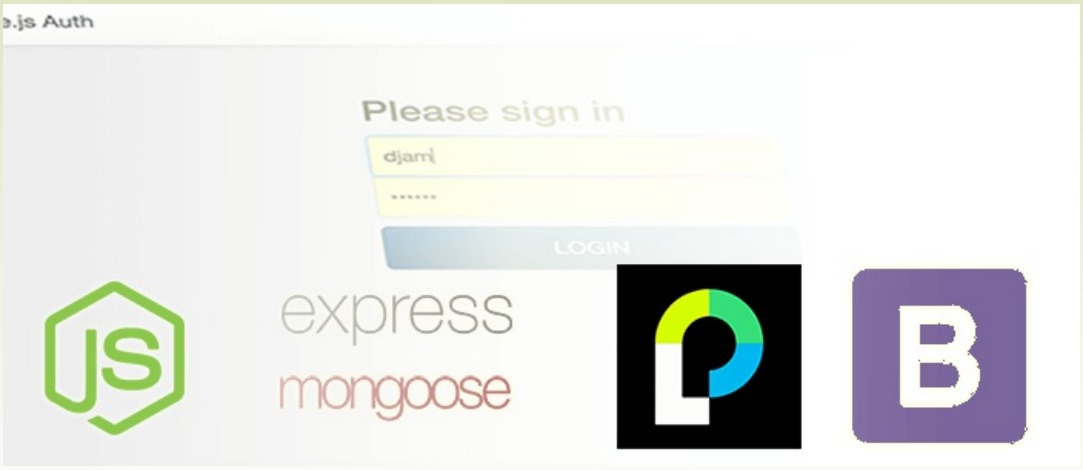
Node.js, Express.js, Mongoose.js and Passport.js Authentication

by Didin J. on Mar 06, 2017



> How to create user authentication using Node.js, Express.js, Mongoose.js and Passport.js with simple application example.

This tutorial is how to create user authentication in the Node.js application with the combination of Express.js, Mongoose.js, and Passport.js. All of that libraries, modules, and dependencies is ready to use in Node environment. And we will use Jade as template engine with Bootstrap 3 as responsive CSS framework for making styling fast and easy. As usually, let's jump to the tutorial.

1. Create Express.js web application

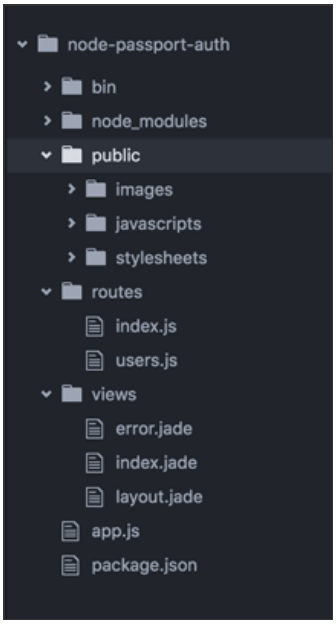
We assume that you already installed all required tools like Node.js and Express.js application generator. Open terminal or cmd then go to projects folder and run this command.

```
express node-passport-auth
```

Different with previous Node.js tutorial that now we are not using '--ejs' prefix because now, we will using 'jade' as template engine that comes as default Express.js application generation. Go to the newly created folder then install npm module.

```
cd node-passport-auth && npm install
```

And here is generated application project structure.



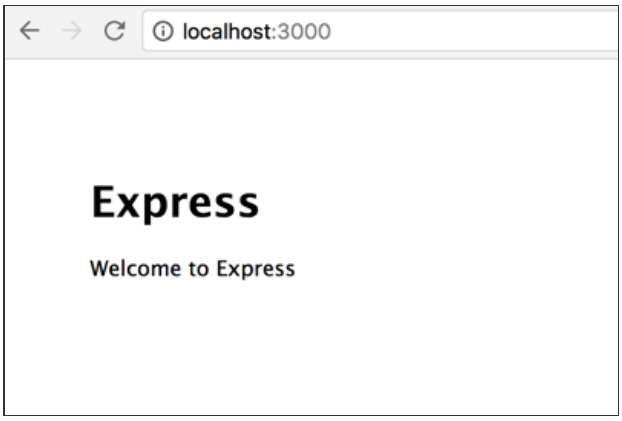
Now, you can run the application to make sure everything working properly. You can use one of the commands below.

```
nodemon
```

or

```
npm start
```

If you see this page, that would be easy to continue to the next steps.



2. Install Mongoose.js and Passport.js modules and dependencies

Now, we have to add database ORM/ODM for connection Node.js application with MongoDB database. For that, type this command after you stop the application.

```
npm install mongoose --save
```

For Passport.js we have to run this command.

```
npm install passport passport-local passport-local-mongoose --save
```

We have to install Express-Session too for storing authentication token in cookies.

```
npm install express-session --save
```

Open and edit app.js from the root of the project folder. Add Mongoose.js to 'require' and call connection to MongoDB.

```
var mongoose = require('mongoose');
mongoose.Promise = global.Promise;

mongoose.connect('mongodb://localhost/node-auth')
  .then(() =>  console.log('connection succesful'))
  .catch((err) => console.error(err));
```

Add require for passport and passport-local.

```
var passport = require('passport');
var LocalStrategy = require('passport-local').Strategy;
```

In 'app.use' section add this lines for initialize passport and express-session.

```
app.use(require('express-session')({
    secret: 'keyboard cat',
    resave: false,
    saveUninitialized: false
}));
app.use(passport.initialize());
app.use(passport.session());
```

Now, add passport configuration.

```
var User = require('./models/user');
passport.use(new LocalStrategy(User.authenticate()));
passport.serializeUser(User.serializeUser());
passport.deserializeUser(User.deserializeUser());
```

Now run again the application, but don't forget to run MongoDB server in another terminal tab. If you run again your application and see the message below, then your application mongoose configuration is ok.

```
[nodemon] 1.11.0
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node ./bin/www`
connection succesful
```

3. Create Mongoose.js User Model

This time to create models for authentication requirement. Create new folder in the root of project folder then add file for User Model.

```
mkdir models
touch models/User.js
```

Open and edit models/User.js then add this lines of codes.

```
var mongoose = require('mongoose');
var Schema = mongoose.Schema;
var passportLocalMongoose = require('passport-local-mongoose');

var UserSchema = new Schema({
    username: String,
    password: String
});

UserSchema.plugin(passportLocalMongoose);

module.exports = mongoose.model('User', UserSchema);
```

4. Create Controller for Authentication

To control access from views to models and vice-versa, we will create Controller for authentication. This is just

implementing basic MVC pattern. Create controllers folder then create controller file on the root of project folder.

```
mkdir controllers
touch controllers/AuthController.js
```

Open and edit AuthController.js then add all these lines of codes.

```
var mongoose = require("mongoose");
var passport = require("passport");
var User = require("../models/User");

var userController = {};

// Restrict access to root page
userController.home = function(req, res) {
  res.render('index', { user : req.user });
};

// Go to registration page
userController.register = function(req, res) {
  res.render('register');
};

// Post registration
userController.doRegister = function(req, res) {
  User.register(new User({ username : req.body.username, name: req.body.name }), req.body.password, function(err, user) {
    if (err) {
      return res.render('register', { user : user });
    }

    passport.authenticate('local')(req, res, function () {
      res.redirect('/');
    });
  });
};

// Go to login page
userController.login = function(req, res) {
  res.render('login');
};

// Post login
userController.doLogin = function(req, res) {
  passport.authenticate('local')(req, res, function () {
    res.redirect('/');
  });
};

// logout
userController.logout = function(req, res) {
  req.logout();
  res.redirect('/');
};

module.exports = userController;
```

5. Create Routes

We need to create routes for authentication mechanism. Open and edit routes/index.js then add this all lines of codes.

```
var express = require('express');
var router = express.Router();
var auth = require("../controllers/AuthController.js");

// restrict index for logged in user only
router.get('/', auth.home);

// route to register page
router.get('/register', auth.register);

// route for register action
router.post('/register', auth.doRegister);

// route to login page
router.get('/login', auth.login);

// route for login action
router.post('/login', auth.doLogin);

// route for logout action
router.get('/logout', auth.logout);

module.exports = router;
```

6. Create views

Now is the time for creating user interface or views. Open and edit views/layout.jade then add bootstrap for styling view via CDN.

```
doctype html
html
  head
    title= title
    link(rel='stylesheet', href='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css', integrity='sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u', crossorigin='anonymous')
    link(rel='stylesheet', href='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css', integrity='sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp', crossorigin='anonymous')
    link(rel='stylesheet', href='/stylesheets/style.css')
  body
    nav.navbar.navbar-default
      div.container-fluid
        div.navbar-header
          button.navbar-toggle.collapsed(type='button', data-toggle='collapse', data-target='#bs-example-navbar-collapse-1', aria-expanded='false')
            span.sr-only Toggle navigation
            span.icon-bar
            span.icon-bar
            span.icon-bar
          a.navbar-brand(href='#') Node.js Auth
        div.collapse.navbar-collapse(id='bs-example-navbar-collapse-1')
          ul.nav.navbar-nav.navbar-right
            if (!user)
              li
                a(href='/login') Login
              li
                a(href='/register') Register
            if (user)
              li
                a Welcome #{user.name}
              li
                a(href='/logout') Logout

    div.container
      div.content
        block content

    script(src='https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js', integrity='sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa', crossorigin='anonymous')
```

Open and edit views/index.jade then replace all codes with this.

```
extends layout

block content
  h1 Node.js, Express.js, Mongoose.js and Passport.js
  p User Authentication Example
```

Create new files for login and register form.

```
touch views/login.jade
touch views/register.jade
```

Open and edit views/login.jade then add this lines of codes.

```
extends layout

extends layout

block content
  .container
    form.form-signin(role='form', action='/login', method='post')
      h2.form-signin-heading Please sign in
      label.sr-only(for='inputEmail')
      input.form-control(type='text', name='username', id='inputEmail', placeholder='Username', required, autofocus)
      input.form-control(type='password', name='password', id='inputPassword', placeholder='Password')
      button.btn.btn-lg.btn-primary.btn-block(type='submit') LOGIN
```

Open and edit views/register.jade then add this lines of codes.

```
extends layout

block content
  .container
    form.form-signin(role='form', action="/register",method="post", style='max-width: 300px;')
      h2.form-signin-heading Sign Up here
      input.form-control(type='text', name="name", placeholder='Your Name')
      input.form-control(type='text', name="username", placeholder='Your Username')
      input.form-control(type='password', name="password", placeholder='Your Password')
      button.btn.btn-lg.btn-primary.btn-block(type='submit') Sign Up
```

Next, we have to do a little styling for this views. Open and edit public/stylesheets/style.css then replace all code with this.

```
body {
  background-color: #eee;
}

.form-signin {
  max-width: 330px;
  padding: 15px;
  margin: 0 auto;
}
.form-signin .form-signin-heading,
.form-signin .checkbox {
  margin-bottom: 10px;
}
.form-signin .checkbox {
  font-weight: normal;
}
.form-signin .form-control {
  position: relative;
  height: auto;
  -webkit-box-sizing: border-box;
     -moz-box-sizing: border-box;
          box-sizing: border-box;
  padding: 10px;
  font-size: 16px;
}
.form-signin .form-control:focus {
  z-index: 2;
}
.form-signin input[type="email"] {
  margin-bottom: -1px;
  border-bottom-right-radius: 0;
  border-bottom-left-radius: 0;
}
.form-signin input[type="password"] {
  margin-bottom: 10px;
  border-top-left-radius: 0;
  border-top-right-radius: 0;
}
```

7. Run and Test the Application

Finally, we have to try and run the application to see if all functionality is working fine.

```
nodemon
```

Open a browser and point to 'localhost:3000'. You should see this page on the browser.

You can start register then it will automatically log in.

Full source code on Github.

> This tutorial uses basic Node.js and Express.js with help of Mongoose.js, Passport.js, and Bootstrap. Feel free to give us input or suggestion in the comment below.

That just the basic. If you need more deep learning about MEAN Stack, Angular, and Node.js, you can find the following books:

- Angular 4 Projects
- Pro MEAN Stack Development
- Practical Node.js
- Pro Express.js

For more detailed on MEAN stack and Node.js, you can take the following course:

- Angular (Angular 2+) & NodeJS - The MEAN Stack Guide
- Start Building Web Apps And Services With Node. js + Express
- Build a REST API with node. js, ExpressJS, and MongoDB
- Angular 5 Bootcamp FastTrack

Thanks.

The following resources might be useful for you:

- Master essential business skills to advance your career or grow your business.
- DATA SCIENTIST: THE SEXIEST JOB OF THE 21ST CENTURY

Download the leading IDE - Studio 3T for MongoDB

The professional choice for serious MongoDB data. Download your free Trial!

studio3t.com/MongoDB-IDE

Previous Article

How to Create Node.js, Express.js and MongoDB CRUD Web Application

Next Article

Tutorial Building CRUD App from Scratch using MEAN Stack (Angular 2)

Related Articles

- How to Create REST API Easily using Node.js, Express.js, Mongoose.js and MongoDB
- How to Create Node.js, Express.js and MongoDB CRUD Web Application
- Tutorial Building CRUD App from Scratch using MEAN Stack (Angular 2)
- Getting Started Angular 4 using Angular CLI
- Building Chat Application using MEAN Stack (Angular 4) and Socket.io
- Node, Express, Mongoose and Passport.js REST API Authentication
- Node Express Passport Facebook Twitter Google GitHub Login
- Building CRUD Web Application using MERN Stack
- MEAN Stack (Angular 5) CRUD Web Application Example
- Mongo Express Vue Node.js (MEVN Stack) CRUD Web Application
- Node.js and MongoDB Slack Bot Example
- Setup Node.js, Nginx and MongoDB on Ubuntu 16.04 for Production
- Securing MEAN Stack (Angular 5) Web Application using Passport
- Securing MERN Stack Web Application using Passport

Programming Blog

- Node.js (15)
- CSS 3 (5)
- Javascript (5)
- Java (12)
- Ionic Framework (34)

Groovy and Grails (18)

HTML 5 Tutorial (3)

MongoDB (5)

All Articles

Popular Articles:

- MEAN Stack (Angular 5) CRUD Web Application Example
- Ionic 3 Consuming REST API using New Angular 4.3 HttpClient
- Ionic 3 and Angular 4 Mobile App Example
- Securing MEAN Stack (Angular 5) Web Application using Passport
- Ionic 3 and Angular 5 Mobile App Example
- Ionic 3, Angular 4 and SQLite CRUD Offline Mobile App
- How to Upload File on Ionic 3 using Native File Transfer Plugin
- Step by Step Tutorial of Ionic 3, Angular 4 and Google Maps Directions Service
- Step by step tutorial building Ionic 2 REST API Authentication
- Build Ionic 3, Angular 5 and Firebase Simple Chat App