KWEB Study
Week8:
Login with Node.js

KWEB 2학기 준회원 스터디

Today's Contents

1. Before Study
2. Board Review
3. Cookie & Session
4. Advanced Topic
5. 실습 & QnA
6. 과제

M.G. BAE

J.H. BAEK

2017 Fall Semester

Before Study

- 지난 주차에는 MySQL (MariaDB)를 이용해 local에서 기본적인 SQL을 다루어 보고 게시판을 제작해보았습니다. (숙제 다 해오셨죠? ㅎ)
- 8주차의 목표는 Cookie와 Session에 대해 이해해보고 로그인 시스템을 구현하여 우리의 게시판 프로젝트에 적용시키는 것입니다.
- 또한, 약간은 Advanced Topic을 조금씩만 살펴보도록 하겠습니다.
- 드디어 2학기 마지막 주차까지 왔네요! 오늘도 지난 스터디 한 번 되짚어보고 시작하겠습니다~

Previous Study - Basic SQL

- SQL은 DB 사용 시 DB에 접근할 수 있는 데이터베이스 하부 언어를 의미하며 대부분의 RDBMS에서 사용됩니다.

- MySQL (MariaDB)에서 사용하는 SQL의 CRUD 기능 쿼리는 아래와 같습니다.
  - Create → **INSERT INTO** ex) INSERT INTO Board (title, content) VALUES (‘title’, ‘content’);
  - Read → **SELECT** ex) SELECT * FROM Board WHERE id=12;
  - Update → **UPDATE** ex) UPDATE Board SET title=‘[noti] ’+title WHERE id<5;
  - Delete → **DELETE** ex) DELETE FROM Board WHERE id>12;

- Node.js에서 사용하는 mysql 모듈에서도 기본적으로 위와 같은 쿼리를 사용합니다.

Previous Study - SQL with Node.js

- Node.js + Express.js에 MySQL (MariaDB)을 사용하는 방법은 npm을 이용해 mysql 모듈을 올려 사용하는 것입니다. 모듈을 통해 DB와 연결하여 사용할 수 있습니다.

- 아래는 mysql 모듈의 메소드들입니다.
  - mysql.createConnection → 새로운 연결 설정을 생성합니다
  - mysql.createPool → 새로운 연결 풀을 생성 합니다.
  - pool.getConnection → 풀에서 새로운 연결을 가져오거나 이미 존재하는 연결을 가져옵니다.
  - conn.connect → 연결 설정에서 실제로 연결을 구성합니다.
  - conn.query → 활성화된 연결을 사용해서 쿼리를 수행합니다.

- 활성화시킨 connectio은 다 쓰고! conn.end 혹은 conn.release 로 꼭 연결을 끊거나 connection Pool에 돌려주어야 합니다.

Previous Study - Board

- 지난 주차에 게시판 리스트 표시하는 것과 게시물 열람, 작성까지 스터디 시간에 구현하였습니다.
- 추가적으로, 과제로 게시판 수정, 삭제 기능을 구현해보았습니다.
- 우리가 구현한 것은 MySQL (MariaDB)와 연결하여 CRUD 기능을 구현해본 것입니다.
- But, 실제 게시판은 위의 기능 외에도 다양한 많은 기능들이 존재합니다. 이미 많은 웹 사이트를 사용하면서 경험해보셨을거라 생각합니다!

Board Review

- 이번주에는 우리의 게시판에 사용할 Login 기능을 구현해볼 것입니다.
- 그러니 먼저 우리가 만든 게시판부터 점검해봅시다.
- CRUD 기능만 똑바로 구현했으면 됩니다.
- 지금부터 보여드리는 것은 예제이므로 본인이 구현한 게시판에 맞게 이해하시면 됩니다.
- (But, 아직 과제 기간이라 비워두었습니다. 추가해서 재배포할게요!)

Board - 게시물 리스트 표시

추가해서 재배포해드리겠습니다.

Board - 게시물 열람

추가해서 재배포해드리겠습니다.

Board - 게시물 작성

추가해서 재배포해드리겠습니다.

Board - 게시물 수정

추가해서 재배포해드리겠습니다.

Board - 게시물 삭제

추가해서 재배포해드리겠습니다.

Login System

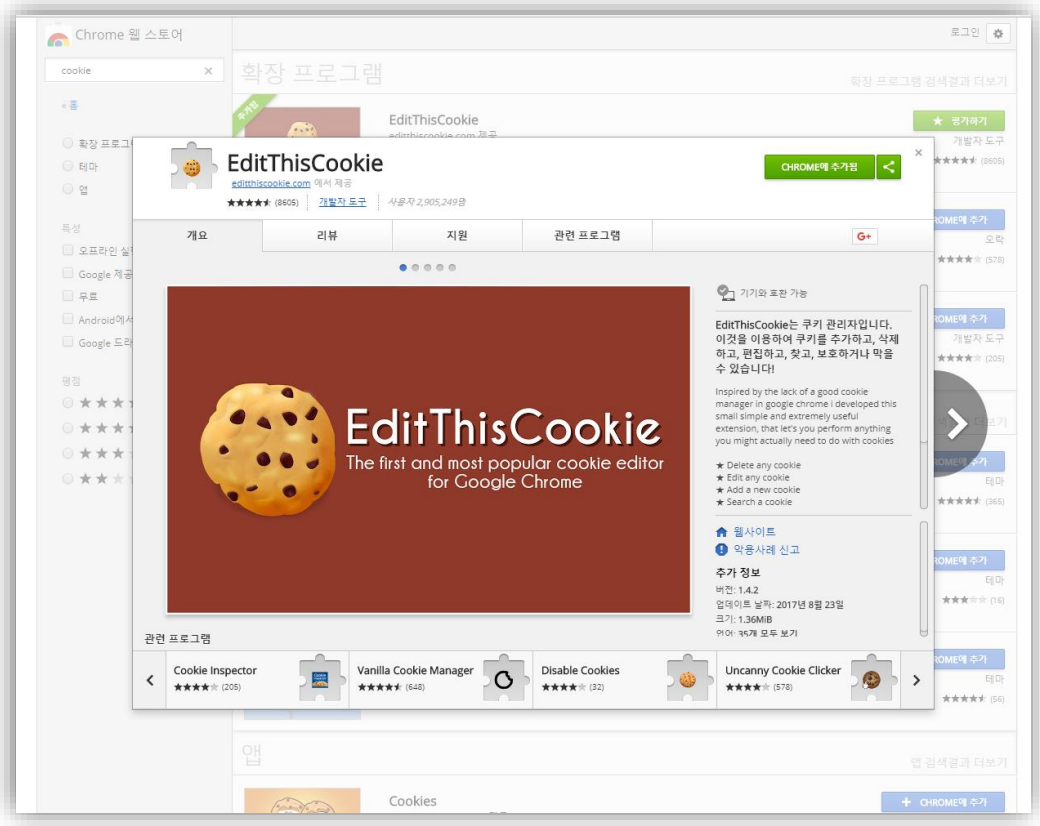
- Login: 사용자가 호스트 컴퓨터나 네트워크에 자신의 아이디(ID)와 암호를 입력해서 자신을 알리고 등록하여 호스트 컴퓨터의 사용 권한을 받아 접속하는 작업
- 사용자가 로그인한 상태인지 아닌지 확인하고 싶을 때에는 쿠키나 세션을 사용합니다.
- Cookie: 클라이언트 브라우저에 저장되는 정보
- Session: 웹 서버에 저장되는 정보

Cookie

- 웹사이트에 접속할 때 자동적으로 만들어지는 임시 파일로 Local 데이터입니다.
- 웹 서버는 쿠키를 읽어 사용자가 설정한 사항을 확인할 수 있습니다.
  - 예를 들어, '팝업창을 다시는 안 본다'라는 체크를 했는지도 확인할 수 있습니다.
- 저장된 쿠키는 요청시에 Cookie http 헤더를 통해서 서버로 전송됩니다. 사용자 기억, 인증 정보 등을 기억할 수 있으나 개인정보 침해 문제가 있으니 적절히 사용해야합니다.
- 쿠키는 사용자측(수동 or Javascript로), 서버측(Set-cookie http 헤더)에서 설정할 수 있습니다.

Cookie with Express

- 우리가 배운 Express.js에서는 cookie-parser 미들웨어를 통해 쿠키를 설정하거나 확인할 수 있습니다.
- 일단 Chrome에서 Cookie를 읽고 수정할 수 있는 Tool인 EditThisCookie를 설치해 봅시다.
- 한번 위의 Tool로 여러 사이트들이 가지고 있는 쿠키 한 번 확인해보세요!
- Chrome 개발자 도구로도 확인해보세요!



실습 - Cookie with Express

- Router에 cookies.js에 오른쪽 코드를 이용하여 실습해봅시다.
- 오른쪽 코드는 클라이언트에 Cookie를 설정하고 볼 수 있는 사이트입니다.
  - /cookies/get: 쿠키를 확인
  - /cookies/set: 쿠키를 설정
- EditThisCookie로도 확인해봅시다!

```
var express = require('express');
var cookieParser = require('cookie-parser');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res, next) {
    res.render('index', { title: 'Express' });
});

router.get('/get', function(req, res, next) {
    res.send(req.cookies);
});

router.get('/set', function(req, res, next) {
    res.cookie('kweb',{
        id: 'study',
        name: '마지막',
        authorized: true
    });
    res.redirect('/cookies/get');
});

module.exports = router;
```

Session

- 웹에선 클라이언트와 웹 서버 간의 활성화된 접속을 의미하며, 서버 측에 저장 되는 정보를 세션 데이터라고 합니다.
  - OSI 계층 모델에선 사용자와 데이터는 묶어 주는 계층을 의미합니다. (전공에서 배울거에요!)

- 앞에서 본 쿠키에 중요한 정보를 쿠키에 저장해 놓으면 조작의 위험이 있습니다. (서비스 포인트 정보, 특정 권한 허용 정보)

- 또한 매 요청마다 http에 포함되어 전송 되기때문에 데이터 유출의 가능성도 높습니다.

- 따라서 서버만이 관리하는 데이터인 세션이 필요하며, 대부분의 경우에 세션은 쿠키에 세션 ID를 설정함으로써 작동합니다.

Session with Express

- Express에서 세션을 지원하기 위해서는 express-session 모듈을 사용해야합니다.

```
$ npm install express-session --save
```

- app.js에 express-session을 미들웨어로 사용해야합니다.
  - 상단에 var expressSession = require('express-session'); 을 추가하고 app.use들이 많이 보이는 적당한 곳에 아래의 예시코드와 같은 코드를 추가해야합니다.

```
//session
app.use(expressSession({
    secret: '@#@$KWEB#@$#$',
    resave: false,
    saveUninitialized: true
}));
```

실습 - Session with Express

- Express-Generator로 세션을 이용한 로그인 시스템을 구현해봅시다.

- 로그인을 하지 않으면 Login 페이지(/login)만 계속 표시되고, Login을 하면 상태 표시 페이지(/login/view)만 표시됩니다. (+ 로그아웃도 구현!)

- Router 설정에 login.js를 추가합시다. 아래는 로그인 페이지의 GET 방식 코드입니다.

```
router.get('/', function(req, res, next) {
    if(req.session.user) {
        res.redirect('/login/view');
    } else {
        res.render('login');
    }
});
```

실습 - Session with Express

- /login/view 페이지는 각자 원하는 대로 만드시고 로그인, 로그아웃 함수를 살펴봅시다. 로그인은 POST 방식, 로그아웃은 GET 방식(POST도 상관X)으로 구현했습니다.

```
router.post('/', function(req, res, next) {
    if(req.session.user) {
        res.redirect('/login/view');
    } else {
        var id = req.body.id;
        var pwd = req.body.pwd;
        req.session.user = {
            id: id,
            pwd: pwd,
            authorized: true
        }
        res.redirect('/login/view');
    }
});
```

Login Code

```
router.get('/logout', function(req, res,
next) {
    if(req.session.user) {
        req.session.destroy(function (err)
{
            if(err) throw err;
            res.redirect('/login');
        });
    } else {
        res.redirect('/login');
    }
});
```

Logout Code

Session with DB

- 이번엔 DB에 있는 정보를 가져와 인증 시 사용해봅시다. 사실 이게 정석적인 방식입니다. 홈페이지에 가입하면 user 정보는 보통 DB에 저장되기 때문입니다.

- Logout 코드는 내버려 두고 Login 코드만 수정하면 됩니다. 아래는 mysql을 로그인을 위한 form태그가 있는 페이지의 Router 설정 예시입니다.

```
router.get('/mysql', function(req, res, next) {
    if(req.session.user) {
        res.redirect('/login/view');
    } else {
        res.render('login', { me: "mysql" });
    }
});
```

Session with DB

- 오른쪽은 MySQL (MariaDB)를 이용해 로그인 함수를 구현한 것입니다.
- 입력된 id와 pwd를 SELECT문을 이용해 DB에서 결과를 가져옵니다.
- 결과가 있으면 로그인! 비어있으면 로그인 실패! 입니다.

```
router.post('/mysql', function(req, res, next) {
    if(req.session.user) {
        res.redirect('/login/view');
    } else {
        pool.getConnection(function (err,conn) {
            if(err) {
                if(conn) {
                    conn.release();
                }
                callback(err,null);
                return;
            }
            var id = req.body.id;
            var pwd = req.body.pwd;
            var sql = "SELECT * FROM board_user WHERE id = ? AND pwd = ?";
            var exec = conn.query(sql,[id,pwd] ,function(err, rows) {
                conn.release();

                if (err) throw err;
                if(rows.length > 0) {
                    req.session.user = {
                        id: id,
                        pwd: pwd,
                        authorized: true
                    }
                    res.redirect('/login/view');
                } else {
                    res.send('<script>alert("해당 유저는 존재하지 않습니다.");history.back();</script>');
                }
            });
        });
    }
});
```

Advanced Topic

- 물론 여러 가지 Advanced Topic이 존재하나 이번주 스터디와 관련 있는 3가지 주제 정도만 짧게 소개해보려 합니다.

- Cryptography

- SQL Advanced

- Passport.js

Cryptography

- 사실 유저 정보 중 Password 같은 경우는 DB에 원문 그대로 저장될 경우 심각한 보안 위협에 놓일 수도 있습니다.
- 따라서, 보통 Password와 같은 주요한 정보일 경우는 암호화를 하여 저장합니다.
- 암호화 알고리즘은 MD5, BASE64, SHA512, DES, AES, RSA 등 상당히 많은 암호화 알고리즘들이 존재합니다. (암호화 알고리즘은 정보보호 등 전공에서 배웁시다!)
- Node.js에서는 crypto 모듈과 같은 것으로 위의 암호화가 가능합니다.

SQL Advanced

- 우리는 지난주에 DB Table에 Author이라는 항목을 따로 빼내서 저장하였습니다.
- 하지만 보통 웹 서비스에서는 게시물과 유저는 다른 Table에 저장되어 있습니다.
- 이 경우 JOIN 이라는 SQL을 사용하여 DB에서 정보를 가져옵니다.
- Ex) SELECT * FROM board INNER JOIN users ON board.uid_fk = users.uid;
  - 위의 쿼리 같은 경우는 기본적으로 board에서 row을 가져오지만 board 테이블의 uid_fk 컬럼값을 users 테이블의 uid 컬럼에서 찾아 같은 값을 가지는 row를 합쳐 row를 반환합니다.

Passport.js

- 로그인 시스템에서 사용할 수 있는 Node.js에서 사용하는 사용자 인증 모듈입니다.
- 미들웨어로 끼워 넣을 수 있어 몇 가지 간단한 설정만으로도! 로그인 기능을 만들 수 있습니다. But, 단순히 인증 기능만을 담당합니다.
- Strategy로 여러 가지 인증 방식을 집어넣을 수 있습니다. Local을 제외하고도 Facebook, KakaoTalk, Naver 등 SNS 로그인도 가능합니다.
- (우리가 구현한 것은 Local 로그인입니다.)

겨울방학 스터디

- 겨울방학 스터디 계획은 아래와 같습니다.

일시		내용
보충 1주차	12.31. - 01.04.	Front-end: 게시판 HTML, CSS 작성
보충 2주차	01.05. - 01.09.	Back-end: Node.js + Express.js로 웹 서버 작성
보충 3주차	01.10. - 01.14.	Local Server: local에서 Front-end, Back-end 통합 작성
1주차	01.15. - 01.20.	Passport.js (Local 로그인) + crypto 모듈 사용
2주차	01.21. - 01.25.	Passport.js (OAuth 로그인) + 게시판 확장
3주차	01.26. - 01.30.	Complete Board (로그인 기능 + 회원가입 + 게시판 기능)
4주차	01.31. - 02.05.	AWS 서버에 올려 운용하기

실습 - Authority

- Express (Node.js)를 이용해 Page 1, Page 2, Page 3을 만들어봅시다.

- 위의 과정을 완료하면 로그인 페이지를 만들어 DB에서 유저 정보를 확인해 인증하는 로그인 기능을 구현해봅시다.

- 아래와 같이 구현해볼 것입니다.
  - Page 1: 로그인 여부와 상관없이 접근가능
  - Page 2: 로그인을 하였을 때만(세션 정보가 있는 경우에만) 접근가능
  - Page 3: 로그인한 유저들 중 특정 권한이 부여된 경우에만 접근가능

실습 - Authority

- test.js를 만들어 Route 설정을 하고 아래와 같이 페이지 4개를 GET 방식으로 만들어 봅시다.
  - /test/page1 → 로그인 여부와 상관없이 볼 수 있습니다.
  - /test/page2 → 로그인 시 볼 수 있습니다.
  - /test/page3 → req.session.users.auth의 값이 10 이상이면 볼 수 있습니다.
  - /test/login → 로그인 페이지입니다. 앞의 로그인 기능을 구현하시면 됩니다. (DB 사용은 자유!)

- 코드는 주어지지 않을거구요! Hint만 몇 개 드리겠습니다. 구현해보세요~
  - Hint 1. Page 1은 그냥 지난주에 배우신대로 rendering 하시면 됩니다.
  - Hint 2. Page 2는 오늘 배운대로 req.session.users가 존재하면 보이게 하면 됩니다.
  - Hint 3. Page 3은 req.session.users를 확인하고 안의 auth 값이 10 이상!일 때만 볼 수 있음!

실습 & QnA

- 앞의 예제들에서 실습 잘해보셨죠? 이번주에도 실습을 따로..
- 할까 하다가 오늘 실습은 이번주까지 냈던 게시판을 완성하는 것입니다. 올바르게 DB와 연결되어 게시판이 CRUD 기능이 동작하는지 시험해봅시다!
- 다시 말해서, 지금까지 진행한 과제를 점검하고 미제출된 과제를 완성하는 시간을 가질 것입니다~
- 스터디장 괴롭히면서 물어보시고! 다 하신 분은 잠깐 뒤의 과제하시면 될 듯합니다.

과제 (~ 12/22 13:00)

- 8주차는 로그인 시스템에 관해 살펴보며, Cookie와 Session에 관해 공부해보았습니다.
- 이번주 과제는 다음 슬라이드대로 하시면 됩니다. 2학기 스터디 듣느라 수고하셨어요!
- 과제는 지난 주와 같이 Github에 올려 링크로 카톡으로 제출해주세요~
- 제출기한: 12월 22일 오후 1시까지
- 과제 제출 E-mail
  - 월 7시: 15 배민근 (baemingun@naver.com)
  - 화 7시: 16 백지훈 (bjh970913@gmail.com)

과제 (~ 12/22 13:00)

- 우리가 완성한 게시판 예제를 로그인 기능과 연동시키는 것이 이번주 과제입니다.

- 아래의 명세사항을 지켜주시면 됩니다. (DB 수정 필요하시면 하시면 됩니다.)
  - 완성된 게시판 기능 (CRUD) + 로그인 페이지 → 로그인 기능을 구현해야합니다!
  - 로그인이 되어 있지 않다면(세션 정보가 없다면) 게시판 리스트는 볼 수 있지만 게시물 내용 열람, 게시물 작성, 수정, 삭제 불가능 → 해당 페이지에 접근이 불가능해야합니다!
  - 로그인이 되어 있다면(세선 정보가 있다면) 게시판 리스트 열람, 게시물 열람과 작성이 가능하나 수정과 삭제는 게시물의 작성자일 때만 가능합니다. → 해당 기능에 권한 설정이 필요합니다.
  - 된다면 회원 가입 기능도! 선택사항입니다.

- **Optional!** 위 과제를 완료하고 댓글 기능 구현해오시면 면제카운트 +2 드릴 것이니 아웃 카운트 많으신 분들은 노려보세요~

It’s all today!