**AngularJS + REST API**

Sommaire

[1. Sources 1](#_Toc406508030)

[2. Etapes 1](#_Toc406508031)

[3. Getting Started 2](#_Toc406508032)

[a. Defining our Node Packages package.json 2](#_Toc406508033)

[b. Installing Our Node Packages 2](#_Toc406508034)

[c. Setting Up Our Server server.js 2](#_Toc406508035)

[4. Starting Our Server and Testing 2](#_Toc406508036)

[5. Database and Bear Model 2](#_Toc406508037)

[a. Creating Our Database and Connecting 2](#_Toc406508038)

[b. Server.js 2](#_Toc406508039)

[c. Bear Model app/models/bear.js 2](#_Toc406508040)

[d. Server.js 2](#_Toc406508041)

[6. Route Middleware 3](#_Toc406508042)

[7. Creating the Basic Routes 3](#_Toc406508043)

[a. Express Router and Routes 3](#_Toc406508044)

[b. Creating a Bear POST /api/bears 3](#_Toc406508045)

[8. Creating Routes for A Single Item 4](#_Toc406508046)

[a. Getting a Single Bear GET /api/bears/:bear\_id 4](#_Toc406508047)

[b. Updating a Bear’s Info PUT /api/bears/:bear\_id 4](#_Toc406508048)

[c. Deleting a Bear DELETE /api/bears/:bear\_id 5](#_Toc406508049)

# ****Sources****

<http://java.dzone.com/articles/angularjs-single-page-app>

<http://www.sitepoint.com/creating-crud-app-minutes-angulars-resource/>

<http://translate.google.fr/translate?hl=en&sl=zh-TW&u=http://blog.amowu.com/2014/03/angularjs-google-spreadsheet.html&prev=search>

# ****Etapes****

**Base Setup** In our base setup, we pull in all the packages we pulled in using npm. We’ll grab express, define our app, get bodyParser and configure our app to use it. We can also set the port for our application.

**Routes for Our API** This section will hold all of our routes. The structure for using the Express Router let’s us pull in an instance of the router. We can then **define routes** and then **apply those routes** to a root URL (in this case, API).

**Start our Server** We’ll have our express app listen to the port we defined earlier. Then our application will be live and we can test it!

# ****Getting Started****

## Defining our Node Packages package.json

## Installing Our Node Packages

## Setting Up Our Server server.js

# ****Starting Our Server and Testing****

# ****Database and Bear Model****

## Creating Our Database and Connecting

<http://modulus.io/>

<https://mongolab.com/>

## Server.js

// server.js

// BASE SETUP

// ===========================================================================

var mongoose = require('mongoose');

mongoose.connect('mongodb://node:node@novus.modulusmongo.net:27017/Iganiq8o'); // connect to our database

## Bear Model app/models/bear.js

// app/models/bear.js

var mongoose = require('mongoose');

var Schema = mongoose.Schema;

var BearSchema = new Schema({

name: String

});

module.exports = mongoose.model('Bear', BearSchema);

## Server.js

// server.js

// BASE SETUP

// ===========================================================================

var Bear = require('./app/models/bear');

# ****Route Middleware****

* Authentification

**Middleware Uses** Using middleware like this can be very powerful. We can do validations to make sure that everything coming from a request is safe and sound. We can throw errors here in case something is wrong. We can do some extra logging for analytics or any statistics we’d like to keep. There are many possibilities here. Go wild.

With middleware, we can do awesome things to requests coming into our API. We will probably want to make sure that the user is authenticated to access our API. We’ll go over that in a future article, but for now let’s just log something to the console with our middleware.

// server.js

// middleware to use for all requests

router.use(function(req, res, next) {

// do logging

console.log('Something is happening.');

next(); // make sure we go to the next routes and don't stop here

});

# ****Creating the Basic Routes****

## Express Router and Routes

| Route | HTTP Verb | Description |
| --- | --- | --- |
| /api/bears | GET | Get all the bears. |
| /api/bears | POST | Create a bear. |
| /api/bears/:bear\_id | GET | Get a single bear. |
| /api/bears/:bear\_id | PUT | Update a bear with new info. |
| /api/bears/:bear\_id | DELETE | Delete a bear. |

## Creating a Bear POST /api/bears

router.route('/bears')

// create a bear (accessed at POST http://localhost:8080/api/bears)

.post(function(req, res) {

var bear = new Bear(); // create a new instance of the Bear model

bear.name = req.body.name; // set the bears name (comes from the request)

// save the bear and check for errors

bear.save(function(err) {

if (err)

res.send(err);

res.json({ message: 'Bear created!' });

});

})

// get all the bears (accessed at GET http://localhost:8080/api/bears)

.get(function(req, res) {

Bear.find(function(err, bears) {

if (err)

res.send(err);

res.json(bears);

});

});

# ****Creating Routes for A Single Item****

## Getting a Single Bear GET /api/bears/:bear\_id

// on routes that end in /bears/:bear\_id

// ----------------------------------------------------

router.route('/bears/:bear\_id')

// get the bear with that id (accessed at GET http://localhost:8080/api/bears/:bear\_id)

.get(function(req, res) {

Bear.findById(req.params.bear\_id, function(err, bear) {

if (err)

res.send(err);

res.json(bear);

});

});

## Updating a Bear’s Info PUT /api/bears/:bear\_id

// update the bear with this id (accessed at PUT http://localhost:8080/api/bears/:bear\_id)

.put(function(req, res) {

// use our bear model to find the bear we want

Bear.findById(req.params.bear\_id, function(err, bear) {

if (err)

res.send(err);

bear.name = req.body.name; // update the bears info

// save the bear

bear.save(function(err) {

if (err)

res.send(err);

res.json({ message: 'Bear updated!' });

});

});

});

## Deleting a Bear DELETE /api/bears/:bear\_id

// delete the bear with this id (accessed at DELETE http://localhost:8080/api/bears/:bear\_id)

.delete(function(req, res) {

Bear.remove({

\_id: req.params.bear\_id

}, function(err, bear) {

if (err)

res.send(err);

res.json({ message: 'Successfully deleted' });

});

});