JAVA OPTIONAL

NULL SAFETY VE BEST PRACTICES

Optional Creation, Transformation, Best Practices

GİRİŞ: OPTIONAL NEDİR?

Optional Nedir?

Optional = Bir değerin var olup olmadığını ifade eden container

Sorun: NullPointerException

```
// Klasik sorun
String isim = getIsim();  // null dönebilir
int uzunluk = isim.length();  // NullPointerException!
```

Çözüm: Optional

```
// Optional ile
Optional<String> isim = getIsimOptional();
int uzunluk = isim.map(String::length).orElse(0);  // Güvenli!
```

Neden Optional?

- NullPointerException'dan kurtulma
- Null kontrollerini daha okunabilir yapma
- Değerin yokluğunu açıkça ifade etme
- Fonksiyonel programlama desteği

1. OPTIONAL OLUŞTURMA

1.1 Optional Oluşturma Yolları

```
import java.util.Optional;

public class OptionalOlusturma {
    public static void main(String[] args) {

        // 1. BOŞ OPTIONAL
        Optional<String> bos = Optional.empty();
        System.out.println("Boş: " + bos);
        System.out.println("Var mı? " + bos.isPresent());

        // 2. DEĞER İLE OPTIONAL - of()
        Optional<String> dolu = Optional.of("Merhaba");
        System.out.println("Dolu: " + dolu);

        // Optional.of(null) → NullPointerException!
        try {
            Optional<String> hata = Optional.of(null);
        } catch (NullPointerException e) {
            System.out.println("❌ of(null) hata verdi!");
        }

        // 3. NULL OLABİLİR - ofNullable()
        Optional<String> belki1 = Optional.ofNullable("Java");
        Optional<String> belki2 = Optional.ofNullable(null);

        System.out.println("ofNullable('Java'): " +
            belki1.isPresent());
        System.out.println("ofNullable(null): " +
            belki2.isPresent());
    }
}
```

Çıktı:

```
Boş: Optional.empty
Var mı? false
Dolu: Optional[Merhaba]
❌ of(null) hata verdi!
ofNullable('Java'): true
ofNullable(null): false
```

2. DEĞER KONTROLÜ VE ALMA

2.1 isPresent() ve isEmpty()

```
import java.util.Optional;

public class DegerKontrol {
    public static void main(String[] args) {

        Optional<String> dolu = Optional.of("Java");
        Optional<String> bos = Optional.empty();

        // isPresent() - Değer var mı?
        System.out.println("Dolu var mı? " + dolu.isPresent());
        System.out.println("Boş var mı? " + bos.isPresent());

        // isEmpty() - Değer yok mu? (Java 11+)
        System.out.println("Dolu boş mu? " + dolu.isEmpty());
        System.out.println("Boş boş mu? " + bos.isEmpty());

        // ESKİ YOL
        if (dolu.isPresent()) {
            String deger = dolu.get();
            System.out.println(deger.toUpperCase());
        }

        // YENİ YOL - ifPresent()
        dolu.ifPresent(deger ->
            System.out.println(deger.toUpperCase()));
    }
}
```

2.2 get() - Değeri Al (TEHLİKELİ!)

```
import java.util.Optional;

public class GetMetodu {
    public static void main(String[] args) {

        Optional<String> dolu = Optional.of("Merhaba");
        Optional<String> bos = Optional.empty();

        // DOĞRU - Önce kontrol et
        if (dolu.isPresent()) {
            String deger = dolu.get();
            System.out.println("✓ Değer: " + deger);
        }

        // YANLIŞ - Kontrol etmeden get()
        try {
```

```
            String deger = bos.get();  // NoSuchElementException!
        } catch (Exception e) {
            System.out.println("❌ Hata: " +
                e.getClass().getSimpleName());
        }

        // ÖNERİ: get() yerine başka metodlar kullan!
        System.out.println("orElse(): " +
            bos.orElse("Varsayılan"));
    }
}
```

2.3 orElse() ve orElseGet()

```
import java.util.Optional;

public class OrElseOrnek {

    public static String pahaliFonksiyon() {
        System.out.println("  → Pahalı fonksiyon çağrıldı!");
        return "Hesaplandı";
    }

    public static void main(String[] args) {

        Optional<String> dolu = Optional.of("Var");
        Optional<String> bos = Optional.empty();

        // orElse() - Değer yoksa varsayılan döner
        System.out.println("Dolu: " +
            dolu.orElse("Varsayılan"));
        System.out.println("Boş: " +
            bos.orElse("Varsayılan"));

        // orElseGet() - Supplier çalışır
        System.out.println("Dolu: " +
            dolu.orElseGet(() -> "Üretildi"));
        System.out.println("Boş: " +
            bos.orElseGet(() -> "Üretildi"));

        // FARK
        System.out.println("orElse() - DOLU:");
        dolu.orElse(pahaliFonksiyon());
        // Fonksiyon YINE DE çağrılır!

        System.out.println("orElseGet() - DOLU:");
        dolu.orElseGet(() -> pahaliFonksiyon());
        // Fonksiyon ÇAĞRILMAZ!
    }
}
```

2.4 orElseThrow() - Exception Fırlat

```
import java.util.Optional;

public class OrElseThrowOrnek {
    public static void main(String[] args) {

        Optional<String> dolu = Optional.of("Java");
        Optional<String> bos = Optional.empty();

        // Değer varsa al
        String deger = dolu.orElseThrow();
        System.out.println("✓ Değer: " + deger);

        // Değer yoksa exception
        try {
            bos.orElseThrow();
        } catch (Exception e) {
            System.out.println("❌ Hata");
        }

        // ÖZEL EXCEPTION
        try {
            bos.orElseThrow(() ->
                new IllegalArgumentException("Değer yok!"));
        } catch (IllegalArgumentException e) {
            System.out.println("❌ " + e.getMessage());
        }
    }
}
```

3. DÖNÜŞÜM VE FİLTRELEME

3.1 map() - Dönüştürme

```
import java.util.Optional;

public class MapOrnek {
    public static void main(String[] args) {

        Optional<String> isim = Optional.of("java");

        // Büyük harfe çevir
        Optional<String> buyuk = isim.map(String::toUpperCase);
        System.out.println("Büyük harf: " + buyuk.get());

        // Uzunluk al
        Optional<Integer> uzunluk = isim.map(String::length);
        System.out.println("Uzunluk: " + uzunluk.get());

        // ZİNCİRLEME
        String sonuc = Optional.of("  merhaba  ")
                               .map(String::trim)
                               .map(String::toUpperCase)
                               .orElse("Yok");
        System.out.println("Zincirleme: " + sonuc);

        // BOŞ OPTIONAL İLE
        Optional<String> bos = Optional.empty();
        String sonuc2 = bos.map(String::toUpperCase)
                           .orElse("Değer yok");
        System.out.println("Boş: " + sonuc2);
    }
}
```

3.2 flatMap() - İç İçe Optional

```
import java.util.Optional;

class Adres {
    private String sehir;

    public Adres(String sehir) {
        this.sehir = sehir;
    }

    public Optional<String> getSehir() {
        return Optional.ofNullable(sehir);
    }
}
```

```
class Kullanici {
    private String isim;
    private Adres adres;

    public Kullanici(String isim, Adres adres) {
        this.isim = isim;
        this.adres = adres;
    }

    public Optional<Adres> getAdres() {
        return Optional.ofNullable(adres);
    }
}

public class FlatMapOrnek {
    public static void main(String[] args) {

        Kullanici k1 = new Kullanici("Ali",
            new Adres("İstanbul"));

        // flatMap() kullanımı
        Optional<String> sehir = Optional.of(k1)
                                         .flatMap(Kullanici::getAdres)
                                         .flatMap(Adres::getSehir);

        System.out.println("Şehir: " +
            sehir.orElse("Bilinmiyor"));
    }
}
```

3.3 filter() - Filtreleme

```
import java.util.Optional;

public class FilterOrnek {
    public static void main(String[] args) {

        Optional<String> isim = Optional.of("Java");

        // Filtreleme
        Optional<String> sonuc1 = isim.filter(s -> s.length() > 3);
        System.out.println("Uzun mu? " + sonuc1.isPresent());

        Optional<String> sonuc2 = isim.filter(s -> s.length() > 10);
        System.out.println("Çok uzun mu? " + sonuc2.isPresent());

        // YAŞ KONTROLÜ
        int yas = 25;
        String mesaj = Optional.of(yas)
                               .filter(y -> y >= 18)
                               .map(y -> "✓ Yetişkin")
```

```
                                .orElse("✗ Reşit değil");
        System.out.println(mesaj);
    }
}
```

4. GERÇEK HAYAT ÖRNEKLERİ

4.1 Veritabanı Kullanıcı Arama

```
import java.util.*;

class User {
    private int id;
    private String name;
    private String email;
    private Integer age;

    public User(int id, String name, String email, Integer age) {
        this.id = id;
        this.name = name;
        this.email = email;
        this.age = age;
    }

    public int getId() { return id; }
    public String getName() { return name; }
    public Optional<Integer> getAge() {
        return Optional.ofNullable(age);
    }
}

class UserRepository {
    private Map<Integer, User> database = new HashMap<>();

    public UserRepository() {
        database.put(1, new User(1, "Ali", "ali@ex.com", 25));
        database.put(2, new User(2, "Ayşe", "ayse@ex.com", null));
    }

    public Optional<User> findById(int id) {
        return Optional.ofNullable(database.get(id));
    }
}

public class UserSearchOrnek {
    public static void main(String[] args) {

        UserRepository repo = new UserRepository();

        // BAŞARILI ARAMA
        Optional<User> user1 = repo.findById(1);
        user1.ifPresent(u -> System.out.println("Bulundu: " + u));

        // BAŞARISIZ ARAMA
        Optional<User> user2 = repo.findById(999);
```

```
        User varsayilan = user2.orElse(
            new User(0, "Guest", "guest@ex.com", null));

        // EXCEPTION
        try {
            repo.findById(888).orElseThrow(() ->
                new RuntimeException("User 888 bulunamadı!"));
        } catch (RuntimeException e) {
            System.out.println("❌ " + e.getMessage());
        }
    }
}
```

4.2 Konfigürasyon Yönetimi

```
import java.util.*;

public class ConfigOrnek {

    private static Map<String, String> config = new HashMap<>();

    static {
        config.put("database.url", "localhost:5432");
        config.put("app.name", "MyApp");
        config.put("app.timeout", "30");
    }

    public static Optional<String> getConfig(String key) {
        return Optional.ofNullable(config.get(key));
    }

    public static void main(String[] args) {

        // VAR OLAN DEĞER
        String dbUrl = getConfig("database.url")
                           .orElse("localhost:3306");
        System.out.println("DB URL: " + dbUrl);

        // OLMAYAN DEĞER
        String dbPassword = getConfig("database.password")
                                .orElse("default_password");
        System.out.println("DB Password: " + dbPassword);

        // DÖNÜŞÜM
        String appName = getConfig("app.name")
                             .map(String::toUpperCase)
                             .orElse("UNKNOWN");
        System.out.println("App Name: " + appName);
    }
}
```

5. OPTIONAL BEST PRACTICES

5.1 Yapılması Gerekenler

- ✅ Dönüş değeri olarak kullan
- ✅ orElseGet() tercih et (lazy evaluation)
- ✅ map() ve filter() ile dönüşüm yap
- ✅ ifPresent() kullan (isPresent() + get() yerine)

```
// ✅ DOĞRU: Optional dönen metod
public Optional<String> findUser(int id) {
    return Optional.ofNullable(database.get(id));
}

// ✅ DOĞRU: orElseGet() lazy evaluation
String name = findUser(1)
    .orElseGet(() -> generateDefaultName());

// ✅ DOĞRU: map() ile dönüşüm
Optional<Integer> age = findUser(1)
    .map(User::getAge);

// ✅ DOĞRU: ifPresent() kullan
findUser(1).ifPresent(user -> process(user));
```

5.2 Yapılmaması Gerekenler

- ❌ Metod parametresi olarak kullanma
- ❌ Class field'ı olarak kullanma
- ❌ Collection elemanı olarak kullanma
- ❌ get() direkt kullanma
- ❌ Optional.of(null) kullanma

```
// ❌ YANLIŞ: Optional parametre
public void processUser(Optional<String> username) {
    // Kullanma!
}

// ✅ DOĞRU: Normal parametre
public void processUser(String username) {
    if (username != null) {
        // işlem
    }
}

// ❌ YANLIŞ: Optional field
class User {
    private Optional<String> name;  // Kullanma!
```

```
}

// ✅ DOĞRU: Nullable field + Optional getter
class User {
    private String name;

    public Optional<String> getName() {
        return Optional.ofNullable(name);
    }
}
```

6. NEDEN OPTIONAL PARAMETRE KULLANILMAZ?

6.1 API Karmaşıklığı

```
// ❌ KÖTÜ: Optional parametre
public void processUser(Optional<String> username) {
    username.ifPresent(name -> {
        System.out.println(name);
    });
}

// Çağrı karmaşık:
processUser(Optional.of("Ali"));
processUser(Optional.empty());
processUser(Optional.ofNullable(null));

// ✅ İYİ: Normal parametre
public void processUser(String username) {
    if (username != null) {
        System.out.println(username);
    }
}

// Çağrı basit:
processUser("Ali");
processUser(null);
```

6.2 Gereksiz Nesne Oluşturma

```
// ❌ KÖTÜ: Her çağrıda Optional nesnesi
public void process(Optional<String> name) { }

for (int i = 0; i < 1_000_000; i++) {
    // 1 milyon Optional nesnesi oluşur!
    process(Optional.of("Ali"));
}

// ✅ İYİ: Direkt parametre
public void process(String name) { }

for (int i = 0; i < 1_000_000; i++) {
    // Sadece String referansı
    process("Ali");
}
```

6.3 Null Geçilebilir!

```
// ❌ KÖTÜ: Optional bile null olabilir!
public void process(Optional<String> name) {
    // name null olabilir!
```

```
    if (name != null) {  // Yine null kontrolü!
        name.ifPresent(System.out::println);
    }
}

// Çağrı:
process(null);  // Bu geçerli!

// ✅ İYİ: Açık ve net
public void process(String name) {
    if (name != null) {
        System.out.println(name);
    }
}
```

6.4 Optional'ın Amacına Ters

```
// ✅ DOĞRU: Dönüş değeri - Belirsizlik var
public Optional<User> findUser(int id) {
    // Kullanıcı bulunabilir veya bulunamayabilir
    return Optional.ofNullable(database.get(id));
}

// ❌ YANLIŞ: Parametre - Çağıran zaten biliyor
public void processUser(Optional<User> user) {
    // Çağıran zaten Optional oluşturmuş
    // Demek ki değerin var olup olmadığını BİLİYOR
}
```

6.5 Alternatifler

```
// 1. NULL KABUL ET
public void sendEmail(String to, String subject) {
    // subject null olabilir
    if (subject == null) {
        subject = "No Subject";
    }
}

// 2. METHOD OVERLOADING
public void sendEmail(String to) {
    sendEmail(to, "No Subject");
}

public void sendEmail(String to, String subject) {
    // Gerçek implementasyon
}

// 3. BUILDER PATTERN
new EmailBuilder("user@example.com")
    .subject("Hello")
```

```
    .body("World")
    .send();
```

SONUÇ

Bu dokümanda Java Optional'ın tüm önemli konularını detaylı olarak inceledik:

- **Optional Oluşturma:** empty(), of(), ofNullable()
- **Değer Kontrolü:** isPresent(), isEmpty(), get(), orElse(), orElseGet()
- **Dönüşüm:** map(), flatMap(), filter()
- **Best Practices:** Doğru ve yanlış kullanımlar
- **Optional Parametre:** Neden kullanılmaz, alternatifler

Optional, NullPointerException'dan kaçınmak ve null güvenliğini sağlamak için güçlü bir araçtır. Doğru kullanıldığında kodunuz daha okunabilir ve güvenli olur.

ÖNEMLİ HATIRLATMALAR

- Optional sadece dönüş değeri için kullan
- get() yerine orElse/orElseGet kullan
- orElseGet() lazy evaluation sağlar (daha verimli)
- Optional.of(null) kullanma, ofNullable() kullan
- Optional parametre/field/collection elemanı kullanma