# Tasks Management with NodeJs Framework

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Author by: Yansen Z Q Yang

Date: 19 March 2022

Abstract:

This is an exercise to use a tech of new trending named "NodeJs" to

develop a typical RESTful API web service.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Project infrastructure : NodeJs

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

After a brief study and investigation, I think NodeJs is a tech of new trending

and it is quite suitable for this project requirement.

As an asynchronous event-driven JavaScript runtime, Node.js is designed to build scalable network applications.

Node.js is similar in design to, and influenced by, systems like Ruby's Event Machine and Python's Twisted.

Node.js takes the event model a bit further. It presents an event loop as a runtime construct instead of as a library.

In other systems, there is always a blocking call to start the event-loop. Typically, behavior is defined through callbacks

at the beginning of a script, and at the end a server is started through a blocking call like EventMachine::run(). In Node.js,

there is no such start-the-event-loop call. Node.js simply enters the event loop after executing the input script. Node.js exits

the event loop when there are no more callbacks to perform. This behavior is like browser JavaScript — the event loop is hidden from the user.

HTTP is a first-class citizen in Node.js, designed with streaming and low latency in mind. This makes Node.js

well suited for the foundation of a web library or framework.

This project, using below components: NodeJS framework + mysql

NodeJs project provide service background and listening http port 80.

when receive http request, NodeJs application will be routed to the specified logic according to HTTP action (GET,POST,PUT,DELETE) and the url,

connect to mysql server to do CRUD operations for task records. and then return to user in JSON format.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

RESTful API List introduction

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. RESTful API, action GET to retrieve a task by id

param @id : (mandatory) type integer, the record id of the task

invoked by:

curl http://your\_server/task/@id

invoking sample:

curl http://111.231.31.105/task/1

output:

All matching data in array JSON format.

------------------------------------------------------------------------------

2 RESTful API, action GET to retrieve a task by due Date

param @due\_date : (mandatory) type date, the record due date of the task. e.g.: 2022-03-29

invoked by:

curl http://your\_server/task/due\_date/@due\_date

invoking sample:

curl http://111.231.31.105/task/due\_date/2022-03-29

output:

All matching data in array JSON format.

------------------------------------------------------------------------------

3. RESTful API, action post to add new task data

param @title : (mandatory)type string, the title of the task

param @due\_date : (mandatory)type string, the due date of the task due. e.g. 2022-05-06

invoked by:

curl -X POST -d "title=@title&due\_date=@due\_date" http://your\_server/task

called by sample:

curl -X POST -d "title=This is an inital task of day.&due\_date=2022-5-19" http://111.231.31.105/task

output:

Message of result.

------------------------------------------------------------------------------

4. RESTful API, action put to modify an existing task data

param @title : (mandatory)type string, the title of the task

param @due\_date : (mandatory)type string, the due date of the task due. e.g. 2022-05-06

param @id : (mandatory)type integer, the id of the task. Must be an existing task id

invoked by:

curl -X PUT -d "title=@title&due\_date=@due\_date&id=@id" http://your\_server/task

called by sample:

curl -X PUT -d "title=This is an initial task of day.&due\_date=2022-5-19&id=6" http://111.231.31.105/task

------------------------------------------------------------------------------

5. RESTful API, action DELETE to delete a task by id

param @id : (mandatory)type integer, the id of the task. Must be an existing task id

invoked by:

curl -X DELETE http://your\_server/task/@id

called by sample:

curl -X DELETE http://111.231.31.105/task/1

------------------------------------------------------------------------------

6. API, stop server ,Graceful shutdown, close db connections before exit

invoked by:

curl http://your\_server/shutdown

called by sample:

curl http://111.231.31.105/shutdown

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Linux Deployment Environment Setup Guide(Centos for example)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 1: install node js by below command

yum install nodejs

step 2: install cnpm by below command

npm install -g cnpm --registry=https://registry.npm.taobao.org

step 3: install expensions by below command

cnpm install express --save

cnpm install body-parser --save

cnpm install cookie-parser --save

cnpm install multer --save

cnpm install mysql

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Deployment Instruction

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

step 1: Get the codes from gitHUB

step 2: put the server end sources into linux server

2.1 open source file "RESTful\_server\_app.js", and fill in the mySQL connection profile info for coding line 20-25

for demo this project, you can use de default mysql server and profile in the script.

2.2 install mysql-server, or use the demo mysql server.

2.3 create 2 tables with script file "tasksDbTables.sql"

2.3 use command to start the server by:

sh startRESTfulServer.sh

2.2 use command to stoop the server gracefully.

sh stopRESTfulServer.sh

step 3: use curl to invoke RESTful API from any endpoint

please refer API format in section "RESTful API introduction" above

step 4: to interact with command line in user end, below commands are supported

Notice: Please change the server host ip in tasks.sh , line 2

(tasks.sh script supports operation name as parameter 1: done/add/list/set/modify/delete)

./tasks.sh list --expiring-today

: to list tasks due date = today

./tasks.sh list

: to list all tasks

./tasks.sh list 1

:to list tasks of id 1

./tasks.sh add "title1" 2022-09-08

: to add a task of title1, and specified due date

./tasks.sh set 15 "title2" 2022-09-08

: to modify a task of id 15 , with new title2, and specified due date

./tasks.sh done 15

: to delete the task of id 15