

# Optional

☰ 분류	JPA Java
📅 등록일	@2024/03/11

## “Optional을 만든 의도” - [공식 API 문서](#) 참조

메소드가 반환할 결과 값이 '없음'을 명백하게 표현할 필요가 있고, null을 반환하면 에러가 발생할 가능성이 높은 상황에서 메소드의 반환 타입으로 Optional을 사용하자는 것이 주된 목적이다. Optional 타입의 변수의 값은 절대 null이어서는 안되며, 항상 Optional 인스턴스를 가리켜야 한다.

JAVA 8 버전에서 등장한 기능으로 고질적인 'null 처리' 문제를 잘 다룰 수 있게 도와주는 클래스이다.

Optional 클래스를 사용하지 않았을 때 null을 사용하는 방식을 알아보자.

```
class Aclass {  
    String str;  
  
    String getStr(){
```

**조건문을 이용한 Null 처리**

```

        if(str!=null) {
            return str;
        }
        return "str is null";
    }
}

```

if문 등과 같이 조건문을 통한 null처리는 핵심 로직과 관련된 코드의 가독성을 해칠 수 있다는 단점이 있다.

```

try {
    System.out.println(a.getStr());
} catch (NullPointerException e) {
    System.out.println("str is null");
}

```

**try catch 문을 이용한 null 처리**

NullPointerException이 발생할만한 상황을 모두 예측해야 한다.

## Optional 클래스란?

Optional이란 'null일수도 있는 객체'를 감싸는 일종의 Wrapper 클래스이다.

**Optional<T> optional**

즉, 이러한 optional 변수 내부에는 null이 아닌 T 객체가 있을 수도 있고 null이 있을 수도 있다. 따라서, Optional 클래스는 여러 가지 API를 제공하여 null일수도 있는 객체를 다룰 수 있도록 돕는다.

### 1) Optional 객체 생성

**Optional.empty()** : 비어있는 Optional 객체로 생성한다. 따라서 이 내부에 있는 객체는 null이다. 그리고 이 객체는 Optional 내부적으로 미리 생성해 놓은 Singleton 인스턴스이다.

```
Optional<Member> optMember = Optional.empty();
```

`Optional.of(value)` : 인자 `value`를 담고 있는 `Optional` 객체로 생성합니다. 이 객체는 미리 만들어 놓은 `null`이 아닌 객체를 넘기면 되는데, 만약 인자로 넘긴 객체가 `null`이라면 `NullPointerException`이 발생할 수 있다.

```
Optional<Member> optMember = Optional.of(member);
```

`Optional.ofNullable(value)` : `null`일수도 있고 아닐 수도 있는 객체를 담아 `Optional` 객체를 생성합니다. 만약 `member`가 `null`인지 `not null`인지 확신이 서지 않는다면 이 방법으로 `Optional` 객체를 생성하면 된다.

```
Optional<Member> optMember = Optional.ofNullable(member);
```

## 2) Optional 객체 접근

`get()` : `Optional` 내부에 담긴 객체를 반환한다. 만약 `null`인 객체라면 `NoSuchElementException`이 발생합니다. 따라서 `isPresent()`로 체크한 후에 이 `get` 메서드를 사용했다.

```
Member member = optMember.get();
```

`orElseThrow(Supplier<? extends X> exceptionSupplier)` : `Optional` 내부에 담긴 객체가 `null`이 아니라면 담겨 있는 객체를 반환하고, `null`이라면 인자로 넘겨준 함수형 인자를 통해 생성된 예외를 발생시킨다.

```
Member member = optMember.orElseThrow(NullPointerException::new);
```

`orElse(T other)` : `Optional` 내부에 담긴 객체가 `null`이 아니라면 담겨있는 객체를 반환하고, `null`이라면 인자로 넘겨준 `member1`이라는 객체를 반환한다.

```
Member member = optMember.orElse(member1);
```

`orElseGet(Supplier<? extends T> other)` : Optional 내부에 담긴 객체가 null이 아니라면 담겨있는 객체를 반환하고, null이라면 넘겨준 함수형 인자를 통해 생성된 객체를 반환한다.


## Optional 올바르게 사용하기!

1. Optional에 null 할당 금지
2. Optional.get() 호출 전에 값을 가지고 있음을 확실히
3. 값이 없을 땐 orElse() , orElseGet() , orElseThrow() 처리
4. 값이 없는 경우 아무 동작도 하지 않는다면 ifPresent() 활용
5. isPresent() - get() 은 orElseXXX 등으로 대체
6. 필드의 타입 및 생성자나 메소드 인자로 Optional 사용 금지
7. 단지 값을 얻는 목적이면 Optional 대신 null 비교
8. Optional 대신 빈 컬렉션 반환
9. Optional 을 컬렉션의 원소로 사용 금지
10. of() 와 ofNullable() 혼동 금지
11. 원시 타입의 Optional은 OptionalInt , OptionalLong , OptionalDouble 사용
12. 내부 값 비교는 Optional.equals 사용을 고려
13. 제약 사항이 있는 경우 filter 사용 고려

## ? Reference

### Optional 클래스 사용하기


스프링 데이터 JPA를 사용하며 CrudRepository의 findById 메서드 리턴 타입인 Optional 클래스에 처음 접하게 되었습니다. Optional은 Java 8에 추가된 새로운 API로 이전에 하던 '고통스러운 null 처리'를 '잘' 다룰 수 있게 도와

 <https://dbbymoon.tistory.com/3>

```
y its id.  
  
{@literal null}.  
th the given id or {@literal Optional  
ntException if {@code id} is {@literal  
id);
```

### [Java] Optional 올바르게 사용하기

개요 Java 언어 설계자인 Brian Goetz는 Optional 을 만든 의도를 다음과 같이 공식 API 문서에 작성해 두었다. API Note: Optional is primarily intended for use as a method return type where there is a clear


 <https://dev-coco.tistory.com/178>

```
/*  
 * If a value is present, returns the value, otherwise throws  
 * {@code NoSuchElementException}.  
 *  
 * @return the non-{@code null} value described by this {@code Optional}  
 * @throws NoSuchElementException if no value is present  
 * @since 1.0  
 */  
public T orElseThrow() {  
    if (value == null) {  
        throw new NoSuchElementException("No value present");  
    }  
    return value;  
}
```

Optional을 올바르게 사용하는 방법 26가지(꼭 참고해서 보기)

### 26 Reasons Why Using Optional Correctly Is Not Optional - DZone

We take a look at the top 25 most important concepts to keep in mind when working with Optionals in Java, focusing on null variables and more.

 <https://dzone.com/articles/using-optional-correctly-is-not-optional>



Optional을 올바르게 사용하는 방법 26가지(원본)