



# 로그인 / 로그아웃 페이지 구성

Web 개발 기초





1. 로그인 / 로그아웃 페이지 구성

2. 로그인 / 로그아웃 기능 구현





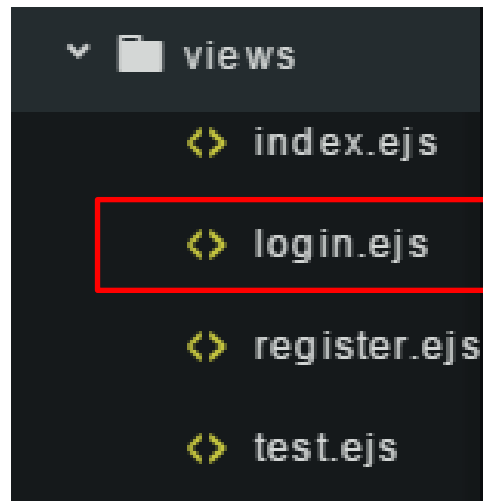
# 1. 로그인 / 로그아웃 페이지 구성

## 로그인 페이지 구성

회원가입을 했으니 이제 로그인을 해야 한다.

로그인 페이지는 회원가입 페이지와 비슷한 형태로 구성한다.  
(로그아웃은 따로 페이지 없이 메뉴만 만든다)

먼저 views 에 login.ejs 파일을 만든다.





# 1. 로그인 / 로그아웃 페이지 구성

## 로그인 페이지 구성

Login.ejs 파일을 열고 아래와 같이 작성한다.

```
<html>
  <head>
    <title>
      로그인
    </title>
    <link rel="stylesheet" href="css/style.css" />
    <script src="js/jquery-3.2.0.min.js"></script>
  </head>
  <body>
    <div class="main-container">
      <div class="main-body">
        <div class="container">
          <form class="login-form" action="/login" method="post">
            
            <div class="input-group">
              <div class="input-label">아이디 :</div>
              <input class="login_id" type="text" name="username" />
            </div>
            <div class="input-group">
              <div class="input-label">비밀번호 :</div>
              <input class="login_pw" type="password" name="password" />
            </div>
            <input class="submit-btn" type="button" onclick="check();" value="로그인"/>
            <input class="register-btn" type="button" onclick="location.href='/register'" value="회원가입"/>
          </form>
        </div>
      </div>
    </div>
  </body>
</html>
```



# 1. 로그인 / 로그아웃 페이지 구성

## 로그인 페이지 구성

Route.js 에 아래와 같이 추가해준다.

```
app.get('/login',function(req,res){  
  res.render('login',{  
  
  });  
});
```

그리고 `http://본인ip:3000/login` 으로 브라우저에서 접속해보면 잘 나오는 것을 확인 할 수 있다.

**KUstagram**

아이디 :

비밀번호 :



# 1. 로그인 / 로그아웃 페이지 구성

## 로그아웃 버튼 구성

로그아웃은 페이지가 따로 필요 없으므로 버튼을 만들어준다.

Index.ejs 에서 header-setting의 내용을 아래와 같이 바꿔준다

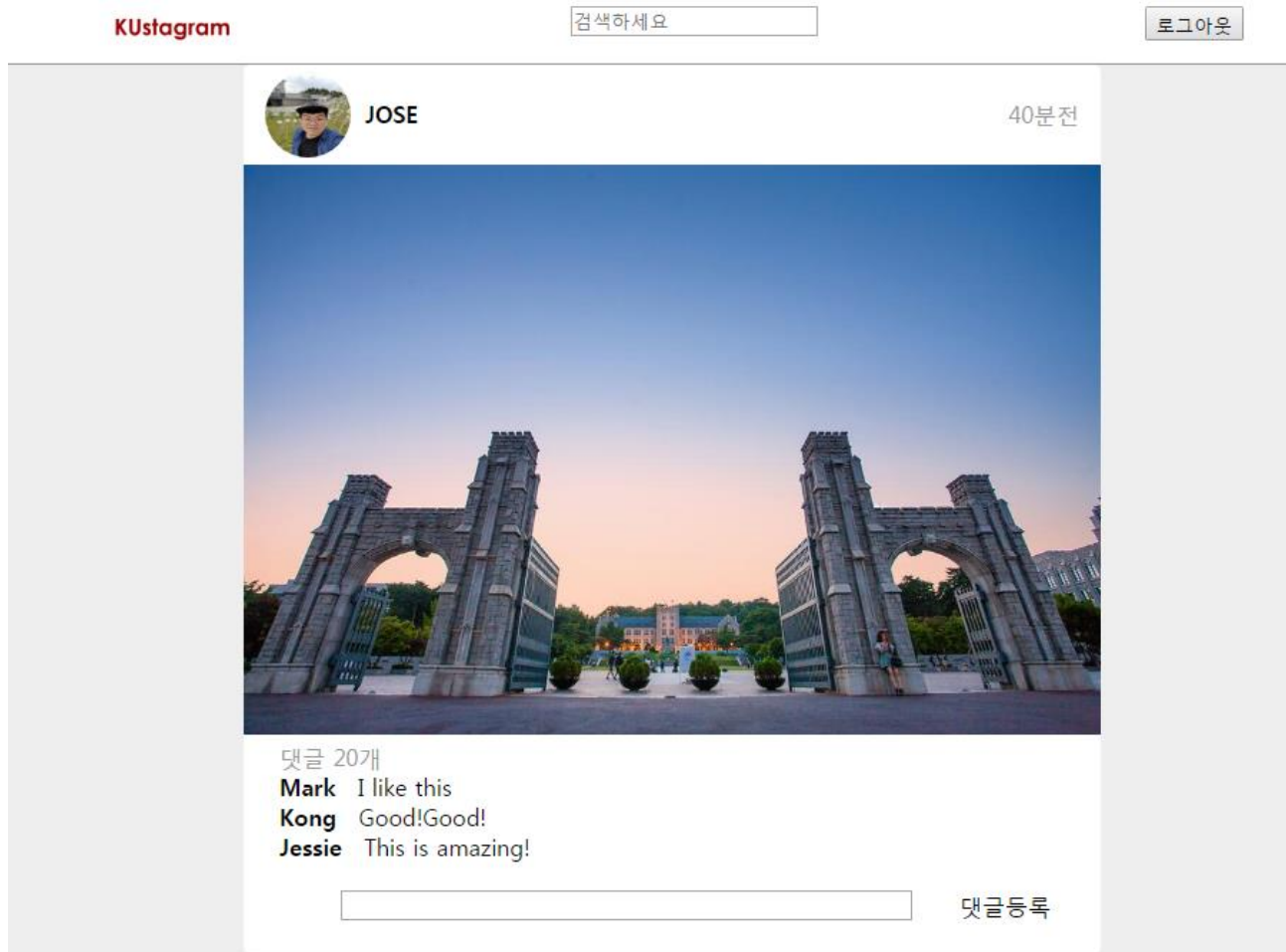
```
<div class="header-search">
  <input type="text" placeholder="검색하세요" />
</div>
<div class="header-setting">
  <input type="button" value="로그아웃" />
</div>
</div>
```



# 1. 로그인 / 로그아웃 페이지 구성

## 로그아웃 버튼 구성

브라우저에서 확인해보면





## 2. 로그인 / 로그아웃 기능 구현

### 로그인 기능 구현

로그인 기능은 passport 라는 미들웨어를 사용한다.

미들웨어는 다양한 기능을 사용하기 쉽게 해주는 모듈이라고 생각하면 된다.

따로 설명을 안했을 뿐, 이미 미들웨어를 사용하고 있다.

(express 또한 미들웨어)

```
var express = require('express');
var path = require('path');
var app = express();
var bodyParser = require('body-parser');

app.use(express.static(path.join(__dirname, 'static')));
app.set('views', __dirname + '/views');
app.set('view engine', 'ejs');
app.engine('html', require('ejs').renderFile);

var server = app.listen(3000, function(){
  console.log("Express server has started on port 3000");
});

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
  extended: true
}));

var router = require('./router/route')(app);
```





## 2. 로그인 / 로그아웃 기능 구현

### 로그인 기능 구현

Passport는 가장 처음 node.js 설정 시 package.json 을 통해 설치가 되어 있으므로 따로 설치하는 법은 다루지 않는다.

먼저 server.js 상단에 미들웨어를 추가해준다.

```
var express = require('express');
var path = require('path');
var app = express();
var bodyParser = require('body-parser');
var passport = require('passport'), LocalStrategy = require('passport-local').Strategy;
var mysql = require('mysql');
var db_config = require('./db');
var session = require('express-session');
var MySQLStore = require('express-mysql-session')(session);
```



## 2. 로그인 / 로그아웃 기능 구현

### 로그인 기능 구현

그리고 미들웨어를 적용한다.

여기서 session은 로그인 정보를 저장하는 임시 파일로, 보안을 위해 DB에 저장하고자 한다.

Secret은 임의의 값을 마음대로 입력한다.

Store항목의 내부 값은 DB.js 파일의 내용과 같이 작성한다.

router 변수에서 받는 input 값에도 passport를 추가해줘야 한다.

```
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({
  extended: true
}));
app.use(session({
  secret: '1234!@#$1234!@#$',
  resave: false,
  saveUninitialized: true,
  store: new MySQLStore({
    host: 'localhost',
    port: 3306,
    user: 'root',
    password: 'webprac',
    database: 'instagram'
  })
}));

app.use(passport.initialize());
app.use(passport.session());

var router = require('./router/route')(app, passport);
```



## 2. 로그인 / 로그아웃 기능 구현

### 로그인 기능 구현

이제 passport를 이용해 로그인 기능을 구현한다.

먼저 route.js 의 input값으로 passport를 추가해준다.

```
module.exports = function(app, passport)
```

그리고 아래 내용을 추가해준다.

(내용에 대한 설명은 구두 전달 예정)

```
app.post('/login',
  passport.authenticate('local', {
    successRedirect: '/',
    failureRedirect: '/logIn_fail',
    failureFlash: false
  })
);

app.get('/logIn_fail', function(req, res){
  res.send('<script type="text/javascript">alert("로그인 실패");location.href="/login"</script>');
});
```



## 2. 로그인 / 로그아웃 기능 구현

### 로그인 기능 구현

Server.js 에 추가

```
app.use(passport.initialize());
app.use(passport.session());

passport.serializeUser(function(user, done) {
  console.log('serializeUser');
  done(null, user.user_id);
});

passport.deserializeUser(function(name, done) {
  console.log('deserializeUser');
  var db = mysql.createConnection(db_config);
  var sql = 'select * from `user` where `user_id`=?';
  db.query(sql, [name], function(err, results){
    if(err){
      console.log(err);
      done('There is no user.');
```

```
    } else {
```

```
      done(null, results[0]);
```

```
    }
```

```
    db.end();
```

```
  });
```

```
});
```

```
passport.use(new LocalStrategy(
  function(username, password, done) {
    console.log('LocalStrategy');
    var db = mysql.createConnection(db_config);
    var uname = username;
    pwd = password;
    var sql = 'select * from `user` where `user_id`=?';
    db.query(sql, [uname], function(err, results){
      if(err){
        return done('There is no user.');
```

```
      }
```

```
      var user = results[0];
```

```
      if(user){
```

```
        if(pwd === user.user_pw){
```

```
          done(null, user);
```

```
        } else {
```

```
          done(null, false);
```

```
        }
```

```
      } else {
```

```
        done(null, false);
```

```
      }
```

```
      db.end();
```

```
    });
```

```
  }
```

```
));
```



## 2. 로그인 / 로그아웃 기능 구현

### 로그인 기능 구현

http://본인ip:3000/login 에서 로그인을 해보면

1. 비밀번호 잘못 입력 또는 없는 아이디 입력시

The screenshot shows a web browser window with the URL "http://13.124.107.219:3000/login". The page features the "Kustagram" logo in large red letters. Below the logo are two input fields: "아이디 : 123123" and "비밀번호 : ...". At the bottom are two buttons: "로그인" and "회원가입". A modal dialog box is open in the center, displaying the message "13.124.107.219:3000 내용: 로그인 실패" and a "확인" button.



## 2. 로그인 / 로그아웃 기능 구현

### 로그인 기능 구현

먼저 index.ejs의 header-setting (아까 로그아웃 버튼 만든 곳) 에 아래와 같이 추가

```
</div>
<div class="header-setting">
  <%=username%>
  <input type="button" value="로그아웃" />
</div>
```

그리고 route.js 에서 app.get('/',function(req,res){ ~ 부분을 아래와 같이 수정

```
app.get('/',function(req,res){
  if(typeof req.user == 'undefined'){
    console.log('Unauthorized access-main');
    res.redirect('/login');
  } else {
    res.render('index',{
      title: "insta_project",
      username: req.user.user_id
    });
  }
});
```



## 2. 로그인 / 로그아웃 기능 구현

### 로그인 기능 구현

먼저

<http://본인ip:3000> 을 치고 들어가려고 하면 로그인 화면으로 자동으로 튕겨 나가  
는 것을 확인 할 수 있다.



## 2. 로그인 / 로그아웃 기능 구현

### 로그인 기능 구현

로그인을 해보면

# KUstagram

아이디 :


비밀번호 :

로그아웃 버튼 옆에 내 아이디가 뜬다

admin1

---

40분전







## 2. 로그인 / 로그아웃 기능 구현

### 로그아웃 기능 구현

로그아웃 기능은 쉽다.

먼저 index.ejs 에서 로그아웃 버튼에 onclick 속성을 아래와 같이 추가한다.

```
<div class="header-setting">
  <%=username%>
  <input type="button" onclick="location.href='/logout'" value="로그아웃" />
</div>
```

그리고 route.js 에 아래를 추가한다.

```
app.get('/logout', function(req, res){
  req.logout();
  res.redirect('/login');
});
```



다음 시간에는...

1. 파일 업로드

2. 게시물 등록