

Pour effectuer une requête post et accéder au corps de notre requête c'est à dire `req.body` il va nous falloir le middleware 'body-parser' téléchargeable sur npm.

`req.body`

Contains key-value pairs of data submitted in the request body. By default, it is `undefined`, and is populated when you use body-parsing middleware such as `body-parser` and `multer`.

The following example shows how to use body-parsing middleware to populate `req.body`.

```
var app = require('express')();
var bodyParser = require('body-parser');
var multer = require('multer'); // v1.0.5
var upload = multer(); // for parsing multipart/form-data

app.use(bodyParser.json()); // for parsing application/json
app.use(bodyParser.urlencoded({ extended: true })); // for parsing application/x-www-form-urlencoded

app.post('/profile', upload.array(), function (req, res, next) {
  console.log(req.body);
  res.json(req.body);
});
```

install

```
MBP-de-Matt:nodeJs_project matt$ npm install body-parser
```

```

1  require('babel-register')
2  const func = require('function');
3  const bodyParser = require('body-parser')
4  const express = require('express')
5  const morgan = require('morgan')//middleware permettant le debug des requete http
6  const app = express();//création d'une instance de express.
7
8  // Array qui contient nos membres
9  const members = [
10     {
11         id: 1,
12         name: 'John'
13     },
14     {
15         id: 2,
16         name: 'Julie'
17     },
18     {
19         id: 3,
20         name: 'Mathieu'
21     }
22 ]
23
24 //middleware permettant de debug les requete http pendant la phase de développement
25 app.use(morgan('dev'))
26
27
28 app.use(bodyParser.json()); // permet de parser le json
29 app.use(bodyParser.urlencoded({ extended: true })); // pour parser l'application/x-www-form-urlencoded
30
31 //fonction qui nous permet de récupérer les membre en fonction de leur id
32 app.get('/api/v1/members/:id', (req, res) => {
33     res.json(func.success(members[(req.params.id)-1]))//par convention on met res.json pour envoyer un fichier json
34 })
35
36 //fonction avec req.query qui nous permet de récupérer au choix le max le membres que l'on veut.
37 app.get('/api/v1/members', (req, res) => {
38     console.log(req)
39     if(req.query.max != undefined && req.query.max > 0){
40         res.json(func.success(members.slice(0, req.query.max)))
41     }else if(req.query.max != undefined){ // gestion de l'erreur
42         res.json(func.error('Wrong max value'))
43     }else{
44         res.json(func.success(members)) //requête permettant de récupérer tous les membres
45     }
46 })
47
48 //quand on arrive avec la méthode post sur cet url cela déclenchera la fonction qui permet de rajoute un membre avec sa propriété name
49 app.post('/api/v1/members', (req, res) => {
50     if(req.body.name){
51         let member = {
52             id: members.length + 1,
53             name: req.body.name
54         };
55         members.push(member);
56         res.json(func.success(member))
57     }else{
58         res.json(func.error('no name value'));
59     }
60 })
61
62 //on donne un port a notre application avec une fonction de call back pour confirmer la conection
63 app.listen(8080, () => console.log('Started on port 8080'))
64

```

On a maintenant accès à nos différents paramètres et le rajout avec le Post fonctionne correctement.

POST http://localhost:8080/api/v1/members Params Send Save

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

form-data x-www-form-urlencoded raw binary

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> name	Mickey	
Key	Value	Description

Body Cookies Headers (6) Test Results Status: 200 OK Time: 7 ms Size: 229 B

Pretty Raw Preview JSON

```
1 {
2   "name": "Mickey"
3 }
```

GET http://localhost:8080/api/v1/members

TYPE

Inherit auth from parent

This request

The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)

Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON

```
1 {
2   "status": "success",
3   "result": [
4     {
5       "id": 1,
6       "name": "John"
7     },
8     {
9       "id": 2,
10      "name": "Julie"
11     },
12    {
13      "id": 3,
14      "name": "Mathieu"
15    },
16    {
17      "id": 4,
18      "name": "Mickey"
19    },
20    {
21      "id": 5,
22      "name": "Mickey"
23    }
24  ]
25 }
```

On vérifie si le prénom du membre que l'on veut rajouter n'est pas déjà pris

```
//quand on arrive avec la méthode post sur cet url cela déclenchera la fonction qui permet de rajoute un membre avec sa propriété name
app.post('/api/v1/members',(req, res)=>{
  if(req.body.name){
    let sameName= false;
    //boucle de test a savoir si le prénom du membres que l'on veut rajouter n'est pas déjà pris
    for(let i = 0; i < members.length; i++){
      if(members[i].name == req.body.name){
        sameName = true;
        break;
      }
    }

    if(sameName){
      res.json(func.error('name already taken'));
    }else{
      let member = {
        id : members.length + 1,
        name : req.body.name
      };
      members.push(member);
      res.json(func.success(member))
    }
  }else{
    res.json(func.error('no name value'));
  }
})
```

http://localhost:8080/ + ...

POST http://localhost:8080/api/v1/members

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary

KEY	VALUE
<input checked="" type="checkbox"/> name	Mickey
Key	Value

Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON

```
1 {
2   "status": "success",
3   "result": [
4     {
5       "id": 1,
6       "name": "John"
7     },
8     {
9       "id": 2,
10      "name": "Julie"
11     },
12     {
13       "id": 3,
14       "name": "Mathieu"
15     }
16   ]
17 }
```

http://localhost:8080/ + ...

POST http://localhost:8080/api/v1/members

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary

KEY	VALUE
<input checked="" type="checkbox"/> name	Mickey
Key	Value

Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON

```
1 {
2   "status": "success",
3   "result": {
4     "id": 4,
5     "name": "Mickey"
6   }
7 }
```

http://localhost:8080/ + ...

POST http://localhost:8080/api/v1/members

Authorization Headers (1) Body Pre-request Script Tests

form-data x-www-form-urlencoded raw binary

KEY	VALUE
<input checked="" type="checkbox"/> name	Mickey
Key	Value

Body Cookies Headers (6) Test Results

Pretty Raw Preview JSON

```
1 {
2   "status": "error",
3   "message": "name already taken"
4 }
```