AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE

Akka

Filip Malawski

fmal@agh.edu.pl

The Reactive Manifesto

- http://www.reactivemanifesto.org/
- Reactive Systems
  - Responsive – niskie czasy odpowiedzi
  - Resilient – system pozostaje responsywny po awarii
  - Elastic – system jest odporny na zmianę obciążenia
  - Message Driven - luźne powiązania, izolacja, transparentność lokalizacji

Akka

- Platforma do tworzenia reaktywnych aplikacji
- Najważniejsze cechy:
  - Aktorzy – wątki komunikujące się przez wiadomości
  - Obsługa błędów – wysokopoziomowe mechanizmy obsługi błędów
  - Klasteryzacja
  - Skalowalność

Aktorzy

- Wszystkie działania wykonywane są przez aktorów
- Aktor posiada:
  - stan oraz zachowanie (podobnie jak obiekt)
  - własny wątek w którym jest wykonywany
  - wątki aktorów są bardzo lekkie
- Komunikacja z Aktorem następuje tylko poprzez wiadomości
  - każdy aktor posiada swoją skrzynkę pocztową - kolejkę wiadomości
  - wiadomości danego aktora przetwarzane są sekwencyjnie

Aktorzy

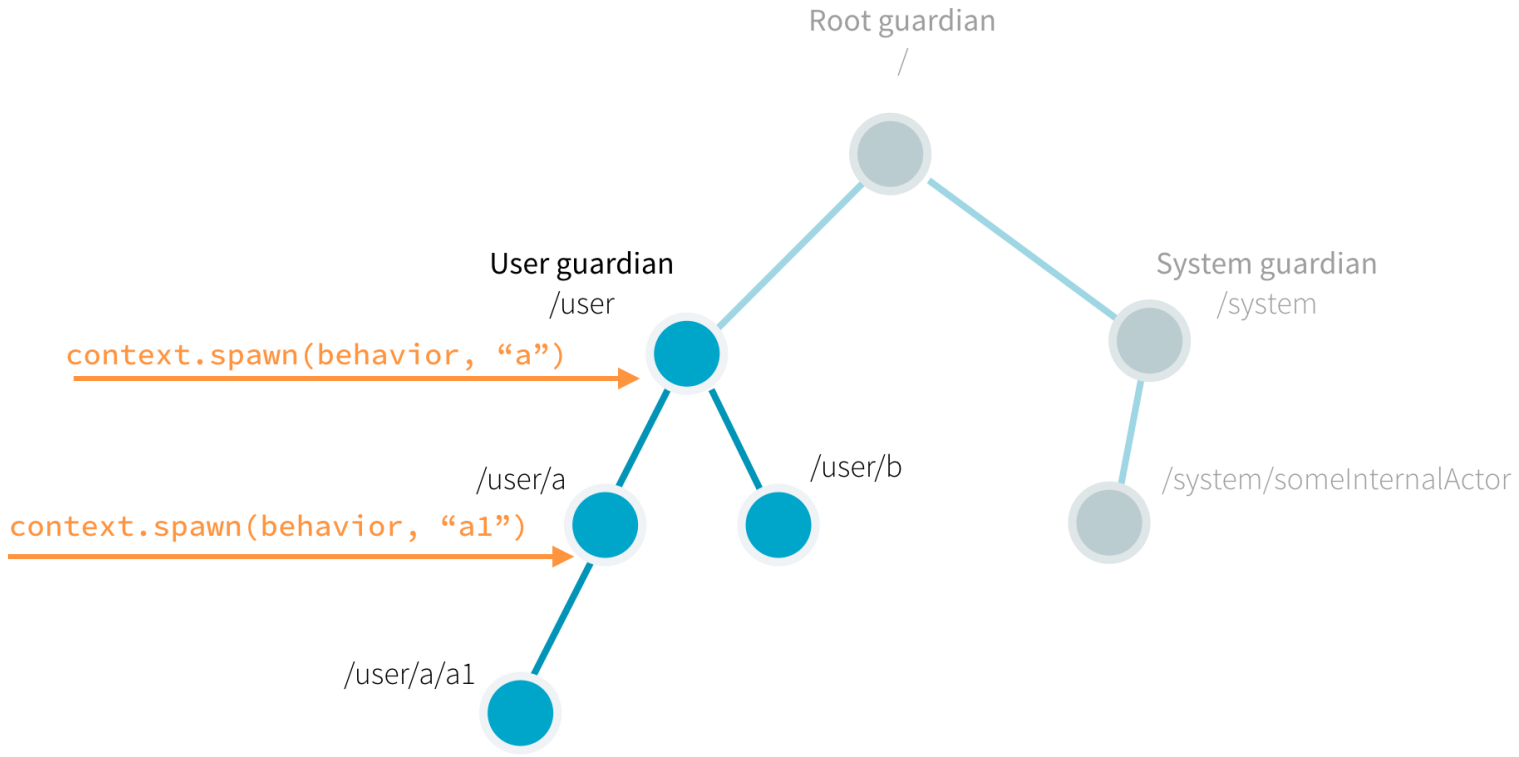
- Aktorzy są typowani przez typ wiadomości jaki obsługują
  - Implementacja przez dziedziczenie z AbstractBehavior<T>
- Dostęp do aktorów:
  - Przez referencję ActorRef<T>
- Uwaga:
  - Wcześniejsza wersja - Akka Classic – bez typowania

Aktorzy

- Aktorzy ułożeni są w hierarchię
  - Struktura drzewa
  - Aktor może tworzyć nowych Aktorów, którymi zarządza
  - Każdy Aktor ma dokładnie jednego rodzica (supervisora)
  - Supervisor decyduje o tym jak zostaną obsłużone błędy u jego podwładnych

Struktura Aktorów

- Aktorzy tworzeni przy starcie:
  - root/user/system guardians



Obsługa błędów

- Błąd aktora może być obsłużony przez:
  - Resume – wznowienie, zachowując stan
  - Restart – kasując jego stan
  - Stop – zatrzymuje działania (domyślna akcja)
- Domyślnie restart supervisora powoduje zatrzymanie podwładnych

Hello World

```
package HelloWorld;

import akka.actor.typed.ActorSystem;

public class HelloMain {

    public static void main(String[] args) throws Exception {

        // create actor system
        final ActorSystem<String> system =
                ActorSystem.create(HelloActor.create(), "helloActor");

        // send messages
        system.tell("hello world");

    }
}
```

Hello World

```
public class HelloActor extends AbstractBehavior<String> {

   // --- constructor and create
   public HelloActor(ActorContext<String> context) {
      super(context);
   }

   public static Behavior<String> create() {
      return Behaviors.setup(HelloActor::new);
   }

   // --- define message handlers
   @Override
   public Receive<String> createReceive() {
      return newReceiveBuilder()
            .onMessage(String.class, this::onMessage)
            .build();
   }

   private Behavior<String> onMessage(String msg) {
      System.out.println("received message: " + msg);
      return this;
   }
}
```

Hello World

- Zależności:
  - Dependencies.xml (Maven)

Z1

- Z1_Main
  - Tworzy system i głównego aktora MathActor
  - Wysyła do niego zadania
- MathActor
  - Obsługuje wiadomości implementujące interfejs MathCommand
  - Dodawanie obsługuje samodzielnie
  - Mnożenie deleguje do dodatkowego aktora
- MathActorMultiply
  - Wykonuje mnożenie, odsyła wynik

Zadanie 1 (2 punkty)

- Zapoznaj się z działaniem kodu Z1
- Dodaj aktora MathActorDivide obsługującego dzielenie (analogicznie jak aktor do mnożenia)
- W aktorach MathActorMultiply oraz MathActorDivide dodaj pole operationCount, które będzie zliczać ile razy dany aktor wykonał swoją operację
- Dodaj wypisywanie wartości operationCount po wykonaniu operacji przez aktora (przed odesłaniem odpowiedzi)
- Porównaj działanie strategii obsługi błędów stop/resume/restart wykorzystując przypadki testowe w Z1_Main (dzielenie przez zero)
- Obsługa strategii – zakomentowana w MathActor
- Uwaga – dzielenie przez zero powoduje logowanie błędu, ale zatrzymuje działania systemu aktorowego

Zadanie 1 (2 punkty)

- W raporcie należy umieścić:
  - Output wypisany przy 3 różnych strategiach (stop/resume/restart)
  - Uwaga – należy wypisywać TYLKO informacje o liczbie wykonanych operacji oraz automatycznie logowane informacje o błędach

Klasteryzacja

- Możliwość uruchomienia wielu węzłów działających w jednym klastrze
- Konfiguracja przez plik konfiguracyjny
- Odnajdywanie aktorów przez recepcjonistę

Z2

- Z2_Main tworzy:
  - Usługę przetwarzania tekstu w postaci aktora ActorTextService
  - Usługi ActorUpperCase, z których ActorTextService będzie korzystał
  - Wysyła jedno zlecenie
- ActorUpperCase:
  - Rejestruje się u recepcjonisty
  - Obsługuje wiadomości typu String
  - Wypisuje UpperCase wiadomości

Z2

- ActorServiceText
  - Przyjmuje zlecenia zawierające tekst do przetworzenia
  - Zlecenia przesyła do workerów ActorUpperCase

Zadanie 2 (2 punkty)

- Zmodyfikuj ActorTextService:
  - Dodaj obsługę subskrypcji na zdarzenia od recepcjonisty
  - Gdy pojawia się nowa wiadomość od recepcjonisty należy zaktualizować listę workerów

Zadanie 2 (2 punkty)

- Wymagane modyfikacje:
  - Zapisanie się do Receptionist.subscribe (w konstruktorze)
  - Stworzenie adaptera wiadomości do obsługi wiadomości recepcjonisty (w konstruktorze)
  - Dodanie nowej klasy z wiadomością, która implementuje interfejs Command oraz ma pole typu Receptionist.Listing
  - Dodanie obsługi powyższego typu wiadomości (w tym miejscu aktualizowana jest lista workerów)
  - https://doc.akka.io/docs/akka/current/typed/actor-discovery.html

Zadanie 2 (2 punkty)

- Zmodyfikuj pliki Z2_NodeA, Z2_NodeB tak, aby tworzyły odpowiednio:
  - Workerów (A)
  - Text service (B)
- Konfiguracja do uruchomienia klastra jest gotowa
- Uruchom Z2_NodeA, potem Z2_NodeB
  - Uwaga może być potrzebny dłuższy sleep lub wpisywanie tekstu z konsoli

Zadanie 2 (2 punkty)

- W raporcie należy umieścić:
  a) Output wypisany przez Z2_Main
  b) Output wypisany przez Z2_NodeA oraz Z2_NodeB

Raport

- Wynikiem ćwiczenia ma być raport w formie **dokumentu PDF** zawierający: **imię, nazwisko, numer indeksu** oraz odpowiedzi do pytań (w opisie zadań podano co ma się znaleźć w raporcie).
- Raport należy przesłać przez UPEL (Akka - zadanie z ćwiczeń). **Nazwa dokumentu ma być zgodna ze schematem**: *Nazwisko_Imie_akka_lab.pdf*, np. *Mickiewicz_Adam_akka_lab.pdf.*

AGH

Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Dziękuję