

Child Actors



Child Actors

Actors can create other actors

```
val childRef: ActorRef[String] = context.spawn(Child(), name)
```

Actor organization

- tree-like hierarchy
- actor paths
- root guardian, /system guardian, /user guardian

ActorRef: a "pointer" to the actor instance we can use to send messages to

```
childRef ! message
```

New best practices

- keep the ActorSystem's behavior empty
- set up all your important actors and interactions at the setup of the ActorSystem

Stopping Actors

Stopping this actor: return `Behaviors.stopped`

Optionally handle a `PostStop` signal

```
.receiveSignal {  
  case (context, PostStop) =>  
    // clean up resources that this actor might use  
    context.log.info("I'm stopping now.")  
    Behaviors.same // not used anymore in case of stopping  
}
```

Stopping a child: use the actor context

```
context.stop(childRef) // only works with CHILD actors
```

Watching Actors

Get a notification (signal) when the actor dies

```
context.watch(ref) // works with any actor
```

```
.receiveSignal {  
  case (context, Terminated(ref)) =>  
    context.log.info(s"[parent] Actor ${ref.path} was killed.")  
    // your business logic here  
}
```

Deregistering

```
context.unwatch(ref) // works with any actor
```

Properties

- the Terminated signal gets sent even if the watched actor is already dead at registration time
- registering multiple times may/may not generate multiple Terminated signals
- unwatching will not process Terminated signals even if they have already been enqueued

Akka rocks

