Passport

설치

Npm install -s passport

여러가지 전략(페이스북이나 트위터, 구글 등으로 로그인) 중 예제에서는

아이디, 비밀번호로 로그인하기를 사용할것임.

설치

Npm install -s passport-local

Passport는 session을 사용하기 때문에 관련 코드 밑에서 시작해야 한다.

var passport = require('passport') , LocalStrategy = require('passport-local').Strategy;

app.post('/auth/login\_process',passport.authenticate('local',{

  successRedirect: '/',

  failureRedirect:'/auth/login'

}))

app.post('/auth/login\_process', passport.authenticate('local', { failureRedirect: '/auth/login' }),

  (req, res) => {

    req.session.save(() => {

      res.redirect('/')

    })

  })

위 코드는 정상적으로 동작하지 않으므로 이 코드로 대체한다.

/auth/login\_process로 데이터를 받으면

passport에서 제공하는 api(함수)로 처리함

인자는 각각 입력되는 주소, 전략,성공시 리다이렉트, 실패시 리다이렉트 이다.

form에서 id의 name 은 username

비밀번호의 name은 password로 지정되어 있으나 수정하고싶다면

하단의 코드에서 객체 형식으로 인자를 앞에서 넣어주면 된다.

passport.use(new LocalStrategy(

{

usernameField:’email’,

passwordFiled:’pwd’

},

function(username, password, done) {

User.findOne({ username: username }, function(err, user) {

if (err) { return done(err); }

if (!user) {

return done(null, false, { message: 'Incorrect username.' });

}

if (!user.validPassword(password)) {

return done(null, false, { message: 'Incorrect password.' });

}

return done(null, user);

});

}

));

위의 코드는 몽고DB를 사용중인 것으로 보임. 현재 사용중인 코드에 맞게 변형

passport.use(new LocalStrategy(

  {

    usernameField: 'email',

    passwordField: 'pwd'

  },

  function (username, password, done) {

    console.log(username, password);

    if (username == testData.email) {

      if (password == testData.password) {

        return done(null, testData);

      }

      else {

        return done(null, false, { message: 'Incorrect password.' });

      }

    }

    else {

      return done(null, false, { message: 'Incorrect username.' });

    }

  }

));

pm2 start main.js --watch --ignore-watch="data/\* sessions/\*" --no-daemon

session디렉토리에 파일이 추가된면 재시작되서 세션정보가 저장되지 않을 수 있으므로 주의.

//누락코드

App.use(passport.initialize()); - passport 미들웨어를 설치

App.use(passport.session()); - passport는 기본적으로 세션위에서 동작한다. 해당 미들웨어 설치

시리얼라이즈 에러가 발생. 사용자 정보를 세션으로 저장하지 않았다는 에러.

세션을 처리하는 코드(수정필요)

passport.serializeUser(function(user, done) {

  done(null, user.id);

});

passport.deserializeUser(function(id, done) {

  User.findById(id, function(err, user) {

    done(err, user);

  });

});

로그인 성공 시 passport.serializeUser의 인자로 전달된 콜백함수가 수행되도록 되어있음

User는 예제의 authData(testData)가 들어있다

Done의 두번째 인자는 각 사용자의 식별자를 넣는다. 예제의 경우 user.email이다.

deserializeUser는 데이터가 저장되어있는 세션에서 실제 데이터를 조회한다.

serialize에서 user.email만 식별자로 주었기 때문에 세션에는 email만 있다.

해당 식별자는 deserialize에서 id로 입력받는다.

실제에서는 db에서 조회하지만 예제에서는 직접 사용하므로 authdata(testdata)를 직접 사용한다.

passport.serializeUser(function(user, done) {

  done(null, user.email);

});

passport.deserializeUser(function(id, done) {

  done(err,testData);

  /\*User.findById(id, function(err, user) {

    done(err, user);

  });\*/

});

SerializeUser는 로그인에 성공했을 때 그 사실을 session-store에 저장. 로그인 성공시 한번 호출

deserializeUser 는 그 후 매번 호출되면서 저장된 데이터를 기준으로 필요한 정보를 조회할 때 사용한다.

deserializeUser의 done의 testData가 index의 request.user라는 객체로 전달된다.

로그아웃

request.logout();

  request.session.destroy((err) => {

    response.redirect('/');

  });

해당코드는 세션을 완전히 지워버리는 코드이고, 같은기능이지만 지우지 않는 방법으로는

request.logout();

  request.session.save(()=>{

    response.redirect('/');

  })

로그아웃을 한 뒤 그 내용을 저장한뒤 리다이렉션 하는 코드이다.