

# 8

2024 GETIT 5기 SW 교육 8차시

## 프론트와 함께 작업하기



GETIT 5기 운영진 김근찬



## 참고사항

해당 강의자료는 **macOS와 window**를 기준으로 제작되었습니다. 추가적으로 영상 설명은 없습니다.

**도움**이 필요하거나 **질의사항**이 있으시다면  
GET-IT 5기 질문방을 이용해주세요!

해당 강의자료는 [ 조현영-Node.js 교과서 ]를 참고하여 제작되었습니다.  
**5기 부원 외의 강의자료는 원칙적으로 금지합니다.**



## 목차

1. **ip주소로** 포트 열기
2. **api** 만들기



01\_ip 주소로 포트 열기



02\_api 만들기

# 01

## ip 주소로 포트 열기



STEP

0

# 내 ip 주소 확인하기



"연결된 네트워크 설정을 눌러 IP주소 확인하기"

## STEP

## 0

## 내 ip 주소 확인하기

무선 LAN 어댑터 Wi-Fi:

연결별 DNS 접미사 . . . . .	:	
링크-로컬 IPv6 주소 . . . . .	:	fe80::86d0:d79e:49b:ff11%8
IPv4 주소 . . . . .	:	192.168.0.2
서브넷 마스크 . . . . .	:	255.255.255.0
기본 게이트웨이 . . . . .	:	192.168.0.1

**"터미널에서 ipconfig 입력 후  
IPv4 주소 확인하기"**



## STEP

# 1

확인된 ip 주소와 port 번호를 .env에 기입

**"HOST에는 ip 주소  
HTTP\_PORT에는 아래 중 하나 사용"**

HOST = 해당 ip 주소

HTTP\_PORT = 8080, 8081, 3000, 5000, 8000, 8888, 8880, 8001, 3001, 5001 중 하나 사용하기



STEP

2

## app.js에 server 열도록 수정하기

```
//port info
const dotenv = require('dotenv');
dotenv.config();
const HTTP_PORT = process.env.HTTP_PORT;
const HOST = process.env.HOST;

//make express server
const express = require('express');
const cors = require('cors');
const app = express();
app.use(express.json());
app.use(cors());
const { sequelize } = require('./models');
sequelize.sync();

module.exports = { app };

//check server is running
app.listen(HTTP_PORT, HOST, () => {
  console.log(`server is on http://${HOST}:${HTTP_PORT}`);
});
```





**프론트로부터 요청을 받을 수 있는  
서버를 만들었습니다.**

**다음 단계로 요청을 처리하는 작업을 진행하겠습니다.**



01\_ip 주소로 포트 열기

02\_api 만들기

# 02

## api 만들기



## STEP

# 0

api 폴더 만들고 해당 폴더에 post 폴더 만들기  
post 폴더 안에 postController.js & postService.js 만들기

✓ api / post

JS postController.js






JS postService.js



STEP

1

## 프론트로부터 요청 정리하기

⌵ Domain	⌵ HttpMethod	Aa API NAME	🔗 URL
AUTH	GET	 <u>getAllPosts</u>	/getAll
AUTH	GET	 <u>getPost</u>	/getOne
AUTH	POST	 <u>updatePost</u>	/postUpdate
AUTH	POST	 <u>createPost</u>	/postCreate
AUTH	POST	 <u>deletePost</u>	/postDelete



**STEP**

**1**

해당 qr코드를 통해 더 자세히 정리된  
api 표를 볼 수 있습니다.





## STEP

## 2

## createPost 구현하기

```
1  const express = require('express');
2  const router = express.Router();
3  const postService = require('./postService');
4
5  const createPost = async(req, res, next) => {
6    try {
7      const { title, description } = req.body;
8      console.log(`createPost - title: ${title}, description: ${description}`);
9      await postService.create(title, description);
10     res.status(200).json({title, description});
11   } catch (err) {
12     res.status(404);
13     next(err);
14   }
```

"post 요청은 req.body에서 데이터를 받을 수 있음  
해당 데이터를 postService에 있는 create 호출할 때 보냄  
성공은 200, 실패는 404 처리"



STEP

2

## createPost 구현하기

```
1  const { where } = require('sequelize');
2  const { Post } = require('../models');
3
4  const create = async (title, description) => {
5      try {
6          const post = await Post.create({
7              title: title,
8              description: description
9          });
10     } catch (err) {
11         console.error("postService.create error");
12         throw err;
13     }
14 }
```

"게시글을 만들어주기위해 디비에 저장하기"



# getPost 구현하기

STEP

3

```
16  const getPost = async(req, res, next) => {
17      try {
18          const id = parseInt(req.params.id); // URL 매개변수에서 id 가져오기
19          console.log(`getPost - postId: ${id}`);
20          const post = await postService.getOne(id);
21          res.status(200).json(post);
22      } catch (err) {
23          res.status(404);
24          next(err);
25      }
26  };
27  router.get('/getOne/:id', getPost);
28  router.post('/postCreate', createPost);
29  module.exports = router;
```

"get 요청은 req.body로 데이터를 보내는게 아니라  
주소를 통해 받을 수 있음,  
받은 데이터를 postService에 있는 getOne을 호출할 때 보냄"





## getPost 구현하기

STEP

3

```
15  const getOne = async (postId) => {
16      try {
17          const post = await Post.findOne({
18              where: {
19                  postId: postId
20              }
21          });
22          console.log(post);
23          return post;
24      } catch (err) {
25          console.error("postService.getOne error");
26          throw err;
27      }
28  };
29  module.exports = {
30      getOne,
31      create
32  };
33
```

"전달받은 postId 데이터로 원하는 게시물 조회  
조회된 데이터를 반환"



STEP

4

app.js에 router 등록하기

**"app.js에 등록해야 요청을 처리할 수 있음"**



STEP

4

```
const dotenv = require('dotenv');
dotenv.config();
const HTTP_PORT = process.env.HTTP_PORT;
const HOST = process.env.HOST;

const express = require('express');
const cors = require('cors');
const app = express();
app.use(express.json());
app.use(cors());
const { sequelize } = require('./models');
sequelize.sync();

module.exports = { app };

const postController = require('./api/post/postController');
app.use('/', postController);

app.listen(HTTP_PORT, HOST, () => {
  console.log(`server is on http://${HOST}:${HTTP_PORT}`);
});
```



**총 5개 요청 중에 2개를 같이 만들어 보았습니다.  
남은 3개를 스스로 만들어보시고  
프론트와 같이 요청 처리가 잘되는지  
확인해봅시다 :)**