

Introduction to Web Development

Part 3: Demo NodeJs Rest Api

arnaud.nauwynck@gmail.com

Course Esilv 2023

This document:

<https://github.com/Arnaud-Nauwynck/presentations/web/angular-demos-3-nodejs-rest-api.pdf>

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl

Test using Postman

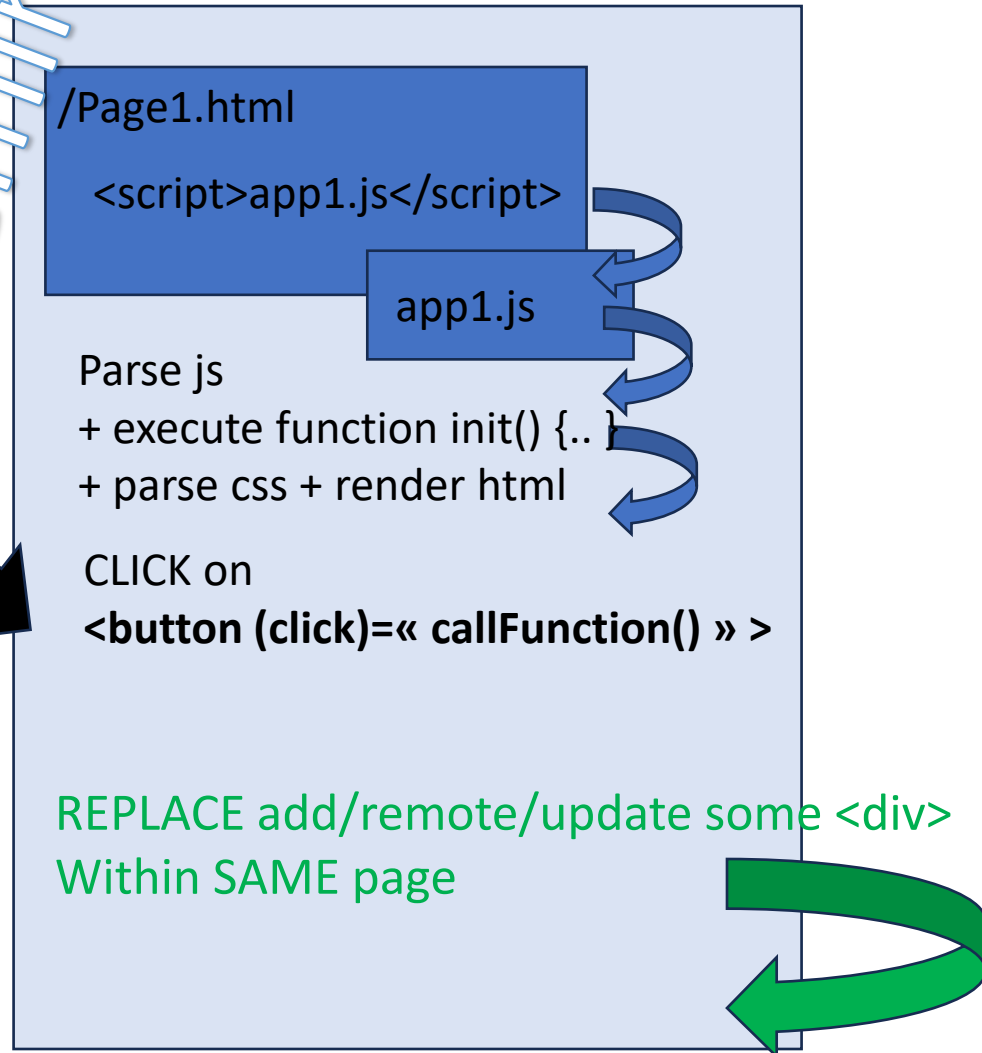
OpenApi – Swagger UI

SPA = Single Page Application

single GET {html|js}, Multiple Rest Json requests

Reminder

Web Browser



http GET /page1.html

http GET /page1.js

http GET /page1.css

http GET /page1-image2.png

Web Server

http POST /api/restEndpoint1 {json}

response {json}

http POST /api/restEndpoint2 {json}

response {json}

Rest using NodeJs - Express

```
const express = require('express');  
const app = express();
```

```
app.get('/', function (req, res) {  
  res.send({ some: 'Hello Json Express' })  
});
```

```
app.listen(3000);
```

Reminder

~ Rest using Springboot: @RequestMapping

http PUT /api/endpoint1/meth2
header1:value1

```
{  
  « reqField »: « value »  
}
```



http 200 OK
header2:value2

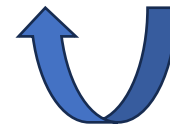
```
{  
  « respField »: « value »  
}
```

```
class RequestDTO {  
  public String reqField;  
}
```

Json -> java



Java -> json



```
class ResponseDTO {  
  public String respField;  
}
```

@RestController

@RequestMapping(« /api/endpoint1 »)
public class MyRestController {

@PutMapping(« meth2 »)

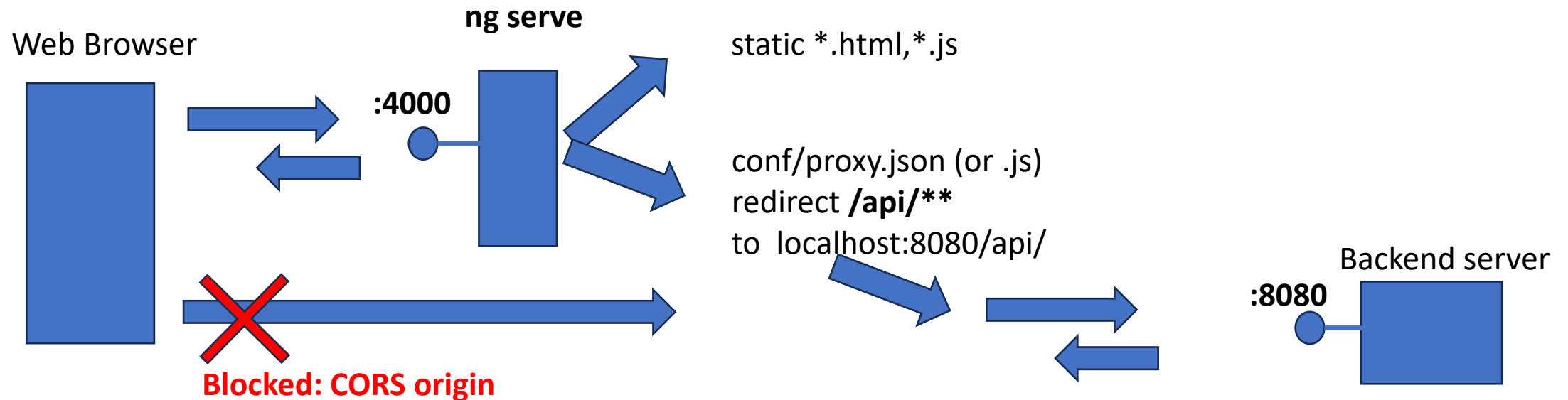
public ResponseDTO handle(
 @RequestBody RequestDTO req) {
 return new ...
}
}

Ng serve... why Proxy to /api/** ? => CORS Origin problem !

Because ...

Web Browser blocks calls to « http://localhost:**8080**/api/** »

not coming from same « origin domain » as « http://localhost:**4000**/index.html » «app.js » !



Demo Outline



Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

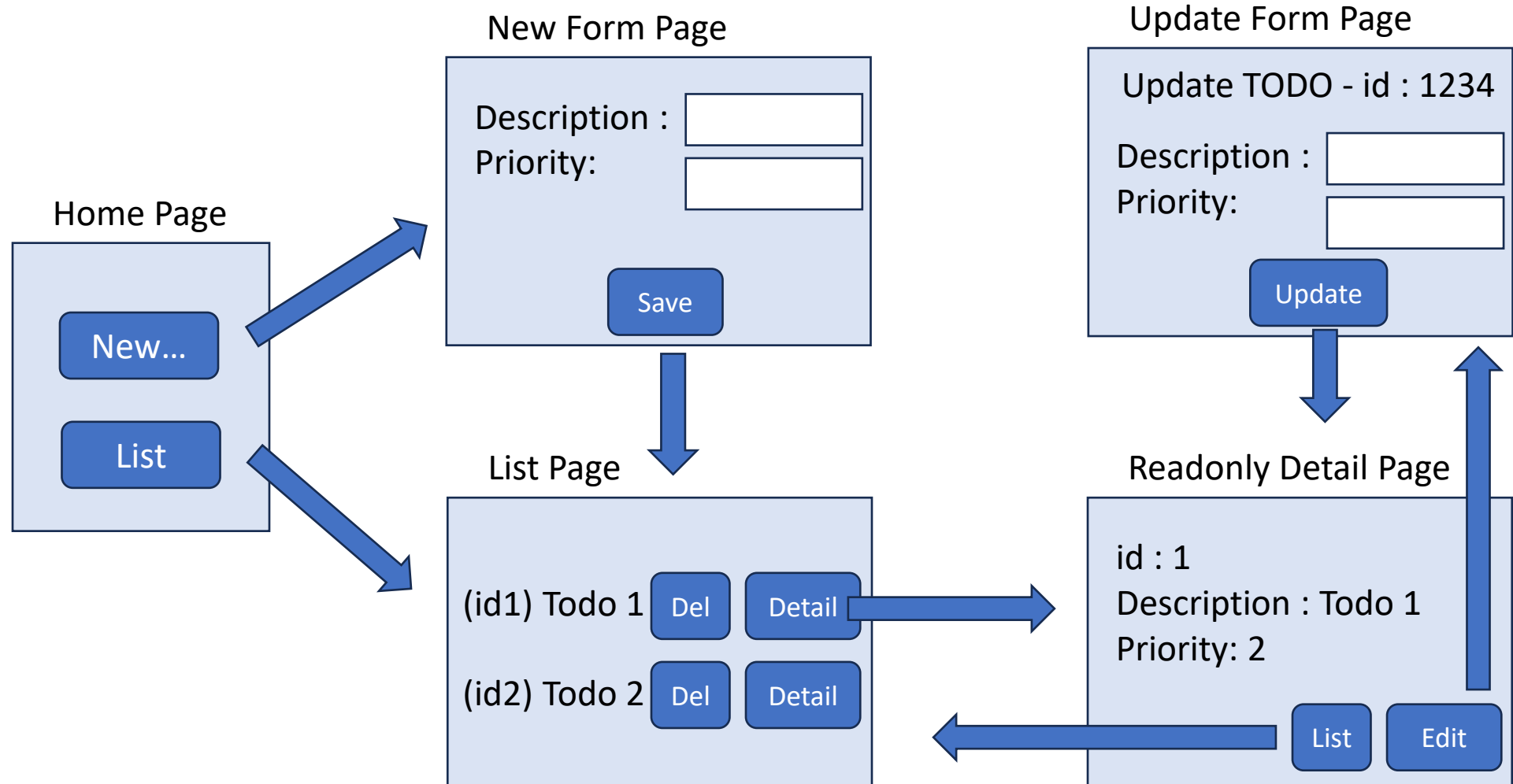
- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl

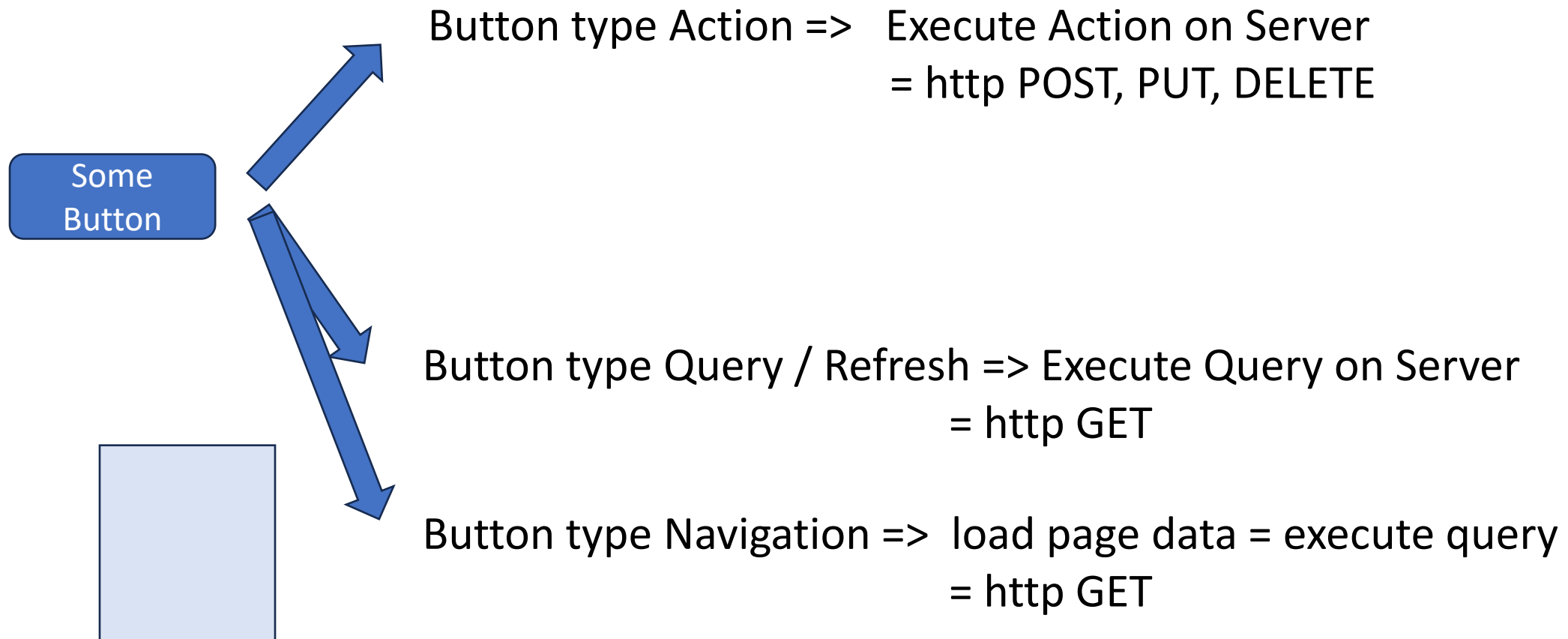
Test using Postman

OpenApi – Swagger UI

Web Site Sketch -> Discover API requirement



App Skeleton Buttons : Action / Query - Navigation



Todo Web App API Description

... CRUD

- C **Create** ➡ Page 1: Simple Html Form with input fields
Click on « Save » button => save « Todo » object on server
- R **Read All** ➡ Page 2: Show list of « Todos »
- Read By Id** ➡ page 3 for url « /todo/{id} »
Show detailed « Todo » page
- U **Update** ➡ Page 4: Editable page + « update » button
- D **Delete** ➡ « delete » button

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »



Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl

Test using Postman

OpenApi – Swagger UI

\$ npm init

```
$ npm init
```

```
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.
```

```
See `npm help init` for definitive documentation on these fields  
and exactly what they do.
```

```
Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.
```

```
Press ^C at any time to quit.
```

```
package name: (demo-node-express-todo-rest)
```

```
version: (1.0.0)
```

```
description: rest server for Todos, using express
```

```
entry point: (index.js)
```

```
test command:
```

```
git repository:
```

```
keywords:
```

```
author:
```

```
license: (ISC)
```

```
About to write to C:\arn\perso\cours\esilv\web\demo-node-express-todo-rest\package.json:
```

```
{  
  "name": "demo-node-express-todo-rest",  
  "version": "1.0.0",  
  "description": "rest server for Todos, using express",  
  "main": "index.js",  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "author": "",  
  "license": "ISC"  
}
```

\$ npm install -s express

```
$ npm install -s express
```

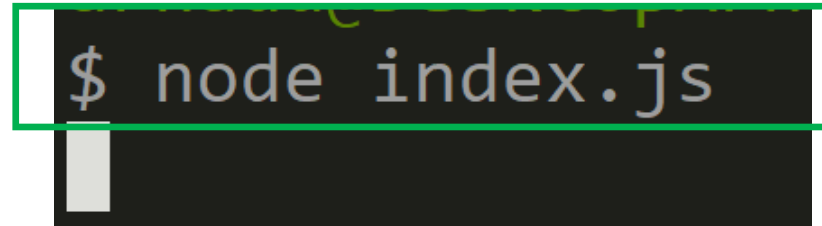
```
$ cat package.json
{
  "name": "demo-node-express-todo-rest",
  "version": "1.0.0",
  "description": "rest server for Todos, using express",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.18.2"
  }
}
```

Hello express endpoint (port 3000)

A code editor window with a tab labeled 'index.js' and a close button. The code is written in JavaScript and uses syntax highlighting: 'express' is green, 'function' is blue, and '3000' is blue. Line numbers 1 through 8 are visible on the left side of the editor.

```
1 const express = require('express');
2 const app = express();
3
4 app.get('/', function (req, res) {
5   res.send({ some: 'Hello Json Express' })
6 });
7
8 app.listen(3000);
```

Running: \$ node index.js

A screenshot of a terminal window with a black background. The text "\$ node index.js" is displayed in a light gray monospaced font. A green rectangular border highlights the text. A white cursor is positioned at the end of the command. Below the command, there is a small gray rectangular block, likely representing a scroll bar or a placeholder for output.

```
$ node index.js
```

Test using Curl

```
$ curl -v http://localhost:3000
```

```
* Trying 127.0.0.1:3000...
```

```
* Connected to localhost (127.0.0.1) port 3000 (#0)
```

```
> GET / HTTP/1.1
```

```
> Host: localhost:3000
```

```
> User-Agent: curl/8.0.1
```

```
> Accept: */*
```

```
>
```

```
< HTTP/1.1 200 OK
```

```
< X-Powered-By: Express
```

```
< Content-Type: application/json; charset=utf-8
```

```
< Content-Length: 29
```

```
< ETag: W/"1d-qhxeNQT6M+LyU0MKyv0e0/zWvbE"
```

```
< Date: Mon, 31 Jul 2023 11:42:47 GMT
```

```
< Connection: keep-alive
```

```
< Keep-Alive: timeout=5
```

```
<
```

```
{"some":"Hello Json Express"}* Connection #0 to host localhost left intact
```


Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »



Test using Curl

Test using Postman

OpenApi – Swagger UI

Rest endpoint : GET « /todo »

```
8  var todos = [  
9      { id: 1, description: "learn Angular", priority: 1},  
10     { id: 2, description: "learn Typescript", priority: 1},  
11     { id: 3, description: "learn http", priority: 1},  
12 ];
```

```
23  app.get('/todo', function (req, res) {  
24      res.send(todos)  
25  });
```

GET "/api/todos" with OpenApi (swagger) doc

With OpenAPI

```
83  ✓ /**
84      * @openapi
85      * /api/todos:
86      *   get:
87      *     description: Get all todos
88      *     responses:
89      *       200:
90      *         description: An array of Todo
91      *         schema:
92      *           type: array
93      *           items:
94      *             $ref: '#/components/schemas/Todo'
95      */
96  ✓ app.get('/api/todos', (req: Request, res: Response) : void => {
97      console.log('handle http GET /api/todos');
98      res.send(todos);
99  });
```

WHY is this JsDoc necessary ?

OpenApi (see next) is generally mandatory in modern IT departments

OpenApi when using NodeJS-Express is unable
to extract Type Information

from generic method `.get((req, resp) => callback)`

More Doc Comment than (TypeScript)Code ! but Comment are untyped/unchecked

Comment are unchecked

NO Tool to validate the syntax / structure of comments

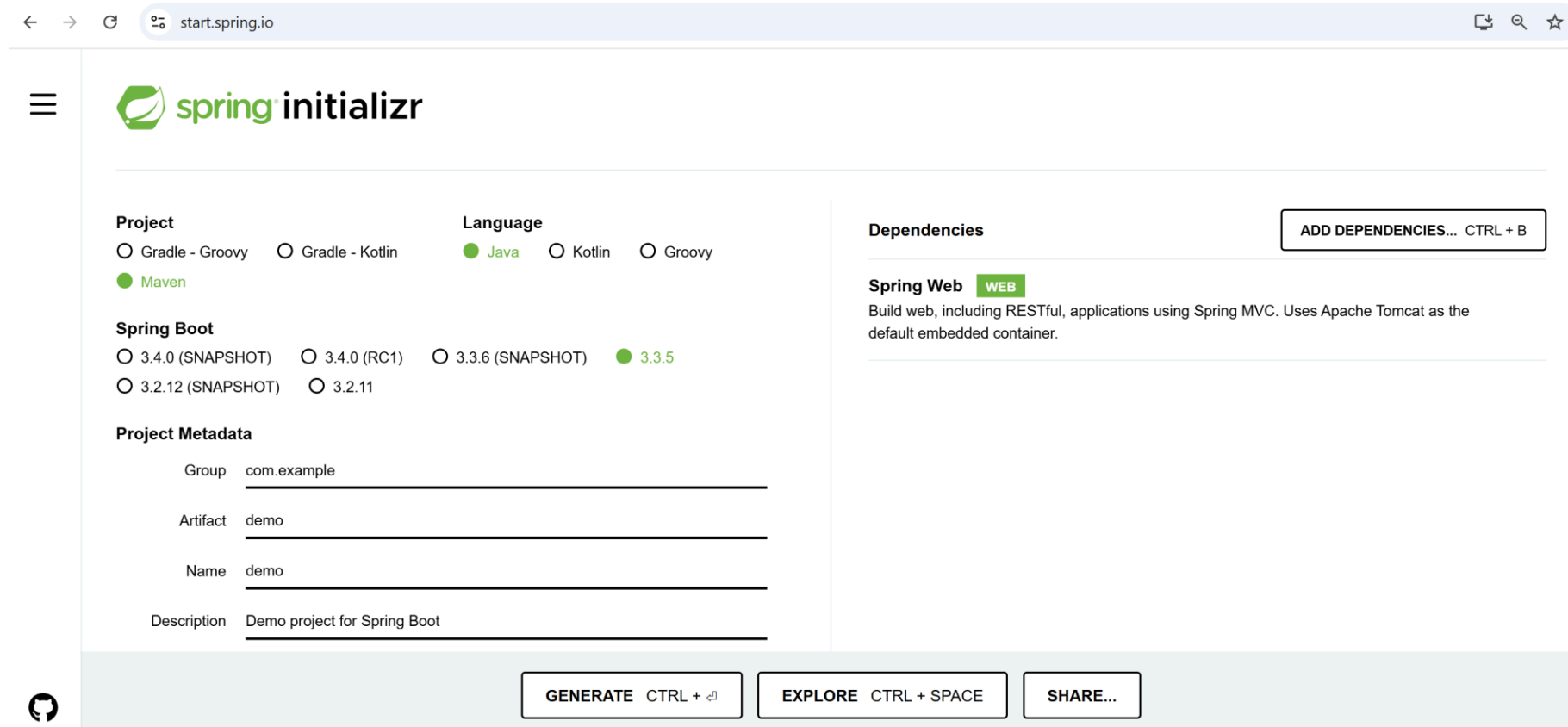
Comment is NOT in-sync with code,
"Json" types in the comment may not reflect the real "type" in TypeScript code

... Difficult to maintain

Spoiler Alert:

NodeJS Rest API are Dead,
Long Life Java SpringBoot Rest API
or Java Quarkus, or C#, ..

Creating a Java Springboot Rest API



The screenshot shows the Spring Initializr web application in a browser. The browser's address bar displays 'start.spring.io'. The page features the Spring logo and the text 'spring initializr'. It is divided into several sections for configuring a new project:

- Project:** Includes radio buttons for 'Gradle - Groovy', 'Gradle - Kotlin', 'Maven' (selected), and 'Language' options: 'Java' (selected), 'Kotlin', and 'Groovy'.
- Spring Boot:** Includes radio buttons for versions: '3.4.0 (SNAPSHOT)', '3.4.0 (RC1)', '3.3.6 (SNAPSHOT)', '3.3.5' (selected), and '3.2.12 (SNAPSHOT)', '3.2.11'.
- Project Metadata:** Includes input fields for 'Group' (com.example), 'Artifact' (demo), 'Name' (demo), and 'Description' (Demo project for Spring Boot).
- Dependencies:** Includes a button 'ADD DEPENDENCIES... CTRL + B' and a section for 'Spring Web' with a 'WEB' tag and a description: 'Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.'

At the bottom, there are three buttons: 'GENERATE CTRL + G', 'EXPLORE CTRL + SPACE', and 'SHARE...'. A GitHub logo is visible in the bottom left corner.

```
@RestController("/todo") class ..  
@GetMapping("/") method() {.. }
```

© TodoRestController.java ×

```
5  
6 import org.springframework.web.bind.annotation.GetMapping;  
7 import org.springframework.web.bind.annotation.RequestMapping;  
8 import org.springframework.web.bind.annotation.RestController;  
9  
10 @RestController  
11 @RequestMapping(path = "/todo")  
12 public class TodoRestController {
```

```
23 app.get('/todo', function (req, res) {  
24   res.send(todos)  
25 });
```

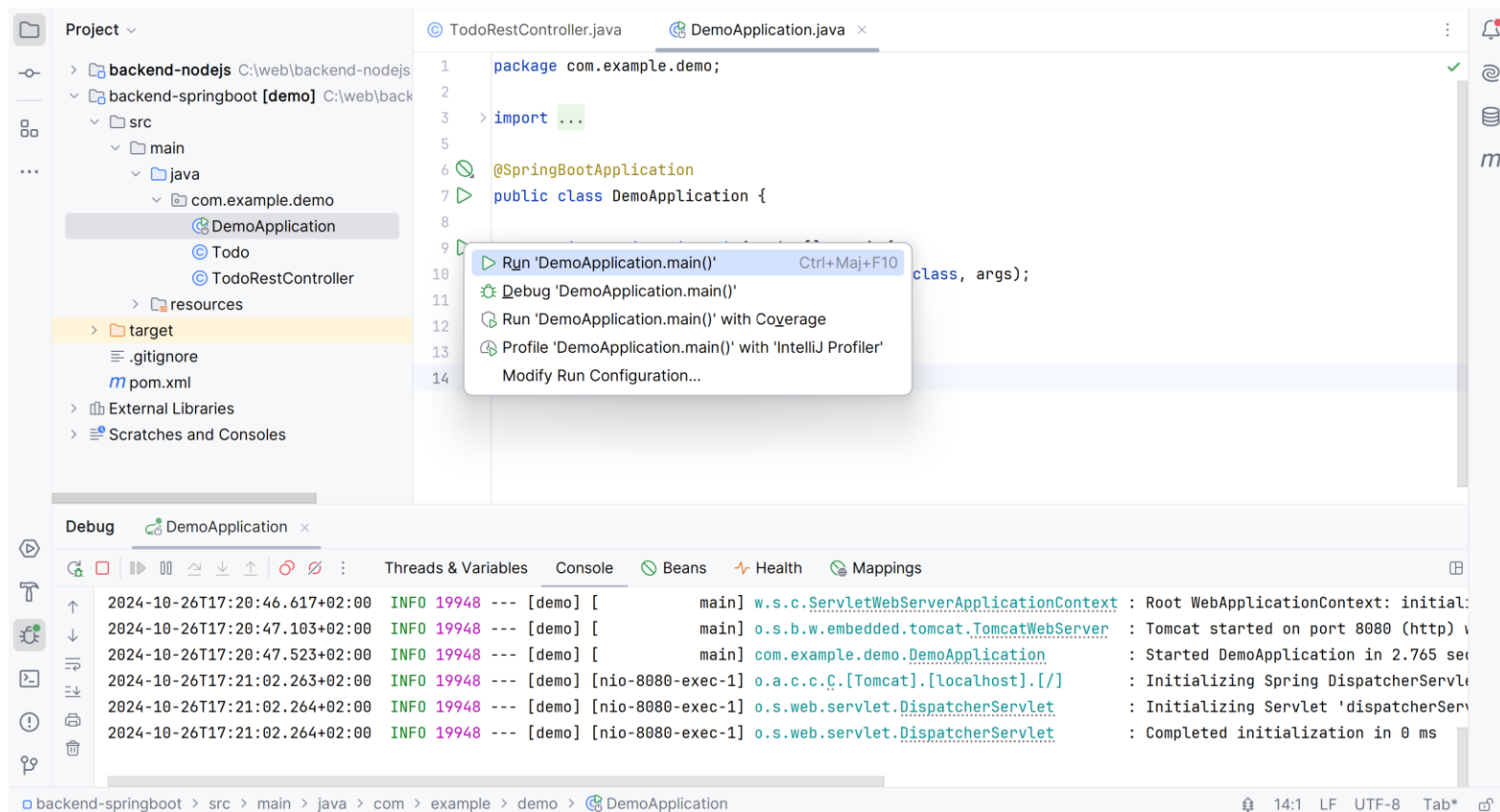


```
24 @GetMapping  
25 public List<Todo> getAll() {  
26   return todos;  
27 }
```

in NodeJS-Express
but WITHOUT OpenAPI types

in Java SpringBoot
and WITH types

SpringBoot : RUN Application.main()



← → ↻ ⓘ localhost:8080/todo

Pretty-print ☐

```
[{"id":1,"title":"Learn Java","description":"","priority":1}, {"id":2,"title":"Learn SpringBoot","description":"","priority":2}, {"id":3,"title":"Learn Rest Spring-Web","description":"","priority":3}]
```

Test GET /todo using curl

```
$ curl http://localhost:3000/todo
```

```
[{"id":1,"description":"learn Angular","priority":1},{"id":2,"description":"learn Typescript","priority":1},{"id":3,"description":"learn http","priority":1}]
```

```
$ curl -v http://localhost:3000/todo
```

```
* Trying 127.0.0.1:3000...
```

```
* Connected to localhost (127.0.0.1) port 3000 (#0)
```

```
> GET /todo HTTP/1.1
```

```
> Host: localhost:3000
```

```
> User-Agent: curl/8.0.1
```

```
> Accept: */*
```

```
>
```

```
< HTTP/1.1 200 OK
```

```
< X-Powered-By: Express
```

```
< Content-Type: application/json; charset=utf-8
```

```
< Content-Length: 157
```

```
< ETag: W/"9d-UoeCn1N4gua0Zh3Qv1vJpsfmzPY"
```

```
< Date: Mon, 31 Jul 2023 12:04:09 GMT
```

```
< Connection: keep-alive
```

```
< Keep-Alive: timeout=5
```

```
<
```

```
[{"id":1,"description":"learn Angular","priority":1},{"id":2,"description":"learn Typescript","priority":1},{"id":3,"description":"learn http","priority":1}]* Connection #0 to host localhost left intact
```

Test GET /todo, using curl --silent + jq ‘

```
$ curl --silent http://localhost:3000/todo | jq '.'
```

```
[
  {
    "id": 1,
    "description": "learn Angular",
    "priority": 1
  },
  {
    "id": 2,
    "description": "learn Typescript",
    "priority": 1
  },
  {
    "id": 3,
    "description": "learn http",
    "priority": 1
  }
]
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :



- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl

Test using Postman

OpenApi – Swagger UI

Rest endpoint POST /todo

```
14 var idGenerator = 4;  
15 function generateNewId() {  
16     return idGenerator++;  
17 }
```

```
43 app.post('/todo', function (req, res) {  
44     console.log('handle http POST /todo, request.body:', req.body);  
45     const newId = generateNewId();  
46     const todo = {id: newId, description: req.body.description, priority: req.body.priority };  
47     todos.push(todo);  
48     res.send(todo);  
49 });
```

use Express Json

ts app.ts ×

```
1 import * as express from 'express';
2 import { Request, Response } from 'express';
3
4 const app : Express = express();
5
6 // use feature to parse request body in Json
7 // with http header "content-type": "application/json"
8 app.use(express.json());
```

(older Deprecaded way)

js index.js ×

```
1 const express = require('express');
2 const bodyParser = require('body-parser')
3
4 var app = express()
5
6 // parse application/json
7 app.use(bodyParser.json())
8
```

Test using curl ...

-H « Content-Type: application/json »
-H « accept: application/json »
-d '{« description»: « ..» }'

```
$ curl -X POST http://localhost:3000/todo -H "content-type:application/json" -H "accept: application/json" -d '{"description":"learn express", "priority":1}'  
{"id":4,"description":"learn express","priority":1}
```

console.log() in nodejs

```
43 app.post('/todo', function (req, res) {  
44   console.log('handle http POST /todo, request.body:', req.body);  
45 }
```

```
$ node index.js  
handle http POST /todo, request.body: { description: 'learn express', priority: 1 }  
█
```


Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »



Test using Curl

Test using Postman

OpenApi – Swagger UI

Rest endpoint GET « /todo/:id »

```
63 app.get('/todo/:id', function (req, res) {  
64     const id = +req.params.id;  
65     console.log('handle http GET /todo/:id, ', id);  
66     const idx = todos.findIndex(x => x.id === id);  
67     if (idx !== -1) {  
68         res.send(todos[idx]);  
69     } else {  
70         res.status(404).send('Todo entity not found for id:' + id);  
71     }  
72 });
```

Notice... type coercion, exact === vs ==

```
const id = +req.param.id; // <= coercion string to number  
// equivalent to:
```

```
const idText = req.param.id; // idText is string
```

```
const id = +idText; // id is number
```

```
(id === 2) // correct, exact equality for number
```

```
(id === '2') // wrong... types differ
```

```
(id == '2') or (id == 2); // both ok, using implicit type coercions
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »



Test using Curl

Test using Postman

OpenApi – Swagger UI

Rest endpoint PUT « /todo » req.body={ id:..., ..}

```
74 app.put('/todo', function (req, res) {  
75     const reqBody = req.body;  
76     console.log('handle http PUT /todo, request.body:', reqBody);  
77     const id = reqBody.id;  
78     const idx = todos.findIndex(x => x.id === id);  
79     if (idx !== -1) {  
80         const todo = todos[idx];  
81         todo.description = reqBody.description;  
82         todo.priority = reqBody.priority;  
83         res.send(todo);  
84     } else {  
85         res.status(404).send('Todo entity not found for id:' + id);  
86     }  
87 });
```

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »



Test using Curl

Test using Postman

OpenApi – Swagger UI

Endpoint DELETE « /todo/:id »

```
51 app.delete('/todo/:id', function (req, res) {  
52     const id = +req.params.id;  
53     console.log('handle http DELETE /todo/:id, ', id);  
54     const idx = todos.findIndex(x => x.id === id);  
55     if (idx !== -1) {  
56         const removed = todos.splice(idx, 1);  
57         res.send(removed);  
58     } else {  
59         res.status(404).send('Todo entity not found for id:' + id);  
60     }  
61 });
```

Test Endpoint DELETE « /todo/:id »

```
$ curl -X DELETE http://localhost:3000/todo/2  
[{"id":2,"description":"learn Typescript","priority":1}]
```

```
$ node index.js  
handle http DELETE /todo/:id, 2
```

```
$ curl -X DELETE http://localhost:3000/todo/999999  
Todo entity not found for id:999999
```


Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »



Test using Curl

Test using Postman

OpenApi – Swagger UI

\$ curl ... c:> ?? PS1> Invoke-WebRequest

Curl on Power-Shell = alias for Invoke-WebRequest

... different command line arguments / result formatting !

```
Windows PowerShell
PS C:\Users\arnaud> curl

applet de commande Invoke-WebRequest à la position 1 du pipeline de la commande
Fournissez des valeurs pour les paramètres suivants :
Uri: http://localhost:3000/todo

StatusCode      : 200
StatusDescription : OK
Content         : [{"id":1,"description":"learn Angular","priority":1},{"id":2,"description":"learn
Typescript","priority":1},{"id":3,"description":"learn http","priority":1}]
RawContent      : HTTP/1.1 200 OK
                  Connection: keep-alive
                  Keep-Alive: timeout=5
                  Content-Length: 157
                  Content-Type: application/json; charset=utf-8
                  Date: Tue, 01 Aug 2023 10:11:58 GMT
                  ETag: W/"9d-UoeCn1N4gua0Zh3Qv1v...

Forms           : {}
Headers         : {[Connection, keep-alive], [Keep-Alive, timeout=5], [Content-Length, 157], [Content-Type,
                  application/json; charset=utf-8]...}
Images          : {}
InputFields     : {}
Links           : {}
ParsedHtml      : mshtml.HTMLDocumentClass
RawContentLength : 157
```

<https://curl.se/download.html>



curl.se/download.html

[Download](#)[Documentation](#)[libcurl](#)[Get Help](#)[Development](#)[News](#)

[curl](#) / **Download**

Releases and Downloads

The curl project mostly provides source packages. Other packages are kindly provided by external persons and organizations.

Source Archives

curl 8.2.1, Released on the **2023-07-26**. [Changelog for 8.2.1](#).

[curl-8.2.1.tar.gz](#) [gpg](#)

[curl-8.2.1.tar.bz2](#) [gpg](#)

[curl-8.2.1.zip](#) [gpg](#)

[curl-8.2.1.tar.xz](#) [gpg](#)

Related:

[Changelog](#)

[Old Releases](#)

[Source code repo](#)

[Daily Snapshots](#)

[GPG Key](#)

[Release log](#)

\$ curl sheet cheat

-X GET,POST,PUT,DELETE

-v verbose

>

> Sent

<

< Received

Default: GET when no -d
Or POST when -d 'request body'

-d 'request body'

@ : request body from file name

Example: \$ curl -X POST <http://loc> @file.json

@- : request body from stdin

Example: \$ cat << EOF

{ «field »: 123 }

EOF | curl -X POST <http://loc> @-

--silent

-H header:value

Example:

-H Content-Type:application/json

-H Accept:application/json

-u user:password

-x proxy or export HTTPS_PROXY

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl

Test using Postman

OpenApi – Swagger UI



<https://www.postman.com/downloads/>

Home Workspaces ▾ Explore


Search Postman

Sign In Create Account

History

New Import

History



You haven't sent any requests.

Any request you send in this workspace will appear here.

[Show me how](#)

GET Untitled Request

HTTP

Untitled Request

Add to collection

</>

GET ▾

Enter URL or paste text


Send ▾

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

	Key	Value	Bulk Edit
	Key	Value	

Response



Enter the URL and click Send to get a response

Console

Postman http GET /todo

Home Workspaces ▾ Explore Search Postman Sign In Create Account

History New Import GET http://localhost:3000/todo

Today
GET http://localhost:3000/todo

GET http://localhost:3000/todo

Params Auth Headers (6) Body Pre-req. Tests Settings Cookies

Query Params

Key	Value	Bulk Edit
Key	Value	

Body Cookies Headers (7) Test Results 200 OK 17 ms 338 B Save Response ▾

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "description": "learn Angular",
5     "priority": 1
6   },
7   {
8     "id": 3,
9     "description": "learn http",
10    "priority": 1
11  }
12 ]
```

Console

Postman http POST /todo req.body { json }

The screenshot shows the Postman interface for a POST request to `http://localhost:3000/todo`. The request is configured with a JSON body containing the following data:

```
{
  "description": "learn postman",
  "priority": 2
}
```

The response is displayed in the bottom panel, showing a 200 OK status with a response time of 194 ms and a body size of 286 B. The response body is formatted as JSON:

```
{
  "id": 4,
  "description": "learn postman",
  "priority": 2
}
```


Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl

Test using Postman

OpenApi – Swagger UI



Swagger (OpenAPI)

<https://www.npmjs.com/package/swagger-ui-express>

+

<https://www.npmjs.com/package/swagger-jsdoc>

```
$ npm i swagger-ui-express swagger-jsdoc  
[REDACTED] / reify:z-schema: http fetch GET 200 https://registry.npmjs.org/z-schema/-/z-schem
```

Configuring express + swagger ..

```
3  const swaggerJsdoc = require("swagger-jsdoc");  
4  const swaggerUi = require("swagger-ui-express");
```

```
155  const swaggerOptions = {  
156    definition: {  
157      openapi: "3.1.0",  
158      info: { title: "Todo Rest API" },  
159      servers: [ { url: "http://localhost:3000" } ]  
160    },  
161    apis: [ "./index.js" ],  
162  };  
163  const swaggerDoc = swaggerJsdoc(swaggerOptions);  
164  console.log("swagger doc from annotated js:", swaggerDoc);  
165  
166  app.use('/api-docs', swaggerUi.serve);  
167  app.get('/api-docs', swaggerUi.setup(swaggerDoc));
```

Adding Js Doc: /** @openapi .. */

```
31  /**
32   * @openapi
33   * /todo:
34   *   get:
35   *     description: list of todos
36   *     responses:
37   *       200:
38   *         description: ok
39   */
40  app.get('/todo', function (req, res) {
41    res.send(todos)
42  });
```

Console.log swagger spec from js doc

```
const swaggerDoc = swaggerJsdoc(swaggerOptions);  
console.log("swagger doc from annotated js:", swaggerDoc);
```

```
$ node index.js  
swagger doc from annotated js: {  
  openapi: '3.1.0',  
  info: { title: 'Todo Rest API' },  
  servers: [ { url: 'http://localhost:3000' } ],  
  paths: {  
    '/todo': { get: [Object], post: [Object], put: [Object] },  
    '/todo/:id': { get: [Object], delete: [Object] }  
  },  
  components: {},  
  tags: []  
}
```

Swagger-UI : <http://localhost:3000/api-docs/>

localhost:3000/api-docs/



Todo Rest API

Servers

http://localhost:3000

default

GET /todo


POST /todo

PUT /todo

GET /todo/:id

DELETE /todo/:id

Test using Swagger-UI « Try it out » .. « Execute »

 **Swagger**
Supported by SMARTBEAR

Todo Rest API

Servers

http://localhost:3000 ▾

default ^

GET /todo ^

list of todos

Parameters

No parameters

Responses

Code	Description	Links
200		No links

Try it out

@openapi GET « /todo/{id} » parameters for express « /todo/:id »

```
46  /**
47   * @openapi
48   * /todo/{id}:
49   *   get:
50   *     description: todo by id
51   *     parameters:
52   *       - name: id
53   *         in: path
54   *         description: id of the todo entity
55   *         required: true
56   *         type: integer
57   *     produces:
58   *       - application/json
59   *     responses:
60   *       200:
61   *         description: ok
62   *       404:
63   *         description: Todo entity not found
64   */
65  app.get('/todo/:id', function (req, res) {
```


@openapi POST

requestBody .. content .. schema

```
75  /**
76  * @openapi
77  * /todo:
78  *   post:
79  *     description: create a new todo
80  *     requestBody:
81  *       required: true
82  *       content:
83  *         application/json:
84  *           schema:
85  *             type: object
86  *             properties:
87  *               description:
88  *                 type: string
89  *                 description: description of the todo entity
90  *               priority:
91  *                 type: number
92  *                 description: priority of the todo entity
93  *             produces:
94  *               - application/json
95  *             responses:
96  *               200:
97  *                 description: ok
98  */
99  app.post('/todo', function (req, res) {
```

Test Swagger POST with requestBody .. Required json

POST /todo

create a new todo

Parameters

No parameters

Request body required

application/json

```
{
  "description": "learn swagger",
  "priority": 2
}
```

Execute

Demo Outline

Reminder : http client/server, SPA: Single Page Application
Classical Web app tutorial: « Todo web application »

Setup NodeJs + Express

CRUD Rest endpoints :

- POST « /todo » {json}
- GET « /todo »
- GET « /todo/:id »
- PUT « /todo » {json}
- DELETE « /todo/:id »

Test using Curl

Test using Postman

OpenApi – Swagger UI



Next Steps :
Create Angular FrontEnd App

Take Away

Simple to code a http server

.. even in NodeJS

(better in Java – Springboot

IDE / Debugger / Type checking / Libraries / ..

more valuable for work in companies

)

Rest endpoint = http server using url + json

+ conventions for state transfer on GET,POST,PUT,DELETE

Use Curl / Postman / OpenApi Swagger

Questions

Next Steps ...