### **Mongoose Population**

#### Objectives and Outcomes

In this exercise you will learn about how you can reference one document from another in MongoDB. In addition you will use Mongoose population to populate information in a document from a referenced document. At the end of this exercise, you will be able to:

* Use Mongoose population to cross-reference users within a comment
* Populate the comments with the users’ information upon querying

#### Using Mongoose Population

* Open user.js and update the code for the schema as follows:
* . . .
* var User = new Schema({
* firstname: {
* type: String,
* default: ''
* },
* lastname: {
* type: String,
* default: ''
* },
* admin: {
* type: Boolean,
* default: false
* }
* });
* . . .
* Open dishes.js and update the comment schema as follows:
* . . .
* var commentSchema = new Schema({
* rating: {
* type: Number,
* min: 1,
* max: 5,
* required: true
* },
* comment: {
* type: String,
* required: true
* },
* author: {
* type: mongoose.Schema.Types.ObjectId,
* ref: 'User'
* }
* }, {
* timestamps: true
* });
* . . .
* Open dishRouter.js and update the routers and the methods as shown below:
* . . .
* dishRouter.route('/')
* .get((req,res,next) => {
* Dishes.find({})
* .populate('comments.author')
* .then((dishes) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dishes);
* }, (err) => next(err))
* .catch((err) => next(err));
* })
* . . .
* dishRouter.route('/:dishId')
* .get((req,res,next) => {
* Dishes.findById(req.params.dishId)
* .populate('comments.author')
* .then((dish) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(dish);
* }, (err) => next(err))
* .catch((err) => next(err));
* })
* . . .
* dishRouter.route('/:dishId/comments')
* .get((req,res,next) => {
* Dishes.findById(req.params.dishId)
* .populate('comments.author')
* .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(authenticate.verifyUser, (req, res, next) => {
* Dishes.findById(req.params.dishId)
* .then((dish) => {
* if (dish != null) {
* req.body.author = req.user.\_id;
* dish.comments.push(req.body);
* dish.save()
* .then((dish) => {
* Dishes.findById(dish.\_id)
* .populate('comments.author')
* .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)
* .populate('comments.author')
* .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));
* })
* . . .
* .put(cors.corsWithOptions, authenticate.verifyUser, (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) => {
* Dishes.findById(dish.\_id)
* .populate('comments.author')
* .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(cors.corsWithOptions, authenticate.verifyUser, (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) => {
* Dishes.findById(dish.\_id)
* .populate('comments.author')
* .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));
* });
* . . .
* Open users.js and update the register function as follows:
* . . .
* router.post('/signup', (req, res, next) => {
* User.register(new User({username: req.body.username}),
* req.body.password, (err, user) => {
* if(err) {
* res.statusCode = 500;
* res.setHeader('Content-Type', 'application/json');
* res.json({err: err});
* }
* else {
* if (req.body.firstname)
* user.firstname = req.body.firstname;
* if (req.body.lastname)
* user.lastname = req.body.lastname;
* user.save((err, user) => {
* if (err) {
* res.statusCode = 500;
* res.setHeader('Content-Type', 'application/json');
* res.json({err: err});
* return ;
* }
* passport.authenticate('local')(req, res, () => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json({success: true, status: 'Registration Successful!'});
* });
* });
* }
* });
* });
* . . .
* Save all the changes and start the server and test.
* If you happen to have a registered user in your MongoDB, then use the Mongo REPL to delete the user.
* Do a Git commit with the message "Mongoose Population".

#### Conclusions

In this exercise you learnt about cross-referencing Mongo documents using the ObjectId. In addition you learnt about using Mongoose population support for populating documents.