Lab 8 – REST APIs

1 Introduction

Welcome to Week 8!

This lab is important for your coursework. RESTful API is an important concept and means for modern websites work and is the difference between a static website and a rich web application.

2 Learning Objectives

- To understand what REST APIs are, how they work, and how to interact with them
- To practice using Postman for testing API specifications

3 Understanding REST APIs

3.1 Exercise 1: Understanding REST API specifications

Familiarise yourself with how RESTful APIs work. The lecture is a good starting point, but do some research until you fully understand the concepts. There are many tutorials and explanations on the web. For example:

- https://restfulapi.net/
- https://www.redhat.com/en/topics/api/what-is-a-rest-api
- https://youtu.be/7YcW25PHnAA

3.2 Exercise 2: Starting the server

This exercise will only work on the lab machines. If you are running on your own device, you will first need to install NodeJS from (https://nodejs.org/en/). You will need NodeJS for later units also.

1. Create a folder somewhere on your one drive.
2. Download movieAPI.zip (moodle) to that folder. Unzip it, and you will find two files, server.js and package.json.
3. Open your command line application from the start menu.
4. The command line application will open in your home, you need to navigate to the project folder by typing “cd < your filepath>”
5. You should now see that you are in the directory where you unzipped the server code. If so, then run ‘npm install’ (a directory called ‘node_modules’ should appear with all of the dependencies).
6. Run ‘npm start’ to run the server. The terminal should print ‘Listening on port 3000…’

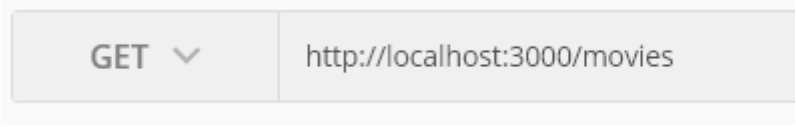
3.3 Exercise 3: Testing our understanding with Postman

Again, you will need to install Postman if you are working on your own device. Postman can be installed from https://postman.com/

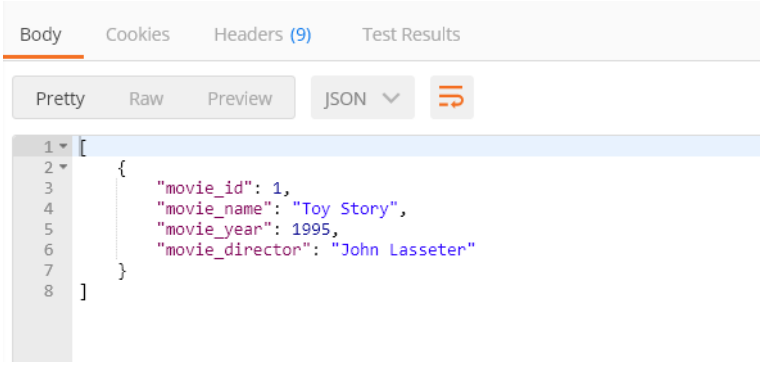
Here is the API specification for the MovieDB API.

Method	Route	Description
GET	/movies	Retrieve a list of all movies
POST	/movies	Add a new movie
GET	/movies/:id	Retrieve a single movie using the ID
PATCH	/movies/:id	Update a movie using the ID
DELETE	/movies/:id	Delete a movie at a specific ID

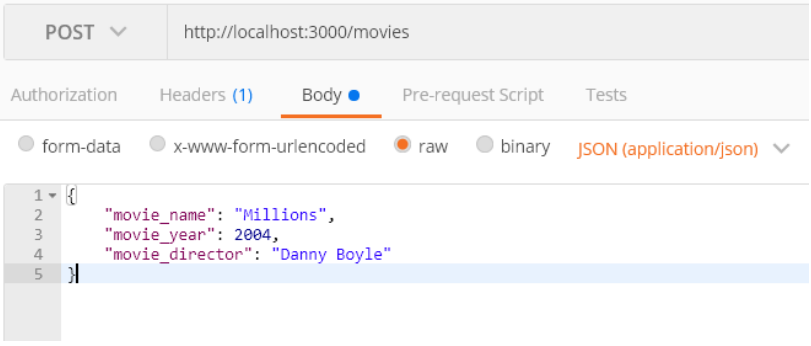
1. Open Postman from your start menu
2. Along the top of Postman, we have all the options necessary for sending HTTP requests. The responses will then appear at the bottom of the application window. Let's start by retrieving a list of all the movies in the DB (e.g., a GET request to /movies). The server is hosted on our local machine (that command line window) on port 3000. Therefore, we need to specify the URL and port in our request. Copy the below to do this, and click the "Send" button.



3. In the response window, we can see that an array of JSON objects has been returned. However, we currently only have one movie in the DB.

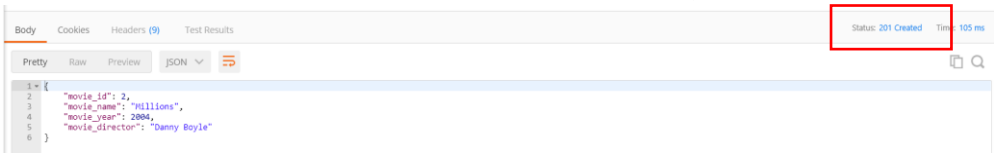


4. Let's add a new movie. For this we will need to send a POST request to /movies. Select POST from the list of methods to send.
5. The POST request also requires a body to be sent with the data for the new movie. Navigate to the body tab and write copy in the JSON object below. Make sure to also check that you are sending "raw" JSON by selecting these options (see the screenshot)

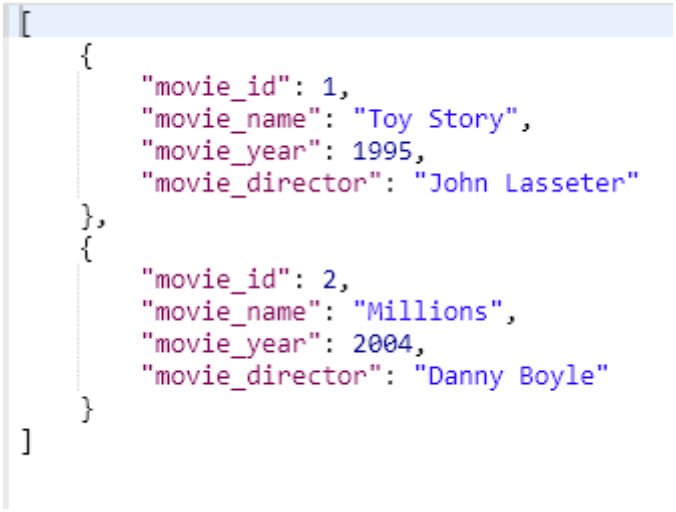


6. Note that the movie_name and movie_director fields are strings, whereas the movie_year is a number. Click "Send" again and look at the response. The server has responded with a

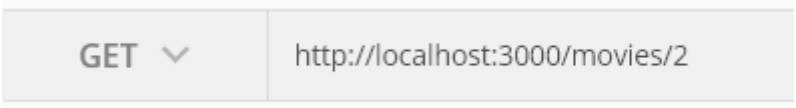
"201 Created" meaning that the movie was successfully added (the screenshot is small, but see the red box for an indication of where to check in Postman)



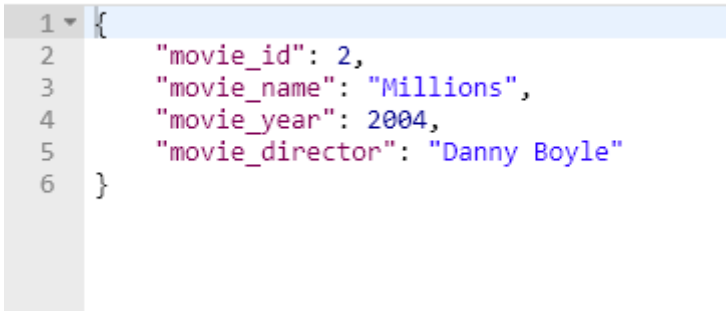
7. Now run the GET request again, can you see the movie in the list?



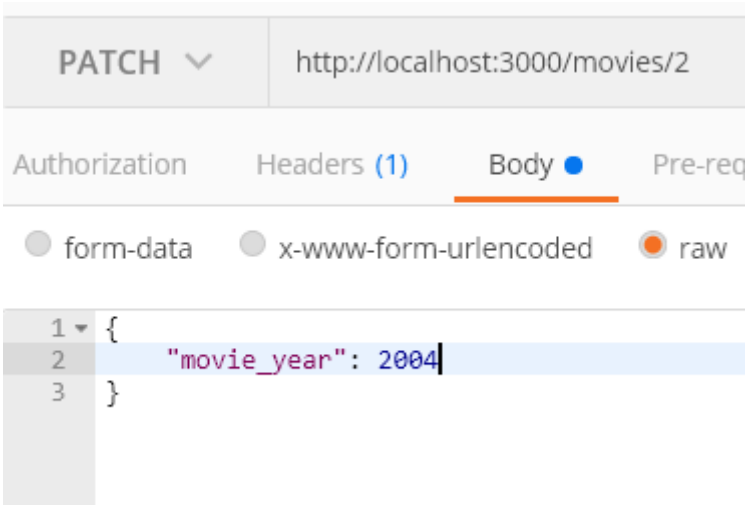
8. Try adding another movie, but with bad data. Try different data types or a year before 1900. Can you break the server, or does it handle all bad data gracefully? (If you can break it, then it's because of Ash's shoddy coding).
9. We can also retrieve a single movie by specifying the ID as a path parameter. Send a GET request to /movies/2 to see just your newly added movie.



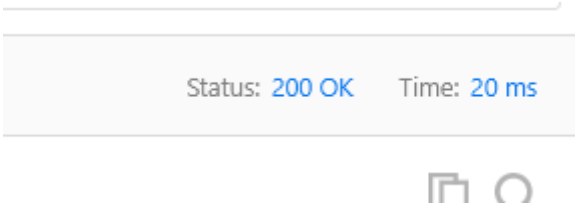
10. Note the response isn't an array this time. It's just the single movie



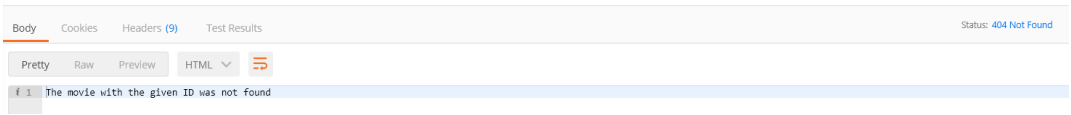
11. We can update the movie using the PATCH method. Update the method to PATCH and keep the URL as /movies/2. Let's update the year to 2021 in the body of the request



12. There is no body to the response, but we know it was successful because the server returned a “200 OK” response.



13. Double check that the update worked by sending a GET request
14. Finally, we can delete the movie. Change the method to DELETE and keep the URL as /movies/2. Send the request and check that it was successful with a “200 OK” response
15. Test that it has been deleted with a GET request. Note that now we have a “404 Not found” response and the server returns a handy error message.



4 Assignments!

Use Postman to test out the assignment API. The assignment specification has details of the post endpoint that are to be implemented in the assignment. Familiarise yourself with both ‘get’ and ‘post’ endpoints.

The URL: https://mudfoot.doc.stu.mmu.ac.uk/node/api/creditcard