My first Express web server

Welcome to this Lab activity

In this lab activity you will be exploring a very popular web application framework called *Express*. The code will be very similar to your previous web server exercise but, to keep things organised, you will work on another subfolder inside *topic2* for this exercise.

What is Express

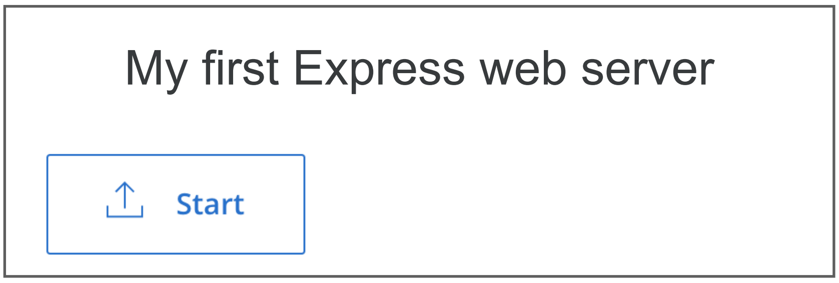
Express is a very popular framework for Node.js designed for building web applications and APIs. Think of it as a module that you can require in your code and use it to create and maintain web servers. It is considered to be the standard server framework for Node.js.

So far you have been using the **http** module to create web servers and, although the code was not really complicated to write, *Express* offers an easier code structure and solutions to do so. Let's get started with it!

Start the Lab environment application

It is simple to launch a lab exercise. You only need to click on the button "Start" below the activity title to enter a lab environment.

Let's explore this lab activity. Go ahead and click on the "Start" button!



Task 1: Create a web server with node.js express module

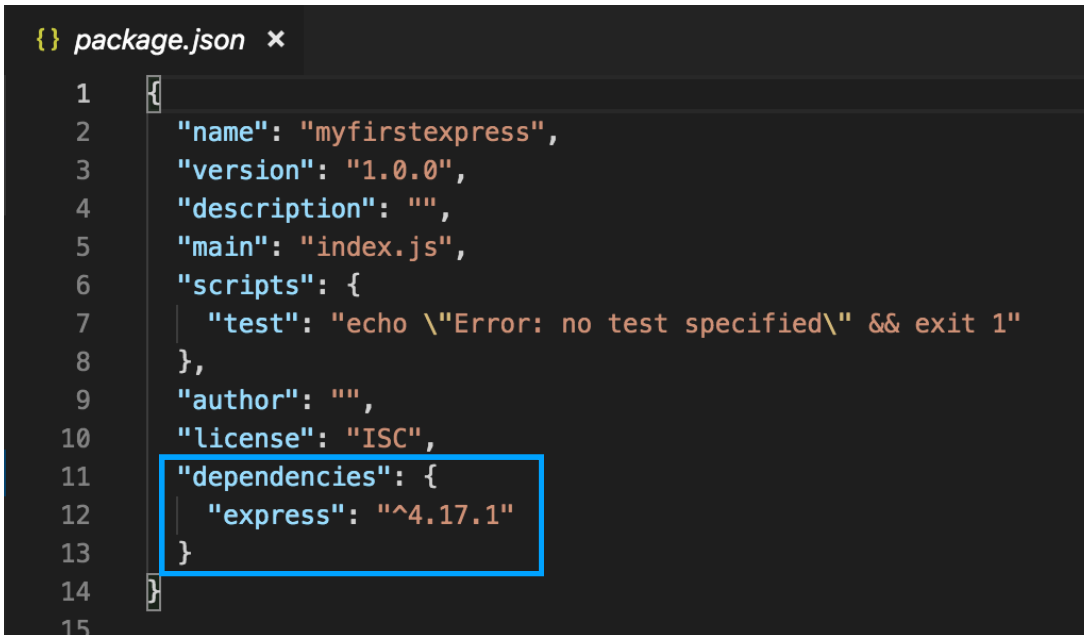
The folder structure has already been partially constructed for you and organised into different topics. For the purpose of this lab, you will be making changes inside the *topic2* folder structure. Let's get started!

Please do not delete or move any existing folders/files inside the lab environment.

Use the Visual Studio Code Terminal and run the following commands:

- **cd topic2/myFirstExpress**: type this command and press *Enter*. This command will change your working directory to be the *myFirstExpress* folder.
- **npm init**: type this command and press *Enter*. This command will set up a new or existing npm package. You can skip all the npm initialisation questions by pressing *Enter.* Once the npm package file has been created, you can view it inside the *mySecondNode* directory (*package.json*). The *package.json* file contains a set of information regarding your current node project.
- **npm install express --save**: type this command and press *Enter*. This command will install the *Express* framework within your working directory so that it will be available for you to use.

When you run the **npm install** command, npm (the node package manager) will search for the package you are trying to install (in this case *Express*) and add it to your project folder. If you take a look inside your *topic02/myFirstExpress* folder, you can now see a completely new folder called *node_modules*. The latter, will store all the packages that you will or have installed in this directory using npm. Furthermore, if you examine the *package.json* file inside the *topic02/myFirstExpress* folder, you will see that *Express* is now listed under the key "*dependencies*". This was achieved by running the **--save** flag on the **npm install** command.



```
{
  "name": "myfirstexpress",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

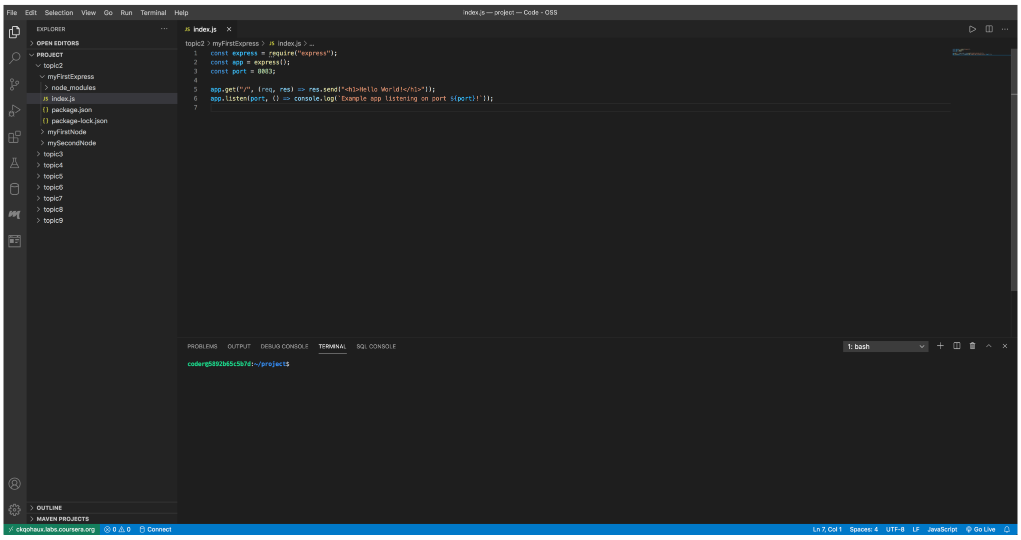
The very next step is to add some code to the *index.js* file located inside the *topic2/myFirstExpress* folder in order to create your web server.

Add the following code to the *index.js* file and remember to save it:

```
const express = require("express");
const app = express();
const port = 8083;

app.get("/", (req, res) => res.send("<h1>Hello World!</h1>"));
app.listen(port, () => console.log(`Example app listening on port ${port}!`));
```

If you have correctly followed all the above steps, your environment should be similar to the one below:


File Edit Selection View Go Run Terminal Help
index.js — project — Code - OSS
EXPLORER
OPEN EDITORS
PROJECT
topic2
myFirstExpress
node_modules
index.js
package.json
package-lock.json
myFirstNode
mySecondNode
topic3
topic4
topic5
topic6
topic7
topic8
topic9
OUTLINE
MAVEN PROJECTS
topic2 > myFirstExpress > index.js > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL SQL CONSOLE
1: bash
ckqohaux.labs.coursera.org
Connect
Ln 7, Col 1 Spaces: 4 UTF-8 LF JavaScript Go Live

```
const express = require("express");
const app = express();
const port = 8083;

app.get("/", (req, res) => res.send("<h1>Hello World!</h1>"));
app.listen(port, () => console.log(`Example app listening on port ${port}!`));
```

```
coder@5892b65c5b7d:~/project$
```

Running the index.js file

Run the *index.js* file with the following Terminal command:

- **node index.js**: type this command and press Enter. The node command, followed by the file name, tells Node.js to execute the content of the file.

The above command will start a web server running on port 8083.

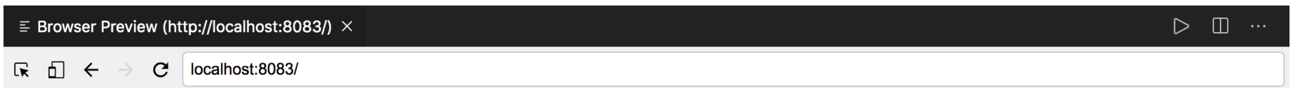
Task 2: Access your server via HTTP

Now that your code is running, serving your application on port 8083, you can access your web page from a browser!

Use the “Browser Preview” plugin to visualise your web application.

If you do not remember how to use the “Browser Preview” plugin, please refer to the “Creating my first Node.js web server” lab instructions.

Type *localhost:8083* on the “Browser Preview” tab and press *Enter* on your keyboard to visualise your web application:



If you have correctly followed all the steps, your web application should show the “Hello World!” message in bold.

End of Section

Congratulations for completing this section. As long as you have saved your work, your files will remain when you close this lab activity so do not worry about losing your data. Next you will explore how to add more routes to your current Express web server and make your web application more interesting.