Optional class orElse() and orElseGet() methods

In previous posts we saw how to [avoid Null Pointer Exception](http://data-structure-learning.blogspot.com/2015/07/avoid-null-pointer-using-optional-class.html) using Optional<T> class, [Optional<T> class introduction](http://data-structure-learning.blogspot.com/2015/07/optional-class-introduction.html), Optional<T> class [of(), ofNullable() and empty()](http://data-structure-learning.blogspot.com/2015/07/optional-class-of-ofnullable-empty.html) method, [Optional<T> class ifPresent() method](http://data-structure-learning.blogspot.com/2015/07/optional-class-ifpresent-method.html), [Optional<T> class filter()](http://data-structure-learning.blogspot.com/2015/07/optional-class-filter-method.html) method and Optional<T> class [isPresent()](http://data-structure-learning.blogspot.com/2015/07/optional-class-ispresent-and-get-method.html) and [get()](http://data-structure-learning.blogspot.com/2015/07/optional-class-ispresent-and-get-method.html) method.

In this post we will explore orElse and orElseGet() methods.

First we will cover orElse() method.

|  |
| --- |
| **public** T orElse(T other) {  **return** value != **null** ? value : other;  }  Before I explain anything for this method just read the name once again. The name orElse() speaks for the job of this method.  This method returns the value if present otherwise it returns other.  The parameter other is the value that is to be returned of value is not present in this Optional<T> instance. |

List<String> days = **new** ArrayList<String>(Arrays.*asList*("Monday",

"Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"));

String prefix = "Wed";

**final** Optional<String> found = days.stream()

.filter(day -> day.startsWith(prefix))

.findFirst();

System.***out***.println(found.orElse("Not Found"));// Wednesday

Let is change the prefix to "abc"

List<String> days = **new** ArrayList<String>(Arrays.*asList*("Monday",

"Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"));

String prefix = "abc";

**final** Optional<String> found = days.stream()

.filter(day -> day.startsWith(prefix))

.findFirst();

System.***out***.println(found.orElse("Not Found"));// Not Found

Now orElseGet() method

|  |
| --- |
| **public** T orElseGet(Supplier<? **extends** T> other) {  **return** value != **null** ? value : other.get();  }  Returns the value if present.  Otherwise it will invoke get method of Supplier interface and return the result of the invocation. If Supplier is null then it throws NullPointerException. |

found.orElseGet(() -> doSomething());

Call doSomething() method with no parameters and returns the result of type T.

System.***out***.println(found.orElseGet(() -> "Not Found"));// Not Found

That’s all on methods orElse() and orElseGet().