

CS 지식을 활용한

~~의식와 흐름대로 작성한~~

이메일 인증 개선



회원가입 인증 메일

본 이메일 인증은 한글과컴퓨터 회원가입을 위한
필수 사항입니다. 아래 [\[이메일 인증하기\]](#) 버튼을 클릭한 후
홈페이지에서 남은 회원가입 절차를 완료하여 주시기 바랍니다.

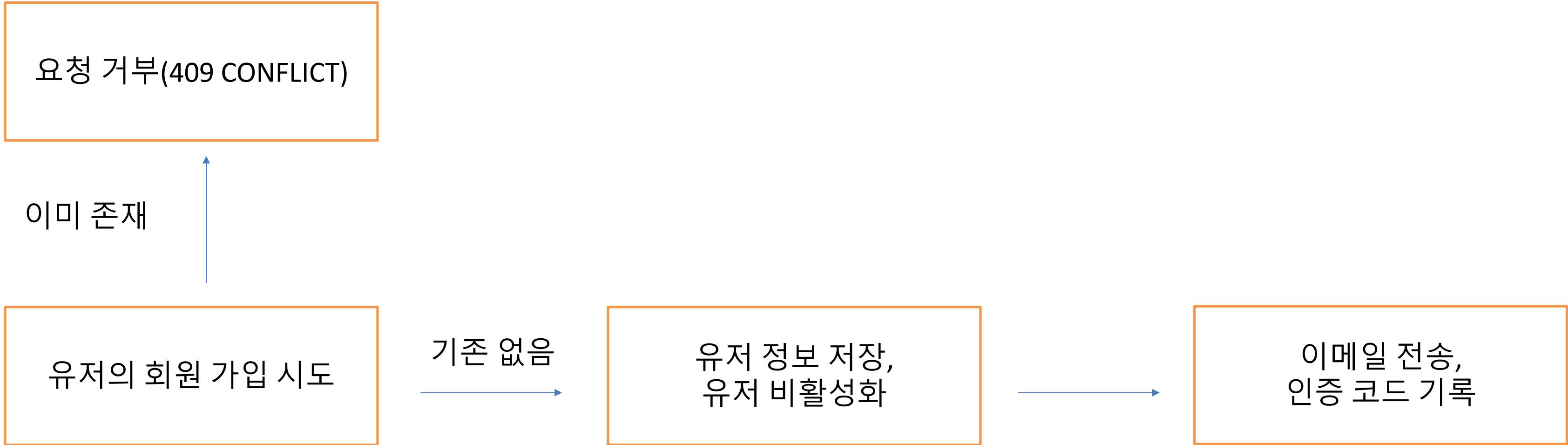
이메일 인증하기

본 메일은 발신전용 메일이므로 문의 및 회신하실 경우 답변되지 않습니다. 메일내용에 대해 궁금한
사항이 있으시면 고객센터로 문의하여 주시기 바랍니다.

(c)HANCOM INC. All Rights Reserved.

01

이메일 인증 구현 로직 플로우 및 설계 과정



01

이메일 인증 구현 로직 플로우 및 설계 과정



회원가입

계정 정보

이메일

업무용 이메일을 입력해 주세요.

비밀번호

영어 대소문자, 숫자, 특수문자 중 2종류 조합의 8-15자

비밀번호 확인

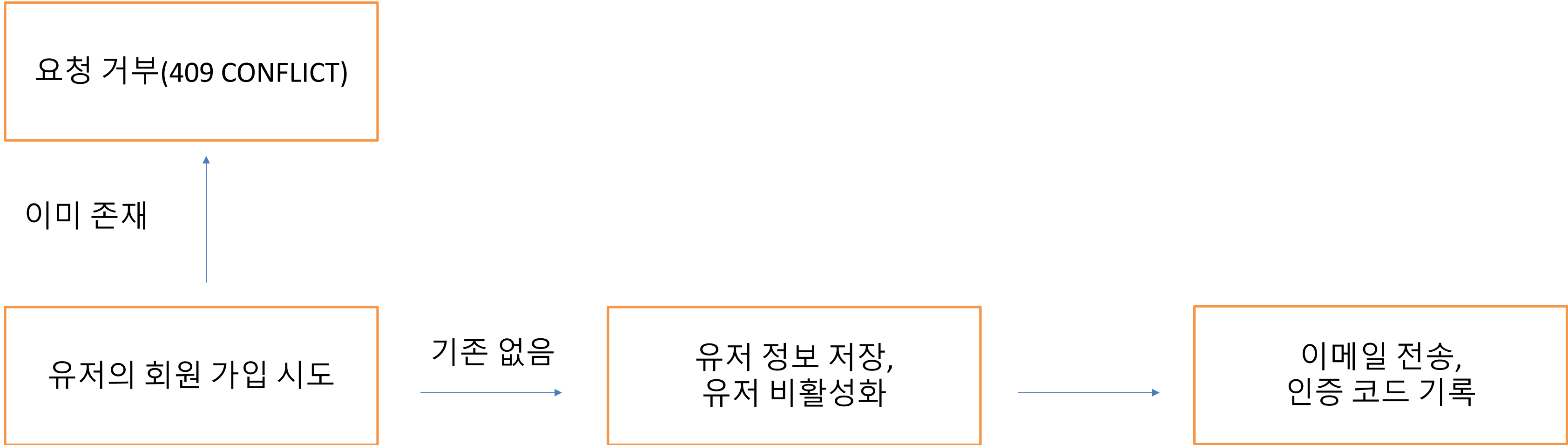
비밀번호를 다시 입력해 주세요.

이름

이름을 입력해 주세요.

01

이메일 인증 구현 로직 플로우 및 설계 과정



01

이메일 인증 구현 로직 플로우 및 설계 과정

인증코드 확인 안내 Inbox x



skkunion2024@gmail.com

to me ▼

<http://localhost:8080/auth/emailVerification/Ru4oY9faMm>

← Reply

→ Forward



HANCOM
한글과컴퓨터

회원가입 인증 메일

본 이메일 인증은 한글과컴퓨터 회원가입을 위한
필수 사항입니다. 아래 [\[이메일 인증하기\]](#) 버튼을 클릭한 후
홈페이지에서 남은 회원가입 절차를 완료하여 주시기 바랍니다.

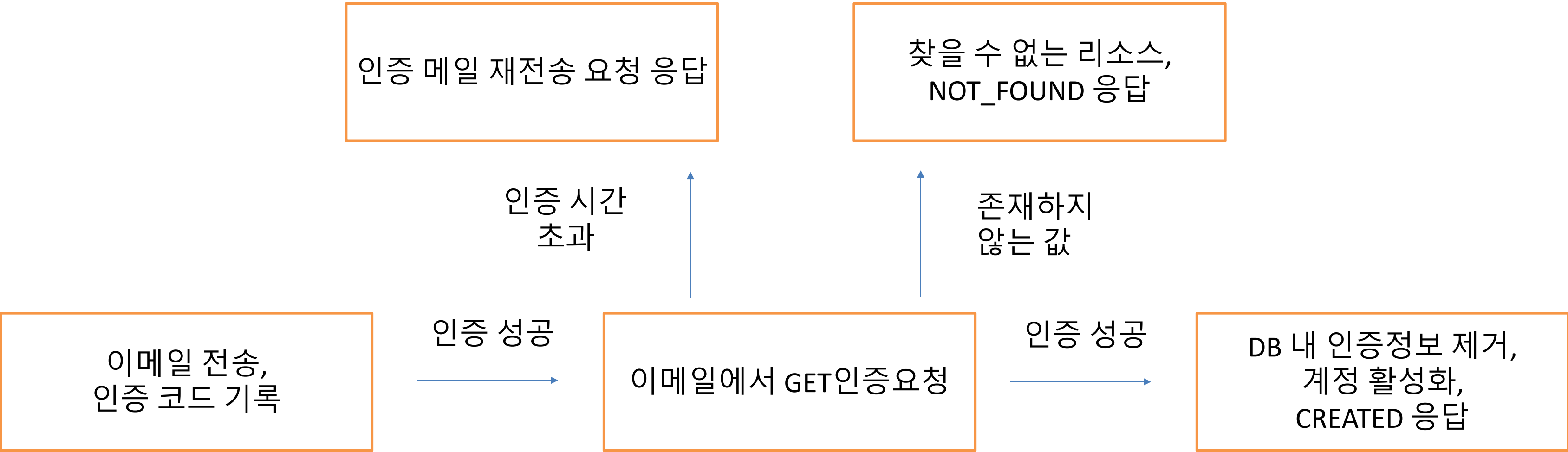
이메일 인증하기

본 메일은 발신전용 메일이므로 문의 및 회신하실 경우 답변되지 않습니다. 메일내용에 대해 궁금한
사항이 있으시면 고객센터로 문의하여 주시기 바랍니다.

(c)HANCOM INC. All Rights Reserved.

01

이메일 인증 구현 로직 플로우 및 설계 과정



02

코드 작성

```
@PostMapping(🌐"/account") 👤 qoraudrb
public ResponseEntity<createAccountResponse> createAccount(
    @RequestBody final createAccountRequest accountRequest
) {
    String nickname = accountRequest.nickname();
    String email = accountRequest.email();
    String password = accountRequest.password();

    accountServiceFacade.createAccountWithEmailVerification(nickname, email, password);
    return ResponseEntity.status(CREATED).body(new createAccountResponse(nickname, email));
}
```


02

코드 작성

```
@Transactional 1 usage 👤 qoraudrb *  
public void createAccountWithEmailVerification(String nickname, String email, String password)  
    if (memberService.isMemberExist(email))  
        throw new EntityExistsException(email);  
  
    memberService.joinMember(new Member(nickname, email, password));  
  
    String token = randomAlphanumeric(TOKEN_LENGTH);  
    emailVerificationService.sendEmailVerificationMessage(email, token);  
    emailVerificationService.createTemporaryEmailAuth(email, token);  
}
```

03

결과 확인

HTTP

http://localhost:8080/account

Save

Share

POST

http://localhost:8080/account

Cancel

Params

Authorization

Headers (9)

Body

Scripts

Tests

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

Beautify

1 {

2 "nickname": "qoraudrb123",

3 "email": "skkunion2024@gmail.com",

4 "password": "123456789"

5 }


Response

Sending request...

Could not send request

03

결과 확인

 http://localhost:8080/account

Save Share

POST http://localhost:8080/account Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```
1 {
2   ... "nickname": "qoraudrb123",
3   ... "email": "skkunion2024@gmail.com",
4   ... "password": "123456789"
5 }
```

Body Cookies Headers (5) Test Results 201 Created 4.70 s 228 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "nickname": "qoraudrb123",
3   "email": "skkunion2024@gmail.com"
4 }
```

03

결과 확인

Email_verification

| expired_at | id | token | email |
|----------------------------|----|------------|------------------------|
| 2024-08-29 11:53:50.230087 | 1 | W9yHG2JNUm | skkunion2024@gmail.com |

Member

| last_login_date | modified_at | nickname ▲ | password | email | status |
|----------------------------|----------------------------|-------------|-----------|------------------------|------------|
| 2024-08-29 11:43:45.984097 | 2024-08-29 11:43:45.984097 | qoraudrb123 | 123456789 | skkunion2024@gmail.com | UNVERIFIED |

03

결과 확인

인증코드 확인 안내

Inbox x



skkunion2024@gmail.com

to me ▼

<http://localhost:8080/auth/emailVerification/Ru4oY9faMm>

↩ Reply

➡ Forward




03

결과 확인

이게 최선입니까?
확실해요?

04

첫 번째 문제점

 http://localhost:8080/account

Save Share

POST http://localhost:8080/account Send

Params Authorization Headers (9) Body Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON Beautify

```
1 {
2   .... "nickname": "qoraudrb123",
3   .... "email": "skkunion2024@gmail.com",
4   .... "password": "123456789"
5 }
```

Body Cookies Headers (5) Test Results 201 Created 4.70 s 228 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "nickname": "qoraudrb123",
3   "email": "skkunion2024@gmail.com"
4 }
```

04

첫 번째 문제점

HTTP

http://localhost:8080/account

Save

Share

POST

http://localhost:8080/account

Send

Params

Authorization

Headers (9)

Body

Scripts

Tests

Settings

Cookies

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

Beautify

```
1 {
2   ... "nickname": "qoraudrb123",
3   ... "email": "skkunion2024@gmail.com",
4   ... "password": "123456789"
5 }
```

Body

Cookies

Headers (5)

Test Results

201 Created

4.70 s

228 B

Save as example

...

Pretty

Raw

Preview

Visualize

JSON

↺

📄

🔍

```
1 {
2   "nickname": "qoraudrb123",
3   "email": "skkunion2024@gmail.com"
4 }
```


04

첫 번째 문제점

HTTP `http://localhost:8080/account` Save Share

POST `http://localhost:8080/account` Send

Params Authorization Headers (9) **Body** Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** Beautify

```
1 {
2   ... "nickname": "qoraudrb123",
3   ... "email": "skkunion2024@gmail.com",
4   ... "password": "123456789"
5 }
```

4.70 s

심각한 응답속도 문제!

Body Cookies Headers (5) Test Results 201 Created **4.70 s** 228 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "nickname": "qoraudrb123",
3   "email": "skkunion2024@gmail.com"
4 }
```

04

첫 번째 문제점

```
@Transactional 1 usage 👤 qoraudrb *  
public void createAccountWithEmailVerification(String nickname, String email, String password)  
    if (memberService.isMemberExist(email))  
        throw new EntityExistsException(email);  
  
    memberService.joinMember(new Member(nickname, email, password)); ← 회원을 DB에 저장한다.(DB로직)  
  
    String token = randomAlphanumeric(TOKEN_LENGTH);  
    emailVerificationService.sendEmailVerificationMessage(email, token); ← 인증용 메일을 보낸다.(외부 로직)  
    emailVerificationService.createTemporaryEmailAuth(email, token); ← 인증용 정보를 저장한다.(DB 로직)  
}
```

04

첫 번째 문제점

@Transactional ← 병목 현상의 원인(2)

```
public void createAccountWithEmailVerification(String nickname, String email, String password)
    if (memberService.isMemberExist(email))
        throw new EntityExistsException(email);

    memberService.joinMember(new Member(nickname, email, password));

    String token = randomAlphanumeric(TOKEN_LENGTH);
    emailVerificationService.sendEmailVerificationMessage(email, token); ← 인증용 메일을 보낸다.(외부 로직)
    emailVerificationService.createTemporaryEmailAuth(email, token); ← 인증용 정보를 저장한다.(DB 로직)
}
```

병목 현상의 원인(1)



04

첫 번째 문제점

```
START TRANSACTION;
```

Member 테이블의 Member를 저장한다. (INSERT INTO VALUES(~~~))

4초 동안 테이블을 쉰다. Sleep(4000)

EmailVerification 테이블에 EmailVerification을 저장한다. (INSERT INTO VALUES(~~~))

```
COMMIT;
```

04

첫 번째 문제점의 해결방안

1. 처리 시작
 - DB connection 생성
 - transaction start
2. 사용자의 로그인 여부 확인
3. 사용자의 글쓰기 내용의 오류 여부 확인
4. 첨부로 업로드된 파일 확인 및 저장
5. 사용자의 입력 내용을 DBMS 에 저장
6. 첨부 file 정보를 DBMS에 저장
7. 저장된 내용 또는 기타 정보를 DBMS 에서 조회
8. 게시물 등록에 대한 알림 메일 발송
9. 알림 메일 발송 이력을 DBMS 에 저장
 - ← transaction commit
 - ← DB connection close
10. 처리 완료



1. 처리 시작
2. 사용자의 로그인 여부 확인
3. 사용자의 글쓰기 내용의 오류 발생 여부 확인
4. 첨부로 업로드된 파일 확인 및 저장
 - DB connection 생성
 - transaction start
5. 사용자의 입력 내용을 DBMS 에 저장
6. 첨부 파일 정보를 DBMS 에 저장
 - ← transaction commit (종료)
7. 저장된 내용 또는 기타 정보를 DBMS 에서 조회
8. 게시물 등록에 대한 알림 메일 발송
 - transaction start
9. 알림 메일 발송 이력을 DBMS에 저장
 - ← transaction commit
 - ← DB connection close
10. 처리 완료

04

첫 번째 문제점의 해결방안

1. 처리 시작

→ DB connection 생성

→ transaction start

2. 사용자의 로그인 여부 확인

3. 사용자의 글쓰기 내용의 오류 여부 확인

4. 첨부로 업로드된 파일 확인 및 저장

5. 사용자의 입력 내용을 DBMS 에 저장

6. 첨부 file 정보를 DBMS에 저장

7. 저장된 내용 또는 기타 정보를 DBMS 에서 조회

8. 게시물 등록에 대한 알림 메일 발송

9. 알림 메일 발송 이력을 DBMS 에 저장

← transaction commit

← DB connection close

10. 처리 완료

1. 처리 시작

2. 사용자의 로그인 여부 확인

3. 사용자의 글쓰기 내용의 오류 발생 여부 확인

4. 첨부로 업로드된 파일 확인 및 저장

→ DB connection 생성

→ transaction start

5. 사용자의 입력 내용을 DBMS 에 저장

6. 첨부 파일 정보를 DBMS 에 저장

← transaction commit (종료)

7. 저장된 내용 또는 기타 정보를 DBMS 에서 조회

8. 게시물 등록에 대한 알림 메일 발송

→ transaction start

9. 알림 메일 발송 이력을 DBMS에 저장

← transaction commit

← DB connection close

10. 처리 완료

04

첫 번째 문제점의 해결방안(트랜잭션의 분리)

장점 1. 데이터 베이스 커넥션 낭비를 줄일 수 있다.(원래의 목적)

단점일 수도 있는 장점 2. 뒷 트랜잭션이 일어나더라도 앞의 트랜잭션은 성공하면 데이터가 저장이 됨.

작업의 특성에 대한 고민

→ 과연 이메일 인증이라도 이게 중요한 걸까? 이 작업의 특성이 무엇일까?

→ 오류가 생겨서 이메일이 안 보내져도 비활성화된 회원 정보만 있으면 어차피 둘 다 복구가 가능함.

→ 이메일 안 보내지면 고객님이 화가 나시겠지만 비활성화된 회원으로 로그인하면 인증 이메일 다시 보내기 화면으로 보내 버리면 됨.

→ 결론: 저 범위의 트랜잭션을 풀어버리면 됨.

04

첫 번째 문제점의 해결방안(트랜잭션의 분리)

작업의 특성에 대한 고민(다른 경우)

- 이메일 말고 주문이라고 생각을 해봅시다. 이 작업의 특성이 무엇일까?
- 주문을 하고 결제를 하고 배송까지 한번에 해야되는 경우로 작업을 바꿔보면 이건 느리다고 트랜잭션을 유지하지 않으면 주문은 들어갔는데 결제가 안되거나 하는 문제가 발생해서 절대 이렇게 하면 안됨.
- 결론: 작업에 따라 다르지만 이메일 보내기는 중요한 일이 아니고 복구가 가능해서 트랜잭션을 분리해도 괜찮다.(작업의 특성에 관해서 고민해보는 게 중요하다!)

04

두 번째 문제점

HTTP `http://localhost:8080/account` Save Share

POST `http://localhost:8080/account` Send

Params Authorization Headers (9) **Body** Scripts Tests Settings Cookies

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** Beautify

```
1 {
2   ... "nickname": "qoraudrb123",
3   ... "email": "skkunion2024@gmail.com",
4   ... "password": "123456789"
5 }
```

4.70 s

심각한 응답속도 문제!

Body Cookies Headers (5) Test Results 201 Created **4.70 s** 228 B Save as example

Pretty Raw Preview Visualize JSON

```
1 {
2   "nickname": "qoraudrb123",
3   "email": "skkunion2024@gmail.com"
4 }
```

04

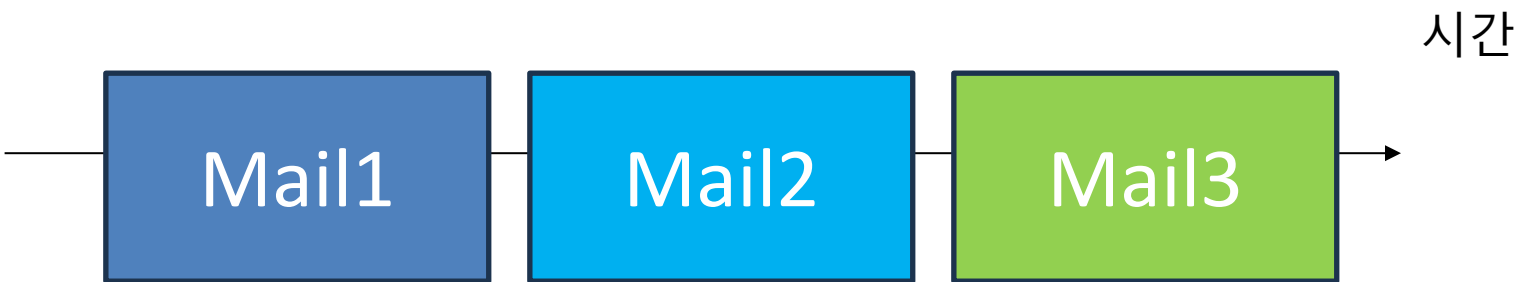
두 번째 문제점 관련 OS 지식

프로세스 vs 쓰레드

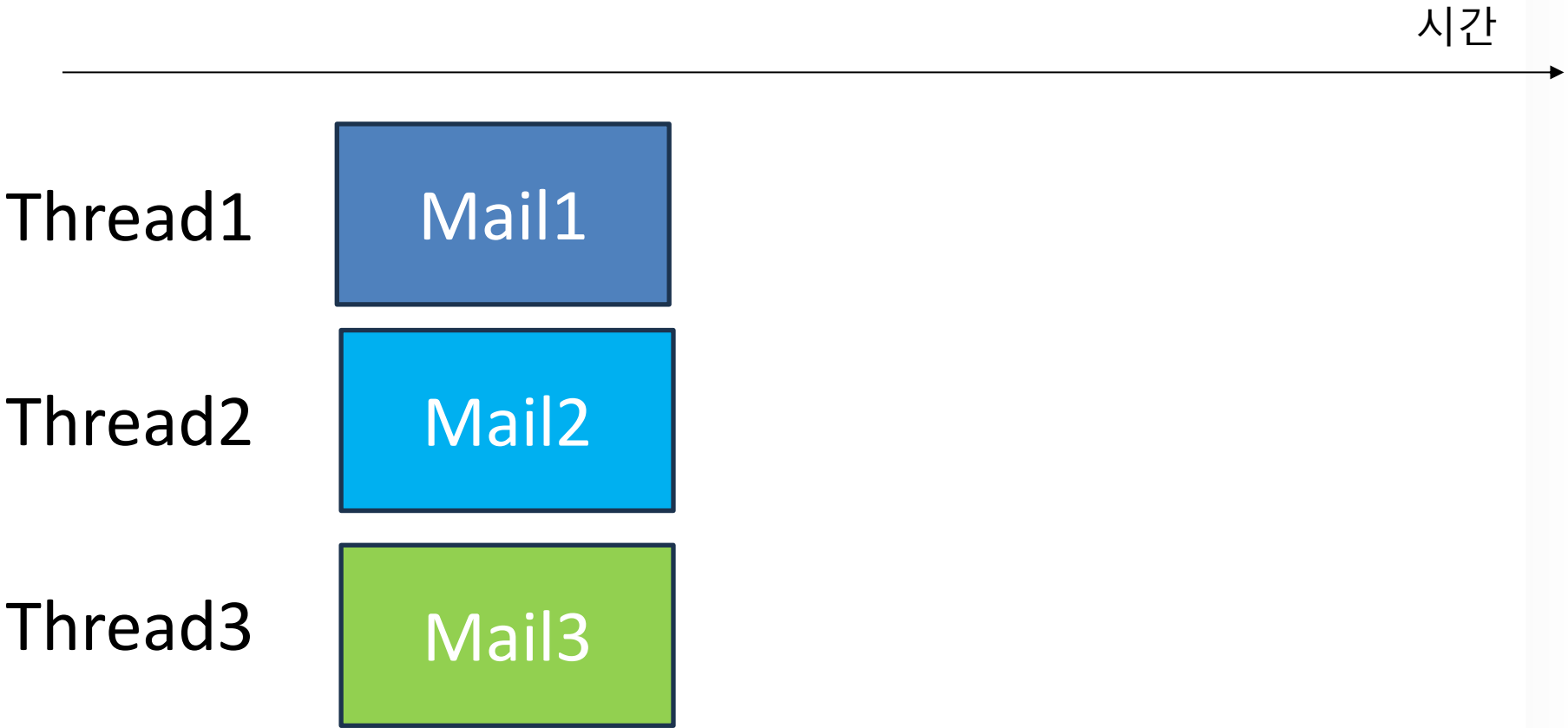
04

두 번째 문제점 관련 OS 지식

동기 처리



비동기 처리



04

두 번째 문제점 관련 OS 지식

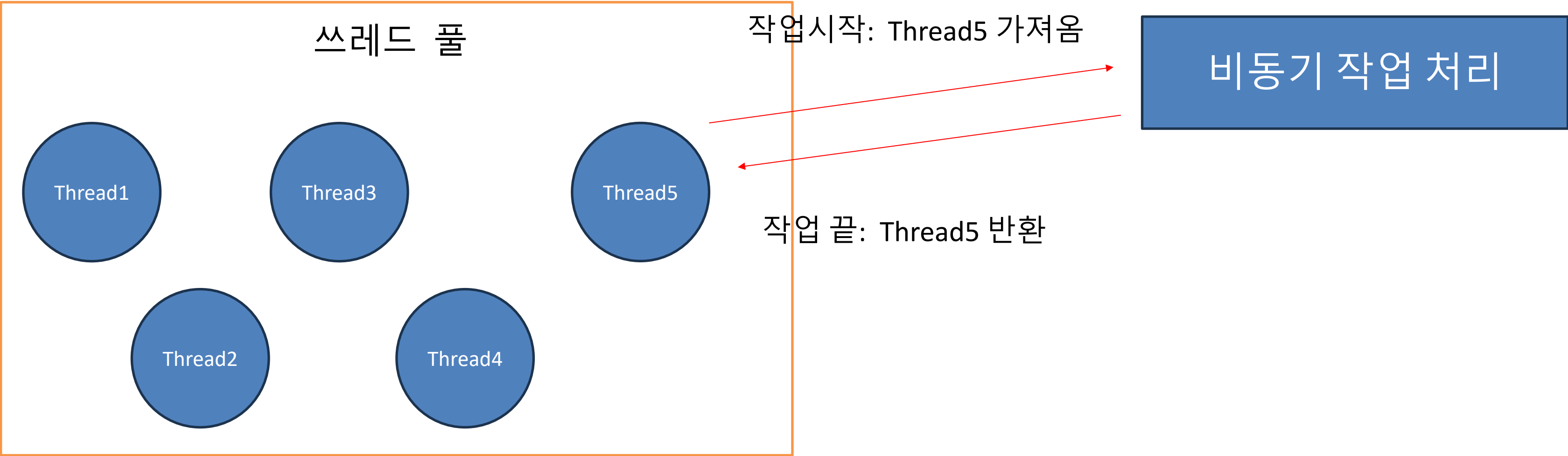
비동기 처리 방법1

```
public static void main(String[] args) throws InterruptedException {  
    while (true){ // 갑자기 엄청나게 많은 작업이라고 생각을 해주세요.  
        new Thread(() -> { // 굉장히 긴 작업이라고 생각을 해주세요.  
            int a = 0;  
            while(true){  
                a++;  
            }  
        }).start();  
    }  
}
```

04

두 번째 문제점 관련 OS 지식

비동기 처리 방법2

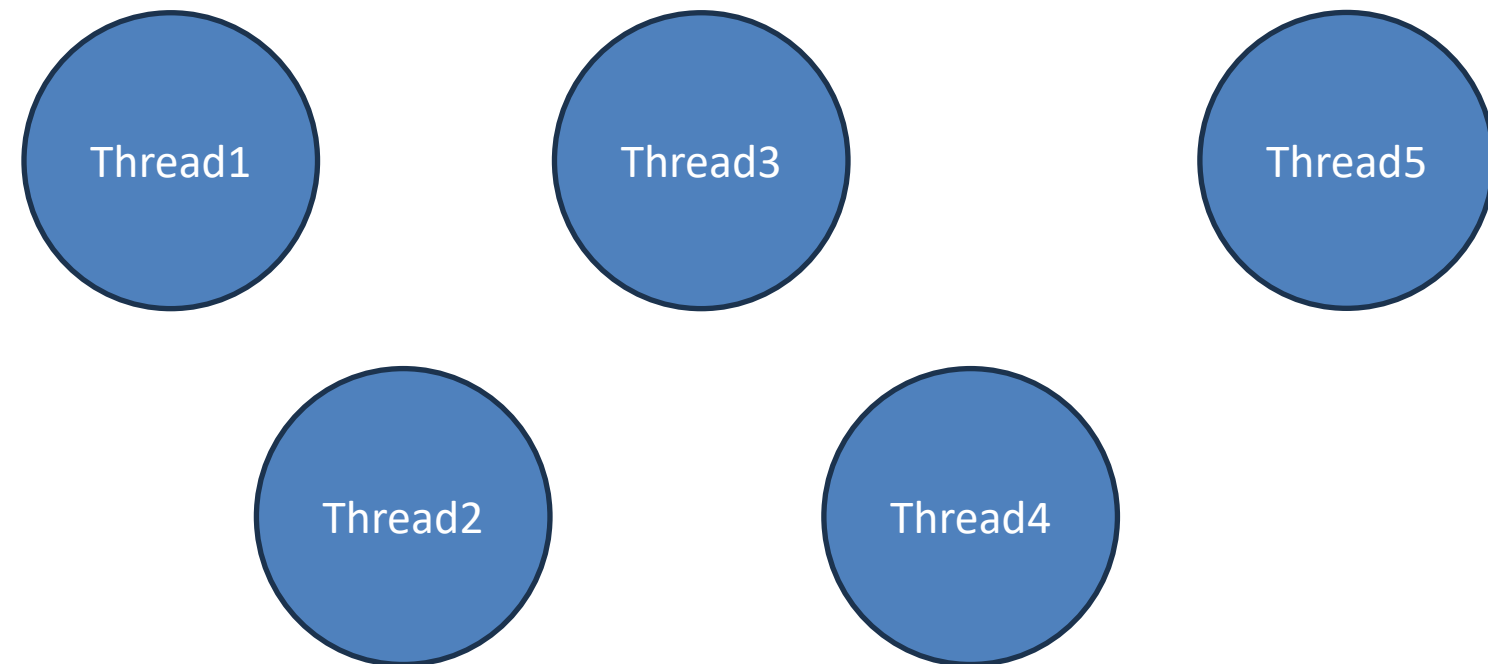


04

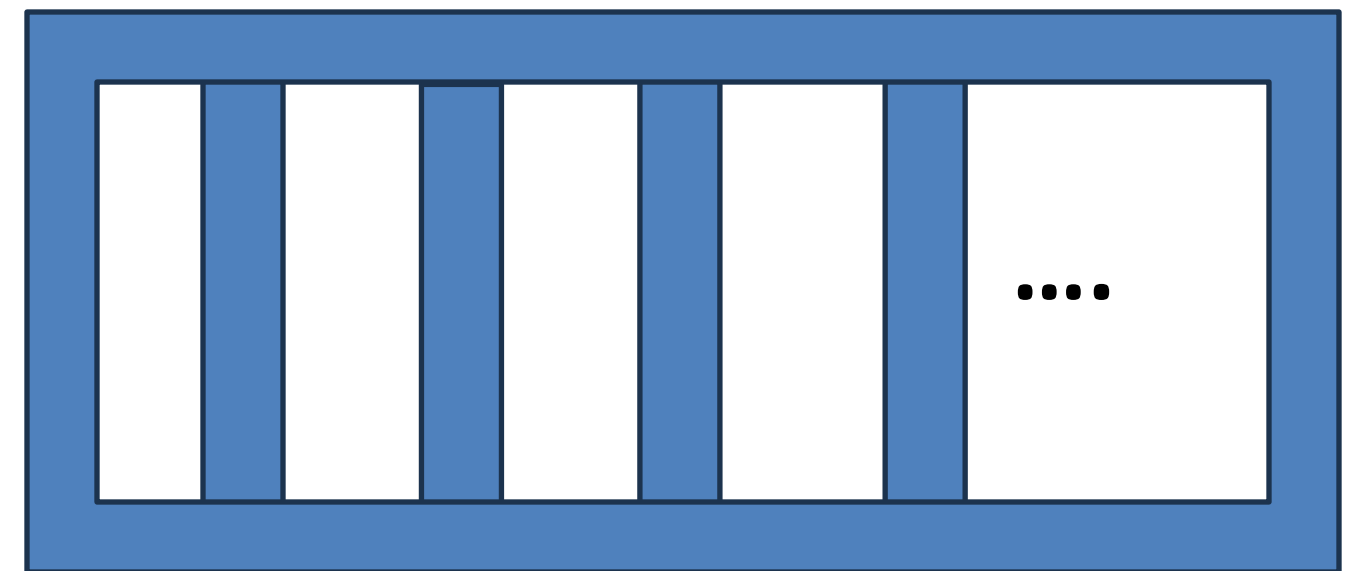
두 번째 문제점 관련 OS 지식

비동기 처리 방법2

쓰레드 풀



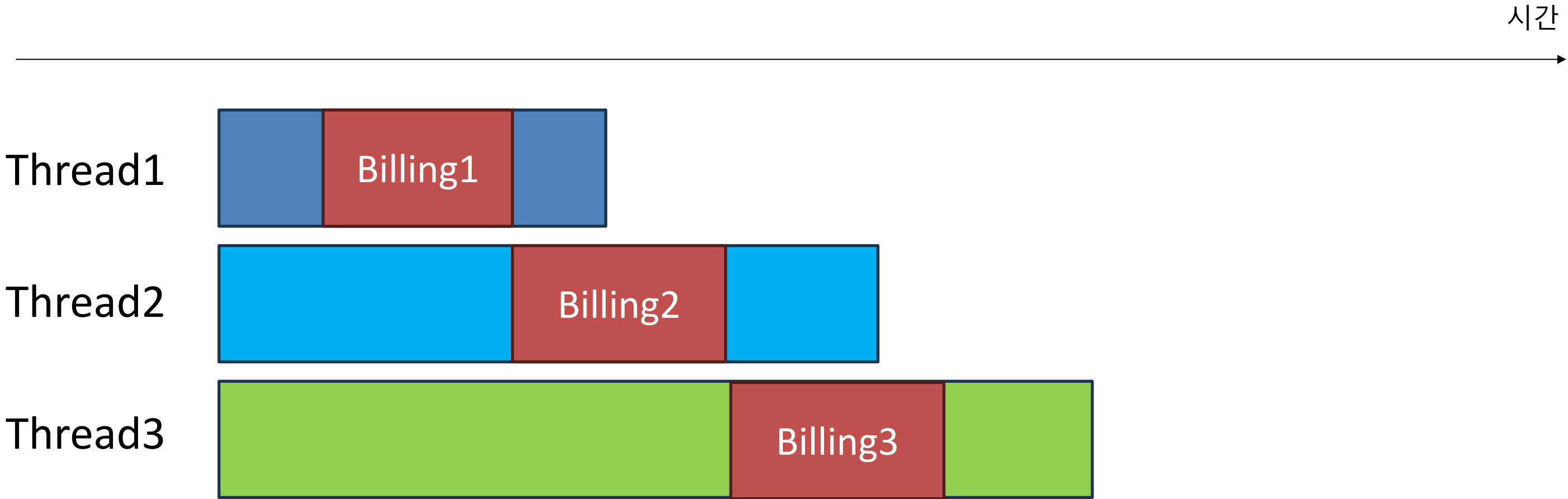
작업 대기 큐



04

두 번째 문제점 관련 OS 지식

비동기 처리(결제)



04

두 번째 문제점 관련 OS 지식

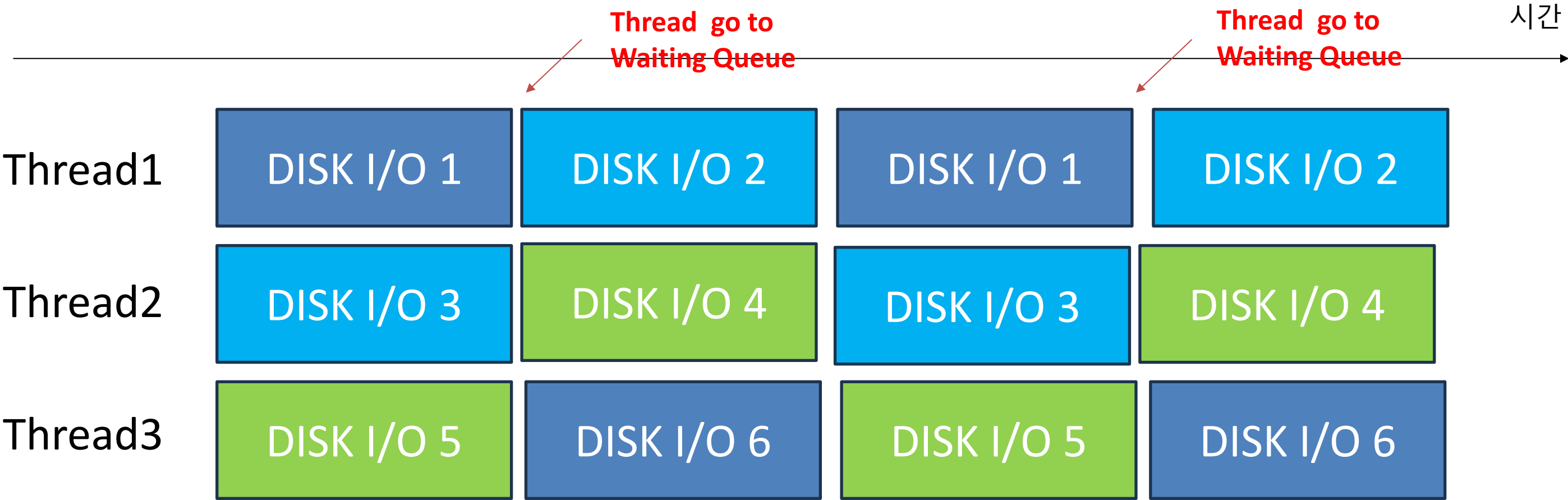
비동기 처리(메일)



04

두 번째 문제점 관련 OS 지식

비동기 처리(DISK 작업)



04 | 두 번째 문제점 해결 과정(쓰레드 풀 설정 및 초기화)

```
@Bean(name = "emailExecutor") new *
public Executor emailExecutor() {
    ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();
    executor.setCorePoolSize(CORE_POOL_SIZE);
    executor.setMaxPoolSize(MAX_POOL_SIZE);
    executor.setQueueCapacity(QUEUE_CAPACITY);
    executor.setThreadNamePrefix("emailExecutor-");
    executor.initialize();
    return executor;
}
```

04 | 두 번째 문제점 해결 과정(작업 스레드 풀 전달)

```
@Async("emailExecutor") 1 usage 👤 qoraudrb
public void sendEmailVerificationMessage(String toEmail, String token) {
    SimpleMailMessage message = new SimpleMailMessage();
    message.setFrom(EMAIL_ID);
    message.setTo(toEmail);
    message.setSubject(SUBJECT);
    message.setText(AUTH_URL + token);
    mailSender.send(message);
}
```

04 | 두 번째 문제점 해결 과정(트랜잭션 제거)

트랜잭션 제거!



```
public void createAccountWithEmailVerification(String nickname, String email, String password) {  
    if (memberService.isMemberExist(email))  
        throw new EntityExistsException(email);  
  
    memberService.joinMember(new Member(nickname, email, password));  
  
    String token = randomAlphanumeric(TOKEN_LENGTH);  
    emailVerificationService.sendEmailVerificationMessage(email, token);  
    emailVerificationService.createTemporaryEmailAuth(email, token);  
}
```

04

두 번째 문제점 해결 결과

The screenshot shows a REST client interface with the following details:

- URL:** `http://localhost:8080/account`
- Method:** `POST`
- Body:** A JSON object with the following fields:

```
{  1  {  2    "nickname": "qoraudrb123"  3    "email": "skkunion2024@gmail.com"  4    "password": "123456789"  5  }
```
- Response:** `201 Created` with a response time of `398 ms` and a size of `228 B`.
- Response Body:** A JSON object with the following fields:

```
{  1  {  2    "nickname": "qoraudrb123",  3    "email": "skkunion2024@gmail.com"  4  }
```

Annotations in the image highlight the `398 ms` response time in both the request body and the response status bar.

응답속도 문제 해결!

05

결론

1. 프로젝트를 하고 나서 더 개선해볼 방법이 없는 지를 고민해보자.
2. CS 관련 책이나 지식을 습득하는 과정에서 어떻게 활용을 할 수 있을 지에 대해서 생각을 해보자.
→ 작업의 특성과 과연 더 이상의 문제는 없는가와 함께 고민한다.
3. 프로젝트에 CS 관련 내용을 학습해 적용해서 개선을 해보자.