Programación Funcional en Java

Círculo Siete Capacitación

Clase 9 de 12

15 Marzo 2025

Uso de Tuplas y Colecciones Inmutables en Vavr

- Vavr es una librería funcional para Java que proporciona estructuras inmutables y funcionales que mejoran la seguridad y la expresividad del código. Revisaremos dos de sus características más importantes:
  - Tuplas (Tuple) → Permiten agrupar múltiples valores sin necesidad de crear una clase.
  - Colecciones inmutables (List, Set, Map) → Evitan efectos secundarios y facilitan la concurrencia.

Uso de Tuplas (Tuple) en Vavr

- ¿Qué es una Tupla?
  - Una tupla es una estructura que almacena múltiples valores de diferentes tipos, sin necesidad de definir una clase.
- Ver Lab01.java

Tipos de Tuplas en Vavr

- Vavr proporciona tuplas de 2 a 8 elementos (Tuple2 hasta Tuple8).
- Ver Lab02.java

Transformación de Tuplas (map)

- Transformar valores en una tupla
  - ***map()*** permite modificar valores sin perder la inmutabilidad.
  - Permite transformar los valores en una sola operación.
- Ver Lab03.java

Uso de Colecciones Inmutables en Vavr

- ¿Qué son las colecciones inmutables?
  - Las colecciones inmutables son estructuras de datos que no pueden modificarse después de su creación. En lugar de modificar el objeto original, cada operación genera una nueva instancia.
- Beneficios:
  - Seguridad en concurrencia (thread-safe sin synchronized).
  - Evita efectos secundarios y mutaciones inesperadas.
  - Facilita la programación funcional y la reutilización de datos.

List<T> Inmutable en Vavr

- A diferencia de ***java.util.List***, esta lista no se puede modificar después de la creación.
- Ver Lab04.java

Set<T> Inmutable en Vavr

- Ver Lab05.java

Map<K, V> Inmutable en Vavr

- Ver Lab06.java

Notas finales

- Las tuplas y colecciones inmutables en Vavr facilitan la programación funcional y hacen el código más seguro.
- Beneficios de usar Vavr:
  - Tuplas (Tuple) → Evitan la creación de clases innecesarias.
  - Listas inmutables (List) → Evitan efectos secundarios y facilitan concurrencia.
  - Conjuntos (Set) y mapas (Map) inmutables → Seguridad en concurrencia sin bloqueos.
- Adoptar estas estructuras permite escribir código más limpio, seguro y funcional en Java.

Uso de Validation en Vavr para Validaciones Funcionales

- El manejo tradicional de validaciones en Java suele depender de if-else y excepciones, lo que puede hacer que el código sea imperativo y difícil de mantener.
- ¿Qué es Validation en Vavr?
- Validation es una estructura funcional de Vavr que permite:
  - Agrupar múltiples errores en una sola validación.
  - Evitar excepciones (try-catch).
  - Hacer validaciones composables y encadenadas.
- Ver Lab07.java y Lab08.java

Validation vs Either (¿Cuál usar?)

- Cuándo usar Validation?
  - Para formularios o validaciones con múltiples reglas (ejemplo: usuario, dirección, pago).
- ¿Cuándo usar Either?
  - Para representar éxito (Right) o error (Left) en una sola operación.

Diferencias clave entre Validation<E, T> y Either<L, R>

Característica	Validation<E, T>	Either<L, R>
Captura múltiples errores	Sí	No (solo el primero)
Combinación de validaciones (combine())	Sí	No
Uso principal	Validaciones de múltiples condiciones	Manejo de estados binarios (éxito/error)

Notas finales

- Validation<E, T> en Vavr permite manejar validaciones de manera funcional, composable y sin try-catch.
- Ventajas de Validation en Vavr:
  - Evita if-else innecesarios.
  - Permite combinar múltiples validaciones (combine()).
  - Maneja errores sin lanzar excepciones.
  - Más expresivo y funcional que try-catch.