Java Optional Api

2022-02-12 이해령

OptioanI이란? (*JDK 1.8부터*)

- 최종 연산의 결과를 OptioanI이라는 객체에 담아서 반환하기 위한 클래스
- 복잡한 조건문 없이 OptioanI method를 이용하여 간결한 코드 작성을 도와줌
- '**T타입의 객체**'를 감싸는 **래퍼 클래스 (지네릭스)**
  - 모든 타입의 **참조변수**를 담을 수 있다.

```
public final class Optional<T> {
    Common instance for empty().
    private static final Optional<?> EMPTY = new Optional<>();

    If non-null, the value; if null, indicates no value is present
    private final T value;
```

지네릭스(Generics)

- 지네릭스는 다양한 타입의 객체들을 다루는 메서드나 컬렉션 클래스의 컴파일 시 타입 체크를 해주는 기능, 타입체크와 형변환을 생략하여 코드가 간결해짐.
-
```
public interface List<E> extends Collection<E> {
```
-
```
@Test
void genericsTest() {
    List list = new ArrayList<>();
    list.add(10);
    list.add("10");

    int one = (int) list.get(0); // 10
    int two = (int) list.get(1); // "10"
}
```
```
@Test
void genericsTest2() {
    List<Integer> list = new ArrayList<>();
    list.add(10);
    list.add("10");

    int one = list.get(0); // 10;
    int two = list.get(1); // "10"
}
```

지네릭스 사용 여부로 변경되는 코드

```java
List list = new ArrayList<>();
list.add(10);
list.add("10");

for (Object o : list) {
    if (o instanceof Integer) {
        Integer inte = (Integer) o;
        System.out.println(inte.intValue());
    }

    if (o instanceof String) {
        String s = (String) o;
        System.out.println(s);
    }
}
```

```java
Optional<String> str = Optional.of("Test11");
System.out.println(str.get());

Optional str1 = Optional.of("test");
Object o = str1.get();
if (o instanceof Integer) {
    Integer o1 = (Integer) o;
    System.out.println(o1);
} else if (o instanceof String) {
    String s1 = (String) o;
    System.out.println(s1);
}
```

Optional 생성하기

of(), ofNullable(), empty

- of()
  - null이 아님을 명시적으로 표현
  - null이 담기는 순간 NullPointerException
- ofNullable()
  - null값이 있을 수 있음
  - null일때 get()을 요청 시 NoSuchElementException
- Empty
  - 빈값으로 초기화
  - get()을 요청 시 NoSuchElementException

```java
@Test
void ofTest() {
    String of = null;
    assertThrows(NullPointerException.class, () -> Optional.of(of));

    String ofNullable = null;
    Optional<String> ofNullableTest = Optional.ofNullable(ofNullable);
    assertThrows(NoSuchElementException.class, () -> ofNullableTest.get());

    Optional<String> empty = Optional.empty();
    assertThrows(NoSuchElementException.class, () -> empty.get());
}
```

Optional 데이터 가져오기

get(), orElse(), orElseGet(), orElseThrow()

- get()
  - Optioanl의 값을 가져옴
- orElse()
  - Optioanl의 값이 Null이면 Else()안에 데이터를 가져옴
- orElseGet()
  - Optioanl의 값이 Null이면 Else()안에 데이터를 가져옴
- orElseThrow()
  - Optioanl의 값이 Null이면 Throw를 실행함.

orEles() vs orElseGet()

- orElse()
  - get()을 성공하여도 호출함.
  - Null 여부와 상관없이 무조건 호출
- orElseGet()
  - Null인 경우에만 호출함.
- 결론
  - Static 값을 가져오지 않는 한 무조건 orElseGet()이 성능적인 측면에서 좋다.

```java
@Test
void optionalTest2() {
    Optional<String> str = Optional.of("Test1");

    String test1 = str.get();
    String test2 = str.orElse(orElse());
    String test3 = str.orElseGet(() -> orElseGet());

    // test가 존재하기 때문에 "Test1"을 출력해야 한다.
    System.out.println(test1);
    System.out.println(test2);
    System.out.println(test3);
}

String orElse() {
    System.out.println("Get Else");
    return "orElse";
}

String orElseGet() {
    System.out.println("Get Else Get");
    return "orElseGet";
}
```

```
Get Else
Test1
Test1
Test1
```

orElseThrow()

- null일 경우 예외를 발생시킴. 아래와 같이 간결한 코드 사용 가능

```java
@Test
void ofElseThrow() {
    String name = Strings.EMPTY;

    if (name == null || name.isEmpty()) {
        throw new NoSuchElementException("Name not found");
    }

    Optional<String> optionalName = findName();
    if (!optionalName.isPresent()) {
        throw new NoSuchElementException("Name not found");
    }

    Optional<String> optionalName2 = findName();
    optionalName2.orElseThrow(() -> new NoSuchElementException("Name not found"));
}
```

감사합니다.