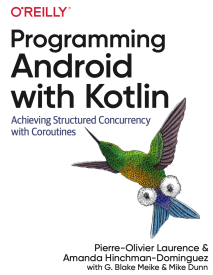
Forensics Masterclass: Coroutine Mastery & Diagnosing Bottlenecks

Advanced Specialty Workshop Package: Mastering coroutines and developing your style of structured concurrency



1. ABSTRACT

Despite its title, O'Reilly's *Programming Kotlin with Android: Achieving Structured Concurrency with Coroutines* applies far beyond the mobile ecosystem. It is a deep-dive blueprint for the JVM. Inspired by excerpts and advanced patterns from the book, this intensive masterclass challenges seasoned developers to break down, stress-test, and leverage Kotlin coroutines in unconventional and high-performance environments.

By moving past basic coroutines syntax, attendees will expose the leaky abstractions of modern concurrency and learn to diagnose complex threading issues under heavy production loads. This workshop is designed to help you gain the ability to diagnose and detect concurrency bottlenecks and structure your coroutines so that jobs are never lost, and parent jobs always wait.

Included in the workshop is a signed copy of *Programming Android with Kotlin.* Breaks are included in estimated times.

2. INSTRUCTOR INFO

Amanda Hinchman-Dominguez is an Android engineer, Kotlin GDE '20-'24, and co-author of *Programming Android with Kotlin: Achieving Structured Concurrency with Coroutines* and author of the Droidcon Academy Master code lab, *Mastering Kotlin Coroutines: From Basics to Advanced*. She hosts the Chicago Kotlin User Group and is a regular international speaker specializing in metaprogramming, Android, performance topics, and the Kotlin compiler.

2.2 Target Audience

For all Kotlin engineers who understand the basics of coroutines but are finding performance problems in their own systems.

3. WORKSHOP DESCRIPTION

Workshop attendees will learn the theories by concept, which will be reinforced with digestible coding exercises to strengthen their understanding. Finally, there will be an intensive hands-on portion on how to measure performance and profile for performance bottlenecks using Perfetto, and how to set up their own tools to track for concurrency regressions.

Total workshop estimation: 7 hours

3.1 Concurrency in Java and Bridging the Threading Gap

Heavy on theory for the first hour - attendees will examine existing thread models in Java, memory overhead, and historical concurrency pitfalls.

Analyze exactly how Kotlin coroutines map to underlying JVM threads, and where they scale, where traditional threads fall.

Lab topics (est. completion: 1 hour)

3.1.1 Thread management
3.1.1.a. Thread v. Runnable
3.1.1.b. Are threads expensive?
3.1.3. ThreadPools
3.1.4. Executors & ExecutorService
3.1.5. Limitations of the threading model
3.1.6. Threading pitfalls
3.1.7. Coroutine introduction & comparisons

3.2 Coroutine Low-Level Mechanics: A Visual Deep Dive

Step inside the compiler: how the state machine actually transforms suspended functions. Here, attendees will get started on the hands-on portion and inspect compiled code, and understand how coroutine components look, function, and exist in hierarchy when intercepting them with coroutine debugging.

Lab topics (est. completion: 1 hour)

3.2.1. Advanced Debugging Coroutine Setup
3.2.2. Decompiling suspend functions
3.2.3. Context Interception Mechanisms by tracing CoroutineContext
3.2.4. Job Lifecycle & Hierarchy Training

3.3 Forensic Diagnostics: Leaks, Blocks, and Stress-Testing

Hands-on labs focusing on diagnosing coroutine leaks, blocked threads, and conditions.

Attendees will also implement advanced testing patterns that can catch concurrency regressions in your CI/CD before they're ever released.

Lab topics (est. completion: 3 hours with lunch break)

3.3.1. Healthy v. Unhealthy Concurrency
3.3.2. Recording performance with Perfetto
3.3.3. Reading Perfetto traces for bottlenecks
3.3.4. Creating your own performance metrics with logs
3.3.5. Advanced Testing Patterns

3.4 Advanced Structured Concurrency & Architectural Style

Lab focusing on design patterns for robust error propagation, supervision for coroutines, and structured cancellation.

For the final section, attendees work on developing a production-grade style guide for asynchronous system design.

Lab topics (est. completion: 1.5 hours)

3.4.1. Suspending functions
3.4.2. Coroutine Cancellation
3.4.2.a. Cancelling Delegated Tasks
3.4.2.b. Causes of Cancellation
3.4.7. supervisorScope
3.4.8. Exposed exceptions
3.4.8 Coroutine Exception Handlers