Lesson 09 Demo 03

Working with Request Handlers

Objective: To illustrate request handling in Express.js by running and validating outputs for different request scenarios

Tools Required: Visual Studio, Node.js, and Express.js

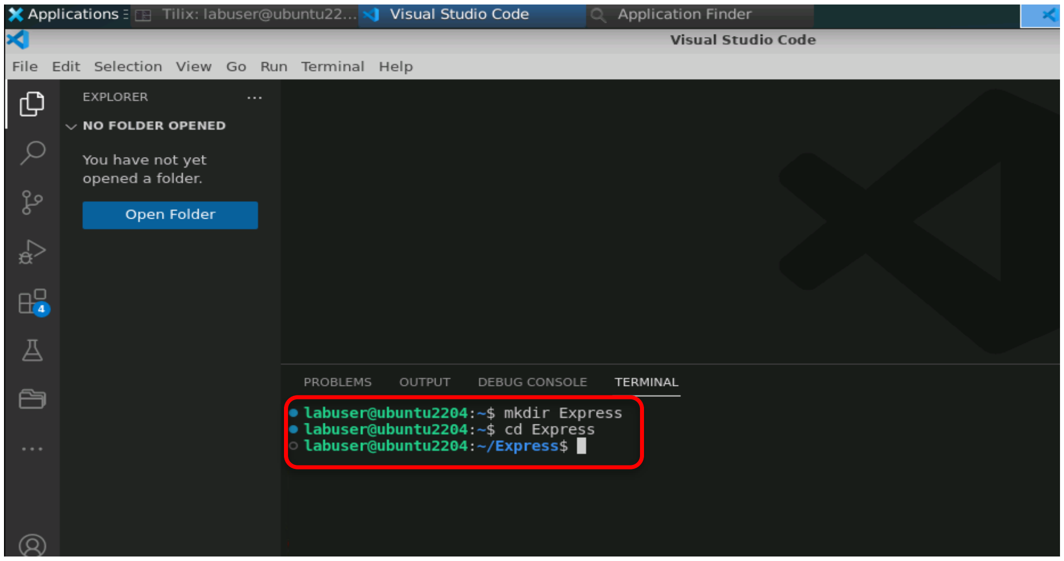
Prerequisites: Knowledge of JavaScript and Node.js

Steps to be followed:

1. Perform routing in Express.js
2. Demonstrate req.params() parameter in Express.js
3. Demonstrate req.header() and req.get() parameters in Express.js

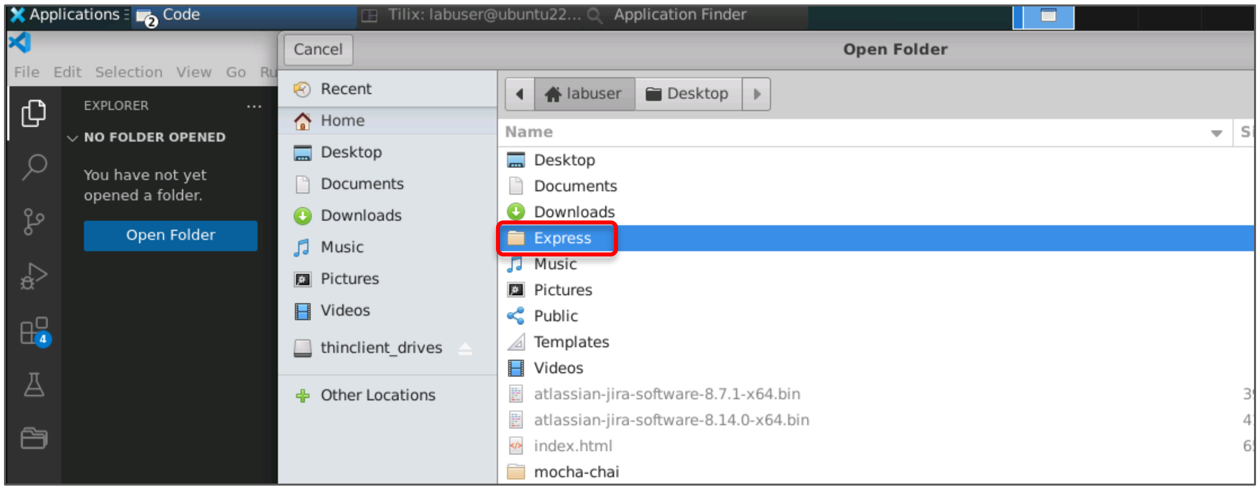
Step 1: Perform routing in Express.js

1.1 Open VS Code, create a folder named **ExpressJS** with the command **mkdir Express**, and change the current working directory using cd Express

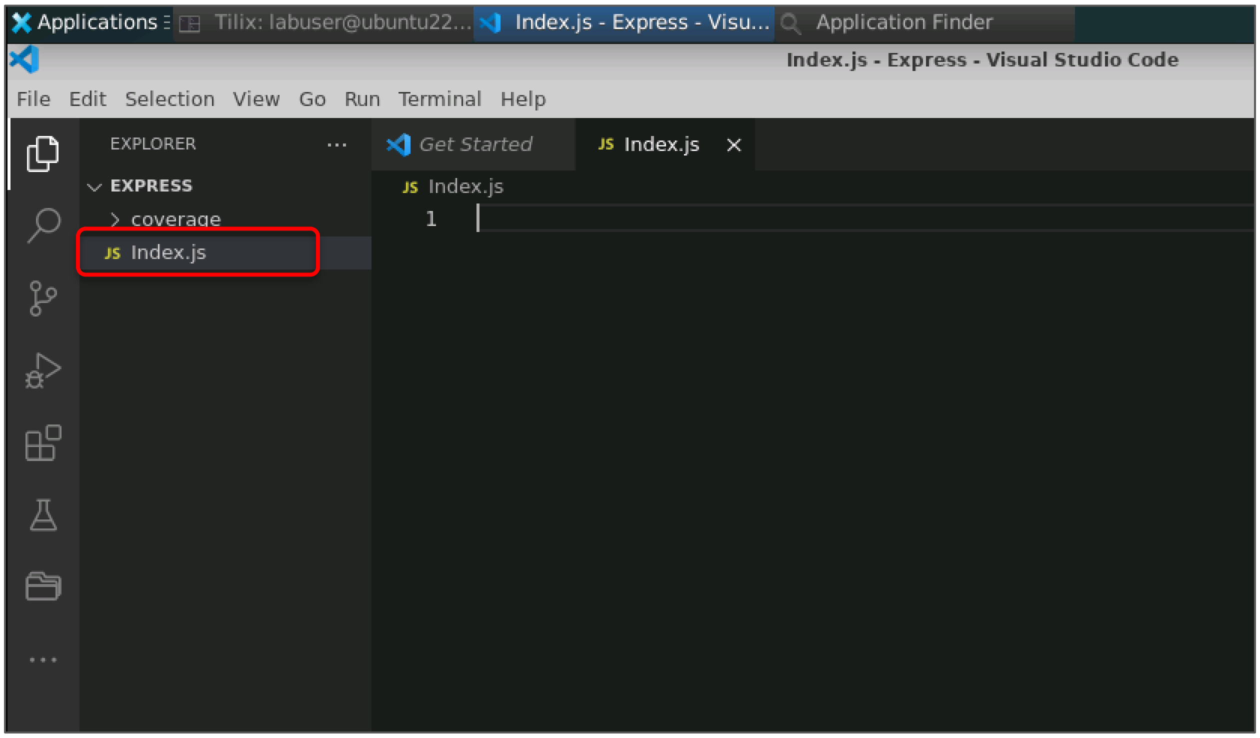

Applications
Tilix: labuser@ubuntu22...
Visual Studio Code
Application Finder
Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
labuser@ubuntu2204:~$ mkdir Express
labuser@ubuntu2204:~$ cd Express
labuser@ubuntu2204:~/Express$
```

1.2 Open the **Express folder** in VS code

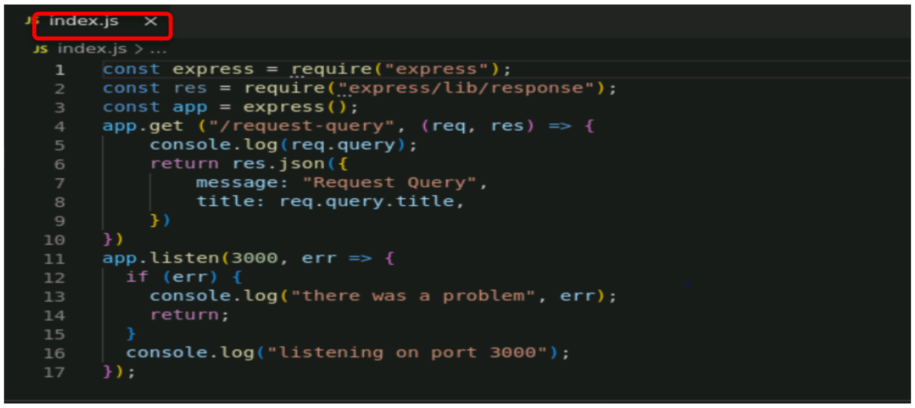


1.3 Create an **index.html** file


Applications
Tilix: labuser@ubuntu22...
Index.js - Express - Visu...
Application Finder
Index.js - Express - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
EXPRESS
coverage
Index.js
Get Started
Index.js
Index.js

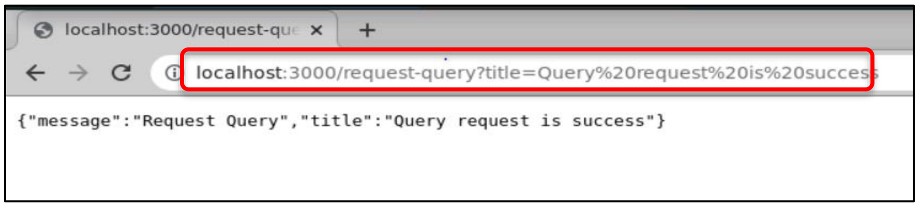
1.3 Add the following code in the **index.js** file:

```
const express = require("express");
const res = require("express/lib/response");
const app = express();

app.get ("/request-query", (req, res) => {
  console.log(req.query);
  return res.json({
    message: "Request Query",
    title: req.query.title,

  })
})

app.listen(3000, err => {
 if (err) {
  console.log("there was a problem", err);
  return;
 }
 console.log("listening on port 3000");
});
```



```
const express = require("express");
const res = require("express/lib/response");
const app = express();
app.get ("/request-query", (req, res) => {
    console.log(req.query);
    return res.json({
        message: "Request Query",
        title: req.query.title,
    })
})
app.listen(3000, err => {
  if (err) {
    console.log("there was a problem", err);
    return;
  }
  console.log("listening on port 3000");
});
```

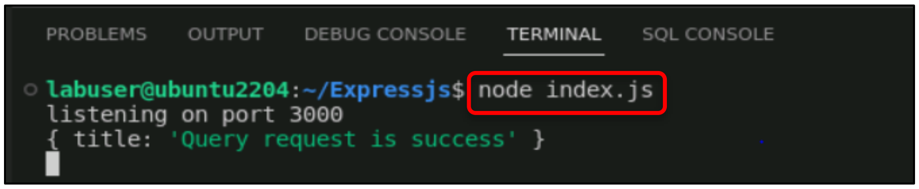
1.4 Run **node index.js,** go to the browser and run http://localhost:3000/request-query?title=Query request is success



localhost:3000/request-query?title=Query%20request%20is%20success

```
{"message":"Request Query","title":"Query request is success"}
```

Note: Here **'?'** shows the query and text return after that passes through the **req.query**

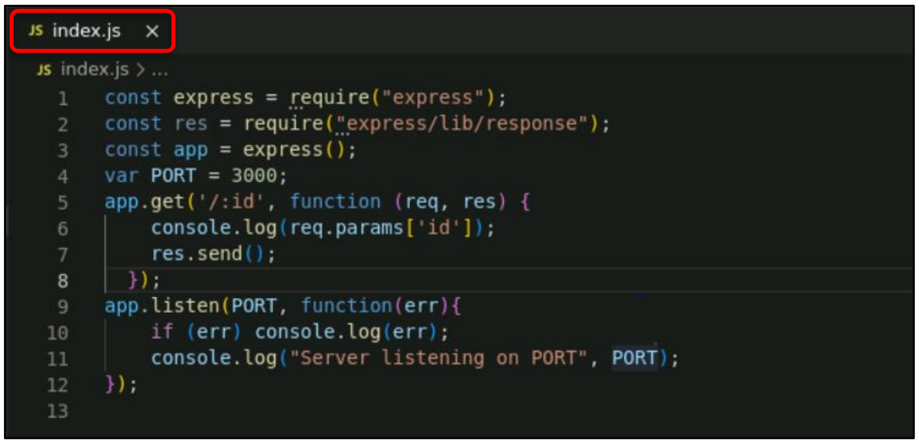


```
labuser@ubuntu2204:~/Expressjs$ node index.js
listening on port 3000
{ title: 'Query request is success' }
```

Step 2: Demonstrate req.params() parameter in Express.js

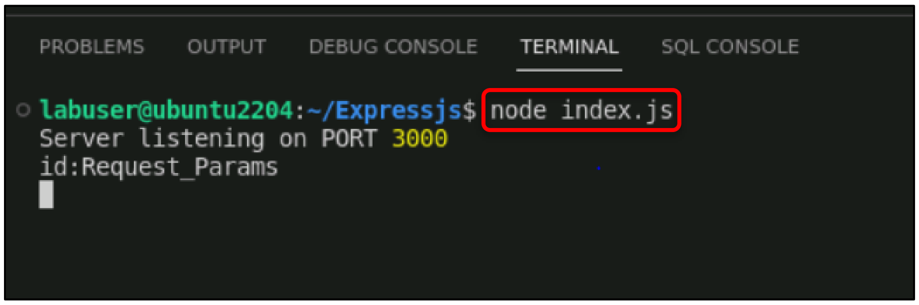
2.1 Write the following code in **index.js** file:

```
const express = require("express");
const res = require("express/lib/response");
const app = express();
var PORT = 3000;

    app.get('/:id', function (req, res) {
       console.log(req.params['id']);
       res.send();
     });

    app.listen(PORT, function(err){
       if (err) console.log(err);
       console.log("Server listening on PORT", PORT);
    });
```



```
const express = require("express");
const res = require("express/lib/response");
const app = express();
var PORT = 3000;
app.get('/:id', function (req, res) {
    console.log(req.params['id']);
    res.send();
  });
app.listen(PORT, function(err){
    if (err) console.log(err);
    console.log("Server listening on PORT", PORT);
});
```

2.2 Run **node index.js**, go to the browser run **http://localhost:3000/id:Request_Params**, and Check the terminal for output

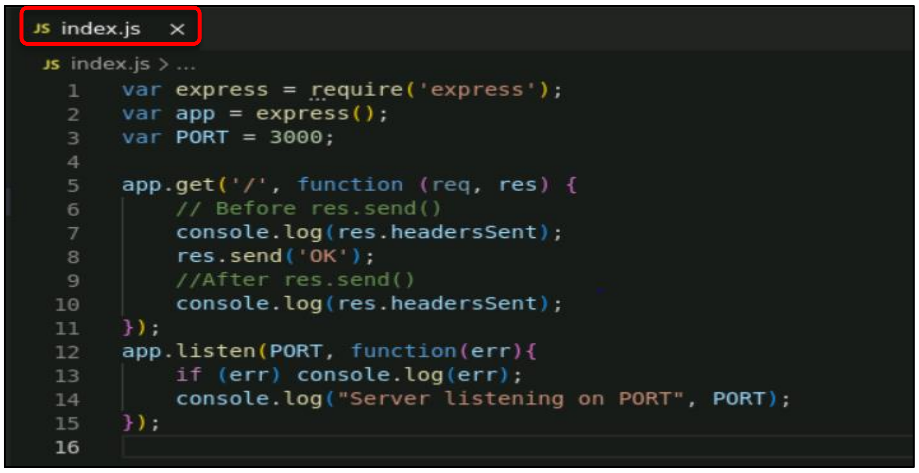


```
labuser@ubuntu2204:~/Expressjs$ node index.js
Server listening on PORT 3000
id:Request_Params
```

Step 3: Demonstrate req.header() and req.get() parameters in Express.js

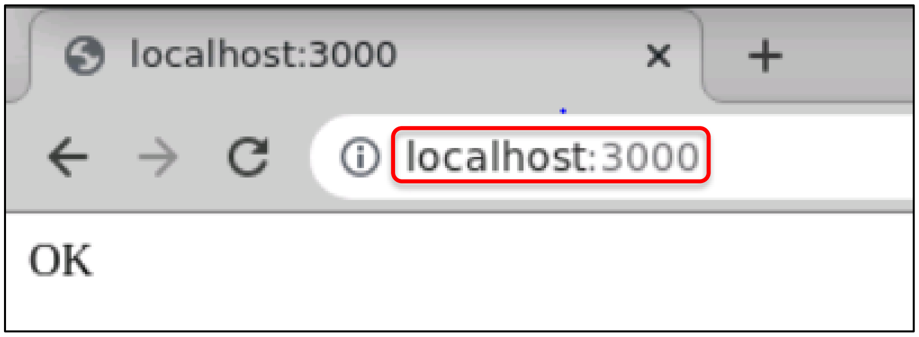
3.1 Replace the following code in the **index.js** file:

```
var express = require('express');
var app = express();
var PORT = 3000;

app.get('/', function (req, res) {
   // Before res.send()
   console.log(res.headersSent);
   res.send('OK');
   //After res.send()
   console.log(res.headersSent);
});
app.listen(PORT, function(err){
   if (err) console.log(err);
   console.log("Server listening on PORT", PORT);
});
```

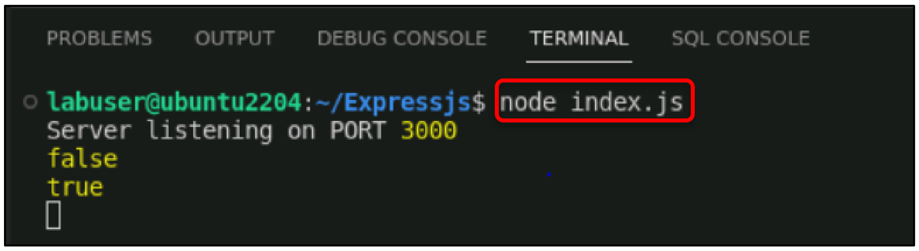


```
var express = require('express');
var app = express();
var PORT = 3000;

app.get('/', function (req, res) {
    // Before res.send()
    console.log(res.headersSent);
    res.send('OK');
    //After res.send()
    console.log(res.headersSent);
});
app.listen(PORT, function(err){
    if (err) console.log(err);
    console.log("Server listening on PORT", PORT);
});
```

3.2 Run node index.js, access the browser at **http://localhost:3000/,** and inspect the terminal for the output



```
OK
```



```
labuser@ubuntu2204:~/Expressjs$ node index.js
Server listening on PORT 3000
false
true
```

Note:

The following are the other attributes and methods:

- req.headers
- req.url
- req.ip
- req.hostname
- req.method --get
- req.protocol --http /https
- req.path --just the path part of the url
- req.subdomains --test.sales.example.com ['test','sales]
- req.query --querystring
- req.params --/user/72 --/product/234234

By following these steps, you have successfully implemented effective request handling in Express.js with validated functionality for diverse request scenarios.