Akka Extensions

Pablo Díaz
@pablo_dilo

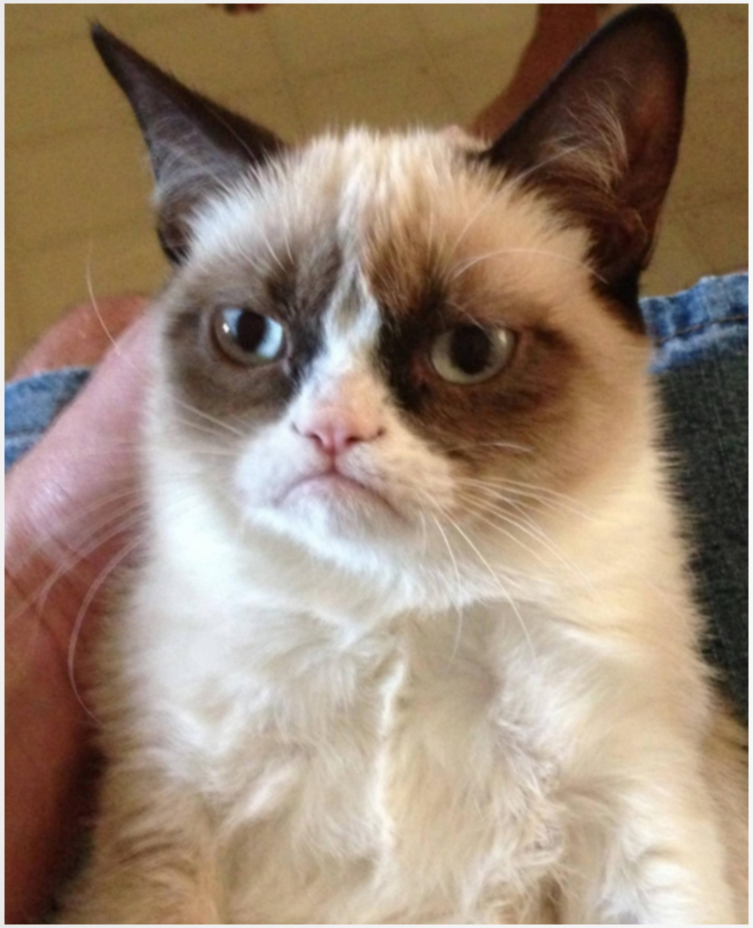
Who am I?

- hAkker, crAkker and whateverAkker
- I collaborate with ESA (European Space Agency) in my spare time

Who I am... Seriously

- Software Engineer
- Currently using Java 8
- Steam game collector
- Escapist (only two games left in BCN)

Are we going to learn anything?



Hope so...

Akka

“Akka is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM”

http://akka.io/

Actor Model

- Unit of work
- Send/receive messages
- Create other actors
- Supervise it
- Change state

Mars Rover

Previously on ScalaBCN...

Mars Rover

Mars Rover

- ESA (European Space Agency)
- Satellite
- Mars Rover

Mars Rover

- Objective: Move the Mars Rover to a concrete position

- Commands accepted by MarsRover:
  - Start engine
  - Stop engine
  - Turn Left/Right
  - Get position

Mars Rover

Lets see some code

Akka extension

- Is the way to add features to akka, in a very simple way:
  - Extension: Implements the functionality (be threadsafe)
  - ExtensionId: Lookups and initialize the Extension

The sky is the limit

Example: Mars Rover

- How many different commands do we send to the rover?

Let's code

Example: LatencyExtension

- Provides a delay to simulate latencies when sending a message to another Actor
- Reads configuration from settings
- Let's see it!

Akka persistence

- Akka persistence enables stateful actors to persist their internal state.
- It can be recovered when an actor is started, restarted after a JVM crash or by a supervisor.
- The persistence storage can be configured/implemented

Akka persistence: Journal

- Akka provides by default LevelDB implementation

- Can be overrided via:
  - akka.persistence.journal.plugin

- Alternatives: https://github.com/dnvriend/akka-persistence-jdbc

How do I persist my Actor?

- Inherit from PersistentActor
- Set a persistentId
- Implement receiveRecover and receiveCommand

Persist operations

- persist[A](message: A) (handler:A=>Unit)
- saveSnapshot(snapshot: Any)

Overriding things

- By default It recovers on Actor initialization

```
def preStart(): Unit = {
    self ! Recover(<SeqNr>)
}
```

Snapshots

- saveSnapshot(snapshot: Any)
  - If fails:

    SaveSnapshotSuccess(metadata)
  - If success:

    SaveSnapshotFailure(metadata, reason)

  - on recovery sends SnapshotOffer(snapshot: Any)

Recovery end

- RecoveryCompleted
- RecoveryFailure(cause: Throwable)

Recovery status

- recoveryRunning
- recoveryFinished

Mars Rover persisted

Let's persist the Rover!

Async things

- persistAsync[A](message: A) (handler:A=>Unit)

Async things to deal with

```
def receiveRecover =
    ...
    case MyMessage(msg) =>
        cmd1()
        persistAsync(Event(msg)) (e => {
            cmd2()
        })
        cmd3()
    ...
```

```
1
[2]
3
[2]
```

More persist things

```
def receiveRecover =
    ...
    case MyMessage(msg) =>
        cmd1()
        persist(Event(msg)) (e => {
            cmd2()
        })
        cmd3()
    ...
```

```
1
[2]
3
[2]
```

Async things to deal with II

```
def receiveRecover =
    ...
    case MyMessage(msg) =>
        cmd1()
        persistAsync(Event(msg)) (e => {
            cmd2()
        })
        persistAsync(Event(msg)) (e => {
            cmd3()
        })
    ...
```

```
1
2
3
```

More persist things

```
def receiveRecover =
    ...
    case MyMessage(msg) =>
        log("1")
        persistAsync(Event(msg)) (e => {
            log("2")
        })
        defer(msg) (_ => {
            log("3")
        })
    ...
```

```
1
2
3
```

PersistentView

- It's a trait
- Set a persistenceId
- Set a viewId

Persistent View

```
def receive = {
  case c:Cmd if isPersistent =>
    …
  case c:Cmd =>
    …
```

Persistent View updates

- Update interval settings:
  - akka.persistence.view.auto-update-interval
  - akka.persistence.view.auto-update
- We can force an update sending a Update message
- Update has a parameter await to force receive first persistent messages.

AtLeastOnceDelivery

- It is a trait too
- After crash-restart message are still delivered

AtLeastOnceDelivery

```
delivery(
  destination: ActorPath,
  deliveryIdToMessage: (Long=>Any)
)

confirmDelivery(deliveryId: Long)
```

AtLeastOnceDelivery

- Has snapshot support for the delivery persistence

  getDeliverySnaphost(): AtLeastOnceDelivery

  setDeliverySnapshot(snapshot:AtLeastOnceDelivery)

And this is Akka Persistence

Akka Streams

- Implementation of Reactive Streams
- Asynchronous stream processing
- Very experimental, currently 0.11, there is no official documentation
- Huge change between releases.

Today we learn

- A bit of Akka
- Some examples using Akka and our friend Mars Rover
- Some of Akka Extensions
- Sky is not the limit, at least Mars

Questions