Oracle MOOC: Node JS Microservices for Oracle Cloud

Session 1

Week 1 Homework

You will learn:

- Installing NodeJS on your VM
- Using the Read-Eval-Print Loop (REPL) or Node shell
- Using NPM to create your NodeJS projects
- Creating and starting a NodeJS server
- Running JavaScript code in NodeJS
- Importing data from a JSON file.
- Implementing the HTTP methods in NodeJS to create RESTful web services
- Using cURL and a browser to interact with your RESTful web services
- Installing Express and the Body Parser module
- Testing your NodeJS microservices with a REST client UI

You will need:

- Download and setup your Oracle pre-built Developer VM

Note:

- The solutions to the homework are NOT provided. We encourage you to try it out and discuss among students in the forum for further learning
- The homework is NOT mandatory

Oracle MOOC: Node JS Microservices for Oracle Cloud

Application Overview

The application you must build in your homework is a Movie Collection and Reviews RESTful web service.

The application stores and lets you manage a movie collection.

A movie is a JavaScript Object with the following properties:

- name : string
- year : number
- studio : string
- genre : string
- rating : string
- runtime : number
- director : string
- description : string
- reviews : array of Review objects

Reviews are JavaScript Objects with the following properties:

- name : string
- score : number
- date : string
- description : number

You may download a sample JSON file from:

http://www.oracle.com/webfolder/technetwork/tutorials/mooc/nodejs/downloads/movies.json

Oracle MOOC: Node JS Microservices for Oracle Cloud

Assignment 1-1: Create a Node.JS Microservice

You are part of the development team that is adopting NodeJS to create microservices. Your first assignment is to provide a microservice to query data in your NodeJS application from a JSON file.

Load Data from JSON

1. Create a Node.JS microservice with express that can handle the following HTTP method and URL path combinations:
    - **`GET /`** list all the movie objects
    - **`GET /:movieIndex`** get a movie in a particular position, 0 is the first one.
    - **`POST /`** add a movie object to the application
    - **`PUT /:movieIndex`** update the movie object in a particular index.
2. Movies are loaded from the movies.json file. Load this file in then assign it to a variable so that you can use that collection of data to return a valid response in your microservices.
3. Remember to use the PORT environment variable or a default value to prepare your application for Oracle Cloud Deployment.

Once you are done you will be ready for next week, for extra challenge try the next two homework assignments.

You may try any of the following assignments in any order.

Oracle MOOC: Node JS Microservices for Oracle Cloud

Assignment 1-A: Generate a Summary Of Movie Review Data

When you request the **`GET /`** resource the application provides all movies with its reviews.

This time you must summarize the review information to return simple movie objects without individual reviews.

Generate Summary Objects

To generate the summaries you must modify the **`GET /`** method

1. Iterate all movie objects and for each one generate a copy with the movie data: name, year, studio, genre, rating, runtime, director and description.
2. Add a score property to the summarized movie object and assign the average of the review scores to it.
    a. Reviews are inside the review property of the movie object, it is an array and it might be undefined.
    b. Reviews have a score property which is a number
3. Add a reviews property with the length of the reviews property
    a. Remember, the reviews property is an array and it might be undefined.

Oracle MOOC: Node JS Microservices for Oracle Cloud

Assignment 1-B: Assign IDs to the Movies to Query by ID Instead of Indexes

Movies inside the JSON file doesn't have IDs, therefore when you try to get, modify or delete a particular movie you use the index of the movie in the array in memory. This is not optimal since deleting an element results in the entire array shifting, the proper way to do update and delete operations is using IDs instead of Indexes.

Create ID based methods

1. Identify and modify he HTTP methods that need to change in order to use IDs:
    - `GET /:movieIndex` → `GET /:id` get the movie using the ID.
    - `POST /` When adding the movie also set the ID of that movie.
    - `PUT /:movieIndex` → `PUT /:id` update the movie object with a particular ID.
2. IDs should not be repeated. To accomplish this I recommend using a number variable and modify it only when adding a new movie to the collection.
   Movie objects inside the collection must have a unique ID.
   You can create a method that adds a movie and sets the ID
3. When you load the JSON file, add each item individually to make sure all items have IDs
4. Figure out how to store and search the collection you can:
    - Just iterate the array and search for items by ID
    - Transform the array into an object where the property name is the ID and the property value is the movie object
    - Have an array and an object, the array has all the movies, the object is a indexed by id

Congratulations, you have successfully completed homework for Week 1 of NodeJS Microservices!