Seoul woman up

스타트업 개발자와 함께 공부하는

Node.js

11. JWT와 인증

강의 내용은 강사가 별도로 명시하지 않는 한 비공개로 간주합니다. 녹음이나 사진 촬영를 허락하지 않으며 콘텐츠를 블로그, SNS 등에 게시하거나 공개적으로 공유하지 마세요.

콘텐츠 공유 가능 여부에 대해 궁금한 점이 있는 경우 강사에게 문의하시기 바랍니다.

목차

Seoul woman up

인증 소개

RESTful API 인증

인증 방식

- 기본 인증 (Basic Authentication)
- 토큰 기반 인증 (Token-based Authentication)
- OAuth 2.0
- API 키 인증 (API Key Authentication)

RESTful API 인증

기본 인증

- 가장 간단한 인증 방식
- 클라이언트가 인증을 위해 사용자 이름과 비밀번호 인코딩
- 동작 원리

1. 클라이언트가 HTTP 요청 헤더에 Authorization 필드 추가
2. Authorization 필드 값에 Basic 과 Base64 인코딩 된 username:password 형식의 문자열 전송
3. 서버는 Base 64 디코딩 후에 인증 처리

```
Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=
```

RESTful API 인증

토큰 기반 인증

- 클라이언트가 서버에서 발급받은 토큰을 이용하여 인증
- 토큰은 주로 JSON Web Token(JWT) 형식
- 동작 방식

1. 클라이언트는 로그인 요청을 보내고, 서버는 유용한 사용자임을 확인 후에 토큰 발급
2. 발급된 토큰은 클라이언트가 저장, 인증이 필요할 때 마다 Authorization 헤더로 전송
3. 서버는 토큰 추출 후 유효성 검증

```
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

RESTful API 인증

OAuth 2.0

- 인증 및 권한 부여 프레임워크
- 클라이언트가 다른 서비스의 리소스에 접근할 수 있는 권한을 부여하는 방식
- 주로 소셜 로그인, 서드 파티 API 접근 등에 사용
- 동작 방식

1. 클라이언트는 OAuth 2.0 프로바이더에게 인증 요청
2. 프로바이더는 인증을 처리하고, 클라이언트에게 액세스 토큰 발급
3. 클라이언트는 액세스 토큰을 사용하여 보로 리소스에 접근, 서버는 토큰 유효성 검증

RESTful API 인증

API 키 인증

- 공개 비공개 키를 사용하여 API 요청
- 주로 서드 파티 애플리케이션에서 사용
- 클라이언트가 API 키를 헤더나 쿼리 매개변수로 전송
- 동작 방식

1. 클라이언트는 API 요청 시 HTTP 헤더나 쿼리 매개변수에 API 키 포함
2. 서버는 받은 API 키 검증, 유효한 경우에 요청 처리

Seoul woman up

토큰 인증

토큰 인증

JWT

❑ 구성요소

1. Header(헤더) : 토큰 유형, 해싱 알고리즘
2. Payload(페이로드) : 클레임(claim) 저장
    1. 등록된 클레임
    2. 공개 클레임
    3. 비공개 클레임
3. Signature(서명) : 헤더와 페이로드 인코딩 후 비밀 키 사용 서명

https://jwt.io/

토큰 인증

Access Token

- Access Token 은 클라이언트가 보호된 리소스에 접근하기 위한 권한을 부여 받음
- 보호된 API 엔드 포인트에 접근할 때 사용
- 구성 요소

1. Payload : 클라이언트 식별 정보, 권한 스코프, 토큰 만료시간
2. 유효 기간 : 짧게 유지
3. 용도 : 보호된 API 엔드 포인트 접근

토큰 인증

Refresh Token

- Access Token을 재발급 받기 위한 특별한 토큰
- Access Token 보다 더 긴 유효 기간을 가짐
- 새로운 Access Token 을 발급 받을 수 있음
- 구성 요소

1. Payload : 클라이언트 식별 정보, 토큰 만료 시간
2. 유효 기간 : Access Token 보다 김
3. 용도 : Access Token 이 만료되었을 때 새로운 Access Token 발급

Seoul woman up

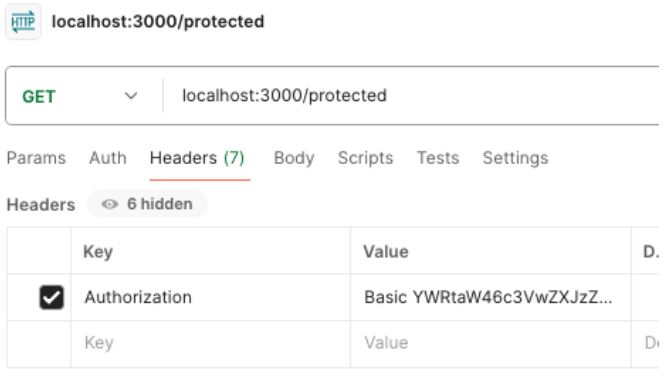
Express 인증

Express 인증

기본 인증

```
1  const express = require('express');
2  const basicAuth = require('express-basic-auth');
3
4  const app = express();
5
6  // 기본 인증 미들웨어 설정
7  app.use(basicAuth({
8    users: { 'admin': 'supersecret' }, // 사용자 이름과 비밀번호 설정
9    unauthorizedResponse: '인증이 실패했습니다.'
10 }));
11
12 // 보호된 API 엔드포인트
13 app.get('/protected', (req, res) => {
14   res.json({ message: '기본 인증으로 보호된 데이터에 접근하였습니다.' });
15 });
16
17 // 서버 시작
18 app.listen(3000, () => {
19   console.log('서버가 http://localhost:3000 포트에서 실행 중입니다.');
20 });
```

- ❑ [C]-[nodejs]-[project]-[11]-[ch11_01]
- ❑ 프로젝트 생성
- ❑ npm install express express-basic-auth

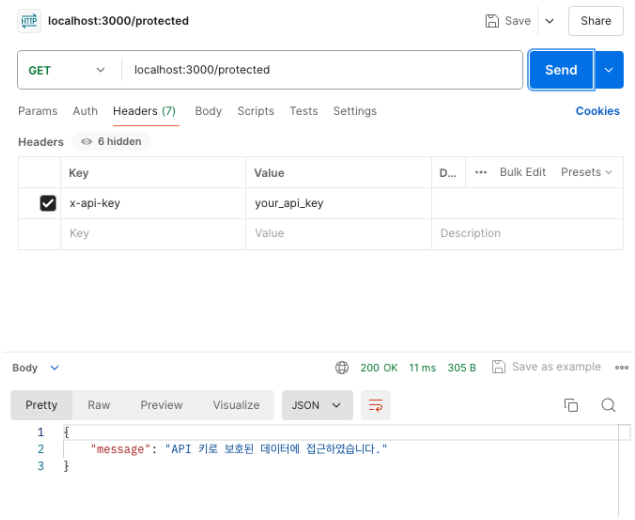


Express 인증

API 키 인증

```
1  const express = require('express');
2
3  const app = express();
4  const apiKey = 'your_api_key';
5
6  // API 키 검증 미들웨어
7  function authenticateApiKey(req, res, next) {
8    const providedKey = req.headers['x-api-key'] || req.query.api_key;
9
10   if (!providedKey || providedKey !== apiKey) {
11     return res.status(401).json({ error: '유효하지 않은 API 키' });
12   }
13
14   next();
15 }
16
17 // 보호된 라우트
18 app.get('/protected', authenticateApiKey, (req, res) => {
19   res.json({ message: 'API 키로 보호된 데이터에 접근하였습니다.' });
20 });
21
22 // 서버 시작
23 app.listen(3000, () => {
24   console.log('서버가 http://localhost:3000 포트에서 실행 중입니다.');
25 });
```

- ❑ [C]-[nodejs]-[project]-[11]-[ch11_01]
- ❑ 프로젝트 생성
- ❑ npm install express

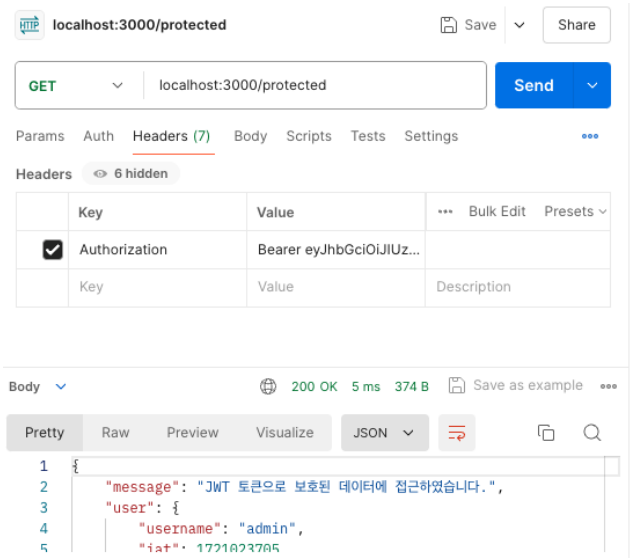


Express 인증

JWT 인증

```
1  const express = require('express');
2  const jwt = require('jsonwebtoken');
3
4  const app = express();
5  const secretKey = 'your_secret_key';
6  app.use(express.urlencoded({extended: true}));
7
8  // 토큰 발급하는 라우트 (로그인)
9  app.post('/login', (req, res) => {
10   const { username, password } = req.body;
11
12   // 실제로는 데이터베이스에서 사용자 인증을 수행합니다.
13   if (username === 'admin' && password === 'password') {
14     const accessToken = jwt.sign({ username }, secretKey, { expiresIn: '30m' });
15     res.json({ accessToken });
16   } else {
17     res.status(401).json({ error: '인증 실패' });
18   }
19 });
20
21 // JWT 토큰 검증 미들웨어
22 function authenticateToken(req, res, next) {
23   const authHeader = req.headers['authorization'];
24   const token = authHeader && authHeader.split(' ')[1]; // Bearer token 형식에서 토큰 추출
25
26   if (!token) {
27     return res.status(401).json({ error: '인증되지 않은 사용자' });
28   }
29
30   jwt.verify(token, secretKey, (err, user) => {
31     if (err) {
32       return res.status(403).json({ error: '토큰 만료 또는 유효하지 않음' });
33     }
34     req.user = user;
35     next();
36   });
```

- [C]-[nodejs]-[project]-[11]-[ch11_01]
- 프로젝트 생성
- `npm install express jsonwebtoken`

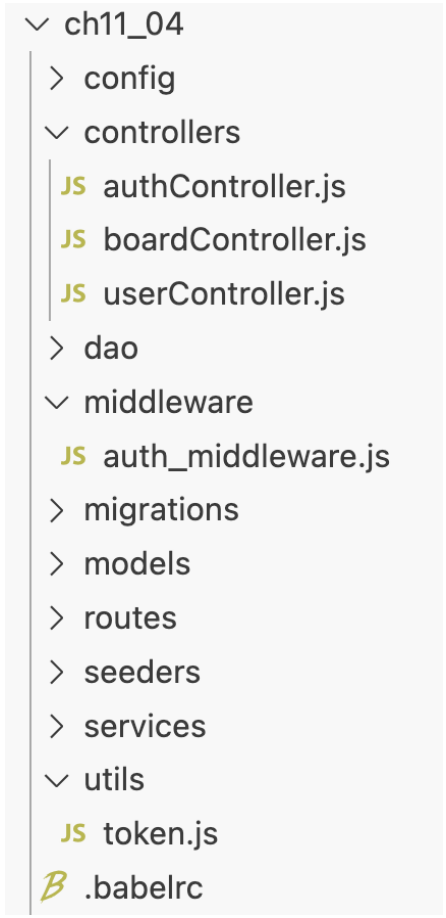


Seoul woman up

게시판 API 인증

게시판 API 인증

프로젝트 생성 및 구조



- [C]-[nodejs]-[project]-[11]-[ch11_04]
- 프로젝트 생성
- `npm install express jsonwebtoken nodemon pg sequelize sequelize-cli jest bcryptjs`

게시판 API 인증

Board 모델

```
1  module.exports = (sequelize, DataTypes) => {
2      const Board = sequelize.define('Board', {
3          id: {
4              type: DataTypes.INTEGER,
5              allowNull: false,
6              primaryKey: true,
7              autoIncrement: true,
8          },
9          title: DataTypes.STRING,
10         content: DataTypes.STRING,
11     }, {
12         tableName: 'board',
13         underscore: true,
14     });
15     Board.associate = function(models){
16         models.Board.belongsTo(models.User);
17     }
18     return Board;
19 };
```

- models/board.js
- 외래 키 관계 추가

게시판 API 인증

토큰 발급

```
1  const jwt = require('jsonwebtoken');
2
3  const generateAccessToken = (user) => {
4      return jwt.sign({
5          id: user.id,
6          email: user.email,
7      }, 'access_secret', {expiresIn: '15m'});
8  };
9
10 const generateRefreshToken = (user) => {
11     return jwt.sign({
12         id: user.id,
13         email: user.email,
14     }, 'refresh_secret', {expiresIn: '14d'});
15 };
16
17 module.exports = {
18     generateAccessToken,
19     generateRefreshToken,
20 }
```

- utils/token.js
- 액세스 토큰
- 리프레시 토큰

게시판 API 인증

미들웨어 작성

```
1  const jwt = require('jsonwebtoken');
2
3  const authenticateToken = (req, res, next) => {
4      let token;
5      if(req.headers.authorization){
6          token = req.headers.authorization.split(' ')[1];
7      }
8      // console.log(`token => ${req.headers.authorization}`)
9      if(!token) return res.sendStatus(401);
10
11     jwt.verify(token, 'access_secret', (err, user) => {
12
13         if (err) return res.sendStatus(404);
14         req.user = user;
15         next();
16     });
17 };
18
19 module.exports = {
20     authenticateToken,
21 };
```

- ❑ middleware/auth_middleware.js
- ❑ request 별 액세스 토큰 검증

게시판 API 인증

회원가입

```
6  const register = async (req, res) => {
7      const {email, name, password } = req.body;
8      const hashedPassword = await bcrypt.hash(password, 10);
9
10     try{
11         const user = await userService.createUser({
12             email: email, name: name, password: hashedPassword,
13         });
14         res.status(201).json(user)
15     }catch(error) {
16         res.status(500).json({error: error.message});
17     }
18 };
```

- routes/authController.js
- 회원가입

게시판 API 인증

로그인

```
20 const login = async (req, res) => {
21     const {email, password} = req.body;
22 
23     try{
24         const user = await userService.getUserByEmail(email);
25         if(!user) {
26             return res.status(400).json({
27                 message: 'Invalid email and password'
28             });
29         }
30         const isMatch = await bcrypt.compare(password, user.password);
31         if(!isMatch) {
32             return res.status(400).json({
33                 message: 'Invalid email and password'
34             });
35         }
36 
37         const accessToken = generateAccessToken(user);
38         const refreshToken = generateRefreshToken(user);
39 
40         res.json({
41             accessToken, refreshToken
42         })
43     }catch(error) {
44         res.status(500).json({error: error.message});
45     }
46 };
```

- routes/authController.js
- 액세스 토큰 및 리프레시 토큰 발급

게시판 API 인증

액세스 토큰 재발급

```
48 const refresh = (req, res) => {
49     const { token } = req.body;
50     if(!token) return res.sendStatus(401);
51 
52     jwt.verify(token, 'refresh_secret', (err,user) => {
53         if (err) return res.sendStatus(403);
54 
55         const accessToken = generateAccessToken(user);
56         res.json({
57             accessToken,
58         })
59     });
60 }
```

- routes/authController.js
- 액세스 토큰 재발급

게시판 API 인증

액세스 토큰으로 자원 보호

```
1  const express = require('express');
2  const boardController = require('../controllers/boardController');
3  const { authenticateToken } = require('../middleware/auth_middleware')
4
5  const  router = express.Router();
6
7  router.post('/', authenticateToken, boardController.createBoard);
8  router.get('/:id', boardController.getBoardById);
9  router.get('/', boardController.getAllBoards);
10 router.put('/:id', boardController.updateBoard);
11 router.delete('/:id', boardController.deleteBoard);
12
13 module.exports = router;
```

- routes/boardRoutes.js
- 요청에 미들웨어 추가

Seoul woman up

백문이 불여일타