MEAN Stack (Angular 5) CRUD Web Application Example

by Didin J. on Nov 11, 2017

Step by step tutorial on building MEAN Stack (Angular 5) Create-Read-Update-Delete (CRUD) Web Application from scratch using Angular CLI.

Step by step tutorial on building MEAN Stack (Angular 5) Create-Read-Update-Delete (CRUD) Web Application from scratch using Angular CLI. As you know that Angular 5 has been launched a few days ago, we need to test out their feature especially with MEAN Stack. MEAN stands for MongoDB, Express, Angular, and Node.js. MongoDB and Express on Node.js environment used as backend and Angular 5 used as the front end. As the previous tutorial (https://www.djamware.com/post/58cf4e1c80aca72df8d1cf7e/tutorial-building-crud-app-from-scratch-using-mean-stack-angular-2) about MEAN stack, we will be wrapping Express and Angular 5 together. run as one server.

The following tools, frameworks, and modules are required for this tutorial:

- Node.js (recommended version)
- Angular CLI 1.5
- Angular 5
- MongoDB
- Express.js
- Mongoose.js
- IDE or Text Editor

We assume that you already installed Node.js and runnable in the Terminal (Linux/Mac) or Node.js command line (Windows). Also, you have installed MongoDB and run Mongo daemon on your machine.

1. Update Angular CLI and Create Angular 5 Application

First, we have to update the Angular CLI to the latest version (1.5 when this tutorial was written). Open the terminal or Node command line then go to your projects folder. Type this command for updating Angular CLI.

```
sudo npm install -g @angular/cli
```

You can exclude `sudo` when you update or install Angular CLI on Windows/Node command line. Now, type this command to create new Angular 2 application.

```
ng new mean-angular5
```

Go to the newly created application folder.

```
cd ./mean-angular5
```

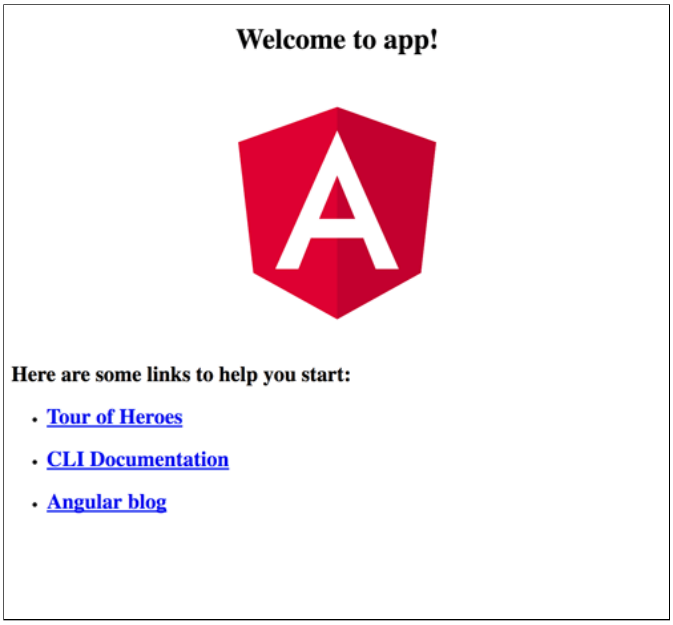
Run the Angular 2 application by typing this command.

```
ng serve
```

You see the compilation process faster than the previous Angular version.

```
** NG Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ (http://localhost:4200/) **
Date: 2017-11-10T23:12:58.186Z                                    - Hash: a8de16d629b34a42bbda
Time: 9459ms
chunk {inline} inline.bundle.js (inline) 5.79 kB [entry] [rendered]
chunk {main} main.bundle.js (main) 20.6 kB [initial] [rendered]
chunk {polyfills} polyfills.bundle.js (polyfills) 553 kB [initial] [rendered]
chunk {styles} styles.bundle.js (styles) 33.8 kB [initial] [rendered]
chunk {vendor} vendor.bundle.js (vendor) 7.03 MB [initial] [rendered]

webpack: Compiled successfully.
```

Now, open the browser then go to `http://localhost:4200 (http://localhost:4200)` you should see this page.



2. Replace Web Server with Express.js

Close the running Angular app first by press `ctrl+c` then type this command for adding Express.js modules and it dependencies.

```
npm install --save express body-parser morgan body-parser serve-favicon
```

Then, add bin folder and www file inside bin folder.

```
mkdir bin
touch bin/www
```

Open and edit www file then add this lines of codes.

```
#!/usr/bin/env node

/**
 * Module dependencies.
 */

var app = require('../app');
var debug = require('debug')('mean-app:server');
var http = require('http');

/**
 * Get port from environment and store in Express.
 */

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

/**
 * Create HTTP server.
 */

var server = http.createServer(app);

/**
 * Listen on provided port, on all network interfaces.
 */

server.listen(port);
server.on('error', onError);
server.on('listening', onListening);

/**
 * Normalize a port into a number, string, or false.
 */

function normalizePort(val) {
  var port = parseInt(val, 10);

  if (isNaN(port)) {
    // named pipe
    return val;
  }

  if (port >= 0) {
    // port number
    return port;
  }

  return false;
}

/**
 * Event listener for HTTP server "error" event.
 */

function onError(error) {
  if (error.syscall !== 'listen') {
    throw error;
  }

  var bind = typeof port === 'string'
    ? 'Pipe ' + port
    : 'Port ' + port;

  // handle specific listen errors with friendly messages
  switch (error.code) {
    case 'EACCES':
      console.error(bind + ' requires elevated privileges');
      process.exit(1);
      break;
    case 'EADDRINUSE':
      console.error(bind + ' is already in use');
      process.exit(1);
      break;
    default:
      throw error;
  }
}

/**
 * Event listener for HTTP server "listening" event.
```

```
 */

function onListening() {
  var addr = server.address();
  var bind = typeof addr === 'string'
    ? 'pipe ' + addr
    : 'port ' + addr.port;
  debug('Listening on ' + bind);
}
```

To make the server run from bin/www, open and edit "package.json" then replace "start" value.

```
"scripts": {
  "ng": "ng",
  "start": "ng build && node ./bin/www",
  "build": "ng build",
  "test": "ng test",
  "lint": "ng lint",
  "e2e": "ng e2e"
},
```

Now, create app.js in the root of project folder.

```
touch app.js
```

Open and edit app.js then add all this lines of codes.

```
var express = require('express');
var path = require('path');
var favicon = require('serve-favicon');
var logger = require('morgan');
var bodyParser = require('body-parser');

var book = require('./routes/book');
var app = express();

app.use(logger('dev'));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({'extended':'false'}));
app.use(express.static(path.join(__dirname, 'dist')));
app.use('/books', express.static(path.join(__dirname, 'dist')));
app.use('/book', book);

// catch 404 and forward to error handler
app.use(function(req, res, next) {
  var err = new Error('Not Found');
  err.status = 404;
  next(err);
});

// error handler
app.use(function(err, req, res, next) {
  // set locals, only providing error in development
  res.locals.message = err.message;
  res.locals.error = req.app.get('env') === 'development' ? err : {};

  // render the error page
  res.status(err.status || 500);
  res.render('error');
});

module.exports = app;
```

Next, create routes folder then create routes file for the book.

```
mkdir routes
touch routes/book.js
```

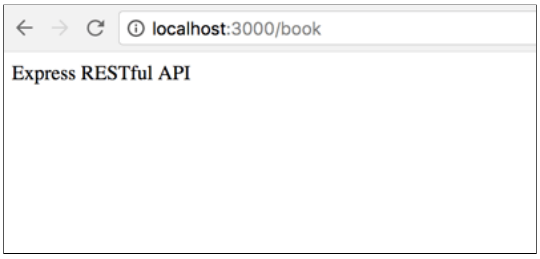
Open and edit `routes/book.js` file then add this lines of codes.

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
  res.send('Express RESTful API');
});

module.exports = router;
```

Now, run the server using this command.

```
npm start
```

You will see the previous Angular landing page when you point your browser to `http://localhost:3000 (http://localhost:3000)`. When you change the address to `http://localhost:3000/book (http://localhost:3000/book)` you will see this page.



Now, we have RESTful API with the compiled Angular 5 front end.

3. Install and Configure Mongoose.js

We need to access data from MongoDB. For that we will install and configure Mongoose.js. On the terminal type this command after stopping the running Express server.

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

Open and edit `app.js` then add this lines after another variable line.

```
var mongoose = require('mongoose');
mongoose.Promise = require('bluebird');
mongoose.connect('mongodb://localhost/mean-angular5', { useMongoClient: true, promiseLibrary: require('bluebird') })
  .then(() =>  console.log('connection succesful'))
  .catch((err) => console.error(err));
```

Now, run MongoDB server on different terminal tab or command line or run from the service.

```
mongod
```

Next, you can test the connection to MongoDB run again the Node application and you will see this message on the terminal.

```
connection succesful
```

If you are still using built-in Mongoose Promise library, you will get this deprecated warning on the terminal.

```
(node:42758) DeprecationWarning: Mongoose: mpromise (mongoose's default promise library) is deprecated, plug in your own promise library instead: ht
tp://mongoosejs.com/docs/promises.html (http://mongoosejs.com/docs/promises.html)
```

That's the reason why we added `bluebird` modules and register it as Mongoose Promise library.

4. Create Mongoose.js Model

Add a models folder on the root of project folder for hold Mongoose.js model files.

```
mkdir models
```

Create new Javascript file that uses for Mongoose.js model. We will create a model of Book collection.

```
touch models/Book.js
```

Now, open and edit that file and add Mongoose require.

```
var mongoose = require('mongoose');
```

Then add model fields like this.

```
var BookSchema = new mongoose.Schema({
  isbn: String,
  title: String,
  author: String,
  description: String,
  published_year: String,
  publisher: String,
  updated_date: { type: Date, default: Date.now },
});
```

That Schema will mapping to MongoDB collections called book. If you want to know more about Mongoose Schema Datatypes you can find it here (http://mongoosejs.com/docs/schematypes.html). Next, export that schema.

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

5. Create Routes for Accessing Book Data via Restful API

Open and edit again "routes/book.js" then replace all codes with this.

```
var express = require('express');
var router = express.Router();
var mongoose = require('mongoose');
var Book = require('../models/Book.js');

/* GET ALL BOOKS */
router.get('/', function(req, res, next) {
  Book.find(function (err, products) {
    if (err) return next(err);
    res.json(products);
  });
});

/* GET SINGLE BOOK BY ID */
router.get('/:id', function(req, res, next) {
  Book.findById(req.params.id, function (err, post) {
    if (err) return next(err);
    res.json(post);
  });
});

/* SAVE BOOK */
router.post('/', function(req, res, next) {
  Book.create(req.body, function (err, post) {
    if (err) return next(err);
    res.json(post);
  });
});

/* UPDATE BOOK */
router.put('/:id', function(req, res, next) {
  Book.findByIdAndUpdate(req.params.id, req.body, function (err, post) {
    if (err) return next(err);
    res.json(post);
  });
});

/* DELETE BOOK */
router.delete('/:id', function(req, res, next) {
  Book.findByIdAndRemove(req.params.id, req.body, function (err, post) {
    if (err) return next(err);
    res.json(post);
  });
});

module.exports = router;
```

Run again the Express server then open the other terminal or command line to test the Restful API by type this command.

```
curl -i -H "Accept: application/json" localhost:3000/book
```

If that command return response like below then REST API is ready to go.

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 2
ETag: W/"2-l9Fw4VUO7kr8CvBlt4zaMCqXZ0w"
Date: Fri, 10 Nov 2017 23:53:52 GMT
Connection: keep-alive
```

Now, let's populate Book collection with initial data that sent from RESTful API. Run this command to populate it.

```
curl -i -X POST -H "Content-Type: application/json" -d '{ "isbn":"123442123, 97885654453443","title":"Learn how to build modern web application with MEAN stack","author": "Didin J.","description":"The comprehensive step by step tutorial on how to build MEAN (MongoDB, Express.js, Angular 5 and Node.js) stack web application from scratch","published_year":"2017","publisher":"Djamware.com" }' localhost:3000/book
```

You will see this response to the terminal if success.

```
HTTP/1.1 200 OK
X-Powered-By: Express
Content-Type: application/json; charset=utf-8
Content-Length: 415
ETag: W/"19f-SB/dEQyffaTjobOBJbvmwCn7WJA"
Date: Fri, 10 Nov 2017 23:58:11 GMT
Connection: keep-alive

{"__v":0,"isbn":"123442123, 97885654453443","title":"Learn how to build modern web application with MEAN stack","author":"Didin J.","description":"The comprehensive step by step tutorial on how to build MEAN (MongoDB, Express.js, Angular 5 and Node.js) stack web application from scratch","published_year":"2017","publisher":"Djamware.com","_id":"5a063d123cf0792af12ce45d","updated_date":"2017-11-10T23:58:10.971Z"}MacBook-Pro:mean-angular5
```

6. Create Angular 5 Component for Displaying Book List

To create Angular 5 Component, simply run this command.

```
ng g component book
```

That command will generate all required files for build book component and also automatically added book component to app.module.ts.

```
create src/app/book/book.component.css (0 bytes)
create src/app/book/book.component.html (23 bytes)
create src/app/book/book.component.spec.ts (614 bytes)
create src/app/book/book.component.ts (321 bytes)
update src/app/app.module.ts (390 bytes)
```

Before add any functionality to the component, we need to add `HttpClientModule` to `app.module.ts`. Open and edit `src/app/app.module.ts` then add this import.

```
import { FormsModule } from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
```

Add it to `@NgModule` imports after `BrowserModule`.

```
imports: [
  BrowserModule,
  FormsModule,
  HttpClientModule
],
```

Now, we will making a request to Book RESTful API using this Angular `HttpClient` module. Open and edit `src/app/book/book.component.ts` then add this import.

```
import { HttpClient } from '@angular/common/http';
```

Inject `HttpClient` to the constructor.

```
constructor(private http: HttpClient) { }
```

Add array variable for holding books data before the constructor.

```
books: any;
```

Add a few lines of codes for getting a list of book data from RESTful API inside `ngOnInit` function.

```
ngOnInit() {
  this.http.get('/book').subscribe(data => {
    this.books = data;
  });
}
```

Now, we can display the book list on the page. Open and edit `src/app/book/book.component.html` then replace all tags with this lines of HTML tags.

```
<div class="container">
  <h1>Book List</h1>
  <table class="table">
    <thead>
      <tr>
        <th>Title</th>
        <th>Author</th>
        <th>Action</th>
      </tr>
    </thead>
    <tbody>
      <tr *ngFor="let book of books">
        <td>{{ book.title }}</td>
        <td>{{ book.author }}</td>
        <td>Show Detail</td>
      </tr>
    </tbody>
  </table>
</div>
```

That HTML tags include style class from Bootstrap CSS library. Open and edit `src/index.html` then add the Bootstrap CSS and JS library.

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>MeanAngular5</title>
  <base href="/">

  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <!-- Latest compiled and minified CSS -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css (https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css)" integrity="sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="anonymous">
  <!-- Optional theme -->
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css (https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css)" integrity="sha384-rHyoN1iRsVXV4nD0JutlnGaslCJuC7uwjduW9SVrLvRYooPp2bWYgmgJQIXwl/Sp" crossorigin="anonymous">
</head>
<body>
  <app-root></app-root>
  <!-- Latest compiled and minified JavaScript -->
  <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js (https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js)" integrity="sha384-Tc5IQib027qvyjSMfHjOMaLkfuWVxZxUPnCJA7l2mCWNIpG9mGCD8wGNIcPD7Txa" crossorigin="anonymous"></script>
</body>
</html>
```

7. Create Angular 5 Routes to Book Component

To use book component as default landing page, open and edit `src/app/app.module.ts` the add import for Routing.

```
import { RouterModule, Routes } from '@angular/router';
```

Create constant router for routing to book component before `@NgModule`.

```
const appRoutes: Routes = [
  {
    path: 'books',
    component: BookComponent,
    data: { title: 'Book List' }
  },
  { path: '',
    redirectTo: '/books',
    pathMatch: 'full'
  }
];
```

In @NgModule imports, section adds ROUTES constant, so imports section will be like this.

```
imports: [
  BrowserModule,
  HttpClientModule,
  RouterModule.forRoot(
    appRoutes,
    { enableTracing: true } // <-- debugging purposes only
  )
],
```

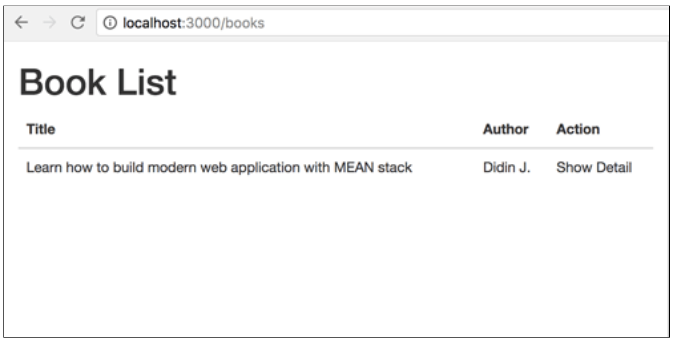
To activate that routes in Angular 5, open and edit `src/app/app.component.html` then replace all codes with this.

```
<router-outlet></router-outlet>
```

Now, we have to test our MEAN app with only list page. Build then run the application.

```
npm start
```

You should see this page when pointing to `http://localhost:3000 (http://localhost:3000)` or `http://localhost:3000/books (http://localhost:3000/books)`.



8. Create Angular 5 Component for Displaying Book Detail

Same as previous section, type this command to generate new component.

```
ng g component book-detail
```

Add router to `src/app/app.module.ts` routes constant.

```
const appRoutes: Routes = [
  {
    path: 'books',
    component: BookComponent,
    data: { title: 'Book List' }
  },
  {
    path: 'book-details/:id',
    component: BookDetailComponent,
    data: { title: 'Book Details' }
  },
  { path: '',
    redirectTo: '/books',
    pathMatch: 'full'
  }
];
```

Open and edit `src/app/book-detail/book-detail.component.ts`. Replace all codes with this.

```
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-book-detail',
  templateUrl: './book-detail.component.html',
  styleUrls: ['./book-detail.component.css'],
  encapsulation: ViewEncapsulation.None
})
export class BookDetailComponent implements OnInit {

  book = {};

  constructor(private route: ActivatedRoute, private http: HttpClient) { }

  ngOnInit() {
    this.getBookDetail(this.route.snapshot.params['id']);
  }

  getBookDetail(id) {
    this.http.get('/book/'+id).subscribe(data => {
      this.book = data;
    });
  }

}
```

Open and edit `src/app/book-detail/book-detail.component.html`. Replace all codes with this.

```
<div class="container">
  <h1>{{ book.title }}</h1>
  <dl class="list">
    <dt>ISBN</dt>
    <dd>{{ book.isbn }}</dd>
    <dt>Author</dt>
    <dd>{{ book.author }}</dd>
    <dt>Publisher</dt>
    <dd>{{ book.publisher }}</dd>
    <dt>Price</dt>
    <dd>{{ book.price }}</dd>
    <dt>Update Date</dt>
    <dd>{{ book.updated_at }}</dd>
  </dl>
</div>
```

9. Create Angular 5 Component for Add New Book

To create a component to add new Book, type this command as usual.

```
ng g component book-create
```

Add router to `src/app/app.module.ts` routes constant.

```
const appRoutes: Routes = [
  {
    path: 'books',
    component: BookComponent,
    data: { title: 'Book List' }
  },
  {
    path: 'book-details/:id',
    component: BookDetailComponent,
    data: { title: 'Book Details' }
  },
  {
    path: 'book-create',
    component: BookCreateComponent,
    data: { title: 'Create Book' }
  },
  { path: '',
    redirectTo: '/books',
    pathMatch: 'full'
  }
];
```

Add 'book-create' link on `src/app/book/book.component.html`.

```
<h1>Book List
  <a [routerLink]="['/book-create']" class="btn btn-default btn-lg">
    <span class="glyphicon glyphicon-plus" aria-hidden="true"></span>
  </a>
</h1>
```

Now, open and edit `src/app/book/book-create.component.ts` then replace all with this codes.

```
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { Router } from '@angular/router';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-book-create',
  templateUrl: './book-create.component.html',
  styleUrls: ['./book-create.component.css'],
  encapsulation: ViewEncapsulation.None
})
export class BookCreateComponent implements OnInit {

  book = {};

  constructor(private http: HttpClient, private router: Router) { }

  ngOnInit() {
  }

  saveBook() {
    this.http.post('/book', this.book)
      .subscribe(res => {
          let id = res['_id'];
          this.router.navigate(['/book-details', id]);
        }, (err) => {
          console.log(err);
        }
      );
  }

}
```

Modify `src/app/book-create/book-create.component.html`, replace all with this HTML tags.

```
<div class="container">
  <h1>Add New Book</h1>
  <div class="row">
    <div class="col-md-6">
      <form (ngSubmit)="saveBook()" #bookForm="ngForm">
        <div class="form-group">
          <label for="name">ISBN</label>
          <input type="text" class="form-control" [(ngModel)]="book.isbn" name="isbn" required>
        </div>
        <div class="form-group">
          <label for="name">Title</label>
          <input type="text" class="form-control" [(ngModel)]="book.title" name="title" required>
        </div>
        <div class="form-group">
          <label for="name">Author</label>
          <input type="text" class="form-control" [(ngModel)]="book.author" name="author" required>
        </div>
        <div class="form-group">
          <label for="name">Published Year</label>
          <input type="number" class="form-control" [(ngModel)]="book.published_year" name="published_year" required>
        </div>
        <div class="form-group">
          <label for="name">Publisher</label>
          <input type="text" class="form-control" [(ngModel)]="book.publisher" name="publisher" required>
        </div>
        <div class="form-group">
          <button type="submit" class="btn btn-success" [disabled]="!bookForm.form.valid">Save</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

10. Create Angular 5 Component for Edit Book

As usual, we will generate component for edit book. Type this command for doing that.

```
ng g component book-edit
```

Add route in `src/app/app.module.ts` so, it looks like this.

```
const appRoutes: Routes = [
  {
    path: 'books',
    component: BookComponent,
    data: { title: 'Book List' }
  },
  {
    path: 'book-details/:id',
    component: BookDetailComponent,
    data: { title: 'Book Details' }
  },
  {
    path: 'book-create',
    component: BookCreateComponent,
    data: { title: 'Create Book' }
  },
  {
    path: 'book-edit/:id',
    component: BookEditComponent,
    data: { title: 'Edit Book' }
  },
  { path: '',
    redirectTo: '/books',
    pathMatch: 'full'
  }
];
```

Open and edit again `src/app/book-details/book-details.component.html` and add edit routeLink in the last line.

```
<div class="row">
  <div class="col-md-12">
    <a [routerLink]="['/book-edit', book._id]" class="btn btn-success">EDIT</a>
  </div>
</div>
```

Now, open and edit `src/app/book-edit/book-edit.component.ts` then replace all codes with this.

```
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-book-edit',
  templateUrl: './book-edit.component.html',
  styleUrls: ['./book-edit.component.css'],
  encapsulation: ViewEncapsulation.None
})
export class BookEditComponent implements OnInit {

  book = {};

  constructor(private http: HttpClient, private router: Router, private route: ActivatedRoute) { }

  ngOnInit() {
    this.getBook(this.route.snapshot.params['id']);
  }

  getBook(id) {
    this.http.get('/book/'+id).subscribe(data => {
      this.book = data;
    });
  }

  updateBook(id, data) {
    this.http.put('/book/'+id, data)
      .subscribe(res => {
          let id = res['_id'];
          this.router.navigate(['/book-details', id]);
        }, (err) => {
          console.log(err);
        }
      );
  }

}
```

Open and edit `src/app/book-edit/book-edit.component.html` then replace all codes with this.

```
<div class="container">
  <h1>Edit Book</h1>
  <div class="row">
    <div class="col-md-6">
      <form (ngSubmit)="updateBook(book._id)" #bookForm="ngForm">
        <div class="form-group">
          <label for="name">ISBN</label>
          <input type="text" class="form-control" [(ngModel)]="book.isbn" name="isbn" required>
        </div>
        <div class="form-group">
          <label for="name">Title</label>
          <input type="text" class="form-control" [(ngModel)]="book.title" name="title" required>
        </div>
        <div class="form-group">
          <label for="name">Author</label>
          <input type="text" class="form-control" [(ngModel)]="book.author" name="author" required>
        </div>
        <div class="form-group">
          <label for="name">Published Year</label>
          <input type="number" class="form-control" [(ngModel)]="book.published_year" name="published_year" required>
        </div>
        <div class="form-group">
          <label for="name">Publisher</label>
          <input type="text" class="form-control" [(ngModel)]="book.publisher" name="publisher" required>
        </div>
        <div class="form-group">
          <button type="submit" class="btn btn-success" [disabled]="!bookForm.form.valid">Update</button>
        </div>
      </form>
    </div>
  </div>
</div>
```

11. Create Delete Function on Book-Detail Component

Open and edit `src/app/book-detail/book-detail`.component.ts then add `Router` module to `@angular/router`.

```
import { ActivatedRoute, Router } from '@angular/router';
```

Inject `Router` in the constructor params.

```
constructor(private router: Router, private route: ActivatedRoute, private http: HttpClient) { }
```

Add this function for delete book.

```
deleteBook(id) {
  this.http.delete('/book/'+id)
    .subscribe(res => {
        this.router.navigate(['/books']);
      }, (err) => {
        console.log(err);
      }
    );
}
```

Add delete button in `src/app/book-detail/book-detail.component.html` on the right of Edit routerLink.

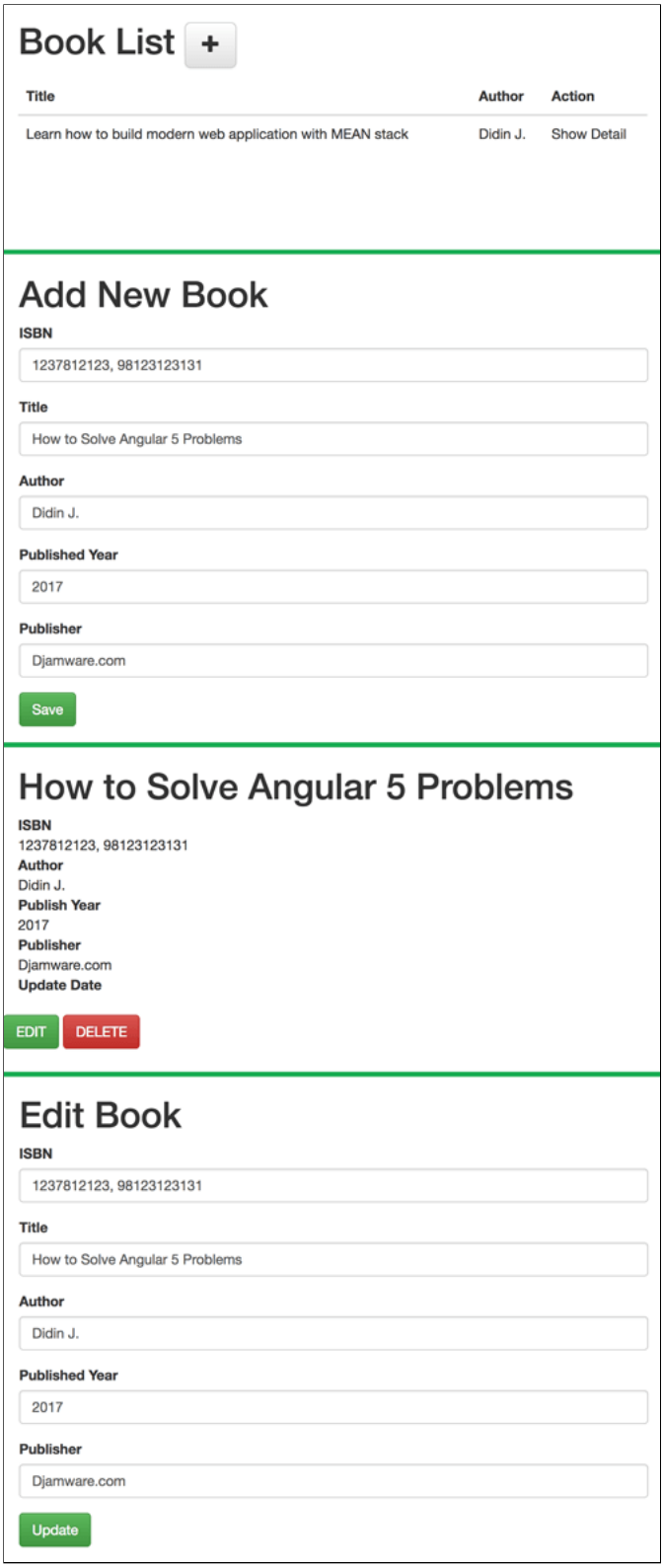
```
<div class="row">
  <div class="col-md-12">
    <a [routerLink]="['/book-edit', book._id]" class="btn btn-success">EDIT</a>
    <button class="btn btn-danger" type="button" (click)="deleteBook(book._id)">DELETE</button>
  </div>
</div>
```

12. Run and Test the MEAN Stack (Angular 5) CRUD Application

Now, it's a time for testing the MEAN Stack (Angular 5) CRUD Application.

```
npm start
```

And here we are.



If you need the full working source code, you can find it on our GitHub (https://github.com/didinj/mean-stack-angular5-crud.git).

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 (http://www.anrdoezrs.net/click-8263647-12169838?url=https%3A%2F%2Fwww.apress.com%2Fus%2Fbook%2F9781484232781%3Futm_medium%3Daffiliate%26utm_source%3Dcommissi
- Pro MEAN Stack Development (http://www.dpbolvw.net/click-8263647-12169838?url=https%3A%2F%2Fwww.apress.com%2Fus%2Fbook%2F9781484220436%3Futm_medium%3Daffiliate%26utm_source%3Dcommiss
- Practical Node.js (http://www.jdoqocy.com/click-8263647-12752006?url=https%3A%2F%2Fwww.apress.com%2Fde%2Fbook%2F9781430265955%3Futm_medium%3Daffiliate%26utm_source%3Dcommiss

- Pro Express.js (http://www.kqzyfj.com/click-8263647-12169838?url=https%3A%2F%2Fwww.apress.com%2Fus%2Fbook%2F9781484200384%3Futm_medium%3Daffiliate%26utm_source%3Dcommis

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

- Angular (Angular 2+) & NodeJS - The MEAN Stack Guide (https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=358574.833442&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fangular-2-and-nodejs-the-practical-guide%2F)
- Start Building Web Apps And Services With Node. js + Express (https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=358574.150396&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fthe-ultimate-guide-to-nodejs-express%2F)
- Build a REST API with node. js, ExpressJS, and MongoDB (https://click.linksynergy.com/link?id=6nYo96*QrJE&offerid=358574.654736&type=2&murl=https%3A%2F%2Fwww.udemy.com%2Fnodejs-api%2F)

Thanks!

Follow

(http://www.facebook.com/djamwarecom)

(http://twitter.com/intent/follow?source=followbutton&variant=1.0&screen_name=djamware)

(https://plus.google.com/105042825018944450705) (http://www.pinterest.com/djamware)

(http://www.linkedin.com/in/didin-jamaludin-7a530351)

(http://www.youtube.com/channel/UCtI81hYLh2Ae_45KHkyy0vw?sub_confirmation=1) (https://github.com/didinj)

The following resources might be useful for you:

- Master essential business skills to advance your career or grow your business. (http://shareasale.com/r.cfm?b=1095021&u=1451683&m=74412&urllink=&afftrack=)
- DATA SCIENTIST: THE SEXIEST JOB OF THE 21ST CENTURY (http://shareasale.com/r.cfm?b=1067437&u=1451683&m=74412&urllink=excelwithbusiness%2Ecom%2Fproduct%2Fintroduction%2Dto%2Ddata%2Dscience%2F&afft

← Previous Article

Building CRUD Web Application using MERN Stack

(/post/59faec0a80aca7739224ee1f/building-crud-web-application-using-mern-stack)

Next Article →

Mongo Express Vue Node.js (MEVN Stack) CRUD Web Application

(/post/5a1b779f80aca75eadc12d6e/mongo-express-vue-nodejs-mevn-stack-crud-web-application)

Related Articles

- How to Create REST API Easily using Node.js, Express.js, Mongoose.js and MongoDB (/post/58a91cdf80aca748640ce353/how-to-create-rest-api-easily-using-nodejs-expressjs-mongoosejs-and-mongodb)
- How to Create Node.js, Express.js and MongoDB CRUD Web Application (/post/58b27ce080aca72c54645983/how-to-create-nodejs-expressjs-and-mongodb-crud-web-application)
- Node.js, Express.js, Mongoose.js and Passport.js Authentication (/post/58bd823080aca7585c808ebf/nodejs-expressjs-mongoosejs-and-passportjs-authentication)

- Tutorial Building CRUD App from Scratch using MEAN Stack (Angular 2) (/post/58cf4e1c80aca72df8d1cf7e/tutorial-building-crud-app-from-scratch-using-mean-stack-angular-2)
- Getting Started Angular 4 using Angular CLI (/post/58d9a49b80aca7105ed7d3f7/getting-started-angular-4-using-angular-cli)
- Building Chat Application using MEAN Stack (Angular 4) and Socket.io (/post/58e0d15280aca75cdc948e4e/building-chat-application-using-mean-stack-angular-4-and-socketio)
- Node, Express, Mongoose and Passport.js REST API Authentication (/post/58eba06380aca72673af8500/node-express-mongoose-and-passportjs-rest-api-authentication)
- Node Express Passport Facebook Twitter Google GitHub Login (/post/59a6257180aca768e4d2b132/node-express-passport-facebook-twitter-google-github-login)
- Building CRUD Web Application using MERN Stack (/post/59faec0a80aca7739224ee1f/building-crud-web-application-using-mern-stack)
- Mongo Express Vue Node.js (MEVN Stack) CRUD Web Application (/post/5a1b779f80aca75eadc12d6e/mongo-express-vue-nodejs-mevn-stack-crud-web-application)
- Node.js and MongoDB Slack Bot Example (/post/5a500c9380aca7059c142973/nodejs-and-mongodb-slack-bot-example)
- Setup Node.js, Nginx and MongoDB on Ubuntu 16.04 for Production (/post/5a593bfc80aca7059c142975/setup-nodejs-nginx-and-mongodb-on-ubuntu-1604-for-production)
- Securing MEAN Stack (Angular 5) Web Application using Passport (/post/5a878b3c80aca7059c142979/securing-mean-stack-angular-5-web-application-using-passport)
- Securing MERN Stack Web Application using Passport (/post/5a90c37980aca7059c14297a/securing-mern-stack-web-application-using-passport)
- Securing MEVN Stack (Vue.js 2) Web Application using Passport (/post/5ac8338780aca714d19d5b9e/securing-mevn-stack-vuejs-2-web-application-using-passport)

angular5/node_modules/mongoose/lib/schema/objectid.js:158:13)
at ObjectId.SchemaType.applySetters (/media/madesh/ff5cd6b3-4786-48b9-9c16-4c75f18697e3/ionicprojects/mean-angular5/node_modules/mongoose/lib/schematype.js:724:12)
at ObjectId.SchemaType._castForQuery (/media/madesh/ff5cd6b3-4786-48b9-9c16-4c75f18697e3/ionicprojects/mean-angular5/node_modules/mongoose/lib/schematype.js:1095:15)
at ObjectId.castForQuery (/media/madesh/ff5cd6b3-4786-48b9-9c16-4c75f18697e3/ionicprojects/mean-angular5/node_modules/mongoose/lib/schema/objectid.js:198:15)
at ObjectId.SchemaType.castForQueryWrapper (/media/madesh/ff5cd6b3-4786-48b9-9c16-4c75f18697e3/ionicprojects/mean-angular5/node_modules/mongoose/lib/schematype.js:1064:15)
at cast (/media/madesh/ff5cd6b3-4786-48b9-9c16-4c75f18697e3/ionicprojects/mean-angular5/node_modules/mongoose/lib/cast.js:300:32)
at model.Query.Query.cast (/media/madesh/ff5cd6b3-4786-48b9-9c16-4c75f18697e3/ionicprojects/mean-angular5/node_modules/mongoose/lib/query.js:3210:12)
at model.Query.Query._castConditions (/media/madesh/ff5cd6b3-4786-48b9-9c16-4c75f18697e3/ionicprojects/mean-

see more

Reply • Share ›

Didin Jamaludin Mod ➚ Danielprabhakaran N • 3 days ago
the value of parameter id is incorrect, it should be MongoDB ObjectID instead of "book" string. Compare your code with the working code on GitHub at the end of article.
1 Reply • Share ›

don greak • 4 days ago
this runs on localhost:3000/books... but running it on localhost:4200/books gives me the table header but empty data below it... why? How can i solve this
Reply • Share ›

Atul Suroshe • 22 days ago
great example for starters. Worked like a charm. Thanks.
Reply • Share ›

Patrick Gourdet • 22 days ago
Hello I have one issue with the routing. I have created my own application using this as a boiler plate (for which I thank you very much), but when attempting to use any other path: other than the ' '(empty string) it gives me a can not find error. I am able to bind any single component to the path: ' ' , component XYZ, this works fine what am I doing wrong.

import { UsersComponent } from './users/users/users.component';
import { HomeComponent } from './home/home.component';
import {HeaderComponent} from "./header/header.component";

const appRoutes: Routes=[

{path: 'profile', component: UsersComponent},
{path: 'home-page', component: HomeComponent},
{path: '', redirectTo: '/profile',pathMatch: 'full'},
];

In the above case the url: localhost:3000 then converts to (do to the redirect) localhost:3000/profile but if I then attempt to refresh with the url: set to localhost:3000/profiles I get can not find /profiles.

Any help is very welcome.
Reply • Share ›

Kamnag R • 25 days ago
No default engine was specified and no extension was provided.

How to solve this error.

Reply • Share ›

Colper ➚ Kamnag R • 2 days ago

Just add to your app.js file:
app.set('view engine', 'ejs');

Or if you prefer Handelbars:
app.set('view engine', 'hbs');

etc...

Reply • Share ›

Show 1 new reply

Thomas Oliver ➚ Kamnag R • 19 days ago

did you find the solution? I have the same problem.

Reply • Share ›

chandra shekar • a month ago

No default engine was specified and no extension was provided.

getting this error what to do??

Reply • Share ›

Patrick Sile • a month ago

Hey, I am just started this tutorial and my first remark is that you are not using `const` when you are requiring modules in your file, but instead `var` . Can I know why?

Reply • Share ›

Sylvan Reinieren • a month ago

I Wanted to run the curl -i -x POST -H command but returns me this:

curl: (6) Could not resolve host: isbn:123442123, 97885654453443,title:Learn how to build modern web application with MEAN stack,author
curl: (3) Illegal port number
curl: (3) [globbing] unmatched close brace/bracket in column 1
HTTP/1.1 400 Bad Request

Reply • Share ›

keith dale cordova ➚ Sylvan Reinieren • a month ago

Try to download postman in chrome. This is much easier to use and test the API's.

Reply • Share ›

Naveen Kumar • 2 months ago

Wow. Super Tutorial. Short and Sweet. :)

Reply • Share ›

Kern • 2 months ago

Great tutorial. But I ran into a problem deploying it to Heroku. I used Heroku's

"Object Modeling in Node.js with Mongoose" tutorial to make some changes for the deployment, but I keep getting this error on Heroku:

heroku[router]: at=error code=H10 desc="App crashed" method=GET path="/"
Anyone successfully deploy to Heroku? It would be great if the tutorial could be extended one or two more
sections to show how to deploy it to Heroku.

Reply • Share ›

Nguyễn Hoài Phước • 3 months ago

I've been searching for a free angular 5 + nodejs tutorial and now i have found it. Thanks a lot, you are the best!

Reply • Share ›

Mauricio Gonzalez Betancur • 3 months ago

Hi, how can I do a routing to a link with an HTML jumper? something like example.com/#section3

Reply • Share ›

Steven • 3 months ago

Thank you very much for this brilliant tutorial!! Do you know how can we redirect all routes other than '/' that arrive on the server to the angular app, so that it can do the routing?

Reply • Share ›

Didin Jamaludin Mod ➚ Steven • 3 months ago

You should define routing that came in to express on `app.js` then you can modify the angular router on `src/app/app.module.ts`

Reply • Share ›

Tuan Jinn Nguyen • 3 months ago
Works like a charm!!! Thanks!
Reply • Share ›

Brecht Lañojan • 3 months ago
I'll just ask what is the use of this line "app.use(express.static(path.join(__dirname, 'dist')));
app.use('/books', express.static(path.join(__dirname, 'dist')));"? I'm a beginner. Please kindly explain it to me :) Thank you in advance
Reply • Share ›

Didin Jamaludin Mod ➚ Brecht Lañojan • 3 months ago
it's mean, express will scan the view/frontend/html in the dist folder which contains frontend that built by angular.
Reply • Share ›

Rua • 3 months ago
Great tutorial!! Thanks
Reply • Share ›

Andy He • 3 months ago
Hi, how do i make the configuration for auto-reloading?
Reply • Share ›

Lars Meyer • 3 months ago
Good tutorial, I was able to do it on Windows 7. However, when I use Internet Explorer 11, all I get is "Loading..." and nothing else happens. With Chrome, everything works fine. Is there a workaround for IE11? BR/ Lars
Reply • Share ›

Alfred Gårdeskog ➚ Lars Meyer • 3 months ago
Read this: https://angular.io/guide/br...
Reply • Share ›

Stanley Eosakul • 4 months ago
Well written, thank you so much! The book edit function and the updated date didn't work correctly for me so I made the following changes:

1. book-detail.component.html
<dd>{{ book.updated_at }}</dd> ==> <dd>{{ book.updated_date | date:'short' }}</dd>

2. book-edit.component.ts
book = {}; ==> book: any = {};

and

updateBook(id, data) {
this.http.put('/book/'+id, data).subscribe(...

becomes

updateBook(id) {
this.book.updated_date = Date.now();
this.http.put('/book/'+id, this.book).subscribe(...

GitHub pull request submitted to you! Also, check out my version of this app at my GitHub Repo at https://github.com/Stanza98...
Reply • Share ›

Kenneth Yong • 5 months ago
import { HttpClientModule } from '@angular/common/http';
do anyone know why is @angular/common/http giving me error message it says that cannot find module
Reply • Share ›

Ghândrį Şêmah XaVį ➚ Kenneth Yong • 4 months ago
you have to add it in the app.module.ts
Reply • Share ›

Евгений Баранов • 5 months ago
Hi. Do you know how to working with select/options when we want to show user what he choosed in previous moment. For example, we have a Selector selector and when editing we want to display this previous choice. My problem lies in the fact that when multiple options are being rewritten, only one item is returned to the server and not an array of objects.
Reply • Share ›

Julen • 5 months ago

```
import { Component, OnInit, ViewEncapsulation } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { ActivatedRoute, Router } from '@angular/router';

@Component({
selector: 'app-book-detail',
templateUrl: './book-detail.component.html',
styleUrls: ['./book-detail.component.css'],
encapsulation: ViewEncapsulation.None
})

export class BookDetailComponent implements OnInit {
book = {};
constructor(private router: Router, private route: ActivatedRoute, private http: HttpClient) { }
ngOnInit() {
this.getBookDetail(this.route.snapshot.params['id']);
}

getBookDetail(id) {
```

see more

^ | v • Reply • Share ›

Julen ➚ Julen • 5 months ago
Ok, I have solved it. Although, when I run the server the html and css dont load properly :(
^ | v • Reply • Share ›

Евгений Баранов • 5 months ago
Thank's man. You are the best. Tell me how to make a deploy of such a project.
^ | v • Reply • Share ›

Didin Jamaludin Mod ➚ Евгений Баранов • 5 months ago
That's on my plan, I'll write the step by step tutorial of how to deploy Node/Express web app on the VPS from scratch. But if you plan using ready to use PaaS, you can use Google Cloud Platform or Heroku that already include Node app and easy to deploy.
1 ^ | v • Reply • Share ›

shamoh19 ➚ Didin Jamaludin • 3 months ago
Thanks dude for the amazing tutorial.any update on deployment tutorial?
^ | v • Reply • Share ›

Didin Jamaludin Mod ➚ shamoh19 • 3 months ago
it's already here few weeks ago. You can find on related article list about setup nginx and node.js for production.
^ | v • Reply • Share ›

fmongarex ➚ Didin Jamaludin • 5 months ago
waiting for this tutorial
^ | v • Reply • Share ›

Евгений Баранов ➚ Didin Jamaludin • 5 months ago
I was able to update the data.
^ | v • Reply • Share ›

Derek Lee • 5 months ago
How do i do the view details tabs in /books?
^ | v • Reply • Share ›

Didin Jamaludin Mod ➚ Derek Lee • 5 months ago
Oh sorry, I forget something in the books page. It should be like this:

a [routerLink]="['/book-details', book._id]"> Show Details

Add '<' in front of 'a' and close 'a' tag with '' because disqus automatically translate that tags as link.
^ | v • Reply • Share ›

Derek Lee ➚ Didin Jamaludin • 5 months ago
Yeah got it working, btw is there something wrong with the update book? I can't seems to update it. Api works though, tested with postman
^ | v • Reply • Share ›

Beth Diane ➚ Derek Lee • 5 months ago

I fixed this by changing the update book function (update data to book for put)

```
updateBook(id, book) {
this.http.put('/book/'+ id, this.book)
.subscribe(res => {
let id = res['_id'];
this.router.navigate(['/books']);
}, (err) => {
console.log(err);
}
);
}
```

-- bt

Reply • Share ›

Alex Hodson ➔ Beth Diane • 5 months ago

Just spotted this and it worked for me. Thanks

Reply • Share ›

Didin Jamaludin Mod ➔ Derek Lee • 5 months ago

Yeah, I know there's a lot of missing here. I'll update it soon, but for now you can refer to Angular 5 side in this tutorial https://www.djamware.com/po...

Reply • Share ›

andrew fraser • 2 months ago

I am a little confuse, i see in others tutorials, that a backe-end (Express) is no really needed using Angular, because you can build it with just Angular native tools. Please correct me if im wrong. By the way i am new from this technologies. Thx :D

Reply • Share ›

Mohamed Hassan Kadri • 2 months ago

the command curl is not working for me how can i sole that

Reply • Share ›

Load more comments

Programming Blog
(/post-category/595f867edbd39e7571e183dc/programming-blog)

HTML 5 Tutorial (3)
(/post-sub-category/584209dffcbe618f680bdc5c/html-5-tutorial)

Groovy and Grails (18)
(/post-sub-category/585b3fa380aca73b19a2efd4/groovy-and-grails)

MongoDB (5)
(/post-sub-category/5845677b80aca7763489d871/mongodb)

Ionic Framework (35)
(/post-sub-category/5845691a80aca7763489d872/ionic-framework)

Node.js (16)
(/post-sub-category/58a9196f80aca748640ce352/nodejs)

Javascript (5)
(/post-sub-category/583d6d30fcbe614473a4c4e9/javascript)

Java (13)
(/post-sub-category/583d6c37fcbe614473a4c4e8/java)

CSS 3 (5)
(/post-sub-category/584249bde4d5d303658d1ecf/css-3)

All Articles
(/public/allArticles)

Popular Articles:
- MEAN Stack (Angular 5) CRUD Web Application Example (/post/5a0673c880aca7739224ee21/mean-stack-angular-5-crud-web-application-example)
- Ionic 3 Consuming REST API using New Angular 4.3 HttpClient

(/post/59924f9080aca768e4d2b12e/ionic-3-consuming-rest-api-using-new-angular-43-httpclient)
- Ionic 3 and Angular 4 Mobile App Example (/post/58e657b680aca764ec903c2d/ionic-3-and-angular-4-mobile-app-example)
- Ionic 3 and Angular 5 Mobile App Example (/post/59fc9da680aca7739224ee20/ionic-3-and-angular-5-mobile-app-example)
- Ionic 3, Angular 4 and SQLite CRUD Offline Mobile App (/post/59c53a1280aca768e4d2b143/ionic-3-angular-4-and-sqlite-crud-offline-mobile-app)
- How to Upload File on Ionic 3 using Native File Transfer Plugin (/post/599da16580aca768e4d2b130/how-to-upload-file-on-ionic-3-using-native-file-transfer-plugin)
- Build Ionic 3, Angular 5 and Firebase Simple Chat App (/post/5a629d9880aca7059c142976/build-ionic-3-angular-5-and-firebase-simple-chat-app)
- Step by Step Tutorial of Ionic 3, Angular 4 and Google Maps Directions Service (/post/58f4da2080aca7414e78a638/step-by-step-tutorial-of-ionic-3-angular-4-and-google-maps-directions-service)
- Ionic 3 and Angular 5 Search and Sort List of Data (/post/5a37ceaf80aca7059c142970/ionic-3-and-angular-5-search-and-sort-list-of-data)
- Ionic 3, Angular 5, Firebase and Google Maps Location Tracking (/post/5a48517280aca7059c142972/ionic-3-angular-5-firebase-and-google-maps-location-tracking)