**Chapter 09. 패스포트로 사용자 인증하기**

* 로컬 인증하기
* [app.js]

passport, flash를 우선 require 불러온다. 이후, passport와 connect-flash 모듈을 app.use를 이용해 초기화 한다.

// ===== Passport 설정 ===== //

app.use(passport.initialize());

app.use(passport.session());

app.use(flash());

* + - * Initialize( )는 초기화, session( )은 세션 사용을 위함이다.
* [user\_schema.js]

아이디를 이메일로 변경하기 위해 스키마 조정이 필요. Id는 email로 변경하고 age를 삭제한다.

path, validate에서 email과 hashed\_password 유효성 검사를 하도록 변경하고 findByEmail로 static 메소드를 수정한다.

* [config.js]

컬렉션 명을 users5로 수정하고, 라우팅 함수들을 등록하기 위해 route\_info 배열 객체는 빈 속성으로 생성한다. 이후 routes 폴더의 users.js를 삭제한다.

* [app.js]

Local strategy를 설정하는 코드 입력.

passport.use( ) 를 사용해 ‘local-login’이라는 방식을 설정하고 new Local Strategy인스턴스를 통해 설정 방식을 정의한다. usernameField와 passwordField를 설정하고 passReqToCallback도 true로 설정한다.

이후 데이터베이스의 유저모델에서 email로 검색 후 검증 결과를 done(null, user)로 리턴한다.

* + - * 사용자가 없는 경우에는 역시 스트래티지를 통해 회원가입도 구현할 수 있다.

역시 passport.use를 사용해 local-signup을 만들어 검증 후 save( ) 를 사용해 사용자를 추가로 생성한다.

로그인 이후에는 인증 정보를 직접 요청하거나 전달하지 않고, 이후 요청 정보는 세션으로 확인을 해야 한다. 인증 성공 시 세션이 만들어지고, 유저 브라우저에 쿠키로 유지가 된다. 패스포트에서는 인증 성공 시 사용자 정보를 세션에 Serialize(저장)하거나 세션으로부터 Deserialize(복원)할 수 있다.

passport.serializeUser( )는 로그인 성공 시 호출, passport.deserializeUser( )는 로그인 이후 사용자 요청이 있을 때 마다 호출하게 된다.

* [app.js] 에 상기 두 개의 함수 추가
* 라우팅 함수 등록

1. get 방식 : 단순 웹 문서 조회
   * + - 홈 화면( / ), 로그인 화면( /login ), 회원가입 회면( / signup ), 프로필 조회( /profile)
2. post 방식 : 인증, 가입 진행 요청
   * + - 로그인, 회원가입 패스

// 라우팅 정보를 읽어들여 라우팅 설정

var router = express.Router();

route\_loader.init(app, router);

* 홈 화면

// 홈 화면 - index.ejs 템플릿으로 홈 화면 출력. [views] 폴더의 index.ejs를 단순 로딩

router.route('/').get(function(req, res){

console.log(' / 패스 요청됨');

res.render('index.ejs');

});

* 로그인

1. 단순 화면 출력

// 로그인 화면 - login.ejs 템플릿 출력

router.route('/login').get(function(req, res){

console.log('/login 패스 요청됨');

res.render('login.ejs', {message: req.flash('loginMessage')});

});

1. POST 처리 : passport 객체의 authenticate( ) 메소드를 사용해 패스포트 모듈에서 사용자 인증을 처리한다. local-login이라는 이름을 선언해 놓았으니 이 이름을 첫 번째 파라미터로 전달하고, 두 번째 파라미터는 사용자 인증 성공 / 실패 시 이동 패스 선언

// 사용자 인증(POST) - 성공 시 /profile, 실패 시 /login 리다이렉트

router.route('/login').post(passport.authenticate('local-login', {

successRedirect: '/profile',

failureRedirect: '/login',

failureFlash: true

}));

* 회원가입

// 회원가입 화면 링크 - signup.ejs 템플릿

router.route('/signup').get(function(req, res){

console.log('/signup 패스 요청');

res.render('signup.ejs', {message: req.flash('signupMessage')});

});

router.route('/signup').post(passport.authenticate('local-signup', {

successRedirect: '/profile',

failureRedirect: '/signup',

failureFlash: true

}));

* 프로필 화면

router.route('/profile').get(function(req, res){

console.log('/profile 패스 요청됨');

// 인증된 경우, req.user 객체에 사용자 정보가 있으며 인증이 안 된 경우req.user는false

console.log('req.user 객체 값');

console.dir(req.user);

// 인증 실패

if(!req.user){

console.log('사용자 인증 실패 상태');

res.redirect('/');

return;

}

// 인증 성공

console.log('사용자 인증 성공 상태');

if(Array.isArray(req.user)){

res.render('profile.ejs', {user: req.user[0].\_doc});

} else {

res.render('profile.ejs', {user: req.user});

}

});

* 로그아웃

router.route('/logout').get(function(req, res){

console.log('/logout 패스 요청됨');

req.logout();

res.redirect('/');

});

* 뷰 템플릿 작성