# Basic CRUD Mean Stack Tutorial

MongoDB – NoSQL database program that uses JSON-like documents with schema to store the data.

ExpressJS - framework of Node.js and classified as middleware, to make writing the code easier. Helps to organize routing and use any templating solution with minimal effort.

AngularJS - front end framework. The interface see and experience. A client side scipt which helps you bind your model with scope.

NodeJS – Platform that allows JavaScript to be used outside the web browsers. Meaning that server and server-side for an application can be created, but using javaScript. backend of the app. Server which app our run

Restful API – how we interact data to database. An Application Program Interface (API) that uses HTTP request.

Tools

* MongoDB
  + MongoDB shell version: 3.2.10
* NodeJS - nodejs.org
  + once installed, on the project to work on, type: npm install express
* Git Bash - git-scm.com/downloads
* AngularJs
  + <https://angularjs.org/>
  + Version 1.5.8
* IDE
  + SublimeText - [www.sublimetext.com](http://www.sublimetext.com)
  + Visual Studio (1.7.1)

## Step by Step for contact list CRUD

Step 1: Go to nodeJS path, create node\_projects/contactlistapp directory.

Step 2: Create server.js under contactlistapp

Step 3: Open Git Bash, and go to contactlistapp directory and type the following:

|  |
| --- |
| npm install express |

Step 4: Modify server.js

|  |
| --- |
| var express = require(‘express’)  var app = express();  app.get('/', function(req, res) {  res.send('Hello world from server.js');  });  app.listen(3000);  console.log('server running on port 3000'); |

Step 5: Verify that it is working, by starting the node server. On Git Bash, contactlistapp directory, type:

|  |
| --- |
| node server |

Open the browser(I’m using Chrome), and navigate to localhost://3000. We can see the Hello World from server.js

Step 6: After validating that it is working as expected, modify the server.js and remove/comment the app.get code, and replace with:

|  |
| --- |
| /\* Initial code to verify that the server.js is working  app.get('/', function(req, res) {  res.send('Hello world from server.js');  });  \*/  //static means HTML and css, javascript.  app.use(express.static(\_\_dirname + "/public")); |

Step 7: Create index.html file under contactlistapp/public directory

Step 8: Edit the index.html, save, restart the node server and refresh the browser

|  |
| --- |
| <!DOCTYPE>  <html>  <head>  <title>Contact List App Header </title>  </head>  <body>  <h1>Contact List App</h1>  </body>  </html> |

Step 9: Just verify that after restarting node server, the index.html was displayed.

Step 10: Add the angular reference to our index.html. Copy the CDN link (angularJs.org), and on index.html file, add the script at the bottom of the body.

|  |
| --- |
| <!DOCTYPE>  <html ng-app>  <head>  <title>Contact List App Header </title>  </head>  <body>  <h1>Contact List App</h1>  <input ng-model="test"> {{test}}  <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script>  </body>  </html> |

NOTE: On code verify that the single/double quote are not a special character. If possible, re-type on your IDE the single / double quote.

Verify that it is working, by refreshing the browser (no need to restart). Try to type something on the text box, and verify that the typed text will be displayed beside the input box.

Step 11: Setup bootstrap for layout – getbootstrap.com, getting started, and copy the 2 links for css, and copy to head section of HTML document

|  |
| --- |
| <head>  <!-- CSS from getBootstrap -->  <!-- Latest compiled and minified CSS -->  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" >  <!-- Optional theme -->  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" >  <title>Contact List App</title>  </head> |

Test bootstrap, by refreshing the browser, to verify that the css was applied.

Step 12: Modify the index.html

|  |
| --- |
| <html ng-app="myApp">  <head>  <!-- CSS from getBootstrap -->  <!-- Latest compiled and minified CSS -->  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" >  <!-- Optional theme -->  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" >  <title>Contact List App</title>  </head>  <body>  <div class="container">  <h1>Contact List App</h1>  <table class="table">  <thead>  <tr>  <th>Name</th>  <th>Email</th>  <th>Number</th>  </tr>  <tr>  <td colspan="3"><input ng-model="test"> {{test}}</td>  </tr>  </thead>  </table>  </div>  </body>  </html> |

Step 13: Create controller – to control interaction between view and model

Modify index.html and add ng-controller on div class

|  |
| --- |
| <div class="container" ng-controller="AppCtrl"> |

Step 14: Create controller.js on contactlistapp/public/controllers directory.

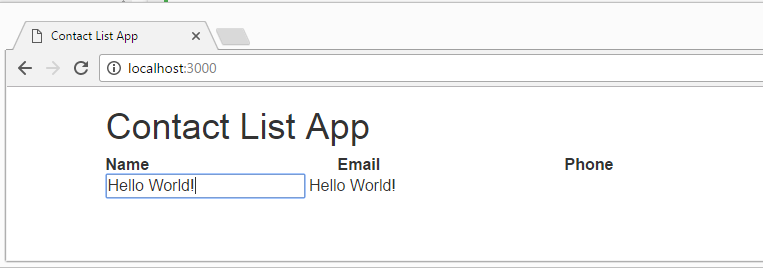
Include the controller.js on index before the end tag of the body

|  |
| --- |
| <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script>  <script src="controllers/controller.js"></script></body> |

Step 15: Open and modify the controller.js

|  |
| --- |
| var myApp = angular.module('myApp', []);  myApp.controller('AppCtrl', ['$scope', '$http', function($scope, $http) {  console.log("Hello World from controller");  }]); |

Verify on Chrome, more tools, then go to developers tool and go to console (or F12)



The page binds the input text, and display the typed value. In this example, Hello World!

Folder Structure

|  |
| --- |
| contactlistapp   * server.js * public   index.js  controllers   * + controller.js |

Compressed code for reference:



Part 2 – Continuation, we will add dummy data to be able to display on UI.

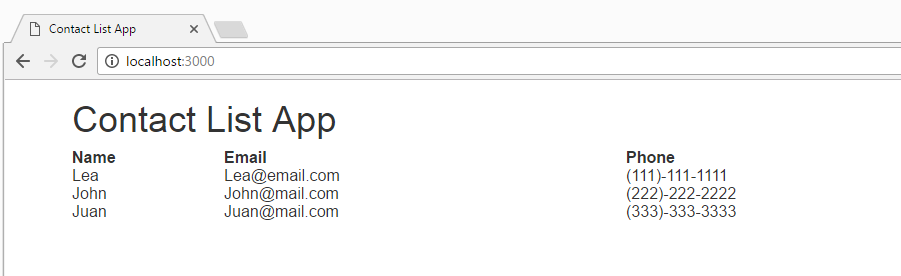
Step 16: Modify controller.js

|  |
| --- |
| var myApp = angular.module('myApp', []);  myApp.controller('AppCtrl', ['$scope', '$http', function($scope, $http) {  console.log("Hello World from controller");  var person1 = {  name: 'Lea',  email: 'Lea@email.com',  number: '(111)-111-1111'  };  var person2 = {  name: 'John',  email: 'John@mail.com',  number: '(222)-222-2222'  };  var person3 = {  name: 'Juan',  email: 'Juan@mail.com',  number: '(333)-333-3333'  };  $scope.contactlist = [person1, person2, person3]; //$scope allows us to use contact list on our UI  }]); |

Modify index.html

|  |
| --- |
| <tbody>  <tr ng-repeat=”contact in contactlist”>  <td>{{contact.name}}</td>  <td>{{contact.email}}</td>  <td>{{contact. number}}</td>  </tr>  </tbody> |

Then, restart the server and refresh the browser, to display the dummy data



Step 17: use GET request to retrieve dummy data from the server instead of controller. To do this, we use $http GET from controller side, which will send the request to the server and the app, and we will use the app.get from server side (server.js) which will respond with the data that we want.

Modify the controller.js file

|  |
| --- |
| var myApp = angular.module('myApp', []);  myApp.controller('AppCtrl', ['$scope', '$http', function($scope, $http) {  console.log("Hello World from controller");  $http.get(‘/contactlist’); //route that we create to get the data from  person 1 = {  name: ‘Tim’,  email: ‘Tim@email.com’,  number: ‘(111) 111-1111’  };  person2 = {  name: ‘Emily’,  email: ‘Emily@email.com’,  number: ‘(222) 222-2222’  };  person3 = {  name: ‘John’,  email: ‘John@email.com’,  number: ‘(333) 333-3333’  };  var contactlist = [person1, person2, person3];  $scope.contactlist = contactlist; //$scope allows us to use contact list on our UI  }]); |

Modify the server.js. Add the line below, after the app.use code

|  |
| --- |
| app.use . . .  app.get(‘/contactlist’, function(req, res) {  console.log(“I received a GET request”);  }); |

To verify if it is working, restart the server, and refresh the page.

On the console, we should see “I received a GET request” log

Step 18: Move the dummy data from controller.js to server.js for

controller.js – the console log, will be displayed on the browser

|  |
| --- |
| var myApp = angular.module('myApp', []);  myApp.controller('AppCtrl', ['$scope', '$http', function($scope, $http) {  console.log("Hello World from controller");  $http.get(‘/contactlist’); //route that we create to get the data from  ~~person 1 = {~~  ~~name: ‘Tim’,~~  ~~email: ‘Tim@email.com’,~~  ~~number: ‘(111) 111-1111’~~  ~~};~~  ~~person2 = {~~  ~~name: ‘Emily’,~~  ~~email: ‘Emily@email.com’,~~  ~~number: ‘(222) 222-2222’~~  ~~};~~  ~~person3 = {~~  ~~name: ‘John’,~~  ~~email: ‘John@email.com’,~~  ~~number: ‘(333) 333-3333’~~  ~~};~~  ~~var contactlist = [person1, person2, person3];~~  ~~$scope.contactlist = contactlist; //$scope allows us to use contact list on our UI -> DELETE tHIS~~  }]); |

Server.js

|  |
| --- |
| var express = require('express');  var app = express();  app.use(express.static(\_\_dirname + "/public"));  App.get(‘/contactlist’, function(req, res) {  Console.log(“I received a GET request”);  person 1 = {  name: ‘Tim’,  email: ‘Tim@email.com’,  number: ‘(111) 111-1111’  };  person2 = {  name: ‘Emily’,  email: ‘Emily@email.com’,  number: ‘(222) 222-2222’  };  person3 = {  name: ‘John’,  email: ‘John@email.com’,  number: ‘(333) 333-3333’  };  var contactlist = [person1, person2, person3];  //respond by sending json format that the controller can use  res.json(contactlist); //respond by sending json format that the controller can use  console.log('server.js res send: ' + res);  }  ); |

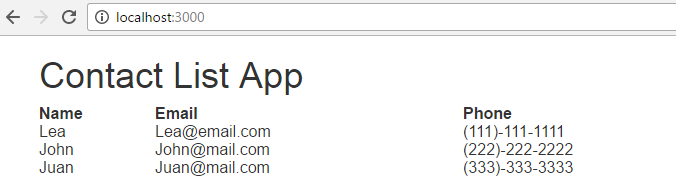
Go back and edit the controller

|  |
| --- |
| var myApp = angular.module('myApp', []);  myApp.controller('AppCtrl', ['$scope', '$http', function($scope, $http) {  console.log("Hello World from controller");  $http.get("/contactlist")  .then(function(response) {//route that we create to get the data from  $scope.contactList = response.data;  }, function (response){  //Second function handles error  console.log('Error - something went wrong');  });  console.log("end . . .");  }]); |

Index.html

|  |
| --- |
| <html ng-app="myApp">  <head>  <!-- CSS from getBootstrap -->  <!-- Latest compiled and minified CSS -->  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" >  <!-- Optional theme -->  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" >  <title>Contact List App</title>  </head>  <body>  <div class="container" ng-controller="AppCtrl">  <h1>Contact List App</h1>  <table class="table">  <thead>  <tr>  <th>Name</th>  <th>Email</th>  <th>Phone</th>  </tr>  </thead>    <tbody>  <tr ng-repeat="contact in contactList">  <td>{{contact.name}}</td>  <td>{{contact.email}}</td>  <td>{{contact.number}}</td>  </tr>  </tbody>      </table>  </div>      <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script>  <script src="controllers/controller.js"></script>  </body>  </html> |

Output





Note: Be careful when doing a copy paste, there might be a special character. If it is not working, might as well, re-type the code.

Part 3 -Continuation, connecting to MongoDB

Step 19: Create directory data/db on mongoDB. It depends where you want to store the database, but I store it inside the node directory. nodeJSDir/node\_projects/data/db

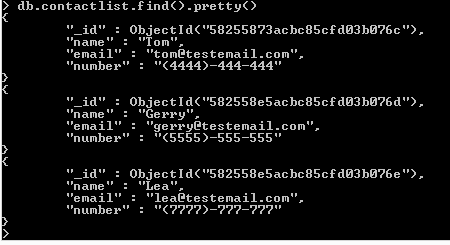
Step 20. Start mongo db (mongoDBDir/bin), on command prompt, type script below, and wait until you see waiting for connection on port 27017.

|  |
| --- |
| mongod --dbpath C:nodeJSDir/node\_projects/data/db |

You can also use RoboMongo to view the records with windows interface.

Step 21: Create initial data by opening another command prompt and navigate to mongo bin folder.

* Type mongo, to open the shell version
* Type show dbs
* Type use contactlist
* db.contactlist.insert({name: 'Tom', email: 'tom@testemail.com', number: '(4444)-444-444'})
* Then enter
* To view: db.contactlist.find().pretty()
* Add two more entries
* db.contactlist.insert([{name: 'Gerry', email: 'gerry@testemail.com', number: '(5555)-555-555'}, { name: 'Lea', email: 'lea@testemail.com', number: '(7777)-777-777'}])
* Query again the database:



Part 4 Continuation – Post and Get request

Step 22: On contactlistapp folder, type

npm install mongojs

Step 23: modify server.js

Add the code below

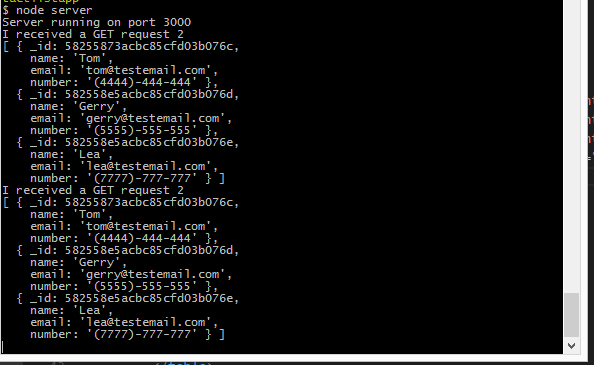
|  |
| --- |
| var mongojs = require('mongojs');  var db = mongojs('contactlist', ['contactlist']);//first param was module, next was the DB |

Step 24: delete the dummy data, and the data will be retrieved on mongodb

Inside app.get (server.js), delete the dummy data and add the code below:

|  |
| --- |
| app.get('/contactlist', function(req, res){  console.log("I received a GET request");  db.contactlist.find(function(err, doc){  console.log(docs);  res.json(docs); //respond by sending json format that the controller can use  });  }); |

Verify first by restarting the server, and verify on the log, that the data will be displayed on the console.



Step 25: modify the index.html to post/insert data

|  |
| --- |
| <table class="table">  <thead>  <tr>  <th>Name</th>  <th>Email</th>  <th>Phone</th>  <th>Action</th>  </tr>  </thead>    <tbody>  <tr>  <td><input class="form-control" ng-model="contact.name"/></td>  <td><input class="form-control" ng-model="contact.email"/></td>  <td><input class="form-control" ng-model="contact.number"/></td>  <td><button class="btn btn-primary" ng-click="addContact()">Add Contact</button></td>  </tr>  <tr ng-repeat="contact in contactList">  <td>{{contact.name}}</td>  <td>{{contact.email}}</td>  <td>{{contact.number}}</td>  </tr>  </tbody>  </table> |

Step 26: Add function to Add Contact button

Modify the controller.js and after the $http.get, add:

|  |
| --- |
| $scope.addContact = function() {  console.log($scope.contact);  }; |

Verify, restart the server, and check if it will display the log

This should be displayed on the browser

|  |
| --- |
| Object  email:"reynods@testemail.com"  name:"ryan"  number:"(8888)-888-888" |

Step 27: Modify the controller.js and add the function to add data from html

|  |
| --- |
| $scope.addContact = function() {  console.log(contact);  $http.post[‘/contactlist’, $scope.contact];//data we are sending to the server  }; |

Modify the server file.

Before app.listen, add the code below:

|  |
| --- |
| app.post('/contactlist', function(req, res){  console.log(req.body);//need body parser  }); |

Step 28: Install body parser first:

|  |
| --- |
| npm install body-parser |

Step 29: Add the code on server.js

|  |
| --- |
| var bodyParser = require('body-parser');  app.use(express.static(\_\_dirname + "/public"));  app.use(bodyParser.json()); |

Then restart the server, verify that the console and the browser display the log, the new record added.



Modify again the server.js to add the function to insert to mongodb ☺

//send the data into the database, and back to the controller

|  |
| --- |
| app.post('/contactlist', function(req, res){  console.log(req.body); //need body parser  db.contactlist.insert(req.body, function(){ //doc represent the item that we parse and receive  res.json(docs);//send the data back to controller  });  }); |

Step 30: Modify the controller.js, add the code below $http.get

|  |
| --- |
| $http.post('/contactlist', $scope.contact).then(function(response){  console.log(response);  }); |

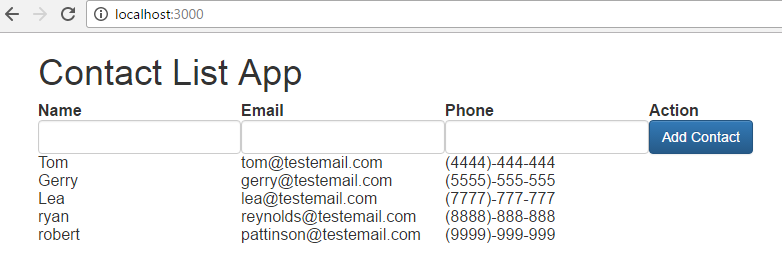
Test the code again to verify (restart and try to add new contact list).

After testing, the data is not yet automatically refresh. Reload the page to display the new insert data on the list.

Step 31. Refresh the page, after adding the data

Modify the controller.js

|  |
| --- |
| var refresh = function() {  $http.get('/contactlist')  .then(function(response) {//route that we create to get the data from  $scope.contactList = response.data;  }, function (response){  //Second function handles error  console.log('something went wrong');  });  };  refresh();  $http.post('/contactlist', $scope.contact).then(function(response){  console.log(response);  });  $scope.addContact = function() {  console.log($scope.contact);  $http.post('/contactlist', $scope.contact);// Data we are sending to the server  $scope.contact = "";  refresh(); // add the refresh() call  }; |





Part 5: Continuation (Delete and Update)

Step 32: Add remove and update button, modify the index.html

|  |
| --- |
| <table width=100%>  <thead>  <tr>  <th>Name</th>  <th>Email</th>  <th>Phone</th>  <th>Action</th>  <th>&nbsp;</th>  </tr>  </thead>    <tbody>  <tr>  <td><input class="form-control" ng-model="contact.name"/></td>  <td><input class="form-control" ng-model="contact.email"/></td>  <td><input class="form-control" ng-model="contact.number"/></td>  <td><button class="btn btn-primary" ng-click="addContact()">Add Contact</button></td>  <td><button class="btn btn-info" ng-click="updateContact()">Update Contact</button></td>  </tr>  <tr ng-repeat="contact in contactList">  <td>{{contact.name}}</td>  <td>{{contact.email}}</td>  <td>{{contact.number}}</td>  <td><button class="btn btn-danger" ng-click="removeContact()">Remove</button></td>  <td><button class="btn btn-warning" ng-click="edit(contact.\_id)">Edit</button></td>  </tr>  </tbody>  </table> |

Controller.js

|  |
| --- |
| $scope.removeContact = function(id){  console.log('removing contact id: ' + id);  } |

Restart the server, and verify that the id was displayed on the browser.

Step 33: Add the line of code on the newly created remove function, so it will be push on the server.js

|  |
| --- |
| $scope.removeContact = function(id){  console.log('removing contact id: ' + id);  $http.delete('/contactlist/' + id);  } |

Modify the server.js, add the code after the app.post

|  |
| --- |
| app.delete('/contactlist/:id', function(req, res) {  var id = req.params.id;  console.log('delete from server id: ' + id);  }); |

Test again, the id should be displayed on console log

Step 34: Delete the contact on database.

Modify the server.js

|  |
| --- |
| app.delete('/contactlist/:id', function(req, res) {  var id = req.params.id;  console.log('delete from server id: ' + id);  db.contactlist.remove({\_id:mongojs.ObjectId(id)}, function(err, docs){  res.json(docs);  });  }); |

Modify the controller.js

|  |
| --- |
| $scope.removeContact = function(id){  console.log('removing contact id: ' + id);  $http.delete('/contactlist/' + id).then(function(response){  refresh();  });  } |

Test again

Step 35: Update/PUT request

We already updated the HTML when we add the remove button.

Step 36:

Modify controller.js

|  |
| --- |
| $scope.edit = function(id) { //set the value of edit on the input box  console.log('editing contact: ' + id);  $http.get('/contactlist/' + id;  } |

Modify the server.js

|  |
| --- |
| app.get('/contactlist/:id', function(req, res){  var id = req.params.id;  console.log('server log id: ' + id);  db.contactlist.findOne({\_id:mongojs.ObjectId(id)}, function(err, docs){  console.log('server log findOne: ' + docs);  res.json(docs);  });  }); |

controller.js

Set the $scope.contact the value of response

|  |
| --- |
| $scope.edit = function(id) { //set the value of edit on the input box  console.log('editing contact: ' + id);  $http.get('/contactlist/' + id).then(function(response){    $scope.contact = response.data;    console.log('editing contact response: ' + response.data.email);  });  } |

Modify controller.js and add the function to update or save the value on the form

|  |
| --- |
| $scope.updateContact = function() {  console.log('updating contact: ' + $scope.contact.name);  } |

Test, then add the code under console.log

|  |
| --- |
| $scope.updateContact = function() {  console.log('updating contact: ' + $scope.contact.\_id);  $http.put('/contactlist/' + $scope.contact.\_id, $scope.contact);  } |

Server.js

|  |
| --- |
| app.put('/contactlist/:id', function(req, res){  var id = req.params.id;  console.log("server app put: " + id );  }); |

Test and verify again.

Step 37: Add code to retrieve and update contact from mongoDB.

Modify the server.js

|  |
| --- |
| app.put('/contactlist/:id', function(req, res){  var id = req.params.id;  console.log("server app put: " + id );  db.contactlist.findAndModify({query:{\_id:mongojs.ObjectId(id)},  update:{$set:{name:req.body.name, email: req.body.email, number:req.body.number}},  new: true},  function(err, docs){  res.json(docs);  });  }); |

On Controller.js, add refresh page

|  |
| --- |
| $scope.updateContact = function() {  console.log('updating contact: ' + $scope.contact.\_id);  $http.put('/contactlist/' + $scope.contact.\_id, $scope.contact).then(function(response){  refresh();  });  } |

Step 38: Test by restarting the server. We are almost done. Just did some minor modification.

Added some toggle button, final code for reference:

Controller.js

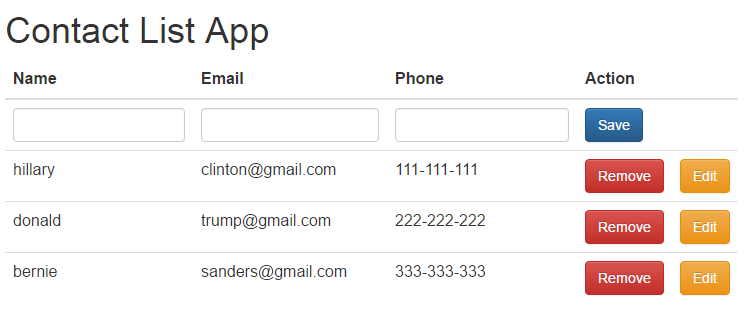
|  |
| --- |
| var myApp = angular.module('myApp', []);  myApp.controller('AppCtrl', ['$scope', '$http', function($scope, $http) {  console.log("Controller . . .");  var refresh = function() {  console.log('refresh was called . . .');  $http.get('/contactlist')  .then(function(response) {//route that we create to get the data from  $scope.contactList = response.data;  $scope.contact = "";  //buttons  $scope.toggleAddBtn = true;  $scope.toggleCancelBtn = false;  $scope.toggleUpdateBtn = false;  console.log('controller retrieve contactlist');  }, function (response){  //Second function handles error  console.log('Error - something went wrong');  });  };  refresh();  $scope.cancelBtnClick = function() {  refresh();  }  /\*  $http.post('/contactlist', $scope.contact).then(function(response){  console.log(response);  });  \*/  $scope.addContact = function() {  console.log('adding contact' + $scope.myForm.$valid);  if ($scope.myForm.$valid) {  $http.post('/contactlist', $scope.contact);// Data we are sending to the server  refresh();  }    };  $scope.removeContact = function(id){  console.log('removing contact id: ' + id);  $http.delete('/contactlist/' + id).then(function(response){  refresh();  });  }  $scope.edit = function(id) { //set the value of edit on the input box  console.log('editing contact: ' + id);  $http.get('/contactlist/' + id).then(function(response){    $scope.contact = response.data;  //buttons  $scope.toggleAddBtn = false;  $scope.toggleCancelBtn = true;  $scope.toggleUpdateBtn = true;  });  }  $scope.updateContact = function() {  console.log('updating contact: ' + $scope.contact.\_id);  $http.put('/contactlist/' + $scope.contact.\_id, $scope.contact).then(function(response){  //buttons  $scope.toggleAddBtn = true;  $scope.toggleCancelBtn = false;  $scope.toggleUpdateBtn = false;  refresh();  });  }  console.log("done . . .");  }]); |

Server.js

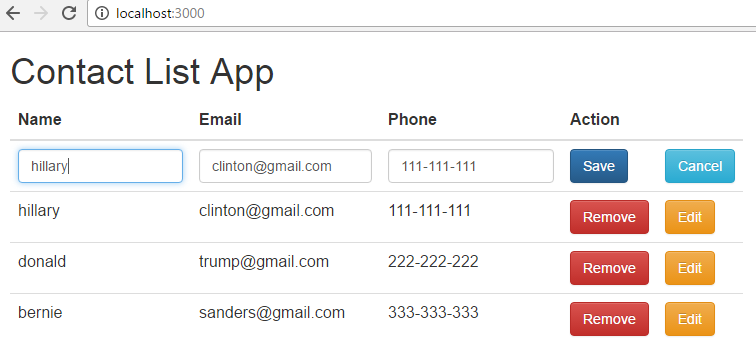
|  |
| --- |
| var express = require('express');  var app = express();  var mongojs = require('mongojs');  var db = mongojs('contactlist', ['contactlist']);//first param was module, next was the DB  var bodyParser = require('body-parser');  /\* Initial code to verify that the server.js is working  app.get('/', function(req, res) {  res.send('Hello world from server.js');  });  \*/  app.use(express.static(\_\_dirname + "/public"));  app.use(bodyParser.json());  app.get('/contactlist', function(req, res){  db.contactlist.find(function(err, docs){  res.json(docs); //respond by sending json format that the controller can use  });  });  app.post('/contactlist', function(req, res){  console.log(req.body); //need body parser  db.contactlist.insert(req.body, function(docs){ //docs represent the item that we parse and receive  res.json(docs);//send the data back to controller  });  });  app.delete('/contactlist/:id', function(req, res) {  var id = req.params.id;  db.contactlist.remove({\_id:mongojs.ObjectId(id)}, function(err, docs){  res.json(docs);  });  });  app.get('/contactlist/:id', function(req, res){  var id = req.params.id;  db.contactlist.findOne({\_id:mongojs.ObjectId(id)}, function(err, docs){  console.log('server log findOne: ' + docs);  res.json(docs);  });  });  app.put('/contactlist/:id', function(req, res){  var id = req.params.id;  db.contactlist.findAndModify({query:{\_id:mongojs.ObjectId(id)},  update:{$set:{name:req.body.name, email: req.body.email, number:req.body.number}},  new: true},  function(err, docs){  res.json(docs);  });  });  app.listen(3000);  console.log('Server running on port 3000'); |

Index.html

|  |
| --- |
| <html ng-app="myApp">  <head>  <!-- CSS from getBootstrap -->  <!-- Latest compiled and minified CSS -->  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" >  <!-- Optional theme -->  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap-theme.min.css" >  <title>Contact List App</title>  </head>  <body>  <div class="container" ng-controller="AppCtrl">  <h1>Contact List App</h1>  <form name="myForm">  <table class="table">  <thead>  <tr>  <th>Name</th>  <th>Email</th>  <th>Phone</th>  <th>Action</th>  <th>&nbsp;</th>  </tr>  </thead>    <tbody>  <tr>  <td><input class="form-control" id="nameVal" ng-model="contact.name" ng-minlength="1" required></td>  <td><input class="form-control" ng-model="contact.email" value="{{contact.email}}"></td>  <td><input class="form-control" ng-model="contact.number"></td>  <td><button class="btn btn-primary" ng-click="addContact()" ng-show="toggleAddBtn">Save</button>  <button class="btn btn-primary" ng-click="updateContact()" ng-show="toggleUpdateBtn">Save</button>  </td>  <td><button class="btn btn-info" ng-click="cancelBtnClick()" ng-show="toggleCancelBtn">Cancel</button></td>  </tr>  <tr ng-repeat="contact in contactList">  <td>{{contact.name}}</td>  <td>{{contact.email}}</td>  <td>{{contact.number}}</td>  <td><button class="btn btn-danger" ng-click="removeContact(contact.\_id)">Remove</button></td>  <td><button class="btn btn-warning" ng-click="edit(contact.\_id)">Edit</button></td>  </tr>  </tbody>  </table>  </form>  </div>      <script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.8/angular.min.js"></script>  <script src="controllers/controller.js"></script>  </body>  </html> |



Edit/Update



Commands:

Start mongodb server

mongod --dbpath <absolute dbPath>

start node server:

node server

Reference:

https://www.quora.com/What-is-Express-js-and-Angular-js

https://www.youtube.com/watch?v=06\_SIzYXgqQ