SVU Assignment Day-22
12/03/2025

Node.js Assignment: Working with Inbuilt Modules and Express.js

Objective: - The objective of this assignment is to understand and implement Node.js inbuilt modules and Express.js by creating a simple backend application. The project will include routing, middleware, logging, file operations, and working with JSON data.

Project Setup

1. **Initialize a Node.js project**:
    - Run npm init -y to create a package.json file.
    - Install required dependencies: npm install express, npm install nodemon.
2. **Create the project structure:**
3. project-folder:- backend/

```
├── index.js
├── data.js
├── logger.js
├── server.log
├── products.json
├── utils/
│   ├── fileHandler.js
│   ├── systemInfo.js
├── package.json
├── README.md
```

Tasks

1. Set Up an Express Server (index.js)

- Create an Express.js server.
- The server should listen on PORT: 3000 and log a message when running.

```
const express = require('express');
const app = express();
```

```
const PORT = 3000;

app.use(express.json());

app.listen(PORT, () => {
   console.log(`Server is running on http://localhost:${PORT}`);
});
```

---

2. Create and Manage Products Data (data.js)

- Create a data.js file with an array of product objects.
- Each product should have an id, name, price, and category.

---

3. Implement Product Routes

- Implement RESTful API routes using Express:
    - GET / → Return a welcome message.
    - GET /products → Return all products.
    - GET /products/:id → Return a specific product by ID.
    - POST /products → Add a new product (data should be sent in JSON format).
    - PUT /products/:id → Update a product by ID.
    - DELETE /products/:id → Delete a product by ID.

---

4. Implement Logging Middleware (logger.js)

- Use the fs module to log each request in a file server.log.
- Format: [TIMESTAMP] METHOD: URL.
- Apply this middleware globally.

```
const fs = require('fs');
const path = require('path');

const logRequest = (req, res, next) => {
   const log = `[${new Date().toISOString()}] ${req.method}: ${req.url}\n`;
```

```
    fs.appendFileSync(path.join(__dirname, 'server.log'), log);
    next();
};

module.exports = logRequest;
```

---

5. Use Inbuilt Modules

5.1 File Handling (utils/fileHandler.js)

- Use the fs module to:
    - Read and write product data from a products.json file.
    - Update product data dynamically when adding or deleting products.

5.2 System Information (utils/systemInfo.js)

- Use the os module to display system information when the server starts.
- Log the hostname, OS type, and total memory.

```
const os = require('os');

console.log(`Hostname: ${os.hostname()}`);
console.log(`OS Type: ${os.type()}`);
console.log(`Total Memory: ${os.totalmem()}`);
```

5.3 Generate Unique IDs (crypto Module)

- Use the crypto module to generate a unique ID for each new product.

```
const crypto = require('crypto');
const generateId = () => crypto.randomUUID();
```

---

6. Bonus Tasks (Optional)

- Implement error handling for invalid routes.
- Create an authMiddleware.js file that verifies a static API key before accessing product routes.
- Store logs in a database instead of a file.

Submission Guidelines

- Submit the following files: index.js, data.js, logger.js, products.json, utils/, and server.log.
- Ensure proper code formatting and comments.

Provide a README with setup instructions.

After completing this assignment your index.js file will look something like that:

```
const express = require('express');
const fs = require('fs');
const path = require('path');
const os = require('os');
const crypto = require('crypto');
const logRequest = require('./logger');
const productsData = require('./data');

const app = express();
const PORT = 3000;

// Middleware
app.use(express.json());
app.use(logRequest);

// System Information (Logged on server start)
console.log(`Hostname: ${os.hostname()}`);
console.log(`OS Type: ${os.type()}`);
console.log(`Total Memory: ${os.totalmem()}`);

// Routes

// Home Route
```

```
app.get('/', (req, res) => {
    res.send('Welcome to the Node.js Express API!');
});

// Get all products
app.get('/products', (req, res) => {
    res.json(productsData);
});

// Get product by ID
app.get('/products/:id', (req, res) => {
    const product = productsData.find(p => p.id === req.params.id);
    if (!product) {
        return res.status(404).json({ message: 'Product not found' });
    }
    res.json(product);
});

// Add a new product
app.post('/products', (req, res) => {
    const { name, price, category } = req.body;
    if (!name || !price || !category) {
        return res.status(400).json({ message: 'All fields are required' });
    }
    const newProduct = {
        id: crypto.randomUUID(),
        name,
        price,
        category
    };
```

```
    productsData.push(newProduct);
    fs.writeFileSync(path.join(__dirname, 'products.json'), JSON.stringify(productsData, null, 2));
    res.status(201).json(newProduct);
});

// Update a product
app.put('/products/:id', (req, res) => {
    const { id } = req.params;
    const { name, price, category } = req.body;
    const productIndex = productsData.findIndex(p => p.id === id);

    if (productIndex === -1) {
        return res.status(404).json({ message: 'Product not found' });
    }

    productsData[productIndex] = { ...productsData[productIndex], name, price, category };
    fs.writeFileSync(path.join(__dirname, 'products.json'), JSON.stringify(productsData, null, 2));
    res.json(productsData[productIndex]);
});

// Delete a product
app.delete('/products/:id', (req, res) => {
    const { id } = req.params;
    const updatedProducts = productsData.filter(p => p.id !== id);

    if (updatedProducts.length === productsData.length) {
        return res.status(404).json({ message: 'Product not found' });
    }
```

```
    fs.writeFileSync(path.join(__dirname, 'products.json'),
JSON.stringify(updatedProducts, null, 2));
    res.json({ message: 'Product deleted successfully' });
});

// Handle invalid routes
app.use((req, res) => {
    res.status(404).json({ message: 'Route not found' });
});

// Start server on some port
const PORT = 8080
app.listen(PORT, () => {
    console.log(`Server is running on http://localhost:${PORT}`);
});
```