

# akka-examples (0.0.1) [\[Build Status\]](#)

Maksim Kostromin

Version 0.0.1, 2018-07-15 02:43:46 UTC

# Table of Contents

1. Implementation .....	2
1.1. scala akka sbt IDEA .....	2
1.2. Java Gradle Maven Docker Starter .....	5
1.3. Kotlin Gradle Maven Docker Starter .....	5
1.4. akka .....	6
1.5. another-akka-try .....	6
2. Links .....	8
3. Enjoy! :) .....	9

# Introduction

This documentation contains some help to [examples from akka-examples repository](#). Akka playground projects

This repository contains Akka playground projects examples.

## Documentation

- [Learning Akka projects](#)
- [ping-pong: yet another java gradle / maven docker akka jar...](#)
- [java \(gradle / maven\) starter in docker](#)
- [kotlin \(gradle / maven\) starter in docker](#)
- [scala \(gradle / maven\) starter in docker](#)

TODO: - spring-boot + akka [1](#) and [2](#)

# Chapter 1. Implementation

## 1.1. scala akka sbt IDEA

*build, test and run*

```
./sbtw clean compile test assembly # run  
ava -jar target/scala-2.12/*-assembly-*.jar
```

*Tell/Ask to actors*

```
package daggerok.tellandaskactors  
  
import akka.actor.{Actor, ActorRef, ActorSystem, Props}  
import akka.util.Timeout  
import daggerok.tellandaskactors.CheckerAPI.{BlacklistUserResponse, CheckUserRequest, WhitelistUserResponse}  
import daggerok.tellandaskactors.RecorderAPI.CreateUser  
import daggerok.tellandaskactors.StorageAPI.AddUser  
  
case class User(name: String)  
  
object RecorderAPI {  
  sealed trait RecorderMessage  
  case class CreateUser(user: User) extends RecorderMessage  
}  
  
class Recorder(checker: ActorRef, storage: ActorRef) extends Actor {  
  import scala.concurrent.duration._ // seconds  
  implicit val timeout = Timeout(5 seconds)  
  
  import akka.pattern.ask // actor ask map  
  import scala.concurrent.ExecutionContext.Implicits.global  
  override def receive: Receive = {  
    case CreateUser(user) =>  
      checker ? CheckUserRequest(user) map {  
        case WhitelistUserResponse(user) =>  
          storage ! AddUser(user)  
        case BlacklistUserResponse(user) =>  
          println(s"Recorder black user: $user")  
      }  
    case _ => println("Checker: unknown message received")  
  }  
}  
  
object CheckerAPI {  
  sealed trait CheckerRequest  
  case class CheckUserRequest(user: User) extends CheckerRequest
```

```

sealed trait CheckerResponse
case class BlacklistUserResponse(user: User) extends CheckerResponse
case class WhitelistUserResponse(user: User) extends CheckerResponse
}

class Checker extends Actor {
  val blacklist = List("bad", "black", "evil")
  override def receive: Receive = {
    case CheckUserRequest(user) if (blacklist.exists(name =>
user.name.contains(name))) =>
      sender() ! BlacklistUserResponse(user)
    case CheckUserRequest(user) =>
      sender() ! WhitelistUserResponse(user)
    case _ => println("Checker: unknown message received")
  }
}

object StorageAPI {
  sealed trait StorageMessage
  case class AddUser(user: User) extends StorageMessage
}

class Storage extends Actor {
  private var users = List.empty[User]
  override def receive: Receive = {
    case AddUser(user) =>
      println(s"user $user added")
      users = user :: users
    case _ => println("Storage: unknown message received")
  }
}

object TellAndAskApp {
  def main(args: Array[String]): Unit = {
    val system = ActorSystem("tell-ask-system")
    val checker = system.actorOf(Props[Checker], "checker")
    val storage = system.actorOf(Props[Storage], "storage")
    val recorder = system.actorOf(Props(new Recorder(checker, storage)), "recorder")

    recorder ! CreateUser(User("white"))
    recorder ! CreateUser(User("black"))
    recorder ! CreateUser(User("one more white"))
    Thread.sleep(1000)
    system.terminate()
  }
}

```

```
package daggerok.musicplayer

import akka.actor.{Actor, ActorSystem, Props}
import daggerok.musicplayer.MusicController.{PlayMsg, StopMsg}
import daggerok.musicplayer.MusicPlayer.{StartMusicMsg, StopMusicMsg}

object MusicController {
  sealed trait MusicControllerMessage
  case object PlayMsg extends MusicControllerMessage
  case object StopMsg extends MusicControllerMessage

  def props = Props[MusicControllerActor]
}

class MusicControllerActor extends Actor {
  override def receive: Receive = {
    case PlayMsg => println("playing music...")
    case StopMsg => println("music is stopped")
  }
}

object MusicPlayer {
  sealed trait MusicPlayerMessage
  case object StopMusicMsg extends MusicPlayerMessage
  case object StartMusicMsg extends MusicPlayerMessage
}

class MusicPlayerActor extends Actor {
  override def receive: Receive = {
    case StopMusicMsg => println("I don't wanna stop!")
    case StartMusicMsg =>
      val ctrlActor = context.actorOf(MusicController.props, "ctrl-actor")
      ctrlActor ! PlayMsg
    case _ => println("received unknown message.")
  }
}

object MusicPlayerApp {
  def main(args: Array[String]): Unit = {
    val system = ActorSystem("music-system")
    val actor = system.actorOf(Props[MusicPlayerActor], "mp-actor")

    actor ! StartMusicMsg
    actor ! StopMusicMsg
    Thread.sleep(1000)
    system.terminate()
  }
}
```

```
package daggerok.helloworld

import akka.actor.{Actor, ActorSystem, Props}

case class HelloMessage(name: String)

class HelloActor extends Actor {
  override def receive: Receive = {
    case HelloMessage(name) => println(s"Hello, $name!")
  }
}

object HelloWorldApp {
  def main(args: Array[String]): Unit = {
    val helloSystem = ActorSystem("hello-actor-system")
    val helloActor = helloSystem.actorOf(Props[HelloActor], "hello-actor")

    helloActor ! HelloMessage("Максимко")
    Thread.sleep(1000)
    helloSystem.terminate()
  }
}
```

## 1.2. Java Gradle Maven Docker Starter

This is just an Akka java gradle / maven / docker / starter project...

*build and test*

```
docker-compose down -v; ./mvnw clean package; ./gradlew clean build; docker-compose up
--build --force-recreate --remove-orphans

# or
docker-compose down -v
./mvnw clean package
./gradlew clean build
docker-compose up --build --force-recreate --remove-orphans
```

links:

- [Akka docs](#)

## 1.3. Kotlin Gradle Maven Docker Starter

This is just an Akka kotlin gradle / maven / docker / starter project...

### *build and test*

```
docker-compose down -v; ./mvnw clean package; ./gradlew clean build; docker-compose up --build --force-recreate --remove-orphans
```

```
# or
docker-compose down -v
./mvnw clean package
./gradlew clean build
docker-compose up --build --force-recreate --remove-orphans
```

links:

- [kotlin with maven](#)
- [Akka docs](#)

## 1.4. akka

generated using [jvm](#) yeoman generator

### *build*

```
./mvnw clean package com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:up
./mvnw com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:down

./gradlew clean build composeUp
./gradlew composeDown
```

## 1.5. another-akka-try

### *test*

```
./gradlew clean installDist
bash build/install/another-akka-try/bin/another-akka-try

./gradlew clean distZip
unzip -o build/distributions/*.zip -d /tmp
bash /tmp/another-akka-try-0.0.1/bin/another-akka-try

./mvnw
java -jar target/*-all.jar
```



## *build*

```
./mvnw clean package com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:up
./mvnw com.dkanejs.maven.plugins:docker-compose-maven-plugin:1.0.1:down

./gradlew clean build composeUp
./gradlew composeDown
```

generated using [jvm](#) yeoman generator

# Chapter 2. Links

- [Learning Akka](#)
- [Videos: Intro to Akka](#)
- [Reactive Programming with Akka](#)
- [Akka and the Zen of Reactive System Design](#)

## Chapter 3. Enjoy! :)