Lesson 10 Demo 02

Demonstrating Error Handling Commands

Objective: To demonstrate the use of error handling commands for routing, local variables, rendering views, and server setup in Express.js

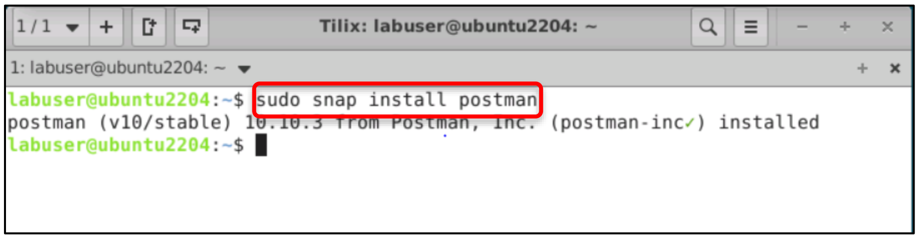
Tools Required: Ubuntu and Visual Studio

Prerequisites: Knowledge of JavaScript and Node.js

Steps to be followed:
1. Install Postman in the system for checks
2. Use app.routes in Express.js
3. Use app.locals in Express.js
4. Use app.render() in Express.js
5. Use app.listen() in Express.js

Step 1: Install Postman in the system for checks

1.1 Check the response method using the Postman application and run the following command in the system terminal to install Postman:
sudo snap install postman

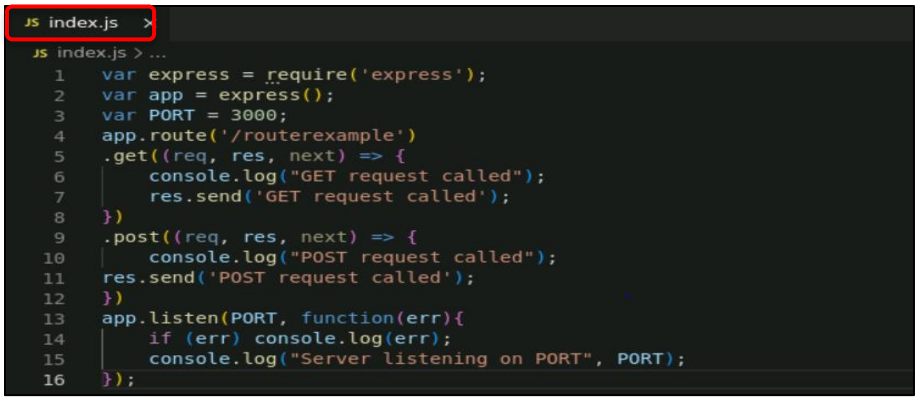


```
labuser@ubuntu2204:~$ sudo snap install postman
postman (v10/stable) 10.10.3 from Postman, Inc. (postman-inc✓) installed
labuser@ubuntu2204:~$
```

Step 2: Use app.routes() in Express.js

2.1 Open the Expressjs folder in VS Code and write the following code in the **index.js** file:
var express = require('express');
var app = express();
var PORT = 3000;

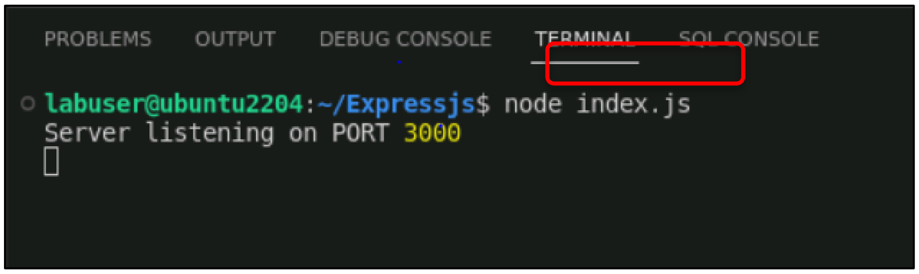
```
app.route('/routerexample')
.get((req, res, next) => {
   console.log("GET request called");
   res.send('GET request called');
})
.post((req, res, next) => {
   console.log("POST request called");
res.send('POST request called');
})

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



index.js

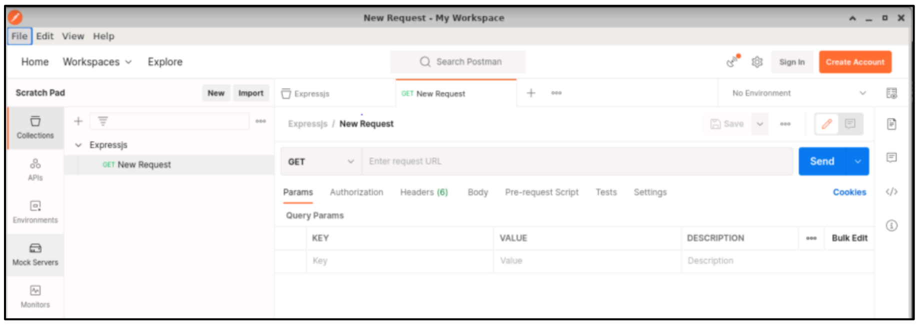
```
var express = require('express');
var app = express();
var PORT = 3000;
app.route('/routerexample')
.get((req, res, next) => {
    console.log("GET request called");
    res.send('GET request called');
})
.post((req, res, next) => {
    console.log("POST request called");
res.send('POST request called');
})
app.listen(PORT, function(err){
    if (err) console.log(err);
    console.log("Server listening on PORT", PORT);
});
```

2.2 Run the **node index.js** command in the terminal

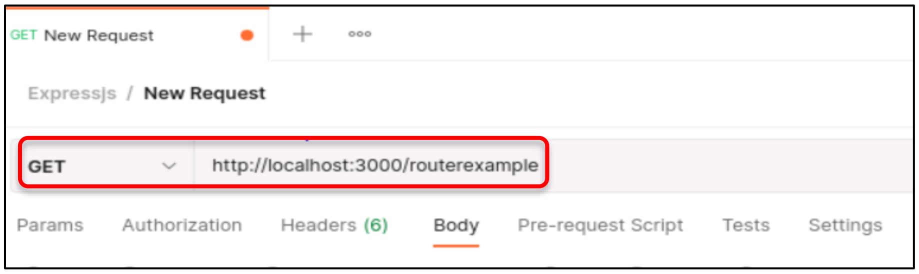


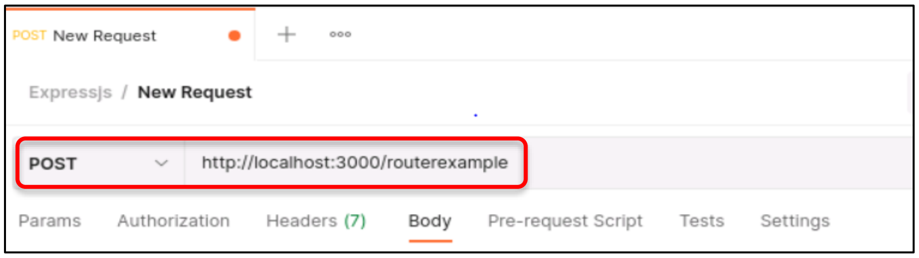
```
labuser@ubuntu2204:~/Expressjs$ node index.js
Server listening on PORT 3000
```

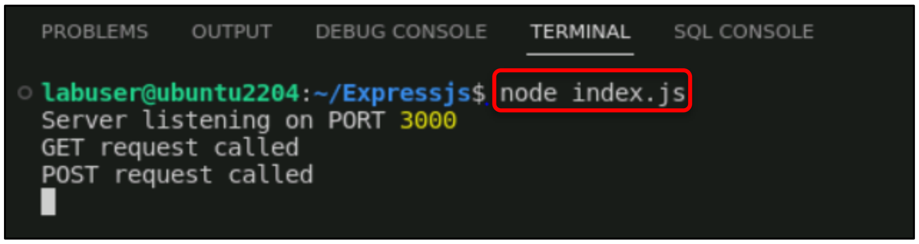
2.3 Open Postman, create an account, or skip it. In the Postman workspace, run a collection of Express.js requests to check function responses, and subsequently add a new request to the collection



2.4 Make the GET and POST requests to **http://localhost:3000/**, and check the output in the VS Code terminal, where the program is executed


GET New Request
Expressjs / New Request
GET http://localhost:3000/routerexample
Params Authorization Headers (6) Body Pre-request Script Tests Settings


POST New Request
Expressjs / New Request
POST http://localhost:3000/routerexample
Params Authorization Headers (7) Body Pre-request Script Tests Settings

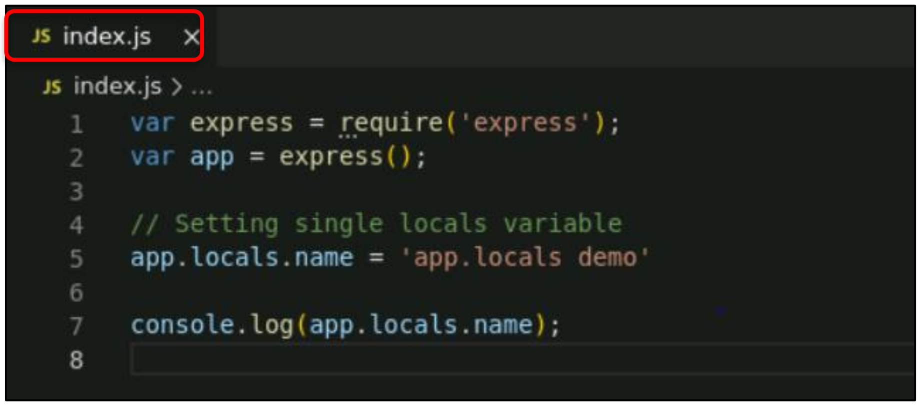


```
labuser@ubuntu2204:~/Expressjs$ node index.js
Server listening on PORT 3000
GET request called
POST request called
```

Step 3: Use App.locals() in Express.js

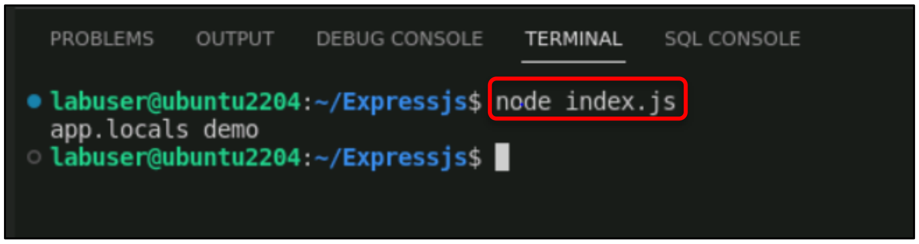
3.1 Add the following code in the **index.js** file:

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

// Setting single locals variable
app.locals.name = 'app.locals demo'

console.log(app.locals.name);
```



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

// Setting single locals variable
app.locals.name = 'app.locals demo'

console.log(app.locals.name);
```

3.2 Run the **node index.js** command in the terminal



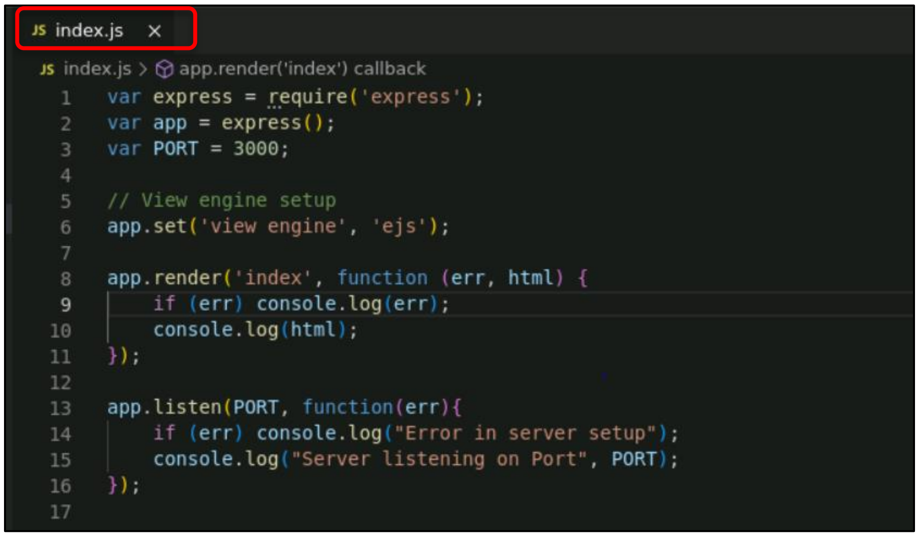
```
labuser@ubuntu2204:~/Expressjs$ node index.js
app.locals demo
labuser@ubuntu2204:~/Expressjs$ 
```

Step 4: Use app.render() in Express.js

4.1 Add the following code to the **index.js** file:

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

// View engine setup
app.set('view engine', 'ejs');

app.render('index', function (err, html) {
  if (err) console.log(err);
```

```
    console.log(html);
});

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

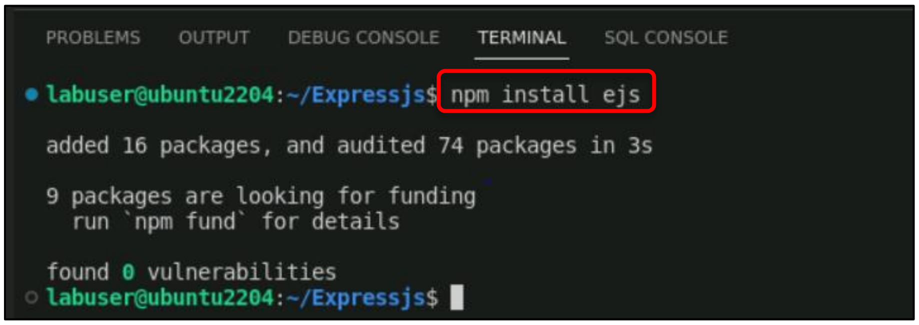


index.js

index.js > app.render('index') callback

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

// View engine setup
app.set('view engine', 'ejs');

app.render('index', function (err, html) {
    if (err) console.log(err);
    console.log(html);
});

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

4.2 Run the following command in the terminal to add **ejs** to the project:
npm install ejs

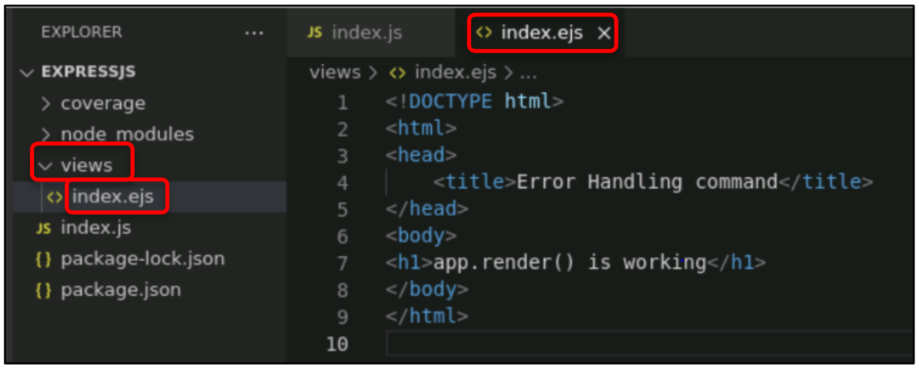


```
labuser@ubuntu2204:~/Expressjs$ npm install ejs

added 16 packages, and audited 74 packages in 3s

9 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
labuser@ubuntu2204:~/Expressjs$
```

4.3 Create a folder named **views,** and in that folder, create an **index.ejs** file and add the following code to that file:

```
<!DOCTYPE html>
<html>
<head>
        <title>Error Handling command</title>
</head>
<body>
<h1>app.render() is working</h1>
</body>
```

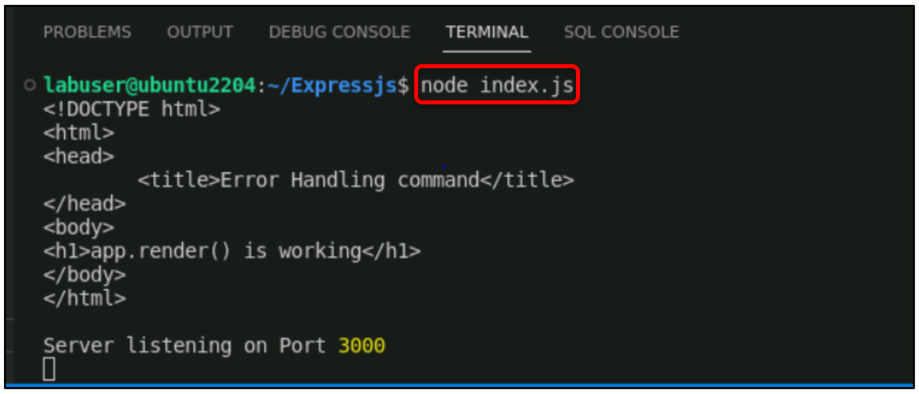
</html>



```
EXPLORER
EXPRESSJS
> coverage
> node_modules
v views
  index.ejs
index.js
package-lock.json
package.json
```

```
views > index.ejs > ...
<!DOCTYPE html>
<html>
<head>
    <title>Error Handling command</title>
</head>
<body>
<h1>app.render() is working</h1>
</body>
</html>
```

4.4 Run the **node index.js** command in the terminal

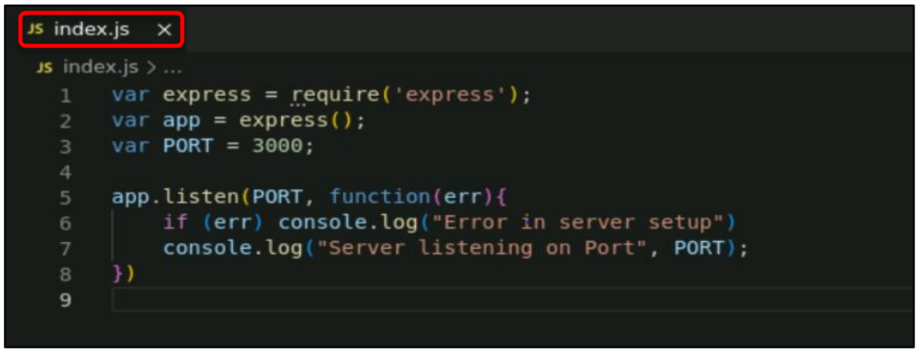


```
labuser@ubuntu2204:~/Expressjs$ node index.js
<!DOCTYPE html>
<html>
<head>
        <title>Error Handling command</title>
</head>
<body>
<h1>app.render() is working</h1>
</body>
</html>

Server listening on Port 3000
```

Step 5: Use app.listen() in Express.js

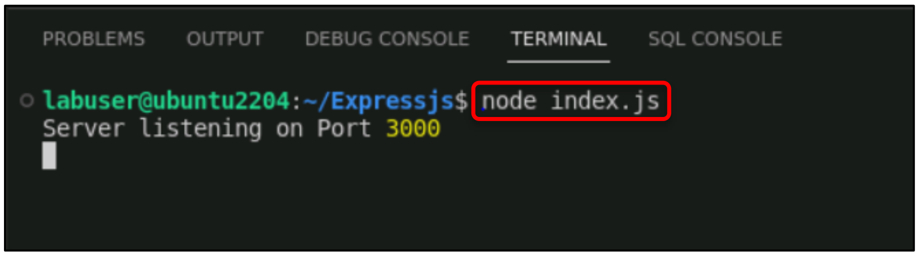
5.1 Open the Express.js folder created in VS code and write the below code in the **index.js** file:

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

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


index.js

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

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

5.2 Run the **node index.js** command in the terminal



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

By following these steps, you have successfully implemented and understood error-handling commands for routing, local variables, view rendering, and server setup.