# **MEAN Stack Deployment on Linux**

You have done a great job with successful completion of Project 1. Here you will learn how to implement other web stacks.

## **Instructions On How To Submit Your Work For Review And Feedback**

### **Follow the steps below to submit your work for review**

1. Create a word document on Google Drive
2. Write down the steps you took to accomplish your work
3. Make screenshots wherever possible.
   1. On Windows you can use Snipping Tool
   2. On Mac you can use Lightshot
4. Name your document according to the Project work
5. Click the Submit your project for review in DAREY.IO project dashboard, and paste the link to your project, and submit.

## **Project Prerequisites**

In order to complete this project, you will need to have a fresh install of Ubuntu 20.04 in your Virtual Box. Do Not use the same server from Project 1.

1. [Here is how to configure Virtual Box](https://www.youtube.com/watch?v=wql6adU2JeE&list=PLtPuNR8I4TvkwU7Zu0l0G_uwtSUXLckvh&index=3)
2. [Here is how to install and configure Ubuntu 20.04](https://www.youtube.com/watch?v=Lw-VAYWJumo&list=PLtPuNR8I4TvkwU7Zu0l0G_uwtSUXLckvh&index=4)

### **MEAN Stack is a combination of**

* MongoDB (Document database) - Stores and retrieve data.
* Express (Back-end application framework) - Makes requests to Database and return a response
* Angular (Front-end application framework) - Handles Client and Server Requests
* Node.js (JavaScript runtime environment) - Accept requests and display results to end user

## **Lets Get Started!**

### **Step1: Install NodeJs**

Node.js is a JavaScript runtime built on Chrome’s V8 JavaScript engine. Node.js is used in this tutorial to set up the Express routes and AngularJS controllers.

$ sudo apt-get install -y nodejs

### **Step 2: Install MongoDB**

MongoDB stores data in flexible, JSON-like documents. Fields in a database can vary from document to document and data structure can be changed over time. For our example application, we are adding book records to MongoDB that contain book name, isbn number, author, and number of pages.

sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 0C49F3730359A14518585931BC711F9BA15703C6

echo "deb [ arch=amd64 ] https://repo.mongodb.org/apt/ubuntu trusty/mongodb-org/3.4 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-3.4.list

### **Next update the package manager**

sudo apt-get update

Install MongoDB

sudo apt-get install -y mongodb

Start The server

sudo service mongodb start

We also need to install the body-parser package to help us process the JSON passed in requests to the server. Install the npm package manager. sudo apt-get install npm

### **Install the body-parser package**

sudo npm install body-parser

Create a folder named Books

mkdir Books

Add a file to it named server.js

touch server.js

Copy and paste the web server code below into the server.js file.

var express = require('express');

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

var app = express();

app.use(express.static(\_\_dirname + '/public'));

app.use(bodyParser.json());

require('./apps/routes')(app);

app.set('port', 3300);

app.listen(app.get('port'), function() {

console.log('Server up: http://localhost:' + app.get('port'));

});

#### **Step 3: Install Express and set up routes to the server**

Express is a minimal and flexible Node.js web application framework that provides features for web and mobile applications. Express is used in this tutorial to pass book information to and from our MongoDB database. Mongoose provides a straight-forward, schema-based solution to model your application data. Mongoose is used in this tutorial to provide a book schema for the database. sudo npm install express mongoose In the Books folder, create a folder named apps

mkdir apps

Move into apps folder

cd apps

Add a file named routes.js

touch routes.js

Copy and paste the code below into routes.js

var Book = require('./models/book');

module.exports = function(app) {

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

Book.find({}, function(err, result) {

**if** ( err ) throw err;

res.json(result);

});

});

app.post('/book', function(req, res) {

var book = new Book( {

name:req.body.name,

isbn:req.body.isbn,

author:req.body.author,

pages:req.body.pages

});

book.save(function(err, result) {

**if** ( err ) throw err;

res.json( {

message:"Successfully added book",

book:result

});

});

});

app.delete("/book/:isbn", function(req, res) {

Book.findOneAndRemove(req.query, function(err, result) {

**if** ( err ) throw err;

res.json( {

message: "Successfully deleted the book",

book: result

});

});

});

var path = require('path');

app.get('\*', function(req, res) {

res.sendfile(path.join(\_\_dirname + '/public', 'index.html'));

});

};

In the apps folder, create a folder named models

mkdir models

Add a file named book.js

touch book.js

Copy and paste the code below into book.js

var mongoose = require('mongoose');

var dbHost = 'mongodb://localhost:27017/test';

mongoose.connect(dbHost);

mongoose.connection;

mongoose.set('debug', true);

var bookSchema = mongoose.Schema( {

name: String,

isbn: {type: String, index: true},

author: String,

pages: Number

});

var Book = mongoose.model('Book', bookSchema);

module.exports = mongoose.model('Book', bookSchema);

### **Access the routes with AngularJS**

[AngularJS](https://angularjs.org/) provides a web framework for creating dynamic views in your web applications. In this tutorial, we use AngularJS to connect our web page with Express and perform actions on our book database.

Change the directory back up to Books

cd ../..

Create a folder named public

mkdir public

Move into public

cd public

Add a file named script.js

touch script.js

Copy and paste the Code below (controller configuration defined) into the script.js file.

var app = angular.module('myApp', []);

app.controller('myCtrl', function($scope, $http) {

$http( {

method: 'GET',

url: '/book'

}).then(function successCallback(response) {

$scope.books = response.data;

}, function errorCallback(response) {

console.log('Error: ' + response);

});

$scope.del\_book = function(book) {

$http( {

method: 'DELETE',

url: '/book/:isbn',

params: {'isbn': book.isbn}

}).then(function successCallback(response) {

console.log(response);

}, function errorCallback(response) {

console.log('Error: ' + response);

});

};

$scope.add\_book = function() {

var body = '{ "name": "' + $scope.Name +

'", "isbn": "' + $scope.Isbn +

'", "author": "' + $scope.Author +

'", "pages": "' + $scope.Pages + '" }';

$http({

method: 'POST',

url: '/book',

data: body

}).then(function successCallback(response) {

console.log(response);

}, function errorCallback(response) {

console.log('Error: ' + response);

});

};

});

In the public folder, create a file named index.html

touch index.html

Cpoy and paste the code below into index.html file.

<!doctype html>

<html ng-app="myApp" ng-controller="myCtrl">

<head>

<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.6.4/angular.min.js"></script>

<script src="script.js"></script>

</head>

<body>

<div>

<table>

<tr>

<td>Name:</td>

<td><input type="text" ng-model="Name"></td>

</tr>

<tr>

<td>Isbn:</td>

<td><input type="text" ng-model="Isbn"></td>

</tr>

<tr>

<td>Author:</td>

<td><input type="text" ng-model="Author"></td>

</tr>

<tr>

<td>Pages:</td>

<td><input type="number" ng-model="Pages"></td>

</tr>

</table>

<button ng-click="add\_book()">Add</button>

</div>

<hr>

<div>

<table>

<tr>

<th>Name</th>

<th>Isbn</th>

<th>Author</th>

<th>Pages</th>

</tr>

<tr ng-repeat="book in books">

<td><input type="button" value="Delete" data-ng-click="del\_book(book)"></td>

<td>{{book.name}}</td>

<td>{{book.isbn}}</td>

<td>{{book.author}}</td>

<td>{{book.pages}}</td>

</tr>

</table>

</div>

</body>

</html>

Change the directory back up to Books

cd ..

Start the server by running this command:

node server.js

Open a web browser to the address http://localhost:3300

_images/project4end.JPG

https://docs.google.com/document/d/1cgEuMptw839urfOmzmrj24KlHyGSmjvh7CnEsYTSp\_Q/edit

[Previous](https://progressive-pbl.darey.io/en/latest/project3.html)