vavr

Nieszablonowe programowanie funkcyjne.
Wprowadzenie.

Łukasz Krauzowicz

Kraków, 14.12.2017

O mnie

ailleron
your digital wings

Łukasz Krauzowicz

- lubi Javę i JVMa
- I niekończącą się eksplorację GitHuba :)
- Software Engineer w Ailleron

Kontakt

- M: lukasz.krauzowicz@ailleron.com
- GH: github.com/kraluk
- T: twitter.com/lkrauzo

Agenda

- Dlaczego?
- Jak używasz Stream API?
- **vavr**
  - **`Option`**
  - **`Match`** (Pattern Matching → JEP-305 / JDK 12+?)
  - **`Try`**
  - **Validation API**
  - **Collection Framework “Extensions”**
  - *`For` / `Either` / i jeszcze kilka rzeczy...*
- Alternatywy
- Minusy - czyli nie ma róży bez kolców

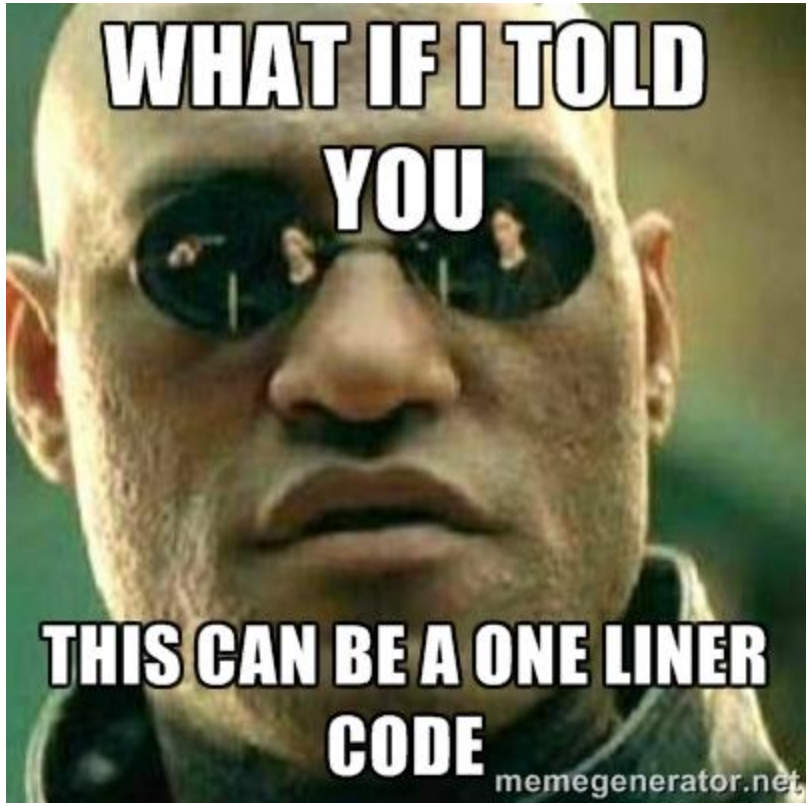
Dlaczego?


FUTURE YOU CAN
CREATE
WITH JAVA 8
memegenerator.net

Dlaczego?

- Programowanie funkcyjne
- Redundantność kodu
- Niezmienność (stałość)

Redundantność kodu


WHAT IF I TOLD YOU
THIS CAN BE A ONE LINER CODE
memegenerator.net

Przed - Przed

```
public static Dinner makeDinner(GrillService service) {
    Charcoal charcoal = service.getCharcoal();
    Lighter lighter = service.getLighter();

    if (charcoal != null && lighter != null) {
        Fire fire = service.lightFire(charcoal, lighter);
        CornCob cornCob = service.getCornCob();

        if (fire != null && corn!= null) {
            return service.grill(fire, cornCob);
        }
    }
    return null; // ???
}
```

Źródło: Vavr Blog

Przed

```
public static Optional<Dinner> makeDinner(GrillService service) {
    Optional<Charcoal> charcoalOptional = service.getCharcoal();
    Optional<Lighter> lighterOptional = service.getLighter();

    if (charcoalOptional.isPresent() && lighterOptional.isPresent()) {
        Optional<Fire> fireOptional = service.lightFire(
                charcoalOptional.get(), lighterOptional.get());
        Optional<CornCob> cornCobOptional = service.getCornCob();

        if (fireOptional.isPresent() && cornCobOptional.isPresent()) {
            return service.grill(fireOptional.get(),
                    cornCobOptional.get());
        }
    }
    return Optional.empty();}
```

Źródło: Vavr Blog

Po

```
public static Option<Dinner> makeDinner(GrillService service) {
    return
        For(service.getCharcoal(), charcoal ->
            For(service.getLighter(), lighter ->
                For(service.lightFire(charcoal, lighter), fire ->
                    For(service.getCornCob(), cornCob ->
                        For(service.grill(fire, cornCob))
                            .yield()
                    )
                )
            )
        ).toOption();
}
```

Źródło: Vavr Blog

Programowanie funkcyjne

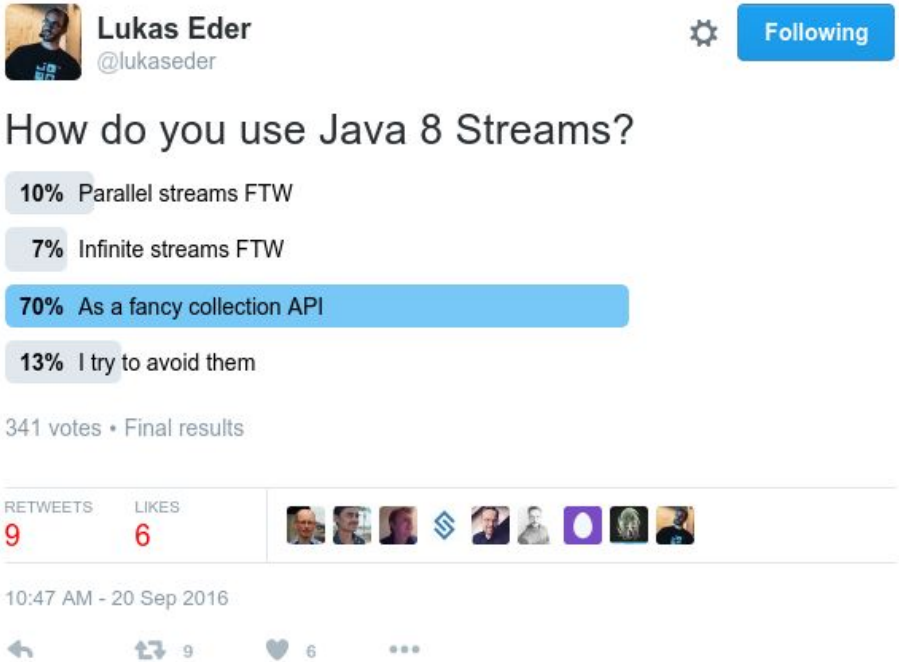
Scala

```scala
def makeDinner(service: GrillService): Option[Dinner] =
  for {
    charcoal <- service.getCharcoal
    lighter  <- service.getLighter
    fire     <- service.lightFire(charcoal, lighter)
    cornCob  <- service.getCornCob
    dinner   <- service.grill(fire, cornCob)
  } yield dinner
```

Źródło: Vavr Blog

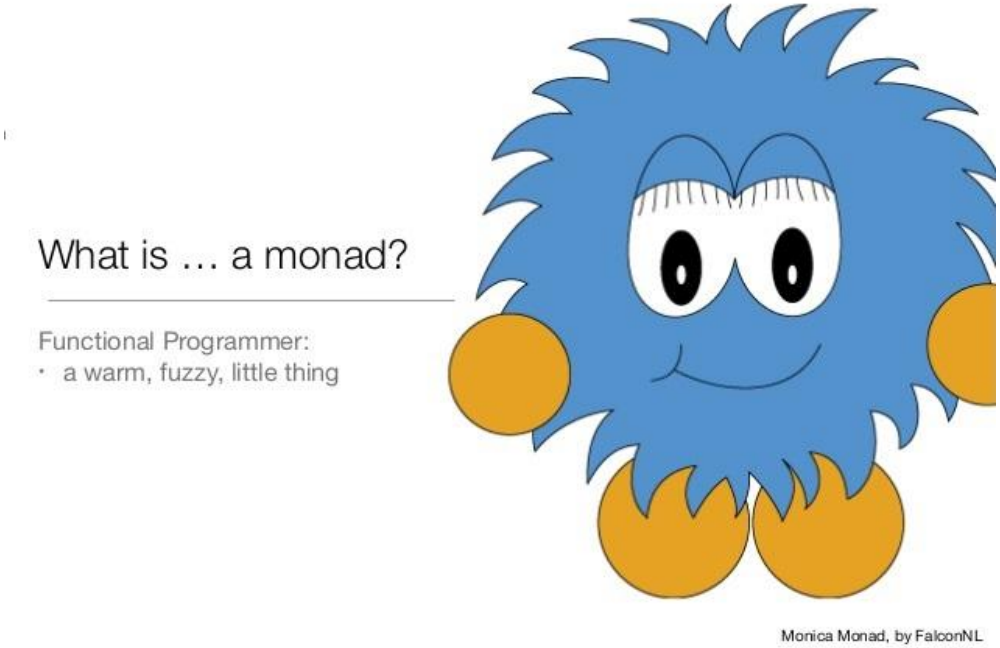
Stream API

Jak używasz Stream API?

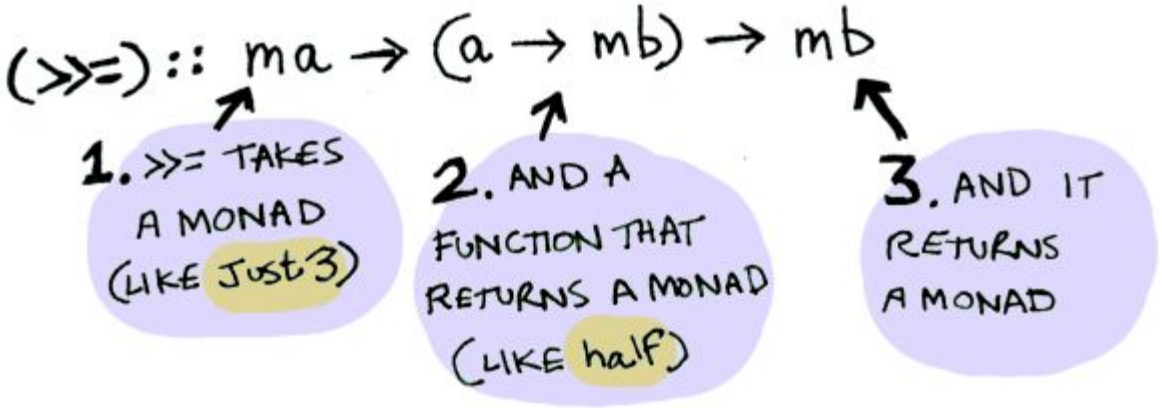


Dygresja

Monada


What is … a monad?
Functional Programmer:
• a warm, fuzzy, little thing
Monica Monad, by FalconNL

Monada


(>>=) :: ma → (a → mb) → mb
1. >>= TAKES A MONAD (LIKE Just 3)
2. AND A FUNCTION THAT RETURNS A MONAD (LIKE half)
3. AND IT RETURNS A MONAD


NO TO
JEDZIEMY Z TYM KOKSEM
Memy.pl

vavr

vavr (formerly JavaSlang)

- www.vavr.io
- Aktualna wersja: **0.9.2**
- Dodatkowe moduły:
  - vavr-test
  - vavr-gwt
  - vavr-jackson
  - vavr-gson
  - **vavr-render**
- Więcej:
  github.com/vavr-io/resources
  www.vavr.io/vavr-docs/


VAVR.IO

Option

Option

```
Option<Integer> option = Option.of(someValue);
```

Optional vs Option

```java
Optional<Integer> optional = Optional.of(1);

optional.get();

optional.isPresent();
```

```java
Option<Integer> option = Option.of(1);

option.get();

option.isDefined();
```

```java
Optional<Integer> empty = Optional.empty();
```

```java
Option<Integer> none = Option.none();
Option<Void> nothing = Option.nothing();
```

Pattern Matching / Match

Pattern Matching

```
if (index == 1) {
    value = "First";
} else if (index == 2) {
    value = "Second";
} else {
    value = "(empty)";
}
```

Pattern Matching (II)

```
if (index == 1) {
    value = "First";
} else if (index == 2) {
    value = "Second";
} else {
    value = "(empty)";
}
```

```
String value = Match(index).of(
    Case($(1), "First"),
    Case($(2), "Second"),
    Case($(), "(empty)")
);
```

Pattern Matching (III)

```
Option<String> value = Match(index).option(
    Case($(1), "First"),
    Case($(2), "Second")
);
```

Pattern Matching z Predykatem

```
String value = Match(index).of(
    Case($(e -> e == 1), "First"),
    Case($(e -> e == 2), "Second"),
    Case($(), "(empty)")
);
```

Pattern Matching z Predykatem (II)

```
String value = Match(index).of(
    Case($(is(1)), "First"),
    Case($(isIn(2, 3)), "Second or Third"),
    Case($(anyOf(is(4), is(5), is(6)), "Forth"),
    Case($(isIn(Stream.from(555).take(300))), "Stream!")
    Case($(instanceOf(BigDecimal.class), "(yolo)"),
    Case($(), "(empty)")
);
```

Pattern Matching a wartość

```
String value = Match(index).of(
    Case($(1), e -> e + ""),
    Case($(2), () -> "Second"),
    Case($(), "(empty)")
);
```

Pattern Matching a void

```
Void value = Match(index).of(
    Case($(1), e -> run(() -> someWork(e))),
    Case($(2), e -> run(() -> anotherWork(e))),
    Case($(), () -> null)
);
```

Sytuacje wyjątkowe / Try

Try

```
Try.of(() -> someWork()).getOrElse(other);
```

Try (II)

```
Result result = Try.of(this::someWork)
    .recover(e -> Match(e).of(
        Case($(instanceOf(IllegalArgumentException.class)),...),
        Case($(instanceOf(FileNotFoundException.class)), ...),
        Case($(instanceOf(NullPointerException.class)), ...)
    ))
    .getOrElse(other);
```

Try (III)

```
Try<Integer> success =
    Try.of(() -> 1);

Try<Integer> fail = Try.of(() -> {
    throw new Exception(
    "Something went wrong...");
});
```

```
String value = Match(success).of(
    Case($Success($()), ":-)"),
    Case($Failure($()), ";-(")
);

String value = Match(fail).of(
    Case($Success($()), ":-)"),
    Case($Failure($()), ";-(")
);
```

Try (IV)

```
value = Match(success).of(
    Case($Success($(1)), "FTW!"),
    Case($Success($(e -> e == 2 )), "YOLO!"),
    Case($Failure($()), "Definitely not OK...")
);
value = Match(fail).of(
    Case($Success($()), "OK!"),
    Case($Failure($(instanceOf(RuntimeException.class))), "Very bad"),
    Case($Failure($()), "Bad")
);
```

Try (V)

```
String url = Try.of(parseDocument)
                .mapTry(findElementsWithPropertyTag)
                .mapTry(findElementsWithFacebookImageProperty)
                .peek(warnIfEmpty)
                .mapTry(findFirst)
                .toOption()
                .map(content)
                .getOrElse(DEFAULT_IMAGE);
```

Źródło: SoftwareMill GitHub

Walidacja / Validation

Validation

```
public class PersonValidator {

    private static final int MIN_AGE = 18;

    public Validation<List<String>, Person> validatePerson(String name, int age) {
        return Validation.combine(validateName(name), validateAge(age)).ap(Person::new);
    }

    private Validation<String, String> validateName(String name) { // some logic... }

    private Validation<String, Integer> validateAge(int age) {
        return age < MIN_AGE
                ? Validation.invalid("Age must be greater than " + MIN_AGE)
                : Validation.valid(age);
    }
}
```

Validation (II)

```
PersonValidator personValidator = new PersonValidator();

// Valid(Person(Leszke Smieszke, 42))
Validation<List<String>, Person> valid = personValidator
    .validatePerson("Leszke Smieszke", 42);

// Invalid(List(Age must be greater than 18))
Validation<List<String>, Person> invalid = personValidator
    .validatePerson("Justynian Bimber", -1);
```

Leniwe operowanie / Lazy

Lazy

```
Lazy<Double> lazy = Lazy.of(Math::random);
lazy.isEvaluated(); // = false
lazy.get();         // = 0.123 (random generated)
lazy.isEvaluated(); // = true
lazy.get();         // = 0.123 (memoized)
```

Lazy (II)

```
CharSequence value = Lazy.val(
            () -> "RTM!", CharSequence.class);
```

Rozszerzenia Collections Framework

Lists

```
java.util.Arrays
    .asList(14, 42, 99)
    .stream()
    .reduce((i, j) -> i + j);
```

```
java.util.stream.IntStream
    .of(14, 42, 99)
    .sum();
```

```
io.vavr.collection.List
    .of(14, 42, 99)
    .sum();
```

Lists (II)

```
Stream.of(14, 42, 99)
      .sorted()
      .collect(Collectors.toList());

IntStream.of(14, 42, 99)
      .sorted()
      .collect(ArrayList::new,
            List::add,
            List::addAll);

IntStream.of(14, 42, 99)
      .sorted()
      .boxed()
      .collect(Collectors.toList());
```

```
io.vavr.collection.List
      .of(14, 42, 99)
      .sort();
```

Rozszerzenia Stream API

Streams

```
io.vavr.collection.Stream
    .from(1)
    .filter(i -> i % 2 == 0);

for (double random : Stream.continually(Math::random).take(666)) {
   // some logic...
}
```

Rozszerzenia - esencja

Collections & Streams

```
interface io.vavr.collection.Traversable<T>
```

```
interface io.vavr.collection.Seq<T>
```

Kompatybilność wsteczna...

Uwaga!

```
java.util.List != io.vavr.collection.List
```

```
java.util.stream.Stream != io.vavr.collection.Stream
```

Immutability!

Konwersja do standardu

```
List.of(14, 42, 99).toJavaList();
List.of(14, 42, 99).toJavaCollection(ArrayList::new);
List.of(14, 42, 99).toJavaArray();
List.of(14, 42, 99).toJavaArray(Integer.class);

Stream.of(14, 42, 99).toJavaStream();
```

Inne możliwości

Inne możliwości

- JDK 9 - Flow API — JSR-166 / www.reactive-streams.org
- cyclops-react — github.com/aol/cyclops-react
- jOOL — github.com/jOOQ/jOOL
- streamex — github.com/amaembo/streamex
- protonpack — github.com/poetix/protonpack
- Więcej: — github.com/akullpp/awesome-java

GitHub

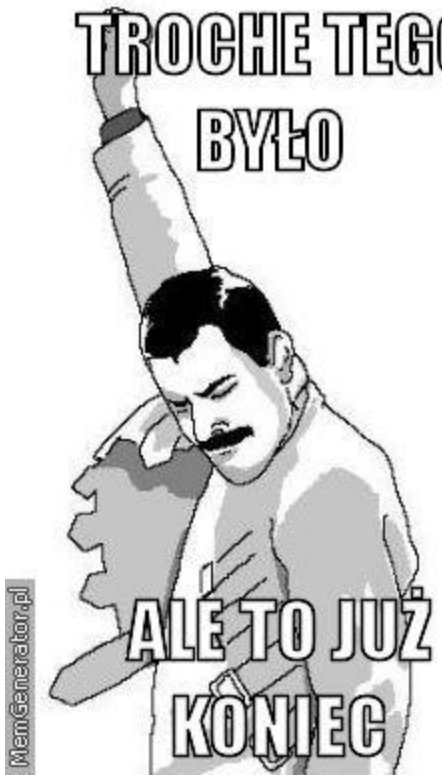
Plusy

Jakie są, każdy widzi :-)

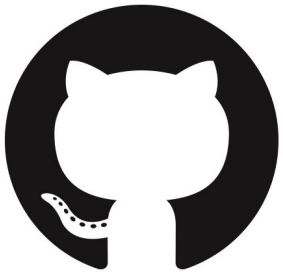
Minusy

Minusy (zawsze muszą być)

- wysoki próg wejścia
- debugging (ale czy potrzebny?)
- utrzymanie
- a jak moda przeminie?


TROCHE TEGO
BYŁO
ALE TO JUŻ
KONIEC
MemGenerator.pl

Materiały



github.com/kraluk/vavr-introduction

Dzięki za uwagę!