目次

第1部

Kotlin入門

第1章　Kotlinをお勧めする理由

この章では、Kotlinの基礎について説明します。言語の生い立ちやサーバーサイドでの利用意義、特徴的な機能や基本的な構文など、まずはKotlinという言語そのものについて知っていただければと思います。

1　なぜKotlinが誕生したのか？

Kotlinは、IntelliJ IDEAなどのIDE（Integrated Development Environment、統合開発環境）で有名なJetBrains社が開発したプログラミング言語です。正式版の1.0リリースが2016年2月と比較的新しい言語になります。

JVM上で動く、いわゆるJVM言語[注1]の一種です。JetBrains社のKotlin開発メンバーが執筆している書籍『Kotlinイン・アクション』[注2]でも紹介されているのですが、もともとはJetBrains社のJavaで開発をしているチームが、C#で開発をしている.NETチームをうらやましく思い、Javaに代わるモダンな言語として開発したという話があります。そのためJavaと比較してシンプルな構文になっていたり、関数型やコルーチンなど最近の言語でよく見られる機能もしっかりと入っていて、開発効率やシステムの品質担保の面でも非常に優れたものになっています。

また、JetBrains社はIntelliJ IDEAをはじめとするIDEをJavaで開発しており、多くのJavaの資産を抱えています。そのため、その資産を失うことは生産性の低下につながると考え、Javaとの相互運用ができることを前提条件としました。そのためKotlinはJavaとの相互互換となっており、もともとJavaを使用しているプロジェクトでは、特に利用価値の高い言語になっています。

注1　他のJVM言語として、Scala、Groovyなどがあります。
注2　Dmitry Jemerov、Svetlana Isakova 著、長澤太郎、藤原聖、山本純平、yy_yank 監訳、マイナビ出版、2017年

2 Kotlinでなにを作れるのか？～サーバーサイドでの利用意義

様々なプラットフォームで使用できるKotlin

Kotlinの利用シーンとして、最も有名なのはAndroidアプリの開発だと思います。2017年に行われたGoogle I/OでAndroidの公式の開発言語としてサポートすることをGoogleが発表しました。さらに2年後のGoogle I/O 2019では推奨言語として、Kotlinファーストを強めていくということが発表され、今後も一般的に使われていくと思われます。

また、iOSやMac、Windowsのアプリへのネイティブバイナリを生成できるKotlin/Native[注3]や、JavaScriptのコードを生成できるKotlin/JS[注4]などを使い、Kotlinのコードから様々なプラットフォームで実行するプログラムを作れます。JetBrains社のIntelliJ IDEAでは、これらのマルチプラットフォームのコードを作るための、Kotlin Multiplatform Project[注5]（通称Kotlin MPP）というプロジェクト構成も用意されています。

その中で、Androidでの開発と並び一般的な使いどころとして挙げられるのが、サーバーサイド開発です。

サーバーサイドでの利用意義

前述の書籍『Kotlinイン・アクション』でも、

> Kotlinを使う最も一般的な場面は、次の2つでしょう。
>
> ・サーバーサイドの実装（通常はWebアプリケーションのバックエンド）
> ・Androidデバイス上で動くモバイルアプリケーションの実装
>
> ----------
> 『Kotlin イン・アクション』より

と書かれており、サーバーサイド開発での利用がベターな方法の一つであることがわかります。

その理由として、まずJavaやC#をはじめとする、静的型付け言語のメリットがそのままKotlinにも当てはまります。例えば次のようなものがあります。

- インタプリタ型の言語に比べ高いパフォーマンス
- 大人数、大規模の開発や長期運用でも保守性の高い設計をしやすい言語仕様

注3　https://kotlinlang.org/docs/native-overview.html
注4　https://kotlinlang.org/docs/js-overview.html
注5　https://kotlinlang.org/docs/multiplatform.html

こうしたメリットのある言語の中でも、比較的新しくモダンな構文や機能が用意されており、開発効率も高い言語と言えます。また、後述するNull安全な言語仕様もあり、システムの品質や保守性の面でも高めることができます。

さらに、Javaとの相互互換があることは前述しましたが、Javaの最もメジャーなフレームワークであるSpring FrameworkがKotlinをサポートしていることも挙げられます。新しい言語を使う際、どうしてもフレームワークやその周辺のエコシステムは発展途上で、迷ってしまうことも多いです。そんな中でSpring Frameworkという実績あるフレームワークが選択肢として存在するのはとても大きいことです。もちろんKotlin製のフレームワークも開発されており、今後そちらも発展してさらに質の高い開発ができるようになっていくことへの期待も大きいです。

3 コードの安全性を高めるKotlinの型とNull非許容／許容

KotlinのNull安全とは?

前述しましたが、Kotlinの言語仕様の大きな特徴の一つとして、**Null安全**が挙げられます。Kotlinの機能として大きなメリットになる部分ですので、先に紹介します。**リスト1.3.1**を見てください。

リスト1.3.1

```
val str1: String = null  // Null非許容、コンパイルエラーになる
val str2: String? = null // Null許容
```

Kotlinでは変数を宣言する際、型（この場合は`String`）に何も付けずに宣言するとNull非許容になり、Nullを入れるとコンパイルエラーになります。?を付けることで変数にNullを入れることができるようになります。

このように型を宣言する段階でNull非許容／許容を明示し、誤った箇所でNullを入れようとするとコンパイルの時点で防いでくれるため、NullPointerExceptionの発生を未然に防ぐことができます。また、型のデフォルトがNull非許容ということにもなるので、通常はNull非許容とし、必要になった箇所だけNull許容にすると意識でき、不要なNullの扱いを減らすこともできます。

そして、Null許容の型の変数にもさらに安全性を保つ仕様があります。Null許容にした場合、そのオブジェクトの関数等に対して、ただアクセスするとコンパイルエラーになります。**リスト1.3.2**の例では、引数でNull許容のString型である`message`という引数を受け取り、`length`にアクセスしようとしていますが、Nullが入っていないことを保証されていないためコンパイルエラーになります。

第2章　様々なKotlinの機能

第2章では、Kotlinで使用できる様々な機能を紹介します。第1章でも紹介したように、Kotlinはシンプルに書くための構文や便利な機能で、開発効率を上げられる要素が多くあります。それらはKotlinのメリットを最大限に活かした実装をしていくために、必須の知識にもなってきます。この章でKotlinの良さを理解しつつ、より「Kotlinらしい」書き方を知っていただければと思います。

1　if、when文を式として扱いコードをシンプルにできる

第1章の「5. Kotlinの基本構文」で、分岐処理としてif文とwhen文について紹介しました。Kotlinではこのif文とwhen文が式として扱われるため、結果の値を返すことができます。例えば**リスト2.1.1**のようなコードがあります。

リスト2.1.1

```
fun printOddOrEvenNumberText(num: Int) {
    var text = ""
    if (num % 2 == 1) {
        text = "奇数"
    } else {
        text = "偶数"
    }

    println(text)
}
```

空文字で初期化したtextという変数を用意し、引数の値によって「奇数」「偶数」という文字列で書き換え、出力しています。これを**リスト2.1.2**のように実装できます。

第3章 JavaとKotlinの相互互換が既存の資産を生かす

第1章の冒頭でも説明しましたが、KotlinはJavaとの相互互換を持っている言語です。KotlinからJava、JavaからKotlinを呼び出すことができ、2つの言語を同一のプロジェクトで共存させることも容易にできます。もちろんKotlinのみで実装することも可能ですが、もしJavaを使用していた組織であれば、長らく開発してきた資産を活用できることは大きなメリットです。この章では、主にKotlinからJavaを使用する方法を中心に、Javaとの相互利用について説明していきます。

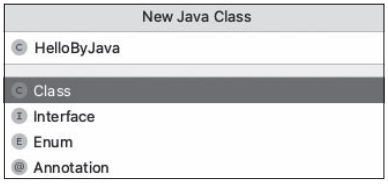
1　Javaのコードを呼び出す

まずは、KotlinからJavaのコードを実行してみます。第1章で作成したプロジェクトで、src/main配下にjavaというディレクトリを作成してください。src/mainディレクトリを右クリックし、[New]→[Directory]を選択し、名前を入力すると作成できます。そしてsrc/main/java配下にJavaのクラスを作成します。Javaのファイルは、IntelliJ IDEAで対象のディレクトリ（ここではsrc/main/java）を右クリックし、[New]→[Java Class]を選択すると作成できます（**図3.1**）。

図3.1

任意の名前（ここではHelloByJava）を入力すると（**図3.2**）、同名のクラスを持ったファイルが作成されます。本章でのコードはすべて、Javaの場合はsrc/main/java、Kotlinの場合はsrc/main/kotlin配下に作成します。

図3.2



第4章 Webアプリケーション開発の基盤となるSpring Bootを導入する

本章ではSpring Bootというフレームワークを使って、Webアプリケーションのサーバーサイドプログラムを実装する方法を解説します。KotlinでのWebアプリケーション開発において、フレームワークの利用は必須になってきます。様々なフレームワークの中でもSpring Bootは特にメジャーなものとなっており、第6章から開発する実践のアプリケーションでも使用していて、アーキテクチャのベースとなる知識になってきます。ここまでの章はKotlinという言語自体についての説明でしたが、ここからいよいよ「サーバーサイドKotlin」の開発を体感していただければと思います。

1 Spring Bootの導入

Spring Bootとは?

Spring Boot[注1]は、Webアプリケーションフレームワークの一つです。Javaのフレームワークとして最もメジャーなものの一つで、多くのサーバーアプリケーションで使用されています。

Spring Framework[注2]というフレームワークがあり、もともとはDI（Dependency Injection、依存性注入）やAOP（Aspect Oriented Programming、アスペクト思考プログラミング）をサポートするものでした。リリース後に多くの機能が作られてゆき、Webアプリケーション開発のためのSpring MVC、認証・認可を実装するためのSpring Securityなど様々なフレームワーク[注3]の集合体となっています。

その様々なフレームワークを個々で使うのではなく、まとめて使いやすい形にしてWebアプリケーション開発を簡単にできるようにしたものが、Spring Bootになります。

Spring FrameworkのKotlinサポート

Spring Frameworkは、5系から正式にKotlin対応を始めています。Spring Bootで言うと2系が、Spring Framework 5系に対応したバージョンです。4系以前のバージョンでも使えるのですが、フレー

注1　https://spring.io/projects/spring-boot
注2　https://spring.io/projects/spring-framework
注3　https://spring.io/projects

第5章 O/Rマッパーを使用してデータベースへ接続する

リレーショナルデータベースは、多くのWebアプリケーションのサーバーサイド開発において使われているミドルウェアです。それをプログラムから扱うためのフレームワークであるO/Rマッパーも、とても重要な要素になってきます。次の第6章から作成する実践のアプリケーションでも使用しています。本章では、MySQLで構築したデータベースに対し、O/RマッパーのMyBatisを使用してKotlinからアクセスするコードを実装し、Kotlinでのデータベースの扱いを学んでいただければと思います。

1 MyBatisとは？

MyBatisは、Java製のO/Rマッパーの一つです。もともとはXMLにSQLを記述し、コードで定義した関数と紐付(ひもづ)けることでSQLの発行や、データベースの操作を実現するものでした。

最近のバージョン（執筆時点での最新バージョンは3.5.6）では、MyBatis Dynamic SQLというコード上でクエリを構築できる方式が追加されており、その実行に必要なコードをKotlinで生成するGeneratorも用意されています。そのためもともとJava製ではありますが、Kotlinからも扱いやすいO/Rマッパーとなっています。

2 DockerでMySQLの環境構築

MyBatisを使用するにあたり、先にローカル環境でデータベースを使えるようにします。今回使用するのはMySQLです。

Docker Desktopのインストール

ローカルにMySQLを直接インストールするのではなく、Dockerのコンテナを立ち上げる形で用意します。そのため、先にDocker Desktopをインストールしてください。

第6章 Spring BootとMyBatisで書籍管理システムのWebアプリケーションを開発する

第5章までで、サーバーサイドKotlinでの開発に必要な主要な技術要素の解説が終わりました。本章からは、ここまで解説してきた要素を使用して、実践的なアプリケーションを作成していきます。第6章でSpring BootとMyBatisを使用したWebアプリケーションを作成し、第7章で認証・認可、第8章で単体テストの実装をして、より実際のプロダクトのような形に近づけていきます。

そのベースとなる部分を作成するのが本章です。まずはここまでの章で得た知識を使って実践的なアーキテクチャのアプリケーションを作成し、さらに理解を深めていただければと思います。

1 書籍管理システムの仕様

本章では、サーバーサイドKotlinの実践的な実装方法を学ぶため、書籍管理システムを題材としたサンプルアプリケーションを作成します。これは組織で所有する書籍の情報や、貸出、返却の状態を管理するアプリケーションのイメージになります。

最初に、システムの機能、仕様を説明します。

実装する機能

機能として、以下のものを実装していきます。

- ログイン、セッション管理
- 権限管理
- 書籍の一覧取得
- 書籍の詳細取得
- 書籍情報の登録
- 書籍情報の更新
- 書籍情報の削除

第7章 書籍管理システムの機能を拡充する

第7章では、第6章で作成したアプリケーションの残りの機能（貸出、返却）を主に実装していきます。それにあたりユーザー情報を扱うようにする必要があるため、Spring Securityを使用した認証、認可の機構の実装方法を解説します。また、追加の要素としてSpring AOPによるロギングの実装もしていきます。Spring Frameworkには様々なモジュールが用意されており、こういった多くのプロダクトにおいて共通で必要とされるような機能も比較的簡単に実装できます。本章ではこれらの機能を実装することで、より実践的なアプリケーションへと近づけていきます。

1 Spring Securityでユーザー認証、認可の機構を実装する

ここまで書籍に対する各種操作のAPIを実装してきましたが、次はユーザー認証、認可の機構を実装していきます。実装にはSpring Securityを使用します。Spring SecurityはSpringプロジェクトの一つで、Webアプリケーションで認証、認可などセキュリティ関連の機能を実現するためのフレームワークです。

第6章で作成したアプリケーションにログインの仕組みを実装し、前述のとおり検索系機能、更新系機能でアクセス権限を分けます。

build.gradle.ktsへの依存関係の追加

まず、Spring Securityを使うためにbuild.gradle.ktsへ依存関係を追加します。`dependencies`に**リスト7.1.1**を追加してください。

リスト7.1.1

```
implementation("org.springframework.boot:spring-boot-starter-security")
```

`spring-boot-starter-security`はSpring Securityを使うためのstarterです。MyBatisのstarterと同様、この記述でSpring Securityと併せて必要な依存関係をすべて追加してくれます。

第8章 JUnitで単体テストを実装する

第2部の最後となる本章では、単体テストの実装について解説します。実際のプロダクトを開発する上で、テストの自動化は必須になってきます。特に単体テストのテストコードの実装や、CI（Continuous Integration、継続的インテグレーション）での実行は多くのプロジェクトで導入されています。サーバーサイドKotlinでもそれは同様で、これから解説するJUnitというテスティングフレームワークを使用して実現されていることが多いです。

第7章までででシステムの機能としてはできあがっていますが、最後にいくつかのテストコードを書いて完成とします。サーバーサイドKotlinを「実践」していくためにも、ここで単体テストの実装についても習得し、実際に開発で使う際はテストコードも充実させていけるようにしていただければと思います。

1　JUnitの導入

JUnitとは?

JUnitは単体テストを実装するためのテスティングフレームワークです。Javaのテスティングフレームワークとして最もポピュラーなものの一つで、Kotlinのプロジェクトでも多く使用されています。

build.gradle.ktsへの依存関係の追加

JUnitの導入は、**リスト8.1.1**の依存関係をbuild.gradle.ktsのdependenciesに追加します。junit-jupiter-engineがJUnitを使ったテストを作成するための基幹のフレームワークになります。もう一つ追加しているassertj-coreは、AssertJ[注1]というテストの中の検証処理で使用するライブラリです。

リスト8.1.1

```
testImplementation("org.junit.jupiter:junit-jupiter-engine:5.7.1")
testImplementation("org.assertj:assertj-core:3.19.0")
```

注1　https://github.com/assertj/assertj-core

第9章 高速な通信フレームワーク gRPC

ここまでの章ではSpring BootやMyBatisといった、もともとJavaでメジャーなフレームワークとして使われていたものを、Kotlinに適用して基本的なWebアプリケーションを実装してきました。本章からはさらに一歩踏み込んで新しい技術スタックや、まだこれから発展していく段階のKotlin製のものなど、様々なフレームワークを組み合わせた使い方を紹介していきます。

まず本章では、マイクロサービスアーキテクチャのサービス間通信などでよく使用されている、gRPCを使った実装を紹介します。

1 gRPCとは？

gRPCは、Googleが開発しているRPC（Remote Procedure Call）フレームワークです[注1]。通信プロトコルとしてHTTP/2、通信のデータ形式としてProtocol Buffers[注2]を標準でサポートし、ハイパフォーマンスな通信を実現します。 現在はマイクロサービスアーキテクチャでの、サービス間通信などでよく使用されています。

Protocol BuffersでgRPC通信に関するコードを生成できる

gRPCで使用するProtocol Buffersは、IDL（Interface Description Language、インターフェース定義言語）を使用して通信のインターフェースを定義し、それを元に通信で使用するデータモデルや、シリアライズ・デシリアライズなどの処理が実装されたコードを、様々なプログラミング言語で生成することができます。また、データの通信はバイナリで行われ、そのバイナリをやり取りするクライアント、サーバーの処理を、生成されたコードを使用して実装します。

IDLは.protoという拡張子で作成し、例としては**リスト9.1.1**のような記述になります。

リスト9.1.1

```
service Greeter {
  rpc Hello (HelloRequest) returns (HelloResponse);
}
```

注1　https://grpc.io/
注2　https://developers.google.com/protocol-buffers

第10章 Kotlin製のWebフレームワーク Ktor

本章ではKotlin製のWebアプリケーションフレームワークである、Ktorを紹介します。Ktorはまだまだ採用事例の数は多くないですが、言語の開発元であるJetBrains社が開発しているKotlin製のフレームワークということもあり、サーバーサイド開発の技術選定の新たな選択肢として注目されています。実際に導入しているプロダクトも徐々に増えてきており、サーバーサイドKotlinをやる上ではぜひ知っておいていただきたい内容になります。

1 Ktorとは？

Ktorは、2018年11月に正式版であるVer1.0がリリースされた、Webアプリケーションフレームワークとなります。現状サーバーサイドKotlinの開発ではSpring Bootがフレームワークのベターな選択肢とされることが多いですが、新たな選択肢として期待できます。言語の開発元であるJetBrains社が開発していて、なおかつSpring Bootと違いKotlin製ということもあり、技術選定で名前が挙げられることも最近は多いです。

Spring Bootはかなり重厚で複雑なフレームワークになっており、ずっとJavaで使ってきた人には便利なのですが、初心者にはとっつきにくい面もあります。それに比べてKtorは軽量で扱いやすくなっており、アプリケーションの起動時間などの面でも有利に働きます。それでいて下記のようなWeb開発において必要な様々な機能（抜粋）を提供しています。

- 認証、認可
- HTTPクライアント
- WebSockets
- 非同期通信
- ロギング
- テンプレート

特に非同期処理を特徴としていて、フレームワーク内でもKotlin Coroutinesが多く使われています。

第11章 Kotlin製のO/Rマッパー Exposed

第10章ではSpring Bootに代わるWebアプリケーションフレームワークとしてKtorについて解説しましたが、本章ではKtorと同じくJetBrains社が開発しているKotlin製フレームワークである、O/RマッパーのExposedを紹介します。KotlinでO/Rマッパーを使用する際、本書で紹介しているMyBatisも含めJava製のものが使われることが現状は多いです。その中で数少ないKotlin製のものとあって、Kotlinエンジニアの間では前々から知られている存在です。こちらもまだまだ採用事例は少ないですが、サーバーサイド開発に必須なO/Rマッパーの選択肢の一つとなりうるので、覚えておいていただければと思います。

1 Exposedとは？

Exposedは、Kotlinの開発元であるJetBrains社が開発している、Kotlin製のO/Rマッパーです[注1]。SQLライクに実装できるDSL（Domain Specific Language）、軽量なDAO（Data Access Object）という2つのアクセス方法が用意されているのが特徴として語られます。

執筆時点での最新バージョンが0.29.1のため、まだプロトタイプの段階ですが、Ktorと同じくKotlin製のフレームワークとして有名なものの一つです。

2 Exposedの導入

IntelliJ IDEAで任意のGradleプロジェクトを作成し、build.gradle.ktsに設定を追加します。

まず**リスト11.2.1**のようにrepositoriesを変更し、jcenter()を追加します。

リスト11.2.1

```
repositories {
    mavenCentral()
    jcenter()
}
```

注1　https://github.com/JetBrains/Exposed

第12章 Kotlin製のテスティングフレームワーク Kotest、MockK

最後に紹介するのは、Kotlin製のテスティングフレームワークであるKotest、そして併せて使用するモックライブラリのMockKです。Kotlinの単体テストではJUnitを使用されることがまだまだ多いのですが、Kotlin製のテスティングフレームワークもいくつか存在します。その中でもKotestは開発が活発に行われており、多機能になっていてかなり使い勝手のいいものになっています。筆者も実際のプロダクトで使用してとても良かったと感じているフレームワークなので、ぜひその魅力を感じていただければと思います。

1 Kotestとは？

Kotest[注1]はKotlin製のテスティングフレームワークで、Kotlinで単体テストを柔軟に実装するための様々な機能が提供されています。以前はKotlinTestという名前でしたが、バージョン4.0からKotestへ変更されました。

様々なコーディングスタイルをサポート

Specという以下の10種類のコーディングスタイルが用意されており、様々な書き方がサポートされていることが一つの特徴として挙げられます。

- Fun Spec
- Describe Spec
- Should Spec
- String Spec
- Behavior Spec
- Free Spec
- Word Spec
- Feature Spec

注1　https://kotest.io/