### **REST API with Express, MongoDB and Mongoose Part 2**

#### Objectives and Outcomes

In this exercise, you will continue the integration of 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:

* Add support for accessing and updating comments within the dishes.

#### Exercise Resources

<db.json>

#### Handling Comments

* Add the following code to dishRouter.js to handle comments:
* . . .
* dishRouter.route('/:dishId/comments')
* .get((req,res,next) => {
* Dishes.findById(req.params.dishId)
* .then((dish) => {
* if (dish != null) {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dish.comments);
* }
* else {
* err = new Error('Dish ' + req.params.dishId + ' not found');
* err.status = 404;
* return next(err);
* }
* }, (err) => next(err))
* .catch((err) => next(err));
* })
* .post((req, res, next) => {
* Dishes.findById(req.params.dishId)
* .then((dish) => {
* if (dish != null) {
* dish.comments.push(req.body);
* dish.save()
* .then((dish) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dish);
* }, (err) => next(err));
* }
* else {
* err = new Error('Dish ' + req.params.dishId + ' not found');
* err.status = 404;
* return next(err);
* }
* }, (err) => next(err))
* .catch((err) => next(err));
* })
* .put((req, res, next) => {
* res.statusCode = 403;
* res.end('PUT operation not supported on /dishes/'
* + req.params.dishId + '/comments');
* })
* .delete((req, res, next) => {
* Dishes.findById(req.params.dishId)
* .then((dish) => {
* if (dish != null) {
* for (var i = (dish.comments.length -1); i >= 0; i--) {
* dish.comments.id(dish.comments[i].\_id).remove();
* }
* dish.save()
* .then((dish) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dish);
* }, (err) => next(err));
* }
* else {
* err = new Error('Dish ' + req.params.dishId + ' not found');
* err.status = 404;
* return next(err);
* }
* }, (err) => next(err))
* .catch((err) => next(err));
* });
* dishRouter.route('/:dishId/comments/:commentId')
* .get((req,res,next) => {
* Dishes.findById(req.params.dishId)
* .then((dish) => {
* if (dish != null && dish.comments.id(req.params.commentId) != null) {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dish.comments.id(req.params.commentId));
* }
* else if (dish == null) {
* err = new Error('Dish ' + req.params.dishId + ' not found');
* err.status = 404;
* return next(err);
* }
* else {
* err = new Error('Comment ' + req.params.commentId + ' not found');
* err.status = 404;
* return next(err);
* }
* }, (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
* + '/comments/' + req.params.commentId);
* })
* .put((req, res, next) => {
* Dishes.findById(req.params.dishId)
* .then((dish) => {
* if (dish != null && dish.comments.id(req.params.commentId) != null) {
* if (req.body.rating) {
* dish.comments.id(req.params.commentId).rating = req.body.rating;
* }
* if (req.body.comment) {
* dish.comments.id(req.params.commentId).comment = req.body.comment;
* }
* dish.save()
* .then((dish) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dish);
* }, (err) => next(err));
* }
* else if (dish == null) {
* err = new Error('Dish ' + req.params.dishId + ' not found');
* err.status = 404;
* return next(err);
* }
* else {
* err = new Error('Comment ' + req.params.commentId + ' not found');
* err.status = 404;
* return next(err);
* }
* }, (err) => next(err))
* .catch((err) => next(err));
* })
* .delete((req, res, next) => {
* Dishes.findById(req.params.dishId)
* .then((dish) => {
* if (dish != null && dish.comments.id(req.params.commentId) != null) {
* dish.comments.id(req.params.commentId).remove();
* dish.save()
* .then((dish) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dish);
* }, (err) => next(err));
* }
* else if (dish == null) {
* err = new Error('Dish ' + req.params.dishId + ' not found');
* err.status = 404;
* return next(err);
* }
* else {
* err = new Error('Comment ' + req.params.commentId + ' not found');
* err.status = 404;
* return next(err);
* }
* }, (err) => next(err))
* .catch((err) => next(err));
* });
* . . .
* 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 2".

#### Conclusions

In this exercise you continued to develop the full-fledged REST API server with Express, Mongo and Mongoose.