Infozettel: Optional

Matthias Eurich – 2017-04-04

Table of Contents

Einführung

Einige der Methoden von `Stream`, die den Datenfluss beenden (*terminale Operationen*), geben einen Rückgabewert vom generischen Typ `Optional<T>` zurück. Ein Objekt vom Typ `Optional<T>` ist ein Container für einen Wert vom Typ `T`. Durch das Verpacken des Wertes in einem Container wird eine direkte Rückgabe von `null` verhindert.

Beispiel:
Im nachfolgenden Code wird eine Liste von Namen gefiltert. Anschließend wird das erste Element zurückgegeben.

```
jshell> $1.stream().filter(n -> n.startsWith("L")).findFirst()
$2 ==> Optional[Liam]
```

Wie im Beispiel zu sehen ist, ist der Rückgabewert von `findFirst()` vom Typ `Optional` und enthält den Wert `Liam`. Würde der Datenstrom nach der Filterung keine Elemente mehr enthalten, würde die Ausgabe auf der JShell wie folgt aussehen:

```
jshell> $1.stream().filter(n -> n.startsWith("X")).findFirst()
$3 ==> Optional.empty
```

Das Objekt vom Typ `Optional` ist als leer (empty) gekennzeichnet.

`isPresent()`

Mit der Methode `public boolean isPresent()` kann überprüft werden, ob das Objekt einen Wert beinhaltet.

Beispiel:
Rufen wir `isPresent()` von dem Objekt mit der Bezeichnung `$2` aus dem obrigen Beispiel auf, so gibt uns `isPresent()` den Wert `true` zurück. Das Objekt beinhaltet also einen Wert.

```
jshell> $2.isPresent()
$4 ==> true
```

Im Falle von `$3` wird der Wert `false` zurückgegeben.

```
jshell> $3.isPresent()
$5 ==> false
```

In klassischem Java-Code entspricht die Methode `isPresent()` einer Abfrage wie `if(x != null)`, wobei `x` hier der tatsächliche Wert und nicht der Container ist. Im Vergleich ist die Variante `if(optionalOfX.isPresent())` verständlicher und damit auf lange Sicht wartbarer.

Verwenden Sie zur Überprüfung, ob ein `Optional` einen Wert beinhaltet, immer `isPresent()`. Ein Vergleich wie `if (optionalOfX != Optional.empty())` vergleicht lediglich die Referenz, aber nicht den eigentlichen Wert eines `Optional`.

`get()`

Mit der Methode `public T get()` erhält man den Wert eines Objekts vom Typ `Optional`.

Beispiel:

```
jshell> $2.get()
$6 ==> "Liam"
```

Ein ungeprüfter Zugriff auf den Wert eines `Optional` führt zu einer `NoSuchElementException`, wenn das `Optional` leer ist.

`orElse()`

Die Methode `public T orElse(T other)` ist eine Zusammenfassung von `isPresent()` `get()` und `if else`. Wenn ein Wert vorhanden ist, wird dieser zurückgegeben. Falls kein Wert vorhanden ist, wird der als Parameter übergebene Wert zurückgegeben.

Beispiel:

```
jshell> $2.orElse("Kein Name gefunden")
$8 ==> "Liam"

jshell> $3.orElse("Kein Name gefunden")
$9 ==> "Kein Name gefunden"
```

Im Vergleich dazu die "ausprogrammierte" Version mit `isPresent()` und `get()`:

```
jshell> $2.isPresent() ? $2.get() : "Kein Name gefunden"
$10 ==> "Liam"

jshell> $3.isPresent() ? $2.get() : "Kein Name gefunden"
$11 ==> "Kein Name gefunden"
```

Last updated 2017-05-08 14:12:51 CEST