# 생활코딩

# Node.js 노드제이에스 프로그래밍

## Gachon Shop – gcshop 프로젝트

2023.10.26

# ※ 프로젝트

* **routing**
* 인증
  - **login**
* 관리자 기능 – 사용자 관리, **DB**에 테이블 **CRUD**, 상품 입력, 코드 테이블 관리
* 게시판 생성 및 게시판에 글 **CRUD**
* 상품 조회 및 구매
* 장바구니 기능
* 경영자 화면

# ※ 8주차 수업의 범위

* routing

* project

  - login

# 01 라우팅

1. 현업에서 사용하는 라우터(URL 분류기)는 1000개 그 이상으로 늘어날 수 있음.
2. 주소 체계 변경 - 계층형으로 수정

| | 사용된 URL |
|---|---|
| '/' | '/' |
| | '/page/:pageId' |
| | '/login' |
| | '/login_process' |
| | '/logout_process' |
| | '/create' |
| | '/create_process' |
| | '/update/:pageId' |
| | '/update_process' |
| | '/delete/:pageId' |
| '/author' | '/author' |
| | '/author/create_process' |
| | '/author/update' |
| | '/author/update_process' |
| | '/author/delete' |

## 3. main.js 파일 수정

- 홈디렉토리에 router 폴더를 생성

```
//사용자 정의 모듈
var db = require('./lib/db');
// var topic = require('./lib/topic');   주석처리
// var author = require('./lib/author'); 주석처리
var authorRouter = require('./router/authorRouter'); 추가
var rootRouter = require('./router/rootRouter');    추가
```

```
// app.get('/',(req,res)=>{                    모든 URL 분류기를 주석처리
//       topic.home(req,res);

// })

// app.get('/page/:pageId',(req,res)=>{
//       topic.page(req,res);
// })
```

```
app.use('/',rootRouter);                    두 줄 추가
app.use('/author',authorRouter);
```

## 4. router/rootRouter.js 파일 생성

```
const express = require('express');
var router = express.Router()

var topic = require('../lib/topic');

router.get('/',(req,res)=>{
    topic.home(req,res);

})

router.get('/page/:pageId',(req,res)=>{
    topic.page(req,res);
})

router.get('/login',(req,res)=>{
  topic.login(req,res);

})

router.post('/login_process',(req,res)=>{
  topic.login_process(req,res);

})

router.get('/logout_process',(req,res)=>{
  topic.logout_process(req,res);
})
```

```
router.get('/create',(req,res)=>{
    topic.create(req,res);
})

router.post('/create_process',(req,res)=>{
    topic.create_process(req,res);
})

router.get('/update/:pageId',(req,res)=>{
    topic.update(req,res);
})

router.post('/update_process',(req,res)=>{
    topic.update_process(req,res);
})

router.get('/delete/:pageId',(req, res)=>{
    topic.delete_process(req, res);
})

module.exports = router;
```

※ **router/authorRouter.js** 파일을 생성하시오

# 02 정적 파일

1. 정적 파일 - 이미지, 자바스크립트, **CSS** 와 같은 파일
2. **public** 폴더 생성하고 하위에 **images** 폴더 생성

   **upsplash** 사이트에서 이미지 다운로드
3. **main.js**에 다음 코드 추가 후 **http://localhost:3000/images/adrian.jpg** 들어가기

```
// 정적 파일
app.use(express.static('public'));
```

4. **home.ejs**에 다음 **img** 태그 추가

```
<h1><a href="/"><%=title%></a></h1>
<img src="/images/adrian.jpg" style="width:200px;height: 100px;display :block;margin-top:10px;">
<a href="/author">저자관리</a>
```

## 5. 파일 업로드

① multer 모듈 설치

```
npm install -s multer
```

② uploadtest.ejs 준비

```
<body>
    <%=lg%>
    <script>

        function displayFileName(){
            var fileName = $("#file").val();
            alert(fileName);
            $(".upload-name").val(fileName);
        }
    </script>
    <form action="/upload_process" method="post", enctype="multipart/form-data">
    <div class="filebox">
        <input class="upload-name" value="" placeholder="첨부파일">
        <label for="file">파일찾기</label>
        <input type="file" id="file" name="uploadFile" onchange="displayFileName()">
        <p><input type="submit" value="upload"></p>
    </div>
    </form>
</body>
```

## 5. 파일 업로드

③ rootRouter.js 에 다음 코드 추가

```
router.get('/upload',(req, res)=>{
    topic.upload(req, res);
})
```

④ topic.js 에 upload 메소드 추가

```
    upload: (req,res) => {
        var context = { lg: ''
                        };
        req.app.render('uploadtest',context, (err, html) => {
            res.end(html);    });
     }
```

## 5. 파일 업로드

⑤ rootRouter.js 에 다음 코드 추가

```
// 파일 upload
const multer = require('multer');

const upload = multer({
    storage: multer.diskStorage({
              destination: function (req, file, cb) { cb(null, 'public/images');  },
              filename: function (req, file, cb) {
                      var newFileName = file.originalname
                      cb(null, newFileName); }
             }),
    });
```

```
router.post('/upload_process',upload.single('uploadFile'), (req, res)=>{
    var file = '/images/' + req.file.filename
    res.send(`
        <h1>Image Upload Successfully</h1>
        <a href="/">Back</a>
        <p><img src="${file}" alt="image 출력"/></p>`
    );
    console.log(file);

})
```

## 5. 파일 업로드

### ⑥ rootRouter.js 코드 설명

```
const multer = require('multer');
```

multer 모듈을 import한다
이미지, 동영상 등을 비롯한 여러 가지 파일들을 멀티파트 형식으로 업로드 할 때 사용하는 미들웨어
멀티파트 형식이란 form 태그의 enctype이 multipart/form-data로 업로드 되는 데이터

```
<form action="/upload_process" method="post", enctype="multipart/form-data">
```

mulipart 폼을 통해 업로드하는 파일은 body-parser로는 처리할 수 없고 multer 모듈 사용해야 함.
multer 패키지 안에는 여러 종류의 미들 웨어가 들어 있음.

## 5. 파일 업로드

### ⑥ rootRouter.js 코드 설명

```
const upload = multer({
    storage: multer.diskStorage({
              destination: function (req, file, cb) { cb(null, 'public/images');  },
              filename: function (req, file, cb) {
                      var newFileName = file.originalname
                      cb(null, newFileName); }
             }),
    });
```

기본 설정 – multer 함수의 인수로 설정을 넣는다.
storage 속성 – 어디에(destination) 어떤 이름으로(filename) 저장할지를 넣는다.
destination과 filename함수 – req : 요청에 대한 정보
file : 업로드 한 파일에 대한 정보
cb : 함수, 첫번째 인수에는 에러가 있다면 에러를 넣고, 두 번째 인수에는 실제 경로나 파일 이름
req, file의 데이터를 가공해서 cb에 넘기는 방식

설정이 끝나면 upload 변수 생성
upload 변수에 다양한 종류의 미들웨어가 들어 있음

upload.single 미들웨어 – 하나의 파일만 업로드 할 때 사용
single 미들웨어를 라우터 미들웨어 앞에 넣어두면 multer 설정에 따라 파일 업로드 후 req.file 객체가 생성

## 5. 파일 업로드

### ⑥ rootRouter.js 코드 설명

```
router.post('/upload_process',upload.single('uploadFile'), (req, res)=>{
    var file = '/images/' + req.file.filename
    res.send(`
        <h1>Image Upload Successfully</h1>
        <a href="/">Back</a>
        <p><img src="${file}" alt="image 출력"/></p>`
    );
    console.log(file);

})
```

upload.single 미들웨어 – 하나의 파일만 업로드 할 때 사용
single 미들웨어를 라우터 미들웨어 앞에 넣어두면 multer 설정에 따라 파일 업로드 후 req.file 객체가 생성
인수는 input 태그의 name

업로드 성공 시 결과는 req.file 객체 안에 들어 있습니다. req.body에는 파일이 아닌 데이터가 들어 있음.

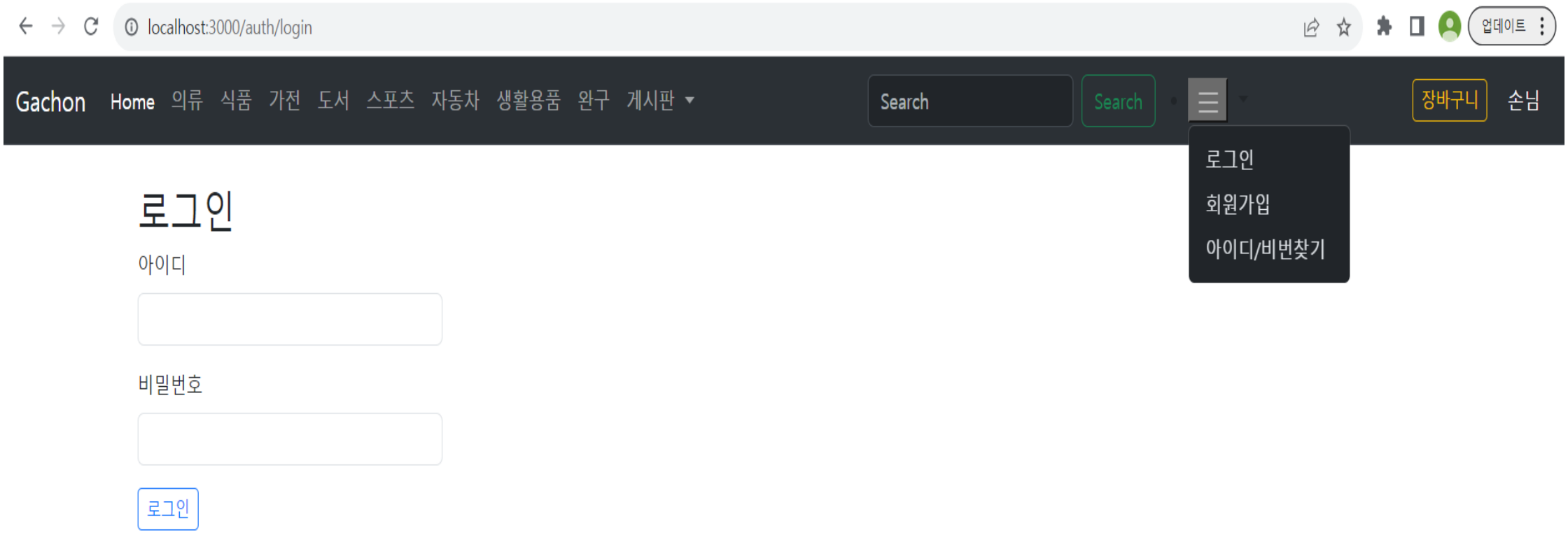
# Project – Gachon Shop(gcshop)

person table 생성 – 사용자 테이블(회원, 관리자, 경영자 모두 저장)

| 필드명 | 데이터형 |
|---|---|
| loginid | varchar(10) NOT NULL |
| password | varchar(20) NOT NULL |
| name | varchar(20) NOT NULL |
| address | varchar(50) |
| tel | varchar(13) |
| birth | varchar(8) NOT NULL |
| class | varchar(2) NOT NULL |
| grade | varchar(2) NOT NULL |

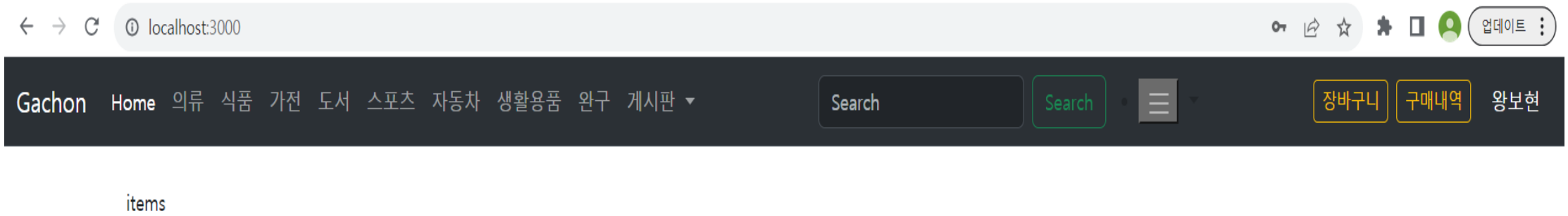
# Project – Gachon Shop(gcshop)

## 로그인 화면



# Project – Gachon Shop(gcshop)

## 로그인 후 화면



# Project – Gachon Shop(gcshop)

## 1. 프로젝트 생성 작업 순서

① person 테이블 생성

② gcshop 폴더 생성

③ 기존 작업 폴더의 package.json 파일을 복사하여 gcshop 폴더에 붙여넣기

④ vs code에서 gcshop 폴더를 열고 npm install 실행하기

⑤ gcshop 폴더에 views, lib, router, public 폴더 생성하기

## 2. 로그인 기능 구현 순서

① main.js 작성

```
//express와 views 정의
const express = require('express')   ;
const app = express()    ;
app.set('views',__dirname + '/views');
app.set('view engine','ejs');

//사용자 정의 모듈
var rootRouter = require('./router/rootRouter');
var authRouter = require('./router/authRouter');
```

## 2. 로그인 기능 구현 순서

① main.js 작성

```
// 세션 모듈, 세션 DB 저장 모듈
var session = require('express-session');
var MySqlStore = require('express-mysql-session')(session);
var options = {
    host : 'localhost',
    user : 'nodejs',
    password : 'nodejs',
    database : 'webdb2023'
    };
var sessionStore = new MySqlStore(options);

app.use(session({
    secret : 'keyboard cat',
    resave : false,
    saveUninitialized : true,
    store : sessionStore
 }));

// body 파서 모듈
var bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({extended: false }));

// 라우터 호출
app.use('/',rootRouter);
app.use('/auth',authRouter);

app.listen(3000, () => console.log('Example app listening on
port 3000'))
```

## 2. 로그인 기능 구현 순서

② /router/rootRouter.js 작성

```
const express = require('express');
var router = express.Router()

var shop = require('../lib/shop');


router.get('/',(req, res)=>{

    shop.home(req, res);
});


module.exports = router;
```

## 2. 로그인 기능 구현 순서

③ /router/authRouter.js 작성

```
const express = require('express');
var router = express.Router()

var auth = require('../lib/auth');

router.get('/login',(req, res)=>{
    auth.login(req, res);
});

router.post('/login_process',(req, res)=>{
    auth.login_process(req, res);
});

router.get('/logout_process',(req,res)=>{
    auth.logout_process(req,res);
});

module.exports = router;
```

## 2. 로그인 기능 구현 순서

④ /lib/db.js 작성

```
var mysql = require('mysql');
var db = mysql.createConnection({
    host     : 'localhost',
    user     : 'nodejs',
    password : 'nodejs',
    database : 'webdb2023'
});
db.connect();
module.exports = db;
```

## 2. 로그인 기능 구현 순서

⑤ /lib/auth.js 작성

```
var db = require('./db');

module.exports = {
    login : (req,res)=>{
        var context = {
                menu : 'menuForCustomer.ejs',
                who : '손님',
                body : 'login.ejs',
                logined : 'NO'
              };
            req.app.render('home',context, (err, html)=>{
                res.end(html); })
    },
```

## 2. 로그인 기능 구현 순서

⑤ /lib/auth.js 작성

```
    login_process : (req,res)=>{
        var post = req.body;

        db.query('select count(*) as num from person where loginid = ? and password
= ?',[post.id,post.pwd],(error, results)=>{
            if (results[0].num === 1){
                db.query('select name, class from person where loginid = ? and password
= ?',[post.id,post.pwd],(error, result)=>{
                    req.session.is_logined = true;
                    req.session.name = result[0].name
                    req.session.class = result[0].class
                    res.redirect('/');
                })
            }
            else {
                req.session.is_logined = false;
                req.session.name = '손님';
                req.session.class = '99';
                res.redirect('/');
            }
        })
    },
    logout_process : (req, res) => {
        req.session.destroy((err)=>{
           res.redirect('/');
        })
     }
}
```

## 2. 로그인 기능 구현 순서

⑥ /lib/shop.js 작성

```
var db = require('./db');

function authIsOwner(req,res)
{
    if(req.session.is_logined)
        {
            return true;
        }
    else {
        return false }
}
```

```
module.exports = {
    home : (req,res)=>{
        var isOwner = authIsOwner(req,res);
        if (isOwner){
            if(req.session.class ===
'00'){
                var context = {
                    menu :
'menuForManager.ejs',
                    who : req.session.name,
                    body : 'items.ejs'
                    };

            }
            else if(req.session.class ===
'01'){
                var context = {
                    menu :
'menuForCustomer.ejs',
                    who : req.session.name,
                    body : 'items.ejs'
                    };

            }
            else if(req.session.class ===
'02'){
                var context = {
                    menu :
'menuForCustomer.ejs',
                    who : req.session.name,
```

```
        else
        {

            var context = {
                menu : 'menuForCustomer.ejs',
                who : '손님',
                body : 'items.ejs',
                logined : 'NO'
                };
        }
        req.app.render('home',context, (err,
html)=>{
            res.end(html); })
    }

}
```

## 2. 로그인 기능 구현 순서

⑦ /views/menuForCustomer.ejs를 수정하기, items.ejs 생성하기