

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 données :

```
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 données :

```
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 shows the Postman application interface. At the top, a header bar indicates a **POST** method and the URL **http://localhost:8081/api/members/**. Below the header, there are tabs for **Docs**, **Params**, **Authorization**, **Headers (8)**, **Body** (which is currently selected), **Scripts**, and **Settings**. Under the **Body** tab, several options are available: **none**, **form-data**, **x-www-form-urlencoded**, **raw** (which is selected), **binary**, and **GraphQL**. A **JSON** dropdown menu is also present. The **raw** section contains the following JSON payload:

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

At the bottom of the interface, there are tabs for **Body**, **Cookies**, **Headers (9)**, and **Test Results**. The **Body** tab is selected. Below these tabs, there are buttons for **{}** (empty JSON), **JSON** (with a dropdown arrow), **Preview**, and **Visualize**. The **Visualize** button has a dropdown arrow. The **JSON** dropdown is currently set to **{} JSON**. The **Preview** section displays the response body, which is identical to the JSON sent in the request:

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

b. Recuperer les membres (avec max) :

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

- Method:** GET
- URL:** http://localhost:8081/api/members/?max=1
- Body:** Raw JSON input:

```
1 {  
2   "name": "PHP"  
3 }
```
- Headers:** (8 items listed)
- Body tab selected:** raw, JSON
- Body output:** JSON response:

```
1 {  
2   "status": "success",  
3   "result": [  
4     {  
5       "id": 1,  
6       "name": "JavaScript"  
7     }  
8   ]  
9 }
```
- Other tabs:** Cookies, Headers (9), Test Results

c. Recuperer a membre avec id :

The screenshot shows a REST API testing interface. At the top, a header bar indicates a **GET** request to **http://localhost:8081/api/members/3**. Below this, a navigation bar includes links for **Docs**, **Params**, **Authorization**, **Headers (8)**, **Body** (selected), **Scripts**, and **Settings**. Under the **Body** tab, the **raw** tab is selected, showing the following JSON payload:

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

Below the body, the **Test Results** tab is selected, showing the following JSON response:

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

d. Mettre à jour un membre :

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

- Method:** POST
- URL:** http://localhost:8081/api/members/3
- Body:** Raw JSON data:

```
1  {
2    "name": "C++"
3 }
```
- Headers:** 8 items (labeled "Headers (8)" in the UI)
- Body tab selected:** The "Body" tab is highlighted with a red underline.
- Test Results tab selected:** The "Test Results" tab is highlighted with a red underline.
- Response Body:** JSON data:

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

The screenshot shows the Postman application interface. At the top, there is a header bar with the method "GET" and the URL "http://localhost:8081/api/members/3". Below the header, there are tabs for "Docs", "Params", "Authorization", "Headers (8)", "Body", "Scripts", and "Settings". The "Body" tab is currently selected, indicated by an orange underline. Under the "Body" tab, there are four radio button options: "none", "form-data", "x-www-form-urlencoded", and "raw". The "raw" option is selected, indicated by a blue outline. Below the radio buttons, there is a code editor area containing the following JSON payload:

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

At the bottom of the interface, there are tabs for "Body", "Cookies", "Headers (9)", and "Test Results". The "Body" tab is selected. Below these tabs, there is a dropdown menu set to "JSON" and a "Preview" button. To the right of the preview button is a "Visualize" button with a gear icon. The "Visualize" button is also highlighted with an orange underline. In the main content area, the response body is displayed as:

```
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 }
```