Project 1: How to Use and Extend HTTP
KECE449 Computer Networks
Spring, 2022

Instructor: **Hwangnam Kim**

TA: **Changmin Park**

Overview

The purpose of this project is to experience and to be familiarized with network programming, using Node.js. Students are required to fill in empty functions or HTML documents in order to get outputs and get grades on those outputs. With this project, the students can run how to devise a network application over HTTP protocol.

Objectives

In this project, students are supposed to implement a simple REST API that lets users to interactively communicate with a web server implemented with Node.js. Note that REST APIs are designed to demonstrate how useful and how the HTTP protocol can be used to build a variety of applications that are not simply a means of communication between a web server and a web browser. Do not panic by "referring to the REST API" as it is simply an extension of the use of HTTP.

Guidelines

All students should work on this project **individually**. Collaboration, discussion, code sharing, and any other form of group workare forbidden. Otherwise, you cannot get any point from this project.

The implementation environment is as follows:

- Ubuntu 18.04 (You can also use Ubuntu 16.04, 20.04, and implement on Virtual Machine)
- HTTP
- HTML
- Javascript

You are required to make sure that your programs run on TA's Ubuntu environment. Programs suffering runtime errors will earn no credit.

Please indent and document your code for better readability. Use meaningful names for variables and follow other style guidelinesthat enhance the clarity of your code.

Project Description

① Specification:

You are required to write two node.js files and one HTML file for this project. If the put and delete parts are implemented additionally, bonus points are given.: The name of your files should be ***restServer_get.js, restServer_post.js, and about.html***, respectively. You can freely use any name for the bonus part.

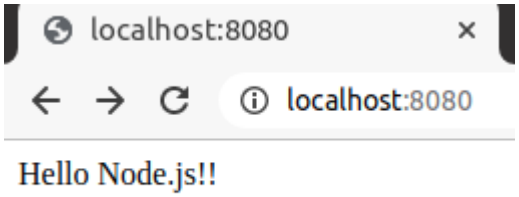
restServer_get.js **about.html** **restServer_post.js**

You are responsible for completing implementation of these files in order to make the program run properly. You are not allowed to modify any codes in these files.

1. restServer_get.js

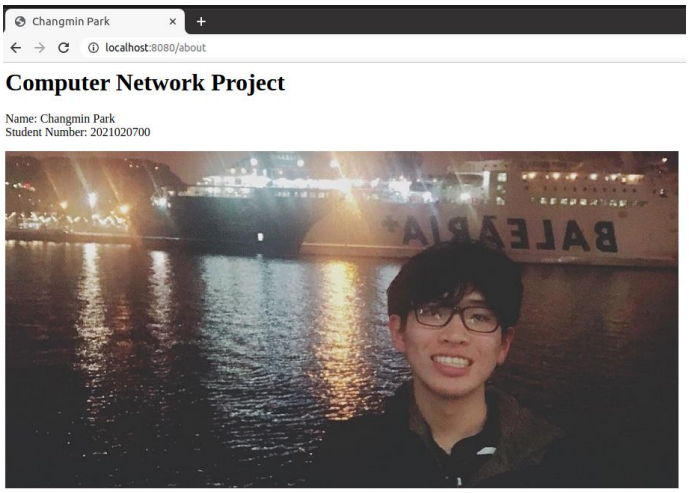
In this part, GET method, one of the REST API's, is implemented as a node. When a client sends a reque st to the server, the server processes the request and sends a response back to the client after processing. Create a node server that responds to HTTP requests and waits for events requested by createServer. Whe n the node restServer_get.js command is given, it waits on the specified local host port, 8080. Additionally , you use fs to load the HTML file on demand. If you input http://localhost:8080/ on web, "Hello Node.js" is output, and if you input "http://localhost:8080/about", your information is output.

```
root@chang-System-Product-Name:/workspaces/nodejs/final# node restServer_get.js
waiting on 8080 port
```



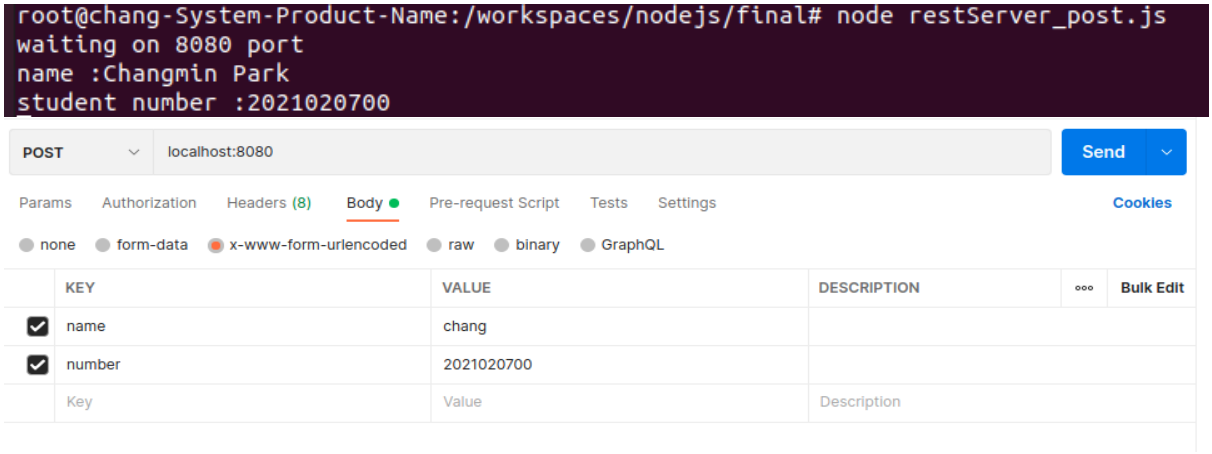
2. about.html

This part should be implemented by inputting your own information using HTML. Attach your name, student number, and a photo of yourself so that the client can check it when the server requests it.



3. restServer_post.js

In this part, you implement the POST method, which is one of the REST APIs, as a node. When a client adds new data to the server, a new resource is registered with the server through POST. Information entered through postman on the built server is transmitted to the server. It should also allow the transmitted information to be printed on the server's console.



4. restServer_put.js, restServer_delete.js (bonus part)

PUT and DELETE are the methods where the client modifies or deletes server information. When the client requests PUT and DELETE, it also prints the server's console. This part is free to implement using AJAX or other methods.

② Description of the functions that are provided is listed below:

1. restServer_get.js

const http = require(“http”)

This task loads the HTTP module using the require function. Create a variable called HTTP, let it refer to the HTTP module, and then call the createServer() method, which is one of the HTTP modules. A callback function for the request is put as an argument, and the callback function is executed whenever a request is received.

const fs = require(“fs”)

The above function calls the "fs" module that handles file input/output. Since the "fs" module is built into node.js, there is no need to install a separate library. "fs" can create/delete directories and write/load data.

2. about.html

<html>

When composing an HTML page, this element wraps the content of the entire page and acts as the root element.

<head>

It acts as a container for everything contained in an HTML page. This includes things like page descriptions, CSS to decorate content, and character set declarations.

<body>

It contains any content you want to show to clients visiting the page, and it can be text, images, etc.

3. restServer_post.js

var queryString = require('querystring')

The querystring module is a module that can manipulate information about the query string appended to the URL in the HTTP GET request. In order for the web server to process the request parameters requested by the client, the string in the query string must be separated into each request parameter again. The querystring module provides methods to handle this task.

③ The assignments (for implementation):

1. restServer_get.js

http.createServer((,))

Create a server object using the createServer() method of the HTTP module and set the port to 8080. When executed, "waiting on 8080 port" is output in the console and "Hello Node.js" is output when "http://localhost:8080/" is input in the address bar in the web.

fs.readFile('./about.html', (,))

Also, when "http://localhost:8080/about" is input, your information is displayed on the web.

2. about.html

<head>

It specifies the character encoding method of the HTML document and sets the IE document mode. Also, set the name of the web page to your own name.

<body>

In <body>, the largest size is output as "Computer Network Project". In the next paragraph, wrap the name and student number and display them in the same size. Finally, attach a photo of yourself to finish.

3. restServer_post.js

http.createServer((,))

Create a server object using the createServer() method of the HTTP module and set the port to 8080. When executed, "waiting on 8080 port" is displayed on the console, but nothing is output on the web. Instead, using postman, when the client enters data, it is passed to the server, and the console prints the received value.

queryString.parse()

A querystring object is created by passing the querystring string to the parse() method of the module. The key value of the query string is entered as a property of the querystring object.

Caveat in your implementation:

1. You need to follow exactly what the descriptions say and implement each function according to the description.
2. You are not allowed to use any library nor codes that are available through the web, friends, and etc.
3. If you cannot make your code when TA ask you to, we look on as cheating

④ **Comments:**

1. You should comment within the function in order to make TAs and yourself what you have been tried to do with codes written. If there is some code that is not understandable and there is no comment on that, points will be deducted.
2. Comments should be all in ENGLISH. Sometimes compile error or running error occurs because the written language is not English. This class is in English so we strongly recommend you to write comments in English. If a student did not write comments in English and some error happened because of that, all the responsibility belongs to the student so functionality points will be deducted.

Project requirements & Scoring Criteria

According to the following specifications, your submitted code will be evaluated. The total point is 120 + bonus 100 points.

1. This is an individual project. Making a copy of the content of your friend's HW, books, google, naver, etc., will get an F for this course. You will not get any partial points for redoing the work. Even if you have different code or variable names, using the same function(s) that is not originated from Node library which is initially installed with your Linux will be regarded as copying. The person who provides the source will also consider as cheating and will face fenalties. Also, asking your friends to do your work will definitely be considered as cheating.
2. If your code does not compile; meaning, there are errors and we cannot fix to grade you work, you will automatically get 0 for Project1 operation part. Please make sure to have compiled your code before turning it in.
3. Codes should have neither error nor warning when compiling: 5 points in total
   A. No errors: 3 points
   B. No warnings: 2 points
4. The following functions should be implemented: 100 points + bonus 40 points
   (a) restServer_get.js (30)
      i. Code with no error (10)
      ii. Console (5)
      iii. Message is printed as instructed (5)
      iv. Comments (10)
   (b) about.html (30)
      i. Code with no error (10)
      ii. Message is printed as instructed (10)
      iii. Comments (10)
   (c) restServer_post.js (40)
      i. Code with no error (10)
      ii. Console (10).
      iii. Postman (10)
      iv. Comments (10)
   (d) restServer_put.js, restServer_delete.js (Bonus: 100)
      i. Code with no error (10)
      ii. Console (10)
      iii. Web (10)
      iv. Comments (10)

5. Submission: 15 points in total
   ① **Due date is 11:59PM, June 13 (Monday)**
      - Submit as early as possible. No excuse for returned email (send failure).
      - No late submission is allowed
   ② Include a README file in your zip file which explains your program: 10 points
   ③ All files should be compressed in one .zip file: 2 points
      - The file name should be: [KECE449_Project1]2023170000_홍길동.zip: 3 points
      - Submit the zip file to ***minpark0120@korea.ac.kr***
      - File should include:
        i. restServer_get.js
        ii. restServer_post.js
        iii. about.html
        iv. README file(It should include short explanation of each function. Korean can be allowed).
        v. restServer_put.js and restServer_delete.js