or PATCH request should include a `Content-Type: application/json` header to use the JSON in the request body. Otherwise it will return a 2XX status code, but without changes being made to the data.

Routes

Based on the previous `db.json` file, here are all the default routes. You can also add **other routes** using `--routes`.

Plural routes

```
GET    /posts
GET    /posts/1
POST   /posts
PUT    /posts/1
PATCH  /posts/1
DELETE /posts/1
```

Singular routes

```
GET    /profile
POST   /profile
PUT    /profile
PATCH  /profile
```

Filter

Use `.` to access deep properties

```
GET /posts?title=json-server&author=typicode
GET /posts?id=1&id=2
GET /comments?author.name=typicode
```

Paginate

Use `_page` and optionally `_limit` to paginate returned data.

In the `Link` header you'll get `first`, `prev`, `next` and `last` links.

```
GET /posts?_page=7
GET /posts?_page=7&_limit=20
```

10 items are returned by default

Sort

Add `_sort` and `_order` (ascending order by default)

```
GET /posts?_sort=views&_order=asc
GET /posts/1/comments?_sort=votes&_order=asc
```

For multiple fields, use the following format:

```
GET /posts?_sort=user,views&_order=desc,asc
```

Slice

Add `_start` and `_end` or `_limit` (an `X-Total-Count` header is included in the response)

```
GET /posts?_start=20&_end=30
GET /posts/1/comments?_start=20&_end=30
GET /posts/1/comments?_start=20&_limit=10
```

*Works exactly as **Array.slice** (i.e. `_start` is inclusive and `_end` exclusive)*

Operators

Add `_gte` or `_lte` for getting a range

```
GET /posts?views_gte=10&views_lte=20
```

Add `_ne` to exclude a value

```
GET /posts?id_ne=1
```

Add `_like` to filter (RegExp supported)

```
GET /posts?title_like=server
```

Full-text search

Add `q`

```
GET /posts?q=internet
```

Relationships

To include children resources, add `_embed`

```
GET /posts?_embed=comments
GET /posts/1?_embed=comments
```

To include parent resource, add `_expand`

```
GET /comments?_expand=post
GET /comments/1?_expand=post
```

To get or create nested resources (by default one level, **add custom routes** for more)

```
GET  /posts/1/comments
POST /posts/1/comments
```

Database

```
GET /db
```

Homepage

Returns default index file or serves `./public` directory

```
GET /
```

Extras

Static file server

You can use JSON Server to serve your HTML, JS and CSS, simply create a `./public` directory or use `--static` to set a different static files directory.

```
mkdir public
echo 'hello world' > public/index.html
json-server db.json
```

```
json-server db.json --static ./some-other-dir
```

Alternative port

You can start JSON Server on other ports with the `--port` flag:

```
$ json-server --watch db.json --port 3004
```

Access from anywhere

You can access your fake API from anywhere using CORS and JSONP.

Remote schema

You can load remote schemas.

```
$ json-server http://example.com/file.json
$ json-server http://jsonplaceholder.typicode.com/db
```

Generate random data

Using JS instead of a JSON file, you can create data programmatically.

```
// index.js
module.exports = () => {
  const data = { users: [] }
  // Create 1000 users
  for (let i = 0; i < 1000; i++) {
    data.users.push({ id: i, name: `user${i}` })
  }
  return data
}
```

```
$ json-server index.js
```

Tip use modules like **Faker**, **Casual**, **Chance** or **JSON Schema Faker**.

HTTPS

There are many ways to set up SSL in development. One simple way is to use **hotel**.

Add custom routes

Create a `routes.json` file. Pay attention to start every route with `/`.

```
{
  "/api/*": "/$1",
  "/:resource/:id/show": "/:resource/:id",
  "/posts/:category": "/posts?category=:category",
  "/articles\\?id=:id": "/posts/:id"
}
```

Start JSON Server with `--routes` option.

```
json-server db.json --routes routes.json
```

Now you can access resources using additional routes.

```
/api/posts # → /posts
/api/posts/1  # → /posts/1
/posts/1/show # → /posts/1
/posts/javascript # → /posts?category=javascript
/articles?id=1 # → /posts/1
```

Add middlewares

You can add your middlewares from the CLI using `--middlewares` option:

```
// hello.js
module.exports = (req, res, next) => {
  res.header('X-Hello', 'World')
  next()
}
```

```
json-server db.json --middlewares ./hello.js
json-server db.json --middlewares ./first.js ./second.js
```

CLI usage

```
json-server [options] <source>

Options:
  --config, -c       Path to config file           [default: "json-serve
  --port, -p         Set port                                 [defaul
  --host, -H         Set host                       [default: "loc
  --watch, -w        Watch file(s)                             [
  --routes, -r       Path to routes file
  --middlewares, -m  Paths to middleware files
  --static, -s       Set static files directory
  --read-only, --ro  Allow only GET requests                   [
  --no-cors, --nc    Disable Cross-Origin Resource Sharing     [
  --no-gzip, --ng    Disable GZIP Content-Encoding             [
  --snapshots, -S    Set snapshots directory                [defau
  --delay, -d        Add delay to responses (ms)
  --id, -i           Set database id property (e.g. _id)    [defaul
  --foreignKeySuffix, --fks  Set foreign key suffix, (e.g. _id as in pos
                                                            [defaul
  --quiet, -q        Suppress log messages from output         [
  --help, -h         Show help                                 [
  --version, -v      Show version number                       [

Examples:
  json-server db.json
  json-server file.js
  json-server http://example.com/db.json
```

```
https://github.com/typicode/json-server
```

You can also set options in a `json-server.json` configuration file.

```
{
  "port": 3000
}
```

Module

If you need to add authentication, validation, or **any behavior**, you can use the project as a module in combination with other Express middlewares.

Simple example

```
$ npm install json-server --save-dev


// server.js
const jsonServer = require('json-server')
const server = jsonServer.create()
const router = jsonServer.router('db.json')
const middlewares = jsonServer.defaults()

server.use(middlewares)
server.use(router)
server.listen(3000, () => {
  console.log('JSON Server is running')
})


$ node server.js
```

The path you provide to the `jsonServer.router` function is relative to the directory from where you launch your node process. If you run the above code from another directory, it's better to use an absolute path:

```
const path = require('path')
const router = jsonServer.router(path.join(__dirname, 'db.json'))
```

For an in-memory database, simply pass an object to `jsonServer.router()`.

To add custom options (eg. `foreginKeySuffix`) pass in an object as the second argument to `jsonServer.router('db.json', { foreginKeySuffix: '_id' })`.

Please note also that `jsonServer.router()` can be used in existing Express projects.

Custom routes example

Let's say you want a route that echoes query parameters and another one that set a timestamp on every resource created.

```
const jsonServer = require('json-server')
const server = jsonServer.create()
const router = jsonServer.router('db.json')
const middlewares = jsonServer.defaults()

// Set default middlewares (logger, static, cors and no-cache)
server.use(middlewares)

// Add custom routes before JSON Server router
server.get('/echo', (req, res) => {
  res.jsonp(req.query)
})

// To handle POST, PUT and PATCH you need to use a body-parser
// You can use the one used by JSON Server
server.use(jsonServer.bodyParser)
```

```
server.use((req, res, next) => {
  if (req.method === 'POST') {
    req.body.createdAt = Date.now()
  }
  // Continue to JSON Server router
  next()
})

// Use default router
server.use(router)
server.listen(3000, () => {
  console.log('JSON Server is running')
})
```

Access control example

```
const jsonServer = require('json-server')
const server = jsonServer.create()
const router = jsonServer.router('db.json')
const middlewares = jsonServer.defaults()

server.use(middlewares)
server.use((req, res, next) => {
 if (isAuthorized(req)) { // add your authorization logic here
   next() // continue to JSON Server router
 } else {
   res.sendStatus(401)
 }
})
server.use(router)
server.listen(3000, () => {
  console.log('JSON Server is running')
})
```

Custom output example

To modify responses, overwrite `router.render` method:

```
// In this example, returned resources will be wrapped in a body prope
router.render = (req, res) => {
  res.jsonp({
    body: res.locals.data
  })
}
```

You can set your own status code for the response:

```
// In this example we simulate a server side error response
router.render = (req, res) => {
  res.status(500).jsonp({
    error: "error message here"
  })
}
```

Rewriter example

To add rewrite rules, use `jsonServer.rewriter()`:

```
// Add this before server.use(router)
server.use(jsonServer.rewriter({
  '/api/*': '/$1',
  '/blog/:resource/:id/show': '/:resource/:id'
}))
```

Mounting JSON Server on another endpoint example

Alternatively, you can also mount the router on `/api`.

```
server.use('/api', router)
```

API

`jsonServer.create()`

Returns an Express server.

`jsonServer.defaults([options])`

Returns middlewares used by JSON Server.

- options
  - `static` path to static files
  - `logger` enable logger middleware (default: true)
  - `bodyParser` enable body-parser middleware (default: true)
  - `noCors` disable CORS (default: false)
  - `readOnly` accept only GET requests (default: false)

`jsonServer.router([path|object], [options])`

Returns JSON Server router.

- options (see CLI usage)

Deployment

You can deploy JSON Server. For example, **JSONPlaceholder** is an online fake API powered by JSON Server and running on Heroku.

Links

Video

- Creating Demo APIs with json-server on egghead.io

Articles

- Node Module Of The Week - json-server
- ng-admin: Add an AngularJS admin GUI to any RESTful API

- Fast prototyping using Restangular and Json-server
- Create a Mock REST API in Seconds for Prototyping your Frontend
- No API? No Problem! Rapid Development via Mock APIs
- Zero Code REST With json-server

Third-party tools

- Grunt JSON Server
- Docker JSON Server
- JSON Server GUI
- JSON file generator
- JSON Server extension

License

MIT

Supporters ✨

Keywords

JSON server fake REST API prototyping mock mocking test testing rest data dummy sandbox

Install

```
> npm i json-server
```

Repository

github.com/typicode/json-server

Homepage

github.com/typicode/json-server

Weekly Downloads

252,825

Version

0.17.3

License

MIT

Unpacked Size

58.5 kB

Total Files

26

Issues

576

Pull Requests

89

Last publish

4 months ago

Collaborators

>_Try on RunKit

Report malware

Support

Help

Advisories

Status

Contact npm

Company

About

Blog

Press

Terms & Policies

Policies

Terms of Use

Code of Conduct

Privacy