Detailed Specification

박용관

서지현

저희의 주제는 **“Unix 환경에 기여하기 위해 편리하게 chatgpt를 사용할 수 있는 명령어 만들기”** 입니다. 저희가 구현해야 할 명령어는 아래와 같습니다.

	형식	설명
1	chatAsk -m “~~” -pri	질문하기 - DB 저장 X (질문 노출 X)
2	chatAsk -m “~~”	질문하기 - DB 저장 O(키워드 추출 후 키워드 저장)
3	chatAsk -gc -m “~~”	문장 맞춤법 검사
4	chatAsk -t -m “~~~”	문장 번역(한국어)
5	chatAsk f —m “~~~”	문장 감정 분석
7	chatAsk -s -m “~~~”	문장 요약
8	chatAsk -q -k “~~~”	키워드를 사용해 사용자들 질문 추출

명령어 형식은 바뀔 수 있습니다.

따라서 저희의 프로젝트는 크게 2가지로 나뉩니다.

첫번째는 chatgpt와 통신하는 spring 서버 개발이고, 두번째는 해당 서버가 구현된 뒤 cli환경에서 해당 서버에 요청을 보낼 수 있는 명령어를 개발하는 것입니다.

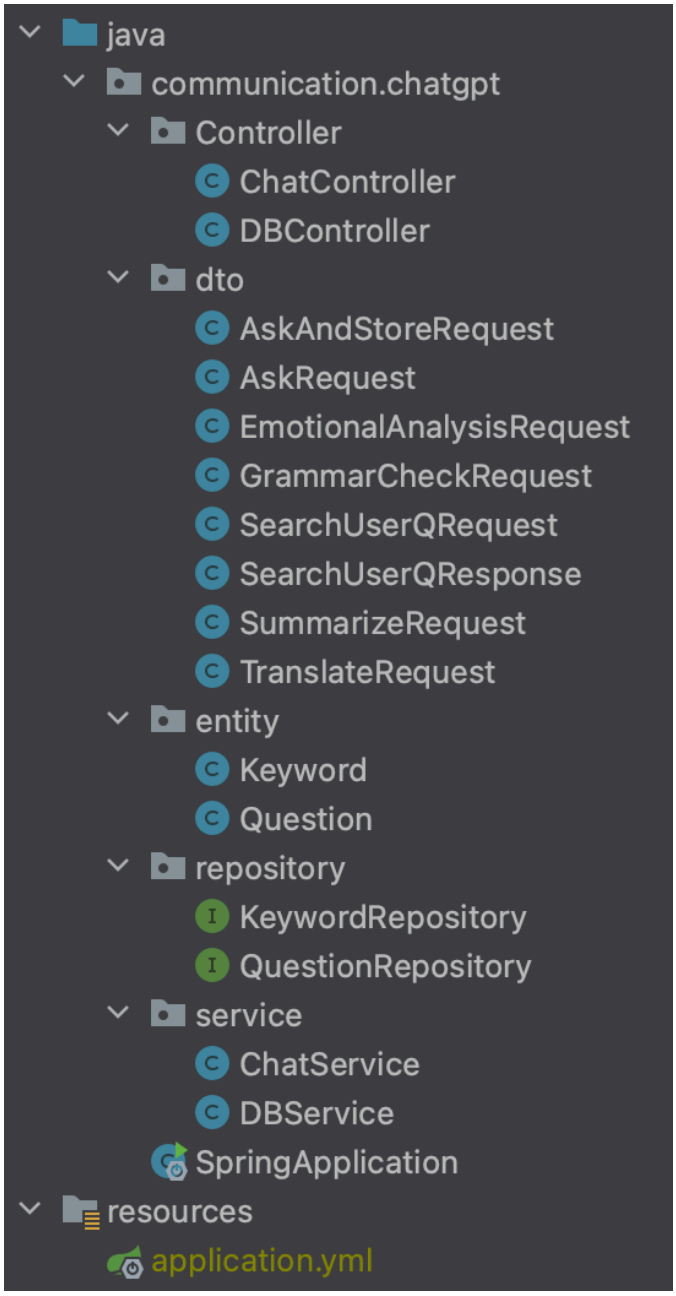
Spring 서버 개발은 springboot를 사용할 예정이며, cli환경에서의 명령어 개발은 쉘 스크립트를 통해 구현할 예정입니다. Spring 서버 개발의 가장 중요한 부분은 저희 서버에서 open ai 서버에 요청을 보내야하므로, 외부 api를 사용하는 것입니다. 이때 사용자의 편의성을 고려하기 위해서 요청 패킷이나 응답 패킷을 파싱해야하며 요청을 보낼때는 요청 해더 부분은 서버에서 만들어주고, 바디 부분은 사용자 측에서는 최대한 간결하게 해야 합니다. 따라서 이러한 작업들을 중점적으로 서버 개발을 진행해야 합니다.

두번째로, 쉘 스크립트는 저희에게는 UI같은 존재입니다. 따라서 이 또한 최대한 사용자들의 편리하게 사용할 수 있도록 구현해야합니다. 이에 대한 자세한 설명은 이후 상세 설계서에서 추가적으로 설명하도록 하겠습니다.

그렇다면, 해당 개발을 어떻게 할 것인지 구체적인 상세 설계서를 작성하도록 하겠습니다.

서버 개발

본격적인 서버 개발에 대한 상세 설계서를 설명하기에 앞서, 저희 서버 구현의 클래스 다이어그램을 소개하도록 하겠습니다.



왼쪽에 보이는 사진과 같이 저희는 크게 5개의 package로 구성되어 있습니다. 해당 패키지에 대해서 간략하게 설명하자면,

Controller는 주로 사용자의 요청이 진입하는 시점이며 요청에 따라 어떤 처리를 할지 결정해주는 중간 제어자 역할을 하는 역할을 담당하는 객체들의 집합입니다.

dto는 계층 간 데이터 교환을 위해 사용되는 객체들의 집합입니다.

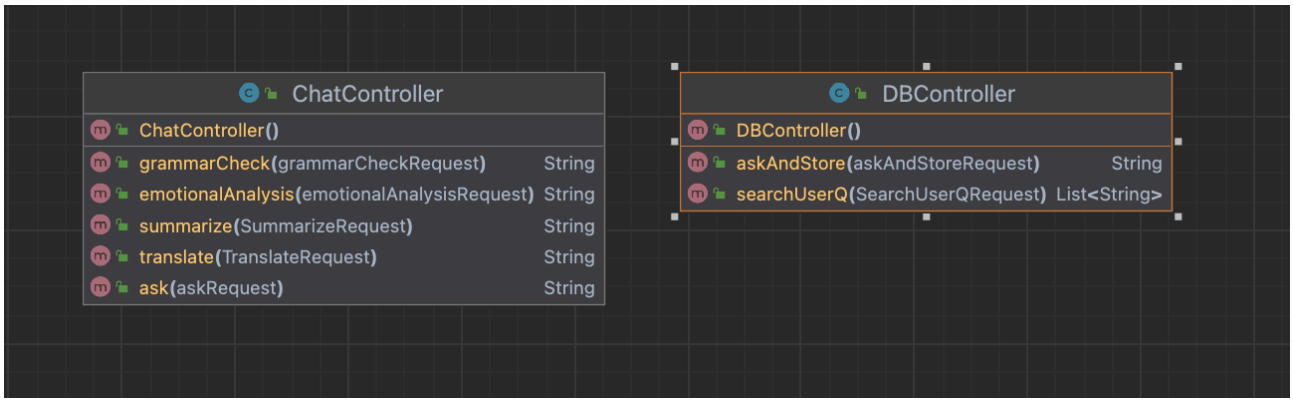
entity는 데이터베이스에서 쓰일 필드와 여러 엔티티간 연관 관계를 정의하는 객체들의 집합입니다.

repository는 entity에 의해 생서된 DB에 접근하는 메서드들을 사용하기 위한 인터페이스들의 집합입니다.

service란 데이터를 처리하기 위한 로직을 작성하는 객체들의 집합입니다. 해당 service에서 작성된 메서드를 controller에서 사용해 값을 처리할 예정입니다.

SpringApplication 클래스는 해당 서버의 동작을 담당하는 클래스입니다. 마지막으로, application.yml 파일은 이 프로젝트에서 사용하는 설정값들을 관리하는 파일입니다. 해당 패키지 및 클래스의 자세한 설명은 아래에서 진행하도록 하겠습니다.

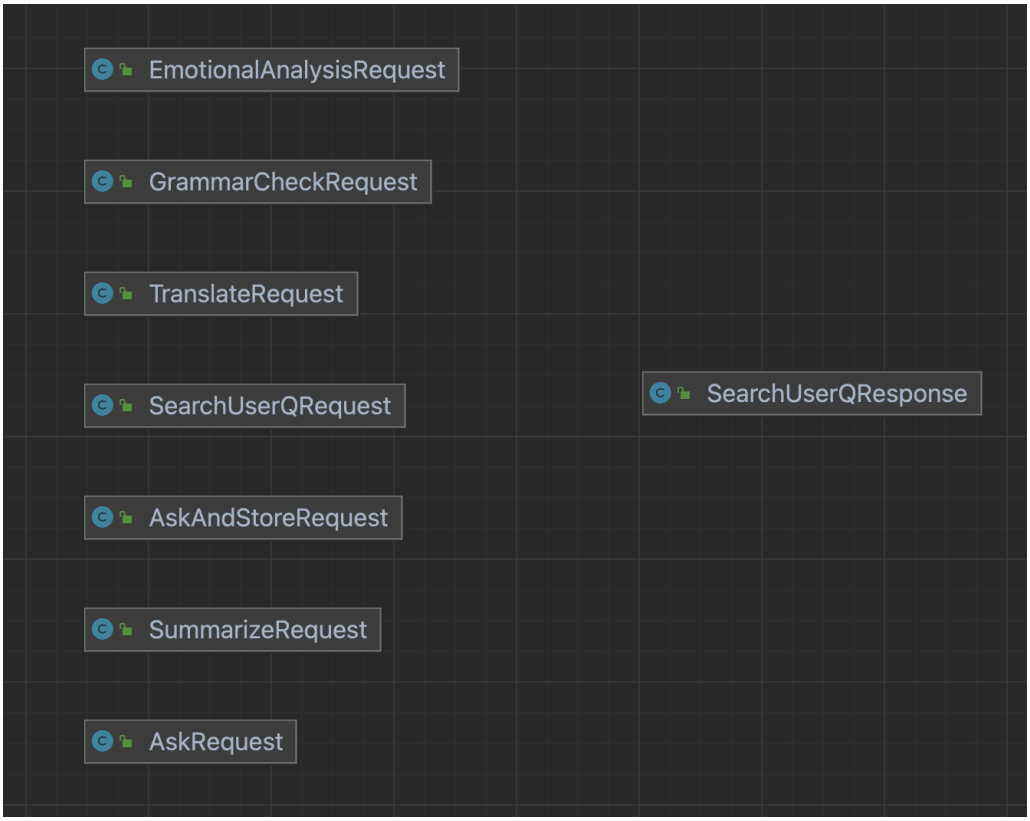
1. Controller



컨트롤러의 패키지의 클래스는 위와 같습니다. 크게 DB의 접근이 필요 없는 명령어의 요청을 처리하는 부분(ChatController)와 DB의 접근이 필요한 명령어의 요청을 처리하는 부분(DBController)로 나누었습니다.
ChatController에서 구현된 메서드는 메서드 명에서 알 수 있듯이 위에서부터 ChatController의 생성자, 문법 체크의 요청을 처리하는 메서드, 감정 분석 요청을 처리하는 메서드, 문장 요약 요청을 처리하는 메서드, 번역하기 요청을 처리하는 메서드, 질문하기의 요청을 처리하는 메서드로 나뉩니다.
두번째로 DBController에서 구현된 메서드는 질문 + 사용자의 질문을 저장하는 요청을 처리하는 메서드, 사용자들의 질문을 보여주는 요청을 처리하는 메서드로 나뉩니다.
기본적으로 해당 메서드는 dto에서 정의된 요청 패킷을 전달 받아 service로 보내고 service에서 구현된 메서드를 통해 open ai서버로 보낼 요청 패킷을 만듭니다. 그 후 open ai 서버로 요청을 보낸 뒤 응답 패킷을 받고 해당 패킷을 파싱해 최종적으로 사용자들에게 보여주는 역할을 담당합니다. 따라서 Controller는 중간 제어자 역할을 담당하는 것입니다.

정리하자면, controller는 전반적인 로직의 중간 제어자 역할을 하며 open ai의 요청 패킷을 전달하고 응답 패킷을 실제 사용자에게 보여주는 역할을 담당합니다. controller와 밀접한 관계가 있는 패키지는 dto와 service 패키지입니다. 해당 패키지들의 역할은 후에 자세히 설명하도록 하겠습니다.

2. DTO

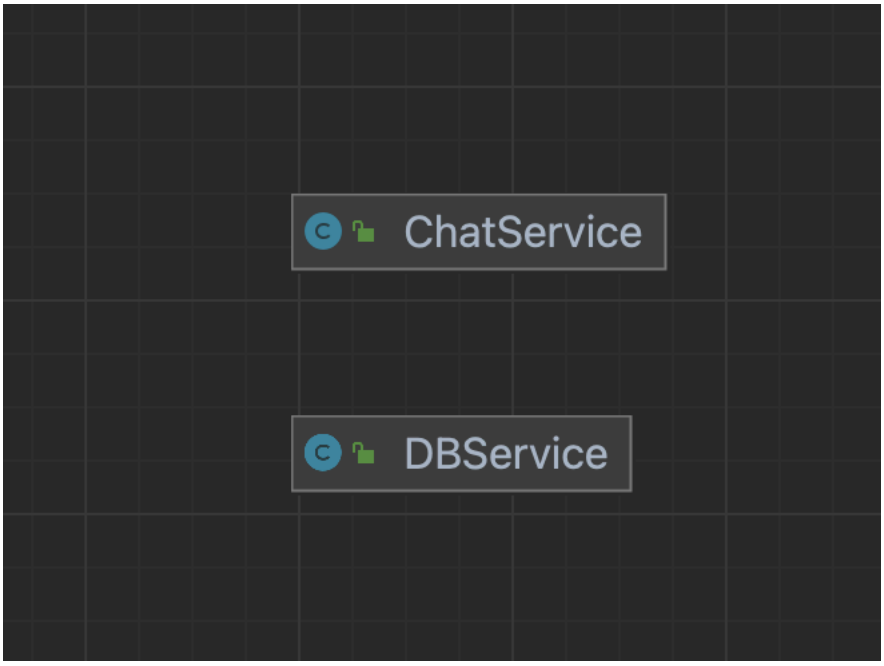


그렇다면 이번에는 dto에 대해서 설명하도록 하겠습니다. DTO는 Data Transport Object의 약자로 계층 간 데이터 교환을 위해 사용되는 객체들의 집합입니다. 여기서는 총 7개의 클래스가 존재합니다. 해당 클래스의 경우 각각 저희가 구현해야 할 명령어에 맞는 요청 형식을 정의하기위한 클래스들입니다. 명령어의 특성 상 open ai 서버에서 필수적으로 필요한 요청 패킷의 값들이 다릅니다. 또한, 저희는 해당 요청 패킷을 다 사용하는 것이 아닌 저희 나름대로 불필요한 값들은 서버에서 처리하고 꼭 필요한 값들은 사용자가 입력할 수 있게끔 처리해야하기 때문에 요청 패킷 형식을 정의하는 클래스들은 각각 필요합니다. 따라서 이런식으로 클래스를 나누어 처리하였습니다.

요청이 있다면 응답 패킷 형식을 정의하는 클래스들도 필요하지 않냐는 궁금증이 생길 수도 있습니다. 저희도 이 부분을 정말 많이 고민하였습니다. 저희가 내린 결론은 하나의 명령어를 제외한 나머지 명령어는 cli환경에서 보여줘야되기 때문에 간단하게 요청에 대한 답만을 보여주는 것이 좋다고 생각해 따로 response를 처리하는 클래스를 두는 것이 아니라 값을 파싱해서 결과에 대한 문자열을 보여주는 형식으로 하기로 결정하였습니다. 하지만, 사용자 질문을 보여주는 것은 keyword값에 따른 여러 결과값이 존재하기 때문에 해당 부분은 추가적인 작업을 통해서 사용자들이 보기 편하게 처리하는 것이 좋다고 생각하였습니다. 따라서 이 명령어만 따로 클래스를 두어 (SearchUserQResponse) 처리 할 예정입니다.

정리하자면, DTO의 경우 응답, 요청 패킷을 정의하는 역할로 사용됩니다.

3. Service



서비스의 패키지의 경우 DB를 다룰 필요가 없는 명령어와 관련된 서비스 로직을 다루는 ChatService 클래스와 DB를 다룰 필요가 있는 명령어와 관련된 서비스 로직을 다루는 DBService 클래스로 나누었습니다.
두개의 클래스의 전반적인 하는 역할은 일단 저희는 cli을 통해 요청을 받으면 해당 요청을 open ai의 요청 형식과 맞게 파싱하여 open ai 서버에 요청을 보내야 합니다. 즉, 저희 서버에 들어온 데이터를 json 형식으로 바꾸어 open ai 서버에 요청을 보내야 합니다. 그 작업을 service 패키지에서 담당합니다. controller와 dto을 통해 값을 받으면, service 단에서 open ai 서버에 맞는 요청 패킷을 생성합니다. 기본적으로는 header을 생성하고 body를 생성합니다. header는 기본적으로 Api key, MediaType등의 데이터가 들어갑니다. 해당 값을 포함한 해더를 생성합니다.(해당 값은 주로 서버에 있는 값으로 처리됩니다.) 그 후, json 형식으로 된 body를 생성합니다. 불필요한 값은 서버에 있는 값으로 처리되고, dto을 통해 사용자에게 전달 받은 값도 추가적으로 body에 저장됩니다. 이때 중요한 것은, json형식으로 body를 생성해야 된다는 것입니다. 해당 형식으로 구현하기 위해선 Map 자료구조를 사용하거나 spring Jackson 라이브러리에 구현되어 있는 ObjectMapper를 사용해야 합니다. 저희 같은 경우 ObjectMapper을 사용해서 구현할 예정입니다.
또한 이렇게 해서 open ai에 요청을 보내면 응답 패킷이 도착하는데 여러 데이터를 포함한 응답 패킷이 전달됩니다. 이를 바로 출력하면 cli환경에서 한눈에 결과값을 알아보기 힘듭니다. 따라서 저희 같은 경우 응답으로 온 json 데이터 중 필요한 부분을 추출하여 최종적인 사용자의 응답으로 보낼 예정입니다. 이를 구현하기 위해서는 spring jackson 라이브러리에 구현되어 있는 JsonNode을 사용해야 합니다.

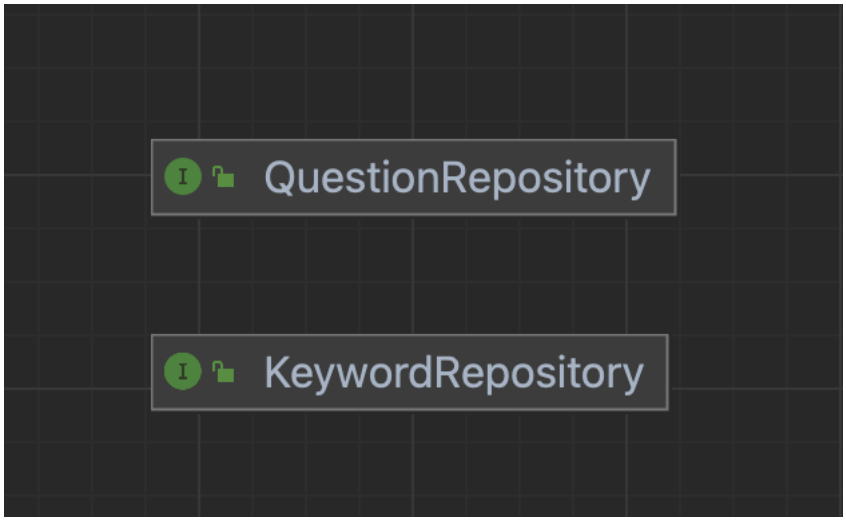
마지막으로 DB에 직접 접근하는 DAO 역할 또한 수행할 예정입니다. 저희는 DB를 다루는 로직이 복잡하지 않고 JPA는 이를 더 간편하게 처리할 수 있도록 구현되어 있습니다. 따라서 저희는 이를 서비스 로직에서 수행할 예정입니다.

정리하자면, service의 경우

1. ObjectMapper을 통한 open ai 요청 패킷 파싱하기
2. JsonNode을 통한 open ai 응답 패킷 파싱하기
3. Repository를 통한 DAO 역할

을 담당합니다.

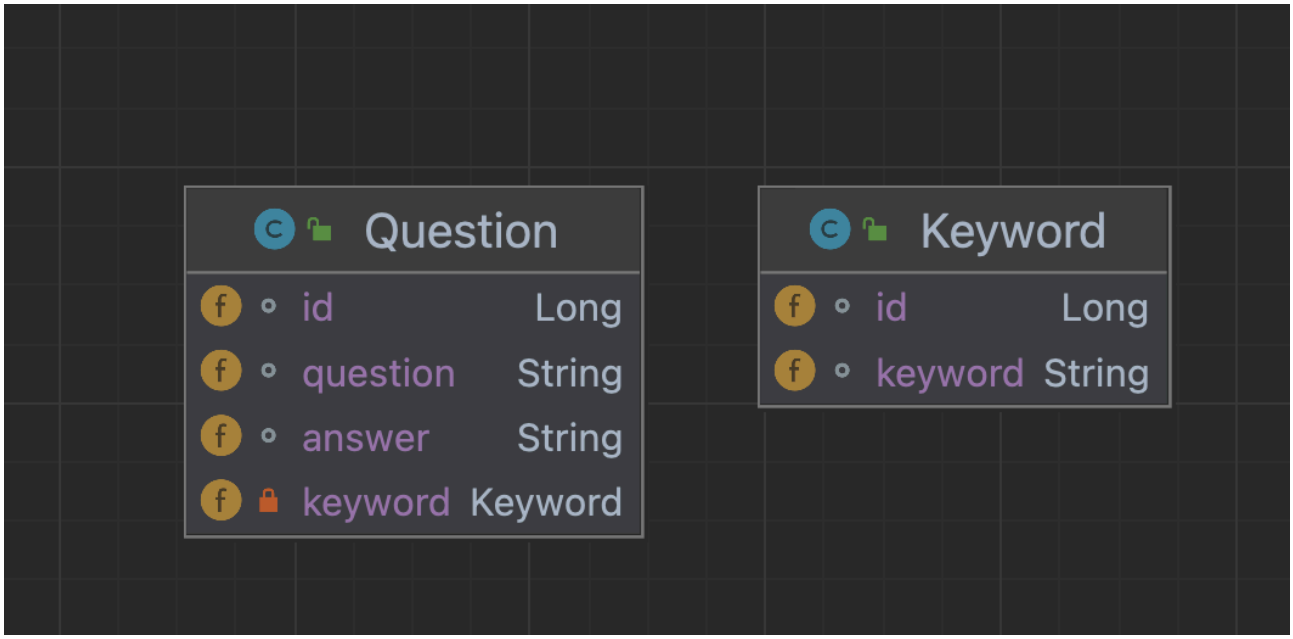
4. Repository



repository는 JpaRepository을 상속받은 인터페이스들의 집합입니다. Spring framework는 DB에 접근할때, 메서드 형식만 맞춰주면 별도의 추가적인 로직 없이 해당 작업을 수행해줍니다. repository에서는 이러한 메서드를 정의하고 DBService에서 해당 메서드를 불러와 DB작업을 처리할 수 있도록 하는 역할을 담당합니다.

정리하자면, Repository의 경우 JpaRepository를 상속하여 내가 원하는 형태이 Repository 형태를 만들 수 있는 역할을 담당합니다.

5. Entity



Entity는 데이터베이스에서 쓰일 필드와 여러 엔티티간 연관 관계를 정의하는 객체들의 집합입니다. 저희는 크게 사용자들의 질문을 저장하는 Question DB와 해당 질문들의 키워드를 추출하여 저장하는 Keyword DB로 구성되어 있습니다. 두 DB의 관계는 ManyToOne으로 매핑되어 있습니다. 왜냐하면 하나의 질문에 여러 키워드들이 연결되어 있을 수 있기 때문입니다.

저희는 이를 통해 사용자들의 질문이 저장될때 질문, 응답, 키워드 이렇게 3가지 데이터가 저장되고 후에 사용자가 다른 사람들의 질문을 요청할때 키워드을 통하여 사용자들의 질문에 접근하여 응답을 받을 수 있도록 구현할 예정입니다.

정리하자면, Entity의 경우 DB에서 쓰일 필드 정보와 엔티티들간의 관계를 정의하는 역할을 담당합니다.

기술 정리

기본적인 스프링부트를 사용하는 코드는 RestAPI를 사용하여 구현할 예정이고 저희 서버에서 중점적으로 다뤄야 하는 기술에 대해서 간단히 정리하도록 하겠습니다.

- HttpHeaders headers = new HttpHeaders();

  해당 코드는 Header를 만들기 위한 코드입니다. headers 변수에다가 open ai 가 원하는 형식의 header 값을 추가한 후 요청을 보낼때 같이 보내기 위해 사용됩니다.

- ObjectMapper objectMapper = new ObjectMapper();

  해당 코드는 body를 만들기 위한 코드입니다. 사용자 측에서 받은 값 & 서버에서 추가한 값으로 open ai 가 원하는 형식의 body를 만듭니다. 그후 요청을 보낼때 같이 보냅니다.

- RestTemplate rt = new RestTemplate()

  해당 코드는 외부 api 즉, open ai와 통신을 하기 위한 RestTemplate 입니다. 스프링에서는 RestTemplate을 통해 외부 api에 GET, POST, PUT …등의 요청을 보내고 응답을 받을 수 있습니다.

- JsonNode jsonNode = objectMapper.readTree(json)

  해당 코드는 open ai에서 받은 json 형태의 응답을 저희가 원하는 형식의 답을 추출하기 위해 사용됩니다. 이로 인해, 복잡한 응답 패킷을 파싱해서 사용자들이 이해하기 편한 답으로 값을 변형할 수 있습니다.

- ResponseBody를 통한 데이터 값 추출

  해당 방법도 open ai에서 받은 json 형태의 응답을 저희가 원하는 형식의 답을 추출하기 위해 사용됩니다. 저희는 위의 JsonNode와 ResponseBody 중 적절한 방법을 골라 값을 파싱할 예정입니다.

세션 유지 방식

Open ai api를 사용한 결과 질문이 연속적으로 진행되지 않는다는 것을 알게 되었습니다. 즉 내가 A를 질문해 답을 얻고 B를 질문하면 원래라면 질문이 연속적으로 진행돼 A의 데이터가 다음 질문에서 반영되는데 Open ai api를 통해 직접 질문하면 A를 질문하고 B를 질문하면 A의 데이터가 B에 반영이 되지 않는 것을 확인하였습니다. 따라서 이 경우에는 매번 질문을 DB에 저장하고 불러오는 것은 데이터가 너무 많아 현실적으로 불가능하였습니다. 이 부분을 조사한 결과 Open ai api를 사용한 많은 사람들이 해당 문제를 겪고 있는 것을 확인하였습니다. 이에 대한 해결책으로는 여러가지가 있지만, 저희 같은 경우 [https://stackoverflow.com/questions/74774018/how-to-keep-the-conversation-going-with-openai-api-php-sdk] 여기서 나온 방식으로 이러한 문제를 해결하려고 합니다. 간단하게 설명드리자면, 질문이 올때마다 DB에 저장하는 것이 아닌 질문들을 이어서 서버에 질문하는 방법으로 문제를 해결하는 방식입니다. 예를 들어, A를 질문하고 B를 질문하면 B는 open ai 서버에서 B의 내용만 가는 것이 아닌 A + B의 내용이 가는 식으로 구현하는 방식입니다. 따라서 저희는 이러한 형식으로 세션을 유지하려고 합니다. 이때, 실제로 chatgpt 서버에도 있는 regenerate response 버튼 같은 기능을 추가로 구현하여 전의 질문을 리셋하는 형식으로 개발할 예정입니다.

이밖에도 개발을 하다가 추가적인 문제가 발생할 수 있습니다. 그때마다 팀원과 협업하여 올바른 정답을 찾고 해결할 수 있도록 노력하겠습니다. 감사합니다.

정리

이렇게 저희의 spring 서버 구현에 대한 전반적인 설계에 대해서 소개를 마쳤습니다. 저희는 일단 spring 서버 개발을 주 목적으로 진행할 예정이며 추후에 서버 개발이 완료되면 cli 환경에서 명령어 처리를 위한 쉘 스크립트 작성, 서버 배포를 위한 docker 파일 작성 등을 할 예정입니다. 명령어의 형식은 일단 첫장에서 보여드린 형식으로 진행할 예정이고 서버 배포를 위한 docker파일을 AWS 인스턴스를 통한 무중단 배포 형식으로 진행할 예정입니다. 감사합니다!