**Creating a REST API**

**Overview**

In this lab, you'll implement a REST API using NodeJS. In order to complete this, you will need to install NodeJS if it is not already available.

**Source Folders**

* <LAB\_HOME>\labs\rest
* <LAB\_HOME>\solutions\rest

**Familiarization**

Using your preferred IDE, open the example REST API implementation <LAB\_HOME>\labs\rest\restapi.js

This is the example you were introduced to in the chapter. You will now refactor this example to return restaurant information.

We have provided four sample JSON files for your REST API. One contains a list of restaurants, one is a Thai restaurant and the other is a Burger restaurant. The final file is a list of orders that you can add new orders to.

Your task is to implement a REST API allowing users to:

1. List restaurants
2. Return a restaurant by ID (which will work for just the two provided restaurants)
3. Place an order
4. Retrieve an order by ID
5. Delete an order by ID

**Exercise 1: Verify the NodeJS Installation**

1. Verify that Node is installed by launching a command prompt typing node --version which will tell you the version you have. If it says command not found, then complete the following steps. If it comes back with version 4.x.x, then move onto step 5.
2. Download the installer from <https://nodejs.org/en/download/>
3. Run the installer accepting all of the defaults.
4. In a new console, repeat step 1 to verify that you now have a working NodeJS installation.
5. Launch a command prompt in <LAB\_HOME>\labs\rest.
6. At the prompt, type node restapi.js
7. Launch a browser and enter the URL <http://localhost:8081/albums>, and verify that a list of albums is returned as JSON.
8. Return to the console and type Ctrl C to terminate your REST API.

**Exercise 2: Update the REST API to Return a list of Restaurants**

Open the restapi.js file in your preferred editor, and update the method that returns a list of albums to return a list of restaurants instead. We have provided the file GetNearbyRestaurants.json that you can return to the caller. Come up with an appropriate endpoint for this API call.

**Exercise 3: Update the REST API to Return a specific Restaurant by ID**

Now update the method that currently returns an album by ID and change it so that it returns a restaurant by ID instead. You will support ID 6 and ID 25. All other IDs will return a 404.

You can use the files burgerRestaurantMenu.json and thaiRestaurantMenu.json to be the returned JSON for the two available restaurants.

**Exercise 4: Place an Order**

In this final main exercise, you will allow users to place an order. You can decide on a JSON structure for the order based on the information required. We will not actually be validating this on the server side since that would take too long, so for now, just create a simple order structure. Some sample orders are provided in the file orders.json.

Orders supplied by your endpoint must be added into this list of orders, but they do not need to match the same JSON structure. You can use your JSON structure from the JSON chapter as your order to be submitted.

**Exercise 5 (If time Permits): Remove an Order**

If you have time, update the delete method to allow users to remove an order by ID.