REST API with Express.JS and SQLite

In this tutorial, you will learn how to create a RESTful API with Express and SQLite.

Content:

- What is an API?
- What is REST?
- What is REST API?
- How to create RESTful APIs in Express?

What is API?

Application **P**rogramming **I**nterface (**API**) is a software interface that allows computer systems and applications to talk to each other. With the wide spread of the Web applications and services, the use of APIs has gained more popularity and became in the heart of software development. We use APIs every day even if we don't notice that. For example, each time you check the weather or book a travel ticket on your smartphone, an API is called in the background to a server that provide the API service.

Types of API

APIs enable businesses to open their data and applications functionality (services) to external third-party users and developers. Based on their use and application, APIs can be of three types:

1. Program API

Program APIs are based on Remote Procedure Call (RPC) technology. The RPC technology makes a remote program component appear local to the rest of the software. The service-oriented architecture API of Microsoft's WS-series is an example a program API.

2. Local API

Local APIs are OS or middleware services offered to an application program. For instance, Microsoft's .NET APIs and database access APIs are forms of Local APIs.

3. Web API

Web API is an Interface that can be accessed using the HTTP Protocol. It allows a large number of client entities, like Smartphones, Tablets, Laptops, or web browsers to interact with the server. A Web API can be developed using various technologies like Java, ASP.NET, and Node.JS providing superior performance and faster service development.

API Protocols & Architecture

Based on the purpose, an API follows a range of protocols and standards. Below are a few essential API protocols and architectures:

1) XML-RPC

The XML-RPC protocol was created by Dave Winer to exchange information between two or more networks. The client performs RPC by using XML to encode its calls and HTTP requests for data transfer.

2) JSON-RPC

The JSON-RPC is a lightweight RPC encoded in JSON, similar to XML-RPC. It allows notifications and multiple calls to the server, which may be asynchronously answered.

3) SOAP Protocol

Simple Object Access Protocol (SOAP) is an established Web API protocol for exchanging structured information. It uses XML to Authenticate, Authorize, and Communicate processes running on operating systems. SOAP allows clients to invoke web services and receive responses independent of language and platform.

4) REST

Representational State Transfer (REST) is an architectural style to provide standards for data exchange between systems on the web. REST is a set of rules and guidelines for creating resource-based web services.

REST architecture makes the implementation of Client and Server independent, so each side can implemented and managed without affecting the operation of the other.

An API that follows the REST architecture (REST API) is called RESTful API.

REST Rules

Here we take a look at some of the most important concepts of a RESTful API. This list is not exhaustive but is designed to give an overview of how REST architecture is designed. Not all APIs are RESTful APIs — and some APIs take inspiration from the principles of REST without following the rules completely.

1. **RESTful APIs have a predefined set of operations available to its users**

API Developers and documentation talk about "functionality being exposed via an API." However, APIs do not allow for a complete control over back-end systems. Instead, predefined set of operations are made available to users of APIs. Good APIs come with clear documentations that describe how to send data in the correct format and what data to expect from each piece of API functionality.

New features are often added to APIs using a versioning system. As a result, you may end up working with differently versions of the API URLs in your applications. For example, you might see https://myawesomeapi.dev/api/v1/users or https://myawesomeapi.dev/api/v2/users (Notice the difference between the URLs — /v1 vs /v2).

2. RESTful APIs are based on resources

Responses from RESTful API calls over HTTP (usually called payloads) are returned as TEXT, HTML, XML, or JSON representations of resources that exist in the server as stored objects. A stored object is called a Resource, and can be anything, such as a blog post stored in a document database, a user data stored in a relational database, or a URL of a hosted image.

The URLs for RESTful API endpoints must be descriptive and self-documenting. Each URL requesting a resource or requesting to modify a resource describes what type of resource is being requested from the API.

For example, the URL that requests information about a single User an HTTP GET request must indicate the name of the resource "/users" and the unique ID of the user.

https://cit.uot.ly/users/{user_id}

3. RESTful APIs allow resources to be read or modified

In the example above, we used HTTP GET request to read information about a User. However, RESTful APIs allow you to Create, Read, Update and Delete resources in separate API calls. These Four operations are called CRUD operations.

RESTful API calls are performed on a single URL representing the type of the resource you wish to create or modify (e.g. http://localhost/users/{id}). The HTTP action method specifies which CRUD action you want to perform. The Read operation uses an HTTP GET request, the Create operation uses the POST request, the Update operation uses HTTP PUT method, and the Delete operation uses HTTP DELETE request. (We will see how to implement this below)

4. RESTful APIs are stateless

Stateless means no information is stored in the server between API calls. In stateless architecture, each request to an API must be processed based only on the information sent with the request.

For example, if you make a call to the API with a userID parameter, the API will not remember that userID in the next new requests. If you wish to make more calls to the same API, you must send a userID in all new requests.

In summary

An API allows different software systems to communicate with each other using code. There are different types of APIs.

REST is a set of rules and guidelines for creating a particular type of API. An API that follows the REST rules is called RESTful API and not all APIs are RESTful APIs.

REST stands for Representation State Transfer. RESTful APIs:

- Have a predefined set of operations available to its users.
- Are based on resources.
- Usually allow CRUD operations on resources.
- Are stateless

5. **Practical Example:** Building RESTful API using Express JS

Now, we will create a simple CRUD REST application for a Library Management System using Node.js and Express.js. To build this application, you will need to create a new project and install the following: Express.js, and nodemon (Node Monitor)

1. Create the project directory
   To create your project directory, open the command prompt and navigate to your projects directory. Once there, use the command mkdir to create a new directory then call npm init as follows:

```
D:\> cd www\projects

D:\ www\projects >mkdir library-api

D:\ www\projects>cd library-api

D:\ www\projects\library-api>npm init -y
```

2. Install dependencies
   Use npm to install express.js, and nodemon

```
D:\ www\projects\library-api>npm install express nodemon
```

3. Open the project in MS Code

```
D:\ www\projects\library-api> code .
```

4. Create app.js file

Create your application server file app.js and use the following code:

```
//load Express
const express = require('express');
const app = express();
app.use(express.json());

const books = [
{title: 'Harry Potter', id: 1},
{title: 'Node js', id: 2},
{title: 'Express', id: 3}
]

//READ Request Handlers
app.get('/', (req, res) => {
res.send('Welcome to the Library REST API with Express.js
Tutorial!!');
});
//READ Request Handlers to send back all books
app.get('/api/books', (req,res)=> {
res.send(books);
});
//READ Request Handlers to find one book given its id
app.get('/api/books/:id', (req, res) => {
const book = books.find(c => c.id ===
parseInt(req.params.id));
if (!book) res.status(404).send('<h2>Sorry... Cant find
what you are looking for!</h2>');
res.send(book);
});
```

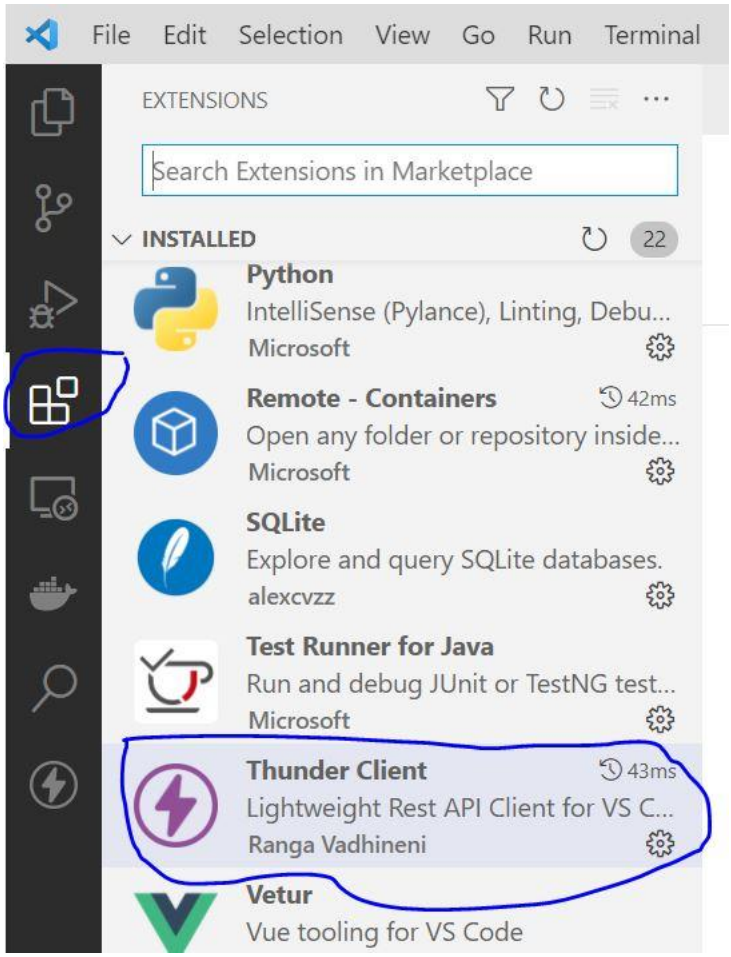
```
//CREATE Request Handler
app.post('/api/books', (req, res)=> {

const book = {
      id: books.length + 1,
      title: req.body.title
};
books.push(book);
res.send(book);
});
//UPDATE Request Handler
app.put('/api/books/:id', (req, res) => {
const book = books.find(c=> c.id ===
parseInt(req.params.id));
if (!book) res.status(404).send('<h2>Not Found!!</h2>');
book.title = req.body.title;
res.send(book);
});
//DELETE Request Handler
app.delete('/api/books/:id', (req, res) => {
const book = books.find( c=> c.id ===
parseInt(req.params.id));
if(!book) res.status(404).send('<h2> Not Found!! </h2>');

const index = books.indexOf(book);
books.splice(index,1);

res.send(book);
});
```

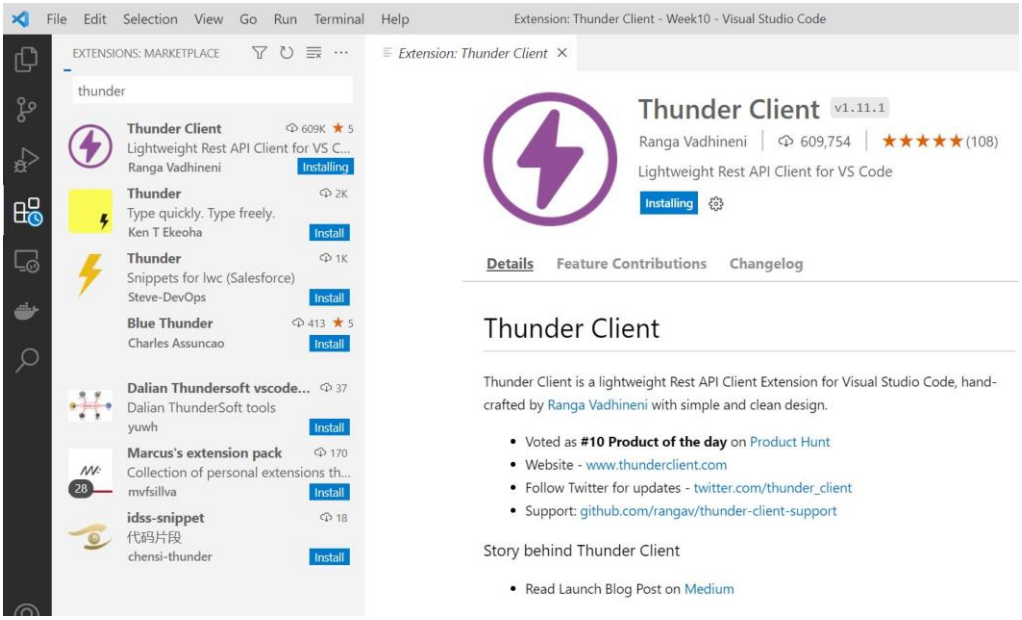
```
//PORT ENVIRONMENT VARIABLE
const port = process.env.PORT || 8080;
app.listen(port, () => console.log(`Listening on port
${port}..`));
```

6. **TEST the API:**

To test your API you need to install the VS Code package Thunder Client, a REST API lightweight testing tool.

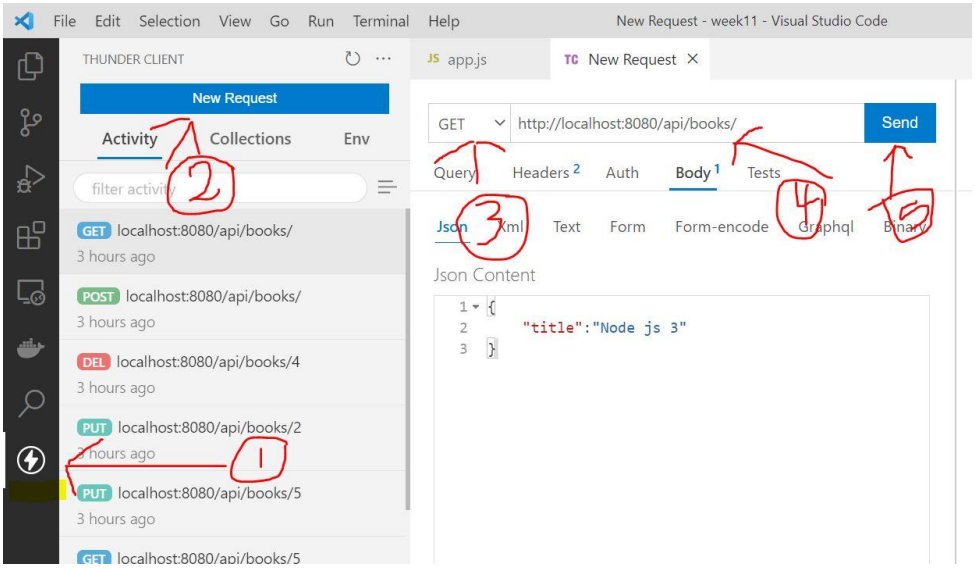
Click in the packages icon on VS Code and search for 'Thunder Client'





After package is installed, run your project file and open new request as follows:

1- Click on the Thunder Client icon on the left menu.
2- Click on New Request
3- Choose GET method
4- Enter the URL http://localhost:8080/api/books
5- Click Send



To READ one book:

1- Choose the GET method
2- In the url type http://localhost:8080/api/books/1
3- Click send

To Create a new book:

4- Choose the post method
5- In the body tab write {"title": "New book 1"}
6- In the url type http://localhost:8080/api/books
7- Click send

To Update/change the title of a book:

1- Choose the put method
2- In the body tab write {"title": "New Book Title"}
3- In the url type http://localhost:8080/api/books/1
4- Click send

To Delete a book:

1- Choose the DELETE method
2- In the url type http://localhost:8080/api/books/3
3- Click send