**Chapter 09. 패스포트로 사용자 인증하기**

* 패스포트로 로그인 하기
* Passport는 익스프레스에서 클라이언트의 요청 인증정보를 사용해 사용자 인증만을 담당하는 인증 모듈.
* 패스포트가 제공하는 인증 방식을 결정하는 것이 Strategy(스트래티지).
  + - * 어떤 인증방식을 사용하느냐에 따라 인증방식이 변경된다.  
        Local Strategy(DB 비교)  
        OAuth 인증(페이스북, SNS 계정)
      * 클라이언트에서 인증 요청을 하면 서버는 Passport 모듈을 통해 알맞은 인증 Strategy를 이용해 인증을 진행하고 세션에 저장해 리턴한다(로그인 이후의 처리는 세션 유무를 확인해서 진행)
* 인증을 위한 패스포트 코드 패턴

router.route('/login').post(passport.authenticat('local',

{

successRedirect: '/',

failureRedirect: '/login'

}

));

* + - * 라우터 객체에 등록하는 라우팅 함수. 클라이언트에서 POST 방식으로 요청하는 요청 패스가 /login일 때 호출되는 함수.
* 인증 함수 예시

router.route('/login').post(passport.authenticat('local'),

function(req, res){

// 인증 성공 시 호출

// 'req.user'는 인증된 사용자의 정보

res.redirect('/users/' + req.user.username);

}

);

* + - * ‘local’은 인증 strategy이고, 인증 성공 시에는 redirect로 추가 라우팅이나 함수가 실행되고, 이 경우 실패하면 default로 401 Unauthorized 상태 응답.
      * 또는 맨 위의 success/failureRedirect를 사용할 수 있다.
* 플래시 메시지와 커스텀 콜백

Redircet를 사용해 응답을 보낼 때 보통 사용하는 메시지로, 상태 메시지를 응답 웹 문서쪽으로 전달할 때 사용한다.(connect-flash 설치)

* + - * req.flash( ) 함수에서 파라미터가 2개면 set, 1개면 get이다.

Passport모듈 🡪 req.flash(‘loginMessage’, ‘등록된 계정이 없습니다.’);  
 🡪 req.flash(‘loginMessage’)

또는 passport.authenticate( ) 호출 시 failureFlash 옵션 설정도 가능하다.  
(req.flash 함수를 사용하지 않고, failureRedirect처럼 옵션 지정)

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

{

successRedirect: '/',

failureRedirect: '/login'

failureFlash: true

}

));

* + - * 이 메시지는 Verify Callback(검증 콜백)이 설정되어 있다면 자동 설정.  
        Verify Callback은 인증을 처리하는 함수
      * True 대신 string 설정도 가능함!
* 커스텀 콜백

Custom Callback으로 인증 성공/실패 시 동작을 직접 함수로 지정할 수 있다.

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

passport.authenticate('local', function(err, user, info){

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

if(!user) { return res.redirect('/login'); }

// 패스포트 인증 결과에 따른 로그인 진행

req.login(user, function(err){

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

return res.redirect('/users/' + user.username);

});

})(req, res, next); // 클로저를 사용해 req, res 객체 접근 가능

});

* + - * 이 경우 authenticate( ) 메소드가 라우팅 함수가 아닌, 다른 라우팅 함수 안에서 호출되도록 선언했으며 이 경우 Closure를 통해 req, res에 접근한다.
      * 여기서 info 파라미터는 스트래티지 설정 시 지정한 검증 콜백에 의해 제공되는 부가정보를 담고 있다. 그리고 req.login( ) 은 임의의 함수이다.
* 스트레티지 설정과 검증 콜백
  + - * 스트레티지 설정은 passport.use( ) 함수로 가능

var passport = require('passport');

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

passport.use(new LocalStrategy(

function(username, password, done){

UserModel.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);

});

}

));

* + - * Local Strategy를 사용하도록 use( ) 메소드를 사용해 설정한다.
      * 그리고 스트레티지 설정 시 Verify Callback(검증 콜백)에서 인증 결과를 처리하도록 만든다. 요청받은 인증 정보(Credential)들을 가지고 있는 사용자를 찾고, 성공/실패한 경우를 done 메소드로 알려준다.   
        > 이 과정이 끝나야 authenticate( ) 메소드를 호출하는 쪽에서 성공/실패 결과를 받아볼 수 있다.
      * done메소드 사용은 return done(null, user) 또는 null, false 형태로 사용 가능하다.