DTO 활용에 대한 고찰

1. 개요

- 현행 JAVA 개발 패턴의 REQUSET 및 RESPONSE를 명확하게 하고자함

2. 현재상황

- 현행 개발 패턴의 REQUEST 및 RESPONSE는 모두 MAP을 통해 전송, 전달 받음
  - 장점
    - 개발 시 DTO 정의를 하지 않아 편리함과 시간단축에 장점이 있음
  - 단점
    - 시스템 개발 후 담당자 이외의 다른 개발자가 유지보수 및 소스 분석시
      어떤 데이터형식으로 전송 받고 전달 하는지 명확하게 확인 할 수 없음
    - 데이터 정보 확인을 위해 디버그를 하거나 쿼리등으로 유추할 수 밖에 없음
    - 데이터 타입에 대한 명확성을 알 수 없어 캐스팅 관련 소요시간 증가
- CONTROL 단에서 VALIDATE 처리 시 소스블럭이 길어지는 현상 발생
  - 장점
    - 소스블럭 내에서 바로 유효성 검증이 이루어 지기에 VALIDATE에 대한 가시성이 좋음
  - 단점
    - REQUSET DATA의 VALIDATE 대상 PARAM이 많을 수록 소스블록이 길어져
      빠르게 중요로직만 확인할 수 없음

- 프론트, 백엔드를 연결할 인터페이스 정의서가 없음
  - 장점
    - 개발자가 싫어하는 문서작성이 없어 편리함
  - 단점
    - 업무 담당자만 내용을 알고 있을 뿐 다른 담당자는
      데이터 통신이 어떻게 되는지 내용 확인하기 힘듬

3. 발전방향

- DTO를 통한 명확한 의사소통 수단으로 활용

    - 장점
        - DTO을 통한 인터페이스 정의로 문서 또는 디버그 없이 의사소통이 가능
        - DTO을 통해 JS, XML 탐색없이 JAVA 단에서 REQUEST, RESPONSE 결과 확인
        - VALIDATE 조건 확인을 통해 필수값 여부 확인 가능
        - 추후 팀 또는 부서에서 프론트, 백엔드로 구분하여 인력 운영시 명확한 의사소통 도구
    - 단점
        - 각 REQUSET, RESPONSE 마다 DTO를 생성해야 함에 따라 개발 소요시간이 늘어남

4. 참고자료

- 유효성 검증
    - DTO 내부의 어노테이션을 통한 유효성 검증
    - 관련 URL
        - https://velog.io/@_koiil/SpringBoot-Spring-Validation을-이용한-유효성-검증
        - https://meetup.nhncloud.com/posts/223
        - https://mangkyu.tistory.com/174
        - https://jeong-pro.tistory.com/m/203
- DTO 처리
    - 관련 URL
        - https://velog.io/@rmswjdtn/Spring-Response-Request-DTO-꽉잡기
        - https://blog.jiniworld.me/155
        - https://velog.io/@p4rksh/Spring-Boot에서-깔끔하게-DTO-관리하기
        - https://jaehoney.tistory.com/157?category=983083
        - https://dev-coco.tistory.com/m/138
- 추가정보
    - 관련 URL
        - https://mangkyu.tistory.com/164
        - https://wildeveloperetrain.tistory.com/m/101
        - https://song8420.tistory.com/m/383
        - https://itmemo.tistory.com/m/105