Pour effectuer une requête post et accédé 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`.

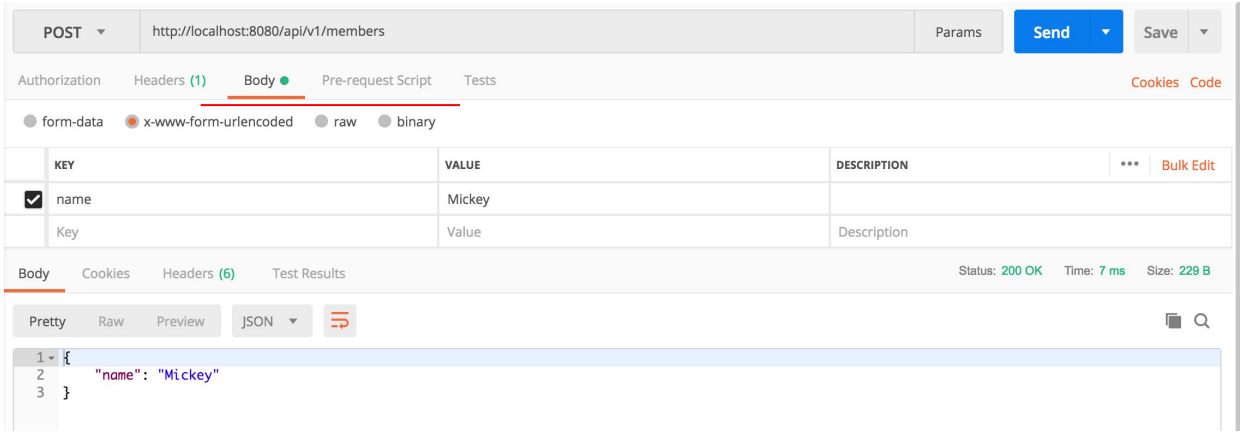
```
var app = require('express')();
var bodyParser = require('body-parser');
var multer = require('multer'); // v1.0.5
var upload = multer(); // for parsing multipart/form-data

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);
});
```

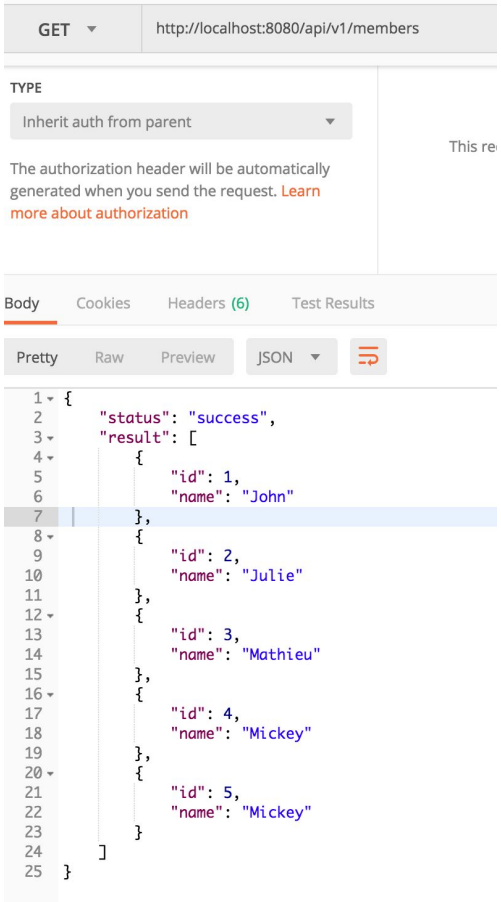
install

app.js | function.js

```
require('babel-register')
const func = require('function');
const bodyParser = require('body-parser')
const express = require('express')
const morgan = require('morgan')//middleware permettant le debug des requete http
const app = express();//création d'une instance de express.

// Array qui contient nos membres
const members = [
    {
        id: 1,
        name: 'John'
    },
    {
        id: 2,
        name: 'Julie'
    },
    {
        id: 3,
        name: 'Mathieu'
    }
]

//middleware permettant de debug les requete http pendant la phase de développement&
app.use(morgan('dev'))


app.use(bodyParser.json()); // permet de parser le json
app.use(bodyParser.urlencoded({ extended: true })); // pour parser l'application/x-www-form-urlencoded

//fonction qui nous permet de récuperer les  membre en fonction de leur id
app.get('/api/v1/members/:id', (req, res)=>{
    res.json(func.success(members[(req.params.id)-1]))//par conventtion on met res.json pour envoyer un fichier json
})

//fonction avec req.query qui nous permet de récuperer au choix le max le membres que l'on veut.
app.get('/api/v1/members',(req, res)=>{
    console.log(req)
    if(req.query.max != undefined && req.query.max > 0){
        res.json(func.success(members.slice(0, req.query.max)))
    }else if(req.query.max != undefined){ // gestion de l'erreur
        res.json(func.error('Wrong max value'))
    }else{
        res.json(func.success(members)) //requête permettant de récuperer tous les membres
    }
})

//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 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'));
  }
})

//on donne un port a notre application avec une fonction de call back pour confirmer la conection
app.listen(8080,()=> console.log('Started on port 8080'))
```

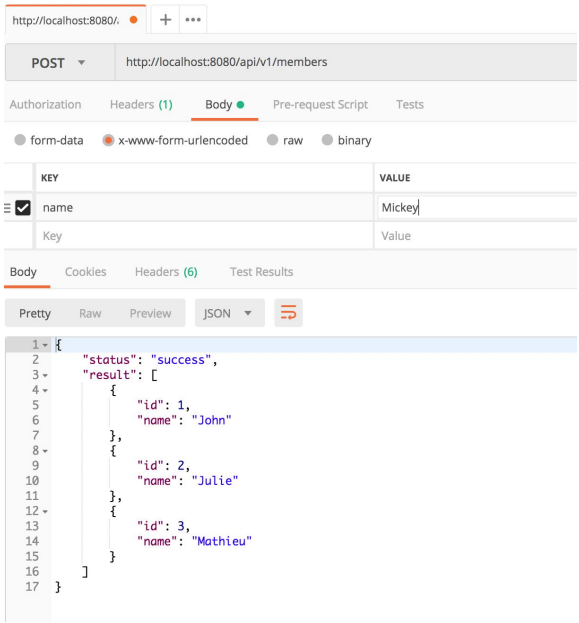
On a maintenant accès à nos différents paramètre et le rajout avec le Post fonctionne correctement.

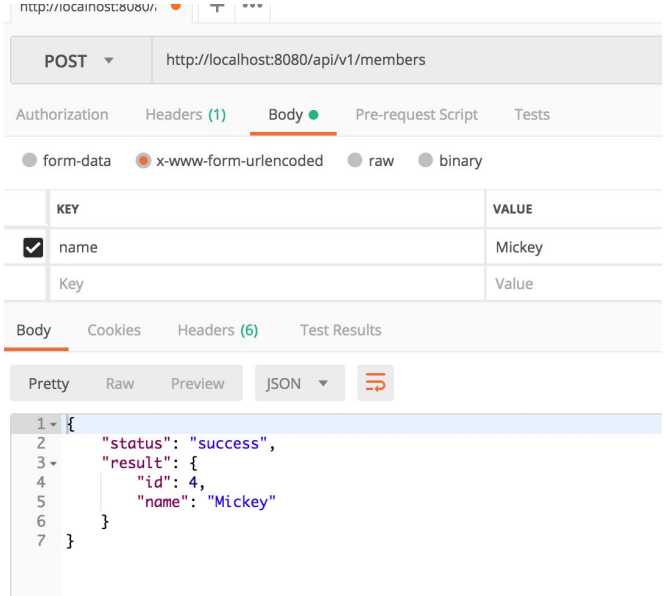


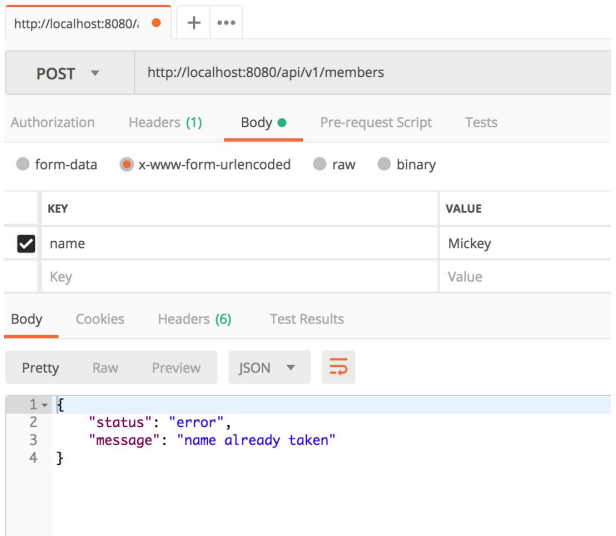


On vérifie si le prénom du membre que l'on veut rajouter n'est pas déjà pris

```
//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 déja 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
name
Mickey
Key
Value
Body
Cookies
Headers (6)
Test Results
Pretty
Raw
Preview
JSON
{
"status": "success",
"result": [
{
"id": 1,
"name": "John"
},
{
"id": 2,
"name": "Julie"
},
{
"id": 3,
"name": "Mathieu"
}
]
}


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
name
Mickey
Key
Value
Body
Cookies
Headers (6)
Test Results
Pretty
Raw
Preview
JSON
{
"status": "success",
"result": {
"id": 4,
"name": "Mickey"
}
}


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
name
Mickey
Key
Value
Body
Cookies
Headers (6)
Test Results
Pretty
Raw
Preview
JSON
{
"status": "error",
"message": "name already taken"
}