

Session 7: Building a RESTful API

Emmanuel BRUNO

Université de Toulon, LIS UMR CNRS 7020

emmanuel.bruno@univ-tln.fr

2026-10-05

Table des matières

1 Improved Session 7: Building a RESTful API with Node.js and Express	1
1.0.a 1. Introduction to RESTful APIs and HTTP Methods	1
1.0.b 2. Designing an API for a Sample E-commerce Application	1
1.0.c 3. Building a RESTful API with Node.js and Express.js	2
1.0.d 4. Testing and Security Considerations	2
1.0.e 5. Hands-on Project: Building a Simple CRUD API	2

1 Improved Session 7: Building a RESTful API with Node.js and Express

Session Structure:

1. Introduction to RESTful APIs and HTTP Methods
2. Designing an API for a Sample E-commerce Application
3. Building a RESTful API with Node.js and Express.js
 - Setting up the Project
 - Routes and Middleware
 - Database Connectivity (Mongoose for MongoDB)
 - Implementing CRUD Operations
4. Testing and Security Considerations

5. Hands-on Project: Building a Simple CRUD API

Detailed Content:

1.0.a 1. Introduction to RESTful APIs and HTTP Methods

- **RESTful APIs:**

- ▶ Architectural style for APIs based on HTTP principles.
- ▶ Advantages: standardized, scalable, and stateless communication.

- **HTTP Methods:**

- ▶ **GET:** Retrieves data from a server (e.g., `/products` - get all products).
- ▶ **POST:** Creates a new resource (e.g., `/products` - create a new product).
- ▶ **PUT:** Updates an existing resource (e.g., `/products/:id` - update product with specific ID).
- ▶ **DELETE:** Removes a resource (e.g., `/products/:id` - delete product with specific ID).

1.0.b 2. Designing an API for a Sample E-commerce Application

- **Identifying Resources:**

- ▶ Products, users, orders, categories (depending on complexity).

- **Defining Endpoints and Methods:**

- ▶ Consider CRUD operations (Create, Read, Update, Delete) for each resource.

- Include additional endpoints as needed (e.g., `/users/login`).
- **Versioning:**
 - Discuss the importance of API versioning for future changes.

Activity: Design an API for a simple e-commerce application, specifying resources, endpoints, and methods.

1.0.c 3. Building a RESTful API with Node.js and Express.js

- **Node.js:** JavaScript runtime environment for server-side development.
- **Express.js:** Popular Node.js framework for building web applications and APIs.

3.1 Setting up the Project:

- Install Node.js and npm (Node Package Manager).
- Create a new project directory and initialize it with `npm init -y`.
- Install Express.js using `npm install express`.

3.2 Routes and Middleware:

- **Routes:** Define API endpoints and corresponding HTTP methods.
- **Middleware:** Functions that handle requests before reaching route handlers.
 - Example: `app.use(express.json())` parses incoming JSON data.

3.3 Database Connectivity (Mongoose for MongoDB):

- **Mongoose:** ODM (Object Data Modeling) library for MongoDB.
 - Simplifies interaction with MongoDB using JavaScript objects.
- Install Mongoose with `npm install mongoose`.
- Connect to the MongoDB database using Mongoose connection string.

3.4 Implementing CRUD Operations:

- **GET:** Retrieve data from the database based on endpoint.
- **POST:** Create a new resource by adding data to the database.
- **PUT:** Update an existing resource by modifying data in the database.
- **DELETE:** Remove a resource by deleting data from the database.
- Implement error handling and validation for user input.

Hands-on Exercise: Implement CRUD operations for a specific resource (e.g., products) using Node.js, Express, and Mongoose.

1.0.d 4. Testing and Security Considerations

- Testing tools like Postman for sending HTTP requests and verifying responses.
- **Security:**
 - Address common vulnerabilities like SQL injection and cross-site scripting (XSS).
 - Consider authentication and authorization mechanisms for user access control (optional for a basic API).

1.0.e 5. Hands-on Project: Building a Simple CRUD API

- Students build a complete CRUD API for a chosen resource (e.g., tasks, blog posts).
- Encourage exploration of additional features like user authentication (optional).

Additional Tips:

- Provide code examples for each step of the development process.
- Offer resources for further learning on Node.js, Express, Mongoose, and API design best practices.
- Encourage students to experiment with different functionalities and explore error handling and security aspects.

This improved plan offers a more structured and practical approach to building RESTful

APIs. It incorporates a sample e-commerce application for design, uses Mongoose for a smoother database interaction experience, and emphasizes testing and security considerations. Feel free to adjust the content based on your course duration and student experience.