Defining and Using Actors and Messages

Jason Roberts

@robertsjason | dontcodetired.com

Overview

Defining actors and good practices

Actor references

Defining messages and good practices

Creating actor instances using Props

Sending simple and custom messages

Defining Actors

Create class

Inherit from Akka.NET base class

Define message handling

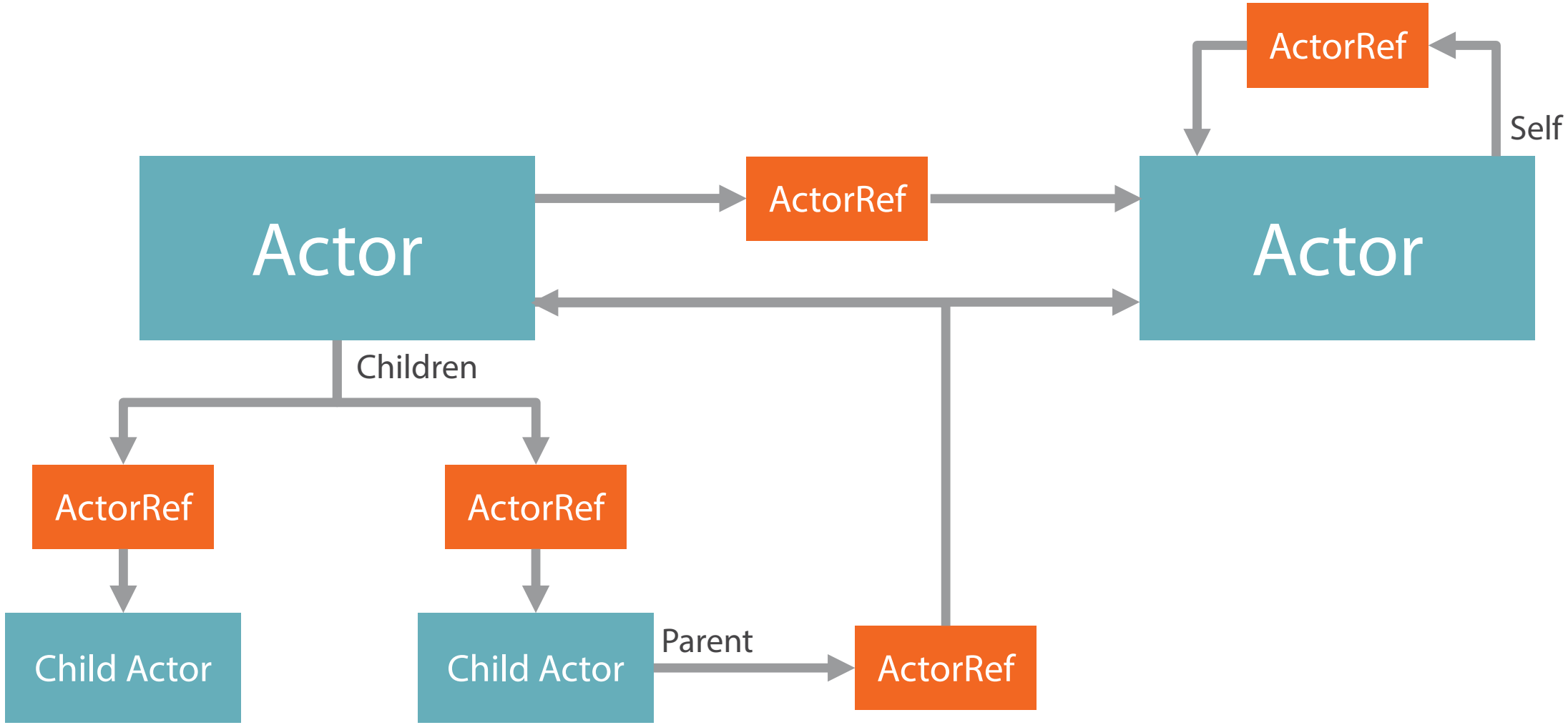
Considerate and efficient

Functionally cohesive

Don’t expose mutable state

Arrange in hierarchies for error handling

Actor References



Obtaining Actor References

Create new actor

ActorOf()

Lookup existing actor

ActorSelection()

Defining Messages

Create POCO class | Add properties | Make setters private

Don’t pass mutable state between actors

Create immutable messages (thread-safe)

Consider remote use (serializable & size)

Types of Message Sending

Tell | Ask | Forward

Types of Message Sending

Tell

- Tell target actor something
- Don't expect a response
- "Fire and forget"
- Doesn't block waiting for a reply
- Better scaling and concurrency
- Use most of the time

Ask

- Ask target actor for something
- Expect a response
- Target actor has to reply to sender
- Timeout (TaskCancelledException)
- Worse scaling and concurrency
- Only use when absolutely necessary

Actor Instantiation

Define “Props”

Think of an instance name

Create using Props and name

Props

The recipe for how to make an instance of a particular type of actor. A Prop contains the information that the actor system needs to create the instance, such as actor constructor argument values and deployment related configuration information.

DO Use factory methods to create props and actors

DO NOT simply use the new keyword

Creating and Instantiating an Actor

Create PlaybackActor class

Inherit from UntypedActor

Create Props

Create actor instance and get its reference

Defining Which Messages an Actor Will Handle

Modify PlaybackActor to respond to string and int messages

Use actor reference and Tell() to send a message to it

Sending a Custom Message

Create PlayMovieMessage class

Modify OnReceive method

Send new message to the actor

ReceiveActors

More elegant way of specifying what message types can be handled

Additionally specify predicates to further filter if a message will be handled

Automatically call the Unhandled method if a message is not matched

Refactoring to a ReceiveActor

Summary

Defining actors and messages

Actor references (IActorRef)

Sending messages: Tell, Ask, Forward

Props.Create<PlaybackActor>()

```
MovieStreamingActorSystem.ActorOf(
                playbackActorProps, "PlaybackActor")
```

UntypedActor & ReceiveActor

Using predicates in ReceiveActors

Next:

Understanding Actor Lifecycles and States