はじめに

筆者は長年、Springに関する開発プロジェクトや教育に携わってきました。その中で痛感した課題が大きく2つあります。

1つは、「Springをきちんと理解せずにコーディングしている開発者が多い」ということです。Springを使用する上で必要となる記述を、おまじないのように行っている開発者をたくさん見てきました。「その記述をすることで何が起きるのか？」がわかっていないため、うまく動かなかったときの対応に苦労したり、意味のない記述をしてしまったり、障害につながるような記述をしてしまう状況になっていました。

もう1つは、「初心者の方がSpringを学習することの難しさ」です。Springを学習しようとすると、Javaを学習したときには出てこなかった新しい用語がたくさん出てきます。難しい用語も多いため、初心者の方が都度理解して覚えることは大変です。また、Springが提供する仕組みには、プログラミングで苦労した経験がないと便利さを実感することが難しいものもあり、プログラミングの初心者にとって、とっつきにくい要因になっています。

これらの課題が解消されることを願って、本書を執筆させていただきました。Springを体系的にきちんと押さえたい方や、Javaの学習がひと段落して、これからSpringを学習する方に特にお勧めの内容になっています。皆様のキャリアに、本書が少しでも貢献できれば幸いです。

2023年6月　土岐 孝平

謝辞

本書を執筆するにあたり、たくさんの方がレビューして下さりました。

池谷 智行さん（日本Springユーザ会）
大野 渉さん（株式会社Comes a Calm）
株式会社ティ・アイ・オーの皆さん
清田 武史さん（株式会社タグバンガーズ）
柴田 洋平さん（日本Springユーザ会/日本Javaユーザーグループ）
進藤 遼さん（Acroquest Technology株式会社、@shindo_ryo）
多田 真敏さん（@suke_masa）
槙 俊明さん（@making）

皆様のレビューのおかげで、本書の内容が劇的に向上しました。貴重なお時間とナレッジをご提供いただき、ありがとうございました。

また、ここでお名前を挙げられなかった方もいらっしゃいます。ご協力いただいた皆様へ、この場を借りて、お礼申し上げます。

前提知識

読者の前提知識として、以下を想定しています。

- Java SE、Servlet、JSP、JDBCを使った簡単なWebアプリケーションを作成またはソースコードを解読できる
- SQLを使って簡単なSELECT・UPDATE・INSERT・DELETE文を記述できる
- 簡単なHTMLを記述できる
- CSSやJavaScriptの役割を知っている（記述できなくてもかまいません）

執筆にあたって使用した主な製品のバージョン

執筆にあたって使用した主な製品のバージョンを**表1**に示します。

▼表1　執筆にあたって使用した主な製品のバージョン

製品	バージョン
Java	17
Spring Boot	3.0.1
Spring Framework	6.0.3
Spring Security	6.0.1
Thymeleaf	3.1.1
JUnit	5.9.1

本書の読み方とアウトライン

本書は、大きく「基本編」と「詳細編」の2つのパートで構成されています。ほとんどの章で簡単なハンズオンを用意しているので、ぜひ手を動かしてプログラムを体験しながら読み進めることをお勧めします。ハンズオン用のソースコードのダウンロードURLはP.11の「付録のソースコードについて」に記載しています。

「第1部　基本編」は、細かな部分に踏み込まずに、Springの機能を全体的につかんでもらう内容です。初心者の方でもつまずきにくい内容になっています。これからSpringを勉強するという方は、基本編を最初からじっくり読み進めることをお勧めします。Springをある程度触ったことがある方も、復習がてらサッと目を通し、気になる章を一読することをお勧めします。

「第2部　詳細編」は、細かい部分も含めて、実際の開発現場で即戦力になるための知識を学習します。特に後半多くのページを割いているテストプログラムの作成方法については、開発現場で重宝する知識になるはずです。詳細編の内容を理解しておけば、開発現場に携わる際の大きな自信につながることでしょう。

なお、本書はアーキテクト（アプリケーション全体の方針を決めたり、共通部品を作成する役割）向けではないため、Springの裏側の細かな挙動や、高度なカスタマイズについての説明は行いません。

また、参考書やマニュアルのように、使い方や設定値の種類を網羅的に説明することも行いません。本書はポイントを押さえることを焦点にしていますので、細かな使い方や設定値の種類は、適宜マニュアル等を参照願います。

【第1部　基本編】

- **第1章　Springの概要**

  Springとは何かから始まり、全体的な機能や特徴を簡単に紹介します。

- **第2章　Webアプリケーションの全体像**

  Springの話の前に、本書が想定するWebアプリケーションの作りについて説明します。後述の章で頻出する用語が説明されています。

- **第3章　DIという考え方**

  Springを使いこなす上で重要な概念となるDIについて、じっくり説明します。DIが何をするためのもので、どのようなメリットがあるのかを説明します。

- **第4章　DIコンテナの概要**

  Springの中心的な機能であるDIコンテナの概要や、基本的な用語について説明します。

- **第5章　ステレオタイプアノテーション**

  Bean定義の方法の1つであるステレオタイプアノテーションの使い方と、併せてDIコンテナの生成方法を説明します。

- **第6章　プロファイルを用いたコンフィグレーションの切り替え**

  実行する環境（テスト環境、本番環境など）に応じてコンフィグレーションを切り替えるのに便利な、プロファイルについて説明します。

- **第7章　JavaConfigと@Beanメソッド**

  Bean定義の方法の1つである@Beanメソッドの使い方と、ステレオタイプアノテーションとの使い分け方法について説明します。

- **第8章　Spring JDBCを使用したデータベースアクセス**

Spring独自のデータベースアクセスの仕組みであるSpring JDBCについて説明します。

- **第9章　宣言的トランザクション**

  データベースのトランザクション制御を自動的に行ってくれる宣言的トランザクションについて説明します。Springの便利さを実感しやすい機能です。

- **第10章　Spring Bootによる生産性の向上**

  SpringイコールSpring Bootと認識している人もいるほど注目度が高いSpring Bootですが、いまいちよくわかっていないという方も多いと思います。本章では、Spring Bootが提供する代表的な機能である「ライブラリの一括取得」「オートコンフィグレーション」「組み込みAPサーバ」について紹介します。

- **第11章　Spring MVC + Thymeleaf**

  Spring MVCという機能を使って、画面まわりのプログラムを作成する方法を説明します。HTMLの生成部分は、Thymeleafというライブラリを使います。

- **第12章　RESTful Webサービスの作成**

  RESTとは何かから始まって、Spring MVCを使った参照系のREST APIの作成方法を説明します。REST自体の特徴についてじっくり説明しているため、なんとなくRESTという言葉を使っている方も、一読することをお勧めします。

- **第13章　更新系のREST APIの作成**

  更新系のREST APIの作成方法を説明します。

- **第14章　Spring Securityを用いた認証と認可**

  Spring Securityを使用した認証・認可、および、CSRFの対応方法を説明します。Spring Securityの裏側の仕組みについても触れているので、Spring Securityがブラックボックスに感じている方にもお勧めの章です。

【第2部　詳細編】

- **第15章　シングルトンとスレッドセーフ**

  DIコンテナが管理するBeanは基本はシングルトンですので、スレッドセーフなプログラムになっていないと致命的なバグにつながる可能性があります。開発現場に携わるにあたって、本章は必須の内容と言えます。

- **第16章　続・Spring JDBC：JOINした結果の取得**

  Spring JDBCを使って、複数のテーブルをJOINした結果をオブジェクトに変換する方法を説明します。

- **第17章　データベースアクセス時の例外**

  Springが提供するデータベースアクセス時の汎用的な例外クラスと、例外のハンドリングのイメージを説明します。

- **第18章　トランザクションの伝搬**

  トランザクションの伝搬とは何かから始まり、トランザクションを伝搬する方法や利用シーンを紹介します。

- **第19章　セッションスコープ**

  セッションスコープを使用し、複数のリクエストをまたがって、サーバ側でデータを共有する方法を説明します。

- **第20章　フラッシュスコープ**

  PRGパターンの説明から始まり、フラッシュスコープを使用して、リダイレクト元と先のリクエスト間で、サーバ側でデータを共有する方法を説明します。

- **第21章　Security Contextの活用**

  Spring Securityは、認証したユーザの情報をSecurity Contextという領域で保持します。本章では、裏側の仕組みに触れつつ、Security Contextの使用方法を説明します。

- **第22章　RESTful Webサービスの呼び出し**

  REST APIを呼び出す際に便利なRestTemplateクラスの使い方を説明します。

- **第23章　プロパティの外部化**

  データベースの接続先などの設定値は、プログラムにハードコーディングするのではなく、実行時に変更できるように外部から読み込んだほうがよいです。本章では、そのような設定値を外部から読み込む方法を説明します。

- **第24章　自動テストとSpringのテストサポートの概要**

  自動テストの代表的な種類や、Webアプリケーションに対してどのような粒度の自動テストのパターンがあるかを説明します。併せて、Springが提供するテストサポート機能の概要や基本的な使い方を説明します。

- **第25章　Repositoryのユニットテスト**

  Repositoryのユニットテストの方法を説明します。データベースにデータを用意したり、更新処理後のデータベースの中身を確認する方法を説明します。

- **第26章　Serviceのユニットテスト**

  Serviceのユニットテストの方法を説明します。主に、Mockのオブジェクトを作成するMockitoというライブラリの使い方を説明します。

- **第27章　Service・Repositoryのインテグレーションテスト**

  ServiceとRepositoryをつなげたインテグレーションテストの方法を説明します。

- **第28章　Controllerのユニットテスト**

  Controllerのユニットテストの方法を説明します。Springが提供するMockMvcという仕組みの使い方や、MockitoとSpringを連動させる方法を説明します。

- **第29章　Controller・Service・Repositoryのインテグレーションテスト**

  Controller、Service、Repositoryをつなげたインテグレーションテストの方法を説明します。セッションスコープやフラッシュスコープの扱いについても紹介します。

- **第30章　RESTful Webサービスのテスト**

  作成したREST APIをテストする方法を説明します。JSONのレスポンスの確認方法や、組み込みAPサーバを使用したテストなどを紹介します。

- **第31章　Spring Securityのテストサポート**

  Spring Securityのテストサポート機能を利用して、テスト実行時のユーザの設定や、CSRFトークンの送信方法を説明します。

- **第32章　Selenideを用いたE2Eテスト**

  Selenideというライブラリを使用し、自動的にブラウザを操作しながらE2Eテストを行う方法を説明します。

【第3部　Appendix】

初心者の方がつまずきやすい用語を各節で説明します。また、末尾の節で、ソースコードの題材となっているアプリケーションの概要を説明します。

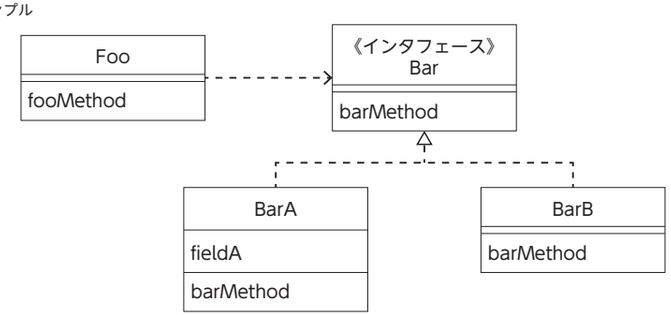
本書で扱っているUMLの表記について

本書で扱っているUMLの表記の見方を説明します。

クラス図

クラス図は、クラスやインタフェースの定義と、お互いの関係を示した図です。**図1**にサンプルを示します。

▼図1　クラス図のサンプル



クラスやインタフェースは四角で表します。四角の中は3つの枠で構成されており、一番上の枠にクラス名やインタフェース名を記載します。インタフェースについては、インタフェース名の上に「《インタフェース》」と記載します。**図1**には、3つのクラスと1つのインタフェースが存在することがわかります。

四角の中の真ん中の枠には、クラスが持つフィールド[注1]を記載します。たとえば、BarAクラスには、fieldAフィールドが存在します。フィールドは複数記載することも可能です。なお、本書ではフィールドの型や修飾子などは省略します。また、フィールドが存在しなかったり、フィールドを記載することが重要でない場合は、空で表記します。

四角の中の一番下の枠には、クラスやインタフェースが持つメソッドを記載します。たとえば、Fooクラスには、fooMethodメソッドが存在します。メソッドは複数記載することも可能です。なお、本書ではメソッドの引数や戻り値などは省略します。また、メソッドが存在しなかったり、メソッドを記載することが重要でない場合は、空で表記します。

また、フィールドとメソッドの記載が重要でない場合は、以下のように、クラス名やインタフェース名だけの四角で記載します。

注1　クラスの中、メソッドの外で宣言した変数。クラス内のすべてのメソッドの中から参照できます。

四角同士をつないでいる以下の矢印は、片方がもう片方を使う（依存とも呼びます）ことを意味します。

「使う」の種類にはさまざまなものがありますが、**図1**のように、クラスからインタフェースに対して引かれている場合は、本書では、FooクラスがBarインタフェースのメソッドを呼び出すことを意味します。

四角同士をつないでいる以下の点線の白抜きの矢印は、矢印の元のクラスが、矢印の先のインタフェースを実装していることを意味します。

図1では、Barインタフェースを実装するクラスとして、BarAクラスとBarBクラスが存在することがわかります。

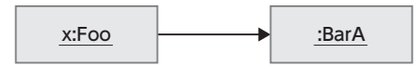
また、**図1**には記載がありませんが、継承を表す場合は、以下のような実線の白抜きの矢印になります。

矢印の元のクラスが、矢印の先のクラスを継承していることを意味します。

オブジェクトの表記

UMLには、オブジェクトを表す表記方法があります。**図2**にサンプルを示します。

▼図2　オブジェクトの表記のサンプル



オブジェクトを四角で表します。クラスの四角と区別しやすいように、本書では薄いグレーの色がついています。四角の中は、「:」で区切った文字列を記載し、アンダーバーを引きます。「:」の右側にはオブジェクトの型を記載します。オブジェクトの型は、オブジェクトの元となる具象クラス[注2]の場合もありますし、実装しているインタフェースの場合もあります。また、本書では、ControllerやServiceといったオブジェクトの役割を記載することもあります。

「:」の左側には、オブジェクトの任意の名前を記載しますが、本書では省略することが多いです。

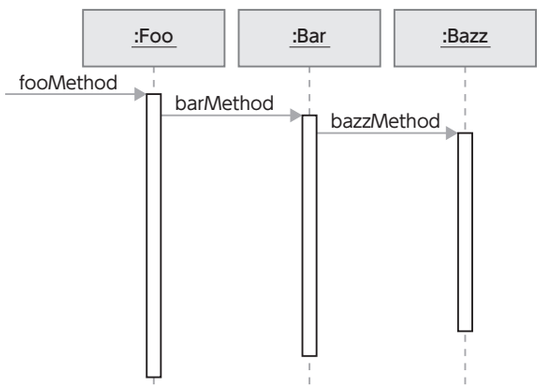
オブジェクト間を結ぶ矢印は、矢印の元のオブジェクトが、矢印の先のオブジェクトを参照することを示します。一般的に、オブジェクト間を結ぶときは、矢印ではなく単なる線で結ぶことが多いですが、本書では便宜的に矢印を使用します。**図2**では、Fooクラスのオブジェクトが、BarAクラスのオブジェクトを参照していることがわかります。

注2　コンストラクタを呼び出してオブジェクトを作ることができるクラス。オブジェクトには、元となる具象クラスが必ず存在します。

シーケンス図

シーケンス図は、メソッドの呼び出し順に沿って処理の流れを示した図です。**図3**にサンプルを示します。

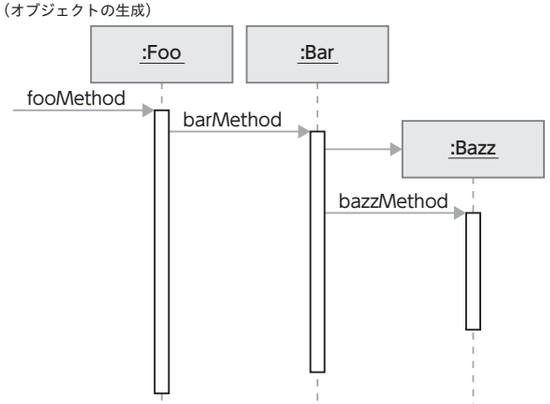
▼図3　シーケンス図のサンプル



処理の中で登場するオブジェクトを四角で表します。四角の表記は前述したオブジェクトの表記です。**図3**では、Fooクラスのオブジェクトの fooMethod メソッドが呼び出されたところから図が始まっています。メソッドの処理が実行されている間は、縦の線が白抜きの太い線になります。fooMethod メソッドの中で、Barオブジェクト（具象クラスは問わない）の barMethod メソッドが呼び出され、barMethod メソッドの中で、Bazzクラス（**図1**には記載していない新たなクラス）のオブジェクトの bazzMethod メソッドが呼び出されていることがわかります。

また、メソッドの中でオブジェクトの生成を行う場合は、**図4**のように記載します。

▼図4　シーケンス図のサンプル（オブジェクトの生成）



生成されるオブジェクトの四角を、生成されるタイミングの場所に下降させ、生成を指示する場所から四角に向けて矢印を引きます。**図4**では、barMethod メソッドの中で、Bazzクラスのオブジェクトを生成していることがわかります。

付録のソースコードについて

ほとんどの章では、章の内容を体験するための簡単なハンズオンが用意されています。ハンズオンが用意されている章の章末には、その旨が記載されています。ハンズオンは、付録のソースコードを使用して行うことができます。付録のソースコードは、以下のURLからダウンロードできます。

https://gihyo.jp/book/2023/978-4-297-13613-0/support

ソースコードを実行するには、Java 17以上が必要です。また、ソースコードは、Maven[注3]を使用する形式になっています。ダウンロードしたZIPファイルを解凍すると「spring-book-src-master」フォルダが展開されますので、「spring-book-src-master」直下の pom.xml ファイルを、Mavenと連携可能なIDE（統合開発環境）で取り込んでください。代表的なIDEであるEclipseおよびIntelliJ IDEAでの取り込み操作をこの後に記載しますので、必要に応じて参照ください。

ソースコードを取り込むと、たくさんのプロジェクトが表示されます。プロジェクトは、対応する章ごとに分かれています。プロジェクト名の先頭に番号が付いていますが、章番号を表すものではありません。プロジェクト名の末尾に「-answer」が付いているものは解答例です。「-answer」が付いていないほうがハンズオン用のプロジェクトです。ハンズオン用のプロジェクトの直下には、ハンズオンの手順が書かれた「Instruction.adoc」というファイルが存在します。

Instruction.adocはAsciiDoc形式で記述されているため、IDEのプラグイン[注4]や、AsciiDocのビューアー[注5]をインストールして表示すると見やすくなります。

ハンズオン用のソースコードは、大きく2種類の題材が使用されています。1つは「研修予約アプリケーション」です。プロジェクト名に「training」が含まれているものは「研修予約アプリケーション」を題材に使用しています。もう1つは「商品注文アプリケーション」です。プロジェクト名に「shopping」が含まれているものは「商品注文アプリケーション」を題材に使用しています。それぞれのアプリケーションの概要を巻末のAppendixの「A.25　研修予約アプリケーションについて」と「A.26　商品注文アプリケーションについて」に記述しています。必要に応じて参照ください。

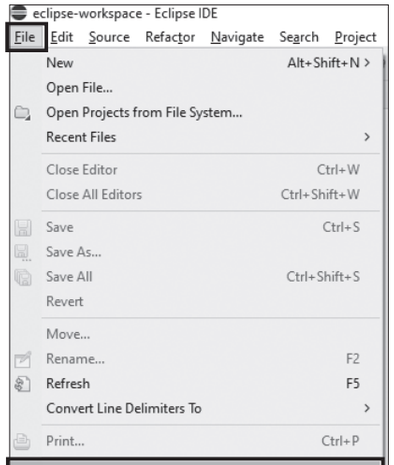
「0001-training-common」と「0002-shopping-common」プロジェクトは、他のプロジェクトから参照される共通のプログラムが格納されています。Entityクラスや例外クラス、データベースのデータ登録用のSQLファイル（schema.sql、data.sql）が存在します。

注3　巻末のAppendixの「A.12　Mavenとは？」を参照。
注4　Eclipseのプラグインの「Asciidoctor Editor」や、IntelliJ IDEAのプラグインの「IntelliJ AsciiDoc Plugin」など。
注5　Chromeの拡張の「Asciidoctor.js Live Preview」など。

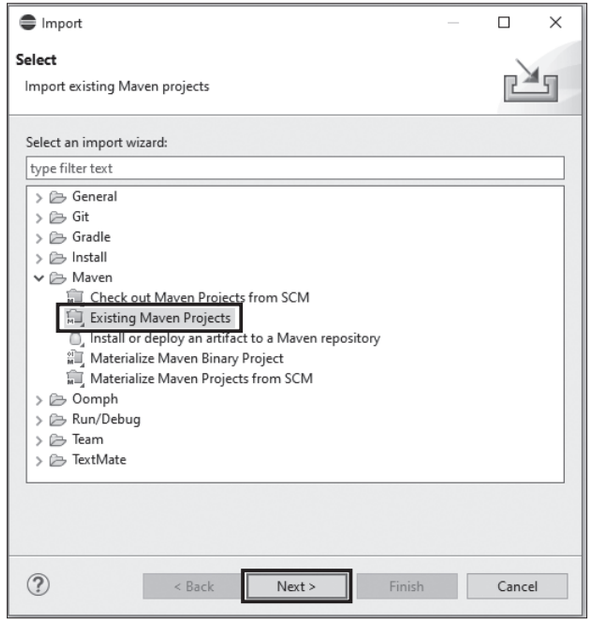
Eclipseでのソースコードの取り込み操作

本節で説明する操作、画面はWindows版の「Eclipse IDE for Java Developers」のバージョン「2022-12」を使用しています。お使いのOSやバージョンによっては、操作や画面が異なる場合があります。

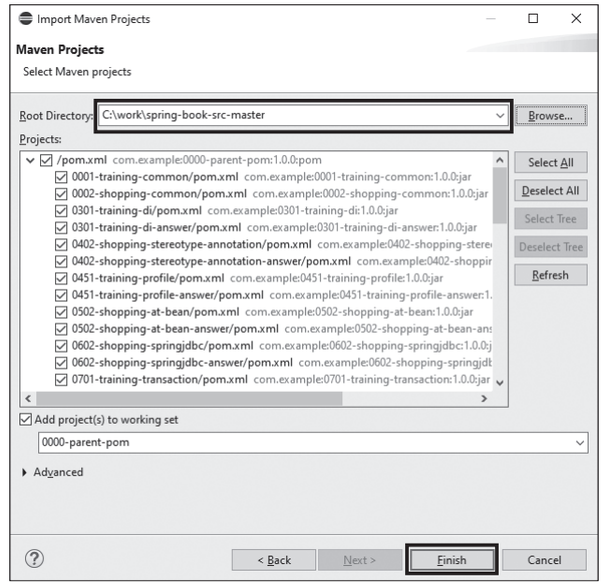
1. [File] → [Import] を選択



2. [Existing Maven Projects] → [Next] を押下

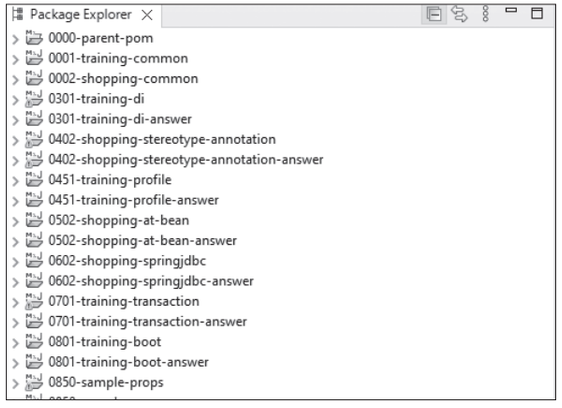


3. ソースコードのフォルダを選択して [Finish] を押下
   （ソースコードのフォルダを選択すると取り込まれるプロジェクトの一覧が [Projects:] 欄に表示されます）



4. 取り込まれたことを確認

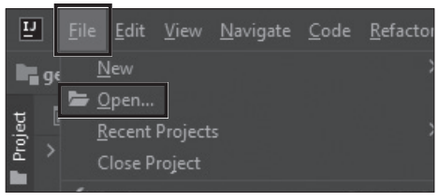
   [Package Explorer] に多数のプロジェクトが表示されます。取り込んだ直後は、プロジェクトに赤いマークが表示されますが、ビルドが完了すると赤いマークが消えます。また、初回取り込み時は、Mavenがライブラリをダウンロードするため、時間がかかる場合があります。
   社内ネットワーク等で、インターネット接続時にプロキシサーバを使用している場合、Mavenによるライブラリのダウンロードが失敗して赤いマークが表示されたままになることがあります。その際は、Mavenにプロキシサーバの設定を行ってから、再度取り込みの操作を行ってください。



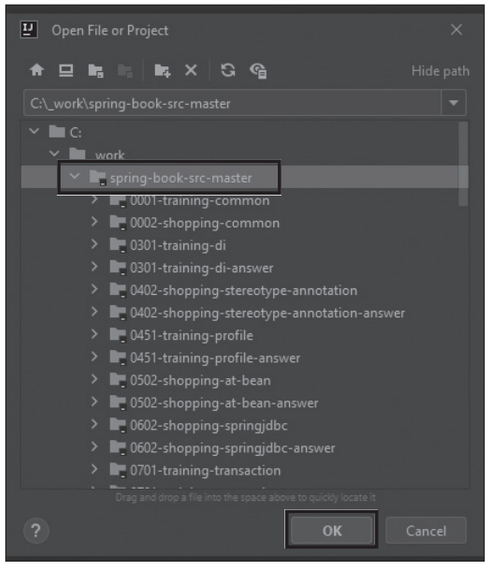
IntelliJ IDEAでのソースコードの取り込み操作

本節で説明する操作、画面はWindows版の「IntelliJ IDEA Community Edition」のバージョン「2022.3.1」を使用しています。お使いのOSやバージョンによっては、操作や画面が異なる場合があります。

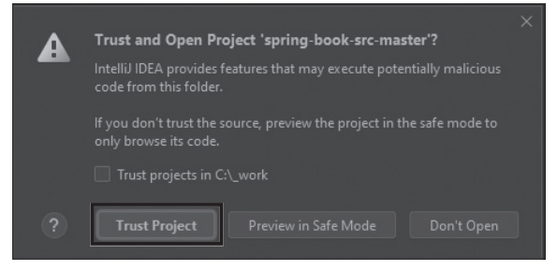
1. [File] → [Open] を選択



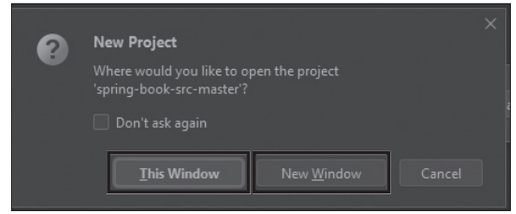
2. ソースコードのフォルダを選択して [OK] を押下



3. [Trust Project] を押下
(取り込むファイルを信頼することの確認です)



4. [This Window] もしくは [New Window] を押下
(表示中のウィンドウを使用する場合は [This Window]、新しくウィンドウを開く場合は [New Window] を押下します)



5. 取り込まれたことを確認
[Project] に多数のプロジェクトが表示されます。初回取り込み時は、Mavenがライブラリをダウンロードするため、時間がかかる場合があります。
社内ネットワーク等で、インターネット接続時にプロキシサーバを使用している場合、Mavenによるライブラリのダウンロードが失敗してプロジェクトがうまく表示されないことがあります。その際は、Mavenにプロキシサーバの設定を行ってから、再度取り込みの操作を行ってください。