akka-examples (0.0.1) [Build Status]

Maksim Kostromin

Version 0.0.1, 2018-07-08 22:24:42 UTC

Table of Contents

Introduction

This documentation contains some help to examples from akka-examples repository. Akka playground projects

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
```

Music Player

```
package daggerok.musicplayer

import akka.actor.{Actor, ActorSystem, Props}
import daggerok.MusicController.{PlayMsg, StopMsg}
import daggerok.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()
  }
}
```

Hello World

```
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! :)