### **REST API with Express, MongoDB and Mongoose Part 1**

#### Objectives and Outcomes

In this exercise, you will integrate the REST API server based on the Express framework that you implemented earlier, together with the Mongoose schema and models to create a full-fledged REST API server. At the end of this exercise, you will be able to:

* Develop a full-fledged REST API server with Express, MongoDB and Mongoose
* Serve up various REST API end points together with interaction with the MongoDB server.

#### Exercise Resources

<db.json>

#### Update the Express Application

* Go to the conFusionServer folder where you had developed the REST API server using Express generator.
* Copy the models folder from the node-mongoose folder to the conFusionServer folder.
* Then install bluebird, mongoose and mongoose-currency Node modules by typing the following at the prompt:

     npm install mongoose mongoose-currency --legacy-peer-deps

* Open app.js file and add in the code to connect to the MongoDB server as follows:
* . . .
* const mongoose = require('mongoose');
* const Dishes = require('./models/dishes');
* const url = 'mongodb://localhost:27017/conFusion';
* const connect = mongoose.connect(url);
* connect.then((db) => {
* console.log("Connected correctly to server");
* }, (err) => { console.log(err); });
* . . .
* Next open dishes.js in the models folder and update it as follows:
* . . .
* require('mongoose-currency').loadType(mongoose);
* const Currency = mongoose.Types.Currency;
* . . .
* const dishSchema = new Schema({
* name: {
* type: String,
* required: true,
* unique: true
* },
* description: {
* type: String,
* required: true
* },
* image: {
* type: String,
* required: true
* },
* category: {
* type: String,
* required: true
* },
* label: {
* type: String,
* default: ''
* },
* price: {
* type: Currency,
* required: true,
* min: 0
* },
* featured: {
* type: Boolean,
* default:false
* },
* comments:[commentSchema]
* }, {
* timestamps: true
* });
* . . .
* Now open dishRouter.js and update its code as follows:
* const express = require('express');
* const bodyParser = require('body-parser');
* const mongoose = require('mongoose');
* const Dishes = require('../models/dishes');
* const dishRouter = express.Router();
* dishRouter.use(bodyParser.json());
* dishRouter.route('/')
* .get((req,res,next) => {
* Dishes.find({})
* .then((dishes) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dishes);
* }, (err) => next(err))
* .catch((err) => next(err));
* })
* .post((req, res, next) => {
* Dishes.create(req.body)
* .then((dish) => {
* console.log('Dish Created ', dish);
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dish);
* }, (err) => next(err))
* .catch((err) => next(err));
* })
* .put((req, res, next) => {
* res.statusCode = 403;
* res.end('PUT operation not supported on /dishes');
* })
* .delete((req, res, next) => {
* Dishes.remove({})
* .then((resp) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(resp);
* }, (err) => next(err))
* .catch((err) => next(err));
* });
* dishRouter.route('/:dishId')
* .get((req,res,next) => {
* Dishes.findById(req.params.dishId)
* .then((dish) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dish);
* }, (err) => next(err))
* .catch((err) => next(err));
* })
* .post((req, res, next) => {
* res.statusCode = 403;
* res.end('POST operation not supported on /dishes/'+ req.params.dishId);
* })
* .put((req, res, next) => {
* Dishes.findByIdAndUpdate(req.params.dishId, {
* $set: req.body
* }, { new: true })
* .then((dish) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dish);
* }, (err) => next(err))
* .catch((err) => next(err));
* })
* .delete((req, res, next) => {
* Dishes.findByIdAndRemove(req.params.dishId)
* .then((resp) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(resp);
* }, (err) => next(err))
* .catch((err) => next(err));
* });
* module.exports = dishRouter;
* Save the changes and start the server. Make sure your MongoDB server is up and running.
* You can now fire up postman and then perform several operations on the REST API. You can use the data for all the dishes provided in the db.json file given above in the Exercise Resources to test your server
* Do a Git commit with the message "Express REST API with MongoDB and Mongoose Part 1"

#### Conclusions

In this exercise you developed a full-fledged REST API server with Express, Mongo and Mongoose.