


```
@ApiTags('Auth')
```

```
@Controller('auth')
```

```
export class AuthController {
```

```
  constructor(private authService: AuthService) {}
```

```
  @ApiOperation({
```

```
    summary: '카카오 로그인',
```

```
    description: '카카오 로그인을 하는 API입니다.',
```

```
  })
```

```
  @Post('kakao')
```

```
  async kakaoLogin(@Body() body, @Res() res) {
```

```
    try {
```

```
      const { payload } = body;
```

```
      console.log(body);
```

```
      if (!payload) {
```

```
        throw new BadRequestException('카카오정보가 없습니다.');
```

```
      }
```

```
      const kakao = await this.authService.kakaoLogin(payload);
```

```
      console.log('카카오 컨트롤러', kakao);
```

```
      const token = await this.authService.kakaoUser(kakao);
```

```
      const { AccessToken, RefreshToken } = token;
```

```
      res.setHeader('Authorization', AccessToken);
```

```
      res.setHeader('RefreshToken', RefreshToken);
```

```
      console.log(token);
```

```
      if (!kakao.id) {
```

```
        throw new BadRequestException('카카오 정보가 없습니다.');
```

```
      }
```

```
      res.send();
```

```
    } catch (e) {
```

```
      console.log(e);
```

```
      throw new UnauthorizedException();
```

```
    }
```

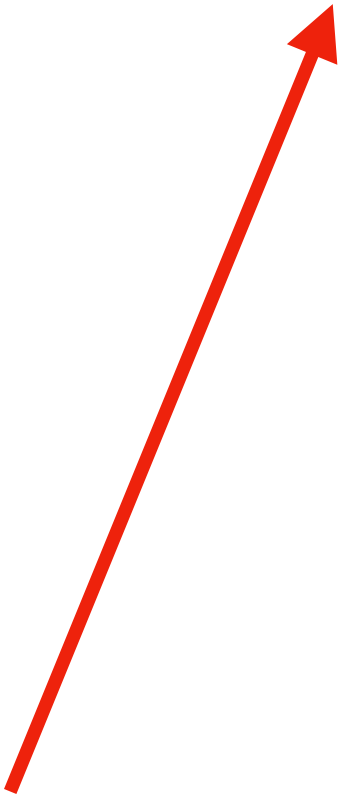
```
  }
```

```
async kakaoLogin(payload: string) {
  const code = payload;
  const kakaoKey = '1b6507f790effacebec0df34314f133'; // 이부분은 REST_API KEY
  const kakaoTokenUrl = 'https://kauth.kakao.com/oauth/token';
  const kakaoUserInfoUrl = 'https://kapi.kakao.com/v2/user/me';
  const body = {
    grant_type: 'authorization_code',
    client_id: kakaoKey,
    redirect_uri: 'http://localhost:3000/oauth',
    code,
  };
  const headers = {
    'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8',
  };
  try {
    const response = await axios({
      method: 'POST',
      url: kakaoTokenUrl,
      timeout: 30000,
      headers,
      data: qs.stringify(body),
    });
    if (response.status === 200) {
      const headerUserInfo = {
        'Content-Type': 'application/x-www-form-urlencoded;charset=utf-8',
        Authorization: 'Bearer ' + response.data.access_token,
      };
      const responseUserInfo = await axios({
        method: 'GET',
        url: kakaoUserInfoUrl,
        timeout: 30000,
        headers: headerUserInfo,
      });
      if (responseUserInfo.status === 200) {
        console.log('리스폰스 서비스 유저인포', responseUserInfo);
        return responseUserInfo.data;
      } else {
        throw new UnauthorizedException();
      }
    } else {
      throw new UnauthorizedException();
    }
  } catch (error) {
    throw new UnauthorizedException();
  }
}
```

```
async kakaoUser(kakao: KakaoDataDto) {  
  const { id, properties, kakao_account } = kakao;  
  const { email } = kakao_account;  
  const kakaoUser = new UserEntity();  
  kakaoUser.password = String(id);  
  kakaoUser.name = 'kakao';  
  kakaoUser.email = email;  
  kakaoUser.nickname = email.split('@')[0];  
  kakaoUser.profileImg = properties.profile_image  
    ? properties.profile_image  
    : process.env.DEFAULT_IMG_URL;  
  
  const existUser: UserEntity = await this.userRepository.findOneBy({  
    email,  
  });  
  
  if (!existUser) {  
    console.log('신규유저', kakaoUser);  
    const newUser = await this.userRepository.insert(kakaoUser);  
    const tokenId: number = newUser.identifiers[0].id;  
    const token = await this.createToken({ tokenId });  
    return token;  
  }  
  
  const ExistUser = existUser;  
  console.log('기존유저', existUser);  
  const tokenId: number = ExistUser.id;  
  const token = await this.createToken({ tokenId });  
  return token;  
}
```



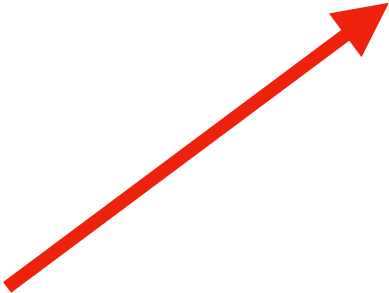
```
async createToken(req: string | object): Promise<TokenType> {  
  const payload = req;  
  
  const accessToken = this.jwtService.sign(payload, {  
    expiresIn: '10m',  
    secret: process.env.JWT_SECRET,  
  });  
  
  const refreshToken = this.jwtService.sign(payload, {  
    expiresIn: '7d',  
    secret: process.env.JWT_SECRET,  
  });  
  
  return {  
    AccessToken: `Bearer ${accessToken}`,  
    RefreshToken: `Bearer ${refreshToken}`,  
  };  
}
```



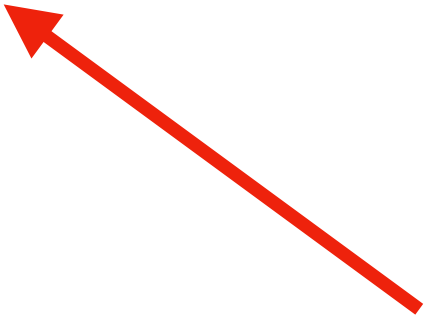




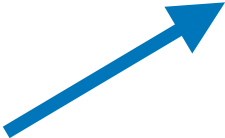










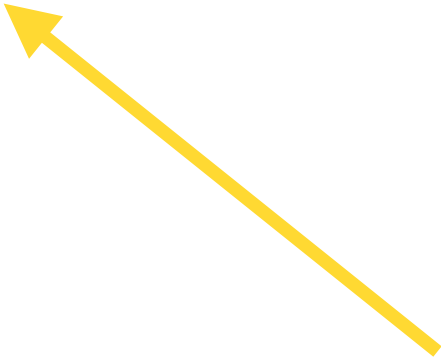
























TS

JS



Error

Social Login Process



Front-End

```

@ApiOperation({
  summary: '카카오 로그인',
  description: '카카오 로그인을 하는 API입니다.',
})
@Post('kakao')
async kakaoLogin(@Body() kakao: KakaoDataDto) {
  try {
    const { payload } = body;
    console.log(body);
    if (!payload) {
      throw new BadRequestException('카카오 정보가 없습니다.');
```



카카오에서 전달받은 데이터를 가공

인가코드

@Post (인가코드)
@kakao token 발급

```

async kakaoLogin(payload: string) {
  const code = payload;
  const kakaoKey = '1b6507f790effacebec0df34314f133'; //여부분은 REST API KEY
  const kakaoTokenUrl = 'https://kauth.kakao.com/oauth/token';
  const kakaoUserInfoUrl = 'https://kapi.kakao.com/v2/user/me';
  const body = {
    grant_type: 'authorization_code',
    client_id: kakaoKey,
    redirect_uri: 'http://localhost:3000/oauth',
    code,
  };
  const headers = {
    'Content-Type': 'application/x-www-form-urlencoded; charset=utf-8',
  };
  try {
    const response = await axios({
      method: 'POST',
      url: kakaoTokenUrl,
      timeout: 30000,
      headers,
      data: qs.stringify(body),
    });
    if (response.status === 200) {
      const headerUserInfo = {
        'Content-Type': 'application/x-www-form-urlencoded; charset=utf-8',
        Authorization: 'Bearer ' + response.data.access_token,
      };
      const responseUserInfo = await axios({
        method: 'GET',
        url: kakaoUserInfoUrl,
        timeout: 30000,
        headers: headerUserInfo,
      });
      if (responseUserInfo.status === 200) {
        console.log('리스폰스 서비스 유저인포', responseUserInfo);
        return responseUserInfo.data;
      } else {
        throw new UnauthorizedException();
      }
    } else {
      throw new UnauthorizedException();
    }
  } catch (error) {
    throw new UnauthorizedException();
  }
}
```

```


async kakaoUser(kakao: KakaoDataDto) {
  const { id, properties, kakao_account } = kakao;
  const { email } = kakao_account;
  const kakaoUser = new UserEntity();
  kakaoUser.password = String(id);
  kakaoUser.name = 'kakao';
  kakaoUser.email = email;
  kakaoUser.nickname = email.split('@')[0];
  kakaoUser.profileImg = properties.profile_image
    ? properties.profile_image
    : process.env.DEFAULT_IMG_URL;
  const existUser: UserEntity = await this.userRepository.findOneBy({
    email,
  });
  if (!existUser) {
    console.log('신규유저', kakaoUser);
    const newUser = await this.userRepository.insert(kakaoUser);
    const tokenId = newUser.identifiers[0].id;
    const token = await this.createToken({ tokenId });
    return token;
  }
  const ExistUser = existUser;
  console.log('기존유저', existUser);
  const tokenId = ExistUser.id;
  const token = await this.createToken({ tokenId });
  return token;
}

async createToken(req: string | object): Promise<TokenType> {
  const payload = req;
  const accessToken = this.jwtService.sign(payload, {
    expiresIn: '10m',
    secret: process.env.JWT_SECRET,
  });
  const refreshToken = this.jwtService.sign(payload, {
    expiresIn: '7d',
    secret: process.env.JWT_SECRET,
  });
  return {
    AccessToken: `Bearer ${accessToken}`,
    RefreshToken: `Bearer ${refreshToken}`,
  };
}
```

토큰 전달

Image-Upload

Trouble - S3 Multer

<input type="checkbox"/>	 1675039900037_스크린샷2023-01-16오후1.43.08.png	png	2023. 1. 30. am 9:51:43 AM KST	0B	Standard
--------------------------	---	-----	--------------------------------	----	----------