Optional class *of()*, *OfNullable()*, *empty()* methods

In previous post we saw How to avoid Null Pointer Exception using Optional<T> class and Optional<T> Class demo.

This post is extension to previous post where we saw different ways of creating Optional<T> instances. In this post we will take deep dive into the code written in Optional<T> class itself.

First, the constructor.

**private** Optional() {

**this**.value = **null**;

}

Constructor is private and it creates empty instance. Generally only one empty instance should exist per VM.

Common Optional<> EMTPY instance

**private** **static** **final** Optional<?> ***EMPTY*** = **new** Optional<>();

Now we will take Optional<T> instance creation methods and explore it.

Optional.*of*(..) method

Optional<String> optional = Optional.*of*("Monday");

*of*(..) method definition

**public** **static** <T> Optional<T> of(T value) {

**return** **new** Optional<>(value);

}

of(T value) is a static method in Optional<T> class. It returns a new instance of Optional<> with specified non-null value. Remember if the value is null then it will throw NullPointerException.

Optional.*OfNullable*( . . ) method

**public** **static** <T> Optional<T> ofNullable(T value) {

**return** value == **null** ? *empty*() : *of*(value);

}

Unlike of(T value) method this method will not throw NullPointerException. If the value passed is null then it return EMPTY instance. If value is not null then it calls *of*(..) method and returns instance with specified value.

Optional.empty() method

**public** **static**<T> Optional<T> empty() {

Optional<T> t = (Optional<T>) ***EMPTY***;

**return** t;

}

This method just returns the EMPTY instance.