Exercise (Instructions): 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

Update User Schema

- Open *user.js* and update the code for the schema as follows:

```
var User = new Schema({
    username: String,
    password: String,
    firstname: {
      type: String,
        default: ''
    },
    lastname: {
      type: String,
        default: ''
    },
    admin:   {
        type: Boolean,
        default: false
    }
});

User.methods.getName = function() {
    return (this.firstname + ' ' + this.lastname);
};
```

Updating Dish Schema

- 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
    },
    postedBy: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'User'
    }
}, {
    timestamps: true
});
```

Updating dishRouter.js

- Open dishRouter.js and update the routers and the methods as shown below:

```
dishRouter.route('/')
.get(Verify.verifyOrdinaryUser, function (req, res, next) {
    Dishes.find({})
        .populate('comments.postedBy')
        .exec(function (err, dish) {
        if (err) throw err;
        res.json(dish);
    });
})

.post(Verify.verifyOrdinaryUser, Verify.verifyAdmin, 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(Verify.verifyOrdinaryUser, Verify.verifyAdmin, function (req, res, next) {
    Dishes.remove({}, function (err, resp) {
        if (err) throw err;
        res.json(resp);
    });
});

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

.put(Verify.verifyOrdinaryUser, Verify.verifyAdmin, 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(Verify.verifyOrdinaryUser, Verify.verifyAdmin, function (req, res, next) {
        Dishes.findByIdAndRemove(req.params.dishId, function (err, resp) {
        if (err) throw err;
        res.json(resp);
    });
});

dishRouter.route('/:dishId/comments')
.all(Verify.verifyOrdinaryUser)

.get(function (req, res, next) {
    Dishes.findById(req.params.dishId)
        .populate('comments.postedBy')
        .exec(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;
        req.body.postedBy = req.decoded._doc._id;
        dish.comments.push(req.body);
        dish.save(function (err, dish) {
            if (err) throw err;
            console.log('Updated Comments!');
            res.json(dish);
        });
    });
})

.delete(Verify.verifyAdmin, 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')
.all(Verify.verifyOrdinaryUser).get(function (req, res, next) {
    Dishes.findById(req.params.dishId)
        .populate('comments.postedBy')
        .exec(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();
                req.body.postedBy = req.decoded._doc._id;
        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 (dish.comments.id(req.params.commentId).postedBy
           != req.decoded._doc._id) {
            var err = new Error('You are not authorized to perform this operation!');
            err.status = 403;
            return next(err);
        }
        dish.comments.id(req.params.commentId).remove();
        dish.save(function (err, resp) {
            if (err) throw err;
            res.json(resp);
        });
    });
});
```

Updating users.js route

- Open users.js and update the register function as follows:

```
router.post('/register', function(req, res) {
    User.register(new User({ username : req.body.username }),
        req.body.password, function(err, user) {
        if (err) {
            return res.status(500).json({err: err});
        }
                if(req.body.firstname) {
            user.firstname = req.body.firstname;
        }
        if(req.body.lastname) {
            user.lastname = req.body.lastname;
        }
                user.save(function(err,user) {
            passport.authenticate('local')(req, res, function () {
                return res.status(200).json({status: 'Registration Successful!'});
```

```
            });
        });
    });
});
```

- Save all the changes and start the server and test.

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.