National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS-213: Advanced Programming

Class: BSCS 7AB

Lab 07: Express JS

Date: 17 October, 2019

Time: 10:00-01:00pm & 02:00-05:00pm

Umaid Zaffar

237560

BSCS 7B

Instructor: Dr. Sidra Sultana

Lab Engineer: Ms. Ayesha Asif

Lab 07: Express JS

Introduction

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications.

Objectives

This lab will get you familiar with the Node Express JS environment.

Tools/Software Requirement

Node.js, Express Js, Notepad

Description

Installing Express

Firstly, install the Express framework globally using NPM so that it can be used to create a web application using node terminal.

$ npm install express --save

The above command saves the installation locally in the **node_modules** directory and creates a directory express inside node_modules. You should install the following important modules along with express −

- **body-parser** − This is a node.js middleware for handling JSON, Raw, Text and URL encoded form data.
- **cookie-parser** − Parse Cookie header and populate req.cookies with an object keyed by the cookie names.
- **multer** − This is a node.js middleware for handling multipart/form-data.

$ npm install body-parser --save
$ npm install cookie-parser --save
$ npm install multer --save

Hello world Example

Following is a very basic Express app which starts a server and listens on port 8081 for connection. This app responds with **Hello World!** for requests to the homepage. For every other path, it will respond with a **404 Not Found.**

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

```
app.get('/', function (req, res) {
  res.send('Hello World');
})

var server = app.listen(8081, function () {
  var host = server.address().address
  var port = server.address().port

  console.log("Example app listening at http://%s:%s", host, port)
})
```

Save the above code in a file named server.js and run it with the following command.

$ node server.js

You will see the following output −

Example app listening at http://0.0.0.0:8081

Open http://127.0.0.1:8081/ in any browser to see the following result.

Express JS Templating: Pug

Pug is a templating engine for Express. Templating engines are used to remove the cluttering of our server code with HTML, concatenating strings wildly to existing HTML templates. Pug is a very powerful templating engine which has a variety of features including **filters, includes, inheritance, interpolation**, etc. There is a lot of ground to cover on this.

To use Pug with Express, we need to install it,

npm install --save pug

Now that Pug is installed, set it as the templating engine for your app. You **don't** need to 'require' it. Add the following code to your **index.js** file.

```
app.set('view engine', 'pug');
app.set('views','./views');
```

Now create a new directory called views. Inside that create a file called **first_view.pug**, and enter the following data in it.

```
doctype html
html
  head
    title = "Hello Pug"
  body
    p.greetings#people Hello World!
```

To run this page, add the following route to your app −

```
app.get('/first_template', function(req, res){
  res.render('first_view');
});
```

You will get the output as − **Hello World!** Pug converts this very simple looking markup to html. We don't need to keep track of closing our tags, no need to use class and id keywords, rather use '.' and '#' to define them. The above code first gets converted to −

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello Pug</title>
  </head>

  <body>
    <p class = "greetings" id = "people">Hello World!</p>
  </body>
</html>
```

Pug is capable of doing much more than simplifying HTML markup.

Lab Tasks

Task 1: Create Index.html file in the root folder of your application and write the HTML FORM POST method code in it. Modify server.js to handle home page requests as well as the input sent by the HTML form.

> **Hint:** To handle HTTP POST request in Express.js version 4 and above, you need to install middleware module called body-parser. The middleware was a part of Express.js earlier but now you have to install it separately. This body-parser module parses the JSON, buffer, string and url encoded data submitted using HTTP POST request. Install body-parser using NPM as shown below.

Task 2: Create a file uploader form in an html file. This form has method attribute set to POST and enctype attribute is set to multipart/form-data. Modify server.js to handle home page requests as well as file upload.

Task 3: By using the Pug templating engine create an HTML registration form.

Solution

Task Code:

App.js:

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

var app = express()

var bodyParser = require('body-parser')

var multer = require('multer');

var upload = multer();

var path = require('path');



app.use(bodyParser.json());

app.use(bodyParser.urlencoded({extended: true}));

app.use(upload.array());
```

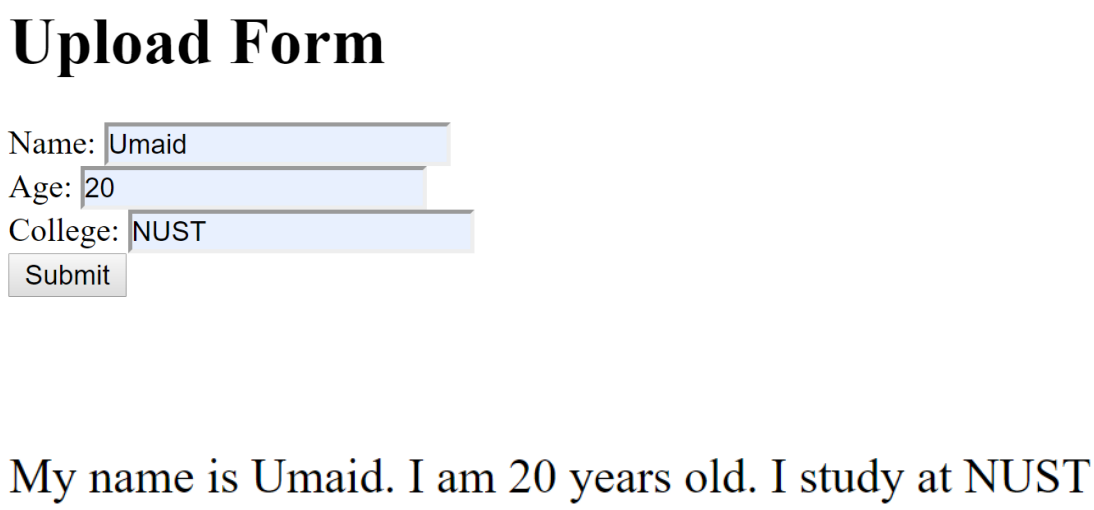
```
app.get('/', function(req, res){
        res.sendFile(path.join(__dirname, './index.html'));
})

app.post('/', function(req, res){
        console.log(req.body);
                res.send('My name is ' + req.body.name + ". I am " + req.body.age + " years old. I
study at " + req.body.colg);
})

app.listen(3000)
```

Index.html:

```
<!DOCTYPE html>
<html>
<head>
        <title>My Form</title>
</head>
<body>
<h1>Upload Form</h1>
```

```
<form method="post" action="/" enctype="multipart/form-data">
        Name: <input type="text" name="name"><br>
        Age: <input type="text" name="age"><br>
        College: <input type="text" name="colg"><br>

        <input type="submit" name="submit">
</form>
</body>
</html>
```

Task Output Screenshot:

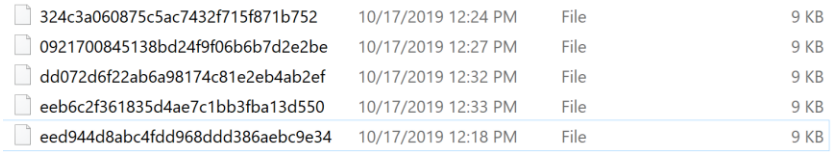


Task Code:

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

```
var bodyParser = require('body-parser')

var multer = require('multer');

var upload = multer({dest: "upload/"});

var path = require('path');




app.get('/', function(req, res){

        res.sendFile(path.join(__dirname, './index.html'));

})


app.post('/uploadfile',upload.single('myfile'), function(req, res){

        var file = req.myfile;

        console.log(file)

        res.send('My name is ' + req.body.name + ". I am " + req.body.age + " years old. I study at " +

req.body.colg);

})


app.listen(3000)
```

Task Output Screenshot:



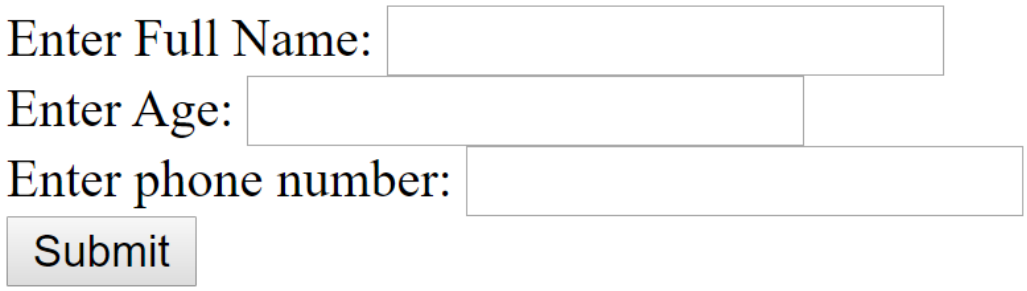

```
{ fieldname: 'file',
  originalname: 'lab3.html',
  encoding: '7bit',
  mimetype: 'text/html',
  destination: 'upload/',
  filename: 'eeb6c2f361835d4ae7c1bb3fba13d550',
  path: 'upload\\eeb6c2f361835d4ae7c1bb3fba13d550',
  size: 8638 }
```

Task Code:

```
doctype html

html

        head

                title = "Html form"

        body

                form(action = "/", method = "POST")

                        div

                                label(for = "fullname") Enter Full Name:

                                input(type= "text" name = "say")

                                br

                        div

                                label(for = "age") Enter Age:
```

```
                input(type= "text" name = "age")
                br
            div
                label(for = "college") Enter phone number:
                input(type = "text" name = "college")
                br
            button(type = "submit") Submit
```

Task Output Screenshot:



Deliverables

Compile a single word document by filling in the solution part and submit this Word file on LMS. This lab grading policy is as follows: The lab is graded between 0 to 10 marks. The submitted solution can get a maximum of 5 marks. At the end of each lab or in the next lab, there will be a viva/quiz related to the tasks. You must show the implementation of the tasks in the designing tool, along with your complete Word document to get your work graded. You must also submit this Word document on the LMS. In case of any problems with submissions on LMS, submit your Lab assignments by emailing it to Ms. Ayesha Asif: ayesha.asif@seecs.edu.pk.