

入門！実践！

サーバーサイドKotlin

こんなにおすすめ！

- そろそろKotlinをやってみたい
- ずっとJavaをやってきたけど新しい言語もやってみたい
- フレームワークを使用したサーバーサイドのWeb開発を経験してみたい
- ずっとレガシーな環境だったのでモダンな開発をしてみたい



入門!実践!サーバーサイド Kotlin

FORTE 著

2019-09-22 版 aozora Project 発行

はじめに

この本を手に取っていただきどうもありがとうございます。著者の FORTE (フォルテ) です。今回は「実践！ 入門！ サーバーサイド Kotlin」という技術書を書きました。この本は 12 年以上レガシーな環境で開発してきた筆者が、モダンな開発を勉強せねば！ と思い、勉強するなら技術書を書くかーと思って書いたのがこの本です。技術書を書くのが目的か、勉強が目的か自分でもよくわからないですが、両方とも大事なポイントなのは確かです。

サーバーサイド Kotlin を学びたい理由

筆者がモダンな開発を学ぶ対象としてサーバーサイド Kotlin を選んだのは次の理由からです。

- Android で Kotlin を使いたい
- ずっと Java をやってきたので互換性が高そう
- ここで Web のモダンなバックエンド開発を経験したい
- フレームワークを使用した Web 開発を経験したい

筆者は IT エンジニア歴 12 年以上のうち、ほぼ 10 年間 Java で Web アプリケーションを開発してきました。このことからまったく新しい言語ではなく、Java 互換の言語の方がハードルが低いと考えました。

さらに私は OSS などで一般に公開されているフレームワークを使用した開発経験が一度もありません。Servlet か、PHP の独自フレームワークでの開発しか経験がありません。そのため、フレームワークを用いた（自分にとっては）モダンな Web 開発が学びたいと思ったとき、需要にぴったりであると思ったのがサーバーサイド Kotlin を選んだ理由です。

どんな人向けか

今回この本を書くにあたり、ペルソナを自分としました。つまりずっと Java のレガシーな環境にいた IT エンジニア向けに書くということです。そんな IT エンジニアでもこの本を読めばサーバーサイド Kotlin が書ける、フレームワークを使った開発経験が積めることを目標に書き始めました。具体的には次のような人におすすめの本となっています。

-
- Kotlin をやってみたい
 - Java を書いたことあるけどそれ以外の JVM 言語を知らない
 - OSS のフレームワークを使って Web アプリを書いたことが無い
 - ずっとレガシーな Web 開発環境にいた（と思っている）
 - モダンな開発の勉強したいけど何をやっていいかわからない
 - 業務ではずっと Windows だったから Mac で例を出されても困る
 - ○○やってます（例：Rails、Spring Boot、React…）と言いたいが、今は言えない

前述したとおり、私自身が経験したことがない言語、経験したことがないフレームワーク（フレームワークの使用自体が初めてのようなもの！）という状況で本が書ければ、きっとこの本を読む“私”のようなエンジニアにも分かりやすく、またきっと自信になると思うのです。同じ様にレガシーな状況にいて少しでもモダンな開発がしたい人に届けば良いなと思っています。

この本で得られること

この本は Java などで Web 開発の経験がある人向けにサーバーサイド Kotlin に入門して実践してみる本です。この本を読み終わると次のような状態になります。

- Kotlin がなんだかわかる
- サーバーサイド Kotlin がなんだかわかる
- サーバーサイド Kotlin の開発環境が作れる
- サーバーサイド Kotlin で掲示板を作成した
- 掲示板を作成した際に以下の開発を経験した
 - Web ページ表示
 - DB 操作
 - テスト

あなたが本書を読むことでより早く簡単にサーバーサイド Kotlin に入門し実践していただけたら、さらに業務や趣味の開発に学べたことを生かしていただけたら、こんなに幸せなことはありません。

この本では解説しないこと

この本は前提として「Java などで Web 開発の経験がある人向け」となっています。そのため、次のように Web 開発の基礎（HTTP とは？ リクエストとは？ など）や Java の基礎については解説しません。解説しないことについては次の一覧をご覧ください。

- Web 開発の基礎

-
- Java の基礎
 - Kotlin の文法
 - フレームワークの詳細な解説

Kotlinについては極力解説を入れていきますが、すべての文法を説明しようとそれだけで本が一冊できてしまいます。クラス定義やメソッド定義くらいであれば、なんとなく見ただけでわかると思いますので、Javaと大きく変わっている点や特筆すべき点のみ本文にて補足していきます。もし不明な点があればぜひ公式サイト基本構文などをご覧ください。

<https://Kotlinlang.org/docs/reference/basic-syntax.html>

もしフレームワークの仕組みなど「これ気になるなー」と思うことがあればぜひ調べてみましょう。それが新たに勉強する良いきっかけになります。

また、本文中の内容で不明点があればTwitter「<https://twitter.com/FORTEgp05>」までご連絡ください。ベストエフォートでご回答いたします。

この本の使い方

この本は筆者がサーバーサイド Kotlin を学ぶ中で疑問に思ったことや調べたことを技術書の形でアウトプットしたものです。そのため、Kotlinとはなにか？その歴史は？みたいな基礎的なところから、実際に開発環境を用意して Web アプリケーションを開発するまでをまとめた形式になっています。もしあなたが「Kotlinは知ってるから、開発環境から知りたい！」と思っていたら第2章「入門！サーバーサイド Kotlin」のページからご覧ください。または「だいたい分かるけど勉強するのに適したお題がほしい！」と思っていたら第3章「実践」からご覧ください。もちろん Kotlin に初挑戦であれば最初から全部読んでいただいても大丈夫です。

本書はこの本の通りにやれば動くものができる、というところを目指して書かれています。この本を読めば理屈や仕組みがすべて理解できるようには書かれていません。この本を入り口としてその更に奥にあることに興味を持っていただけたら幸いです。

このようにご自身の需要に合わせて本書をご活用いただければ幸いです。必ずしもすべて読む必要はありません。もし必要になったら前のページに戻ればよいのです。ぜひご自身のペースに合わせて、サーバーサイド Kotlin に入門して実践してみてください。

免責事項

本書に記載する内容は筆者の所属する組織の公式見解ではありません。また、本書は可能な限り正確を期すように努めていますが、筆者がその内容を保証するものではありません。そのため、本書の記載内容に基づいた読者の行為、及び読者が被った損害について筆者はなんら責任を負うものではありません。

目次

はじめに	2
サーバーサイド Kotlin を学びたい理由	2
どんな人向けか	2
この本で得られること	3
この本では解説しないこと	3
この本の使い方	4
免責事項	4
第 1 章 Kotlin とは	8
1.1 歴史	9
1.1.1 Kotlin の名前の由来	9
1.2 特徴	10
1.2.1 簡潔	10
1.2.2 安全	12
1.2.3 JVM 上の既存ライブラリとの相互運用可能	13
1.2.4 IDE の対応が手厚い	13
1.3 Kotlin が利用できる開発	13
1.3.1 JVM	13
1.3.2 Android アプリ開発	14
1.3.3 JavaScript (Kotlin/JS)	15
1.3.4 ネイティブ (Kotlin/Native)	15
1.4 その他の特徴	15
1.4.1 関数型プログラミングのエッセンス	15
1.4.2 Google と JetBrains による Kotlin/Everywhere	15
1.4.3 日本の Kotlin ユーザーグループ	16
1.4.4 Kotlin Koans (公式チュートリアル)	17
第 2 章 入門!サーバーサイド Kotlin	18
2.1 開発環境について	18
2.1.1 バージョン一覧 (執筆時)	19
2.1.2 Java のインストール	19

2.1.3	Kotlin コンパイル環境をインストール	27
2.1.4	VS Code のインストール	29
2.1.5	VS Code で Hello World!	33
2.1.6	IntelliJ IDEA のインストール	36
2.1.7	IntelliJ IDEA で Hello World!	44
2.2	Web フレームワーク	45
2.2.1	Spring Framework	45
2.2.2	Ktor	48
2.3	ビルドシステム	48
2.3.1	Ant	49
2.3.2	Maven	49
2.3.3	Gradle	51
2.4	Spring Boot で Hello World!	52
2.4.1	Spring Initializr で雛形を作成する	53
2.4.2	雛形のインポート	54
2.4.3	Hello World! の実装	55
2.4.4	Thymeleaf による HTML テンプレート表示	57
2.5	Spring Boot でデータベース操作	61
2.5.1	Spring Boot でデータベースにご挨拶	61
2.6	Spring Boot でテストを書こう	67
2.6.1	テスト用 DB の準備	68
2.6.2	テストを実行する	73
2.7	依存関係で追加したフレームワークについて	76
2.7.1	Spring Web	76
2.7.2	Spring Data JPA	77
2.7.3	Thymeleaf	77
2.7.4	H2 Database	77
2.8	この章のまとめ	78
第3章	実践	80
3.1	設計	80
3.2	プロジェクトの作成	85
3.3	ディレクトリ構成	85
3.4	投稿機能の実装	87
3.5	更新機能の実装	101
3.6	削除処理を実装する	109
3.7	より良くしていく	114
3.7.1	ユーザー操作に対するメッセージを表示する	115
3.7.2	記事選択時のスムーズスクロール追加	123
3.7.3	バリデーションの実装	124

3.7.4 ソートとページネーションの追加	135
3.8 実践編を振り返って	140
3.9 この章のまとめ	141
付録 A DB を docker で動かしたい！	143
A.1 前提	143
A.2 ディレクトリ構成について	143
A.3 docker-compose.yml の作成	144
A.4 その他のファイルの作成	144
A.5 docker の起動と DB 初期化	145
A.6 アプリケーションに接続設定の追加と動作確認	146
付録 B ローカルに MySQL をインストールする	148
B.1 前提	148
B.2 バージョン	148
B.3 Windows のインストール手順	148
B.4 Mac でのインストール手順	148
B.5 Spring Boot プロジェクトで利用する	150
あとがき	152
電子版について	153
著者紹介	154
文章	154
表紙イラスト担当	154
レビュアー	155
スペシャルサンクス	155

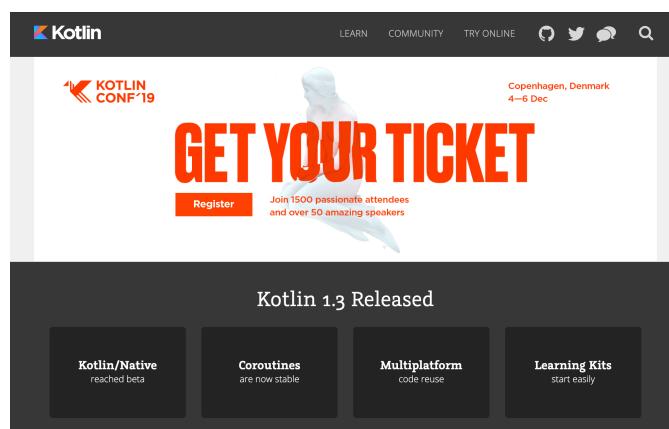
第 1 章

Kotlin とは

まずは Kotlin について解説していきましょう。Kotlin は JetBrains 社が開発した JVM、Android、JavaScript、Native がターゲットの OSS 静的型付けプログラミング言語です。ここで Native とはプラットフォームネイティブのことを指し、例えば iOS や MacOS、Linux、Windows などの各 OS で動作するネイティブアプリの開発向けである、ということです。本書は Web アプリケーションの開発に特化した本ですが、Kotlin がこれだけ様々な分野で活用できるので Kotlin が書けると様々な場面で役に立つのがわかります。

Kotlin は OSS 静的型付けプログラミング言語です。OSS であるため無料で提供され、Apache 2.0 ライセンスの下で開発されておりソースコードは GitHub で入手できます。静的型付け言語であるため、プログラムの実行前に型が解決される言語です。しかし、Java よりもより洗練された静的型付け言語となっています。

また Kotlin はオブジェクト指向のみならず、関数型プログラミングの特徴も含んでいます。いわゆる高階関数、関数型、ラムダといったような機能のことです。



▲図 1.1 2019 年 8 月末時点の Kotlin 公式サイト（英語）

1.1 歴史

Kotlin の開発がスタートしたのは 2010 年です。そして 2011 年 7 月に JetBrains 社のウェブサイトで Kotlin を開発していることを公表しました。^{*1}当時、モダン Java と呼ばれる Java の後継言語は Scala が有名でしたが、JetBrains の開発リーダーである Dmitry Jemеров 氏は Scala の欠点は非常に遅いコンパイルだと言っていました。また公表した記事で Java は下位互換性の問題から修正が不可能であるか非常に難しい問題があると書かれています。さらに他の言語は JetBrains が求める機能セットに対する要件を満たしていない問題があるとも言っています。

そのため JetBrains は、Kotlin を Java より安全にすること、null ポインタ対策や Java より簡潔なこと、型を静的にチェックすることを望んでいました。もう 1 つ、最も成熟した競争相手である Scala よりもシンプルにすることを目指したそうです。その後、2012 年 2 月に JetBrains は Apache 2.0 ライセンスの下で Kotlin をオープンソース化しました。

Kotlin のバージョン 1.0 は 2016 年 2 月 15 日にリリースされました。JetBrains はこのバージョンから長期の後方互換性を約束しています。さらに翌年の Google I/O 2017 で Google は Android の公式言語として Kotlin をサポートすることを表明しました。その後バージョン 1.2 では JVM と JavaScript 間でコードを共有する機能を、バージョン 1.3 ではコルーチンが追加されました。2019 年の現在、多くの Android の開発者が Kotlin を使用して開発しています。実際、Android Kaigi という Android の国内カンファレンスではほぼすべてのセッションのサンプルコードが Kotlin になっています。

このように Android で活用されている事例が目立っていますが、JVM で動作するモダン Java 言語として Web サービスのバックエンド側としての事例も増えています。開発元である JetBrains ではライセンス販売などを管理する JetBrains アカウントでは 100% Kotlin で書かれており、2015 年から運用されていて問題はまったく無いそうです。また Corda というオープンソースの分散元帳プラットフォームでも採用されています。信頼性が重要な Blockchain 技術にも使用されている実績があります。

1.1.1 Kotlin の名前の由来

Kotlin の開発に携わっている Andrey Breslav によると「Java がインドネシアの Java 島にちなんで命名されたのであれば島にちなんで命名することに決めた」そうです。そのため、ロシアのサンクトペテルブルクの近くにあるコトリニ島から名付けられました。ここはフィンランドに近い場所でもあります。

^{*1} <https://www.infoworld.com/article/2622405/jetbrains-readies-jvm-based-language.html>

ちなみに Java は恐らく島ではなくコーヒーにちなんで名付けられたようです。

1.2 特徴

Kotlin の言語としての特徴を説明していきます。わかりやすく説明するため、Java と比較して説明していきます。

1.2.1 簡潔

Kotlin は Java と比べて様々なことがシンプルに書けます。いくつか Kotlin と Java のコードの例を挙げて比べてみましょう。

データクラス

例えば ID や名称などを持つ単純なデータクラスであれば以下のような実装になります。

▼リスト 1.1 Kotlin の場合

```
1: data class HogeData(val id: Integer, val name: String, val memo: String)
```

同じコードを Java で表現すると以下のようになります。

▼リスト 1.2 Java の場合

```
1: public class HogeData {  
2:     public Integer id;  
3:     public String name;  
4:     public String memo;  
5:  
6:     private Integer getId() { return id; }  
7:  
8:     private void setId(Integer id) { this.id = id; }  
9:  
10:    private String getName() { return name; }  
11:  
12:    private void setName(String name) { this.name = name; }  
13:  
14:    private String getMemo() { return memo; }  
15:  
16:    private void setMemo(String memo) { this.memo = memo; }  
17: }
```

Java の場合、Getter と Setter が非常に長いのが分かります。IDE で自動生成できるのでそこまで手間ではないですが、それでも Kotlin では Getter と Setter の生成が不要なのはメリットとなります。こういった手間の軽減が積み重なりが開発の楽さや開発のスピードに繋がってきます。

リスト操作

次にラムダ式を使ってリストをフィルタリングします。次のコードは数値リストから0以上の値を取り出すコードです。

▼リスト 1.3 Kotlin の場合

```
1: val list = listOf(-3, -2, -1, 0, 1, 2, 3)
2:
3: // 1, 2, 3
4: val positiveNumbers = list.filter { it > 0 }
```

同様に Java で表現すると以下のようになります。

▼リスト 1.4 Java の場合

```
1: List<int> list = Arrays.asList(-3, -2, -1, 0, 1, 2, 3);
2:
3: // 1, 2, 3
4: List<int> positiveNumbers = list.stream()
5:                     .filter(o -> o > 0)
6:                     .collect( Collectors.toList() );
```

こちらは Stream API を使用しているため、Stream API 未使用のコードと比べてだいぶ短くなっていますが、それでも Kotlin に比べれば文字数や行数が多くなっています。特に文字数や呼び出すメソッドの数が少ないと IDE で補完する操作が少なくなりますし、単純に手を動かす労力が少なくなるので簡潔であるこのメリットは非常に大きいものがあります。

シングルトン

Kotlin でシングルトンパターンを実装する場合は以下の通りです。

▼リスト 1.5 Kotlin の場合

```
1: object ThisIsASingleton {
2:     val companyName: String = "JetBrains"
3: }
```

Java では以下のようになります。

▼リスト 1.6 Java の場合

```
1: public class ThisIsASingleton {
2:     private static ThisIsASingleton thisIsASingleton;
3:     public String companyName = "JetBrains";
4:
5:     private ThisIsASingleton() {
```

```

6:
7:      }
8:
9:      public static ThisIsASingleton getInstance() {
10:         if (thisIsASingleton == null) {
11:             thisIsASingleton = new ThisIsASingleton();
12:         }
13:
14:         return thisIsASingleton;
15:     }
16: }

```

Kotlin では「object」を指定することで生成される Java コードが、Java のシングルトン生成処理と同じ内容になります。そのため、Kotlin では「object」を指定するだけでシングルトンが実現できるようになっています。

null チェックやインスタンスの宣言、保持などのコードはどこでも同じ実装になるので省略して実装できるのがありがたいです。このあたりが「簡潔」という特徴を感じられる部分でしょう。

1.2.2 安全

Kotlin は Java に比べて安全な実装が可能です。ここで言う「安全」とは不具合の原因になるようなコードを書きづらい、実行時例外を起こしにくいコードを書きやすくなる、システムを運用していて安全なシステムを構築しやすい、ということです。Kotlin が安全である特徴を見ていきましょう。

NullPointerExceptions

Java でシステムを開発して運用した際に悩むことの代表格が、NullPointerExceptions です。これは処理しようとしたオブジェクトが Null だったときに発生する例外ですが、意図しないタイミングでオブジェクトが初期化されたり、異常時に null がリターンされて発生することが多い例外です。

この例外に対する Kotlin の対応を見ていきましょう。

▼リスト 1.7 Kotlin の特徴 安全 Null 非許可

```

1: var output: String          // Null 非許可
2:
3: output = null   // コンパイルエラーとなる

```

Kotlin の場合、同じ String でも実際には 2 種類の型が存在します。それは Null 許可の String と Null 非許可の String の 2 種類です。そのまま String として実装すると Null 非許可になるので、null を代入しようとするとコンパイルエラーとなります。そのため、Null が入りそうな実装をコンパイル段階で気づくことができます。これによって知らずのうちに NullPointerExceptions が起こりそうな実装を防ぐことができます。

さらに Kotlin では Null 許可の型を使用している場合でも安全ではない実装をコンパイル段階で検知してくれます。

▼リスト 1.8 Kotlin の特徴 安全 Null 許可

```
1: val name: String? = null    // Null 許可
2:
3: println(name.length())      // コンパイルエラーとなる
```

Java の場合、こういった実装をしてもコンパイルが通り実行できてしまします。IDE によっては警告を出してくれるので気づく可能性がありますが、コンパイルは通ってしまうので見過ごしてしまうことはあるでしょう。Kotlin の場合はコンパイルが通らないので、こういった危険性がプロダクトコードとして本番環境に入り込んでしまうことはありません。こういった安全性も Kotlin の特徴となっています。

1.2.3 JVM 上の既存ライブラリとの相互運用可能

Kotlin は JVM で動作する既存のライブラリを使用することができます。そのため、新しい言語を導入するといっても既存のライブラリ資産がそのまま流用できるのが非常に楽になります。特に Java 自体が歴史が長い言語なので、フリーのライブラリから、自分たちが作成したライブラリをそのまま使用できるのは大きなメリットになります。

1.2.4 IDE の対応が手厚い

開発元が IDE を開発している JetBrains なので、IntelliJ を使用することで Kotlin の開発をスムーズに始める事ができます。なにより言語の開発元が提供している IDE なので、そのサポート具合はまさに 100% といって良いでしょう。Kotlin は他の IDE でもサポートしていますが、公式サポートという安心感は大きいです。

1.3 Kotlin が利用できる開発

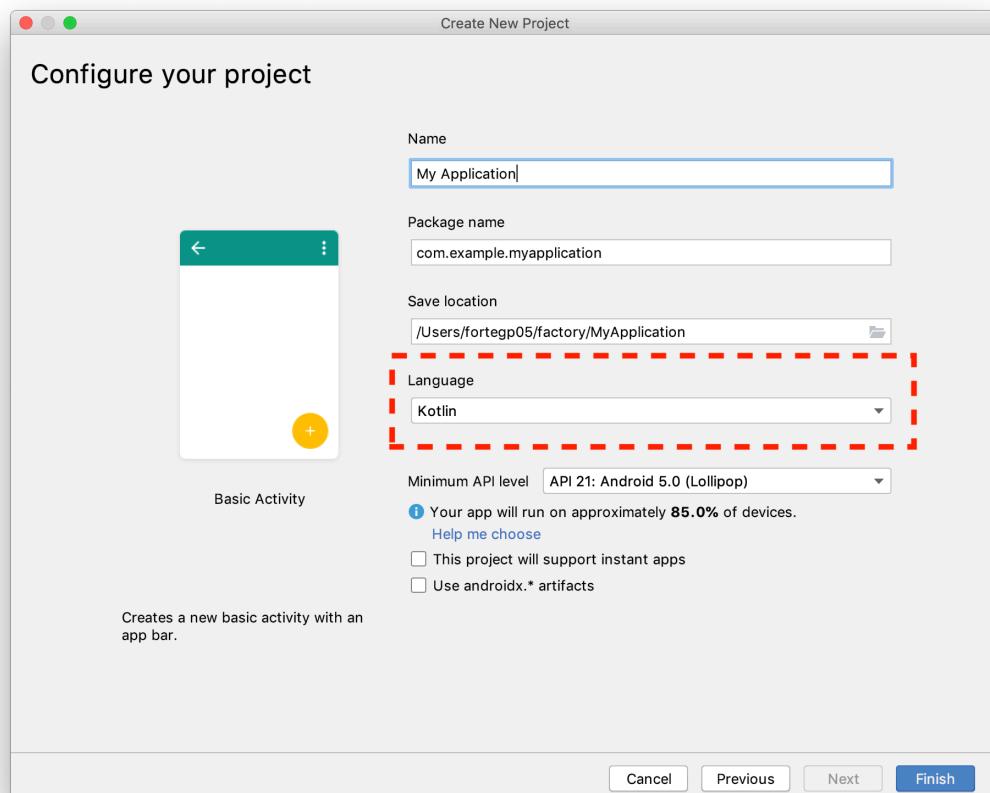
1.3.1 JVM

本書のテーマであるサーバーサイド Kotlin を代表とした JVM 環境での開発です。100%Java 互換である Kotlin は当然 JVM での開発に適しています。そして JVM の主戦場のひとつであるサーバーサイドで Kotlin を使用することができます。Kotlin は既存の Java で書かれた資産を流用することができます。また Java で開発されたプロジェクトの一部分、例えば古い部分は Java のまま、これから開発する新しい機能だけを Kotlin に置き換えることも可能です。Kotlin 自

体、Java からの移行がしやすく古いシステムを良くしていく手助けになります。

1.3.2 Android アプリ開発

Kotlin と聞いて真っ先に思い浮かべる開発シーンは Android アプリの開発だと思います。特に Google が公式に開発言語として公認していることもあり、最も有名だと言っても過言ではないでしょう。さらに Google I/O 2019 では、今後の Android 開発は Kotlin ファーストになるでしょうという発表を行っています。Android アプリで Kotlin が正式にサポートされたのは 2017 年ですが、たった 2 年でプロの Android 開発者の 50% が Kotlin を使用し、Stack Overflow や GitHub でも人気の言語の一つとなっています。筆者の周りの Android エンジニアでもコミュニティでも、Kotlin を使用しているのが当たり前になっていると感じています。Android アプリ開発では今後ますます Kotlin の利用が加速していくことが予想されます。



▲図 1.2 Andoird プロジェクト生成時のデフォルト言語は Kotlin

1.3.3 JavaScript (Kotlin/JS)

意外にも Kotlin で書いたコードを JavaScript に変換することで Kotlin で JavaScript を書く方式がサポートされています。フロントエンドで DOM の処理に使用したり、WebGL などのグラフィック処理に使用できます。さらに Node.js などのサーバーサイド JS との連携にも利用可能です。特に React との連携のために、kotlin-react などのラッパーが提供されています。また初期のレビュー版ではありますが、ビルド構成なしで Kotlin で React アプリを開発可能な create-react-kotlin-app もあります。インストールすればコマンド一発でアプリの構築が可能です。環境構築に煩わされずに開発に集中できます。

1.3.4 ネイティブ (Kotlin/Native)

仮想マシンが望ましくないまたは不可能な組込機器や iOS 向けなどのプラットフォーム用にコンパイルすることができます。これは Kotlin/Native として提供されています。Kotlin/Native は C/C++ 用の静的、動的ライブラリ、Swift や Objective-C 用の Apple フレームワークを作成可能です。また Kotlin/Native はそれらの環境の既存ライブラリを利用可能にするための相互運用性をサポートしているため、Kotlin で開発しつつ既存のライブラリ資産も利用可能です。ネイティブはその環境でサポートされている言語で開発できればベストかもしれませんが、それぞれの環境用にプログラマを雇用するのがベストとは限りません。Kotlin ですべての環境に価値を提供するという方式によって取れる選択肢が広がり、より柔軟な対応が可能となります。

1.4 その他の特徴

1.4.1 関数型プログラミングのエッセンス

Kotlin は関数型プログラミングの機能として高階関数、関数型、ラムダなどをサポートしています。本書の内容的に使用する機会がなかったので、今回は細かく説明しません。しかし、Kotlin で実装されたフレームワーク Ktor や Kotlin/Native では多用されています。理解して使いこなせば非常に便利なのが伺いします。

1.4.2 Google と JetBrains による Kotlin/Everywhere

Kotlin/Everywhere とは開発者、Kotlin ユーザーグループ、GDG (Google Developer Group) などを対象としたコミュニティ主導のイベントです。Kotlin とそのエコシステムについて学びたい方、知識を共有したい方、Kotlin をテーマとしたイベントを主催したい方など、誰でも参加できるようです。開催は世界で

行われます。日本でも5月に大分で開催されました。

このようにAndroidを開発しているGoogleもサポートしているのも特徴のひとつと言えるでしょう。これからますます盛り上がりしていくと思われます。

Kotlin/Everywhereのページ（英語）<https://events.withgoogle.com/kotlin-everywhere/>

1.4.3 日本のKotlinユーザーグループ

Kotlinユーザーグループは日本にもあります。「日本Kotlinユーザーグループ」という名前で活動していて、頭文字を取ってJKUGと呼ばれています（Japan Kotlin User Group）。私は先日、JKUGによって開催されたKotlin Fest 2019に参加してきましたが大変な盛り上がりでした。Kotlinの開発に参加しているエンジニアをお呼びしたり、Android、サーバーサイド、Native、JSと様々な環境でのKotlinに関する発表がありました。

大変な賑わいだったので、世界に負けず劣らず日本でもこれからKotlinが盛り上がっていくことが確信できました！

JKUGのページ<https://kotlin.connpass.com/>



▲図1.3 JKUGのロゴ（かわいい）

1.4.4 Kotlin Koans (公式チュートリアル)

Kotlin Koans は公式から提供されている Kotlin の構文に慣れるためのチュートリアルです（現在は英語のみ）。このチュートリアルはオンラインで受けることができ、ローカルに開発環境を必要としません。チュートリアルは失敗したテストが通るように Kotlin コードを修正する形になっています。また参考として公式ドキュメントへのリンクが付属しているため、勉強しながら進めることができます。

もしローカルの環境で実施したい場合は「EduTools プラグイン」をインストールすることで、IntelliJ IDEA または Android Studio で実施可能です。

英語しかないのでハードルが高いと思われるかもしれません、私がいくつか解いてみた感じではブラウザの翻訳機能とググることで普通に解けました。興味がある方はぜひチャレンジしてみてください。おすすめです。

Kotlin Koans (英語) <https://kotlinlang.org/docs/tutorials/koans.html>