Programowanie funkcyjne w Java

Sławomir Ginter, SPARTEZ

O mnie...

- Inżynier, nie naukowiec-teoretyk
- Zawodowiec

  czas = pieniądz $^{-1}$
- Obecnie: core i pluginy dla Atlassian JIRA - web based Issue Tracker
- Hobby: jak przepisać klasę w Java na 1 linijkę w Scali :-)

Programowanie Obiektowe

- Obiekt

  Pola (stan)

  Metody (logika)

- Przekazywanie komunikatów - side effects
- Enkapsulacja
- GoF

  Strategy, Visitor, Command, Iterator, Template...

Programowanie Funkcyjne

- Wartości
- Funkcje
  - pure: no side effects = referential transparency
  - funkcje wyższego poziomu
  - częściowa aplikacja
- Rekurencja zamiast pętli
- Udajemy że nie ma stanu

Jaki kod chcemy?

- Kod ma być poprawny

  włącznie z wydajnością w skali makro

- Ma sie go dać czytać

  i utrzymywać!

- Ma się łatwo pisać

  reużycie

  testowanie

Co możemy poświęcić?

- Wyniki microbenchmarków
- Czas na naukę języka / paradygmatu
- Parę megabajtów RAMu i dysku

Co nas wiąże?

- Kompatybilność z istniejącym kodem

  API servera

  biblioteki - Jersey, Soy, ORM

- Koledzy :-)
- Terminy

  czas do pierwszej linijki produkcyjnego kodu

  big-bang nie przejdzie

Biblioteki funkcyjne

- Guava
- Fugue (Either, Option, fold)
- Functional Java?
- Scala!

Czemu weszliśmy w Scalę

- Guava i Fugue lokalnymi standardami
- Statyczne typowanie, compile-time checks
- Nic na siłę - dobre wsparcie dla “normalnej” Javy
- Trendy :-)

Czego się baliśmy?

- Problemów z OSGi i Springiem

  nie zarejestrowano

- Problemów z Reflection-based APIs

  Soy, ORM, Jersey - do obejścia

- Brak wsparcia narzędzi

  Maven współpracuje (lifecycle)

  Idea trochę gorzej (debugger, refactoring)

Czego się baliśmy ? c.d.

- Problemów wydajnościowych

  na razie OK

- Problemów z wdrażaniem kolegów

  spora część nowego kodu jest w Javie

- Obsunięcia terminów z powodu użycia Scali

  Nie golić jaków!

Wnioski?

- Miesiąc na wyrównanie wydajności
- Zużycie GC w normie
- Łatwiej przeoczyć złożoność (profiler!)
- Mniej Unit Testów, drastycznie mniej Mockito
- Business as Usual - większość problemów niezależna od języka i paradygmatu.
- No more Java :-)

Jak to zrobić w Javie

- Wartości

  public final String nazwa;

  konstruktor kopiujący

  builder pattern

- Może projekt Lombok?
- A może po prostu case class ;-)

Jak to zrobić w Javie

Funkcje

```
final Function<String, Integer> parseInt = new Function<String, Integer>() {
    @Override
    public Integer apply(String input) {
        return Integer.parseInt(input);
    }
};

final Function2<String, Integer, Integer> parseIntRadix =
        new Function2<String, Integer, Integer>() {
            @Override
            public Integer apply(String s, Integer radix) {
                return Integer.parseInt(s, radix);
            }
        };
```

Jak to zrobić w Javie

Funkcje - zwijamy boilerplate

```
final Function<String, Integer> parseInt =
        (input) -> { return Integer.parseInt(input); };

final Function2<String, Integer, Integer> parseIntRadix =
        (s, radix) -> { return Integer.parseInt(s, radix); };
```

Jak to zrobić w Javie

Funkcje - częściowa aplikacja (closure!)

```
public <A, B, C> Function<A, C> bind2(final Function2<A, B, C> func, final B param2) {
    return new Function<A, C>() {
        @Override
        public C apply(A input) {
            return func.apply(input, param2);
        }
    };
}
```

Jak to zrobić w Javie

Funkcje - jak użyć?

```java
final Integer v1 = parseIntRadix.apply("2a", 16);

final Function<String, Integer> parseHex = bind2(parseIntRadix, 16);
final Integer v2 = parseHex.apply("2a");

final List<String> inputs = ImmutableList.of("6", "7", "42");

final List<Integer> fromHex = Lists.transform(inputs, parseHex);
```

Jak to zrobić w Javie

A może nie w Javie?

```
val parseIntRadix = (s: String, radix: Int) => Integer.parseInt(s, radix)

val v1 = parseIntRadix("2a", 16)

val parseHex = (s: String) => parseIntRadix (s, 16)
val v2 = parseHex("2a")

val inputs = List("6", "7", "42")
val fromHex = inputs.map(parseHex)
```

Przydatne funktory

- Iterables.transform(collection, function)
- Iterables.filter(collection, predicate)
- Maps.transformValues(map, function)
- Iterables.reverse(*list*)
- fugue.Functions.fold(func, zero, collection)
- fugue.Options.flatten(collection)

Kodujemy

```
// zadanie: wiedząc że (prawdopodobnie) 3 pierwsze liczby to
// Odpowiedz na Pytanie o Sens Zycia, Swiata i Wszystkiego,
// znalezc pozostale odpowiedzi

final List<String> inputs = ImmutableList.of("6", "7", "2a",
        "A w Polsce?", "2c", "Ile wystarczy każdemu?", "280");
```

Pytania?

Sławek Ginter, SPARTEZ