Inhalt

TEIL IV Backend und Server

Vorwort

Kotlin ist eine moderne Programmiersprache, deren erste Version 2011 veröffentlicht wurde. Seit 2016 gilt Kotlin als stabil. 2017 hat Kotlin gleichsam den Segen von Google erhalten und wurde neben Java und C++ die dritte *First Class Language* zur Entwicklung von Android-Apps. 2019 hat Google seine Meinung nochmals geändert: Kotlin ist seither *die* empfohlene Programmiersprache zur App-Entwicklung. Die Devise lautet jetzt: **Kotlin first**.

Kotlin als reine App-Sprache zu bezeichnen, greift aber zu kurz. Kotlin findet auch im Backend-Segment eine immer größere Verbreitung. Hilfreich ist dabei, dass Kotlin den Alles-oder-nichts-Ansatz vermeidet: Es ist möglich, bei einer ursprünglich in Java entwickelten Server-Anwendung die vorhandenen Klassen in Java zu belassen, neue Klassen aber in Kotlin zu programmieren und auf diese Weise eine existierende Code-Basis Schritt für Schritt zu modernisieren.

Unsicher bin ich mir, ob Kotlin eine ideale Programmiersprache für Einsteiger ist. Noch immer werden Generationen von Schülern und Studenten mit Java als *First Programming Language* beglückt. Aufgrund der konsistenteren Syntax wäre Kotlin hierfür viel besser geeignet. Nie wieder müsste ich im Unterricht begründen, dass bestimmte Java-Features (z. B. Arrays) eben so sind, weil man es vor 20 Jahren nicht besser wusste und weil man grundlegende Funktionen heute aus Kompatibilitätsgründen nicht mehr ändern kann. Die viel jüngere Sprache Kotlin ist diesbezüglich natürlich im Vorteil. Dem stehen aber überbordende Syntaxvarianten und die Präferenz für Lambda-Ausdrücke entgegen, die gerade Anfänger überfordern werden.

Warum Kotlin?

Mit Kotlin können Sie die gleichen Programme, Apps und Backend-Anwendungen schreiben wie mit Java. Der Unterschied besteht darin, dass gleichwertiger Kotlin-Code viel eleganter, kompakter und besser wartbar ist. Das Programmieren macht deutlich mehr Spaß!

Ein weiterer Vorteil besteht darin, dass Kotlin in gewöhnlichen Variablen den Zustand `null` nicht zulässt. *Null Pointer Exceptions* sollten damit der Vergangenheit angehören. Kotlin-Code ist deswegen erwiesenermaßen weniger fehleranfällig als Java-Code. (Es ist übrigens auch in Kotlin möglich, eine Variable als *nullable* zu deklarieren. Der Zugriff auf den Inhalt der Variable wird dann durch spezielle Mechanismen abgesichert, die die versehentliche Verarbeitung von nicht initialisierten Werten ausschließen.)

Trotz des Umstiegs auf Kotlin müssen Sie weder auf bewährte Java-Bibliotheken und -Frameworks verzichten noch auf moderne Entwicklungsumgebungen. Ihr resultierendes Programm kann auf jeder *Java Virtual Machine* (JVM) ausgeführt werden. Bemerkenswert ist, dass sich Kotlin mit der JVM 8 zufriedengibt. Somit stehen auch in Umgebungen, wo nicht die neueste JVM installiert ist, sämtliche Kotlin-Features zur Verfügung. Das ist gerade bei Backend- und Android-Apps ein riesiger Vorteil: In diesen Umgebungen tut sich mittlerweile eine riesige Lücke zwischen der gerade aktuellen Java-Version und dem verfügbarem Software-Stack auf, der auf dem Gerät bzw. Server verfügbar ist. Oracles halbjährliche Versions-Updates haben diese Situation nur noch verschlimmert.

In der Praxis kommt heute heute überwiegend die JVM als Kotlin-Plattform zum Einsatz. Darüber hinaus gibt es zwei weitere Kotlin-Varianten, die auf JavaScript aufsetzen bzw. nativ kompiliert werden. Längerfristig könnte das Kotlin die Möglichkeit geben, sich ganz von der JVM zu emanzipieren.

Aus meiner Sicht hat Kotlin für Java-Entwickler eine ähnliche Relevanz wie Swift für Objective-C-Entwickler: Kotlin ist ein Sprung in ein neues Zeitalter der Programmierung und befreit Sie von altem Ballast. Die Syntax von Java kann aus Kompatibilitätsgründen nie so umfassend modernisiert werden, wie dies in Kotlin der Fall ist. Manchmal hilft eben nur ein Neuanfang!

Dieses Buch

Ich habe dieses Buch in vier Teile gegliedert:

- **Kotlin:** Teil I beschreibt systematisch die Syntax von Kotlin. Falls Sie bisher in Java programmiert haben, muss ich Sie an dieser Stelle warnen: Die Syntax von Kotlin ist grundlegend anders und wesentlich vielseitiger als die von Java. Auch wenn die Sprachen auf JVM-Ebene kompatibel sind – für die Syntax gilt dies nicht!

  Zu den inhaltlichen Schwerpunkten im ersten Teil zählen der Umgang mit Variablen und elementaren Datentypen (Zeichenketten), die Verwendung von Listen, Sets und anderen Aufzählungsklassen sowie die Anwendung von Funktionen und Lambda-Ausdrücken (funktionale Programmierung).

  Weiter geht es mit den Features zur objektorientierten Programmierung. Kotlin hat diesbezüglich eine Menge anzubieten: Computed Properties, Datenklassen, Delegation, Extensions (also die nachträgliche Erweiterung von Klassen), Receiver- und Infix-Funktionen, Operator Overloading etc. Sogenannte *Type-safe Builder* ermöglichen es, auf Basis dieser Features eigene, Kotlin-ähnliche Sprachen zu formulieren (*Domain-specific Languages*). Diverse Kotlin-Frameworks machen davon Gebrauch, z. B. Ktor.

Teil I ist keineswegs so trocken und theoretisch, wie es hier den Anschein hat: Eingestreute Beispiele zeigen, wie Sie Kotlins Sprachmerkmale praktisch anwenden, z. B. um den Weg aus einem Labyrinth zu suchen oder um ein Quiz zu programmieren.

- **Programmiertechniken:** Sie wollen Code asynchron ausführen? Auf Dateien zugreifen? XML- und JSON-Dokumente verarbeiten? Objekte serialisieren? Mit Datenbanken arbeiten? Dann liefere ich Ihnen in Teil II konkrete Anleitungen. Unter anderem lernen Sie hier Koroutinen kennen. Das ist Kotlins Konzept zur asynchronen Programmierung. Außerdem stelle ich Ihnen das Datenbank-Framework *Exposed* vor.
- **App-Programmierung:** Viele Entwickler lernen Kotlin primär, um damit Apps zu programmieren. Teil III beschreibt den Umgang mit Android Studio und bildet ein erstes Fundament, das Ihnen bei der Orientierung in der komplexen Welt der Android-Frameworks helfen wird. Zwei größere Beispiele, ein Währungsumrechner und eine App zur Evaluierung von Lehrveranstaltungen, helfen Ihnen beim Sprung von der Theorie zur Praxis.

  Ich konzentriere mich in diesem Teil auf moderne Bibliotheken und bemühe mich, Ihnen die App-Entwicklung frei von Altlasten zu präsentieren. In einem eigenen Kapitel stelle ich Ihnen das ganz neue (zum Zeitpunkt der Fertigstellung dieses Buchs aber leider noch unausgereifte) Toolkit *Jetpack Compose* vor.
- **Backend/Server:** Dass Java bis heute eine derart weit verbreitete Programmiersprache ist, hat nicht zuletzt mit ihrer Dominanz im Backend- und Server-Bereich zu tun. Aber es ist nicht alles Gold, was glänzt. Wer mit Java für das Backend entwickelt, ist in der Regel auf die Features von Java 8 limitiert. Der resultierende Code sieht fast so steinzeitlich aus wie der von uralten Cobol- oder Fortran-Programmen, die noch in manchen Banken und Versicherungen laufen. Entsprechend groß ist der Frust der Entwickler, die diese alte Code-Basis erweitern müssen. Kotlin bietet auch hier einen Ausweg: Sie können alte Projekte um neue Kotlin-Klassen erweitern, die nahtlos mit dem vorhandenen Code zusammenspielen.

  Die Aspekte der Backend-Programmierung sind jedoch sehr komplex, und die Integration in eine bestehende Java-Codebasis hängt davon ab, wie Ihre individuelle Umgebung aussieht. Deswegen konzentriere ich mich in Teil IV darauf, Ihnen das Kotlin-eigene Framework Ktor vorzustellen. Damit können Sie moderne, von Grund auf asynchron konzipierte Server-Anwendungen programmieren. Auf zwei Grundlagenkapitel folgt auch hier ein konkretes Beispiel – eine Server-App mit REST-API zur Evaluierung von Lehrveranstaltungen in einer Schule oder Universität.

Auch wenn ich in Teil I mit den Basics beginne (Operatoren, Variablenverwaltung etc.), gehe ich in diesem Buch davon aus, dass Sie bereits programmieren können. Ich werde Ihnen also nicht erklären, wozu Variablen dienen, was eine Schleife ist und

warum es sinnvoll ist, objektorientiert zu programmieren. Vielmehr konzentriere ich mich darauf, Ihnen zu zeigen, wie Sie diese elementaren Konstrukte in Kotlin optimal nutzen und welche Möglichkeiten und Eigenheiten Kotlin dabei bietet.

Es ist nicht erforderlich, dass Sie mit Java vertraut sind – es schadet aber auch nicht. Da Kotlin für Java-Entwickler besonders attraktiv ist, weise ich gelegentlich auf Unterschiede zwischen Java und Kotlin hin.

Eine neue Welt

Kotlin zählt aktuell zu den fortschrittlichsten Programmiersprachen. Die Sprache vereint die besten Ideen, die sich in den letzten 20 Jahren etabliert haben. Sie können damit Algorithmen und Apps auf eine moderne, elegante Art und Weise realisieren, ohne sich ständig über Anachronismen aus der Vergangenheit zu ärgern.

Egal, ob Sie aus der Java-Welt kommen oder nicht: Kotlin beweist, dass Programmieren auch heute noch Spaß machen kann!

Michael Kofler (*https://kofler.info*)