Virtusity.com -- IT training by professionals for professionals

Lightbend Akka for Scala - Professional
by Paweł Lipski

bio: Scala Engineer @ VirtusLab, with a long-standing inclination towards git hacking :) Currently maintaining git machete (https://github.com/VirtusLab/git-machete), aka "probably the sharpest git repository organizer & rebase workflow automation tool you've ever seen" as a hobby project.

Level: Intermediate - knowledge of and practical experience with Scala is assumed
Length: 2 days - 8 hours each day, plus breaks
Approach: Extensive hands-on coding - students develop a workshop case study and produce a fully functional application that is event-driven and resilient
Requirements: Attendees bring their own laptops with Java 8

next sessions: July 6th and July 13th

About

Lightbend Akka for Scala - Professional
This course introduces experienced Scala developers to the reactive Akka toolkit. The combination of hands-on work and exercises in this course provide the perfect environment to best learn to use Akka with Scala.

Participants
Developers with basic knowledge of Scala, as covered in “Lightbend Scala Language - Professional”
Developers with a familiarity of Reactive Architecture, as covered in “Lightbend Reactive Architecture - Professional”
Developers who want to develop resilient, event-driven, scalable applications
Architects who want to have hands-on experience building Reactive Akka applications
Benefits
Developers gain knowledge and skills to design fault-tolerant apps using Akka
Certified Lightbend Trainers share how to configure, test, and tune Akka apps
Production readiness - create asynchronous, event-driven systems
Outline

- Introduction to Akka
    - Why Akka is Reactive
    - Akka's single model for Concurrency, Distribution, Fault Tolerance
- Actor Basics
    - The Actor Model
    - Anatomy of an Actor

    - Actors and Mutability
    - Actor Systems
    - Creating/Implementing Actors and Behaviors
    - Sending/Forwarding Messages
    - Sender
    - Child Actors
    - Actor Selections
    - Actor State
    - Scheduler
- Testing Actors
    - Synchronous Unit Testing with TestActorRef
    - Asynchronous Unit Testing with TestProbe
- Actor Lifecycle
    - Starting/Stopping Actors
    - Lifecycle Hooks
    - Death watch
- Fault Tolerance
    - Let it Crash design philosophy
    - Supervision
    - Supervision Directives
    - Restart Hooks
    - Self Healing
    - Routers and Dispatchers
    - Concurrency vs Parallelism
- Routers and Routing Strategies
    - Group Routers vs Pool Routers
    - Dispatchers and Dispatcher Types
- Modifying Actor Behavior
    - Become and Unbecome
    - Stash
- Ask Pattern
    - Ask Pattern and Pipe Pattern
- Akka Extensions
    - Creating and using Akka Extensions
- Finite State Machine
    - Using the FSM DSL to implement Finite State Machines

authorized by Lightbend