Creating Actors Using Coroutines

Kevin Jones

@kevinrjones www.rocksolidknowledge.com

Actors

Why actors?

What are actors?

What Are Actors?

Actors

- Lightweight process
- No shared state
- Communication via messages

Sound familiar?

- Channels with state

Why Actors?

Actors provide a way of protecting data

- rather than a mutex or critical section

Actors never share state

- So never need to compete for locks

Using Actors

Kotlin has an 'actor' coroutine builder

Motivating Actors

How do we protect shared state?

- Volatiles?
- Atomic types
- Locks
- Thread confinement

Demo

Shared state

Actors

Three parts to an actor

- Coroutine
- State
- Messages

```
fun myActor() = actor<MessageType> {}
```

Creating an Actor

'actor' coroutine builder

Actors are also typed

Demo

Using actors to share state

Summary

Actors are channels with state

- Avoid some of the pitfalls of concurrency
- Are lightweight
- Are directly supported by Kotlin

What's Next