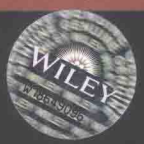
Patterns, Principles, and Practices of Domain-Driven Design

领域驱动设计模式、原理与实践

[美] Scott Millett 著
Nick Tune
蒲 成 译



清华大学出版社

领域驱动设计模式、原理与实践

[美] Scott Millett
Nick Tune 著
蒲成 译

清华大学出版社
北京

Scott Millett，Nick Tune
Patterns, Principles, and Practices of Domain-Driven Design
EISBN：978-1-118-71470-6

图书在版编目(CIP)数据

领域驱动设计模式、原理与实践 /(美) 米利特(Millett,S.) 等著；蒲成 译. —北京：清华大学出版社，2016
书名原文：Patterns, Principles, and Practices of Domain-Driven Design
ISBN 978-7-302-42890-9

Ⅰ. ①领… Ⅱ. ①米… ②蒲… Ⅲ. ①软件设计 Ⅳ. ①TP311.5

中国版本图书馆 CIP 数据核字(2016)第 030112 号

责任编辑：王　军　于　平
装帧设计：牛静敏
责任校对：成凤进
责任印制：杨　艳

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：清华大学印刷厂
经　　销：全国新华书店
开　　本：185mm×260mm　　印　　张：47　　字　　数：1144 千字
版　　次：2016 年 2 月第 1 版　　印　　次：2016 年 2 月第 1 次印刷
印　　数：1～3000
定　　价：99.80 元

产品编号：062761-01

译 者 序

如今，越来越多的公司和开发人员都愈发重视设计模式的应用，那么，到底我们为什么要用设计模式呢？这么多设计模式为什么要这么设计呢？根本原因就是为了代码复用，增加可维护性。因此，各种设计模式都有大量的实践和指导原则，遵循这些实践并且实现这些原则，就能达到代码复用、增加可维护性的目的。

领域驱动设计就是这样一种软件开发设计模式，其目的是让软件系统在实现时准确地基于对真实业务过程的建模并随时根据真实业务过程的调整而调整。领域模型使开发人员可以表达丰富的软件功能需求，由此实现的软件可以满足用户真正的需要，因此被公认为是软件设计的关键所在，其重要性显而易见。

本书作为一本关于领域驱动设计(DDD)的实践指导类书籍，能让你全面理解如何在你自己的项目上应用领域驱动设计模式和实践。本书将用大量贴合实际的示例并结合领域驱动设计的模式、原则和实践来介绍其思想体系。重点在于让读者理解其价值和实现的机制及原理，而不是一味泛泛而谈各种枯燥乏味的理论知识点。相信在阅读完本书后，你将对 DDD 有一个全面了解，能帮助你构架可维护和可扩展的应用程序，做好为用于大型复杂问题域的复杂软件进行构架和维护管理的准备。

本书通篇都在强调一种观点，这当然也是领域驱动设计的核心基础，即领域驱动设计重视的是专注于业务问题域的需要：其专业术语、开发该软件的关键原因，以及对于业务来说什么才是成功。开发人员需要与领域专家一起协作以便达成对领域概念的共识，这一点对于应用领域驱动设计来说是至关重要的前提。因而，本书基于这一观点结合领域驱动设计的各方面实践以及各种可用工具，在实例的引导下呈现了从头至尾完整构建一个低耦合、可扩展、可维护系统的过程。使得广大读者在阅读完本书内容后能够从此迈入领域驱动设计的大门。

本书内容详实，翻译工作量巨大，在此要特别感谢清华大学出版社的编辑们，在本书翻译过程中他们为译者提供的巨大帮助，没有其热情付出，本书将难以成功付梓。

本书全部章节由蒲成翻译，参与本次翻译的还有王佳、何东武、杨达辉、申成龙、李文强、杨帆、王滨、李鹏、负书谦、赵栋、林超、刘洋洋、潘丽臣、郑斌、杨晔。

由于译者水平有限，难免会出现一些错误或翻译不准确的地方，如果有读者能够指出并勘正，译者将不胜感激。

译者

作 者 简 介

Scott Millett 是 Iglu.com 的 IT 总监，并且从 1.0 版本开始就一直使用.NET 工作。他在 2010 年和 2011 年被授予 ASP.NET MVP。他还是 *Professional ASP.NET Design Patterns* 和 *Professional Enterprise .NET* 两本书的作者。如果你希望与 Scott 联系并交流与 DDD 有关的内容或者在 Iglu 工作，欢迎发送电子邮件到其邮箱 scott@elbandit.co.uk，或者通过 https://www.linkedin.com/in/scottmillett 添加好友。

合著者简介

Nick Tune 热衷于解决业务问题、构建有远大目标的产品以及不断学习。成为一名软件开发人员的确就是他的梦想。到目前为止他职业生涯最精彩的时期就是在 7digital 工作的时候，在那里他是那些自我组织、专注于业务的团队的一份子，他们每天最多会进行 25 次生产环境的部署。他未来的志向是与富有激情的人一起致力于令人激动的新产品的开发，并且逐渐变成一个能更完美解决问题的人。

你可以在其网站(www.ntcoding.co.uk)和推特(@ntcoding)上了解与 Nick 有关的更多信息及其关于软件开发、软件交付和他偏爱的技术的观点。

技术编辑简介

Antony Denyer 是一位开发人员、咨询顾问和培训师，他从 2004 年起就开始专业开发软件。他已经开发了有效使用 DDD 概念和实践的各种项目。特别是最近，他在其大多数项目中都倡导使用 CQRS 和 REST。可以通过向 antonydenyer.co.uk 发电子邮件及其推特号@tonydenyer 来联系他。

致　谢

首先我要对 Nick Tune 同意帮助我完成本书的编著表示最真挚的感谢，他对许多章节都做出了巨大的贡献。我还要感谢 Rosemarie Graham、Jim Minatel 以及 Wrox 中所有帮助过本书编著出版的人。同样要感谢 Antony Denyer，他作为技术编辑出色地完成了任务。最后，非常感谢 Isabel Mack 指出的语法错误以及对本书 Leanpub 草稿提供的早期反馈。

前　言

编写软件很容易——至少开发全新的软件是这样的。当你要修改由其他开发人员编写的代码或者你六个月之前编写的代码时，最乐观的情形可能是有点无聊，但最糟的情况可能就是一场噩梦了。该软件可以运行，但你无法确切肯定是如何运行的。它包含所有正确的框架和模式，并且是使用敏捷方法创建的，但是将新的特性引入到代码库本来应该更容易才对。即使业务专家也无济于事，因为代码完全不类似于他们所使用的语言。开发这样的系统会变成烦人的杂事，让开发人员懊恼并且没有任何编码乐趣。

领域驱动设计(DDD)是让你的代码与问题域保持一致的处理过程。随着你的产品的演化，添加新的特性会变得像之前在全新开发阶段一样容易。尽管 DDD 理解软件模式、原则、方法论以及框架的需求，但它也看重开发人员和领域专家协同工作以便均等地理解领域概念、策略和逻辑。有了对问题域的更好理解以及与业务的协同合作，开发人员更可能构建出更容易阅读和更易于适应未来功能扩展的软件。

遵循 DDD 思想体系将让开发人员获得他们需要的知识和技能，以便高效地处理大型或复杂的业务系统。不需要担心未来的功能扩展，开发人员不再会为遗留应用程序而头疼。实际上，遗留这个词在开发人员头脑中会意味着：一个持续为业务带来价值的系统。

本书及技术概览

本书能让你全面理解如何在你自己的项目上应用 DDD 模式和实践，但在开始阅读详细内容之前，最好大致了解一下这一思想体系，这样你就能对 DDD 到底是什么有一个直观的感受。

问题空间

在你可以开发一个解决方案之前，你必须理解其问题。DDD 强调专注于业务问题域的需要：其专业术语、为何开发该软件的关键原因，以及对于业务来说什么才是成功。业务团队需要像技术专家一样看重领域知识，这一点对于获得问题域的更深刻见解以及将大的领域分解成较小的子域至关重要。

图 1 显示了 DDD 问题空间的一个高层次概览，本书的第 I 部分将会介绍它。

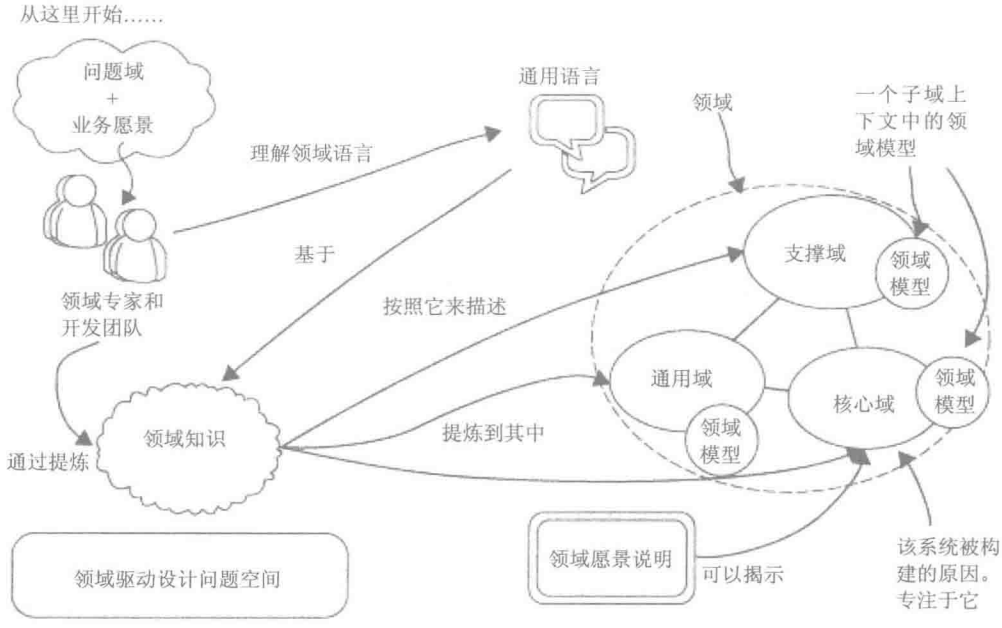


图 1　DDD 问题空间的一张蓝图

解空间

当你很好地理解问题域时，DDD 的战略模式就能帮助你实现与问题空间协作的技术解决方案。模式能让对于产品成功至关重要的系统的核心部分受到保护，以免被通用区域破坏。隔离整体组建使得它们可以被修改而无须对整个系统造成意外影响。

产品中非常复杂或者将频繁修改的核心部分应该基于一个模型。DDD 的战术模式以及模型驱动设计会帮助你在代码中创建领域的一个有用模型。模型就是让应用程序满足业务用例的所有领域逻辑的存放位置。将模型与技术复杂性分离开来能够让业务规则和策略得以演化。与问题域协作的模型将让其他开发人员和业务专家适应并理解软件。

图 2 显示了 DDD 解空间的高层次概览，本书第 I 部分将介绍它。

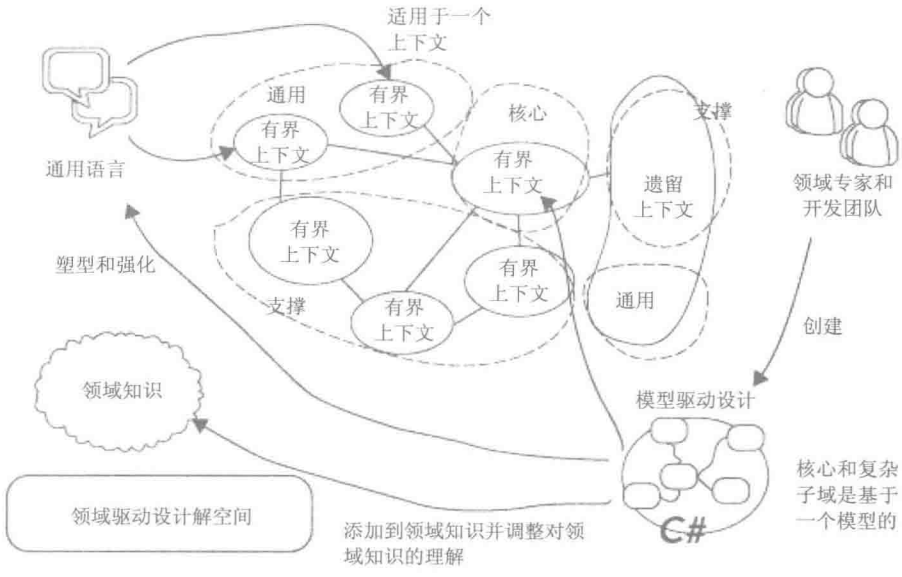


图 2　领域驱动设计解空间的一张蓝图

本书内容结构

本书分为四个部分。第 I 部分主要是介绍 DDD 的思想体系、原则以及实践。第 II 部分详细介绍集成有界上下文的战术模式。第 III 部分将介绍创建有效领域模型的战略模式。第 IV 部分将深入介绍你可以应用的设计模式，以便利用领域模型和构建有效应用程序。

第 I 部分：领域驱动设计的原则与实践

第 I 部分将介绍 DDD 的原则与实践。

第 1 章：什么是领域驱动设计

DDD 是一种思想体系，它有助于应对为复杂领域构建软件的挑战。这一章将介绍该思想体系并说明为何语言、协作和上下文是 DDD 最重要的方面，以及为何它远不止是一个代码模式的集合。

第 2 章：提炼问题域

理解一个复杂问题域对于创建可维护的软件来说是必不可少的。与领域专家一起进行知识提炼是学习该知识的关键。第 2 章将详细介绍让开发团队能够协作、实验并且向领域专家学习以便创建一个有效领域模型的技术。

第 3 章：专注于核心领域

第 3 章将说明如何提炼大型问题域以及识别出一个问题最重要的部分：核心领域。然后本章将说明为何你应该将时间和精力主要花在核心领域，以及为何应该将它从不重要的支撑域和通用域中隔离出来。

第 4 章：模型驱动设计

业务同事会理解基于你正在处理的问题区域的分析模型。开发团队有其专用的这个模型的代码版本。为了让业务人员和技术团队协作，就需要单一模型。通用语言和对问题空间的共识能够将分析模型与代码模型绑定起来。公共语言的概念是 DDD 的核心并且会强化该思想体系。由开发团队和业务专家共同创建的描述领域术语和概念的语言对于帮助复杂系统的沟通至关重要。

第 5 章：领域模型实现模式

第 5 章在应用程序中领域模型的作用及其承担的职责上进行了扩充介绍。这一章也介绍了能用于实现领域模型的各种模式以及它们最适用于哪些情况。

第 6 章：使用有界上下文维护领域模型的完整性

在大型解决方案中，可能存在多个模型。重要的是保护每个模型的完整性，以避免语言和概念中的歧义被不同团队不恰当地重复使用。被称为有界上下文的战略模式旨在隔离和保护有界上下文中的模型，同时确保它能与其他模型协作。

第 7 章：上下文映射

使用一个上下文映射来理解应用程序中不同模型之间的关系以及它们如何集成对于战略模式至关重要。上下文映射不仅会涵盖技术集成，还会涵盖团队之间的政治组织关系。上下文映射提供了环境的一个视图，它能帮助团队在整个环境的上下文中理解其模型。

第 8 章：应用程序架构

应用程序需要能够利用领域模型来满足业务用例。第 8 章将介绍结构化应用程序的架构模式，以便维持领域模型的完整性。

第 9 章：团队开始应用领域驱动设计通常会遇到的问题

第 9 章描述了在应用 DDD 时团队将面临的常见问题以及为何知道何时不使用它很重要。这一章还会重点介绍为何将 DDD 应用于简单问题会导致过度设计的系统和不必要的复杂性。

第 10 章：应用 DDD 的原则、实践与模式

第 10 章会介绍推广应用 DDD 以及开始将其原则和实践应用于你的项目的技术。这

一章将解释，相对于尝试创建完美的领域模型，探究和实验为何对于构建伟大软件更为有用。

第 II 部分　战略模式：在有界上下文之间通信

第 II 部分向你说明了如何集成有界上下文，并且提供了可用于架构有界上下文的选项的详细介绍。所提供的代码示例详细说明了如何集成遗留应用程序。还会介绍用于跨有界上下文通信的技术。

第 11 章：有界上下文集成介绍

现代软件应用程序都是具有可扩展性和可靠性需求的分布式系统。这一章将混合使用 DDD 的分布式系统理论，以便你在应用时可以两全其美。

第 12 章：通过消息传递集成

构建了一个示例应用程序来说明如何使用用于异步消息传递的消息总线来应用与 DDD 协作的分布式系统原则。

第 13 章：通过使用 RPC 和 REST 的 HTTP 来集成

构建了另一个示例应用程序来说明构建异步分布式系统的一种可替代方式。这一方式使用了像超文本传输协议(HTTP)、REST 以及 Atom 这样的标准协议来替代消息总线。

第 III 部分　战术模式：创建有效的领域模型

第 III 部分将介绍你可以用来在代码中构建领域模型的设计模式、持久化模型的模式以及管理组成模型的领域对象的生命周期的模式。

第 14 章：构造块领域建模的介绍

这一章会介绍可随你支配处理的所有战术模式，它们允许你构建一个有效的领域模型。本章重点介绍在代码中生成更可管理和更具表述性模型的一些最佳实践指导。

第 15 章：值对象

这是对代表了像资金这样的无身份领域概念的 DDD 建模构造的介绍。

第 16 章：实体

实体是具有身份的领域概念，比如顾客、事务和酒店。这一章将介绍各种示例以及互补的实现模式。

第 17 章：领域服务

有些领域概念是不属于值对象或实体的无状态操作。它们被称为领域服务。

第 18 章：领域事件

在许多领域中，专注于事件将比只专注于实体揭示出更深刻的见解。这一章将介绍允许你在领域模型中更加清晰表述事件的领域事件设计模式。

第 19 章：聚合

聚合代表领域概念的领域对象的群集。聚合是围绕不变条件定义的一致性边界。它们是最强有力的战术模式。

第 20 章：工厂

工厂是从复杂领域对象的构造中分离出使用的生命周期模式。

第 21 章：存储库

存储库介于领域模型和底层数据模型之间。它们会确保领域模型与所有基础架构问题保持分离。

第 22 章：事件溯源

就像第 18 章中的领域事件一样，事件溯源也是在代码中强调发生在问题域中的事件的重要性的一种有用技术。通过将领域模型的状态存储为事件，事件溯源超越了领域事件的应用。这一章提供了若干示例，其中包括使用以构建事件存储为目的的示例。

第 IV 部分　有效应用程序的设计模式

第 IV 部分展示用于架构利用和保护领域模型完整性的应用程序的设计模式。

第 23 章：应用程序用户界面的架构设计

对于由许多有界上下文构成的系统来说，用户界面通常需要从若干有界上下文中组合数据，尤其是在你的有界上下文组成了一个分布式系统的时候。

第 24 章：CQRS：一种有界上下文的架构

CQRS 是一种设计模式，它会创建两个模型，而原本只有一个模型。相对于单个模型处理读取和写入这两个不同的上下文，现在会创建两个明确的模型来处理命令或者服务于报告的查询。

第 25 章：命令：用于处理业务用例的应用程序服务模式

了解应用程序和领域逻辑之间的区别以保持模型的专注性以及系统的可维护性。

第 26 章：查询：领域报告

业务人员需要信息来制定富有见解的业务和产品开发的决策。本章将阐释一系列构建增加业务人员自主权的报告的技术。

本书读者对象

本书将介绍 DDD 背后的主题——其事件、模式和原则，以及对该思想体系的个人经验和理解。本书旨在为那些对该思想体系有兴趣或开始应用该思想体系的人提供学习方面的帮助。本书并非 Eric Evans 编著的 *Domain-Driven Design: Tackling Complexity in the Heart of Software*(Addison-Wesley 出版社于 2003 年出版)一书的替代书籍。相反，本书采用了 Evans 提出的一些概念并且将它们提炼成简单直接的具有实践示例的文字，以便所有的开发人员都能在继续深入学习其主题之前快速理解该思想体系。

本书内容基于作者具有主观认识的个人经验。如果你是一位经验丰富的 DDD 实践者，那么你可能并不会总是认同这些经验，但你仍旧应该从本书中汲取一些知识。

概述

本书的目的在于用一种脚踏实地及事件的方式来介绍 DDD 的思想体系，以便有经验的开发人员能为复杂领域构建应用程序。本书的一部分内容会专注于介绍分解复杂问题空间的原则和实践，以及构成可维护解空间的实现模式和最佳实践。你将了解如何通过使用战术模式构建有效领域模型，以及如何通过应用 DDD 的战略模式来保持它们的完整性。

阅读完本书后，你将对 DDD 有一个全面了解。你将能够表述其价值以及何时使用它。你将理解，即便 DDD 的战术模式很有用，但帮助你构架可维护和可扩展应用程序的是其原则、实践和战略模式。有了从本书中获得的信息，你就做好了为用于大型复杂问题域的复杂软件管理构造和维护的准备。

勘误表

尽管我们已经尽了各种努力来保证文章或代码中不出现错误，但错误总是难免的，如果你在本书中找到了错误，例如拼写错误或代码错误，请告诉我们，我们将非常感激。通过勘误表，可以让其他读者节省时间、避免阅读和学习受挫，当然，这还有助于提供更高质量的书籍。请给 wkservice@vip.163.com 发电子邮件，我们就会检查你的信息，如果是正确的，就把它发送到该书的勘误表页面上，或在后续版本中采用。

要在网站上找到本书的勘误表，可以登录 www.wrox.com，通过 Search 框或书名列表查找本书，然后在本书的细目页面上，单击 Book Errata 链接。在这个页面上可以查看到 Wrox 编辑已提交和粘贴的所有勘误项。完整的图书列表还包括每本书的勘误表，网址是 www.wrox.com/misc-pages/booklist.shtml。

如果读者没有在 Book Errata 页面上找到自己发现的错误，那么请转到页面 http://www.wrox.com/contact/techsupport.shtml，针对你所发现的每一项错误填写表格，并将表格发给我们，我们将对表格内容进行认真审查，如果确实是我们书中的错误，我们将在该书的 Book

Errata 页面上标明该错误信息，并在该书的后续版本中改正。

p2p.wrox.com

P2P 邮件列表是为作者和读者之间的讨论而建立的。读者可在 p2p.wrox.com 上加入 P2P 论坛。该论坛是一个基于 Web 的系统，用于传送与 Wrox 图书相关的信息和相关技术，与其他读者和技术用户交流。该论坛提供了订阅功能，当论坛上有新帖子时，会给你发送你选择的主题。Wrox 作者、编辑和其他业界专家和读者都会在这个论坛上进行讨论。

在 http://p2p.wrox.com 上有许多不同的论坛，帮助读者阅读本书，在读者开发自己的应用程序时，也可以从这个论坛中获益。要加入这个论坛，需要执行下面的步骤：

(1) 进入 p2p.wrox.com，单击 Register 链接。

(2) 阅读其内容，单击 Agree 按钮。

(3) 提供加入论坛所需的信息及愿意提供的可选信息，单击 Submit 按钮。

(4) 然后就会收到一封电子邮件，其中的信息描述了如何验证账户，完成加入过程。

提示：不加入 P2P 也可以阅读论坛上的信息，但只有加入论坛后，才能发送自己的信息。

加入论坛后，就可以发送新信息，回应其他用户的帖子。可以随时在 Web 上阅读信息。如果希望某个论坛给自己发送新信息，可以在论坛列表中单击该论坛对应的 Subscribe to this Forum 图标。

对于如何使用 Wrox P2P 的更多信息，可阅读 P2P FAQ，了解论坛软件的工作原理，以及许多针对 P2P 和 Wrox 图书的常见问题的解答。要阅读 FAQ，可以单击任意 P2P 页面上的 FAQ 链接。

源代码

学习本书中的示例时，可以手工输入所有的代码，也可以使用本书附带的源代码文件。本书使用的所有源代码都可以从本书合作站点 www.wrox.com 和 http://www.tupwk.com.cn/downpage 上下载。登录到站点 www.wrox.com，使用 Search 框或书名列表就可以找到本书，接着单击本书细目页面上的 Download Code 链接，就可以获得所有的源代码。

下载了代码后，只需用自己喜欢的解压缩软件对它进行解压缩即可。另外，也可以进入 www.wrox.com/dynamic/books/download.aspx 上的 Wrox 代码下载主页，查看本书和其他 Wrox 图书的所有代码。

目　录