

Atelier 3 :

REST APIs avec db (mysql)

1. Installations des dépendances (mysqljs):

```
● houdaifa@houdaifa:~/everything/node/ateliers/atelier3$ npm install mysql  
  
added 10 packages in 5s  
  
22 packages are looking for funding  
run `npm fund` for details  
❖ houdaifa@houdaifa:~/everything/node/ateliers/atelier3$
```

2. Configuration de la base de donnees :

```
const mysql = require('mysql')  
const db = mysql.createConnection({  
  host: 'mysql-houdaifa.alwaysdata.net',  
  user: 'houdaifa',  
  password: 'hodofozodo',  
  database: 'houdaifa_nodejs_rest',  
})
```

3. Connection a la base de donnees :

```
db.connect((err) => {  
  if (err){  
    console.log(err.message)  
  }  
  else {  
    console.log('connected to database')  
  }  
})
```

4. Des routes :

```
membersRouter.route('/:id')
  .get((req,res) => {
    db.query('select * from members where id = ?', [req.params.id], (err, results) => {
      if (err) return res.json(error(err.message))
      if (results[0] === undefined) return res.json(error("member not found"))
      res.json(success(results[0]))
    })
  })

  .put((req,res) => {
    if (!req.body.name) return res.json(error("Missing required parameter: name"))
    db.query('select * from members where id = ?', [req.params.id], (err, results) => {
      if (err) return res.json(error(err.message))
      if (results[0] === undefined) return res.json(error("member not found"))

      db.query('select * from members where name = ? and id != ?', [req.body.name, req.params.id], (err, results) => {
        if (err) return res.json(error(err.message))
        if (results[0] !== undefined) return res.json(error("Member name already exists"))

        db.query('update members set name = ? where id = ?', [req.body.name, req.params.id], (err, results) => {
          if (err) return res.json(error(err.message))
          res.json(success(true))
        })
      })
    })
  })

  .delete((req,res) => {
    db.query('select * from members where id = ?', [req.params.id], (err, results) => {
      if (err) return res.json(error(err.message))
      if (results[0] === undefined) return res.json(error("member not found"))

      db.query('delete from members where id = ?', [req.params.id], (err, results) => {
        if (err) return res.json(error(err.message))
        res.json(success(true))
      })
    })
  })
})

membersRouter.route('/')
  .get((req, res) => {
    if (req.query.max === undefined){
      db.query('select * from members', (err, results) => {
        if (err) return res.json(error(err.message))
        return res.json(success(results))
      })
    }
    return
  })
}
```

5. Tests :

a. Ajouter un membre :

The screenshot displays a REST client interface with the following details:

- Request Method:** POST
- URL:** http://localhost:8081/api/members/
- Request Body:** A JSON object with the key "name" and value "PHP".

```
1 {  
2   "name": "PHP"  
3 }
```
- Response Body:** A JSON object indicating success.

```
1 {  
2   "status": "success",  
3   "result": true  
4 }
```

b. Recuperer les membres (avec max) :

The screenshot displays a REST client interface. At the top, a GET request is configured for the URL `http://localhost:8081/api/members/?max=1`. The 'Body' tab is selected, showing a raw JSON request:

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

 Below this, the 'Body' tab of the response is active, showing a JSON response:

```
1 {
2   "status": "success",
3   "result": [
4     {
5       "id": 1,
6       "name": "JavaScript"
7     }
8   ]
9 }
```

c. Recuperer a membre avec id :

The screenshot displays a REST client interface. At the top, a GET request is configured for the URL `http://localhost:8081/api/members/3`. The 'Body' tab is selected, showing a raw JSON request:

```
{
  "name": "Python"
}
```

. Below this, the 'Body' tab of the response is selected, showing a JSON response:

```
{
  "status": "success",
  "result": {
    "id": 3,
    "name": "Python"
  }
}
```

. The interface includes tabs for Docs, Params, Authorization, Headers (8), Body, Scripts, and Settings. The response section also includes tabs for Body, Cookies, Headers (9), and Test Results, along with options for JSON, Preview, and Visualize.

GET `http://localhost:8081/api/members/3`

Docs Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   |   "name": "Python"
3   }
```

Body Cookies Headers (9) Test Results ↻

{ } JSON ▾ ▶ Preview 🖼 Visualize ▾

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

d. Metre a jour un membre :

The screenshot displays a REST client interface. At the top, a POST request is configured for the URL `http://localhost:8081/api/members/3`. The 'Body' tab is selected, showing a JSON payload: `{ "name": "C++" }`. Below the request, the 'Test Results' tab is active, showing the response body as JSON: `{ "status": "success", "result": true }`. The interface includes tabs for Docs, Params, Authorization, Headers (8), Body, Scripts, and Settings. The response section also has tabs for Body, Cookies, Headers (9), and Test Results, along with options to Preview or Visualize the response.

```
POST http://localhost:8081/api/members/3
```

Docs Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL [JSON](#)

```
1 {
2   |   "name": "C++"
3   }
```

Body Cookies Headers (9) Test Results

`{}` JSON Visualize

```
1 {
2   |   "status": "success",
3   |   "result": true
4   }
```

GET ▼ http://localhost:8081/api/members/3

Docs Params Authorization Headers (8) **Body ●** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL

```
1 {
2   "name": "C++"
3 }
```

Body Cookies Headers (9) Test Results ↺

{ } JSON ▼ ▶ Preview 🖼 Visualize ▼

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

e. Supprimer un membre :

DELETE

http://localhost:8081/api/members/4

Docs

Params

Authorization

Headers (8)

Body

Scripts

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

```
1 {  
2   "name": "C++"  
3 }
```

Body

Cookies

Headers (9)

Test Results



{ } JSON

Preview

Visualize

```
1 {  
2   "status": "success",  
3   "result": true  
4 }
```

GET

http://localhost:8081/api/members/4

Docs

Params

Authorization

Headers (8)

Body

Scripts

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

```
1 {  
2   "name": "C++"  
3 }
```

Body

Cookies

Headers (9)

Test Results



{ } JSON

Preview

Visualize

```
1 {  
2   "status": "error",  
3   "result": "member not found"  
4 }
```