

# Akka

## Reactive Applications made easy

Michael Pisula, München, 2014-04-30



# Akka



[AkkamountainbyArvelius](#)

# What are reactive applications?

---

*New requirements demand new technologies -  
The Reactive Manifesto*

---

interactive

scalable

resilient

event-driven

# What is Akka?

actor programming model  
+ fault tolerance  
+ location transparency



# Actors

# Creating an Actor is easy

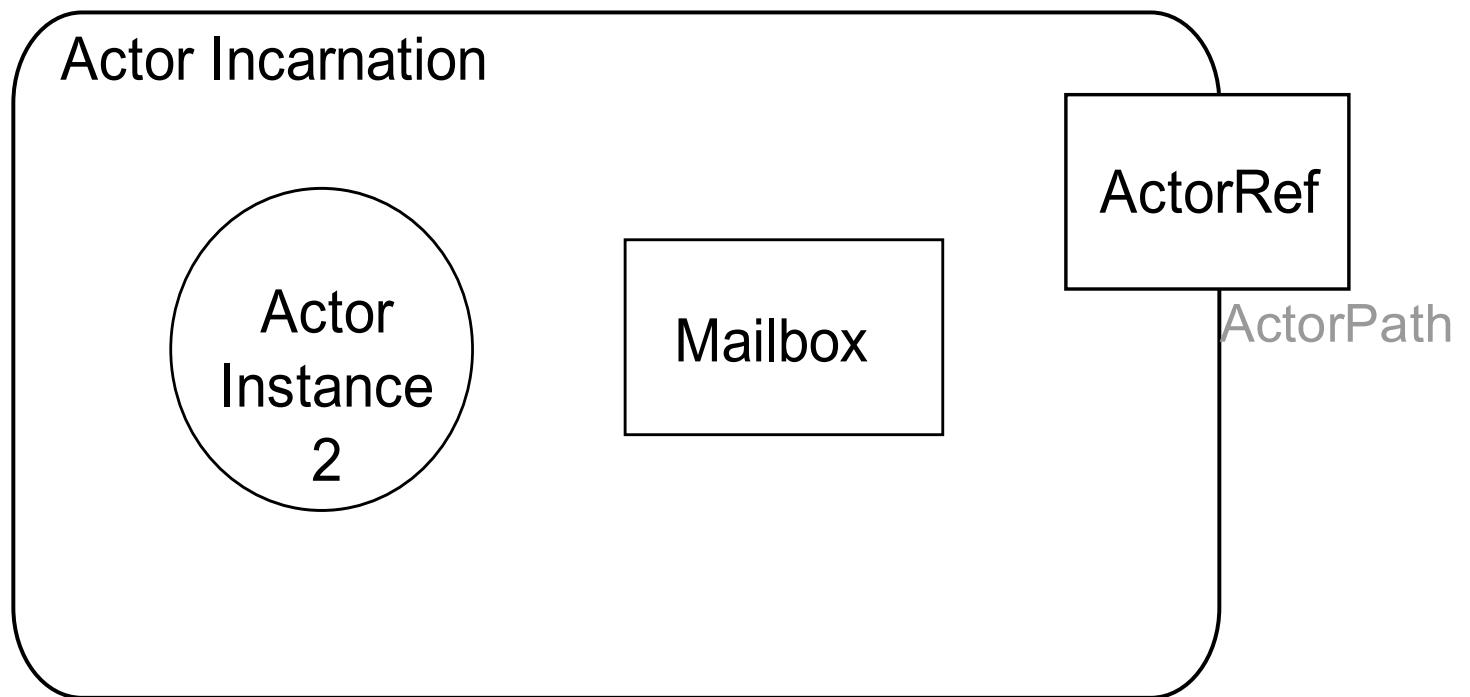
```
class Adder extends UntypedActor {  
  
    private int sum = 0;  
  
    @Override  
    public void onReceive(Object message) throws Exception {  
        if (message instanceof Add) {  
            sum += ((Add)message).value;  
        } else if (message instanceof GetSum) {  
            getSender().tell(new Sum(sum), getSelf());  
        }  
    }  
}
```

# Working with Actors

```
ActorSystem system = ActorSystem.create("AdderSystem");
ActorRef add = system.actorOf(Props.create(Adder.class));
add.tell(new Add(1), null);
```

Code

# Inside an Actor



# Resilience

---



[Eltern haften für ihre Kinder by Katharina Hamacher](#)

# Creating a hierarchy is easy

```
// top-level actor
system.actorOf(Props.create(Adder.class));
// child actor
getContext().actorOf(Props.create(Adder.class));
```

# The Akka Hierarchy

# Supervision strategies

- One For One
- All For One

# Supervision options

- Resume
- Restart
- Stop
- Escalate

# Defining supervision is easy

```
private SupervisorStrategy strategy =
    new OneForOneStrategy(10, Duration.create(5, TimeUnit.SECONDS),
        new Function() {
            @Override
            public Directive apply(Throwable t) throws Exception {
                return SupervisorStrategy.resume();
            }
        });
@Override
public SupervisorStrategy supervisorStrategy() {
    return strategy;
}
```

Code

# Remoting & Clustering

---



[Server room by Torkild Retvedt](#)

# Configure the node first

```
akka {  
    actor {  
        provider = "akka.remote.RemoteActorRefProvider"  
    }  
    remote {  
        enabled-transports = ["akka.remote.netty.tcp"]  
        netty.tcp {  
            hostname = "127.0.0.1"  
            port = 2552  
        }  
    }  
}
```

# Remoting actors now is easy

```
akka {  
    actor {  
        deployment {  
            /echoActor {  
                remote = "akka.tcp://remoteSystem@127.0.0.1:2553"  
            }  
        }  
    }  
}
```

**Code**

# Clustering

```
akka {  
    actor {  
        provider = "akka.cluster.ClusterActorRefProvider"  
    }  
    remote {  
        ...  
    }  
    cluster {  
        seed-nodes = [  
            "akka.tcp://ClusterSystem@127.0.0.1:2551",  
            "akka.tcp://ClusterSystem@127.0.0.1:2552"]  
        auto-down-unreachable-after = 10s  
    }  
}
```

Code

# Camel Integration

# Greeting Web Service

```
public class GreetingService extends UntypedConsumerActor {

    @Override
    public void onReceive(Object message) throws Exception {
        if (message instanceof CamelMessage) {
            CamelMessage camelMsg = (CamelMessage) message;
            Map headers = camelMsg.getHeaders();
            getSender().tell("Hello " + headers.get("greetee"), getSelf())
        }
    }

    @Override
    public String getEndpointUri() {
        return "jetty:http://localhost:4242/greet";
    }
}
```

## Example Code



# Testing



[almeraeuroncap](#)

# Testing with scalatest

```
class AdderSpec extends TestKit with ImplicitSender with WordSpec ... {  
    ...  
    "An Adder actor" must {  
        "return 3 as sum of 1 and 2" in {  
            val greeter = system.actorOf(Props[Adder])  
            greeter ! new Add(1)  
            greeter ! new Add(2)  
            greeter ! new GetSum()  
            expectMsg(new Sum(3))  
        }  
    }  
}
```

# Testing with JUnit

```
@Test
public void add2to1() {
    ActorRef adder = system.actorOf(Props.create(Adder.class));
    JavaTestKit probe = new JavaTestKit(system);
    adder.tell(new Adder.Add(1), probe.getRef());
    adder.tell(new Adder.Add(2), probe.getRef());
    adder.tell(new Adder.GetSum(), probe.getRef());

    probe.expectMsgEquals(new Adder.Sum(3));
}
```

# TestActorRef

```
"An Adder actor" must {
    "know the Answer to the Question of Life, the Universe, and Everything" in {
        val greeter = TestActorRef[Adder]
        greeter.underlyingActor.sum = 42
        greeter ! new GetSum()
        expectMsg(new Sum(42))
    }
}
```

## Code

# Persistence



Fossil by Yaffa Phillips

# Persisting state

```
public class PersistentAdder extends UntypedProcessor {  
    private int sum = 0;  
  
    @Override  
    public void onReceive(Object message) throws Exception {  
        if (message instanceof Persistent) {  
            Object payload = ((Persistent) message).payload();  
            if (payload instanceof Add) {  
                sum += Integer.parseInt(((Add) payload).value);  
            }  
        }  
    }  
}
```

Code

# Guaranteed delivery

```
public void onReceive(Object message){  
    if (message instanceof ConfirmablePersistent) {  
        ((ConfirmablePersistent) message).confirm();  
        getSender().tell(((ConfirmablePersistent) message).payload(), getSelf());  
    }  
}  
...  
ActorRef channel = system.actorOf(Channel.props());  
channel.tell(Deliver.create(  
    Persistent.create("Hello Echo!"), echoActor.path()), inbox.getRef());
```

Code

# Learning Akka

# Typesafe Activator

- Great tool to learn about Akka
- Many templates exist, from Hello World to complex examples
- 81 templates are currently available, >30 for Akka

[Link](#)

# Books

- Akka Concurrency, Derek Wyatt
- Akka in Action, Raymond Roestenburg

# Thank you!

Questions?

# Happy hAkking!

[michael.pisula@tngtech.com](mailto:michael.pisula@tngtech.com)

[https://www.xing.com/profile/Michael\\_Pisula](https://www.xing.com/profile/Michael_Pisula)

[@MichaelPisula](https://twitter.com/MichaelPisula)

