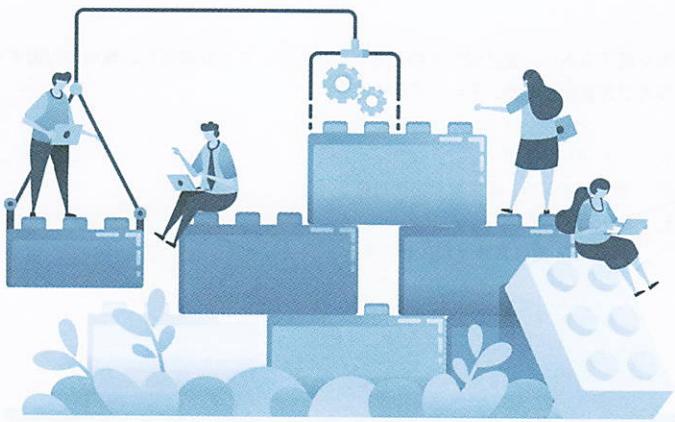
ドメイン駆動設計入門

ボトムアップでわかる! ドメイン駆動設計の基本

成瀬 允宣 著



SE
SHOEISHA

本書内容に関するお問い合わせについて

このたびは翔泳社の書籍をお買い上げいただき、誠にありがとうございます。
弊社では、読者の皆様からのお問い合わせに適切に対応させていただくため、以下のガイドラインへのご協力をお願い致しております。
下記項目をお読みいただき、手順に従ってお問い合わせください。

ご質問される前に

弊社Webサイトの「正誤表」をご参照ください。これまでに判明した正誤や追加情報を掲載しています。

正誤表　https://www.shoeisha.co.jp/book/errata/

ご質問方法

弊社 Web サイトの「刊行物Q&A」をご利用ください。

刊行物 Q&A　https://www.shoeisha.co.jp/book/qa/

インターネットをご利用でない場合は、FAXまたは郵便にて、下記翔泳社愛読者サービスセンターまでお問い合わせください。電話でのご質問は、お受けしておりません。

回答について

回答は、ご質問いただいた手段によってご返事申し上げます。ご質問の内容によっては、回答に数日ないしはそれ以上の期間を要する場合があります。

ご質問に際してのご注意

本書の対象を越えるもの、記述個所を特定されないもの、また読者固有の環境に起因するご質問等にはお答えできませんので、予めご了承ください。

郵便物送付先およびFAX番号

送付先住所　〒160-0006　東京都新宿区舟町５
FAX 番号　03-5362-3818
宛先　㈱翔泳社 愛読者サービスセンター

※本書に記載されたURL等は予告なく変更される場合があります。
※本書の対象に関する詳細はvページをご参照ください。
※本書の出版にあたっては正確な記述につとめましたが、著者や出版社などのいずれも、本書の内容に対してなんらかの保証をするものではなく、内容やサンプルに基づくいかなる運用結果に関してもいっさいの責任を負いません。
※本書に掲載されているサンプルプログラムやスクリプト、および実行結果を記した画面イメージなどは、特定の設定に基づいた環境にて再現される一例です。
※本書に記載されている会社名、製品名はそれぞれ各社の商標および登録商標です。
※本書の内容は、2020年１月執筆時点のものです。

Preface はじめに

開発者にとってドメイン駆動設計を学ぶことはその後のエンジニアリングに対するスタンスを変えうるほどの大きな学びです。

ドメイン駆動設計のコンセプトは単純です。ビジネスの問題を解決するためにビジネスの理解を進め、ビジネスの表現をする。ビジネスとコードを結びつけて継続的かつ反復的な改良を施せるように枠組みを作ることにより、ソフトウェアをより役立つものにしようというものです。いわば、ソフトウェアシステムにおいて当たり前のことを行おうとしています。

しかし、いざドメイン駆動設計を学ぼうとした初学者の多くは、その難解さに打ちのめされてしまいます。なぜ打ちのめされてしまうのでしょうか。その原因は多種多様な専門用語の難しさにあるのではないかと考えます。

ドメイン駆動設計の解説はユビキタス言語という用語の解説から始まります。ページを少しめくってみればドメインエキスパートという言葉を筆頭に、境界付けられたコンテキスト、エンティティ、値オブジェクトといった多くの専門用語が現れます。これらの用語は当然のことながら、初学者にとっては初めて見聞きするものです。専門用語の中には多くの予備知識を前提とするものが存在し、ほとんど恐怖と似たような感情を思い起こさせ、理解を妨げてしまっています

人が恐怖する原因をご存知でしょうか。

人は知らないことに恐怖します。たとえ風に揺れるカーテンも、正体を知らなければ幽霊になり替わります。まさにドメイン駆動設計の専門用語は幽霊で、恐怖を与えるものに違いありません。恐怖は人を鈍らせます。恐怖は人を遠ざけます。

幽霊が怖い理由はもちろん、その正体がわからないからです。怖さを和らげるためにすべきことは単純です。幽霊の正体を確かめて、それがカーテンであることを知ってしまえばよいのです。

ドメイン駆動設計を学習するうえで登場する用語は大別すると2種類に分かれます。ソフトウェアにとって重要な概念を抽出するためのモデリングと概念を実装に落とし込むためのパターンです。モデリングに関しては言葉による説明を理解し実践して学ぶほかありませんが、パターンに関しては詳細なサンプルをもって説明できます。開発者にとって理解しやすいのはもちろん後者です。

そこで本書はモデリングについてはいったん棚上げにして、具体的なコードをベースにパターンを集中的に解説し、全体を通して最終的なコードを提示します。どこかへ行こうとするとき、目的地がわからないということはとても不安を覚えます。本書が提示するコードは明確な形をもった目的地としてあなたの道しるべになるものです。

本書はパターンを解説することに重きを置いています。しかし、これはパターンを実践することがすなわちドメイン駆動設計の実践であり、そのすべてであるという主張をしているわけではありません。本書が目的としているのは、ドメイン駆動設計という巨大な試練に立ち向かうための準備です。ある知識を理解しようとするとき、また別の知識が手掛かりとなることは多く、なるほど知識は連鎖するものです。ドメイン駆動設計という大きなテーマと対峙する準備として必要なことは、あらかじめ理解の領地を広げておくことです。

本書から得られるパターンに関しての理解はモデリングを含めたドメイン駆動設計の全貌を理解するのに役に立つものです。これからドメイン駆動設計の世界を旅するあなたの武器となるでしょう。

よりよいソフトウェアを開発するために、開発者として一段と高みへ上るために、あなたがドメイン駆動設計を学ぶことを選んだのは間違いではありません。

2020年1月吉日
成瀬允宣

DDD Acknowledgments 謝辞

本書を執筆するにあたり、多くの方々に助けをいただき、支えていただきました。

末安 章花氏の熱烈な後押しがなければ本書は始まりませんでした。
大平 道介氏が背中を押さなければここまではたどり着けませんでした。
林 宏勝氏の丁寧な査読は多くの言葉に磨きをかけました。
森 怜峰氏の鋭いコメントはその度に大幅な加筆につながりました。
松岡 幸一郎氏の芯の通った考えは筆者の考察をより強固にしました。
加藤 潤一氏の忌憚ない意見は考察の機会を与えてくれました。
増田 亨氏の言葉は筆者に勇気を与えてくれました。

貴重な時間を惜しみなく費やし、支えてくれた彼等に感謝を。
そして私を励まし続けてくれた妻に感謝を。

DDD Introduction 本書の対象読者

本書は主にオブジェクト指向プログラミング言語を用いているソフトウェア開発者向けに書かれていますが、これはドメイン駆動設計を実践するにはオブジェクト指向プログラミング言語が必須であるということを意味しません。オブジェクト指向以外のパラダイムをもつ言語であっても、本書のエッセンスをソフトウェア開発に役立てることは可能です。

本書を読み進めるには一般的なオブジェクト指向プログラミングの基礎知識が必要です。具体的に求められるレベルとしては、クラスとインスタンスについての理解が必要です。他にはインターフェース（抽象型）機能をふんだんに利用しているので、インターフェースを用いたポリモーフィズムの処理の流れは理解しておくと読み進める助けになります。

本書で取り扱うサンプルコードのプログラミング言語は C# を採用しています。C# は一般的なオブジェクト指向プログラミング言語としての機能を網羅しており、また記述方法にあまり癖がない言語と考えているからです。たとえ C# についてまったく知識がなかったとしても、オブジェクト指向プログラミング言語を利用している開発者であれば読み換えることはそう難しいことでないでしょう。

いずれにせよ深い知識は必要なく、コードを書くことを生業とする読者であれば読み進めることは可能でしょう。

取り扱うC#特有の構文

本書ではいくつかC#特有の構文が利用されています。いずれも冗長さを排除するための記述ですが、事前にここで解説をしておきます。

readonly

フィールド（インスタンス変数）に再代入される可能性があるか否かというのは貴重な情報です。再代入が不可能であれば、その値が変化したときのことを考慮せずに済みます。

C#ではインスタンス変数にreadonly修飾子を付けることで再代入を不可能にできます。

リスト1：readonly修飾子を用いたフィールドの定義

```
class MyClass
{
  private readonly string invariant; // 再代入不可能
  private string variant; // 再代入可能
  (…略…)
}
```

なお、readonlyで修飾されたインスタンス変数はコンストラクタ以外で代入を行うことができません。

リスト2：readonlyへの代入

```
class MyClass
{
  private readonly string value;

  public MyClass(string value) {
    this.value = value; // OK
  }

  public void ChangeValue(string value) {
    this.value = value; // コンパイルエラー
  }
}
```

プロパティ

プロパティはフィールドに外部からアクセスするための機能です。**リスト 3**のように定義すると内部でフィールドが自動実装され、アクセスできるようになります。

リスト 3：プロパティによる定義

```
class MyClass
{
  public string Property { get; private set;}
}
```

プロパティは値を取得するためのゲッターと値を設定するためのセッターを定義できます。ゲッターとセッターはそれぞれ異なるアクセス修飾子を設定できます。ゲッターのみを定義した場合は読み取り専用のプロパティになり、コンストラクタでのみ代入が可能になります。

またゲッターやセッターはメソッドのようにふるまいをもたせることも可能です。

リスト 4：ふるまいをもったプロパティ

```
class MyClass
{
  private string property;

  public string Property
  {
    get { return property; }
    set { property = value; }
  }
}
```

using 句

ファイルなどのリソースを保持するオブジェクトは、オブジェクトが不要になったときにリソースを解放するために、終了処理を実行する必要があります。C# ではそういった終了処理を確実に呼び出すために using 句を利用します。

using 句はオブジェクトのスコープを明示し、スコープから抜ける際にそのオブジェクトの終了処理を呼び出します。

リスト 5：using 句を用いたリソースの取得

```
using(var connection = new SqlConnection(connectionString))
{
    (…略…)
}
```

using 句に指定されるオブジェクト（**リスト 5** でいうところのSqlConnection）はIDisposable インターフェースを実装する必要があります。スコープを抜ける際にIDisposable インターフェースのDispose メソッドが呼び出されます。

Introduction 本書のサンプルの動作環境とサンプルプログラムについて

本書の第14章のサンプルは**表1**の環境で、問題なく動作することを確認しています。

項目	内容
OS	Windows 10 Home
IDE	Visual Studio 16.4

表1：実行環境

付属データのご案内

付属データは、以下のサイトからダウンロードできます。

・付属データのダウンロードサイト

URL https://github.com/nrslib/itddd

注意

付属データに関する権利は著者および株式会社翔泳社が所有しています。許可なく配布したり、Webサイトに転載したりすることはできません。

付属データの提供は予告なく終了することがあります。予めご了承ください。

会員特典データのご案内

会員特典データは、以下のサイトからダウンロードして入手いただけます。

・会員特典データのダウンロードサイト

URL https:///www.shoeisha.co.jp/book/present/9784798150727

注意

会員特典データをダウンロードするには、SHOEISHA iD（翔泳社が運営する無料の会員制度）への会員登録が必要です。詳しくは、Webサイトをご覧ください。

会員特典データに関する権利は著者および株式会社翔泳社が所有しています。許可なく配布したり、Webサイトに転載したりすることはできません。

会員特典データの提供は予告なく終了することがあります。予めご了承ください。

免責事項

付属データおよび会員特典データの記載内容は、2020年1月現在の法令等に基づいています。

付属データおよび会員特典データに記載されたURL等は予告なく変更される場合があります。

付属データおよび会員特典データの提供にあたっては正確な記述につとめましたが、著者や出版社などのいずれも、その内容に対して何らかの保証をするものではなく、内容やサンプルに基づくいかなる運用結果に関してもいっさいの責任を負いません。

付属データおよび会員特典データに記載されている会社名、製品名はそれぞれ各社の商標および登録商標です。

著作権等について

2020年1月

株式会社翔泳社　編集部

Contents 目次