Lecture 7. REST API's Part-2

10 April 2025 21:50

🔁 Express.js API – Revision Summary

📦 Modules Used

```
js
Copy code
const express = require("express");
const users = require("./MOCK_DATA.json");
const fs = require("fs");
```

- express: For creating the server and API routes.
- fs: To read/write user data to a local JSON file.
- MOCK_DATA.json: Acts as your **mock database**.

🚀 Server Setup

```
js
Copy code
const app = express();
const PORT = 8000;
```

- Initializes the Express app on port 8000.

🔧 Middleware

```
js
app.use(express.urlencoded({ extended: false }));
app.use(express.json());
```

- express.urlencoded: Parses form data.
- express.json: Parses JSON payloads (needed for POST and PATCH).

🖥️ GET /users – Render User List in HTML

```
js
app.get("/users", (req, res) => {
  // Sends a list of first names in <ul>
});
```

📋 GET /api/users – Fetch All Users (JSON)

```
js
Copy code
app.get("/api/users", (req, res) => {
  return res.json(users);
});
```

🔍 GET /api/users/:id – Get User by ID

```
js
app.get("/api/users/:id")
```

- Finds and returns user by id.

🩹 PATCH /api/users/:id – Update User by ID

```
js
.patch((req, res) => {
   // Finds user by ID
   // Merges new data from req.body
   // Saves updated array to JSON file
})
```

❌ DELETE /api/users/:id – Delete User by ID

```
js
Copy code
.delete((req, res) => {
   // Finds and removes user
   // Writes updated user list to JSON
})
```

➕ POST /api/users – Create a New User

```
js
Copy code
app.post("/api/users", (req, res) => {
   // Adds new user with auto-increment ID
   // Writes back to JSON file
})
```

📣 Start Server

```
js
Copy code
app.listen(PORT, () => console.log(`Server started at Port: ${PORT}`));
```

☑ Features Implemented

- Full CRUD: Create, Read, Update, Delete
- HTML rendering for /users
- File persistence via fs module
- Middleware for parsing body data

Code for HTTP methods in Express.js API's:-

```
const express = require("express");
const users = require("./MOCK_DATA.json");
const fs = require('fs');

const app = express();
const PORT = 8000;

// Middleware - Plugin
app.use(express.urlencoded({ extended: false }));
```

```
app.use(express.json()); // For JSON body parsing (important for PATCH & POST)

// Routes

// Render user first names as HTML
app.get("/users", (req, res) => {
  const html = `
  <ul>
    ${users.map((user) => `<li>${user.first_name}</li>`).join("")}
  </ul>`;
  res.send(html);
});

// REST API Endpoints

// Get all users
app.get("/api/users", (req, res) => {
  return res.json(users);
});

// Get, Update, or Delete a user by ID
app
.route("/api/users/:id")
.get((req, res) => {
  const id = Number(req.params.id);
  const user = users.find(user => user.id === id);
  return res.json(user || { message: "User not found" });
})
.patch((req, res) => {
  const id = Number(req.params.id);
  const userIndex = users.findIndex(user => user.id === id);

  if (userIndex === -1) {
    return res.status(404).json({ message: "User not found" });
  }

  // Update the user fields
  users[userIndex] = { ...users[userIndex], ...req.body };

  fs.writeFile('./MOCK_DATA.json', JSON.stringify(users, null, 2), (err) => {
    if (err) {
      return res.status(500).json({ message: "Error updating user" });
    }
    return res.json({ status: "User updated", user: users[userIndex] });
  });
})
.delete((req, res) => {
  const id = Number(req.params.id);
  const userIndex = users.findIndex(user => user.id === id);

  if (userIndex === -1) {
    return res.status(404).json({ message: "User not found" });
  }

  // Remove the user
  users.splice(userIndex, 1);
```

```
    fs.writeFile('./MOCK_DATA.json', JSON.stringify(users, null, 2), (err) => {
        if (err) {
            return res.status(500).json({ message: "Error deleting user" });
        }
        return res.json({ status: "User deleted", id: id });
    });
});

// Create a new user
app.post('/api/users', (req, res) => {
    const body = req.body;
    const newUser = { ...body, id: users.length ? users[users.length - 1].id + 1 : 1 };
    users.push(newUser);

    fs.writeFile('./MOCK_DATA.json', JSON.stringify(users, null, 2), (err) => {
        if (err) {
            return res.status(500).json({ message: "Error saving user" });
        }
        res.json({ status: "User created", id: newUser.id });
    });
});

// Start server
app.listen(PORT, () => console.log(`Server started at Port: ${PORT}`));
```