Mastering Optional In Java

Eliminates the null checks

→

The Problem :

You need to send a email to a user

I'm sure many of you are doing that in below way !

```
if (user != null &&
          user.getEmail() != null) {
    sendEmail(user.getEmail());
}
```

Cons :

It works but null check is repetitive and not scalable

Solution

Use Optional : Feature introduced in Java 8

```
Optional.ofNullable(user)
        .map(User::getEmail)
        .ifPresent(this::sendEmail);
```

Pros :

🎯 Null-safe

📉 Reduced boilerplate

📖 Functional & readable code

Let’s Beak it Down →

Breakdown of Code

```
//wraps null-safe object
Optional.ofNullable(user)

     //transforms if present
     .map(User::getEmail)

     //executes only if non-null
     .ifPresent(this::sendEmail) ;
```

Mistakes to Avoid →

Mistakes to Avoid :

```
Optional<String> email =
Optional.ofNullable(user.getEmail());
if (email.isPresent()) {
    sendEmail(email.get());
}
```

Pros :

❌ You're still doing manual null checks

✅ Use operator functions like map and ifPresent instead!

When to use optional?

When to use optional ?

- ✅ Method return types
- ✅ Value transformation
- ✅ Safer chaining

❌ Avoid for fields or parameters (overhead)

→ Bonus tip

Bonus Tip

Chain multiple checks and operations

```
Optional.ofNullable(user)
        .flatMap(User::getProfile)
        .map(Profile::getEmail)
        .ifPresent(this::sendEmail);
```

Make your Java code null-safe, expressive, and future-proof.

The End