### **Integrating the React Client and Server: Updating the REST Server**

#### Objectives and Outcomes

In this exercise you will update the Express REST server that you designed earlier. You will update the server to allow users to query and retrieve featured dishes, promotions and leaders. In addition you will update the routes to allow users to do GET requests without authentication, but the remaining requests require user authentication. At the end of this exercise, you will be able to:

* Use query parameters within the URL to select items and return them in response to a query
* Add a Comment Router to serve the comments independent of the dishes. This is required to support the React client that accesses comments independent of the dishes.
* Enable users to retrieve some information without authentication and other routes with authentication

#### Exercise Resources

<db1.json>

<images.zip>

#### Adding Images to the Server

* Download the images.zip file provided above, unzip it, and move the images into the public/images folder of the conFusionServer.

#### Updating the CORS Whitelist

* Open cors.js and update the whitelist as shown below, to add the Angular site's URL to the whitelist:

var whitelist = ['http://localhost:3000', 'https://localhost:3443', 'http://<Your Computer's Name>:3001']

#### Updating the Routes

* Open dishRouter.js and update the following:
* . . .
* dishRouter.route('/')
* . . .
* .get(cors.cors, (req,res,next) => {
* Dishes.find(req.query)
* . . .
* })
* . . .
* Furthermore, remove everything related to the comments route from dishRouter.js since we will be introducing a separate Comment Router that serves comments independent of the dishes from the /comments REST API end point.
* Open promoRouter.js and update it as follows:
* . . .
* promoRouter.route('/')
* . . .
* .get(cors.cors, (req,res,next) => {
* Promotions.find(req.query)
* . . .
* })
* . . .
* Open leaderRouter.js and update it as follows:
* . . .
* leaderRouter.route('/')
* . . .
* .get(cors.cors, (req,res,next) => {
* Leaders.find(req.query)
* . . .
* })
* . . .
* Open favoriteRouter.js and update it as follows:
* . . .
* favoriteRouter.route('/')
* . . .
* .post(cors.corsWithOptions, authenticate.verifyUser, (req, res, next) => {
* . . .
* favorite.save()
* .then((favorite) => {
* Favorites.findById(favorite.\_id)
* .populate('user')
* .populate('dishes')
* .then((favorite) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(favorite);
* })
* })
* .catch((err) => {
* return next(err);
* });
* . . .
* else {
* for (i = 0; i < req.body.length; i++ )
* if (favorite.dishes.indexOf(req.body[i].\_id) < 0)
* favorite.dishes.push(req.body[i]);
* favorite.save()
* .then((favorite) => {
* Favorites.findById(favorite.\_id)
* .populate('user')
* .populate('dishes')
* .then((favorite) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(favorite);
* })
* })
* .catch((err) => {
* return next(err);
* });
* . . .
* favoriteRouter.route('/:dishId')
* . . .
* .get(cors.cors, authenticate.verifyUser, (req,res,next) => {
* Favorites.findOne({user: req.user.\_id})
* .then((favorites) => {
* if (!favorites) {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* return res.json({"exists": false, "favorites": favorites});
* }
* else {
* if (favorites.dishes.indexOf(req.params.dishId) < 0) {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* return res.json({"exists": false, "favorites": favorites});
* }
* else {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* return res.json({"exists": true, "favorites": favorites});
* }
* }
* }, (err) => next(err))
* .catch((err) => next(err))
* })
* . . .
* .post(cors.corsWithOptions, authenticate.verifyUser, (req, res, next) => {
* . . .
* favorite.dishes.push({ "\_id": req.params.dishId });
* favorite.save()
* .then((favorite) => {
* Favorites.findById(favorite.\_id)
* .populate('user')
* .populate('dishes')
* .then((favorite) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(favorite);
* })
* })
* .catch((err) => {
* return next(err);
* });
* . . .
* else {
* if (favorite.dishes.indexOf(req.params.dishId) < 0) {
* favorite.dishes.push(req.body);
* favorite.save()
* .then((favorite) => {
* Favorites.findById(favorite.\_id)
* .populate('user')
* .populate('dishes')
* .then((favorite) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(favorite);
* })
* })
* .catch((err) => {
* return next(err);
* })
* }
* . . .
* .delete(cors.corsWithOptions, authenticate.verifyUser, (req, res, next) => {
* . . .
* favorite.save()
* .then((favorite) => {
* Favorites.findById(favorite.\_id)
* .populate('user')
* .populate('dishes')
* .then((favorite) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(favorite);
* })
* })
* . . .
* Update users.js as follows:
* . . .
* favoriteRouter.route('/')
* . . .
* .post(cors.corsWithOptions, authenticate.verifyUser, (req, res, next) => {
* . . .
* favorite.save()
* .then((favorite) => {
* Favorites.findById(favorite.\_id)
* .populate('user')
* .populate('dishes')
* .then((favorite) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(favorite);
* })
* })
* .catch((err) => {
* return next(err);
* });
* . . .
* else {
* for (i = 0; i < req.body.length; i++ )
* if (favorite.dishes.indexOf(req.body[i].\_id) < 0)
* favorite.dishes.push(req.body[i]);
* favorite.save()
* .then((favorite) => {
* Favorites.findById(favorite.\_id)
* .populate('user')
* .populate('dishes')
* .then((favorite) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(favorite);
* })
* })
* .catch((err) => {
* return next(err);
* });
* . . .
* favoriteRouter.route('/:dishId')
* . . .
* .get(cors.cors, authenticate.verifyUser, (req,res,next) => {
* Favorites.findOne({user: req.user.\_id})
* .then((favorites) => {
* if (!favorites) {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* return res.json({"exists": false, "favorites": favorites});
* }
* else {
* if (favorites.dishes.indexOf(req.params.dishId) < 0) {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* return res.json({"exists": false, "favorites": favorites});
* }
* else {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* return res.json({"exists": true, "favorites": favorites});
* }
* }
* }, (err) => next(err))
* .catch((err) => next(err))
* })
* . . .
* .post(cors.corsWithOptions, authenticate.verifyUser, (req, res, next) => {
* . . .
* favorite.dishes.push({ "\_id": req.params.dishId });
* favorite.save()
* .then((favorite) => {
* Favorites.findById(favorite.\_id)
* .populate('user')
* .populate('dishes')
* .then((favorite) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(favorite);
* })
* })
* .catch((err) => {
* return next(err);
* });
* . . .
* else {
* if (favorite.dishes.indexOf(req.params.dishId) < 0) {
* favorite.dishes.push(req.body);
* favorite.save()
* .then((favorite) => {
* Favorites.findById(favorite.\_id)
* .populate('user')
* .populate('dishes')
* .then((favorite) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(favorite);
* })
* })
* .catch((err) => {
* return next(err);
* })
* }
* . . .
* .delete(cors.corsWithOptions, authenticate.verifyUser, (req, res, next) => {
* . . .
* favorite.save()
* .then((favorite) => {
* Favorites.findById(favorite.\_id)
* .populate('user')
* .populate('dishes')
* .then((favorite) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(favorite);
* })
* })
* . . .

#### Adding a /comments REST API end point

* Add a new file named comments.js to the models folder and add the following code to it:
* const mongoose = require('mongoose');
* const Schema = mongoose.Schema;
* 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'
* },
* dish: {
* type: mongoose.Schema.Types.ObjectId,
* ref: 'Dish'
* }
* }, {
* timestamps: true
* });
* var Comments = mongoose.model('Comment', commentSchema);
* module.exports = Comments;
* Remove the Comment schema from dish.js in the models folder, and also remove the comments property from the Dish schema.
* Add a new file named commentRouter.js to the routes folder and add the following code to it:
* const express = require('express');
* const bodyParser = require('body-parser');
* const mongoose = require('mongoose');
* var authenticate = require('../authenticate');
* const cors = require('./cors');
* const Comments = require('../models/comments');
* const commentRouter = express.Router();
* commentRouter.use(bodyParser.json());
* commentRouter.route('/')
* .options(cors.corsWithOptions, (req, res) => { res.sendStatus(200); })
* .get(cors.cors, (req,res,next) => {
* Comments.find(req.query)
* .populate('author')
* .then((comments) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(comments);
* }, (err) => next(err))
* .catch((err) => next(err));
* })
* .post(cors.corsWithOptions, authenticate.verifyUser, (req, res, next) => {
* if (req.body != null) {
* req.body.author = req.user.\_id;
* Comments.create(req.body)
* .then((comment) => {
* Comments.findById(comment.\_id)
* .populate('author')
* .then((comment) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(comment);
* })
* }, (err) => next(err))
* .catch((err) => next(err));
* }
* else {
* err = new Error('Comment not found in request body');
* err.status = 404;
* return next(err);
* }
* })
* .put(cors.corsWithOptions, authenticate.verifyUser, (req, res, next) => {
* res.statusCode = 403;
* res.end('PUT operation not supported on /comments/');
* })
* .delete(cors.corsWithOptions, authenticate.verifyUser, authenticate.verifyAdmin, (req, res, next) => {
* Comments.remove({})
* .then((resp) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(resp);
* }, (err) => next(err))
* .catch((err) => next(err));
* });
* commentRouter.route('/:commentId')
* .options(cors.corsWithOptions, (req, res) => { res.sendStatus(200); })
* .get(cors.cors, (req,res,next) => {
* Comments.findById(req.params.commentId)
* .populate('author')
* .then((comment) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(comment);
* }, (err) => next(err))
* .catch((err) => next(err));
* })
* .post(cors.corsWithOptions, authenticate.verifyUser, (req, res, next) => {
* res.statusCode = 403;
* res.end('POST operation not supported on /comments/'+ req.params.commentId);
* })
* .put(cors.corsWithOptions, authenticate.verifyUser, (req, res, next) => {
* Comments.findById(req.params.commentId)
* .then((comment) => {
* if (comment != null) {
* if (!comment.author.equals(req.user.\_id)) {
* var err = new Error('You are not authorized to update this comment!');
* err.status = 403;
* return next(err);
* }
* req.body.author = req.user.\_id;
* Comments.findByIdAndUpdate(req.params.commentId, {
* $set: req.body
* }, { new: true })
* .then((comment) => {
* Comments.findById(comment.\_id)
* .populate('author')
* .then((comment) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(comment);
* })
* }, (err) => 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) => {
* Comments.findById(req.params.commentId)
* .then((comment) => {
* if (comment != null) {
* if (!comment.author.equals(req.user.\_id)) {
* var err = new Error('You are not authorized to delete this comment!');
* err.status = 403;
* return next(err);
* }
* Comments.findByIdAndRemove(req.params.commentId)
* .then((resp) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json(resp);
* }, (err) => next(err))
* .catch((err) => 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));
* });
* module.exports = commentRouter;
* Finally, open app.js and add the new /comments REST API end point as follows:
* . . .
* var commentRouter = require('./routes/commentRouter');
* . . .
* app.use('/comments',commentRouter);
* . . .
* Save all the changes and do a Git commit with the message "Integrating React Client and Server".
* You can now start your updated server.

#### Conclusions

In this exercise you updated the server in preparation of integration with the client slide.