

@RequestBody로 받았는데 null인 경우

Ssol · 2023년 2월 7일

팔로우

❤️ 1

[JavaBeans 규약](#)[Lombok](#)[RequestBody](#)[getter](#)[jackson](#)[json](#)[null](#)

null



아반떼 HD
480만원

아반떼 HD
430만원

[보기](#)

더
2,680

이게 왜 null?

데이터 삽입/수정을 테스트하기 위해 Postman에 @RequestBody에 필요한 값들을 넣어서 서버에 요청을 보냈는데 자꾸 null값을 가져오는 문제를 만난적 있는가?

```
{
    "sGroupId" : 102,
    "sGroupName" : "네트워크 스터디",
    ...
}
```

위와 같이 body를 전달하고 컨트롤러에서 받아서 로그를 찍어보면

```
{"sGroupId":null, "sGroupName":null, ...}
```

이렇게 나와버린다...

무엇이 문제였을까?

원인은 Jackson?

이유를 찾아보니 스프링부트에서 json으로 데이터를 변환하고 맵핑하기 위해 사용하는 **Jackson**이라는 라이브러리를 비롯한 복합적 문제였다.

이 Jackson 라이브러리의 무엇이 문제였는지 알기 위해 우선 Jackson이 json으로 데이터를 변환하는 과정이 어떻게 되는지 알아야 한다.

Object를 json으로 변환할 때 key 값을 필드명으로 잡는것 같지?

사실은 필드명이 아니라 Getter의 이름을 기준으로 변경해준다. 즉, name이라고 필드명을 사용해도 Getter 메서드로 getUsername이라고 써버리면 json의 키 값이 userName이 되는 것이다.

```
public class UserDto {
    private String name;

    public String getUsername() {
        return name;
    }
}

UserDto userDto = new UserDto("Sol");
String content = objectMapper.writeValueAsString(jacksonDto);

System.out.println(content); // {"userName":"Sol"}
```

바로 이렇게...

Jackson이 json key로 변환하는데에는 일정한 규칙이 있다.

Jackson은 기본적으로는 JavaBeans 규약을 따르지만 다른 부분이 있다.

여기서 JavaBeans 규약이란?

자바빈을 사용하기 위해서 몇가지 규칙을 정해둔 것이다.

자바빈을 사용하는 이유는 디자인(프론트엔드)와 로직(백엔드)을 분리하기 위해서이다. 공통의 약속을 지키며 사용함으로써 프론트엔드에 백엔드의 로직을 구현하는 등의 일이 없이 일관된 방식으로 자바 클래스를 사용할 수 있도록 도와준다.

이 규약의 내용을 몇가지 소개하자면

1. 기본 생성자를 반드시 가지고 있어야 한다.
2. 빈이 패키지화 되어 있어야한다.
3. 멤버 변수의 접근자는 private으로 선언한다.
4. 멤버 변수에 접근하기 위한 public 접근자인 getter/setter 메서드가 존재해야 한다.
5. get 메서드는 파라미터가 존재하지 않아야 한다
6. set 메서드는 반드시 하나 이상의 파라미터가 존재해야 한다

등이 있다.

이 중에 이번 Jackson 문제 관련으로 확인해야 할 것은

클래스의 이름은 일반적으로 대문자로 시작하지만, 개발자들은 식별자가 소문자로 시작하는 것에 익숙하기 때문에 첫 번째 글자를 소문자로 변환한다. 다만, 모든 문자를 대문자로 사용하는 경우도 있기 때문에 이런 경우는 예외로 둔다.
그리고 예외 케이스를 판별하기 위해 첫 두 문자가 모두 대문자인지를 확인한다.

라는 규약이다.

java.beans 패키지에 있는 *Introspector* 클래스를 확인해보면 실제로 어떤 로직이 들어가있는 지 알 수 있다.

```
public class Introspector {
    /**
     * Utility method to take a string and convert it to normal Java variable
     * name capitalization. This normally means converting the first
     * character from upper case to lower case, but in the (unusual) special
     * case when there is more than one character and both the first and
     * second characters are upper case, we leave it alone.
     * <p>
     * Thus "FooBah" becomes "fooBah" and "X" becomes "x", but "URL" stays
     * as "URL".
     *
     * @param name The string to be decapitalized.
     * @return The decapitalized version of the string.
     */
    public static String decapitalize(String name) {
        if (name == null || name.length() == 0) {
            return name;
        }
    }
```

```

        return name;
    }
    char chars[] = name.toCharArray();
    chars[0] = Character.toLowerCase(chars[0]);
    return new String(chars);
}

//...
}

```

- 맨 앞 두개가 전부 대문자라면 그대로 리턴하고 아니라면 맨 앞 문자 하나만 소문자로 바꿔서 리턴

자바빈 규약과는 다른 Jackson의 규칙

Jackson도 JavaBeans 규약을 따르지만 다른 점이 하나 있다.

1. 맨 앞 두 글자가 모두 대문자인 경우 이어진 대문자를 모두 소문자로 변경한다.
2. 나머지 모든 케이스에서는 맨 앞 글자만 소문자로 바꿔준다.

저 첫번째 규칙이 바로 자바빈 규약과 다른 점이다.

자바빈 규약에선 첫 두글자가 대문자이면 그대로 사용한다고 되어 있지만 Jackson은 첫 두글자가 대문자이면 모두 소문자로 바꿔버린다.

테스트

```

@NoArgsConstructor
public class SampleDto {
    private String AAaa;
    private String BBBb;
    private String CCcC;
    private String DDDD;

    public String getAAaa() {
        return AAaa;
    }

    public String getBBBb() {
        return BBBb;
    }

    public String getCCcC() {
        return CCcC;
    }

    public String getDDDD() {
        return DDDD;
    }
}

```

[Request]

```
{
  "AAaa": "a",
  "BBBb": "b",
  "CCcC": "c",
  "DDDD": "d"
}
```

[컨트롤러]

```
@RestController
public class SampleController {

    @PostMapping("/api")
    public ResponseEntity<SampleDto> postSample(@RequestBody SampleDto dto) {
        System.out.println(dto);

        return ResponseEntity.ok(dto);
    }
}
```

응답 결과를 확인해보면

[Response]

```
{
  "aaaa": null,
  "bbbb": null,
  "cccC": null,
  "dddd": null
}
```

값이 전부 null이다.

왜? 😞

자세히 보면 json key 값들이 원래 설정했던 값과 다르게 응답된 것을 확인할 수 있다.

당연히 request로 보낸 값과 대소문자가 안맞으니 null인 것이겠지.

위에서 봤던 Jackson 라이브러리의 규칙을 생각해보자.

- AAaa → aaaa : 앞 두 글자가 대문자라서 소문자로 변경
- BBBb → bbbb : 앞 두 글자가 대문자라서 이어진 세번째 문자까지 소문자로 변경
- CCcC → cccC : 앞 두 글자를 소문자로 변경하지만 맨 뒤의 대문자는 이어져 있지 않아서 그대로 사용
- DDDD → dddd : 앞 두 글자부터 이어진 대문자를 모두 소문자로 변경



```
@NoArgsConstructor
public class Sample2Dto {
    private String aaaa;
    private String bbbB;

    private String Cccc;
    private String DddD;

    private String eEee;
    private String fFfF;

    public String getAaaa() {
        return aaaa;
    }

    public String getBbbB() {
        return bbbB;
    }

    public String getCccc() {
        return Cccc;
    }

    public String getDddD() {
        return DddD;
    }

    public String geteEee() {
        return eEee;
    }

    public String getfFfF() {
        return fFfF;
    }
}
```

컨트롤러는 동일하게 사용해서 다음과 같은 응답을 보내보았다.

[Request]

```
{
  "aaaa": "a",
  "bbbB": "b",
  "Cccc": "c",
  "DddD": "d",
  "eEee": "e",
  "fFfF": "f"
}
```

이제 Jackson의 Convert 규칙을 알게되었으니 이것의 json 변환 결과도 예상할 수 있겠지?

- getAaaa()는 첫글자가 대문자이므로 소문자로 바뀌어서 → aaaa
- getBbbB()는 첫 대문자와 연결된 대문자가 아니므로 → bbbB

- geteEee()는 첫글자가 대문자가 아니므로 기존 그대로 사용해서 → eEee
- getfFff()도 첫글자가 대문자가 아니므로 기존 그대로 사용해서 → fFff

이렇게 변환된 json key와 Dto의 필드 값과 매칭 상태를 보면되는데

```
getter json 변환 결과: aaaa = 필드 명: aaaa
getter json 변환 결과: bbbB = 필드 명: bbbB
getter json 변환 결과: cccc != 필드 명: Cccc
getter json 변환 결과: dddD != 필드 명: DddD
getter json 변환 결과: eEee = 필드 명: eEee
getter json 변환 결과: fFff = 필드 명: fFff
```

[Response]

```
{
  "aaaa": "a",
  "bbbB": "b",
  "cccc": null,
  "dddD": null,
  "eEee": "e",
  "fFff": "f"
}
```

필드 명과 매칭이 안되는 cccc 와 dddD 는 당연히 null인 예상대로의 결과가 나온 것을 확인할 수 있다.

Jackson 규칙 결론

여기서 확인할 수 있는 결론은 **필드 명의 첫 글자가 대문자이면 값이 제대로 들어오지 않는다는 점**이다.

필드명이 대문자로 시작해도 Getter는 보통 대문자로 시작한다. 더군다나 Lombok으로 Getter를 생성하는 것이라면 당연히 필드 명에서 첫글자를 대문자로 사용하게 되겠지.

하지만 Jackson의 규칙에 따라서 get 이후가 대문자로 시작하면 최소한 첫 글자는 항상 소문자로 바뀌게 되고 대문자로 시작하는 필드와 매치가 안되게 되는 것이다.

또 다른 원인 Lombok?

위 Jackson 변환 규칙 결론에서 보듯이 필드명의 첫 글자가 대문자이면 값이 제대로 들어오지 않는데 이것은 Lombok으로 생성한 Getter의 문제도 있다.

(사실 Lombok Getter의 문제라기 보다는 애초에 필드명을 대충 지은 책임이 있겠지만 우선은 Lombok을 중점으로 보도록 하자.)

이 중 `@Getter` 는 특히 거의 모든 Object에서 사용하는 어노테이션이다.
이 `@Getter`는 어떤 방식으로 get 메서드를 자동 생성해줄까?

@Getter의 작동 방식

사실 `@Getter` 는 딱히 복잡한 규칙없이 필드의 첫 글자를 대문자로 변경해서 get 메서드를 생성해준다.
즉, 제일 처음에 예시를 들었던 `sGroupId`라는 필드의 `@Getter` 생성 메서드는 `getSGroupId()`인 것이다.

하지만 우리는 Jackson의 json 변환 방식을 알게 되었다.

`getSGroupId()`를 사용해 `sgroupId`로 변환 되겠지? 그런데 이렇게 변환된 json key 값과 필드의 값이 일치하지 않아서 값이 들어가지 않는 문제가 발생하게 된다.

이 문제를 피하기 위해서는 Lombok이 아닌 수동으로 Getter를 만들어주면 된다.
IntelliJ에서 제공하는 Getter 생성 기능을 사용하면

```
public int getsGroupId() {  
    return aCount;  
}
```

이렇게 만들어주기 때문에 Jackson의 변환 과정을 거쳐도 문제 없도록 만들어준다.

하지만 대부분의 개발자는 클래스에 직접 get 메서드를 선언하는 것보다는 `@Getter`를 사용하는 편이잖아? 그 편이 코드도 더 적어지고, 편하니까...

그렇다면 `@Getter`를 사용해도 문제가 없도록 필드 명을 잘 설정해서 문제가 없도록 하는게 최선의 방법이 되겠다.

즉, 니가 필드 명을 제대로 지었으면 이런 문제가 발생하지 않았을 것이란 것이다!! 😊

이름 좀 잘 지어!!

클린코드 두 번째 챕터에 나오는 것이 바로 '**의미있는 이름**'이다.

그 중 불필요한 맥락을 없애라라는 파트에 나오는 설명을 인용하자면 '고급 휘발유 충전소(Gas Station Deluxe)'라는 애플리케이션을 짤 때 모든 클래스 이름을 GSD라고 시작하는 것은 바람직하지 않다고 한다.

G를 입력하고 자동완성 단축키를 누르면 IDE는 모든 클래스를 추천되게 될 것이다. IDE는 개발자를 지원하는 도구인데 굳이 이렇게 IDE를 방해할 필요가 없기 때문이다.

일반적으로는 짧은 이름이 긴 이름보다 좋지만, 이것은 의미가 분명한 경우에 한해서이다.

처음 예시로 들었던 '`sGroupName`'처럼 스터디그룹을 줄여서 쓰지말고 명확하게 풀어서 썼다면 이런 문제를 겪지 않았을 것이다.

당장 새 팀원이 합류했을 때 프로젝트의 클래스, 변수, 메서드 이름만 보고도 이게 무엇인지 알 수 있도록 만들자.

그래서 개발자들이 좋은 변수 명을 지으려고 고민을 하는 것이 아니겠어?

해결 방법

그래서 RequestBody에 null이 들어가는 문제를 해결하는 방법이 무엇이냐? 세 가지가 있는데

1. 필드 명을 수정하지 않고 그대로 사용하고 싶다면 직접 **Getter 메서드를 생성해서 사용한다.**
2. 해당 필드에 **@JsonProperty** 를 붙여주면 된다. 필드에 선언한 그대로 json 키 값을 만들겠다는 어노테이션이다.
3. **애초에 필드 명을 작성할 때 첫 번째는 소문자, 두 번째는 대문자인 케이스로 만들지 않는다.**

정도가 되겠다.

현 프로젝트 경우는 기존에 필드명을 사용하는 부분이 좀 많은 편이었고 프론트에서도 이미 이렇게 받아서 사용 중이었기 때문에 일단 2번 방법으로 해결을 하였다.

3번 해결법이 가장 바람직하다고 생각은 들지만 실무에선 어쩔 수 없이 어느정도 타협을 해야 할 때도 있으니까...

나중에 프론트 분이랑 같이 한번에 바꾸는 시간을 잡던가 해야지.

참고

- <https://www.baeldung.com/spring-httpmessageconverter-rest>
- <https://bcp0109.tistory.com/309>



Ssol

Junior Back-end Developer ☐

팔로우



이전 포스트

@PathVariable에 email이 안들어가??



제네시스 DH	아반떼 HD	아반떼 HC
1,250만원	430만원	480만원

0개의 댓글

댓글을 작성하세요

댓글 작성

관심 있을 만한 포스트



개발을 위한 이름 짓기 📝

개발을 위한 이름 짓기

네이밍 컨벤션에 관한 간단한 고찰

2021년 2월 2일 · 1개의 댓글



by 예술

♥ 2

```
bernate:
select
    user0_.u_id as u_id1_0_,
    user0_.id as id2_0_,
    user0_.nick_name as nick_nam3_0_
from
    tb_user user0_
where
    user0_.id=?
```

[JPA] Spring Boot JPA Entity Table 대, 소문자 구분 못하는 경우 해결

SpringBoot에서 JPA를 통해서 Entity Class에 @Table Annotation으로 DB Table 명을 아래 사진처럼 대문자로 입력했는데, 실제로 Hibernate의 Query 실행 결과를 보니 소문자로 매핑되고 있었고, 대문자로 생성된 Table...



by GilLog

♥ 5



개발자의 글쓰기

변수 네이밍부터
릴리스 노트,
장애 보고서,
기술 블로그까지

김철수 지음

네이밍 컨벤션과 변수이름 짓기

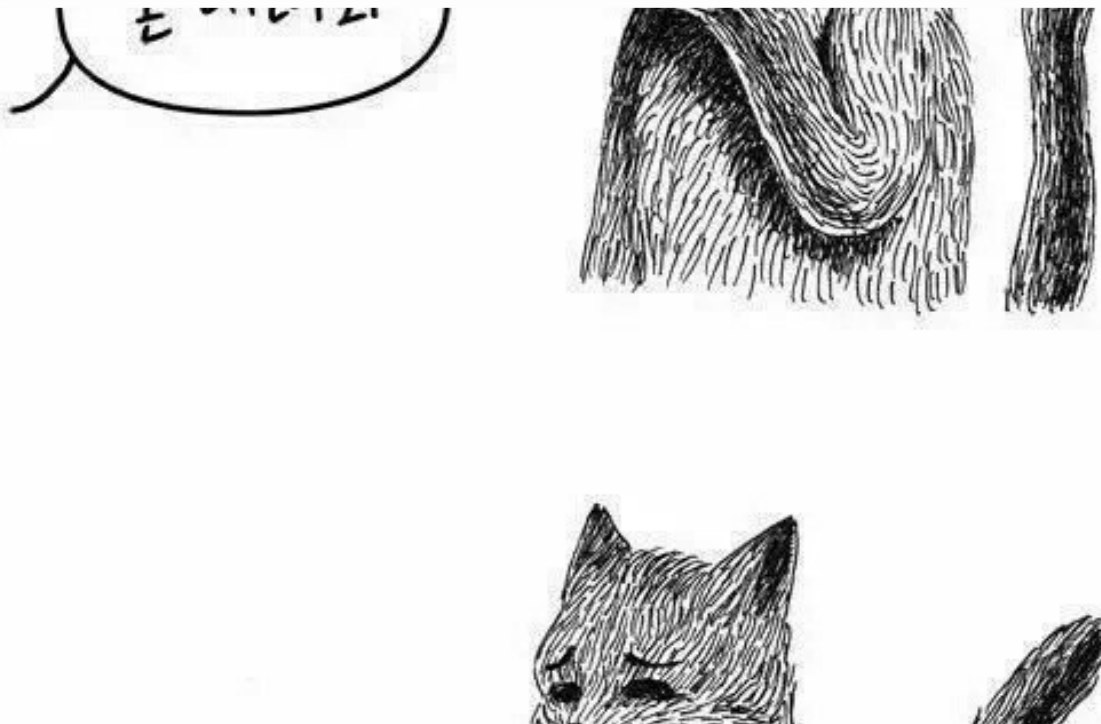
네이밍 컨벤션 알고쓰자! 주석, 악순환의 늪에 빠지지 않도록 잘 쓰는 법은?

2021년 12월 15일 · 0개의 댓글



by JueunPark

♥ 6



 ssol_916.log

해서는 안된다.면수병은 안알이 안된다.면수 이름 사이에 공백 불가특수문자는 '₩_' 와 '\$'만 허용예약어를 사용 하면 안된다.예약어란 자바에 등록되어 있고, 시스...

2021년 4월 6일 · 0개의 댓글



by Jinsung

♥ 1

코드 작성 규칙들 (Coding Conventions)

여기까지 오느라 고생했어! 지금까지 배웠던 것들은 모두 프로그래밍의 기본중의 기본이야. 이런 기초적인 지식을 배웠으니 이제부터는 본격적으로 개발이라는걸 해봐야겠지?!그 전에 지켜줘야 할 것들이 있어.우리의 인생에는 '상식' 이라는게 있잖아. 국어사전에 따르면 '상식' ...

2022년 3월 6일 · 0개의 댓글



by Rex

♥ 5

[Elasticsearch] Token Filter 정리

apostrophe : '을 삭제. ' 뒤에 붙어있는 글자는 삭제됨asciifolding : ascii 형태가 아닌 글자를 ascii 형태로 변형.
preserve_original: true로 하면 원본도 저장 가능cjk_bigram: 한국어, 중국어, 일본어를 분석...

2020년 5월 19일 · 0개의 댓글



by Dahea Moon

♥ 0

CSS에서의 네이밍 규칙

CSS에서의 네이밍 규칙


개발을 하다보면 가장 어려운 것은 역시 변수명 짓기가 아닐까요?

2021년 3월 17일 · 0개의 댓글

[Java] 변수(Variable)

변수는 값을 저장할 수 있는 메모리(RAM)의 특정 번지에 붙이는 이름이다. 변수의 값은 변경이 가능하고 하나의 변수에는 하나의 값만 저장할 수 있다. 따라서 값을 여러번 저장할 경우 마지막에 저장된 값을 가지게 된다. 변수를 사용하기 위해선 어떤 타입의 데이터를 저장할 것...

2021년 5월 3일 · 0개의 댓글

 by chael_lo

♥ 1

[JS/Basic] 식별자(Identifier)

식별자 : 자바스크립트에서 이름을 붙일 때 사용하는 단어식별자의 종류 변수명 함수명 속성명 메소드명 식별자 금지 규칙 식별자는 자바스크립트에서 이름을 붙일 때 사용하는 단어이다. 식별자의 예로는 변수명과 함수명이 있다. 키워드를 사용하면 안 된다. 숫자로 시작하면 안 된다. 특수 문자...

2020년 3월 31일 · 0개의 댓글

 by Jay Kim

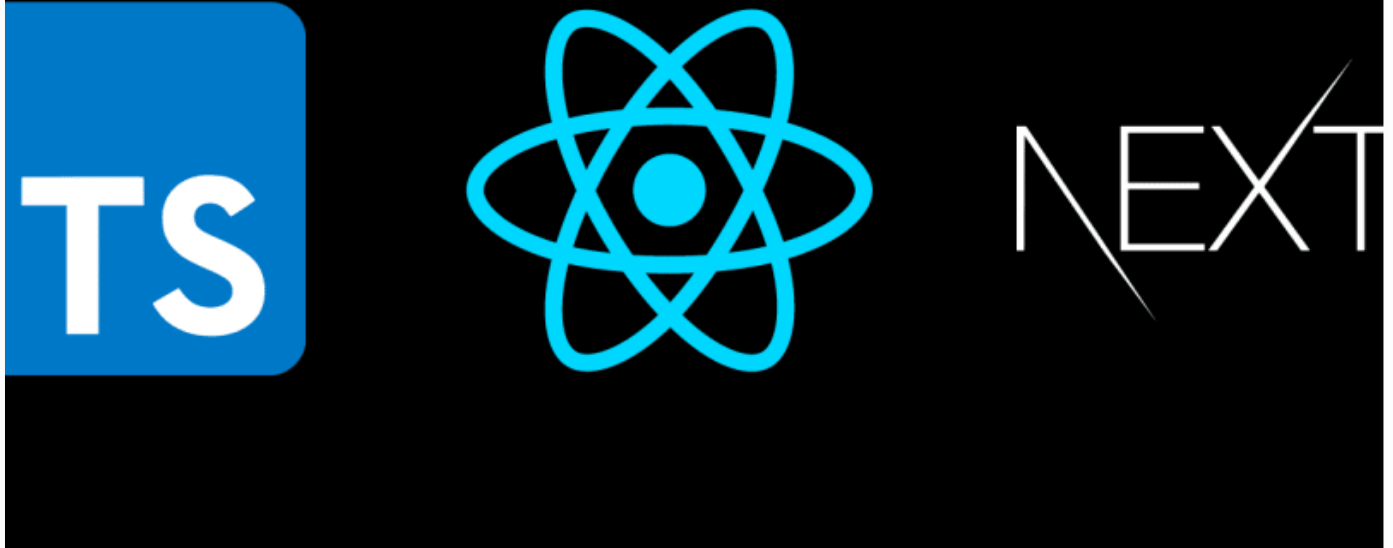
♥ 2



JDBC, MySQL 연동 1 - 기본구조와 연동

JDBC API 사용을 위해 Class.forName 메서드를 이용하여 JDBC 드라이버 로딩 DB 마다 클래스 이름이 다르므로 해당 DB 이름을 정확하게 넣어야 한다.

2021년 12월 10일 · 0개의 댓글



Next.js + TypeScript 페이지 만들기 2

본 내용은 next.js 와 nextjs. + typescript 를 비교하기 위해 학습용 으로 작성되었습니다 아래 내용대로 따라하시면 기본적인 세팅과 nextJS + typescript 개발을 배우실 수 있습니다

2022년 4월 15일 · 0개의 댓글



by Arenacast

❤ 9

Powered by
Stellate