**CS628 Full-Stack Development II – Backend**

**PE02 - Basic Node and Express - Start a Working Express Server**

Developed by Clark Ngo on December 29, 2020

School of Technology & Computing (STC)

City University of Seattle (CityU)

**Before You Start**

* Some steps are not explained in the tutorial**.** If you are not sure what to do:
  1. Consult the resources listed below.
  2. If you cannot solve the problem after a few tries, ask a TA for help.

**Learning Outcomes**

Students will be able to:

* Learn Basics of Node and Express

**Resources**

* https://www.freecodecamp.org/learn/apis-and-microservices/basic-node-and-express/start-a-working-express-server

**How to Submit**

* **Upload .zip file**
* **Write a 150-word summary to explain your understandings and findings from this lab assignment.**

## **Basic Node and Express - Start a Working Express Server**

In the first two lines of the file myApp.js, you can see how easy it is to create an Express app object. This object has several methods, and you will learn many of them in these challenges. One fundamental method is app.listen(port). It tells your server to listen on a given port, putting it in running state. For testing reasons, we need the app to be running in the background so we added this method in the `server.js` file for you.

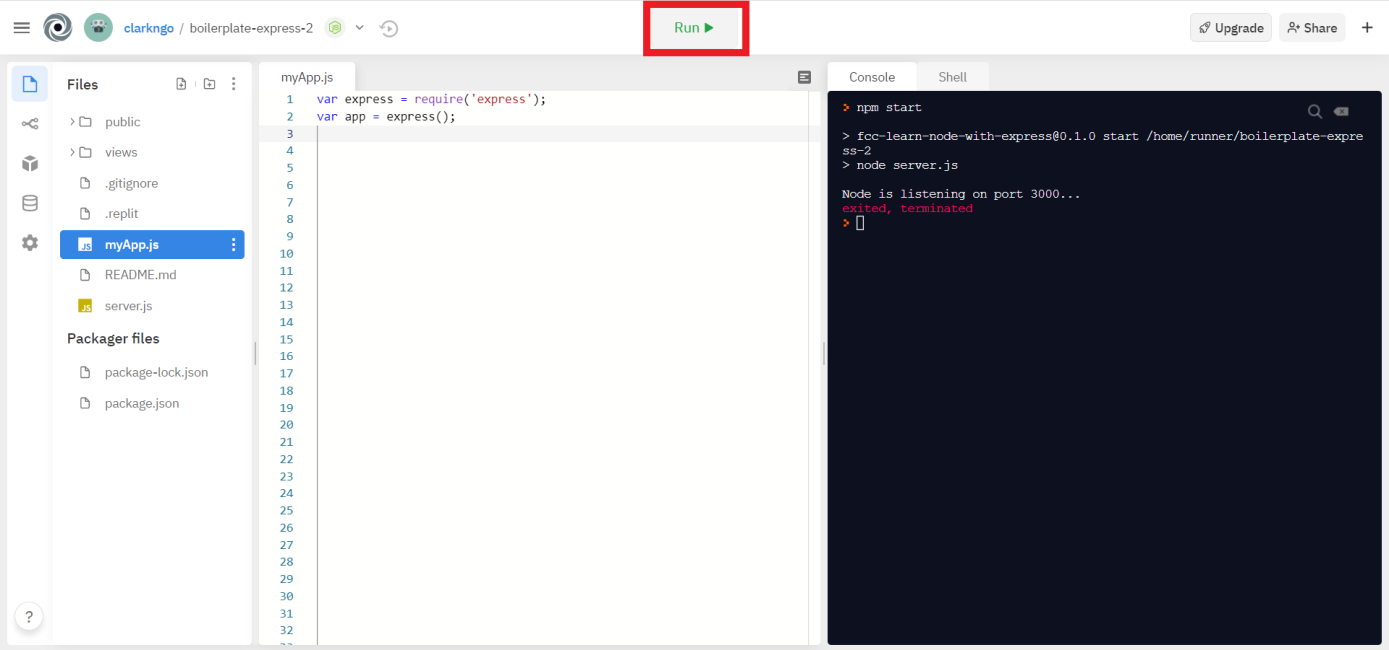
Let’s serve our first string! In Express, routes takes the following structure: app.METHOD(PATH, HANDLER). METHOD is an http method in lowercase. PATH is a relative path on the server (it can be a string, or even a regular expression). HANDLER is a function that Express calls when the route is matched. Handlers take the form function(req, res) {...}, where req is the request object, and res is the response object. For example, the handler

function(req, res) { res.send('Response String');}

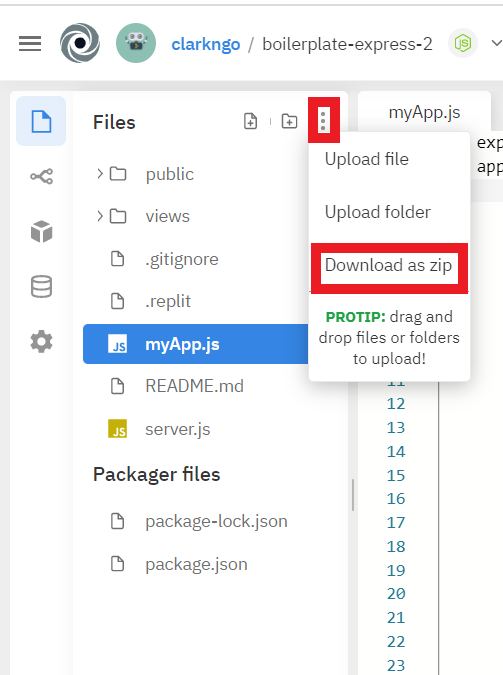
will serve the string 'Response String'.

Use the app.get() method to serve the string "Hello Express" to GET requests matching the / (root) path. Be sure that your code works by looking at the logs, then see the results in the preview if you are using Repl.it.

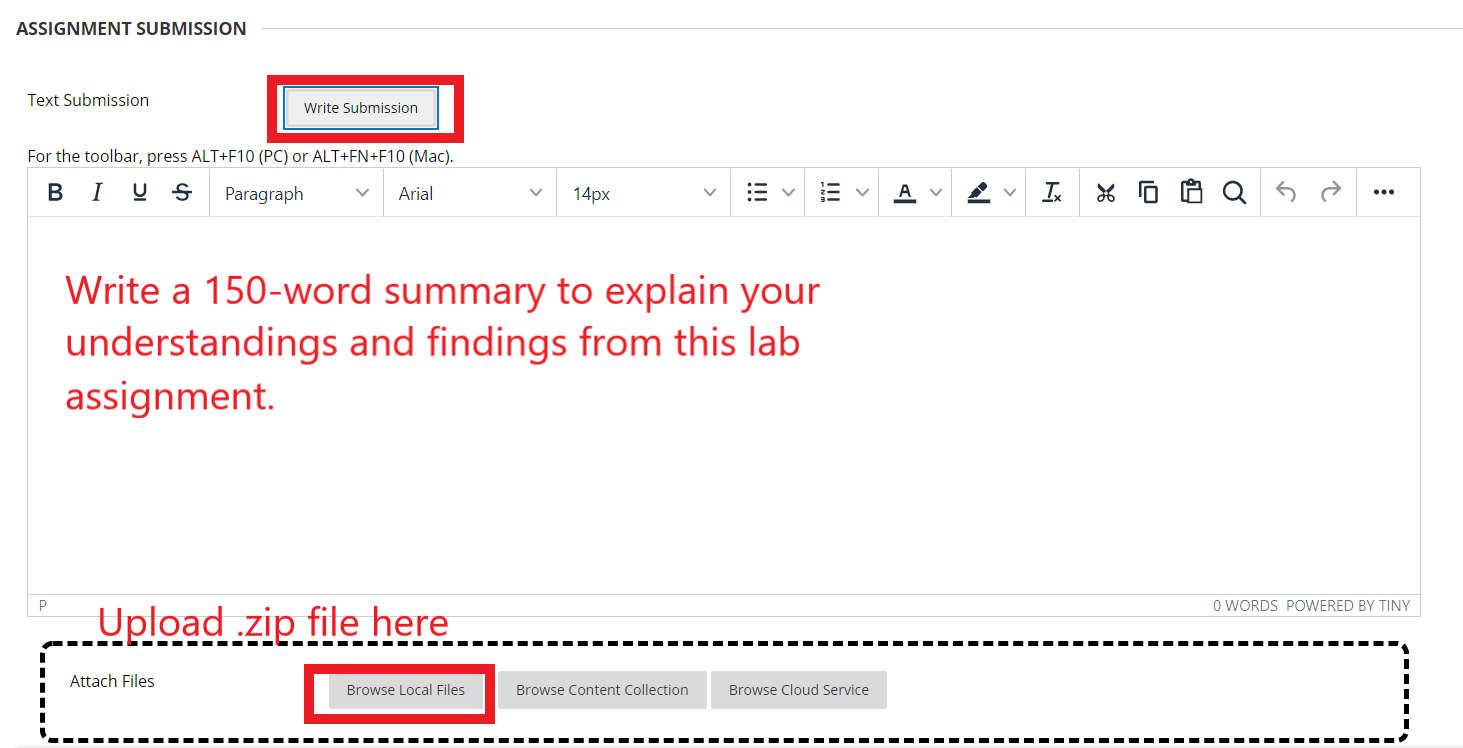
1. Start a new project on Repl.it using [this link](https://repl.it/github/freeCodeCamp/boilerplate-express).
2. Click Run to test your code



1. Download the project as .zip.



1. In BlackBoard, upload the .zip file and write a 150-word summary to explain your understandings and findings from this lab assignment.



Seems very similar to how HTTP request-response programming works in Java with Servlet and JSP technologies. We have methods for get and post requests. And these requests get the parameters request and response. And these methods come up with what response needs to be sent back to the user. Of course, this is an example of how dynamic programming works because instead of sending back a static html page, we can send back different things depending on what the user is requesting.

The express framework gets rid of boilerplate code for us so we can focus on our business logic. We can see the boilerplate code is moved to the server.js file while we work on the myApp.js file. In the myApp.js file, we include the express framework in our program using the “require(‘express’)” line and we create an object of express using the “express()” in the second line. Then we start coding our business logic.