



Express.js CRUD Post



Create



Read



Update



Delete

POST

- Here we need want to send data from the client to the server

```
//lets add a new landmark
//the url here shall remain the same,
//the only thing that changes is the http method
app.post('/api/v1/landmarks', (req,res)=>{
```

```
  })
```

url is the
same

http method
is that
changes

POST

- Here we need want to send/submit data from the client to the server
- These data **we would like** to be available at the **request**
- We want to **send** data in a payload also known as **request body**
- This **request body** we want to be referenced with the **request object**

POST

- Express **does not put body data** on the request
- **But we want** to extract incoming data of a **POST request**.
- In order to have that data available -> we use something called *middleware*.

```
//lets add a new landmark
//the url here shall remain the same,
//the only thing that changes is the http method
app.post('/api/v1/landmarks', (req, res)=>{
})
```

Before proceed to post, lets see what
Middleware is....

Middleware

- With express in order to **have** that **data** in the **request body** available we need to use something called middleware
 - **Middleware functions** are functions that have access to
 - the request object (req)
 - the response object (res)
 - the **next middleware function**
- in the application's **request-response cycle**

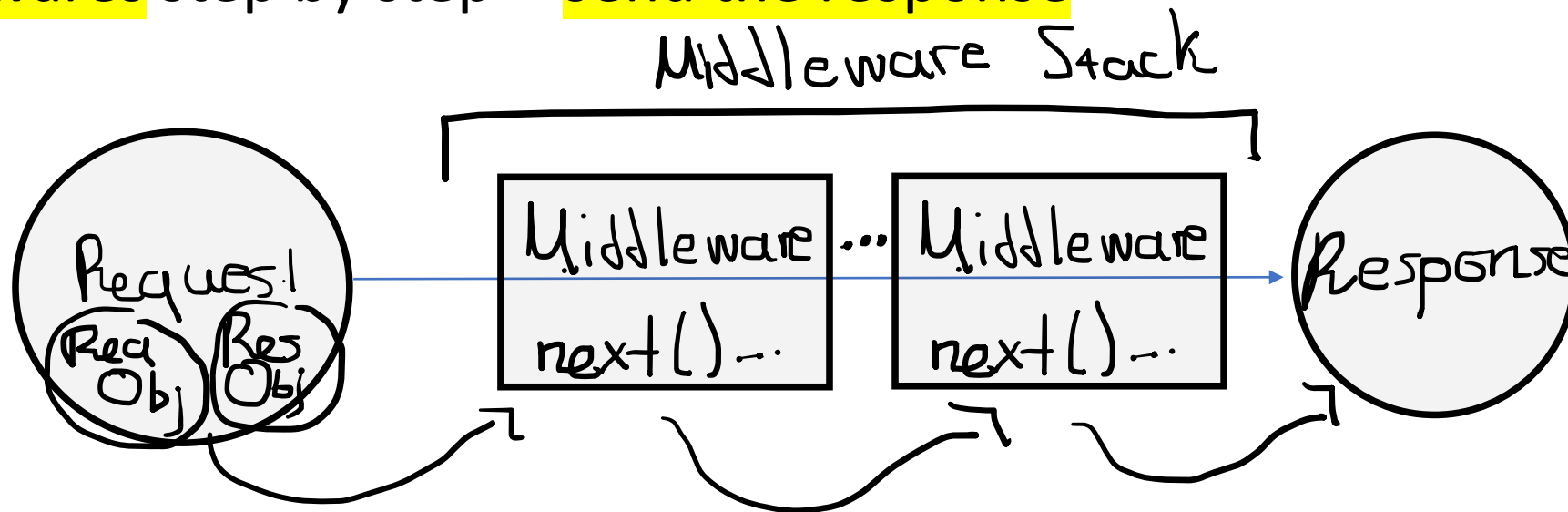
Middleware

- Middleware functions can
 - **Execute** any code.
 - Make **changes** to the **request** and the **response** objects.
 - **End** the request-response cycle.
 - Call the **next** middleware in the stack.

If the current middleware function does not end the request-response cycle, it must call **next()** to pass control to the next middleware function. Otherwise, the request **will be left hanging**.

Middleware

- Middleware-> is a function executed **between** receiving the request and sending the response
- A **step** that the **request** goes through -> while it's being processed.
- **request-response cycle**: starts with **incoming request**-> execute all **middlewares** step by step-> **send the response**



express.json()

- express.json()-> is a **built-in middleware** function in Express

```
// create app variable -> assigned result of calling express function
// add a wide number of methods to app variable
const app= express();

//creating middleware express.json() :it is called middleware because
//it is in the middle of the request and the response
|
app.use(express.json());
```

express.json()

- `express.json()` -> is a **built-in middleware method** in Express
- it **parses incoming requests** with **JSON payloads** and is based on `body-parser`.
 - payload in API -> is the **actual data pack** that is sent
 - It can be sent/ received in various formats JSON included

App.use

- app.use function
 - is used to **mount** the specified middleware function(s) at the **path** which is being **specified**
 - It allows us to create our **own middleware**
- **app.use(path, function)**
- path: It is the path for which the middleware function is being called (we can leave it blank...)
- function: It is a middleware function or a series/array of middleware functions.

Lets create our own middleware

- With **app.use** we are creating our own middleware
- Lets see what we get to the console when we run a get request

Middleware functions can take any number of parameters, but the first three are conventionally named req, res, and next

```
app.use((req,res,next)=>{
  console.log("hi my friend, I am your middleware!")
  next();
});
```

Never

forget next

```
{ id: 1 }
hi my friend, I am your middleware!
█
```

Our middleware applies to all requests because it is added before all request in file

3rd party middleware

- We can also install and use 3rd party middlewares.
- I.e. we may want to use *morgan* -> a Node.js middleware to log HTTP requests and errors

3rd party middleware

Steps:

- `//install morgan`
- `npm i morgan`

- `//import morgan`
- `const morgan = require('morgan');`

- `//use 3rd party middleware morgan`
- `app.use(morgan('dev'));`

More...

- More about middleware you may check below...
- <https://expressjs.com/en/guide/using-middleware.html>

POST

- Here in our example, in order to add something in our json file we are going to use middleware
- `express.json()`-> parses incoming requests with body data

```
// create app variable -> assigned result of calling express function
// add a wide number of methods to app variable
const app= express();

//creating middleware express.json() :it is called middleware because
//it is in the middle of the request and the response
|
app.use(express.json());
```

POST

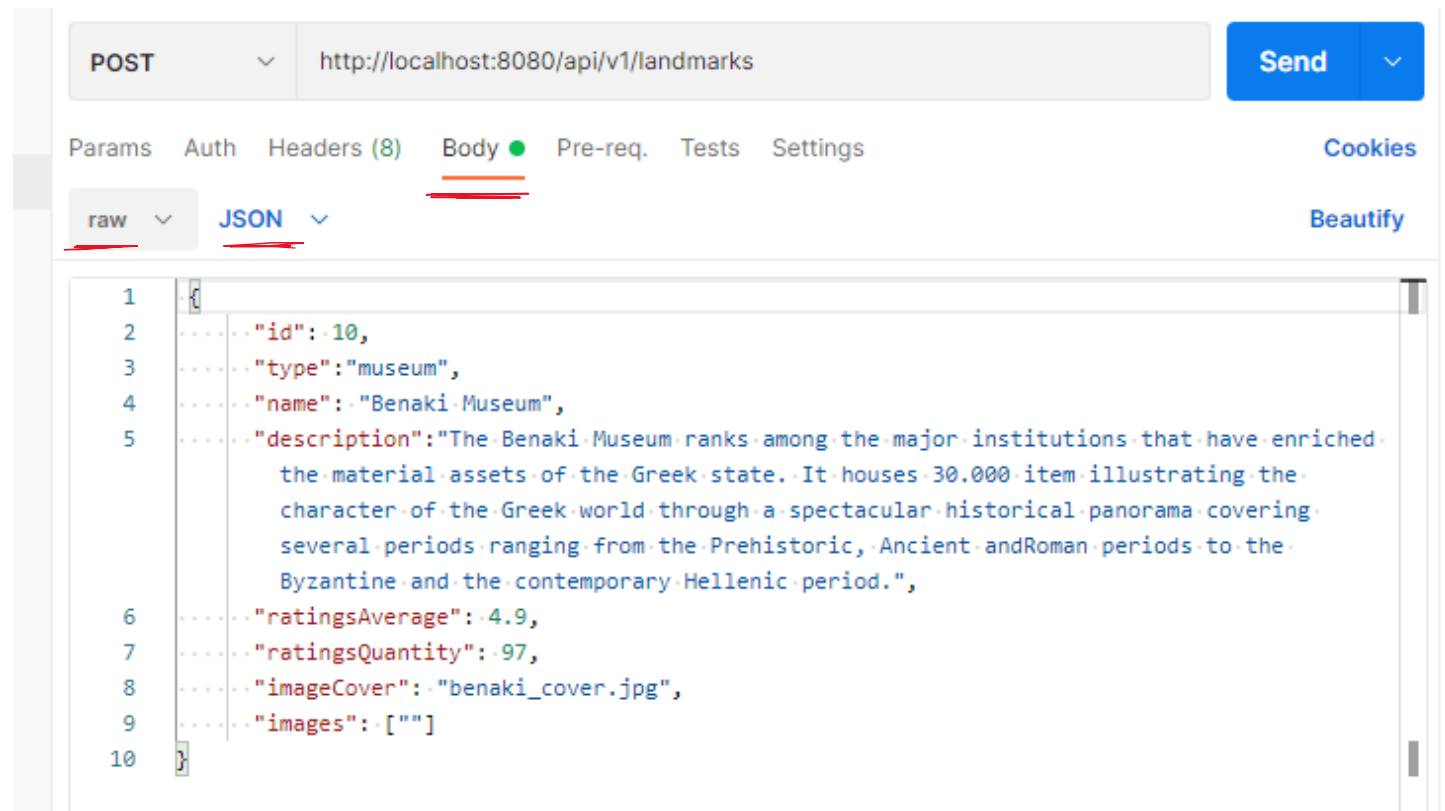
- Our code to check the request we are sending after adding the middleware

```
//lets add a new landmark
//the url here shall remain the same,
//the only thing that changes is the http method
app.post('/api/v1/landmarks', (req,res)=>{
  console.log(req.body)
  res.status(200).send('COOOL');
})
```

POST

- Lets now use postman in order to send a request to store some new data in our json.

Check
underlined
section



The screenshot shows a Postman interface for a POST request. The URL is `http://localhost:8080/api/v1/landmarks`. The request body is in JSON format, with the `Body` tab selected and underlined. The JSON content is as follows:

```
1 {
2   "id": 10,
3   "type": "museum",
4   "name": "Benaki Museum",
5   "description": "The Benaki Museum ranks among the major institutions that have enriched the material assets of the Greek state. It houses 30.000 item illustrating the character of the Greek world through a spectacular historical panorama covering several periods ranging from the Prehistoric, Ancient and Roman periods to the Byzantine and the contemporary Hellenic period.",
6   "ratingsAverage": 4.9,
7   "ratingsQuantity": 97,
8   "imageCover": "benaki_cover.jpg",
9   "images": []
10 }
```

POST

- Next we will need to store the data we have sent in our json file
- (In the future we will see how we can do this with databases!)
- We have already read the json file and-> turn it to js object



```
//before sending data we need to read it, we do this outside the route
//in the top level code that is executed once
//${__dirname}= where current script is located
var landmarks = fs.readFileSync(`${__dirname}/dev-data/data/landmarks.json`)

//convert json to javascript object
landmarks = JSON.parse(landmarks);
```

POST

1. We push the request in the json object
2. Convert landmarks from an object to a string with JSON.stringify
3. Write it to json file
4. Return the result-> newly created object

```
app.post('/api/v1/landmarks', (req,res)=>{
  console.log(req.body);

  //lets store the data!
  landmarks.push(req.body);

  //here we are NOT going to use the synchronous
  //as we are in the callback function

  //Convert landmarks from an object to a string with JSON.stringify
  fs.writeFile(`${__dirname}/dev-data/data/landmarks.json`,JSON.stringify(landmarks),err=>{
    res.status(201).json({
      status:"success",
      data:{
        landmarks
      }
    });
  });
})
```

To be continued...

<https://expressjs.com/en/resources/middleware>