Cập nhật tháng 8 năm 2024

[Bài Đọc] Optional

Optional là một lớp được giới thiệu trong **Java 8** nằm trong gói java.util. Mục đích của nó là giúp giảm thiểu lỗi NullPointerException và làm cho code trở nên an toàn hơn khi làm việc với các giá trị có thể là null.

1. **Optional là gì**
- Optional là một container (hộp chứa) có thể chứa hoặc không chứa một giá trị. Nó đại diện cho một giá trị tùy chọn:
    - **Nếu giá trị có tồn tại**, nó sẽ được đóng gói bên trong đối tượng Optional
    - **Nếu giá trị không tồn tại**, Optional sẽ là rỗng (empty)

2. **Tại sao cần Optional**
- Trước Java 8, khi làm việc với các giá trị có thể là null, chúng ta thường phải kiểm tra bằng cách:

```
if (object != null) {
    // Do something
}
```

- Nếu bỏ qua kiểm tra này, việc gọi phương thức trên một giá trị null có thể dẫn đến lỗi NullPointerException
- Optional giúp loại bỏ nhu cầu kiểm tra thủ công bằng cách cung cấp các phương thức tiện lợi và an toàn

3. **Các phương thức của Optional**
- Kiểm tra sự tồn tại của giá trị
    - **isPresent()**: Trả về true nếu giá trị tồn tại, ngược lại trả về false

```
if (optional.isPresent()) {
    System.out.println(optional.get());
}
```

- **isEmpty()** (Java 11+): Trả về true nếu Optional rỗng

```
if (optional.isEmpty()) {
    System.out.println("No value present");
}
```

- Lấy giá trị
  - **get()**: Lấy giá trị từ Optional. Nếu Optional rỗng, sẽ ném ra NoSuchElementException

```
String value = optional.get();
```

- **orElse()**: Trả về giá trị mặc định nếu Optional rỗng

```
String value = optional.orElse( other: "Default Value");
```

- **orElseGet()**: Tương tự orElse, nhưng nhận một Supplier để tạo giá trị mặc định

```
String value = optional.orElseGet(() -> "Generated Value");
```

- **orElseThrow()**: Ném ngoại lệ nếu Optional rỗng

```
String value = optional.orElseThrow(() -> new IllegalArgumentException("No value present"));
```

- Xử lý giá trị (Nếu tồn tại)
  - **ifPresent()**: Thực hiện hành động nếu giá trị tồn tại

```
optional.ifPresent( String value -> System.out.println("Value: " + value));
```

- Biến đổi giá trị
  - **map()**: Biến đổi giá trị bên trong Optional và trả về một Optional mới

```
Optional<String> upperCaseName = optional.map(String::toUpperCase);
```

- **flatMap()**: Tương tự map(), nhưng tránh lồng nhau của Optional

```
Optional<Integer> length = optional.flatMap( String value -> Optional.of(value.length()));
```

- Kết hợp Optional
  - **filter()**: Lọc giá trị trong Optional theo một điều kiện

```
Optional<String> filtered = optional.filter( String value -> value.startsWith("J"));
```

4. **Lợi ích của Optional**

- Giảm thiểu lỗi NullPointerException
- Làm cho mã nguồn rõ ràng, dễ hiểu và an toàn hơn
- Tận dụng phong cách lập trình hàm

Link tài nguyên đọc thêm: https://www.javatpoint.com/java-8-stream

Danh sách các bài học