

LU11c - CRUD Server and Postman

Learning Objectives

1. How to connect the server to the db
2. How to fetch data from the db and display it in POSTMAN
3. How to perform the CRUD operations on the server
4. How to perform the CRUD operations by using PoSTMAN

Sources

- SampleServer that can perform CRUD

Introduction

At last chapter of our database journey we want to perform all CRUD operations within the server AND the client (POSTMAN). For reasons of efficiency we are going to start with the R = READ of CRUD.

R = READ: Fetching list of data

First of all, we want to display the list of users, which are store in the table users. The following code show how to get that list from the database.

```
// Method to get data from the user table
app.get('/user', (req, res) => {
  const query = 'SELECT * FROM users'; // SQL query to select all rows
  from user table
  db.query(query, (err, results) => {
    if (err) {
      console.error('Error retrieving users:', err);
      res.status(500).send('Server error');
      return;
    }
    res.json(results); // Send the results as JSON
  });
});
```

The image shows a REST client interface on the left and a code editor on the right. The REST client shows a GET request to `http://localhost:3000/user` with a `200 OK` response. The response body is a JSON array of user objects. The code editor shows the JavaScript code for the `app.get('/data')` endpoint, which sends a `200` status and a `body` containing the system date and time. The `app.get('/users')` endpoint sends a `200` status and a `body` containing the results of a SQL query to select all rows from the `users` table. The `app.get('/user/:id')` endpoint sends a `200` status and a `body` containing the results of a SQL query to fetch a user by ID.

Key	Value	Description
Key	Value	Description

```
1 [
2   {
3     "user_id": 1,
4     "username": "demis",
5     "email": "demis@mail"
6   },
7   {
8     "user_id": 2,
9     "username": "jane_smith",
10    "email": "jane.smith@example.com"
11  },
12  {
13    "user_id": 4,
14    "username": "sarah_johnson",
15    "email": "sarah.johnson@example.com"
16  },
17  {
18    "user_id": 5,
19    "username": "chris_williams",
20    "email": "chris.williams@example.com"
21  }
22 ]
```

```
app.get('/data', (req, Response, ResBody, ResBody, ResQuery, LocalObj, res, Response, ResBody) => {
  db.query('SELECT NOW() AS system_date', (err, results) => {
    res.status(200).send({ body: 'Server error' });
    return;
  });
  res.send({ body: 'Current system date and time: ${results[0].system_date}' });
});

// MyThird.js get data from the user table
app.get('/users', (req, Response, ResBody, ResBody, ResQuery, LocalObj, res, Response, ResBody, LocalObj) => {
  const query = `SELECT * FROM users`; // SQL query to select all rows from user table
  db.query(query, (err, results) => {
    if (err) {
      console.error('Error retrieving users:', err);
      res.status(500).send({ body: 'Server error' });
      return;
    }
    res.json(results); // Send the results as JSON
  });
});

app.get('/user/:id', (req, Response, ResBody, ResBody, ResQuery, LocalObj, res, Response, ResBody, LocalObj) => {
  const userId = req.params.id;
  // SQL query to fetch user by ID
  const query = `SELECT * FROM users WHERE user_id = ?`;
  db.query(query, [userId], (err, results) => {
    if (err) {
      console.error('Error fetching user:', err);
      res.status(500).send({ body: 'Server error' });
      return;
    }
  });
});
```

From: <https://wiki.bzz.ch/> - BZZ - Modulwiki

Permanent link: <https://wiki.bzz.ch/modul/m290/learningunits/lu09/theorie/03?rev=1730810222>

Last update: 2024/11/05 13:37

