

# Node.js 프로그래밍

## • Node.js란

Node.js는 구글의 Chrome Javascript Engine인 V8 Engine을 기반으로 구현된 일종의 Server Side Javascript 소프트웨어 시스템이다. Java언어가 각각의 OS 운영체제의 JVM(Java Virtual Machine)을 통해 구동되듯이 Node.js는 웹브라우저에 종속적인 Javascript를 각각 OS 운영체제에서 실행할 수 있도록 Javascript Runtime 환경을 제공하고 있다.

## • 노드 설치하기

노드 공식 사이트(<https://nodejs.org/ko/>)에 접속하여 프로그램을 다운로드 받아 설치한다.

## • 노드 환경 구성하기

노드를 설치했으므로 이제 노드의 패키지 관리자인 npm을 사용하여 익스프레스 프로젝트를 자동으로 생성해 주는 express-generator 모듈을 설치한다. 참고로 익스프레스는 노드 기반에서 웹 애플리케이션 개발을 지원하는 프레임워크이다. npm의 기본 사용법은 아래와 같다.

### • npm install [모듈명] -g

모듈을 전역으로 설치할 때 사용하는 방식이다. 일반적으로 특정 프로젝트가 아닌 전체 프로젝트에서 공통으로 사용할 수 있는 익스프레스 템플릿 생성 모듈인 express-generator나 노드 프로젝트를 실행하는 nodemon 등을 전역으로 설치한다.

### • npm install [모듈명] --save

현재 작업 중인 프로젝트에 모듈을 설치할 때 사용하는 방식이다. --save 옵션으로 현재 프로젝트의 package.json에 모듈 이름과 버전을 추가한다. 이를 통해 현재 프로젝트가 사용하고 있는 모듈 이름과 버전을 알 수 있고 나중에 동일한 프로젝트를 다른 곳에 생성할 때 package.json을 복사해서 설치하면 되므로 매우 편리하다.

### • npm install

프로젝트 package.json에 작성된 모듈을 한 번에 설치하고 싶을 때 사용하는 명령어이다. package.json만 있다면 같은 버전의 모듈을 손쉽게 설치할 수 있다.

1) 프로젝트 디렉터리를 생성한 후 생성된 디렉터리로 이동한다.

```
C:\Wproject>
```

2) 익스프레스 프로젝트를 쉽게 생성해 주는 express-generator 모듈을 전역으로 설치한다. 전역으로 설치하면 다른 프로젝트에서도 사용할 수 있다. 사용법은 express -h로 살펴볼 수 있다.

```
C:\Wproject>npm install express-generator -g
```

3) express 명령어를 실행해서 프로젝트 디렉터리를 생성한다. 이때 템플릿 엔진은 ejs로 설치할 것이므로 -e 옵션을 추가한다. 이렇게 express-generator를 사용해서 프로젝트를 생성하면 기본적인 모듈들이 자동으로 package.json에 추가된다.

```
C:\Wproject>express -e web
```

4) package.json에 추가된 모듈을 설치한다. npm install 명령어를 사용하면 package.json에 선언된 모듈이 자동으로 설치된다. 이때 명령어는 web 디렉터리 아래에서 실행한다.

```
C:\Wproject\Wweb>npm install
```

5) 추가로 필요한 모듈을 설치한다. 이때는 설치 옵션에 --save를 주어 package.json에 설치된 모듈을 추가해야한다. 이렇게 추가해 놓으면 어떤 모듈을 사용하고 있는지 알 수 있으며 나중에 동일한 프로젝트를 다른 곳에 생성할 때 package.json을 복사해서 설치하면 매우 편리하다. 파일 업로드 모듈인 formidable과 mysql 연결 모듈인 mysql을 함께 설치한다.

```
C:\Wproject\Wweb>npm install formidable mysql --save
```

6) 노드는 npm start 명령어를 사용해서 실행할 수 있다. 참고로, 실행을 종료하고 싶다면 CTRL + C를 입력하면 된다.

```
C:\Wproject\Wweb>npm start
```

- 포트를 강제 종료하려면 아래와 같이 포트번호의 해당 pid번호 검색 후 포트번호를 kill한다. (`npm kill-port 3000`)

```
netstat -ano|findstr :3000 (포트번호)
taskkill /f /pid 18064 (pid번호)
```

- 결과는 브라우저에 `http://localhost:3000`으로 접속해서 확인할 수 있다. 노드는 기본적으로 3000번 포트에서 동작하도록 되어 있다. 다른 포트 번호로 변경하려면 C:\Wproject\Wweb\Wbin 디렉터리의 `www` 파일의 내용을 수정해야 한다.

7) `npm start`로 노드를 실행하면 소스 코드를 수정한 것이 반영되지 않는다. 그래서 이런 경우를 위한 여러 모듈이 있는데 그중에서 간단히 사용할 수 있는 노드몬을 설치한다.

```
C:\Wproject\Wweb>npm install nodemon -g
```

- 프로젝트 디렉터리에서 `nodemon`이라고 실행하면 소스 코드가 수정될 경우 수정을 감지하고 다시 시작하는 것을 확인할 수 있다. 따라서 소스 코드를 변경할 때마다 서버를 재 시작해야 하는 번거로움을 줄일 수 있다.

```
C:\Wproject\Wweb>nodemon
```

- `nodemon` 실행 오류 시에는 관리자 권한으로 PowerShell을 실행한다.

```
C:\Windows\System32>ExecutionPolicy (현재정책확인)
C:\Windows\System32>Set-ExecutionPolicy Unrestricted (Restricted를 Unrestricted로 변경한다.)
```

## • 프로젝트의 기본 디렉터리

- 1) `bin` 디렉터리 : `bin` 디렉터리에는 `www` 이름의 파일이 한 개 존재한다. 이 파일은 확장자가 없지만, 내부는 노드가 서버로서 동작하기 위한 기본적인 코드가 자바스크립트로 작성되어 있다. 또한 서버를 시작할 포트가 지정되어 있다.
- 2) `node_modules` 디렉터리 : 이 디렉터리는 `npm install`을 실행하면서 생긴 디렉터리이다. 그래서 `package.json`에 선언되어 있는 모듈과 이 모듈을 실행하기 위해서 필요한 의존관계의 모듈이 설치되어있다.
- 3) `public` 디렉터리 : 이 디렉터리에는 이미지, 자바스크립트, 스타일시트와 관련된 디렉터리로 구성되어있다.
- 4) `routes` 디렉터리 : 이 디렉터리에는 `index.js`와 `users.js` 파일이 존재한다. 기본적으로 생성되는 파일이며, 라우트를 처리하기 위한 코드가 작성되어 있다.
- 5) `views` 디렉터리 : 뷰를 처리하는 파일이 위치한 곳이며, 프로젝트를 생성할 때 템플릿 엔진으로 `ejs`를 지정했으므로 이 디렉터리에는 확장자가 `ejs`인 파일이 위치한다.

## • 프로젝트의 기본 파일

- 1) `bin/www`에 위치한 `www` 파일

이 파일에서는 기본적으로 노드를 실행할 포트를 설정하고 웹서버를 생성하는 코드가 작성되어 있는 자바스크립트 파일이다.

```
var app = require('..../app');
var debug = require('debug')('web:server');
var http = require('http');

var port = normalizePort(process.env.PORT || '3000'); //환경 설정에 지정된 포트가 있다면 이를 사용하고 없다면 지정한 포트(3000)를 사용하도록 설정
app.set('port', port);

var server = http.createServer(app); //HTTP 서버 생성

server.listen(port); //지정된 포트 상에서 응답 대기하도록 설정
```

- 2) `app.js`

이 파일은 웹 애플리케이션을 위한 기본적인 설정을 가지고 있는 파일이다. 모듈을 로딩하고 템플릿 엔진을 설정하며, 라우트를 설정한다. 파일 상단부분에는 사용할 모듈을 로딩 하는 코드가 작성되어있다. 외부 모듈을 해당 파일에서 사용하고 싶다면 `require()` 함수를 호출해야 한다.

```
var express = require('express');
var path = require('path');
var cookieParser = require('cookie-parser');
var logger = require('morgan');
```

라우트 코드를 로딩 하는 코드이다 require() 함수를 사용하며, 로딩 한 라우트 함수들을 지정된 변수로 사용할 수 있게 해 준다.

```
var indexRouter = require('./routes/index');
var usersRouter = require('./routes/users');
```

익스프레스 객체를 app 변수로 선언한다. 앞으로 app을 통해 익스프레스 함수를 호출할 수 있게 된다.

```
var app = express();
```

익스프레스에서 사용할 템플릿 엔진을 설정하는 코드이다. \_\_dirname은 현재 디렉토리를 의미하며 path.join() 함수는 경로를 연결하는 기능을 한다. 그래서 현재 디렉토리에 있는 views 디렉토리를 의미한다. 그리고 app.set() 함수는 익스프레스의 환경을 설정하는 함수이다. 첫 번째 라인은 경로를 두 번째 라인은 템플릿 엔진의 종류를 설정한다. 참고로 app.get() 함수도 정의되어 있으며, 이 함수는 값을 반환한다.

```
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
```

app.use() 함수는 지정된 인자를 실행하는 함수이다. 여기서는 각각의 모듈을 사용하도록 설정하고 있다.

```
app.use(logger('dev'));
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());
app.use(express.static(path.join(__dirname, 'public')));
```

라우트를 설정하는 코드이다. 여기서 말하는 라우트는 url 경로의 뒷부분을 의미한다. 그래서 '/'와 관련된 라우트는 routes 폴더의 index 파일에 설정된 라우트 함수를 통해 처리되며 '/users'와 관련된 라우트는 users 파일에 작성된 라우트 함수를 통해 처리된다.

```
app.use('/', indexRouter);
app.use('/users', usersRouter);
```

### 3) /routes/index.js

'app.js'에서 라우트를 처리할 때는 app.get()나 app.post() 함수를 사용한다. 그리고 별도 파일에서 라우트 함수를 작성할 경우에는 express.Router() 함수를 통해 호출하며 별도 파일에서 함수를 사용할 수 있도록 module.export=router를 추가해야 한다.

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

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

module.exports = router;
```

### 4) /db.js

데이터베이스를 연동하는 방법은 두 가지가 있다. 첫 번째는 매번 데이터베이스의 커넥션을 얻어서 사용하는 방법이고, 두 번째는 커넥션 풀을 생성해서 커넥션을 미리 생성한 후에 하나씩 꺼내서 사용하는 방법이다. 커넥션 생성 비용은 매우 크므로 가능하면 커넥션 풀을 사용하는 것이 좋다. 이 파일에는 두 개의 함수가 있다. connect는 데이터베이스 커넥션 풀을 생성하는 함수이고, get은 생성한 커넥션 풀을 반환하는 함수이다. get 함수를 통해 풀을 반환받은 후에 데이터베이스에 질의를 실행할 수 있다.

데이터베이스, 사용자 생성

```
create database webdb;
create user web identified by 'pass';
drop user web;
grant all privileges on webdb.* to web@'%';
```

```

var mysql = require('mysql');
var config = {
  connectionLimit:100,
  host:'localhost',
  user:'root',
  password:'1234',
  database:'webdb',
  port:'3306'
}

var pool = mysql.createPool(config);
var connection;
exports.connect = function() {
  pool.getConnection(function(err, con) {
    if(err) console.log('db접속 오류:', err)
    else connection = con;
  });
}

exports.get = function() {
  return connection;
};

```

//카페24 데이터베이스 사용한 경우

```

var config = {
  connectionLimit:100,
  host:'hd9536.cafe24.com',
  user:'xxxx',
  password:'xxxx!',
  database:'xxxx',
  typeCast: function (field, next) {
    if (field.type == 'VAR_STRING') return field.string();
    return next();
  }
}

```

connect 함수는 한 번만 호출하면 되므로 **app.js**에서 db.js 모듈을 로딩한 후에 connect 함수를 실행한다.

```

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

```

라우트 파일에서 get함수를 호출한 후에 query 함수를 호출해서 질의를 실행한다.

[usertinfo] 테이블 생성

```

create table userinfo(
  id varchar(20) not null primary key,
  password varchar(20) not null,
  name varchar(20)
);

```

```

var express = require('express');
var router = express.Router();
var db = require('../db');

router.get('/list.json', function(req, res) {
  var query='%'+ req.query.keyword + '%';
  var sql='select * from userinfo where id like ?';
  db.get().query(sql, [query], function(err, rows) {
    if(err) return res.sendStatus(400);
    res.status(200).json(rows); //res.status(200).send(rows) 또는 res.send(rows)
  });
});

router.post('/insert', function(req, res) {
  var id=req.body.id;
  var password=req.body.password;
  var name=req.body.name;
  var sql='insert into userinfo(id, password, name) values(?, ?, ?)';
  db.get().query(sql, [id, password, name], function(err, result) {
    if(err) return res.sendStatus(400);
    res.sendStatus(200);
  });
});

module.exports = router;

```

## • Oracle DB 접속

- 1) 오라클 사이트로 가서 Oracle Instant Client Downloads 페이지에서 2가지 종류의 파일을 받아야 한다.
  - SDK Package 라고 불리 우는 압축 파일 (파일 이름이 instantclient-sdk-...로 시작한다.)
  - Basic Package 라고 불리 우는 압축 파일 (파일 이름이 instantclient-basic-...로 시작한다.)

```
https://www.oracle.com/database/technologies/instant-client.html
```

- 2) c:/oraclexe/client라는 폴더를 생성한 후 압축이 풀린 2개의 디렉터리의 내용을 이곳으로 옮긴다.
- 3) 환경변수에 c:/oraclexe/client 폴더를 추가해준다. 추가 후 가장 위쪽으로 이동시킨다.
- 4) oracledb를 설치한다.

```
npm install oracledb
```

- 5) /db.js 파일을 생성 후 아래 내용을 입력한다.

```
var oracledb = require('oracledb');
oracledb.outFormat = oracledb.OBJECT;
var connection;
exports.connect = function() {
  oracledb.getConnection({
    user: 'system',
    password: '1234',
    connectString: 'localhost/xepdb1'
  },function(err,con) {
    if(err) {
      console.log('접속이 실패했습니다.', err);
    }
    conn = con;
  });
}

exports.get=function(){
  return connection;
};
```

- 6) /app.js 파일에 아래 내용을 추가한다.

```
var db = require('./db');
db.connect(function(err) {
  if(err){
    console.log('Unable to connect to Oracle');
    process.exit(1);
  }
});
```

- 7) 라우트 파일에서 get함수를 호출한 후에 query 함수를 호출해서 질의를 실행한다.

```
var express = require('express');
var router = express.Router();
var db = require("../db");

router.get('/', function(req, res) {
  db.get().execute('select * from students', function(err, result) {
    res.send(result.rows);
  });
});

module.exports = router;
```

## • 사용자 관리 프로그램

app.js

```
...
app.use('/', indexRouter);
app.use('/users', usersRouter);
app.use('/user', require('./routes/user'));
...
```

[routes] user.js

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

router.get('/', function(req, res, next) {
  var sql="select * from userinfo";
  db.get().query(sql, function(err, rows) {
    if(err) return res.sendStatus(400);
    console.log(rows);
    res.render('user/list', { users: rows });
  });
});

router.get('/insert', function(req, res, next) {
  res.render('user/insert', { title: "[사용자 등록]" });
});

//사용자 등록
router.post('/insert', function(req, res) {
  var id=req.body.id;
  var password=req.body.password;
  var name=req.body.name;
  console.log('id:' + id + ',password:' + password + ',name:' + name);
  var sql='insert into userinfo(id, password, name) values(?, ?, ?)';
  db.get().query(sql, [id, password, name], function(err, result) {
    if(err) return res.sendStatus(400);
    res.redirect('/user')
  });
});

module.exports = router;
```

## • 크롬 JSON Viewer 프로그램 추가하기

chrom 웹 스토어에서 아래의 'JSON Viewer'를 Chrom에 확장 프로그램에 추가한다.



Chrome에 추가

[views]-[user] list.ejs

```
<head><title>사용자목록</title></head>
<body>
  <h1>[사용자 목록]</h1>
  <ul>
    <% if(users == null || users.length == 0){ %>
      <li>등록된 사용자가 없습니다.</li>
    <% } %>
    <% users.forEach(function(user) { %>
      <li><%=user.id%> ( <%=user.name%> ) </li>
    <% }); %>
  </ul>
  <button onClick="location.href='/user/insert'">입력</button>
</body>
```

[views]-[user] inser.ejs

```
<html>
  <head>
    <title><%=title%></title>
  </head>
  <body>
    <h1><%=title%></h1>
    <form action="insert" method="post">
      <table border=1 width=500>
        <tr> <td width=100>아이디</td><td><input type="text" name="id"></td></tr>
        <tr><td width=100>비밀번호</td><td><input type="password" name="password"></td></tr>
        <tr><td width=100>이름</td><td><input type="name" name="name"></td></tr>
      </table>
      <input type="submit" value="저장"/>
      <input type="reset" value="취소"/>
      <input type="button" value="목록" onclick="location.href='/user'"/>
    </form>
  </body>
</html>
```

app.js

```
...
app.use('/userinfo', require('./routes/userinfo'));
...
```

[routes] userinfo.js

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

router.get('/', function(req, res, next) {
  res.render('userinfo', { title: '사용자관리' });
});

//전체사용자 출력
router.get('/list.json', function(req, res) {
  var query='%'+ req.query.keyword + '%';
  //console.log('query:' + query);
  var sql='select * from userinfo where id like ?';
  db.get().query(sql, [query], function(err, rows) {
    if(err) return res.sendStatus(400);
    //console.log("rows:" + JSON.stringify(rows) + ":" + row.length);
    res.status(200).json(rows);
  });
});

//사용자 등록
router.post('/insert', function(req, res) {
  var id=req.body.id;
  var password=req.body.password;
  var name=req.body.name;
  //console.log('id:' + id + ',password:' + password + 'name:' + name);
  var sql='insert into userinfo(id, password, name) values(?, ?, ?)';
  db.get().query(sql, [id, password, name], function(err, result) {
    if(err) return res.sendStatus(400);
    res.sendStatus(200);
  });
});
```



```

<html>
  <head>
    <title><%=title%></title>
    <script src="http://code.jquery.com/jquery-3.1.1.min.js"></script>
  </head>
  <body>
    <h1>[사용자등록]</h1>
    <form>
      <table border=1 width=500>
        <tr><td>아이디</td><td><input type="text" id="id"></td></tr>
        <tr><td>비밀번호</td><td><input type="password" id="password"></td></tr>
        <tr><td>이름</td><td><input type="text" id="name"></td></tr>
      </table>
      <input type="button" value="저장" id="btnSave"/>
      <input type="reset" value="취소"/>
    </form>
    <hr/>
    <h1>[사용자목록]</h1>
    <table border=1 width=500 id="tbl"></table>
  </body>
  <script>
    getList();
    $("#btnSave").on("click", function() {
      var id=$("#id").val();
      var password=$("#password").val();
      var name=$("#name").val();
      $.ajax({
        type: "post",
        url: "/userinfo/insert",
        data: { "id": id, "password":password, "name":name },
        success: function() {
          alert("저장되었습니다!");
          getList();
        },
        error:function() {
          alert("저장을 실패했습니다!");
        }
      });
    });
    function getList() {
      $.ajax({
        type: "get",
        url: "/userinfo/list.json",
        datatype: "json",
        data: { "keyword": "" },
        success: function(data) {
          var str="";
          $(data).each(function() {
            var id=this.id;
            var name=this.name;
            var password=this.password;
            str += "<tr>";
            str += "<td>" + id + "</td>";
            str += "<td>" + name + "</td>";
            str += "<td>" + password + "</td>";
            str += "</tr>";
          });
          $("#tbl").html(str);
        }
      });
    }
  </script>
</html>

```

## • 상품관리 프로그램

### [productinfo] 테이블 생성

```
create table productinfo(  
  code char(4) primary key,  
  pname varchar(100) not null,  
  price int,  
  image varchar(50)  
);
```

### [productinfo] Sample 데이터 입력

```
insert into productinfo(code, pname, price, image)  
values('P001', '5단서랍장', 50000, 'img01.jpg');
```

```
insert into productinfo(code, pname, price, image)  
values('P002', '월목싱글침대', 800000, 'img02.jpg');
```

```
insert into productinfo(code, pname, price, image)  
values('P003', '싱글침대', 800000, 'img03.jpg');
```

```
insert into productinfo(code, pname, price, image)  
values('P004', '4인용 가죽 쇼파', 1800000, 'img04.jpg');
```

### 1) 상품목록 출력 프로그램

#### app.js 아래내용 추가

```
app.use('/product', require('./routes/product'));
```

#### [routes] product.js 생성

```
var express = require('express');  
var router = express.Router();  
var db = require('../db');  
  
router.get('/', function(req, res, next) {  
  res.render('product/list', { title: '상품관리' });  
});  
  
router.get('/list.json', function(req, res) {  
  var sql='select * from productinfo';  
  db.get().query(sql, function(err, rows) {  
    if(err) return res.sendStatus(400);  
    res.status(200).json(rows);  
  });  
});  
  
module.exports = router;
```

#### [views]-[product] product.css 생성

```
#container {  
  width: 680px;  
  background: gray;  
  padding: 10px;  
  overflow:hidden;  
}  
  
.box {  
  width: 150px;  
  background: pink;  
  padding: 5px;  
  margin: 5px;  
  float: left;  
}
```

[views]-[product] list.ejs 생성

```
<html>
  <head>
    <title>상품목록</title>
    <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/3.0.1/handlebars.js"></script>
    <link rel="stylesheet" href="product.css"/>
  </head>
  <body>
    <h1>[상품목록]</h1>
    <a href="/product/insert">상품등록</a>
    <hr/>
    <div id="container"></div>
    <script id="temp" type="text/x-handlebars-template">
      {{#each .}}
        <div class="box">
          
          <div>{{code}}</div>
          <div>{{pname}}</div>
          <div>{{price}}원</div>
        </div>
      {{/each}}
    </script>
  </body>

  <script>
    getList();
    function getList() {
      $.ajax({
        type: "get",
        url: "/product/list.json",
        dataType: "json",
        success: function(data) {
          var temp=Handlebars.compile($("#temp").html());
          $("#container").html(temp(data));
        }
      });
    }
  </script>
</html>
```

## 2) 상품등록 프로그램

[views]-[product] product.css 아래 내용 추가

```
...
table {
  border-collapse: collapse;
}
td {
  border: solid 1px black;
  height: 30px;
}
.title {
  background: gray;
  color: white;
  text-align: center;
}
```

### • 파일 업로드를 위한 라이브러리 추가

C:\Wproject\Wweb>npm install multer

[views]-[product] insert.ejs 생성

```
<head>
  <script src="http://code.jquery.com/jquery-3.1.1.min.js"></script>
  <title>상품등록</title>
  <link rel="stylesheet" href="product.css"/>
</head>
<body>
  <h1>[상품등록]</h1>
  <form name="frm" action="insert" method="post" enctype="multipart/form-data">
    <table>
      <tr>
        <td width=100 class="title">상품코드</td>
        <td width=500><input type="text" name="code"></td>
      </tr>
      <tr>
        <td width=100 class="title">상품명</td>
        <td width=500><input type="text" name="pname" size=50></td>
      </tr>
      <tr>
        <td width=100 class="title">상품가격</td>
        <td width=500><input type="text" name="price">원</td>
      </tr>
      <tr>
        <td width=100 class="title">상품이미지</td>
        <td width=500>
          
          <input type="file" name="image" accept="image/*" style="visibility:hidden;">
        </td>
      </tr>
    </table>
    <input type="submit" value="저장">
    <input type="reset" value="취소">
    <input type="button" value="목록" onClick="location.href='/product/'">
  </form>
</body>

<script>
  $("#image").on("click", function() {
    $(frm.image).click();
  });
  //이미지 미리보기
  $(frm.image).on("change", function(e) {
    $("#image").attr("src", URL.createObjectURL(e.target.files[0]));
  });
</script>
```

[views]-[product] insert.ejs (Ajax을 이용한 submit)

```
<script>
  ...
  var formData = new FormData();
  formData.append('title', title);
  formData.append('price', price);
  formData.append('image', $(frm.image)[0].files[0]);
  $.ajax({
    type: 'post',
    url : '/product/insert',
    data: formData,
    processData: false,
    contentType: false,
    success: function() {
      alert('상품등록 성공!');
      location.href="/product";
    }
  });
</script>
```

#### [routes] product.js 수정

```
var multer = require('multer')
var fs = require('fs');

// ./public는 현재프로젝트 폴더의 public 폴더이다.
var uploadPath = './public/upload';

if(!fs.existsSync(uploadPath)) {
  fs.mkdirSync(dir);
}

var upload = multer({
  storage: multer.diskStorage({
    destination: (req, file, done)=> {
      done(null, uploadPath);
    },
    filename: (req, file, done)=> {
      var exe = file.originalname.substring(file.originalname.lastIndexOf('.'));
      var fileName=Date.now() + exe;
      done(null, fileName);
    },
  }),
  limits: {
    fileSize: 5 * 1024 * 1024
  },
});

router.get('/insert', function(req, res, next) {
  res.render('product/insert', { title: '상품입력' });
});

router.post('/insert', upload.single('image'), function(req, res) {
  var code=req.body.code;
  var pname=req.body.pname;
  var price=req.body.price;
  var image='';
  if(req.file != null) var image=req.file.filename;

  var sql='insert into productinfo(code,pname,price,image) values(?, ?, ?, ?)';
  db.get().query(sql, [code, pname, price, image], function(err, result) {
    if(err) return res.sendStatus(400);
    res.status(200).redirect('/product');
  });
});
```

#### [views]-[product] list.ejs 수정

```
...
<script id="temp" type="text/x-handlebars-template">
  {{#each .}}
    <div class="box">
      
      <div>{{code}}</div>
      <div>{{pname}}</div>
      <div>{{price}}원</div>
    </div>
  {{/each}}
</script>
...
<script>
  $("#container").on("click", ".box img", function() {
    var code=$(this).attr("code");
    location.href='/product/read?code=' + code;
  });
</script>
```

#### [routes] prduct.js 추가

```
router.get('/read', function(req, res, next) {  
  var code=req.query.code;  
  var sql='select * from productinfo where code=?';  
  db.get().query(sql, [code], function(err, rows) {  
    res.render('product/read', { product:rows[0] });  
  });  
});
```

#### [views]-[product] read.ejs 생성

```
<head>  
  <script src="http://code.jquery.com/jquery-3.1.1.min.js"></script>  
  <title>상품정보</title>  
  ...  
</head>  
<body>  
  <h1>[상품정보]</h1>  
  <form name="frm" action="update" method="post" enctype="multipart/form-data">  
    <table>  
      <tr>  
        <td width=100 class="title">상품코드</td>  
        <td width=500>  
          <input type="text" name="code" value="<%=product.code%>" disabled>  
        </td>  
      </tr>  
      <tr>  
        <td width=100 class="title">상품명</td>  
        <td width=500>  
          <input type="text" name="pname" value="<%=product.pname%>">  
        </td>  
      </tr>  
      <tr>  
        <td width=100 class="title">상품가격</td>  
        <td width=500>  
          <input type="text" name="price" value="<%=product.price%>">  
        </td>  
      </tr>  
      <tr>  
        <td width=100 class="title">상품이미지</td>  
        <td width=500>  
            
          <input type="file" name="image" accept="image/*" style="visibility:hidden;">  
        </td>  
      </tr>  
    </table>  
    <input type="submit" value="수정">  
    <input type="button" value="삭제" id="btnDelete">  
    <input type="reset" value="취소">  
    <input type="button" value="목록" onClick="location.href='list'">  
  </form>  
</body>
```

### 3) 상품삭제 프로그램

#### [views]-[product] read.ejs 추가

```
<script>  
  $("#btnDelete").on("click", function() {  
    if(!confirm("삭제하실래요?")) return;  
    var code = "<%=product.code%>";  
    var image = "<%=product.image%>";  
    location.href="/product/delete?code="+code + "&image=" + image;  
  });  
  ...  
</script>
```

#### [routes] product.js 추가

```
...
router.get('/delete', function(req, res) {
  var code=req.query.code;
  var image=req.query.image;
  var sql='delete from productinfo where code=?';
  db.get().query(sql, [code], function(err, result) {
    if (image != '') {
      fs.unlink(uploadPath + '/' + image, function(err) {
        if(err) throw err;
      });
    }
    res.status(200).redirect('/product');
  });
});
...
```

#### 4) 상품수정 프로그램

##### [views]-[product] read.ejs 추가

```
<body>
  ...
  <table>
    ...
    <tr>
      <td width=100 class="title">상품이미지</td>
      <td width=500>
        
        <input type="file" name="image" accept="image/*" style="visibility:hidden;">
        <input type="text" name="oldImage" value="<%=product.image%>"/>
      </td>
    </tr>
    ...
  </table>
</body>
<script>
  $(frm).on("submit", function(e) {
    e.preventDefault();
    if(!confirm("수정하실래요?")) return;
    frm.submit();
  });
  ...
</script>
```

#### [routes] product.js 추가

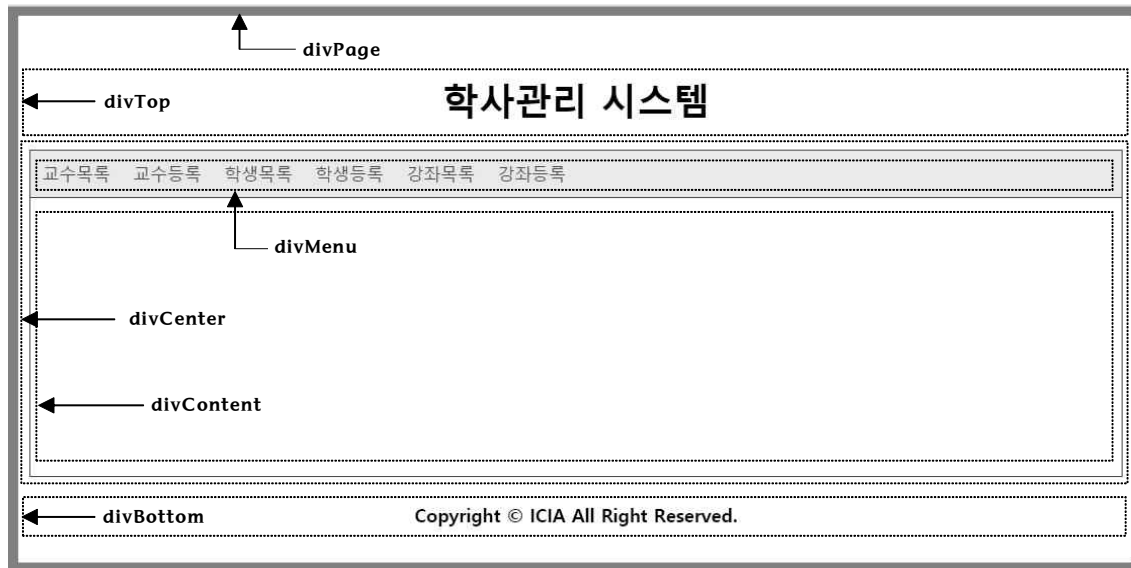
```
router.post('/update', upload.single('image'), function(req, res) {
  var code=req.body.code;
  var pname=req.body.pname;
  var price=req.body.price;
  var image=req.body.oldImage;

  //이미지가 변경된 경우
  if(req.file != null) {
    //예전 이미지 삭제
    fs.unlink(uploadPath + '/' + image, function(err) {
      if(err) throw err;
    });
    image=req.file.filename;
  }

  var sql='update productinfo set pname=?,price=?,image=? where code=?';
  db.get().query(sql, [pname, price, image, code], function(err, result) {
    if(err) return res.sendStatus(400);
    res.status(200).redirect('/product');
  });
});
```

## • 로그인/로그아웃 프로그램

### 1) 화면 레이아웃 프로그램



[public]-[stylesheets] style.css

```
body {
    background: gray;
}
#divPage {
    width:980px; background:white;
    box-shadow: 10px 10px 10px black;
    margin: auto;
}
#divTop {
    padding: 30px 0px 0px 0px;
    text-align: center;
}
#divCenter {
    border: 1px solid green;
    margin: 0px 10px 0px 10px;
    padding-bottom: 20px;
}
#divMenu {
    overflow: hidden;
    background: #EAEAEA;
    margin: 0px 0px 20px 0px;
    padding: 10px;
    border-bottom: 1px solid green;
}
#divMenu .item{
    width: 80px; float:left;
}
#divContent {
    width: 980px; height:500px;
}
#divBottom {
    padding: 0px 0px 10px 0px;
    text-align: center;
}
#divHeader {
    width: 980px;
    text-align: center;
}
a {
    color: green;
    text-decoration: none;
}
```



[views] index.ejs

```
<html>
  <head>
    <title><%=title%></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
    <script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/3.0.1/handlebars.js"></script>
  </head>
  <body>
    <div id="divPage">
      <div id="divTop">
        <h1><%=title%></h1>
      </div>
      <div id="divCenter">
        <div id="divMenu">
          <div class="item"><a href="#">교수관리</a></div>
          <div class="item"><a href="#">학생관리</a></div>
          <div class="item"><a href="#">강좌관리</a></div>
          <div class="item"><a href="#">수강신청</a></div>
          <div style="float:right;"><a href="/user/login">로그인</a></div>
        </div>
        <div id="divContent">
          <%-include(pageName)%>
        </div>
      </div>
      <div id="divBottom">
        <h4>Copyright 인천일보아카데미 All Rights Reserved.</h4>
      </div>
    </div>
  </body>
</html>
```

[views] info.ejs 생성

```
<h2><%=title%></h2>
```

[views] info.ejs 생성

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

router.get('/', function(req, res, next) {
  res.render('index', { title: '학교소개', pageName: 'info.ejs' });
});
module.exports = router;
```

## 2) 로그인 프로그램

[routes] user.js 추가

```
router.get('/login', function(req, res, next) {
  res.render('index', {title: "로그인", pageName: 'user/login.ejs' });
});

router.post('/login', function(req, res) {
  var id = req.body.id;
  var password = req.body.password;
  var sql = 'select * from userinfo where id=?';

  db.get().query(sql, [id], function(err, rows) {
    if(err) return res.sendStatus(400);
    var result=0;
    if(rows[0]!=null) {
      if(rows[0].password == password) result = 1;
      else result = 2;
    }
    res.status(200).send({ result: result });
  });
});
```

[views]-[user] login.ejs 생성

```
<body>
  <form name="frm">
    <table width=300 border=1 style="margin:0px auto; margin-top:100px;">
      <tr>
        <td colspan="2"><h1>로그인</h1></td>
      </tr>
      <tr>
        <td width=100>아이디</td>
        <td><input type="text" name="id" /></td>
      </tr>
      <tr>
        <td width=100>비밀번호</td>
        <td><input type="password" name="password" /></td>
      </tr>
      <tr>
        <td colspan="2"><input type="submit" value="로그인" /></td>
      </tr>
    </table>
  </form>
</body>
<script>
  $(frm).on("submit", function(e) {
    e.preventDefault();

    var id=$(frm.id).val();
    var password=$(frm.password).val();

    if(id == "" || password == "") {
      alert("아이디나 비밀번호를 입력하세요! ");
      return;
    }

    $.ajax({
      type: "post",
      url: "/user/login",
      data: { "id":id, "password":password },
      dataType: "json",
      success: function(data) {
        if(data.result==0) {
          alert("아이디가 존재하지 않습니다!");
        } else if(data.result==1) {
          location.href='/';
        } else if(data.result==2) {
          alert("비밀번호가 일치하지 않습니다!");
        }
      }
    });
  });
</script>
```

### 3) 아이디 세션 저장 및 로그아웃 프로그램

npm install express-session

app.js 추가

```
var session = require('express-session');
...
var app = express();
app.use(session({
  secret: 'mykey',
  resave: false, //세션 데이터가 바뀌기 전에는 세션 저장소에 값을 저장하지 않음
  saveUninitialized: true //세션이 필요한 경우에만 구동
}));
...
```

#### [routes] user.js 추가

```
router.post('/login', function(req, res) {
  var id = req.body.id;
  var password = req.body.password;
  var sql = 'select * from userinfo where id=?';

  db.get().query(sql, [id], function(err, rows) {
    if(err) return res.sendStatus(400);
    var result=0;
    if(rows[0]!=null) {
      if(rows[0].password == password) {
        result = 1;
        req.session.userid=id;
      }else {
        result = 2;
      }
    }
    res.status(200).send({ result: result });
  });
});
```

#### [routes] index.js

```
router.get('/', function(req, res, next) {
  res.render('index', {
    title: '학교소개',
    pageName: "info.ejs",
    userid:req.session.userid
  });
});
```

#### [routes] user.js

```
router.get('/login', function(req, res, next) {
  res.render('index', {
    title: '로그인',
    pageName: 'user/login.ejs',
    userid:req.session.userid
  });
});
```

#### [views] index.ejs 추가

```
<div id="divMenu">
  ...
  <%if(userid==null) {%>
    <div style="float:right;">
      <a href="/user/login">로그인</a>
    </div>
  <%}else {%>
    <div style="float:right;">
      <a href="/user/logout">로그아웃</a>
      <span>
        <b><%=userid%></b>
      </span>
    </div>
  <%}%>
</div>
...

```

#### [routes] user.js 추가

```
router.get('/logout', function(req, res) {
  req.session.destroy();
  res.clearCookie('userid');
  res.redirect('/');
});
```

#### 4) 쿠키 저장 프로그램

[views]-[user] login.js 추가

```
<body>
...
<tr>
  <td colspan="2">
    <input type="submit" value="로그인"/>
    <input type="checkbox" name="chkLogin"/>아이디저장
  </td>
</tr>
...
</body>
<script>
$(frm).on("submit", function(e) {
  var id=$(frm.id).val();
  var password=$(frm.password).val();
  var chkLogin = $(frm.chkLogin).is(":checked") ? 1 : 0;
  e.preventDefault();
  $.ajax({
    type: "post",
    url: "/user/login",
    data: { "id":id, "password":password, "chkLogin":chkLogin },
    dataType: "json",
    success:function(data) {
      if(data.result==0) {
        alert("아이디가 존재하지 않습니다!");
      }else if(data.result==1) {
        location.href='/';
      }else{
        alert("비밀번호가 일치하지 않습니다!");
      }
    }
  });
});
</script>
```

[routes] user.js 추가

```
router.post('/login', function(req, res) {
  var id = req.body.id;
  var password = req.body.password;
  var chkLogin = req.body.chkLogin;

  var sql = "select * from userinfo where id=?";
  db.get().query(sql, [id], function(err, rows) {
    if(err) return res.sendStatus(400);
    var result=0;
    if(rows[0]!=null) {
      if(rows[0].password == password) {
        result = 1;
        req.session.userid=id;
        if(chkLogin==1) res.cookie('userid', id, { maxAge: 1000 * 60 * 60 *24 });
      }else{
        result = 2;
      }
    }
    res.status(200).send({result: result});
  });
});
```

[routes] index.js 추가

```
router.get('/', function(req, res, next) {
  if(req.cookies.userid) req.session.userid=req.cookies.userid;
  res.render('index', {title:"학교소개", pageName: "info.ejs", userid:req.session.userid});
});
```

## • 블로그 웹사이트 만들기

### • 테이블 생성 및 데이터 입력

#### posts 테이블 생성

```
create table posts(  
  id int auto_increment primary key,  
  title varchar(1000) not null,  
  content text,  
  created_at datetime default now()  
);
```

#### posts 테이블 샘플 데이터 입력

```
insert into posts(title, content) values('부트스트랩 시작하기 및 사용 방법','웹 프론트 개발에서의 구조를 미리 만들어준 프레임워크입니다.');
```

```
insert into posts(title, content) values('프론트엔드 CSS 프로그래밍 기초','CSS 라이브러리인 Bootstrap의 기본 구조와 사용법을 배웁니다.');
```

```
insert into posts(title, content) values('웹페이지를 제작할 때 무엇을 도와준다는 것일까?','코딩의 양을 줄여준다.');
```

```
insert into posts(title, content) values('부트스트랩(Bootstrap)이란?','부트스트랩은 HTML, CSS, JS 도우미다.');
```

```
insert into posts(title, content) values('부트스트랩의 개념 및 사용법','구현 방식들이 혼잡해져서 이를 해결하기 위해 만들었다고 합니다.');
```

### • 프로젝트 생성하기

```
C:\Wproject>express -e blog
```

### • 기본 라이브러리를 설치한다.

```
C:\Wproject\Wblog>npm install
```

### • async/await을 사용하기 위해서 mysql2를 설치한다.

```
C:\Wproject\Wblog>npm install mysql2
```

### • 데이터베이스 연결을 위한 파일을 생성한다.

#### db.js

```
var mysql = require('mysql2/promise');  
var connection;  
exports.connect=function() {  
  connection=mysql.createPool({  
    connectionLimit:100,  
    host:'localhost',  
    user:'root',  
    password:'1234',  
    database:'blogdb'  
  });  
}  
  
exports.get=function() {  
  return connection;  
};
```

### • 웹서버 실행 시 데이터베이스를 연결한다.

#### app.js

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

- **포스트 목록 페이지 만들기**

[routes] blog.js

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

//포스트 목록
router.get('/', async function(req, res, next) {
  var sql = 'select *, DATE_FORMAT(created_at,"%Y-%m-%d") date from posts order by created_at';
  var [posts] = await db.get().query(sql);
  res.render('index', {
    title: 'Blog',
    pageName: 'blog/list',
    posts: posts
  });
});

module.exports = router;
```

app.js

```
app.use('/', indexRouter);
app.use('/users', usersRouter);
app.use('/blog', require('./routes/blog'));
...
```

[views] index.ejs

```
<html>
  <head>
    <title><%=title%></title>
    <link rel="stylesheet" href="/stylesheets/style.css"/>
  </head>
  <body>
    <%-include(pageName)%>
  </body>
</html>
```

[views]-[blog] list.ejs

```
<html>
  <title>Blog</title>
  <body>
    <h1>Blog</h1>
    <% for(var p of posts) { %>
      <hr>
      <h2><%=p.title%></h2>
      <h4><%=p.date%></h4>
      <p><%=p.content%></p>
    <% } %>
  </body>
</html>
```

- **포스트 상세 페이지**

[views] index.ejs

```
<h1>Blog</h1>
<% for(var p of posts) { %>
  <hr>
  <a href="/blog/<%=p.id%>"><h2><%=p.title%></h2></a>
  <h4><%=p.date%></h4>
  <p><%=p.content%></p>
<% } %>
```

#### [routes] blog.js

```
...
//포스트 상세페이지
router.get('/:id', async function(req, res) {
  var id = req.params.id;
  var sql = 'select *, DATE_FORMAT(created_at,"%Y-%m-%d %H %T") date from posts where id=?';
  var [post] = await db.get().query(sql, [id]);
  res.render('index', {
    title: 'Blog',
    pageName: 'blog/read',
    post: post
  });
});
...

```

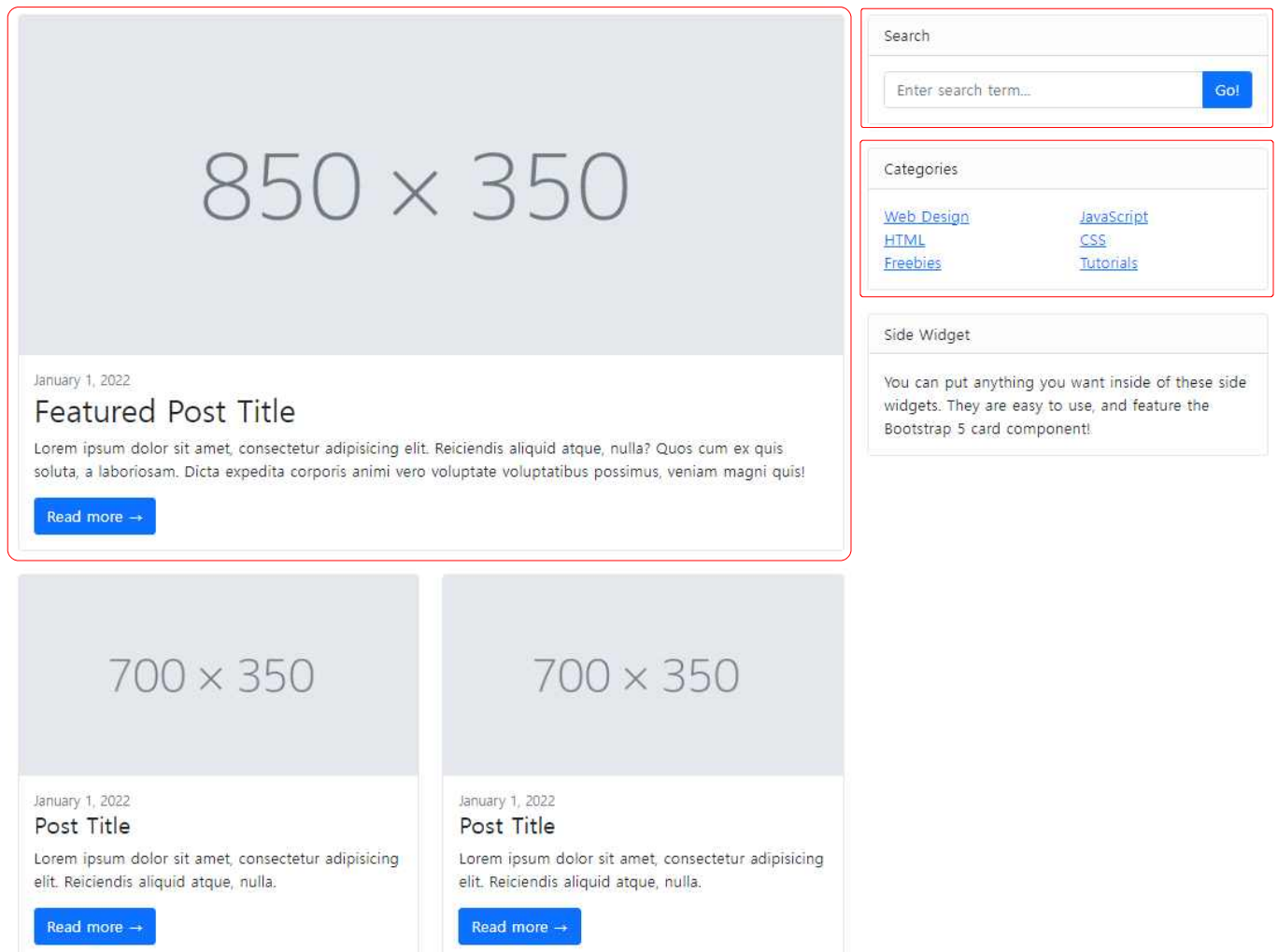
#### [views]-[blog] read.ejs

```
<h1><%=post.title%></h1>
<h4><%=post.date%></h4>
<p><%=post.content%></p>
<hr/>

```

#### • 포스트 목록 페이지에 부트스트랩 적용하기

- 1) Start Bootstrap(startbootstrap.com)으로 이동한다.
- 2) [Templates]-[Blog]-[Blog Home]으로 들어가서 Free Download 버튼을 클릭해 압축 파일을 내려 받는다.
- 3) 압축을 풀고 안에 있는 index.html을 실행하고 Ctrl + U를 누른 후 소스코드를 복사해 사용한다.



[public]-[stylesheets] style.css

```
@font-face {
  font-family: 'GmarketSansMedium';
  src: url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2001@1.1/GmarketSansMedium.woff') format('woff');
  font-weight: normal;
  font-style: normal;
}

* {
  font-family: 'GmarketSansMedium';
}
```

[views] index.ejs

```
<html>
  <head>
    <title><%=title%></title>
    <link rel="stylesheet" href="/stylesheets/style.css"/>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/css/bootstrap.min.css" rel="stylesheet"/>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.2/dist/js/bootstrap.bundle.min.js"></script>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.8.1/font/bootstrap-icons.css">
  </head>
  <body>
    <div class="container">
      <div class="row">
        <!-- Blog post -->
        <div class="col-lg-8">
          <%-include(pageName)%>
        </div>
        <!-- Search widget -->
        <div class="col-lg-4 mt-5">
          <div class="card mb-4">
            <div class="card-header">Search</div>
            <div class="card-body">
              <div class="input-group">
                <input class="form-control" type="text" placeholder="Enter search term..."
                  aria-label="Enter search term..." aria-describedby="button-search"/>
                <button class="btn btn-primary" id="button-search" type="button">Go!</button>
              </div>
            </div>
          </div>
          <!-- Categories widget -->
          <div class="card mb-4">
            <div class="card-header">Categories</div>
            <div class="card-body">
              <div class="row">
                <div class="col-sm-6">
                  <ul class="list-unstyled mb-0">
                    <li><a href="#">Web Design</a></li>
                    <li><a href="#">HTML</a></li>
                    <li><a href="#">Freebies</a></li>
                  </ul>
                </div>
                <div class="col-sm-6">
                  <ul class="list-unstyled mb-0">
                    <li><a href="#">JavaScript</a></li>
                    <li><a href="#">CSS</a></li>
                    <li><a href="#">Tutorials</a></li>
                  </ul>
                </div>
              </div>
            </div>
          </div>
        </div>
      </div>
    </div>
  </body>
</html>
```



[views]-[blog] list.ejs

```
<h1>Blog</h1>
<% for(var post of posts) { %>
  <div class="card mb-4">
    <a href="#"></a>
    <div class="card-body">
      <div class="card-title h4"><%=post.title%></div>
      <p class="card-text"><%=post.content%></p>
      <a class="btn btn-primary" href="/blog/<%=post.id%>">Read more →</a>
    </div>
    <div class="card-footer text-muted">Posted on <%=post.date%> by <a href="#">작성작명</a></div>
  </div>
<% } %>
```

- 포스트 상세 페이지에 부트스트랩 적용하기

[views]-[blog] read.ejs

```
<h1 class="mt-4"><%=post.title%></h1>
<p class="lead">by <a href="#">작성작명</a></p>
<hr>
<p><%=post.date%></p>
<hr>

<hr>
<p><%=post.content%></p>
<hr>
```

- 네비게이션 바와 푸터 모듈화하기

[views] header.ejs

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container">
    <a class="navbar-brand" href="#">Start Bootstrap</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent"
      aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav ms-auto mb-2 mb-lg-0">
        <li class="nav-item"><a class="nav-link" href="#">Home</a></li>
        <li class="nav-item"><a class="nav-link" href="#">About</a></li>
        <li class="nav-item"><a class="nav-link active" aria-current="page" href="/blog">Blog</a></li>
      </ul>
    </div>
  </div>
</nav>
```

[views] footer.ejs

```
<footer class="py-5 bg-dark">
  <div class="container">
    <p class="m-0 text-center text-white">Copyright 2020. 인천일보아카데미 All rights reserved.</p>
  </div>
</footer>
```

[views] index.ejs

```
<body>
  <%-include("header.ejs")%>
  <div class="container">
    ...
  </div>
  <%-include("footer.ejs")%>
</body>
```

- 작성자 추가하기

#### users 테이블 생성

```
create table users(  
    id varchar(20) not null primary key,  
    pass varchar(20) not null,  
    name varchar(20) not null  
);
```

#### users 샘플 데이터 입력

```
create table users(id, pass, name) values('blue', 'pass', '홍길동');  
create table users(id, pass, name) values('red', 'pass', '심청이');
```

#### posts 테이블에 author 컬럼 추가

```
alter table posts add column author varchar(20); /*작성자 컬럼 추가*/  
alter table posts add foreign key(author) references users(id); /*foreign key추가*/  
update posts set author='blue' where id is not null; /*작성자수정*/
```

#### [views]-[blog] list.ejs

```
...  
<div class="card-footer text-muted">  
    Posted on <%=post.date%> by <a href="#"><%=post.author%></a>  
</div>  
...
```

#### [views]-[blog] read.ejs

```
...  
<p class="lead">  
    by <a href="#"><%=post.author%></a>  
</p>  
...
```

- 카테고리 기능 구현하기

#### categories 테이블 생성

```
create table categories(  
    id int auto_increment primary key,  
    name varchar(100) not null  
);
```

#### categories 샘플 데이터 입력

```
insert into categories(name) values('웹디자인');  
insert into categories(name) values('자바스트립트');  
insert into categories(name) values('리엑트');  
insert into categories(name) values('HTML');
```

#### posts 테이블에 category 컬럼 추가

```
alter table posts add column category varchar(100); /*카테고리 컬럼 추가*/  
alter table posts add foreign key(category) references categories(id); /*foreign key추가*/  
update posts set category=1 where id in(1,3); /*카테고리 수정*/
```

- 포스트 목록 페이지에 카테고리 추가하기

posts view 생성

```
create view view_posts as
select p.*, date_format(created_at, '%Y-%m-%d %T') date,
       case when category is null then '미분류'
       else c.name
       end as cname
from posts p left join categories c
on p.category=c.id;
```

[routes] blog.js

```
//포스트 목록
router.get('/', async function(req, res, next) {
  var category = req.query.category;
  var sql = 'select * from view_posts'; //포스트 목록
  var condition = ''; //조건절
  if(category) {
    condition = ` where cname='${category}'`;
  }
  sql += condition + ' order by created_at desc';
  var [posts] = await db.get().query(sql);

  var sql = 'select count(*) cnt, category, cname from view_posts'; //카테고리 목록
  sql += ' group by category order by id desc';
  var [categories] = await db.get().query(sql);

  res.render('index', {
    title: 'Blog',
    pageName: 'blog/list',
    posts: posts,
    categories: categories
  });
});
```

[views] index.ejs

```
...
<!-- Categories widget-->
<div class="card mb-4">
  <div class="card-header">Categories</div>
  <div class="card-body">
    <div class="row">
      <div class="col-sm-6">
        <ul class="list-unstyled mb-0">
          <% for(var c of categories) { %>
            <li><a href="/blog?category=<%=c.cname%>"><%=c.cname%>(<%=c.cnt%>)</a></li>
          <% } %>
        </ul>
      </div>
    </div>
  </div>
</div>
```

[views]-[blog] list.ejs

```
<h1>
  Blog
  <% if(category) { %>
    <span class="badge text-bg-secondary"><%=category%></span>
  <% } %>
</h1>
<div class="card-body">
  <span class="badge text-bg-secondary float-end"><%=post.cname%></span>
  <div class="card-title h4"><%=post.title%></div>
  ...
</div>
...
```

[routes] blog.js

```
...
//포스트 상세페이지
router.get('/:id', async function(req, res) {
  var id = req.params.id;
  var sql = 'select * from view_posts where id=?'; //포스트상세정보
  var [post] = await db.get().query(sql, [id]);

  var sql = 'select count(*) cnt, category, cname from view_posts '; //카테고리 목록
  sql += ' group by category order by id desc';
  var [categories] = await db.get().query(sql);

  res.render('index', {
    title: 'Blog',
    pageName: 'blog/read',
    post: post,
    categories: categories
  });
});
...
```

[views]-[blog] read.ejs

```
...
<div>
  <span class="badge text-bg-secondary float-end mt-2">
    <%=post.cname%>
  </span>
  <h1 class="mt-4"><%=post.title%></h1>
</div>
...
```

## • 태그 페이지 만들기

tags 테이블 생성

```
create table tags(
  postid int not null,
  name varchar(100) not null,
  primary key(postid, name),
  foreign key(postid) references posts(id)
);
```

tags 테이블 샘플 데이터 입력

```
insert into tags values(1, '장고');
insert into tags values(1, '파이썬');
insert into tags values(1, '프로그래밍');
insert into tags values(2, '리액트');
insert into tags values(2, '프론트엔드');
insert into tags values(2, '노드');
insert into tags values(2, '리눅스');
insert into tags values(3, '장고');
insert into tags values(3, '파이썬');
insert into tags values(3, '리액트');
insert into tags values(3, '프론트엔드');
insert into tags values(4, '리눅스');
insert into tags values(4, '장고');
insert into tags values(4, '파이썬');
insert into tags values(4, '리액트');
insert into tags values(5, '프론트엔드');
insert into tags values(5, '리눅스');
insert into tags values(5, '장고');
insert into tags values(5, '파이썬');
```

## [routes] blog.js

```
//포스트 목록
router.get('/', async function(req, res, next) {
  var category = req.query.category;
  var tag = req.query.tag;
  var sql = 'select * from view_posts'; //포스트 목록
  var condition = ''; //조건절
  if(category) {
    condition = ` where cname='${category}'`;
  } else if(tag) {
    condition = ` where id in (select postid from tags where name='${tag}'))`;
  }
  sql += condition + ' order by created_at desc';
  var [posts] = await db.get().query(sql);

  var rows = [];
  for(var row of posts) {
    var sql='select * from tags where postid=?'; //태그 목록
    var [tags]=await db.get().query(sql, [row.id]);
    rows.push({ post: row, tags: tags });
  }

  var sql = `select count(*) cnt, name from tags where name='${tag}'`; //태그 포스트 수
  var [tag] = await db.get().query(sql);

  var sql='select count(*) cnt, category,cname from view_posts '; //카테고리 목록
  sql+='group by category order by id desc';
  var [categories] = await db.get().query(sql);

  res.render('index', {
    title: 'Blog',
    pageName: 'blog/list',
    posts: rows,
    categories: categories,
    category: category,
    tag: tag[0]
  });
});
```

## [views]-[blog] list.ejs

```
<h1>
  Blog
  <% if(category) { %>
    <span class="badge text-bg-secondary"><%=category%></span>
  <% }else if(tag.name) { %>
    <span class="badge bg-light text-dark">
      <i class="bi bi-tags-fill"></i>
      <%=tag.name%> (<%=tag.cnt%>)
    </span>
  <% } %>
</h1>
...
<div class="card-body">
...
  <p class="card-text"><%=row.post.content%></p>
  <% if(row.tags.length > 0) { %>
    <div class="mb-4">
      <i class="bi bi-tags-fill"></i>
      <% for(var tag of row.tags) { %>
        <a href="/blog?tag=<%=tag.name%>">
          <span class="badge bg-dark"><%=tag.name%></span>
        </a>
      <% } %>
    </div>
  <% } %>
  <a class="btn btn-primary" href="/blog/<%=row.post.id%>">Read more →</a>
</div>
...
```

[routes] blog.js

```
//포스트 상세페이지
router.get('/:id', async function(req, res) {
  var id = req.params.id;
  var sql = 'select * from view_posts where id=?'; //포스트상세정보
  var [post] = await db.get().query(sql, [id]);

  var sql = 'select count(*) cnt, category,cname from view_posts '; //카테고리 목록
  sql += 'group by category order by id desc';
  var [categories] = await db.get().query(sql);

  var sql = 'select name from tags where postid=' + id;
  var [tags] = await db.get().query(sql);

  res.render('index', {
    title: 'Blog',
    pageName: 'blog/read',
    post: post[0],
    categories: categories,
    tags: tags
  });
});
```

[views]-[blog] read.ejs

```
...
<p><%=post.content%></p>
<hr>
<% if(tags.length > 0) {%>
  <div class="mb-4">
    <i class="bi bi-tags-fill"></i>
    <% for(var tag of tags) {%>
      <a href="/blog?tag=<%=tag.name%>">
        <span class="badge bg-dark"><%=tag.name%></span>
      </a>
    <% } %>
  </div>
<% } %>
```

- 로그인/로그아웃 페이지 만들기

npm install express-session

app.js 추가

```
var session = require('express-session');
...
var app = express();
app.use(session({
  secret: 'mykey',
  resave: false, //세션 데이터가 바뀌기 전에는 세션 저장소에 값을 저장하지 않음
  saveUninitialized: true //세션이 필요한 경우에만 구동
}));
...

```

[routes] users.js

```
//Login
router.get('/login', function(req, res) {
  res.render('index', {
    title: '로그인',
    pageName: 'users/login'
  });
});
```

[views]-[users] login.ejs

```
<div class="card" style="width: 25rem; margin:0px auto; margin-top:5rem; margin-bottom:15rem;">
  <div class="card-header p-3">
    <h3><i class="bi bi-box-arrow-in-right"></i> Log In</h3>
  </div>
  <div class="card-body">
    <form name="frm">
      <input name="id" class="form-control p-2 my-3">
      <input name="pass" type="password" class="form-control p-2 my-3">
      <button class="btn btn-primary p-2 my-3" style="width:100%;">Log In</button>
    </form>
  </div>
</div>

<script>
  $(frm).on("submit", function(e) {
    e.preventDefault();
    var id=$(frm.id).val();
    var pass=$(frm.pass).val();
    if(id=="" || pass=="") {
      alert("아이디 또는 비밀번호를 입력하세요!");
      return;
    }

    $.ajax({
      type: "post",
      url: "/users/login",
      data: { id: id, pass: pass },
      success: function(data) {
        //console.log('.....', data);
        if(data=="0") {
          alert("아이디가 존재하지 않습니다!");
        }else if(data=="2") {
          alert("비밀번호가 일치하지 않습니다!");
        }else if(data=="1") {
          location.href="/blog";
        }
      }
    });
  });
</script>
```

[routes] user.js

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

//Login 체크
router.post('/login', async function(req, res) {
  var id=req.body.id;
  var pass=req.body.pass;

  var sql = 'select * from users where id=?';
  var [check] = await db.get().query(sql, [id]);
  var result = 0;
  if(check[0] != null) {
    if(check[0].pass == pass) {
      result = 1;
      req.session.user=check[0].id; //로그인 성공 시 세션(user변수)에 아이디 저장
    }else {
      result = 2;
    }
  }
  res.status(200).json(result);
});

module.exports = router;
```

#### [routes] users.js

```
...

//Login
router.get('/login', function(req, res) {
  res.render('index', {
    title: '로그인',
    pageName: 'users/login',
    user: req.session.user
  });
});

//Logout
router.get('/logout', function(req, res) {
  req.session.destroy();
  res.redirect('/blog');
});
...
```

#### [routes] blog.js

```
...
//포스트 목록
router.get('/', async function(req, res, next) {
  ...
  res.render('index', {
    title: 'Blog',
    pageName: 'blog/list',
    posts: rows,
    categories: categories,
    category: category,
    tag: tag[0],
    user: req.session.user
  });
});

//포스트 상세페이지
router.get('/:id', async function(req, res) {
  ...
  res.render('index', {
    title: 'Blog',
    pageName: 'blog/read',
    post: post[0],
    categories: categories,
    tags: tags,
    user: req.session.user
  });
});
});
```

#### [views] header.ejs

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark">
  <div class="container">
    ...
    <li class="nav-item">
      <a class="nav-link <%=pageName=='blog/list' ? 'active':''%>" aria-current="page" href="/blog">Blog</a>
    </li>
    <% if(user) {%>
      <li class="nav-item">
        <a class="nav-link <%=pageName=='users/login' ? 'active':''%>" href="/users/logout">Logout (<%=user%>)</a>
      </li>
    <% }else { %>
      <li class="nav-item">
        <a class="nav-link <%=pageName=='users/login' ? 'active':''%>" href="/users/login">Login</a>
      </li>
    <% } %>
    ...
  </div>
</nav>
```



- 포스트 등록 페이지

[views]-[blog] list.ejs

```
<h1>
  Blog
  ...
  <% if(user) {%>
    <a class="btn btn-primary float-end" href="/blog/insert">게시글 등록</a>
  <% } %>
</h1>
```

[routes] blog.js

```
//포스트 등록 페이지
router.get('/insert', async function(req, res) {
  var sql = 'select * from categories';
  var [categories] = await db.get().query(sql);
  res.render('index', {
    title: '포스트 등록',
    pageName: 'blog/insert',
    categories: categories,
    user: req.session.user
  });
});
```

[views] index.ejs

```
<script src="https://code.jquery.com/jquery-3.1.1.min.js"></script>
...
<div class="col-lg-8">
  <%-include(pageName)%>
</div>
<% if(pageName=='blog/list' || pageName=='blog/read') { %> <!--블로그 목록 페이지와 블로그 상세페이지 인 경우에만 보인다.-->
<div class="col-lg-4 mt-5">
  <!--Search widget-->
  <div class="card mb-4">....</div>
  <!--Categories widget-->
  <div class="card mb-4">...</div>
</div>
<% } %>
```

[views]-[blog] insert.ejs

```
<div class="card p-5" style="width:50rem; margin:0px auto; margin-bottom:10rem;">
  <h1>게시글 등록</h1>
  <form name="frm" method="post" enctype="multipart/form-data">
    <input name="title" placeholder="제목을 입력하세요." class="form-control p-2 my-3">
    <textarea name="content" placeholder="내용을 입력하세요." rows="10" class="form-control p-2 my-3"></textarea>
    <input name="file" type="file" class="form-control my-3">
    <select name="category" class="form-select my-3">
      <% for(var category of categories) { %>
        <option value="<%=category.id%>"><%=category.name%></option>
      <% } %>
    </select>
    <input name="tags" placeholder="해시태그를 입력하세요." class="form-control p-2 my-2">
    <button class="btn btn-primary float-end">게시글 등록</button>
  </form>
</div>
<script>
  $(frm).on("submit", function(e) {
    e.preventDefault();
    var title=$(frm.title).val();
    if(title == "" || content == "") {
      alert("제목이나 내용을 입력하세요!");
      return;
    }
    frm.submit();
  });
</script>
```

- dummy 이미지 다운로드

```
https://picsum.photos/seed/1/800/200
```

```
npm install multer
```

```
[routes] blog.js
```

```
//파일 업로드 설정
var multer = require('multer');
var upload = multer({
  storage: multer.diskStorage({
    destination: (req, file, done)=> {
      done(null, './public/images');
    },
    filename: (req, file, done)=> {
      var exe=file.originalname.substring(file.originalname.lastIndexOf('.'));
      var fileName=Date.now() + exe;
      done(null, fileName)
    }
  })
});

//포스트 등록
router.post('/insert', upload.single('file'), async function(req, res) {
  var title=req.body.title;
  var content=req.body.content;
  var author=req.session.user;
  var category=req.body.category;
  var image=!req.file ? '':req.file.filename;

  var sql = 'insert into posts(title,content,author,category,image) values(?,?,?,?,?)';
  await db.get().query(sql, [title,content,author,category,image]); //포스트 정보 등록

  var [last] = await db.get().query('select distinct last_insert_id() postid from posts');
  var postid = last[0].postid; //마지막 insert 아이디

  if(req.body.tags) { //태그등록
    var tags = req.body.tags.split(',');
    for(var i=0; i<tags.length; i++) {
      var sql = 'insert into tags(postid,name) values(?,?)';
      await db.get().query(sql, [postid, tags[i]]);
    }
  }
  res.redirect('/blog');
});
```

```
[views]-[blog] list.ejs
```

```
<div class="card mb-4">
  <a href="#">
    <% if(row.post.image) {%>
      
    <% }else {%>
      
    <% } %>
  </a>
  ...
</div>
```

```
[views]-[blog] read.ejs
```

```
<p><%=post.date%></p>
<hr>
<% if(post.image) {%>
  
<% }else {%>
  
<% } %>
<hr>
```

- 포스트 수정 페이지

[routes]-[blog] update.ejs

```
...
//포스트 수정 페이지
router.get('/update/:id', async function(req, res) {
  var id = req.params.id;
  var sql = 'select * from view_posts where id=?'; //포스트상세정보
  var [post] = await db.get().query(sql, [id]);

  var sql = 'select * from categories';
  var [categories] = await db.get().query(sql);

  var sql = 'select name from tags where postid=' + id;
  var [tags] = await db.get().query(sql);
  var strTags='';
  for(var i=0; i<tags.length; i++) {
    strTags += tags[i].name;
    if(i != tags.length-1) strTags += ',';
  }

  res.render('index', {
    title: '포스트 수정',
    pageName: 'blog/update',
    categories: categories,
    user: req.session.user,
    post: post[0],
    tags: strTags
  });
});
...
```

[views]-[blog] update.ejs

```
<div class="card p-5" style="width:50rem; margin:0px auto; margin-bottom:10rem;">
  <h1>게시글 수정</h1>
  <form name="frm" method="post" action="/blog/update" enctype="multipart/form-data">
    <input name="id" value="<%=post.id%>" type="hidden">
    <input name="image" value="<%=post.image%>" type="hidden">
    <input name="title" value="<%=post.title%>" class="form-control p-2 my-3">
    <textarea name="content" rows="10" class="form-control p-2 my-3"><%=post.content%></textarea>
    <% if(post.image) { %>
      
    <% } %>
    <input name="file" type="file" class="form-control my-3">
    <select name="category" class="form-select my-3">
      <% for(var category of categories) {%>
        <option value="<%=category.id%>" <%=post.category==category.id? 'selected':''%>>
          <%=category.name%>
        </option>
      <% } %>
    </select>
    <input name="tags" value="<%=tags%>" placeholder="해시태그를 입력하세요." class="form-control p-2 my-2">
    <button class="btn btn-primary float-end">게시글수정</button>
  </form>
</div>

<script>
  $(frm).on("submit", function(e) {
    e.preventDefault();
    var title=$(frm.title).val();
    var content=$(frm.content).val();
    if(title == "" || content == "") {
      alert("제목이나 내용을 입력하세요!");
      return;
    }
    if(!confirm("게시글 내용을 수정하실래요?")) return;
    frm.submit();
  });
</script>
```

#### [routes] blog.js

```
//포스트 수정
router.post('/update', upload.single('file'), async function(req, res) {
  var id=req.body.id;
  var title=req.body.title;
  var content=req.body.content;
  var author=req.session.user;
  var category=req.body.category;
  var image=!req.file ? req.body.image:req.file.filename;

  //포스트 수정
  var sql='update posts set title=?,content=?,author=?,category=?,image=? where id=?';
  await db.get().query(sql, [title,content,author,category,image,id]);

  //태그들 수정
  var sql='delete from tags where postid=?' //기존 태그들 삭제
  await db.get().query(sql, id);
  if(req.body.tags) { //새로운 태그들 입력
    var tags = req.body.tags.split(',');
    for(var i=0; i<tags.length; i++) {
      var sql='select count(*) cnt from tags where postid=? and name=?'; //중복 태그 체크
      var [check]=await db.get().query(sql, [id, tags[i]]);
      if(check[0].cnt == 0) {
        var sql = 'insert into tags(postid,name) values(?, ?)';
        await db.get().query(sql, [id, tags[i]]);
      }
    }
  }

  //이미지가 바뀌면 예전 이미지 삭제
  if(req.file && req.body.image != '') {
    fs.unlink('./public/images/' + req.body.image, function(err) {
      if(err) console.log('에러.....', err);
    });
  }
  res.redirect('/blog');
});
...

```

#### • 이미지 미리보기

#### [views]-[blog] update.ejs

```
...
<% if(post.image) {%>
  
<% } else {%>
  
<% } %>
<input name="file" type="file" class="form-control my-3">
...
<script>
  $(frm.file).on("change", function(e) {
    $("#image").attr("src", URL.createObjectURL(e.target.files[0]));
  });
</script>

```

#### [views]-[blog] insert.ejs

```
...

<input name="file" type="file" class="form-control my-3">
...
<script>
  $(frm.file).on("change", function(e) {
    $("#image").attr("src", URL.createObjectURL(e.target.files[0]));
  });
</script>

```

- 목록 페이지 페이지징 기능

[routes] blog.js

```
//포스트 목록
router.get('/', async function(req, res, next) {
  var page=!req.query.page ? 1: parseInt(req.query.page);
  var category = req.query.category;
  var tag = req.query.tag;

  var sql = 'select * from view_posts'; //포스트 목록
  var condition = ''; //조건절
  if(category) {
    condition = ` where cname='${category}'`;
  }else if(tag) {
    condition = ` where id in (select postid from tags where name='${tag}'))`;
  }
  sql += condition + ' order by created_at desc limit ?, 3';
  var [posts] = await db.get().query(sql, [(page-1) * 3]);

  var sql = 'select count(*) cnt from view_posts ' + condition; //포스트 수
  var [result] = await db.get().query(sql);
  var total = result[0].cnt;

  ...

  res.render('index', {
    title: 'Blog',
    pageName: 'blog/list',
    posts: rows,
    categories: categories ,
    category: category,
    tag: tags[0],
    user: req.session.user,
    page: page,
    last: Math.ceil(total/3)
  });
});
...

```

[views]-[blog] list.ejs

```
...
<% for(var row of posts) { %>
  <p class="card-text"><%=row.post.content%></p>
  <% if(row.tags.length > 0) {%>
    <div class="mb-4">
      <i class="bi bi-tags-fill"></i>
      <% for(let tag of row.tags) {%>
        <a href="/blog?tag=<%=tag.name%>&page=1">
          <span class="badge bg-dark"><%=tag.name%></span>
        </a>
      <% } %>
    </div>
  <% } %>
  ...
<% } %>

<!--Pagination-->
<ul class="pagination justify-content-center mb-4">
  <li class="page-item <%=page==1 ? 'disabled' : ''%>">
    <a class="page-link" href="/blog?category=<%=category%>&tag=<%=tag.name%>&page=<%=page-1%>">
      &larr; Older
    </a>
  </li>
  <li class="page-item <%=page==last ? 'disabled' : ''%>">
    <a class="page-link" href="/blog?category=<%=category%>&tag=<%=tag.name%>&page=<%=page+1%>">
      Newer &rarr;
    </a>
  </li>
</ul>

```

- 목록 페이지 검색 기능

[views] index.ejs

```
<div class="card mb-4">
  <div class="card-header">Search</div>
  <form name="frm">
    <div class="card-body">
      <div class="input-group">
        <input name="search" value="<%=search%>" class="form-control" type="text"
          aria-label="Enter search term..." aria-describedby="button-search"/>
        <button class="btn btn-primary" id="button-search" type="submit">Go!</button>
      </div>
    </div>
  </form>
</div>
</div>
...
<script>
  $(frm).on("submit", function(e) {
    e.preventDefault();
    var search=$(frm.search).val();
    frm.action="/blog?page=1&search=${search}";
    frm.submit();
  })
</script>
```

[routers] blog.js

```
//포스트 목록
router.get('/', async function(req, res, next) {
  var page=!req.query.page ? 1: parseInt(req.query.page);
  var category = req.query.category;
  var tag = req.query.tag;
  var search = req.query.search;

  var sql = 'select * from view_posts'; //포스트 목록
  var condition = ''; //조건절
  if(category) {
    condition = ` where cname='${category}'`;
  }else if(tag) {
    condition = ` where id in (select postid from tags where name='${tag}'))`;
  } else if(search) {
    condition = ` where title like '%${search}%' or content like '%${search}%'`;
  }
  sql += condition + ' order by created_at desc limit ?, 3';
  var [posts] = await db.get().query(sql, [(page-1) * 3]);
  ...
  res.render('index', {
    ...
    search: search,
    total: total
  });
});
```

[views]-[blog] list.ejs

```
<!--Pagination-->
<ul class="pagination justify-content-center mb-4">
  <li class="page-item <%=page==1 ? 'disabled' : ''%>">
    <a href="/blog?category=<%=category%>&tag=<%=tag.name%>&page=<%=page-1%>&search=<%=search%>">
      &larr; Older
    </a>
  </li>
  <li class="page-item <%=page==last ? 'disabled' : ''%>">
    <a href="/blog?category=<%=category%>&tag=<%=tag.name%>&page=<%=page+1%>&search=<%=search%>">
      Newer &rarr;
    </a>
  </li>
</ul>
```

[views] right.ejs 생성

```
<!--Search widget-->
<div class="card mb-4">
  ...
</div>
<!--Categories widget-->
<div class="card mb-4">
  ...
</div>
```

[views] index.ejs

```
<body>
  <%-include("header.ejs")%>
  <div class="container">
    <div class="row mt-4">
      <% if(pageName=='blog/list' || pageName=='blog/read') { %>
        <div class="col-lg-8">
          <%-include(pageName)%>
        </div>
        <div class="col-lg-4 mt-5">
          <%-include("right.ejs") %>
        </div>
      <% } else { %>
        <div class="col-lg-12">
          <%-include(pageName)%>
        </div>
      <% } %>
    </div>
  </div>
  <%-include("footer.ejs")%>
</body>
```

## • 실행결과

Start Bootstrap

Home About Blog Logout (red)

Blog

📁 프로그램 (4)

게시글 등록

800 × 200

미분류

웹페이지를 제작할 때 무엇을 도와준다는 것일까?  
코딩의 양을 줄여준다. 즉, 웹페이지 제작 시 가장 많이 사용되는 CSS, JS 파일을 미리 만들어 놓고 다운로드 또는 CDN 방식으로 링크해서 사용하기만 하면된다.

📁 노트 📁 프로그램

Read more →

Posted on 2022-11-07 13:07:48 by red

← Older Newer →

Search

Enter search term... Go!

Categories

[CSS\(2\)](#)  
[미분류\(2\)](#)  
[Freebies\(1\)](#)  
[Html\(1\)](#)  
[자바스크립트\(1\)](#)  
[웹디자인\(5\)](#)

Copyright 2020. 인천일보아카데미 All rights reserved.

- 댓글 기능 구현하기

Leave a Comment

Submit

red

2022-11-11 08:40:41

조상 노드(ancestor node)란 부모 노드를 포함해 계층적으로 현재 노드보다 상위에 존재하는 모든 노드를 가리킵니다.

수정

red

2022-11-11 08:29:30

HTML 문서의 정보는 노드 트리(node tree)라고 불리는 계층적 구조에 저장됩니다. 이러한 노드 트리는 노드들의 집합이며, 노드 간의 관계를 보여줍니다.

수정

blue

2022-11-11 08:29:30

노드 트리는 최상위 레벨인 루트 노드(root node)로부터 시작하여, 가장 낮은 레벨인 텍스트 노드까지 뿔어 내려갑니다. 코드와 주석을 보고 충분히 알 수 있으니 필요한 부분을 사용하시면 될 것 같습니다.

red

자바스크립트에서는 HTML DOM을 이용하여 노드 트리에 포함된 모든 노드에 접근할 수 있습니다.

취소 저장

#### comments 테이블 생성

```
create table comments(
  id int auto_increment primary key,
  postid int not null,
  author varchar(20) not null,
  content text,
  created_at datetime default now(),
  modified_at datetime default now(),
  foreign key(postid) references posts(id),
  foreign key(author) references users(id)
);
```

#### comments 샘플 데이터 입력

```
insert into comments(postid, author, content)
values(12, 'red', 'HTML 문서의 정보는 노드 트리(node tree)라고 불리는 계층적 구조에 저장됩니다.');
```

```
insert into comments(postid, author, content)
values(12, 'blue', '노드 트리는 최상위 레벨인 루트 노드(root node)로부터 시작하여, 가장 낮은 레벨인 텍스트 노드까지 뿔어 내려갑니다.');
```

```
insert into comments(postid, author, content)
values(12, 'red', '자바스크립트에서는 HTML DOM을 이용하여 노드 트리에 포함된 모든 노드에 접근할 수 있습니다.');
```

```
insert into comments(postid, author, content)
values(12, 'blue', '루트 노드를 제외한 모든 노드는 단 하나의 부모 노드(parent node)만을 가집니다.');
```

```
insert into comments(postid, author, content)
values(12, 'red', '자손 노드(descendant node)란 자식 노드를 포함해 계층적으로 현재 노드보다 하위에 존재하는 모든 노드를 가리킵니다.');
```

```
insert into comments(postid, author, content)
values(12, 'red', '조상 노드(ancestor node)란 부모 노드를 포함해 계층적으로 현재 노드보다 상위에 존재하는 모든 노드를 가리킵니다.');
```



- 포스트 상세 페이지에 작성한 댓글 나타내기

[routes] comments.js

```
...
router.get('/:postid', async function(req, res) {
  var postid = req.params.postid;
  var sql = 'select *, date_format(created_at, "%Y-%m-%d %T") as cdate,';
  sql += ' date_format(modified_at, "%Y-%m-%d %T") as mdate from comments';
  sql += ' where postid=? order by created_at desc';
  var [comments] = await db.get().query(sql, [postid]);
  res.send(comments);
});

module.exports = router;
```

app.js

```
...
app.use('/comments', require('./routes/comments'));
...
```

[views]-[blog] comments.ejs

```
<div class="my-5">
  <div class="card my-4">
    <h5 class="card-header">Leave a Comment</h5>
    <div class="card-body">
      <form>
        <textarea class="form-control" rows="3"></textarea>
        <button type="submit" class="btn btn-primary mt-3">Submit</button>
      </form>
    </div>
  </div>
</div>
<div id="divComments"></div>
<script id="temp" type="text/x-handlebars-template">
  {{#each .}}
    <div class="row mb-4">
      <div class="col-2">
        
      </div>
      <div class="col-10" style="font-size:0.8rem;">
        <div>
          <small>{{author}}</small>
          <small class="text-muted">{{cdate}}</small>
        </div>
        <p>{{content}}</p>
      </div>
    </div>
  {{/each}}
</script>
<script>
  var postid="<%=post.id%>";

  getComments();
  function getComments() {
    $.ajax({
      type: "get",
      url: "/comments/" + postid,
      dataType: "json",
      success: function(data) {
        var temp = Handlebars.compile($("#temp").html());
        $("#divComments").html(temp(data));
      }
    });
  }
</script>
```

```
[views]-[blog] read.ejs
```

```
...
<% if(user==post.author) { %>
  <a class="btn btn-primary float-end" href="/blog/update/<%=post.id%>">게시글 수정</a>
  <br>
<% } %>

<%-include("comments.ejs")%>
```

- 로그인 상태에 따라 댓글 입력란 또는 로그인 버튼 나타내기

```
[views]-[users] login.js
```

```
...
<div class="card-body">
  <% if(user) { %>
    <form>
      <textarea class="form-control" rows="3"></textarea>
      <button type="submit" class="btn btn-primary mt-3">Submit</button>
    </form>
  <% }else { %>
    <a role="button" class="btn btn-outline-dark btn-sm" style="width:100%;" href="/users/login?target=/blog/<%=post.id%>">
      Log in and Leave a comment
    </a>
  <% } %>
</div>
...
```

```
[routes] users.js
```

```
...
//Login
router.get('/login', function(req, res) {
  var target=req.query.target;
  res.render('index', {
    title: '로그인',
    pageName: 'users/login',
    user: req.session.user,
    target: target
  });
});
...
```

```
[views]-[users] login.js
```

```
...
<script>
  var target="<%=target%>";
  $(frm).on("submit", function(e) {
    ...
    $.ajax({
      type: "post",
      url: "/users/login",
      data: { id: id, pass: pass },
      success: function(data) {
        if(data=="0") {
          alert("아이디가 존재하지 않습니다!");
        }else if(data=="2") {
          alert("비밀번호가 일치하지 않습니다!");
        }else if(data=="1") {
          if(target) location.href=target;
          else location.href="/blog";
        }
      }
    });
  });
</script>
```

- Comment 입력 기능 구현하기

```
[routes] comments.js
```

```
...
router.post('/insert', async function(req, res) {
  var postid = req.body.postid;
  var author = req.body.author;
  var content = req.body.content;
  var sql = 'insert into comments(postid, author, content) values(?,?,?)';
  await db.get().query(sql, [postid, author, content]);
  res.sendStatus(200);
});
```

```
[views]-[blog] comments.ejs
```

```
...
<div class="card-body">
  <% if(user) { %>
    <form name="frmComment">
      <textarea name="content" class="form-control" rows="3"></textarea>
      <button type="submit" class="btn btn-primary mt-3">Submit</button>
    </form>
  <% }else { %>
    <a role="button" class="btn btn-outline-dark btn-sm" style="width:100%;" href="/users/login?target=/blog/<%=post.id%>">
      Log in and Leave a comment
    </a>
  <% } %>
</div>
...
<script>
  var postid="<%=post.id%>";
  var author="<%=user%>";
  getComments();

  $(frmComment).on("submit", function(e) {
    e.preventDefault();
    var content=$(frmComment.content).val();
    if(content == "") {
      alert("댓글 내용을 입력하세요!");
      $(frmComment.content).focus();
      return;
    }
    $.ajax({
      type: "post",
      url: "/comments/insert",
      data: { postid: postid, author: author, content: content },
      success: function(){
        $(frmComment.content).val('');
        getComments();
      }
    });
  });

  function getComments() {
    $.ajax({
      type: "get",
      url: "/comments/" + postid,
      dataType: "json",
      success: function(data) {
        //console.log('.....', data);
        var temp = Handlebars.compile($("#temp").html());
        $("#divComments").html(temp(data));
      }
    });
  }
</script>
```

- 댓글 수정 기능 구현하기

```
[views]-[blog] comments.ejs
```

```
<style>
  .none { display: none; }
</style>
...
<div id="divComments"></div>
<script id="temp" type="text/x-handlebars-template">
  {{#each .}}
    <div class="row mb-4">
      <div class="col-2">
        
      </div>
      <!--Comment Display-->
      <div id="display{{id}}" class="col-10" style="font-size:0.8rem;">
        <div>
          <small>{{author}}</small><small class="text-muted">{{cdate}}</small>
        </div>
        <p>{{content}}</p>
        <div class="{{displayButton author}}">
          <button class="btn btn-sm btn-primary float-end mx-3" onClick="onClickToggle('{{id}}')">수정</button>
          <small class="float-end py-2 text-muted {{displayDate cdate mdate}}">Modified: {{mdate}}</small>
        </div>
      </div>
      <!--Comment Modify Form-->
      <div id="modify{{id}}" class="col-10" style="display:none;">
        <textarea id="content{{id}}" class="form-control mb-2" rows="3" style="font-size:0.8rem;">{{content}}</textarea>
        <button class="btn btn-sm btn-primary float-end" onClick="onClickModify('{{id}}')">저장</button>
        <button class="btn btn-sm btn-primary float-end mx-2" onClick="onClickCancel()">취소</button>
      </div>
    </div>
  {{/each}}
</script>
<script>
  var user="<%=user%>";
  Handlebars.registerHelper("displayButton", function(author) {
    if(user != author) return "none";
  });

  Handlebars.registerHelper("displayDate", function(cdate, mdate) {
    if(cdate == mdate) return "none";
  });
</script>
<script>
  var postId="<%=post.id%>";
  var author="<%=user%>";
  getComments();

  function onClickToggle(id) {
    $('#display${id}`).toggle();
    $('#modify${id}`).toggle();
  }

  function onClickModify(id) {
    if(!confirm("댓글 내용을 수정하실래요?")) return;
    var content=$('#content${id}`).val();
    $.ajax({
      type: "post",
      url: "/comments/update",
      data: { id: id, content: content },
      success: function() {
        getComments();
      }
    });
  }

  function onClickCancel() {
    getComments();
  }
</script>
```

- 댓글 삭제 기능 구현하기

[views]-[blog] comments.ejs

```
...
<div id="divComments"></div>
<!--Modal-->
<div class="modal fade" id="deleteModal" tabindex="-1" aria-labelledby="deleteModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title fs-5" id="deleteModalLabel">Are You Sure?</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <div class="ellipsis">
          <span id="delete_id"></span>:<span id="delete_content"></span>
        </div>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal" aria-label="Close">Calcel</button>
        <button type="button" class="btn btn-primary" onClick="onClickDelete()">Delete</button>
      </div>
    </div>
  </div>
</div>
<script id="temp" type="text/x-handlebars-template">
  {{#each .}}
    <div class="row mb-4">
      <div class="col-2">
        
      </div>
      <!--Comment Dispay-->
      <div id="display{{id}}" class="col-10" style="font-size:0.8rem;">
        <div>
          <small>{{author}}</small><small class="text-muted">{{cdate}}</small>
        </div>
        <p>{{content}}</p>
        <div class="{{displayButton author}}">
          <button class="btn btn-sm btn-primary float-end mx-1" onClick="onClickToggle('{{id}}')">수정</button>
          <button class="btn btn-sm btn-primary float-end mx-1" onClick="onClickModal('{{id}}', '{{content}}')">
            삭제
          </button>
          <small class="float-end py-2 text-muted {{displayDate cdate mdate}}">Modified: {{mdate}}</small>
        </div>
      </div>
    </div>
  ...
</div>
{{/each}}
</script>
<script>
  ....
  function onClickModal(id, content) {
    $("#delete_id").html(id);
    $("#delete_content").html(content);
    $("#deleteModal").modal("show");
  }

  function onClickDelete() {
    var id = $("#delete_id").html();
    $.ajax({
      type: "post",
      url: "/comments/delete",
      data: { id: id },
      success: function() {
        $("#deleteModal").modal("hide");
        getComments();
      }
    });
  }
  ...
</script>
```

[routes] blog.js

```
..
router.post('/delete', async function(req, res) {
  var id = req.body.id;
  var sql = 'delete from comments where id=?';
  await db.get().query(sql, [id]);
  res.sendStatus(200);
});
...
```

[public]-[stylesheets] style.css

```
..
.ellipsis {
  overflow: hidden;
  text-overflow: ellipsis;
  display: -webkit-box;
  -webkit-line-clamp: 1;
  -webkit-box-orient: vertical;
}
```

- 때문에 최신 포스트 나타내기

[routes] index.js

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

//home 페이지
router.get('/', function(req, res, next) {
  res.render('index', {
    title: 'Home',
    pageName: 'home',
    user: req.session.user
  });
});

module.exports = router;
```

[views] home.ejs

```
<style>
  body {
    background: url('/images/amsterdam-sky.jpg') no-repeat center fixed;
    -webkit-background-size: cover;
    -moz-background-size: cover;
    background-size: cover;
    -o-background-size: cover;
  }
</style>
<div class="row justify-content-between" style="margin-top:10rem; margin-bottom:10rem;">
  <div class="col-lg-6 text-light">
    <h1>홍길동의 홈페이지</h1>
    <p>Node.js는 크롬 V8 자바스크립트 엔진으로 만들어진 JavaScript 런타임이다.</p>
  </div>
  <div class="col-lg-6">
    <h2 class="text-light">Blog - Recent Posts</h2>
    <div class="card mt-1">
      <div class="card-body">This is some text within a card body</div>
    </div>
    <div class="card mt-1">
      <div class="card-body">This is some text within a card body</div>
    </div>
  </div>
</div>
```

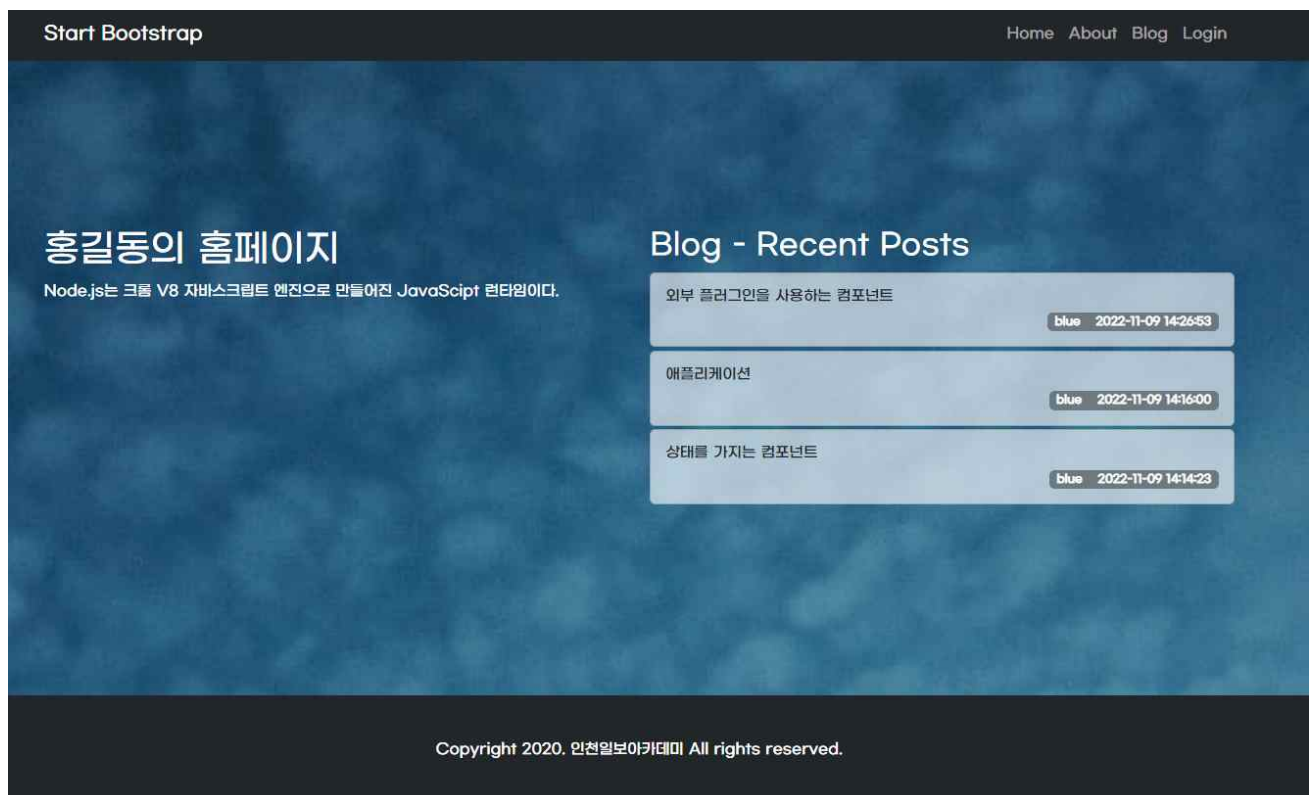
#### [routes] index.js

```
//home 페이지
router.get('/', async function(req, res, next) {
  var sql = 'select * from view_posts order by created_at desc limit 0, 3';
  var [posts] = await db.get().query(sql);
  res.render('index', {
    title: 'Home',
    pageName: 'home',
    user: req.session.user,
    recent_posts: posts
  });
});
```

#### [views] home.ejs

```
<style>
...
.card {
  background-color: rgba(255, 255, 255, 0.6);
}
</style>
<div class="row justify-content-between" style="margin-top:10rem; margin-bottom:10rem;">
...
<div class="col-lg-6">
  <h2 class="text-light">Blog - Recent Posts</h2>
  <% for(var post of recent_posts) { %>
    <div class="card mt-1">
      <div class="card-body">
        <h6><a href="/blog/<%=post.id%>"><%=post.title%></a></h6>
        <span class="badge bg-secondary float-end">
          <%=post.author%> &nbsp;&nbsp;&nbsp;<%=post.date%>
        </span>
      </div>
    </div>
  <% } %>
</div>
</div>
```

#### • 실행결과



## • 도서검색 웹사이트 만들기

[routes] index.js

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

router.get('/', function (req, res, next) {
  res.render('index', { title: 'Home', pageName: 'home.ejs' });
});

module.exports = router;
```

[public]-[stylesheets] style.css

```
@font-face {
  font-family: 'GmarketSansMedium';
  src: url('https://cdn.jsdelivr.net/gh/projectnoonnu/noonfonts_2001@1.1/GmarketSansMedium.woff') format('woff');
  font-weight: normal;
  font-style: normal;
}

* {
  font-family: 'GmarketSansMedium';
}

.ellipsis {
  overflow: hidden;
  text-overflow: ellipsis;
  display: -webkit-box;
  -webkit-line-clamp: 1;
  -webkit-box-orient: vertical;
}

.ellipsis2 {
  overflow: hidden;
  text-overflow: ellipsis;
  display: -webkit-box;
  -webkit-line-clamp: 2;
  -webkit-box-orient: vertical;
}
```

[views] header.ejs

```
<nav class="navbar navbar-expand-lg bg-primary navbar-dark">
  <div class="container-fluid">
    <a class="navbar-brand" href="/">한빛미디어</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent"
      aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="/">Home</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```



[views] home.ejs

```
<div class="row my-5">
  <div class="col">
    <h1 class="text-center mb-5">Home</h1>
  </div>
</div>
```

[views] footer.ejs

```
<div class="row my-5">
  <div class="col">
    <hr>
    <h5 class="text-center">
      Copyright 2023. 인천일보아카데미 all rights reserved.
    </h5>
  </div>
</div>
```

[views] index.ejs


```
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" .../>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js" ...></script>
    <script src="https://code.jquery.com/jquery-1.12.4.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/handlebars.js/3.0.1/handlebars.js"></script>
  </head>
  <body>
    <div class="container">
      <%-include("header.ejs")%>
      <%-include(pageName)%>
      <%-include("footer.ejs")%>
    </div>
  </body>
</html>
```

## • 홈 페이지에 출력할 도서목록

한빛미디어 Home


안드로이드

검색



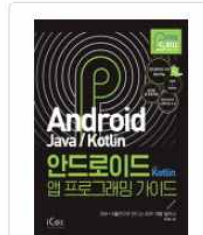
Do it! 안드로이드 ...

40,000원




안드로이드 앱 프로그래...

40,000원




안드로이드 with...

40,000원




안드로이드 with...

40,000원



안드로이드 앱 성능 ...

23,000원



안드로이드 프로그래...

32,000원

이전 1/84 다음

[views] home.ejs

```
<div class="row my-5">
  <div class="col">
    <%-include("book/list_book.ejs")%> <!--도서목록-->
  </div>
</div>
```

[views]-[book] list\_book.ejs

```
<div class="row justify-content-end">
  <div class="col-6 col-md-4">
    <form name="frm">
      <div class="input-group">
        <input type="text" class="form-control" id="query" value="자바">
        <button class="btn btn-primary">검색</button>
      </div>
    </form>
  </div>
</div>
<hr>
<div id="list_book" class="row"></div>

<!--도서목록 템플릿-->
<script id="temp_book" type="text/x-handlebars-template">
  {{#each documents}}
    <div class="col-6 col-md-4 col-lg-2">
      <div class="card my-2">
        <div class="card-body text-center">
          
          <div class="ellipsis mt-2">{{ title }}</div>
        </div>
        <div class="card-footer">
          <div>{{ format price }}</div>
        </div>
      </div>
    </div>
  {{/each}}
</script>
<!--템플릿 Register Helper-->
<script>
  Handlebars.registerHelper("image", function(thumbnail){
    if(!thumbnail) return "https://via.placeholder.com/120x174"
    else return thumbnail;
  });

  Handlebars.registerHelper("format", function(price){
    if(price) return price.toString().replace(/\B(?=(\d{3})+(?!\d))/g, ",") + "원";
  });
</script>

<script>
  getList();
  function getList() {
    $.ajax({
      type: "get",
      url: "https://dapi.kakao.com/v3/search/book?target=title",
      headers: { Authorization:"KakaoAK 4dc52ede9437e2cff0a338f1bd13b1c5" },
      data: { query:"자바", page:1, size:6 },
      dataType: "json",
      success: function(data) {
        let temp=Handlebars.compile($("#temp_book").html());
        $("#list_book").html(temp(data));
      }
    });
  }
</script>
```

- 도서목록 검색과 페이지 기능

```
[views]-[book] list_book.ejs
```

```
<div id="list_book" class="row"></div>
<div class="row my-3">
  <div class="col text-center">
    <button id="prev" class="btn btn-sm btn-primary px-4">
      이전
    </button>
    <span id="page" class="mx-2"> 1 </span>
    <button id="next" class="btn btn-sm btn-primary px-4">
      다음
    </button>
  </div>
</div>
...
<script>
  let page=1;
  let query="자바";c
  getList();

  //다음버튼을 클릭한 경우
  $("#next").on("click", function() {
    page++;
    getList();
  });

  //이전버튼을 클릭한 경우
  $("#prev").on("click", function() {
    page--;
    getList();
  });

  //검색어를 입력한 경우
  $(frm).on("submit", function(e) {
    e.preventDefault();
    query=$("#query").val();
    page=1;
    getList();
  });

  function getList() {
    $.ajax({
      type:'get',
      url:'https://dapi.kakao.com/v3/search/book?target=title',
      headers:{ Authorization:"KakaoAK 4dc52ede9437e2cff0a338f1bd13b1c5" },
      data:{ query:query, page:page, size:6 },
      dataType:'json',
      success:function(data){
        let temp=Handlebars.compile($("#temp_book").html());
        $("#list_book").html(temp(data));
        //마지막페이지 체크
        let is_end=data.meta.is_end;
        //검색 가능한 데이터 수
        let total=data.meta.pageable_count;
        //마지막 페이지
        let last = Math.ceil(total/6);
        $("#page").html(`${ page } / ${ last }`);
        if(page==1) $("#prev").attr("disabled", true); else $("#prev").attr("disabled", false);
        if(is_end) $("#next").attr("disabled", true); else $("#next").attr("disabled", false);
      }
    });
  }
</script>
```

- 도서정보 모달창

```
[views]-[book] modal_book.ejs
```

```
<div class="modal fade" id="modal_book{{@index}}" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1"
  aria-labelledby="staticBackdropLabel" aria-hidden="true">
  <div class="modal-dialog modal-lg">
    <div class="modal-content">
      <div class="modal-header">
        <h1 class="modal-title fs-5" id="staticBackdropLabel">도서정보</h1>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <div class="row">
          <div class="col-md-4">
            
          </div>
          <div class="col">
            <h5 class="my-2">{{ title }}</h5>
            <div class="my-2">저 자: <span>{{ authors }}</span></div>
            <div class="my-3">가 격: <span>{{ format price }}</span></div>
            <div class="my-3">출판사: <span>{{ publisher }}</span></div>
            <div class="my-3">출판일: <span>{{ datetime }}</span></div>
            <div class="my-3">ISBN: <span>{{ isbn }}</span></div>
          </div>
        </div>
        <hr>
        <div class="row">
          <div class="col mt-3" style="font-size:0.8rem;">{{ contents }}</div>
        </div>
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">
          닫기
        </button>
      </div>
    </div>
  </div>
</div>
```

```
[views]-[book] list_book.ejs
```

```
...
<!--도서목록 템플릿-->
<script id="temp_book" type="text/x-handlebars-template">
  {{#each documents}}
    <div class="col-6 col-md-4 col-lg-2">
      <div class="card my-2">
        <div class="card-body text-center">
          
          <div class="ellipsis mt-2">{{ title }}</div>
        </div>
        <div class="card-footer">
          <div>{{ format price }}</div>
        </div>
      </div>
      <%-include("modal_book.ejs")%>
    </div>
  {{/each}}
</script>
...
<script>
  ...
  $("#list_book").on("click", ".card img", function() {
    let index=$(this).attr("index");
    $("#modal_book" + index).modal("show");
  });
</script>
```

- 홈 페이지에 출력할 베스트 도서목록 (내비게이션 & 탭)

```
[views]-[book] best_book.ejs
```

```
<ul class="nav nav-tabs" id="myTab" role="tablist">
  <li class="nav-item" role="presentation">
    <button class="nav-link active" id="html-tab" data-bs-toggle="tab"
      data-bs-target="#tab-pane" type="button"
      role="tab" aria-controls="java-tab-pane" aria-selected="true">HTML</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="android-tab" data-bs-toggle="tab"
      data-bs-target="#tab-pane" type="button"
      role="tab" aria-controls="android-tab-pane" aria-selected="false">안드로이드</button>
  </li>
  <li class="nav-item" role="presentation">
    <button class="nav-link" id="database-tab" data-bs-toggle="tab"
      data-bs-target="#tab-pane" type="button"
      role="tab" aria-controls="database-tab-pane" aria-selected="false">데이터베이스</button>
  </li>
</ul>
<div class="tab-content" id="myTabContent">
  <div class="tab-pane fade show active" id="tab-pane" role="tabpanel" aria-labelledby="java-tab" tabindex="0">
    <div id="best_book" class="row my-5"></div>
  </div>
</div>
<!--베스트목록템플릿-->
<script id="temp_best" type="text/x-handlebars-template">
  {{# each documents }}
    <div class="col-6 col-md-4 col-lg-2">
      <div class="card my-2">
        <div class="card-body text-center">
          
        </div>
      </div>
    </div>
  {{/each}}
</script>

<script>
  let best="HTML";
  getBest();

  $("#html-tab").on("click", function(){
    best="HTML";
    getBest();
  });

  $("#android-tab").on("click", function(){
    best="안드로이드";
    getBest();
  });

  $("#database-tab").on("click", function(){
    best="데이터베이스";
    getBest();
  });

  function getBest() {
    $.ajax({
      type: 'get',
      url: 'https://dapi.kakao.com/v3/search/book?target=title',
      headers: { Authorization:"KakaoAK 4dc52ede9437e2cff0a338f1bd13b1c5" },
      data:{ query:best, page:1, size:6 },
      dataType: 'json',
      success:function(data) {
        let temp=Handlebars.compile($("#temp_best").html());
        $("#best_book").html(temp(data));
      }
    });
  }
</script>
```



[views] home.ejs

```

<div class="row my-5">
  <div class="col">
    <%-include("book/list_book.ejs")%> <!--도서목록-->
    <%-include("book/best_book.ejs")%> <!--베스트도서목록-->
  </div>
</div>

```

## • 베스트 도서정보 모달창



[views]-[book] modal\_book.ejs

```

<!--베스트목록템플릿-->
<script id="temp_best" type="text/x-handlebars-template">
  {{#each documents }}
    <div class="col-6 col-md-4 col-lg-2">
      <div class="card my-2">
        <div class="card-body text-center">
          
        </div>
      </div>
      <%-include("modal_book.ejs")%>
    </div>
  {{/each}}
</script>
...
<script>
  $("#best_book").on("click", ".card img", function(){
    let index=$(this).attr("index");
    $("#best_book #modal_book" + index).modal("show");
  });
</script>

```

- 헤더 페이지에 이미지 Slick Slider 구현하기



[views] index.ejs

<head>

...

<script src="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.9.0/slick.min.js"></script>

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.9.0/slick.min.css">

<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/slick-carousel/1.9.0/slick-theme.min.css">

...

</script>

[public]-[styleheets] slick.css

.slick-arrow.slick-prev,

.slick-arrow.slick-next {

background: #E2E2E2;

width: 50px;

height: 50px;

display: flex;

align-items: center;

justify-content: center;

z-index: 1;

border-radius: 50%;

opacity: 80%;

margin: 0px 40px;

}

.slick-prev::before,

.slick-next::before {

display: none;

}

.slick-dots {

bottom: 20px;

}

.slick-dots li button::before {

color: gray;

opacity: 100%;

}

.slick-dots li.slick-active button::before {

color: skyblue;

}

[views] header.ejs

```
<link rel="stylesheet" href="/stylesheets/slick.css"/>
<style>
  @media screen and (max-width: 960px) {
    #slider-div { display: none; }
  }
</style>

<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="/">한빛미디어</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
      data-bs-target="#navbarNav"
      aria-controls="navbarNav" aria-expanded="false"
      aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="/">Home</a>
        </li>
      </ul>
    </div>
  </div>
</nav>

<div id="slider-div">
  <div></div>
  <div></div>
  <div></div>
  <div></div>
</div>

<script>
  applySlider();

  function applySlider() {
    $('#slider-div').slick({
      slide : 'div', //슬라이드 되어야 할 태그 ex) div, li
      infinite : true, //무한 반복 옵션
      slidesToShow : 1, // 한 화면에 보여 질 콘텐츠 개수
      slidesToScroll : 1, //스크롤 한 번에 움직일 콘텐츠 개수
      speed : 100, //다음 버튼 누르고 다음 화면 뜨는데 까지 걸리는 시간(ms)
      arrows : true, //옆으로 이동하는 화살표 표시 여부
      dots : true, //스크롤바 아래 점으로 페이지네이션 여부
      autoplay : true, //자동 스크롤 사용 여부
      autoplaySpeed : 10000, //자동 스크롤 시 다음으로 넘어가는데 걸리는 시간 (ms)
      pauseOnHover : true, //슬라이드 이동시 마우스 하버하면 슬라이더 멈추게 설정
      vertical : false, //세로 방향 슬라이드 옵션
      prevArrow : "<button type='button' class='slick-prev'><div style='font-size:1.5rem;color:white;'>&lt;</div></button>",
      nextArrow : "<button type='button' class='slick-next'><div style='font-size:1.5rem;color:white;'>&gt;</div></button>",
      dotsClass : "slick-dots", //아래 나오는 페이지네이션(점) css class 지정
      draggable : true, //드래그 가능 여부
      responsive : [ //반응형 웹구현 옵션
        {
          breakpoint : 960, //화면 사이즈 960px
          settings : {
            slidesToShow : 3
          } //위에 옵션이 디폴트, 여기에 추가하면 그걸로 변경
        }, {
          breakpoint : 768, //화면 사이즈 768px
          settings : {
            slidesToShow : 2
          } //위에 옵션이 디폴트, 여기에 추가하면 그걸로 변경
        }
      ]
    });
  }
</script>
```



- 홈 페이지에 출력할 최신 도서목록



[views]-[book] recent-book.ejs

```
<div id="recent_book" class="row"></div>
```

```
<!--최신도서목록템플릿-->
```

```
<script id="temp_recent" type="text/x-handlebars-template">
```

```
  {{#each documents}}
```

```
    <div class="col-6 col-md-2 my-5">
```

```
      <div class="card my-2 mx-2">
```

```
        <div class="card-body text-center">
```

```
          
```

```
        </div>
```

```
      </div>
```

```
    </div>
```

```
  {{/each}}
```

```
</script>
```

```
<script>
```

```
  getRecent();
```

```
  function getRecent() {
```

```
    $.ajax({
```

```
      type: 'get',
```

```
      url: 'https://dapi.kakao.com/v3/search/book?target=title',
```

```
      headers: { Authorization:"KakaoAK 4dc52ede9437e2cff0a338f1bd13b1c5" },
```

```
      data: { query:'챗GPT', page:1, size:10 },
```

```
      dataType: 'json',
```

```
      success:function(data) {
```

```
        let temp=Handlebars.compile($("#temp_recent").html());
```

```
        $("#recent_book").html(temp(data));
```

```
        recentSlider();
```

```
      }
```

```
    });
```

```
  }
```

```
  function recentSlider() {
```

```
    $('#recent_book').slick({
```

```
      slidesToShow : 6,
```

```
      slide : 'div',
```

```
      infinite : true,
```

```
      ...
```

```
    });
```

```
  }
```

```
</script>
```

[views] home.ejs

```
<div class="row my-5">
```

```
  <div class="col">
```

```
    <%-include("book/list_book.ejs")%> <!--도서목록-->
```

```
    <%-include("book/best_book.ejs")%> <!--베스트도서목록-->
```

```
    <%-include("book/recent_book.ejs")%> <!--최근도서목록-->
```

```
  </div>
```

```
</div>
```

- Firebase 설정

[public]-[javascripts] firebaseInit.js SDK 설정 및 구성 CDN

```
import { initializeApp } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-app.js";
import { getAnalytics } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-analytics.js";
```

```
const firebaseConfig = {
  apiKey: "xxxxxxxxxxx",
  authDomain: "xxxxxxxxxxx",
  databaseURL: "xxxxxxxxxxx",
  projectId: "xxxxxxxxxxx",
  storageBucket: "xxxxxxxxxxx",
  messagingSenderId: "xxxxxxxxxxx",
  appId: "xxxxxxxxxxx",
  measurementId: "xxxxxxxxxxx"
};
```

```
export const app = initializeApp(firebaseConfig);
const analytics = getAnalytics(app);
```

- 로그인/로그아웃

[routes] users.js

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

router.get('/login', function (req, res) {
  res.render('index', { title: '로그인', pageName: 'users/login.ejs' });
});
module.exports = router;
```

[views]-[users] login.ejs

```
<div class="row my-5 justify-content-center">
  <div class="col-8 col-md-6 col-lg-4">
    <h3 class="text-center mb-5">로그인</h3>
    <form name="frm" method="post">
      <div class="input-group my-2">
        <div class="input-group-text">이 메 일</div>
        <input class="form-control" name="email" value="hong@inha.com">
      </div>
      <div class="input-group">
        <div class="input-group-text">비밀번호</div>
        <input class="form-control" name="password" type="password" value="12345678">
      </div>
      <div class="my-3"><button class="btn btn-primary w-100">로그인</button></div>
      <div class="text-end"><a href="/users/join">회원가입</a></div>
    </form>
  </div>
</div>
<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getAuth, signInWithEmailAndPassword } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-auth.js";
  const auth=getAuth(app);

  $(frm).on("submit", function(e){
    e.preventDefault();
    let email=$(frm.email).val();
    let password=$(frm.password).val();
    signInWithEmailAndPassword(auth, email, password).then(success=>{
      sessionStorage.setItem('email', email);
      sessionStorage.setItem('uid', success.user.uid);
      location.href="/";
    }).catch(error=>{
      alert(error.message);
    });
  });
</script>
```

[views] header.ejs

```
<nav class="navbar navbar-expand-lg bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="/">한빛미디어</a>
    <button class="navbar-toggler" type="button" data-bs-toggle="collapse"
      data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent"
      aria-expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarSupportedContent">
      <ul class="navbar-nav me-auto mb-2 mb-lg-0">
        <li class="nav-item">
          <a class="nav-link active" aria-current="page" href="/">Home</a>
        </li>
      </ul>
      <div class="d-flex">
        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
          <li class="nav-item" id="login">
            <a class="nav-link" aria-current="page" href="/users/login">로그인</a>
          </li>
          <li class="nav-item" id="email" style="display:none">
            <a class="nav-link" aria-current="page" href="#">email</a>
          </li>
          <li class="nav-item" id="logout" style="display:none">
            <a class="nav-link" aria-current="page" href="#">로그아웃</a>
          </li>
        </ul>
      </div>
    </div>
  </div>
</nav>
<div id="slider-div">
  <div></div>
  <div></div>
  <div></div>
  <div></div>
</div>
<script>
  if(sessionStorage.getItem("email")){
    $("#login, #logout, #email").toggle();
    $("#email a").html(sessionStorage.getItem("email"));
  }

  $("#logout").on("click", function(){
    sessionStorage.clear();
    location.href="/";
  });

  applySlider();
  function applySlider() {
    $('#slider-div').slick({
      slide : 'div',
      infinite : true,
      slidesToShow :1,
      slidesToScroll : 1,
      ...
    });
  }
</script>
```

- 회원가입

[routes] users.js

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

router.get('/', function(req, res, next) {
  res.send('respond with a resource');
});

router.get('/login', function(req, res) {
  res.render('index', {title: '로그인', pageName: 'users/login.ejs'})
});

router.get('/join', function(req, res) {
  res.render('index', { title: '회원가입', pageName: 'users/join.ejs' })
});

module.exports = router;
```

[views]-[users] join.ejs

```
<div class="row my-5 justify-content-center">
  <div class="col-8 col-md-6 col-lg-4">
    <h3 class="text-center mb-5">회원가입</h3>
    <form name="frm" method="post">
      <div class="input-group my-2">
        <div class="input-group-text">이 메 일</div>
        <input class="form-control" name="email" value="hong@inha.com">
      </div>
      <div class="input-group">
        <div class="input-group-text">비밀번호</div>
        <input class="form-control" name="password" type="password" value="12345678">
      </div>
      <div class="my-3"><button class="btn btn-primary w-100">회원가입</button></div>
      <div class="text-end"><a href="/users/login">로그인</a></div>
    </form>
  </div>
</div>

<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getAuth, createUserWithEmailAndPassword } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-auth.js";
  const auth=getAuth(app);

  $(frm).on("submit", function(e) {
    e.preventDefault();
    let email=$(frm.email).val();
    let password=$(frm.password).val();
    if(email == "" || password == "") {
      alert("이메일과 비밀번호를 꼭 입력하세요!");
    }else {
      createUserWithEmailAndPassword(auth, email, password).then(success=> {
        location.href="/users/login";
      }).catch(error=> {
        alert(error.message);
      });
    }
  });
</script>
```

- 장바구니에 도서정보 저장

[views]-[book] list\_book.ejs

```
<!--도서목록 템플릿-->
<script id="temp_book" type="text/x-handlebars-template">
  {{#each documents}}
    <div class="col-6 col-md-4 col-lg-2">
      <div class="card my-2">
        <div class="card-body text-center">
          
          <div class="ellipsis mt-2">{{ title }}</div>
        </div>
        <div class="card-footer" style="font-size:0.5rem;">
          <div class="text-center">
            {{ format price }}
            <span class="cart ms-3" book="{{ book @this }}" style="cursor:pointer;color:green;">CART</span>
          </div>
        </div>
      </div>
    </div>
    <%-include("modal_book.ejs")%>
  </div>
  {{/each}}
</script>
<!--템플릿 Register Helper-->
<script>
  ...
  Handlebars.registerHelper("book", function(book) {
    return JSON.stringify(book);
  });
</script>
<script type="module">
  import { app } from "/javascripts/firebaseinit.js";
  import { getDatabase, ref, set, get } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-database.js";
  const db = getDatabase(app);
  const uid=sessionStorage.getItem("uid");
  const email=sessionStorage.getItem("email");
  ...
  //장바구니에 넣기
  $("#list_book").on("click", ".card .cart", async function(){
    if(uid) {
      const ref_book=ref(db,`cart/${uid}/${book.isbn}`);
      const snapshot=await get(ref_book);
      if(snapshot.val()){
        alert("장바구니에 이미 존재하는 도서입니다!");
      }else {
        if(!confirm("장바구니에 저장하실래요?")) return;
        await set(ref_book, book);
        alert("장바구니에 등록되었습니다.");
      }
    }else {
      location.href="/users/login";
    }
  });
  ...
</script>
```

- 장바구니 목록

[routes] users.js

```
//장바구니 페이지
router.get('/cart', function(req, res) {
  res.render('index', { title: '장바구니', pageName: 'users/cart.ejs' });
});
```

[views] header.ejs

```
...
<div class="collapse navbar-collapse" id="navbarSupportedContent">
  <ul class="navbar-nav me-auto mb-2 mb-lg-0">
    <li class="nav-item">
      <a class="nav-link active" aria-current="page" href="/">Home</a>
    </li>
    <li class="nav-item" id="cart" style="display:none">
      <a class="nav-link active" aria-current="page" href="/users/cart">장바구니</a>
    </li>
  </ul>
</div>

<script>
  if(sessionStorage.getItem("email")) {
    $("#login, #logout, #email, #cart").toggle();
    $("#email a").html(sessionStorage.getItem("email"));
  }
...
</script>
```

[views]-[users] cart.ejs

```
<div class="row my-5">
  <div class="col">
    <h1 class="text-center mb-5">장바구니</h1>
    <div id="div_cart"></div>
  </div>
</div>

<!--장바구니 템플릿-->
<script id="temp_cart" type="text/x-handlebars-template">
  <table class="table table-striped">
    <tr>
      <td>{{title}}</td>
      <td>{{authors}}</td>
      <td>{{price}}</td>
      <td><button class="btn btn-danger btn-sm isbn="{{ isbn }}">삭제</button></td>
    </tr>
  </table>
</script>

<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getDatabase, ref, onValue, remove } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-database.js";
  const db = getDatabase(app);
  const uid=sessionStorage.getItem('uid');

  //장바구니 목록
  onValue(ref(db, `cart/${uid}`), snapshot=>{
    let rows=[];
    snapshot.forEach(row=>{
      rows.push({ key:row.key, ...row.val() });
    });
    //console.log(rows);
    let temp=Handlebars.compile($("#temp_cart").html());
    $("#div_cart").html(temp(rows));
  });

  //장바구니 삭제
  $("#div_cart").on("click", ".btn-danger", async function(){
    let isbn=$(this).attr("isbn");
    if(confirm(isbn + ' 삭제하실래요?')){
      await remove(ref(db, `cart/${uid}/${isbn}`));
    }
  });
</script>
```

- 마이페이지

[routes] users.js

```
//마이 페이지
router.get('/mypage', function(req, res) {
  res.render('index', { title: '회원정보', pageName: 'users/mypage.ejs' })
});
```

[views] header.ejs

```
<div class="collapse navbar-collapse" id="navbarSupportedContent">
  ...
  <ul class="navbar-nav me-auto mb-2 mb-lg-0">
    <li class="nav-item" id="email" style="display:none">
      <a class="nav-link" aria-current="page" href="/users/mypage">MyPage</a>
    </li>
    <li class="nav-item" id="logout" style="display:none">
      <a class="nav-link" aria-current="page" href="#">로그아웃</a>
    </li>
  </ul>
</div>
```

[views]-[users] mypage.ejs

```
<div class="row my-5">
  <div class="col">
    <h1 class="text-center mb-3">회원정보</h1>
    <div id="mypage"></div>
    <div class="text-center my-3">
      <a href="/users/update"><button class="btn btn-primary px-5">정보수정</button></a>
    </div>
  </div>
```

<!--마이페이지 템플릿-->

```
<script id="temp_mypage" type="text/x-handlebars-template">
  <div class="card">
    <div class="row p-3">
      <div class="col-md-4 col-lg-5 text-center">
        
      </div>
      <div class="col mt-3">
        <h5>이름: {{name}}</h5><hr>
        <div class="my-3">주소: {{address}}</div>
        <div class="my-3">전화: {{phone}}</div>
      </div>
    </div>
  </div>
</script>
```

```
<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getFirestore, getDoc, doc, setDoc } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  const db = getFirestore(app);
  const uid=sessionStorage.getItem("uid");
  const email=sessionStorage.getItem("email");

  let user={ name: '무기명', photo:'https://via.placeholder.com/200x200', address: '인천 남구 학익동', phone: '010-0000-0000' };
  const snapshot = await getDoc(doc(db, `users/${uid}`));
  if(snapshot.data()) {
    user=snapshot.data();
    user={ ...user, photo:snapshot.data().photo ? snapshot.data().photo : "http://via.placeholder.com/200x200" }
  };
  const temp=Handlebars.compile($("#temp_mypage").html());
  $("#mypage").html(temp(user));
</script>
```

- 회원정보 수정

[routes] users.js

```
router.get('/update', function(req, res) {
  res.render('index', { title: '회원정보수정', pageName: 'users/update.ejs' })
});
```

[views]-[users] update.ejs

```
<div class="row my-5">
  <div class="col">
    <h1 class="text-center mb-5">회원정보수정</h1>
    <div class="card p-5">
      <form name="frm" method="post">
        <div class="input-group my-2">
          <div class="input-group-text px-5">성명</div>
          <input class="form-control" name="name" value="무기명">
        </div>
        <div class="input-group my-2">
          <div class="input-group-text px-5">주소</div>
          <input class="form-control" name="address" value="인천 남구 학익동 인천일보아카데미">
        </div>
        <div class="input-group my-2">
          <div class="input-group-text px-5">전화</div>
          <input class="form-control" name="phone" value="010-1010-2020">
        </div>
        <div>
          
          <input class="form-control mt-2" type="file">
        </div>
        <div class="text-center mt-3">
          <button class="btn-primary btn px-5">정보수정</button>
        </div>
      </form>
    </div>
  </div>
</div>
<script src="https://t1.daumcdn.net/mapjsapi/bundle/postcode/prod/postcode.v2.js"></script>
<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getFirestore, setDoc, doc, getDoc } from "https://www.gstatic.com/firebasejs/9.22.0/firebase-firestore.js";
  const db = getFirestore(app);
  const uid=sessionStorage.getItem("uid");

  const snapshot = await getDoc(doc(db, `users/${uid}`));
  if(snapshot.data()) {
    const user=snapshot.data();
    $(frm.name).val(user.name);
    $(frm.phone).val(user.phone);
    $(frm.address).val(user.address);
  }

  $(frm).on("submit", async function(e){
    e.preventDefault();
    if(window.confirm("변경 내용을 수정하실래요?")) {
      let name=$(frm.name).val();
      let address=$(frm.address).val();
      let phone=$(frm.phone).val();
      let email=sessionStorage.getItem("email");
      await setDoc(doc(db, `users/${uid}`), {
        email: email,
        name: name,
        address: address,
        phone: phone
      });
      alert("사용자 정보가 변경되었습니다.");
      location.href="/users/mypage";
    }
  });
</script>
```

```
//주소검색 버튼을 클릭한 경우
$("#searchaddress").on("click", function(e){
  e.preventDefault();
  new daum.Postcode({
    oncomplete: function(data){
      if(data.buildingName!="") {
        address = data.address + " " + data.buildingName;
        $(frm.address).val(address);
      } else {
        $(frm.address).val(data.address);
      }
    }
  }).open();
});
```



- 사용자 정보 사진미리보기 및 등록

[views]-[users] update.ejs

```
<div class="row my-5">
  <div class="col">
    <h1 class="text-center mb-5">회원정보수정</h1>
    <div class="card p-5">
      <form name="frm" method="post">
        ...
        <div>
          
          <input class="form-control mt-2" type="file" name="file">
        </div>
        <div class="text-center mt-3">
          <button class="btn-primary btn px-5">정보수정</button>
        </div>
      </form>
    </div>
  </div>
</div>

<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getFirestore, setDoc, doc, getDoc } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  import { getStorage, uploadBytes, ref, getDownloadURL } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-storage.js";
  const storage = getStorage(app);
  const db = getFirestore(app);
  const uid = sessionStorage.getItem("uid");

  const snapshot = await getDoc(doc(db, `users/${uid}`));
  if(snapshot.data()) {
    const user = snapshot.data();
    $(frm.name).val(user.name);
    $(frm.phone).val(user.phone);
    $(frm.address).val(user.address);
    if(user.photo) $("#fileName").attr("src", user.photo)
  }

  $(frm).on("submit", async function(e){
    e.preventDefault();
    if(window.confirm("변경 내용을 수정하시겠습니까?")) {
      let name = $(frm.name).val();
      let address = $(frm.address).val();
      let phone = $(frm.phone).val();
      let email = sessionStorage.getItem('email');
      if(frm.file.files[0]) {
        //이미지업로드
        const snapshot = await uploadBytes(ref(storage, `photo/${Date.now().jpg}`), frm.file.files[0]);
        //업로드한 이미지 URL
        const url = await getDownloadURL(snapshot.ref);
        await setDoc(doc(db, `users/${uid}`),
          { email:email, name:name, address:address, phone:phone, photo:url })
      } else {
        const photo = $("#fileName").attr("src");
        await setDoc(doc(db, `users/${uid}`),
          { email:email, name:name, address:address, phone:phone, photo:photo })
      }
      alert("사용자 정보가 변경되었습니다.");
      location.href="/users/mypage";
    }
  });

  //이미지 미리보기
  $(frm.file).on("change", function(e) {
    $("#fileName").attr("src", URL.createObjectURL(e.target.files[0]));
  })
</script>
```

- 게시글목록

[routes] index.js

```
//게시글 목록
router.get('/posts', function(req, res) {
  res.render('index', { title: '게시글', pageName: 'posts/list.ejs' });
});
```

[views]-[posts] list.ejs

```
<div class="row my-2">
  <div class="col">
    <h1 class="text-center mb-2">게시글</h1>
    <div class="text-end">
      <button class="btn btn-primary btn-sm" id="btn-write">글쓰기</button>
    </div>
    <div id="posts"></div>
  </div>
</div>

<!--게시글목록 템플릿-->
<script id="temp" type="text/x-handlebars-template">
  {{#each .}}
  <div class="card my-2" style="font-size:0.8rem">
    <div class="card-body">
      <h5>{{ title }}</h5>
      <div class="ellipsis">{{ body }}</div>
    </div>
    <div class="card-footer">
      <span>Posted on {{ date }} by {{ email }}</span>
    </div>
  </div>
  {{/each}}
</script>

<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getFirestore, collection } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  import { query, orderBy, onSnapshot } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  const db = getFirestore(app);
  const email=sessionStorage.getItem('email');

  getList();
  function getList() { //게시글 목록출력
    const q=query(collection(db, "posts"), orderBy("date", "desc"));
    onSnapshot(q, snapshot=>{
      let rows=[];
      snapshot.forEach(row=>{
        rows.push({
          id: row.id,
          ...row.data()
        });
      });
      let temp=Handlebars.compile($("#temp").html());
      $("#posts").html(temp(rows));
    });
  }
</script>
```

[routes] header.ejs

```
...
<li class="nav-item">
  <a class="nav-link active" aria-current="page" href="/">Home</a>
</li>
<li class="nav-item">
  <a class="nav-link active" aria-current="page" href="/posts">게시글</a>
</li>
...
```

- 글쓰기

[views]-[posts] writer.ejs

```
<script>
...
$("#btn-write").on("click", function(){
  if(email){
    location.href="/posts/write";
  }else{
    sessionStorage.setItem("target", "/posts/write");
    location.href="/users/login";
  }
});
</script>
```

[routes]-[users] login.ejs

```
...
signInWithEmailAndPassword(auth, email, password).then(success=>{
  sessionStorage.setItem("email", email);
  sessionStorage.setItem("uid", success.user.uid);
  if(sessionStorage.getItem("target")) {
    location.href=sessionStorage.getItem("target");
  } else {
    location.href="/";
  }
}).catch(error=>{
  alert(error.message);
});
```

[routes] index.js

```
router.get('/posts/write', function(req, res) {
  res.render('index', { title: '글쓰기', pageName: 'posts/write.ejs' })
});
```

[views]-[posts] writer.ejs

```
<div class="row my-5">
  <div class="col">
    <h1 class="text-center mb-5">글쓰기</h1>
    <form name="frm" method="post">
      <input class="form-control my-2" name="title" placeholder="제목을 입력하세요.">
      <textarea class="form-control" name="body" rows="10" placeholder="내용을 입력하세요."></textarea>
      <div class="text-center my-2">
        <button class="btn btn-primary">글등록</button>
        <button class="btn btn-secondary" type="reset">등록취소</button>
      </div>
    </form>
  </div>
</div>
<script src="https://cdn.jsdelivr.net/npm/moment@2.29.4/moment.min.js"></script>
<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getFirestore, collection, addDoc } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  const db = getFirestore(app);

  $(frm).on("submit", async function(e) {
    e.preventDefault();
    let title=$(frm.title).val();
    let body=$(frm.body).val();
    let date=moment(new Date()).format("YYYY-MM-DD HH:mm:ss");
    let email=sessionStorage.getItem("email");

    await addDoc(collection(db, "posts"), { email: email, title: title, body: body, date: date });
    location.href="/posts";
  });
</script>
```

- 게시글 정보

[views]-[posts] list.ejs

```
<!--게시글목록 템플릿-->
<script id="temp" type="text/x-handlebars-template">
  {{#each .}}
    ...
    <a href="/posts/{{ id }}"><h5>{{ title }}</h5></a>
    ...
  {{/each}}
</script>
```

[routes] index.js

```
router.get('/posts/:id', function(req, res) {
  let id=req.params.id;
  res.render('index', { title: '글정보', pageName: 'posts/read.ejs', id:id });
});
```

[views]-[posts] read.ejs

```
<div class="row my-5">
  <div class="col">
    <h1 class="text-center mb-5">글정보</h1>
    <div class="text-end mb-2" id="btn-group" style="display:none">
      <button class="btn btn-primary btn-sm" id="btn-update">수정</button>
      <button class="btn btn-danger btn-sm" id="btn-delete">삭제</button>
    </div>
    <div id="post"></div>
  </div>
</div>
<!--게시글정보 템플릿-->
<script id="temp" type="text/x-handlebars-template">
  <div class="card">
    <div class="card-body">
      <h5>{{ title }}</h5>
      <div>{{ body }}</div>
    </div>
    <div class="card-footer">
      Posted on <span>{{ date }}</span> by <span>{{ email }}</span>
    </div>
  </div>
</script>
<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getFirestore, doc, getDoc, deleteDoc } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  const db = getFirestore(app);
  const id="<%=id%>";

  const snapshot = await getDoc(doc(db, "posts", id));
  const post=snapshot.data();
  let temp=Handlebars.compile($("#temp").html());
  $("#post").html(temp(post));

  if(sessionStorage.getItem("email") == post.email){
    $("#btn-group").toggle();
  }

  $("#btn-delete").on("click", async function() { //게시글삭제
    if(confirm(id + "번 게시글을 삭제하실래요?")) {
      await deleteDoc(doc(db, `posts/${id}`));
      location.href="/posts";
    }
  });

  $("#btn-update").on("click", function() { //게시글수정
    location.href="/posts/update/" + id;
  });
</script>
```

[routes] index.js

```
...
//게시글 수정
router.get('/posts/update/:id', function(req, res) {
  let id=req.params.id;
  res.render('index', { title: '글정보수정', pageName: 'posts/update.ejs', id: id });
});
...
```

[views]-[posts] update.ejs

```
<div class="row my-5">
  <div class="col">
    <h1 class="text-center mb-5">글수정</h1>
    <form name="frm" method="post">
      <input class="form-control my-2" name="title" placeholder="제목을 입력하세요.">
      <textarea class="form-control" name="body" rows="10" placeholder="내용을 입력하세요."></textarea>
      <div class="text-center my-2">
        <button class="btn btn-primary">글수정</button>
        <button class="btn btn-secondary" type="reset">수정취소</button>
      </div>
    </form>
  </div>
</div>

<!-- 날짜포맷 지정함수 -->
<script src="https://cdn.jsdelivr.net/npm/moment@2.29.4/moment.min.js"></script>
<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getFirestore, doc, getDoc, setDoc } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  const db = getFirestore(app);
  //게시글 id 읽어오기
  const id="<%=id%>";
  let post=null;

  getPost();

  //게시글정보 읽어오기
  async function getPost() {
    const snapshot = await getDoc(doc(db, `posts/${id}`));
    post=snapshot.data();
    $(frm.title).val(post.title);
    $(frm.body).val(post.body);
  }

  $(frm).on("submit", async function(e) {
    e.preventDefault();
    if(confirm("수정된 정보를 저장하실래요?")) {
      let title=$(frm.title).val();
      let body=$(frm.body).val();
      await setDoc(doc(db, `posts/${id}`), {
        ...post,
        title: title,
        body: body
      });
      location.href="/posts";
    }
  });

  $(frm).on("reset", function(e) {
    e.preventDefault();
    getPost();
  });
</script>
```

- 좋아요 추가 또는 삭제

[views]-[posts] list.ejs

```
<style>
  .heart0, .heart1 { cursor: pointer; float: right; color: red; font-size: 1.5rem; }
  .fcnt { font-size: 0.5rem; float:right; margin-top: 15px; }
</style>
...
<!--게시글목록 템플릿-->
<script id="temp" type="text/x-handlebars-template">
  ...
  <div class="card-footer">
    <span>Posted on {{ date }} by {{ email }}</span>
    <span class="heart{{ucnt}}" id="{{ id }}"><span>{{ heart ucnt }}</span><span class="fcnt">{{ fcnt }}</span></span>
  </div>
  ...
</script>
<script>
  Handlebars.registerHelper("heart", function(ucnt){
    if(ucnt==0) return "♡"; else return "♥";
  });
</script>

<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getFirestore, collection, addDoc, deleteDoc } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  import { query, orderBy, getDocs, where, doc } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  const db = getFirestore(app);
  const email=sessionStorage.getItem('email');

  getList();
  async function getList(){
    const q=query(collection(db, "posts"), orderBy("date", "desc"));
    const snapshot = await getDocs(q);
    let rows=[];
    snapshot.forEach(async row=> {
      //전체 좋아요 개수
      const q1=query(collection(db,"favorite"), where("pid","=",row.id));
      const snap1=await getDocs(q1);
      //로그인한 사용자의 좋아요 체크 유무
      const q2=query(collection(db,"favorite"), where("pid","=",row.id), where("email","=",email));
      const snap2=await getDocs(q2);
      //출력 항목 추가
      rows.push({ id: row.id, ...row.data(), fcnt: snap1.docs.length, ucnt: snap2.docs.length });
      let temp=Handlebars.compile($("#temp").html());
      $("#posts").html(temp(rows));
    });
  }

  $("#posts").on("click", ".heart0", async function() { //좋아요 추가
    let id=$(this).attr("id");
    if(email) {
      await addDoc(collection(db,"favorite"),{ pid:id, email:email });
      alert("add favorite"); getList();
    }else {
      sessionStorage.setItem("target", "/posts");
      location.href="/users/login";
    }
  });

  $("#posts").on("click", ".heart1", async function() { //좋아요 삭제
    let id=$(this).attr("id");
    const q=query(collection(db,"favorite"), where("pid","=",id), where("email","=",email));
    const snapshot=await getDocs(q);
    snapshot.forEach(row=> {
      deleteDoc(doc(db, "favorite", row.id));
      alert("delete favorite"); getList();
    });
  });
  ...
</script>
```

- 로딩바 출력

```
[public]-[stylesheets] style.css
```

```
#loading {
  width: 100%;
  height: 100%;
  top: 0;
  left: 0;
  position: fixed;
  display: block;
  opacity: 0.8;
  background: white;
  z-index: 99;
  text-align: center;
}
```

```
[views] index.js
```

```
<div class="container">
  ...
  <div id="loading" class="row my-5"><div class="col my-5"></div></div>
  <%-include("footer.ejs")%>
</div>
<script>
  $(document).on("ready", function() {
    setTimeout(function() { $("#loading").hide(); }, 300);
  });
</script>
```

- 게시글 페이지

```
[views]-[posts] list.ejs
```

```
<div id="posts"></div>
<div class="text-center my-3" id="btn-group" style="display:none">
  <button class="btn btn-primary btn-sm" id="btn-prev">이전</button>
  <span class="px-2" id="span-page">1</span>
  <button class="btn btn-primary btn-sm" id="btn-next">다음</button>
</div>
<script>
  let page=1;
  getList();

  $("#btn-next").on("click", function(){
    page++; getList();
  });
  $("#btn-prev").on("click", function(){
    page--; getList();
  });

  async function getList(){
    ...
    snapshot.docs.forEach(async(row, index)=> {
      $("#loading").show();
      let size=3;
      let last = Math.ceil(snapshot.docs.length/size);
      let start = (page-1) * size;
      let end = (page * size) - 1;
      if(index >= start && index <= end) {
        ...
        if(snapshot.docs.length > 0) $("#btn-group").show();
        $("#span-page").html(page + "/" + last);
        if(page==1) $("#btn-prev").attr("disabled", true) else $("#btn-prev").attr("disabled", false);
        if(page==last) $("#btn-next").attr("disabled", true) else $("#btn-next").attr("disabled", false);
        $("#loading").hide();
      }
    });
  }
</script>
```

- 게시글에 대한 실시간 채팅하기

[views]-[posts] read.ejs

```
<div class="row my-5">
  <div class="col">
    <h1 class="text-center mb-5">글정보</h1>
    <div class="text-end mb-2" id="btn-group" style="display:none">
      <button class="btn btn-primary btn-sm" id="btn-update">수정</button>
      <button class="btn btn-danger btn-sm" id="btn-delete">삭제</button>
    </div>
    <div id="post"></div>
  </div>
</div>
<%-include("chat.ejs")%>
```

[views]-[posts] chat.ejs

```
<div class="row">
  <div class="col">
    <button class="btn btn-primary w-100" id="btn-chat">채팅하기</button>
    <div class="card" id="card-chat" style="display:none">
      <div class="card-body">
        <div id="wrap" style="height:400px;"></div>
      </div>
      <div class="card-footer">
        <form name="frm" method="post">
          <input class="form-control" name="text">
        </form>
      </div>
    </div>
  </div>
</div>
<script src="https://cdn.jsdelivr.net/npm/moment@2.29.4/moment.min.js"></script>
<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getFirestore,getDoc,doc,addDoc,collection } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  const db = getFirestore(app);

  const pid="<%=id%>";
  const uid=sessionStorage.getItem("uid");
  const email=sessionStorage.getItem("email");
  let user = { name: "무기명", photo: "https://via.placeholder.com/50x50" };

  //로그인한 경우 사용자 정보 읽어오기
  if(uid) {
    $("#btn-chat, #card-chat").toggle();
    const snapshot = await getDoc(doc(db, "users", uid));
    if(snapshot.data()) user=snapshot.data();
  }

  $("#btn-chat").on("click", function(){
    sessionStorage.setItem("target", `/posts/${pid}`);
    location.href="/users/login";
  });

  $(frm).on("submit", async function(e) {
    e.preventDefault();
    let text=$(frm.text).val();
    if(text=="") {
      alert("메시지 내용을 입력하세요!");
      $(frm.text).focus();
    }else {
      const date = moment(new Date()).format("YYYY-MM-DD HH:mm:ss");
      await addDoc(collection(db, "chat"), {
        pid: pid, email: email, date: date, name: user.name, photo: user.photo, text: text
      });
      $(frm.text).val("");
    }
  });
</script>
```



- 채팅목록 출력 (복합쿼리인 경우 아래와 같이 인텍스를 생성한다.)

				색인 추가	
컬렉션 ID	색인이 생성된 필드			쿼리 범위	상태
chat	pid 오름차순	date 오름차순	__name__ 오름차순	컬렉션	사용 설정됨

[views]-[posts] chat.ejs

```

<style>
  #wrap {
    overflow-y: scroll;
    overflow-x: hidden;
  }
</style>
...
<!--채팅목록 템플릿-->
<script id="temp_chat" type="text/x-handlebars-template">
  {{#each .}}
    <div class="my-3 chat row" style="font-size:0.8rem;">
      <div class="col-md-1">
        
      </div>
      <div class="col">
        <div>{{ name }} : {{ text }}</div>
        <div style="font-size:0.5rem;">{{ date }}</div>
      </div>
    </div>
  {{/each}}
</script>
...
<script type="module">
  import { app } from "/javascripts/firebaseInit.js";
  import { getFirestore,getDoc,doc,addDoc,collection } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  import { onSnapshot,where,orderBy,query } from "https://www.gstatic.com/firebasejs/9.22.1/firebase-firestore.js";
  const db = getFirestore(app);
  ...

  //로그인한 경우 사용자 정보 읽어오기
  if(uid) {
    $("#btn-chat, #card-chat").toggle();
    const snapshot = await getDoc(doc(db, 'users', uid));
    if(snapshot.data()) user=snapshot.data();
    getChat();
  }

  function getChat() {
    const q=query(collection(db,"chat"), where("pid","==",pid), orderBy("date"));
    onSnapshot(q, snapshot=> {
      let rows=[];
      snapshot.forEach(row=> {
        rows.push({ id: row.id, ...row.data() });
      });
      const temp=Handlebars.compile($("#temp_chat").html());
      $("#wrap").html(temp(rows));
      $("#wrap").scrollTop($('#wrap')[0].scrollHeight);
    });
  }

  $("#btn-chat").on("click", function(){
    sessionStorage.setItem("target", `/posts/${pid}`);
    location.href="/users/login";
  });

  $(frm).on("submit", async function(e) {
    ...
  });
</script>

```

- Alert, Conform 모달 창

[views]-[common] modal\_alert.ejs

```
<div class="modal fade" id="modal-alert" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1"
  aria-labelledby="staticBackdropLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h1 class="modal-title fs-5" id="staticBackdropLabel">알림</h1>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">...</div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Close</button>
      </div>
    </div>
  </div>
</div>
```

[views]-[common] modal\_confirm.ejs

```
<div class="modal fade" id="modal-confirm" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1"
  aria-labelledby="staticBackdropLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h1 class="modal-title fs-5" id="staticBackdropLabel">알림</h1>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">...</div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">아니오</button>
        <button type="button" class="btn btn-primary">예</button>
      </div>
    </div>
  </div>
</div>
```

[views] index.ejs

```
<div class="container">
  ...
  <%-include("common/modal_alert.ejs")%>
  <%-include("common/modal_confirm.ejs")%>
</div>
```

[views]-[posts] write.ejs

```
$(frm).on("submit", async function(e){
  ...
  if(title==" " || body=="") {
    $("#modal-alert .modal-body").html("제목 또는 내용을 입력하세요!");
    $("#modal-alert").modal("show");
  } else {
    ...
  }
});
```

[views]-[posts] update.ejs

```
$(frm).on("submit", function(e){
  e.preventDefault();
  $("#modal-confirm .modal-body").html("수정된 정보를 저장하실래요?");
  $("#modal-confirm").modal("show");
  $("#modal-confirm").on("click", ".btn-primary", async function() {
    ...
  });
});
```

## • 카카오 지도 API

- [내 애플리케이션]-[앱 키]-[JavaScript 키]를 home.jsp에 추가한다.

```
[src]-[main]-[webapp] home.jsp
```

```
<head>
...
<script type="text/javascript" src="//dapi.kakao.com/v2/maps/sdk.js?appkey=xxxxxx"></script>
</head>
```

```
[src]-[main]-[webapp]-[local] search.jsp
```

```
...
<!-- 지도 출력 -->
<div class="row">
  <div class="col">
    <div id="map" style="width:100%;height:400px;display:none;" class="card"></div>
  </div>
</div>
<script id="temp_local" type="text/x-handlebars-template">
  <table class="table">
    <tr>
      <td>{{id}}</td>
      <td class="place">{{place_name}}</td>
      <td>{{address_name}}</td>
      <td class="phone">{{phone}}</td>
      <td><button class="btn btn-success btn-sm" x="{{x}}" y="{{y}}">위치</button></td>
    </tr>
  </table>
</script>
<script>
  ...
  $("#div_local").on("click", ".btn-success", function() { //위치 버튼을 클릭한 경우
    $("#map").show();
    const x=$(this).attr("x");
    const y=$(this).attr("y");
    var container = document.getElementById('map');
    var options = {
      center: new kakao.maps.LatLng(y, x),
      level: 3
    };
    var map = new kakao.maps.Map(container, options);

    var markerPosition = new kakao.maps.LatLng(y, x); // 마커가 표시될 위치입니다
    var marker = new kakao.maps.Marker({ // 마커를 생성합니다
      position: markerPosition
    });
    marker.setMap(map); // 마커가 지도 위에 표시되도록 설정합니다

    let row = $(this).parent().parent();
    let place = row.find(".place").text();
    let phone = row.find(".phone").text();
    var str = "<div style='padding:5px;font-size:12px;'>" + place + "<br>" + phone + "</div>";
    var info=new kakao.maps.InfoWindow({ content:str });
    kakao.maps.event.addListener(marker, "mouseover", function() {
      info.open(map, marker);
    });
    kakao.maps.event.addListener(marker, "mouseout", function() {
      info.close(map, marker);
    });
  });
</script>
```