**Express Session (Part 2)**

#### Objectives and Outcomes

In this exercise you will continue to use Express sessions to track authenticated users so as to enable authenticated access to server resources. At the end of this exercise you will be able to:

* Set up a method for users to register.
* Authenticate registered users to enable them to access secure resources.
* Enable clients to access secure resources on the server after authentication.

#### User Model and User Authentication

* Add a new Mongoose model for users in the file named user.js in the models folder, and add the following to it:
* var mongoose = require('mongoose');
* var Schema = mongoose.Schema;
* var User = new Schema({
* username: {
* type: String,
* required: true,
* unique: true
* },
* password: {
* type: String,
* required: true
* },
* admin: {
* type: Boolean,
* default: false
* }
* });
* module.exports = mongoose.model('User', User);
* Update users.js in the routes folder to support user registration, login and logout:
* . . .
* const bodyParser = require('body-parser');
* var User = require('../models/user');
* router.use(bodyParser.json());
* . . .
* router.post('/signup', (req, res, next) => {
* User.findOne({username: req.body.username})
* .then((user) => {
* if(user != null) {
* var err = new Error('User ' + req.body.username + ' already exists!');
* err.status = 403;
* next(err);
* }
* else {
* return User.create({
* username: req.body.username,
* password: req.body.password});
* }
* })
* .then((user) => {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'application/json');
* res.json({status: 'Registration Successful!', user: user});
* }, (err) => next(err))
* .catch((err) => next(err));
* });
* router.post('/login', (req, res, next) => {
* if(!req.session.user) {
* var authHeader = req.headers.authorization;
* if (!authHeader) {
* var err = new Error('You are not authenticated!');
* res.setHeader('WWW-Authenticate', 'Basic');
* err.status = 401;
* return next(err);
* }
* var auth = new Buffer.from(authHeader.split(' ')[1], 'base64').toString().split(':');
* var username = auth[0];
* var password = auth[1];
* User.findOne({username: username})
* .then((user) => {
* if (user === null) {
* var err = new Error('User ' + username + ' does not exist!');
* err.status = 403;
* return next(err);
* }
* else if (user.password !== password) {
* var err = new Error('Your password is incorrect!');
* err.status = 403;
* return next(err);
* }
* else if (user.username === username && user.password === password) {
* req.session.user = 'authenticated';
* res.statusCode = 200;
* res.setHeader('Content-Type', 'text/plain');
* res.end('You are authenticated!')
* }
* })
* .catch((err) => next(err));
* }
* else {
* res.statusCode = 200;
* res.setHeader('Content-Type', 'text/plain');
* res.end('You are already authenticated!');
* }
* })
* router.get('/logout', (req, res) => {
* if (req.session) {
* req.session.destroy();
* res.clearCookie('session-id');
* res.redirect('/');
* }
* else {
* var err = new Error('You are not logged in!');
* err.status = 403;
* next(err);
* }
* });
* . . .
* Next, update app.js as follows to use the user authentication support:
* . . .
* app.use('/', indexRouter);
* app.use('/users', usersRouter);
* function auth (req, res, next) {
* console.log(req.session);
* if(!req.session.user) {
* var err = new Error('You are not authenticated!');
* err.status = 403;
* return next(err);
* }
* else {
* if (req.session.user === 'authenticated') {
* next();
* }
* else {
* var err = new Error('You are not authenticated!');
* err.status = 403;
* return next(err);
* }
* }
* }
* . . .
* Save the changes and test the server.

#### Conclusions

In this exercise you built the support for user registration and authentication.