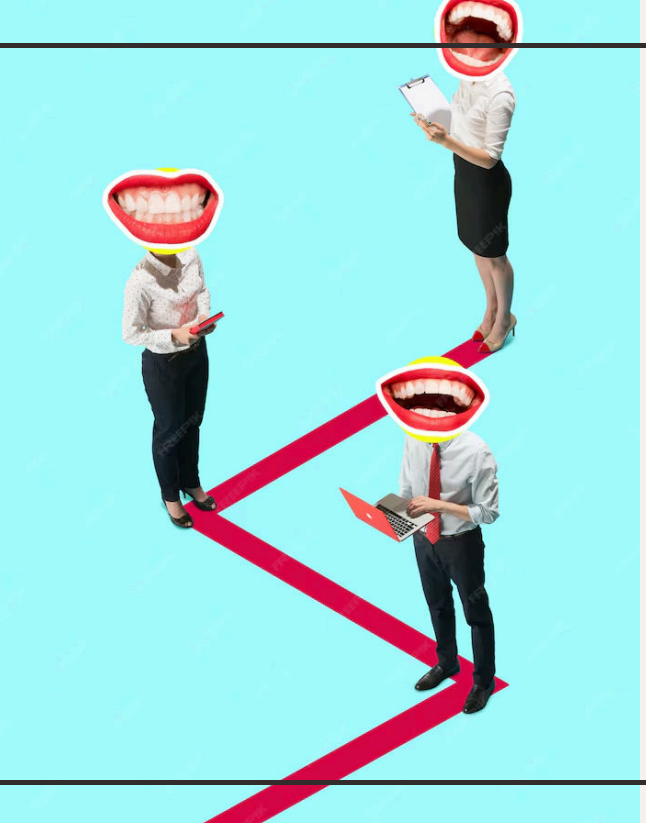
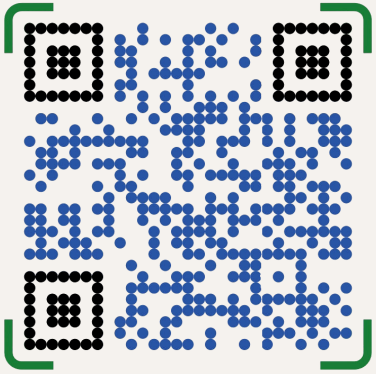
Exploring Vavr's Either: Java's Alternative to Scala's Either



Introduction

Vavr's Either is a powerful data type in Java that provides an alternative to Scala's Either. It allows for better error handling and functional programming in Java. With Either, you can represent two possible outcomes, success or failure, and handle them in a more concise and expressive way. In this presentation, we will explore the features and benefits of Vavr's Either.

SLIDO #2842836



INITIAL QUIZ ABOUT HANDLE EXCEPTIONS AND USE OF EITHER

LET'S GET TO KNOW A LITTLE

What is Either?

Either represents a value of one of two possible types. It is commonly used for error handling or representing two possible outcomes. In Vavr, Either is a functional alternative to Java's traditional try-catch approach. It provides a more declarative and composable way to handle errors and control flow.

Creating an Either

To create an Either in Vavr, you can use the static methods `Either.left()` and `Either.right()` to represent the left (failure) and right (success) values respectively. This allows you to explicitly define the two possible outcomes and handle them accordingly.

CODE

V1.0

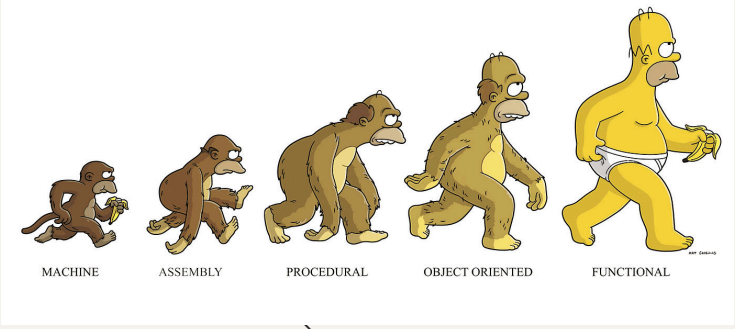
STEP 1 - EXAMPLE USE EITHER

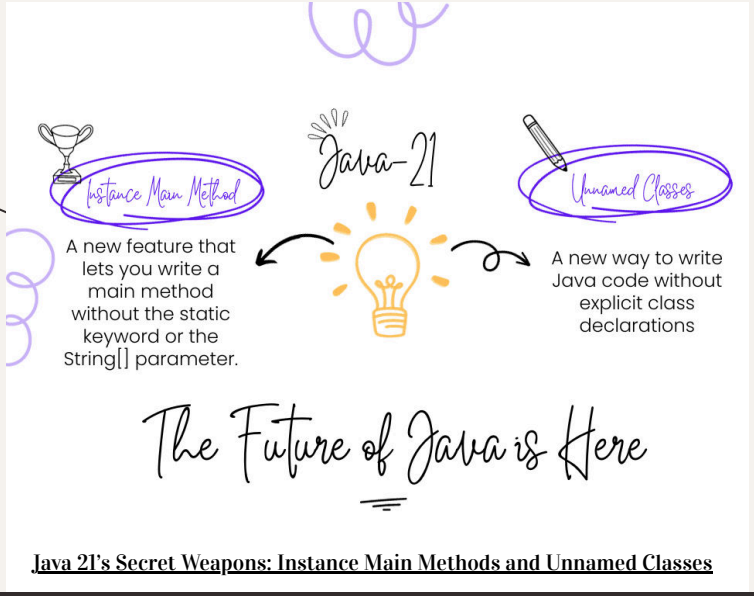
Functional Programming

Functional programming emphasizes:

1. Immutability.
2. Pure functions (Avoiding side effects).

In this context, functional values or objects cannot be modified (i.e., they are in normal form), which aligns with the concept of immutable variables in Java.





Java 21's Secret Weapons: Instance Main Methods and Unnamed Classes

Functional Programming

In functional programming, we strive for **pure functions** that have no side effects and are deterministic.

Pure functions always produce the same output for the same input, regardless of external state.

Traditional try-catch blocks in Java introduce side effects because they can alter program flow based on exceptions. They violate the purity of functions by mixing computation with error handling.

CODE

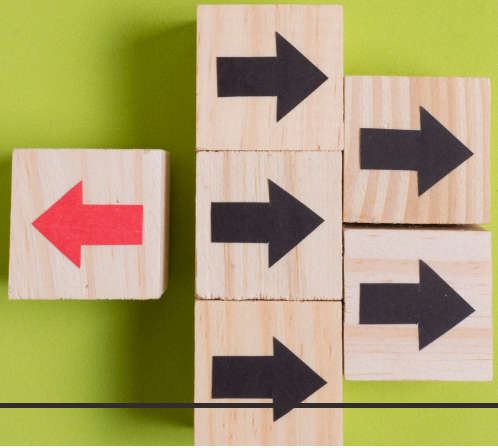
V2.0

STEP 2 - EXAMPLE CONTROLLER ADVICE

Working with Either

Once you have an Either, you can perform various operations on it. You can use `map()` and `flatMap()` to transform the right value, and `mapLeft()` and `flatMapLeft()` to transform the left value. This allows you to chain operations and handle both success and failure cases elegantly.

Either provides a convenient way to handle errors. You can use `getOrElse()` to get the right value or a default value in case of failure. Alternatively, you can use `orElse()` to provide an alternative Either in case of failure. This allows for concise and expressive error handling in Java.



Fold with Either

Vavr's Either supports `fold()` method allows us to handle both success and failure cases in a more functional and expressive way. You can define different actions based on the outcome of the Either, making your code more readable and maintainable.

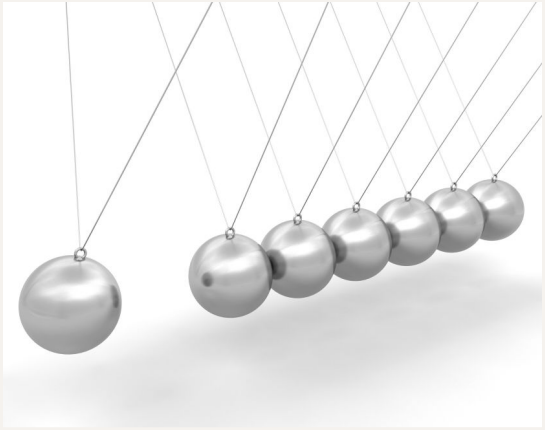
Vavr's Either seamlessly integrates with Java 8 features. You can use `ifPresent()` and `ifLeft()` to conditionally execute code based on the presence of a right or left value. This allows you to leverage the functional programming capabilities of Either while working with existing Java code.

CODE

V3.2

STEP 3.1 - MORE OPERATIONS WITH EITHER (JPA SPECIFICATION)

STEP 3.2 - MORE OPERATIONS WITH EITHER - CHECK METHODS



Comparison with Scala's Either

Vavr's Either provides a similar functionality to Scala's Either, but with a more Java-friendly syntax. It allows Java developers to leverage the power of Either without having to switch to Scala. Both options have their strengths and weaknesses, and the choice depends on the specific use case and project requirements.

Using Vavr's Either brings several benefits to Java developers. It promotes a more functional programming style, improves error handling, and provides a more expressive and concise way to handle two possible outcomes. With Either, you can write code that is more readable, maintainable, and robust.

Use Cases for Vavr's Either

Vavr's Either is particularly useful in scenarios where there are two possible outcomes, such as validation, data processing, or handling external dependencies. It allows you to handle success and failure cases explicitly, providing a clear and structured approach to error handling and control flow.

CODE

V4.1

STEP 4.1 - USE CASE, HANDLE ERROR

CODE

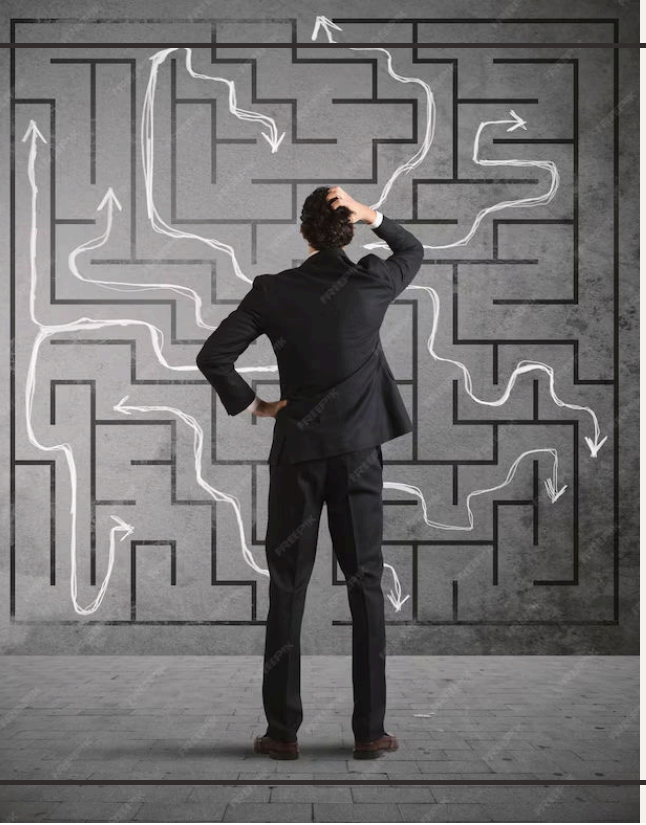
V4.2.1

STEP 4.2.1 - MULTIPLE VALIDATIONS. THE EXAMPLE DOES NOT CHAIN THE ERRORS

CODE

V4.2.2

STEP 4.2.2 - MULTIPLE VALIDATIONS. CHAIN ERRORS IN A LIST



Limitations of Vavr's Either

While Vavr's Either is a powerful tool, it has some limitations:

- It can become complex when dealing with multiple nested Eithers, and the syntax can be verbose compared to simpler error-handling approaches.
- Note that Either also exists in some AWS SDKs and that using Either from an external library requires permissions and support from the libraries.

It is important to consider these limitations and evaluate if Vavr's Either is the right choice for your specific use case.

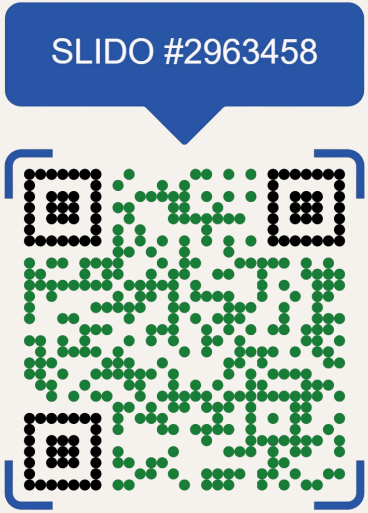
CODE

V5.0

STEP 5- CUSTOM EITHER, NO LIBRARIES

Conclusion

Vavr's Either is a valuable addition to Java's functional programming capabilities. It offers a concise and expressive way to handle two possible outcomes, improving error handling and control flow. By leveraging Either, Java developers can embrace a more functional programming style and write code that is more readable, maintainable, and robust.



FINAL QUIZ ABOUT HANDLE EXCEPTIONS AND USE OF EITHER

LET'S MAKE SURE THAT WE ARE LEFT WITH
THE MINIMUM IDEAS OF THIS TECH TALK

Thanks!

Do you have any question?

edison.gonzalez@globant.com
+57 304 3306803

Edison González

Edison González