

# Introducción a Akka con Java

Cristian Rodríguez Bernal

# ¿Quién soy?

- ▶ Doble ingeniero en Informática y Software por esta casa, perteneciente a la primera promoción.
- ▶ Actualmente estudiante del Máster en Gráficos, Juegos y Realidad Virtual.
- ▶ Mi TFG de Ing. del Software está basado en un estudio de distintos modelos y arquitecturas para el desarrollo de servidores concurrentes.



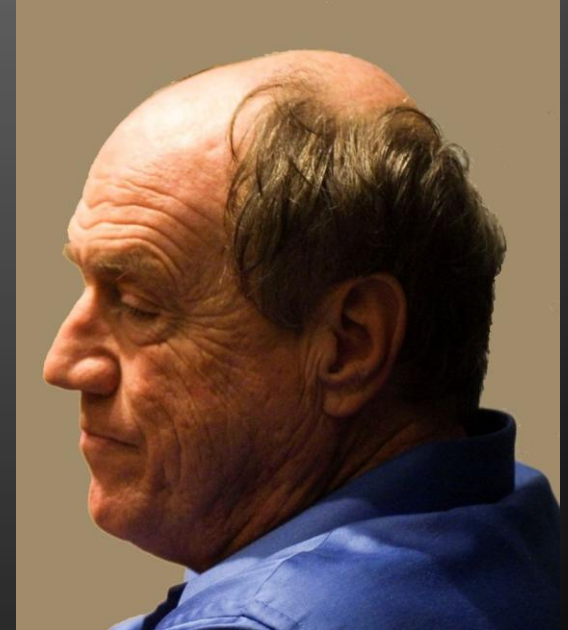
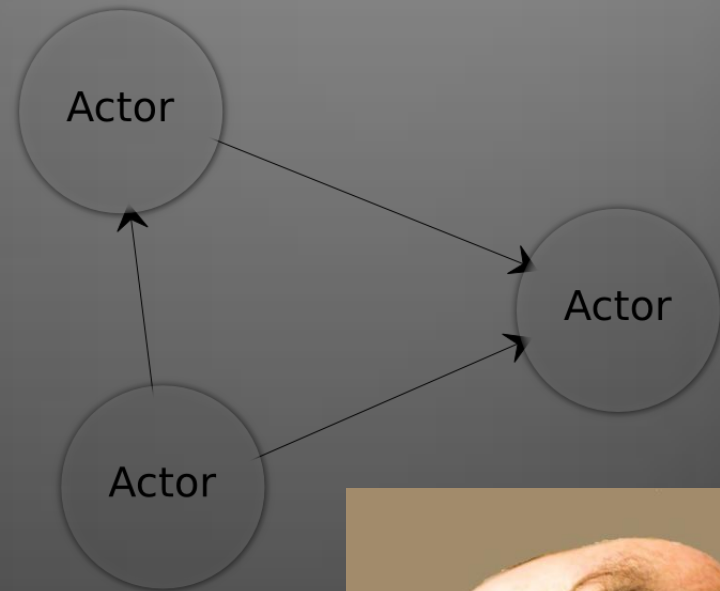
# ¿Qué es Akka?

- ▶ *Akka* es un conjunto de herramientas open source para la construcción de sistemas concurrentes y distribuidos en la JVM. Aunque implementa múltiples modelos de programación concurrente, su principal fuerza se encuentra en el modelo de concurrencia con actores, inspirado en *Erlang*.
- ▶ Se puede desarrollar código con librerías *Akka* tanto en *Java* como *Scala* (su lenguaje natural).



# ¿Modelo de concurrencia con actores?

- ▶ Creado por Carl Hewitt en 1973.
- ▶ Este modelo separa cada “hilo” como una entidad, que se comunica con otras a través de un paso de mensajes o un buzón.
- ▶ Las tareas principales de un actor son:
  - Enviar mensajes a otros actores.
  - Crear otros actores.
  - Cambiar de comportamiento según los mensajes recibidos.
- ▶ Comunicación de forma asíncrona.
- ▶ No hay estado compartido, todo se consulta mandando mensajes a los otros actores.



# Otras alternativas a Akka

- ▶ Existen diferentes alternativas para la implementación basada en actores:
  - Erlang: Nativos.
  - Scala: Implementación propia hasta la versión ~11.
  - Gpars: Framework que incorpora actores a Groovy.
  - Pikka: Implementación de actores en Python.
  - Celluloid (Ruby), Akka.NET (C# y F#), ...
- ▶ No obstante, es importante destacar que no todo se soluciona con actores. Existen otros modelos y arquitecturas muy utilizadas en la industria:
  - Basado en procesos.
  - Basado en hilos.
  - Programación asíncrona.
  - Modelo reactivo.

# ¿Quién usa Akka en la vida real?

- ▶ Aunque Akka parezca una librería solo apta para “aprobar” DAS, muchas compañías utilizan Akka para trabajar:
  - Intel: Procesos de Streaming.
  - Samsung: Sistemas reactivos.
  - LinkedIn: Play Framework.
  - DataBricks: Apache Spark.
  - Heluna: Servicio Anti-Spam
  - Usos variados para IoT, ....
- ▶ Podemos consultar más en:  
<https://www.typesafe.com/resources/case-studies-and-stories>
- ▶ En mi vida profesional, he utilizado *Akka* para distribuir el procesamiento de datos a la hora de realizar procesos de Big Data.



# Ejemplo práctico

- ▶ Dejémonos de teoría, vamos a realizar un caso práctico parecido al que tenéis que implementar en la práctica.
- ▶ He decidido utilizar Java para facilitar la vida a los alumnos (aunque me gusta más Scala ;)).



- ▶ <https://www.youtube.com/watch?v=S5eLybE03gc>

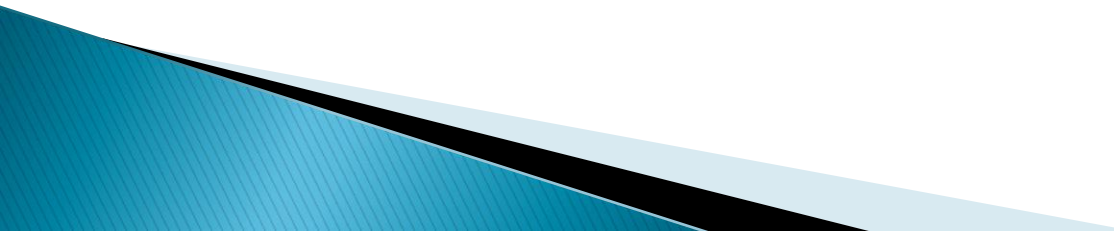


# Vamos a resumir un poco

- ▶ *“Las tijeras cortan el papel, el papel cubre a la piedra, la piedra aplasta al lagarto, el lagarto envenena a Spock, Spock destroza las tijeras, las tijeras decapitan al lagarto, el lagarto se come el papel, el papel refuta a Spock, Spock vaporiza la piedra, y, como es habitual... la piedra aplasta las tijeras.”*



# ¿Qué pasos ejecutaremos para realizar el ejemplo?

- ▶ Elegimos el tipo de enrutado: *Round Robin*.
  - ▶ Elegimos el protocolo: Maestro-Esclavo.
  - ▶ Elegimos los actores del sistema: *Master*, *Player*.
  - ▶ Elegimos posibles estados: *PlayGame*, *Result* y *FinishGame*.
  - ▶ Sistema de comunicación: Comunicación entre actores a través de un paso de mensajes.
- 

# Planificación Round-Robin

- ▶ *Round-robin* es un método para seleccionar todos los elementos en un grupo de manera equitativa y en un orden racional, normalmente comenzando por el primer elemento de la lista hasta llegar al último y empezando de nuevo desde el primer elemento.
- ▶ También es conocido como “secuencia para tomar turnos”.



# Master

- ▶ Su principal tarea es mandar las "tareas" a los jugadores.
- ▶ Para ello utilizaremos la siguiente información:
  - A cada jugador se le asigna una mano y un identificador único. Además se le asigna una referencia al propio máster (por facilidad).
  - Cuando recibo un "Result" de un jugador, lo almacena. Si recibe todas las respuestas, obtenemos el resultado.
  - **IMPORTANTE:** El Máster tiene asignado un Listener que se encargará de obtener los resultados finales. Esto se hace para dejar libre al maestro con el fin de escuchar nuevas peticiones.



# Slave (Player)

- ▶ Su principal tarea es obtener un resultado utilizando los estados disponibles: Piedra, papel, tijeras, lagarto y Spock.
- ▶ Para ello utilizaremos la siguiente información:
  - El estado del jugador (posición de la mano), dependerá de un número aleatorio (como ejemplo).
  - Una vez obtenido el resultado, podemos dormir el agente durante un número indeterminado de tiempo.
  - Para finalizar, el jugador avisará al máster que ha finalizado, enviando como mensaje el resultado de su mano





# Listener

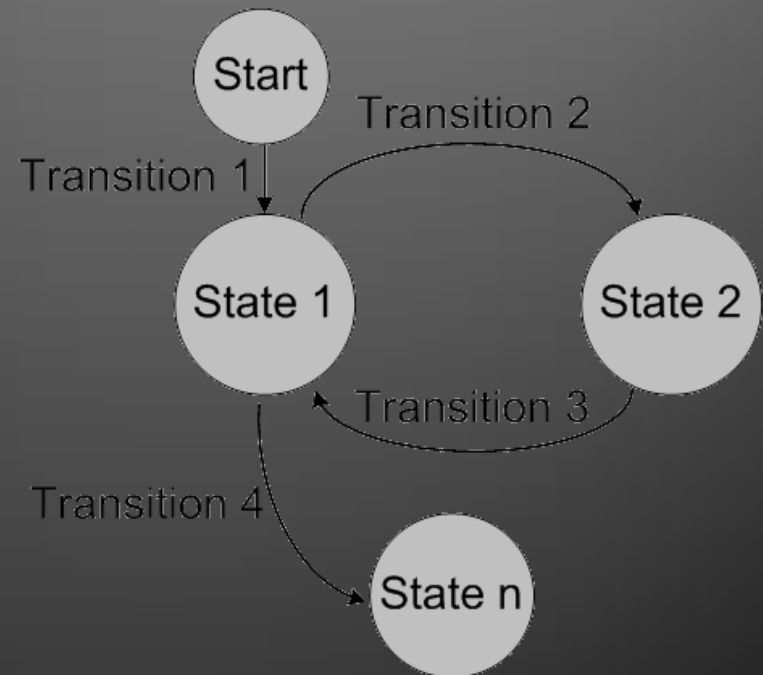
- ▶ Su principal tarea es procesar la salida final cuando todos los jugadores muestran su resultado.
- ▶ También avisa al actor master si ha acabado el juego o si hay que seguir.





# Estados

- ▶ Podemos pensar que es una FSM (Máquina de Estados Finitos).
- ▶ ***PlayGame:***
  - Estado inicial del actor Master. Se encarga de llamar a los jugadores asignándoles un turno (Hand).
- ▶ ***Result:***
  - Este estado es recibido por el Master cuando un Slave finaliza. Cuando tiene todas las respuestas, envía los resultados al Listener.
- ▶ ***FinishGame:***
  - En este estado el actor notifica de que se va a destruir.



# CODE TIME



ONEEYEDMAN THE HUMAN



DEUMAN THE CAT

# ¿Dónde ampliar información?

- ▶ Página web de presentación del TFG (incluye códigos fuentes y memoria):  
<http://maldicion069.github.io/tfg-gis/>
- ▶ Documentación oficial de Akka:  
<http://akka.io/>



A medium shot of Sheldon Cooper from the TV show 'The Big Bang Theory'. He is sitting in a black office chair, looking slightly to his left with a neutral expression. He is wearing a purple and grey striped t-shirt over a red long-sleeved shirt. The background is a blurred office setting with shelves and a computer monitor.

# ¿Alguna pregunta?

- ▶ De la práctica no ;)

# Código fuente:

[https://bitbucket.org/maldicion069/das\\_akka\\_java](https://bitbucket.org/maldicion069/das_akka_java)

# Y de regalo la versión Scala:

[https://bitbucket.org/maldicion069/das\\_akka\\_scala](https://bitbucket.org/maldicion069/das_akka_scala)

(Ojo: Completado, pero no optimizado al máximo)

Para cualquier pregunta, mi correo personal: [ccrisrober@gmail.com](mailto:ccrisrober@gmail.com)

