Exercise (Instructions): REST API with Express, MongoDB and Mongoose

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 that you developed as part of Assignment 2 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.

Scaffold out an Express Application

- Scaffold out an Express application named rest-server using the Express generator at a convenient location on your computer by typing the following at the prompt:

```
     express rest-server
```

- Now you will copy in some of the code you developed in the Express-generator exercise in the previous module in the *node-express-gen* folder.
- Go to *node-express-gen* folder and copy the *app.js* file into the *rest-server* folder.
- From the *node-express-gen/routes* folder, copy the *dishRouter.js*, *promoRouter.js* and *leaderRouter.js* files to the *rest-server/routes* folder.
- Copy the *models* folder from your Assignment 2 solution to the *rest-server* folder.
- First do an npm install in the *rest-server* folder to install all the modules. Install mongoose and mongoose-currency Node modules by typing the following at the prompt:

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

- Open app.js file and add in the code to connect to the MongoDB server as follows:

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


var url = 'mongodb://localhost:27017/conFusion';
mongoose.connect(url);
var db = mongoose.connection;
```

```
db.on('error', console.error.bind(console, 'connection error:'));
db.once('open', function () {
    // we're connected!
    console.log("Connected correctly to server");
});
```

- Now open *dishRouter.js* and update its code as follows:

```
var express = require('express');
var bodyParser = require('body-parser');
var mongoose = require('mongoose');

var Dishes = require('../models/dishes');

var dishRouter = express.Router();
dishRouter.use(bodyParser.json());

dishRouter.route('/')
.get(function (req, res, next) {
    Dishes.find({}, function (err, dish) {
        if (err) throw err;
        res.json(dish);
    });
})

.post(function (req, res, next) {
    Dishes.create(req.body, function (err, dish) {
        if (err) throw err;
        console.log('Dish created!');
        var id = dish._id;

        res.writeHead(200, {
            'Content-Type': 'text/plain'
        });
        res.end('Added the dish with id: ' + id);
    });
})
```

```
.delete(function (req, res, next) {
    Dishes.remove({}, function (err, resp) {
        if (err) throw err;
        res.json(resp);
    });
});

dishRouter.route('/:dishId')
.get(function (req, res, next) {
    Dishes.findById(req.params.dishId, function (err, dish) {
        if (err) throw err;
        res.json(dish);
    });
})

.put(function (req, res, next) {
    Dishes.findByIdAndUpdate(req.params.dishId, {
        $set: req.body
    }, {
        new: true
    }, function (err, dish) {
        if (err) throw err;
        res.json(dish);
    });
})

.delete(function (req, res, next) {
    Dishes.findByIdAndRemove(req.params.dishId, function (err, resp) {        if (err) th
row err;
        res.json(resp);
    });
});
```

- Do a similar update to *promoRouter.js* and *leaderRouter.js* to support the REST API operations on promotions and leadership information.

Handling Comments

- Add the following code to *dishRouter.js* to handle comments:

```
dishRouter.route('/:dishId/comments')
.get(function (req, res, next) {
    Dishes.findById(req.params.dishId, function (err, dish) {
        if (err) throw err;
        res.json(dish.comments);
    });
})

.post(function (req, res, next) {
    Dishes.findById(req.params.dishId, function (err, dish) {
        if (err) throw err;
        dish.comments.push(req.body);
        dish.save(function (err, dish) {
            if (err) throw err;
            console.log('Updated Comments!');
            res.json(dish);
        });
    });
})

.delete(function (req, res, next) {
    Dishes.findById(req.params.dishId, function (err, dish) {
        if (err) throw err;
        for (var i = (dish.comments.length - 1); i >= 0; i--) {
            dish.comments.id(dish.comments[i]._id).remove();
        }
        dish.save(function (err, result) {
            if (err) throw err;
            res.writeHead(200, {
                'Content-Type': 'text/plain'
            });
            res.end('Deleted all comments!');
        });
    });
});

dishRouter.route('/:dishId/comments/:commentId')
.get(function (req, res, next) {
    Dishes.findById(req.params.dishId, function (err, dish) {
        if (err) throw err;
        res.json(dish.comments.id(req.params.commentId));
    });
})

.put(function (req, res, next) {
    // We delete the existing commment and insert the updated
    // comment as a new comment
    Dishes.findById(req.params.dishId, function (err, dish) {
        if (err) throw err;
        dish.comments.id(req.params.commentId).remove();
        dish.comments.push(req.body);
        dish.save(function (err, dish) {
            if (err) throw err;
            console.log('Updated Comments!');
            res.json(dish);
        });
    });
})

.delete(function (req, res, next) {
    Dishes.findById(req.params.dishId, function (err, dish) {
        dish.comments.id(req.params.commentId).remove();
        dish.save(function (err, resp) {
            if (err) throw err;
            res.json(resp);
        });
    });
});

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, promotions and leadership provided in the db.json file given below to test your server:

```
db.json
```

Conclusions

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