Akka

A Brief Introduction

Martin Anlauf

Focus

- Classification
- Programming model
- Toy Code Inspection
- See it running

What's Akka?

- Actor model on the JVM
  - 1973 Hewitt, Bishop & Steiger
- Precursor: Erlang on the OTP
  - 1986, Reputation for robustness, availability & scalability
- Developed by Typesafe / Lightbend
  - Part of their reactive stack

Claimed Properties

- Responsiveness
- Resilience
- Elasticity (a.k.a. Scalability)
- Message driven
- Check out the Reactive Manifesto

What are Actors?

- Various viewpoints
  - Independent units of behavior, communicating with messages
  - Communicating state machines / sequential processes (Hoare)
  - Mail boxes with attached behavior
  - Technique for concurrency and distribution
- Distinct programing model, idioms & patterns

What are Actors not?

- Functional
  - Behavior may depend on state
  - Focus on sending messages, not on providing values
- Type safe
  - Message processing does not provide type guarantees
  - There is experimental work to provide type safety

Technical Properties, I

- Each Actor is processed sequentially (*)
  - Processing of different Actors may overlap
  - „Unit of sequential behavior in a sea of concurrency“
- Asynchronous communication
  - Send messages without any guarantees („Fire and forget“)
  - No mutable state shared (*)

(*) Not enforced in Akka. Respect conventions

Technical Properties, II

- Scalability
  - Many actors can be dispatched to a smaller number of threads
  - Claimed difference of about 3 magnitudes
- Distribution by location transparency
  - Different actors may run on the same JVM, on different JVM's on the same machine, or on different machines
  - The infrastructure will handle it

Programming Model

- Define a System of collaborating actors
  - Divide system into small entities
  - Give each entity a dedicated role (a task)
  - Define communication between entities
    - Define messages to exchange (Case classes)
    - Define for each role how to processes messages
  - Behavior is defined by several protocols

Operations

- create, send/receive, become & supervise
- create
  - Hierarchical organisation
  - Creation hierarchy = supervision hierarchy
- send
  - asynchronous, no blocking
  - no guarantees (fire & forget)

Operations II

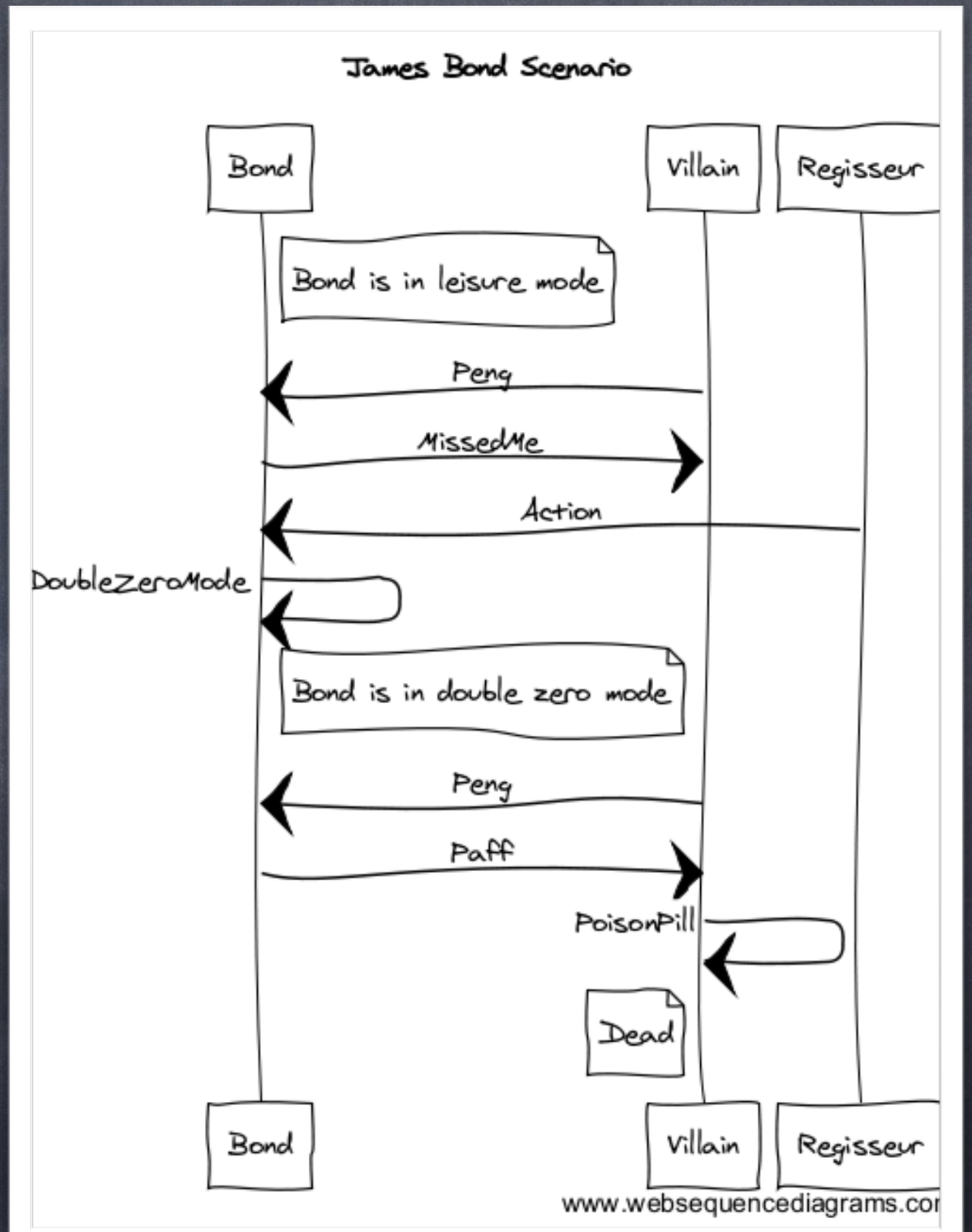
- become
  - Different modes for different behavior
  - State machine flavor
- supervision
  - Watch child's lifecycle
  - Stop or restart child
  - Allows error recovery, fault tolerance, resilience

Example

- Entertainment instead of real life
- Movie script instead of requirements
- Illustrate collaborating actors with simple James Bond scenario

Scenario

- A villain tries to kill Bond
- In leisure mode, Bond ignores it
- In double zero mode, he shoots back
- Thanks to www.websequencediagrams.com



Code

- > activator new, minimal-akka-scala-seed
- First the references
- Then to the IDE...

References

- http://akka.io
- Akka in Action, Roestenburg, Bakker & Williams, Manning Early Access
- Reactive Design Patterns, Kuhn & Allen, Manning Early Access