TURING 图灵程序设计丛书

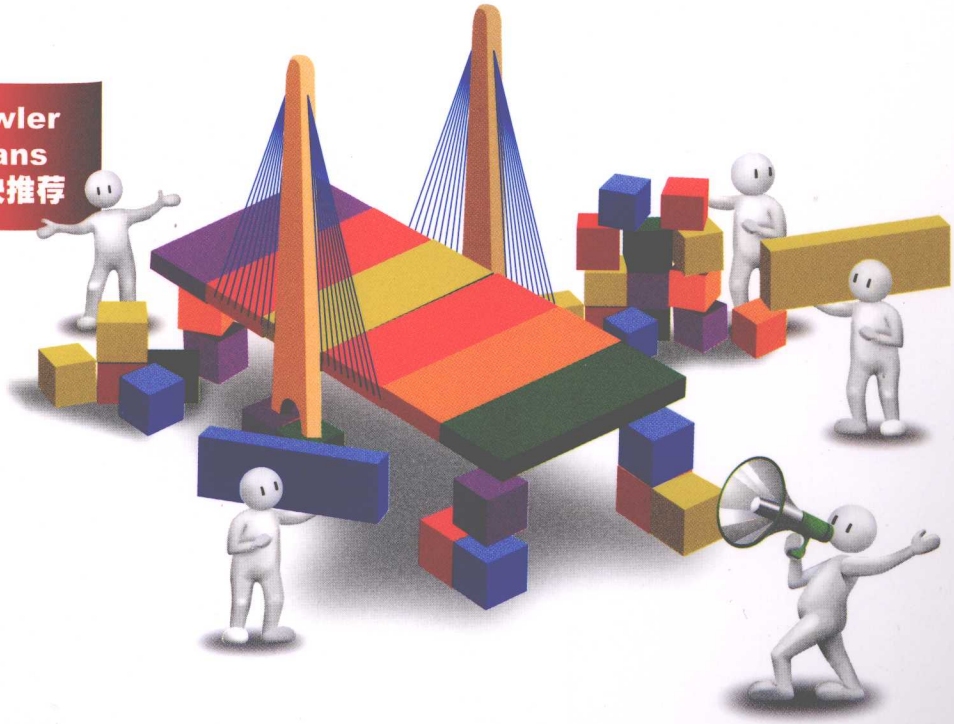
Addison Wesley

# Applying Domain-Driven Design and Patterns

# 领域驱动设计与模式实战

[瑞典] Jimmy Nilsson 著
赵俐 马燕新 等译

Martin Fowler
和Eric Evans
两位大师联袂推荐



- .NET开发人员必读之作
- 《企业应用架构模式》与《领域驱动设计》两大名著精髓的实战演练
- 教你穿越业务层、数据层和UI层之间重重障碍，打通任督二脉

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

# Applying Domain-Driven Design and Patterns

# 领域驱动设计与模式实战

[瑞典] Jimmy Nilsson 著
赵俐 马燕新 等译

人民邮电出版社
北 京

**图书在版编目（CIP）数据**

领域驱动设计与模式实战 /（瑞典）尼尔森（Nilsson, J.）著；赵俐等译. —北京：人民邮电出版社，2009.11
（图灵程序设计丛书）
ISBN 978-7-115-21277-1

Ⅰ. 领… Ⅱ. ①尼…②赵… Ⅲ. 软件设计 Ⅳ. TP311.5

中国版本图书馆CIP数据核字（2009）第149514号

## 内 容 提 要

本书全面详细地解释了领域驱动设计、测试驱动开发、依赖注入、持久化、重构、模式等很多基本概念，并以 C# 和 .NET 实例为依托，展示了这些概念的实际应用和重要价值。更重要的是，本书还将这些概念整合到一起，为开发人员从头至尾地揭示了完整的开发路线。阅读本书后，读者将能真正掌握这些重要概念，并有效地将它们结合起来，应用到实际开发过程中。

本书适合软件架构师和开发人员阅读。

图灵程序设计丛书

**领域驱动设计与模式实战**

◆ 著 [瑞典] Jimmy Nilsson
译 赵俐 马燕新 等
责任编辑 傅志红
执行编辑 陈兴璐

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京顺义振华印刷厂印刷

◆ 开本：800×1000 1/16
印张：23.75
字数：561 千字 2009 年 11 月第 1 版
印数：1－3 000 册 2009 年 11 月北京第 1 次印刷

著作权合同登记号 图字：01-2007-1481号
ISBN 978-7-115-21277-1

定价：69.00元

**读者服务热线：(010)51095186 印装质量热线：(010)67129223**
**反盗版热线：(010)67171154**

# 序 一

构建企业应用程序并非易事。尽管我们拥有大量工具和框架来简化这项任务，但仍然需要弄清楚如何更好地使用这些工具。大量方法可供我们选用，但关键是要知道在特定情况下应使用哪种方法，因为一种方法很难适用于所有情况。在过去几年中，有一个社区逐渐成长起来，人们不断寻找用于设计企业应用的方法，并以模式的形式将它们记录下来（我整理了一份概述，在http://martinfowler.com/articles/enterprisePatterns.html站点上）。参与这项工作的人们（例如我）试图找到公共方法，并描述如何更好地使用它们，以及它们何时适用。最后的结果过于宽泛，会导致为读者提供了过多的选择。

当我开始创作*Patterns of Enterprise Application Architecture*（《企业应用架构模式》，Addison-Wesley公司2002出版）时，我曾在微软技术方面寻找过这种类型的设计建议。几经努力之后几乎一无所获，只找到了一本讨论此领域的书，就是Jimmy的前一本书。我喜欢他平易近人的写作风格，也喜欢他深入挖掘易被他人忽略的概念的热情。因此，Jimmy决定从我和企业模式社区中的其他人借鉴一些思想，并向读者展示在编写.NET应用程序时如何应用这些思想，我觉得再合适不过了。

这个企业模式社区的主旨是记录好的设计，但另一个思路也贯穿整个社区。我们也是敏捷方法的忠实爱好者，信奉诸如测试驱动开发（TDD）和重构这样的技术。因此，Jimmy也把这些思想带到这本书中。很多人认为模式与TDD是不一致的，因为模式的重点是设计，而TDD的重点是演进。但有很多人正在将这两种技术结合起来使用，这证明这种观点是错误的，Jimmy将这些思路也整合到本书中。

结果就诞生了这本有关在.NET环境中进行设计的书。它受到敏捷方式的推动，又倾注了企业模式社区的成果。它是一本向读者展示了如何将TDD、对象-关系映射和领域驱动设计等方法应用于.NET项目的书。如果读者以前未遇到过这些概念，那么会发现它是一本入门书，书中介绍的技术在很多开发人员看来是未来软件开发的关键。如果你熟悉这些思想，那么本书将帮助你将这些思想传递给你的同事。

很多人感觉微软技术社区在传播好的企业应用设计建议方面不如其他社区。随着技术越来越强大，复杂度越来越高，理解如何更好地使用技术就变得越来越重要。本书在推进这种理解方面迈出了可贵的一步。

Martin Fowler

http://martinfowler.com

# 序　二

学习领域驱动设计（DDD）的最好方法是坐在一位友好、耐心且经验丰富的从业者身边，一步一步地共同研究问题。阅读本书正是这种体验。

本书并不是推出一个新的宏大方案，而是自然地报告了一位专家从业者是怎样使用及组合那些吸引他的当前实践的。

Jimmy Nilsson重申了我们很多人都曾经说过的话：几个时下流行的主题，特别是领域驱动设计（DDD）、企业应用架构模式（PoEAA）和测试驱动开发（TDD），它们并不是彼此互斥的替代品，而是成功开发中互相促进的元素。

此外，这三种技术实际上比它们乍看起来要难得多。它们需要各个领域的渊博知识。本书在倡导这些方法上花费了一定的时间，但主要还是关注如何让它们发挥作用的细节。

有效的设计不仅仅是通过死记硬背来学会的一组技术，它更是一个思考的过程。当Jimmy深入剖析一个示例时，他为我们打开了了解他思维的一扇窗子。他不仅展示和解释了他的解决方案，而且让我们看到他是如何得到解决方案的。

当我设计某个系统时，有数十种想法会在脑海中闪现。如果它们是我经常处理的因素，那么会很快地闪过，我几乎意识不到。如果它们属于我不太确信的领域，那么我会更多地思考它们。我认为这是设计者遇到的典型现象，但很难将它传达给另一个人。当Jimmy仔细讨论他的示例时，就好像他把这个过程放了慢动作一样。在每个小的转折点，都会呈现3～4种选择，Jimmy对它们进行权衡和扬弃，最终选择最有利的一个。

例如，我们希望在不涉及持久化技术的情况下实现模型对象。那么，持久化框架会强制性“污染”领域对象实现的8种（是8种！）方式都是什么？是什么因素导致我们在其中的一些要点上做出折中？当前流行的框架（包括.NET）都强加了什么约束？

Jimmy的思考注重实效。他利用他的经验做出一种可能实现最终目标，并且遵守更深层次设计原则的设计选择，而不是看起来最符合教科书示例的选择。而且他的所有决定都是临时的。

Jimmy奉行的第一条设计原则就是DDD的基本目标：一个反映了对业务问题深刻理解的设计，而且在设计形式上，要使设计能够根据新方法做出调整。那么为什么还要讨论这么多技术框架和架构呢？

有一种常见（也很自然）的误解是：这种强调领域问题的优先考虑不需要太多技术才能和技巧。如果这是真的那就好了，要想成为一名胜任的领域设计师并不十分困难。可是，要想在软件中呈现清晰且有用的领域概念，以便它们不被大量的技术细节所淹没，就要求能娴熟地运用技术。据我观察，那些熟练掌握技术和架构原则的人通常知道如何把技术放在其适合的位置上，而且这

些人也是最高效的领域建模人员。

我并不是说应该面面俱到地掌握复杂工具的所有知识，而是强调应该掌握Martin Fowler的PoEAA当中所展示的那类知识，因为盲目应用技术反而会导致对应用产生干扰。

对很多人来说，本书将填补一个空白，教会大家怎样在实践中实现富有表现力的对象模型。我从本书学到不少思考技术框架应用的有用方法，尤其是加强了我对在.NET平台上使用DDD方法的一些理解。

除了技术架构以外，Jimmy还在如何编写测试上花费了大量时间。TDD以一种不同方式补充了DDD。当对于精化更有用模型的重视程度不够时，TDD容易导致零散的应用，这时对某一特性的一味追求会使系统最终无法扩展。综合的测试套件实际上使开发团队能够继续取得进展（与没有TDD时相比），但这只是TDD的最基本价值。

当测试套件充分发挥作用时，它是域模型的实验室，也是通用语言的技术表示。特殊风格的测试推动建模过程向前发展，并保持整个建模过程始终围绕重点展开。开发这些测试的示例贯穿了本书。

Jimmy Nilsson集自信和谦逊于一身，我注意到这是最优秀设计人员的特征。当他说到以前习惯于相信什么，后来为什么观点又改变的时候，我们可以管窥到他是如何获得当前理解的，这帮助读者穿越技术细节，从而理解基本原则。这种谦虚的品质帮助他集思广益，并因此为我们奉上了这本融合各种不同思想的书。他做了各种各样的尝试，并用结果和经验作为自己的指导。他的结论不是以被揭示的真理的面目出现，而是作为他到目前为止的最佳理解，其中的奥妙令我们回味无穷。所有这些都使得他的建议对读者更有用。而且这种态度本身也是领导成功软件开发所需的一个重要元素。

Eric Evans

# 前言：跨越鸿沟

本书的封面图片[①]是连接瑞典和丹麦的厄勒海峡大桥。似乎所有软件架构图书的封面都要有一座大桥，但本书选用大桥作为封面却有一些其他原因。

就是这座大桥取代了我小时候乘坐过很多次的渡船。驱车穿越大桥是一种绝佳的体验，虽然我已经不下数十次穿越它，但还是乐此不疲。

题外话，我父亲曾经就在负责架设这座大桥最高部分的施工队中工作。

但这些并不是最主要的原因，主要原因是本书内容正是跨越鸿沟，即跨越用户与开发人员之间的鸿沟，跨越业务与软件之间的鸿沟，跨越逻辑与存储之间的鸿沟，跨越DB人员与OO人员之间的鸿沟……

我并不是拿桥接模式[GoF Design Patterns]来信口开河，这毕竟是前言啊！

## 本书的主旨

本书的主旨是如何将领域模型构建得整洁，且仍具有便于持久化的特性。书中展示了在这样一个领域模型中，持久化解决方案将以什么面貌出现，特别是如何在领域模型与数据库这道鸿沟上架起一座桥梁。

换言之，我的愿望是在本书中将Eric Evans的《领域驱动设计》[Evans DDD]和Martin Fowler的《企业应用架构模式》[Fowler PoEAA]融为一体。

人们可能认为DDD有些抽象。因此，具体示例有助于对本书的理解，例如对持久化概念的理解。这些示例可能比较基础，但它们可作为良好的开端。本书不仅解释了如何使用模式，而且阐述了如何在O/R映射工具中使用模式。

我非常清楚在架构方面“一种规格并非处处适用”。尽管如此，事实证明模式还是具有足够的通用性的，我们可以在各种各样的情况下使用和重用它们。

本书的重点并不是讲模式本身，而是在各个章节中穿插使用它们，将其作为工具和语言，用于讨论不同的设计方面。在这个过程中，不熟悉模式的读者可以获得模式的一些知识并对模式产生兴趣。

对于TDD也是如此。但是，并非所有开发人员都会对此感兴趣，在.NET社区中尤其如此，因为TDD（仅作为模式）被认为是一种应用范围很狭窄的技术，甚至完全是一种默默无闻的技术。本书将教会读者应用TDD。

---

① 指本书的英文原版封面图片。——编者注

## 本书的目的

在不耽搁其他日常项目和工作的情况下撰写我的第一本书[Nilsson NED]，确实让我体会到了什么是艰苦。我当时已下定决心不再写书了。但时间还是改变了这个决定，因为我有话要说，而且不能不说了。

想法的改变源于最近阅读的两本书给我的灵感和触动。第一本书是Martin Fowler的《企业应用架构模式》[Fowler PoEAA]。这本书激发了我再次尝试领域模型模式的兴趣，尽管此前曾有过几次失败经历。

然后我读到了Eric Evans的《领域驱动设计》[Evans DDD]。这本书使我深入理解了如何思考和进行以领域为核心的开发，以及如何在这样的开发中使用特定的领域模型模式风格。

另一个重要影响就是在我多年的模式课程教学过程中所积累的知识。通过与学生们的交流以及教学内容的发展，我自己也有所感悟。

我参与一个宏大的（遗憾的是尚未完成）开源项目Valhalla上工作时，对DDD的认识也发生了转变，这个项目是我和Christoffer Skjoldborg协作开发的。（到目前为止，大部分工作都是由Christoffer完成的。）

总之，我觉得需要写一本应用多于理论的书，但这本书又要有坚实的基础，就像DDD和PoEAA那样。“应用”更贴近我的心声，因为我始终认为自己归根到底是一名开发人员。

## 本书的读者

本书面向广泛的读者。具备以下知识有助于阅读本书。

- 面向对象
- .NET或类似平台
- C#或类似语言
- 关系数据库，例如SQL Server

但是，兴趣和热情可以弥补任何经验的不足。

这里再详细解释一下为什么目标读者是广泛的。首先，我们从平台框架来考虑一下适用的人员。本书面向那些需要一种更关注核心的方法而不是简单的拖放操作（可能这个概括并不是很恰当）的.NET开发人员。Java开发人员可以从如何结合DDD和O/R映射的讨论及示例中获益。

我认为选定的语言和平台越来越不重要了，因此谈论.NET开发人员和Java开发人员看起来有些不合时宜。那么就让我们试着用另一种角度来描述目标读者，即，本书是面向开发人员、团队领导者和架构师的。

选择这种角度后，我认为本书中有些内容既适合中级读者，也适合高级读者。可能也会有些内容适合初级读者。

## 本书的组织

本书共分四部分：背景知识、应用DDD、应用PoEAA和下一步骤。

## 第一部分：背景知识

背景知识这一部分概括地讨论架构和过程，重点讲解领域模型和DDD（领域驱动设计）[Evans DDD]，还将介绍模式和TDD（测试驱动开发）。第一部分包括以下几章。

- 第1章，应重视的价值

该章讨论要想得到高质量的系统开发结果，应重视的架构属性和过程。这一章的讨论受到极限编程（XP）的影响。

- 第2章，模式起步

该章主要提供示例，并讨论不同类型的模式，例如设计模式、架构模式和领域模式。

- 第3章，TDD与重构

第1章讨论了很多TDD和重构内容，而这一章则通过较长的示例并着眼于不同的TDD风格来深入阐述这些内容。

## 第二部分：应用 DDD

这一部分介绍DDD的应用。此外还会为基础架构准备领域模型，重点关注规则方面。

- 第4章，新的默认架构

该章列出了示例应用程序的各方面需求，并初步创建了一个模型作为后续章节的起点。这里使用了基于领域模型的架构。

- 第5章，领域驱动设计进阶

这一章使用第4章中提出的需求作为开始构建DDD风格的领域模型的基础。在这一章中，由于是刚开始使用TDD方法，因此我们放慢构建速度。

- 第6章，准备基础架构

尽管我们尽量推迟基础架构方面，但还是有必要稍微提前思考一下，并针对基础架构需求为领域模型做一些准备。这一章讨论持久化透明（PI）领域模型的利弊。

- 第7章，遵守规则

这一章讨论验证形式的业务规则，以及基于领域模型的解决方案如何处理这样的规则需求，从而与第4章中的需求联系起来。

## 第三部分：应用 PoEAA

在这一部分中，我们将Fowler的《企业应用架构模式》[Fowler PoEAA]中的几种模式作为上下文，以此来讨论基础架构需要为领域模型提供哪些持久化支持。我们将看一下如何通过示例工具来满足这些需求。

- 第8章，用于持久化的基础架构

当我们拥有了很好的领域模型时，就该好好考虑基础架构了，本书中的主要基础架构类型是用于持久化的基础架构。第8章讨论持久化解决方案的不同属性，以及如何对特定解决方案进行分类。

- 第9章，NHibernate实战

这一章通过一个持久化解决方案的示例来使用前一章中的分类，这个示例就是

NHibernate[NHibernate]。

### 第四部分：下一步骤

这一部分主要关注并开始使用其他一些设计技术。另外一个重点是如何在领域模型中处理表示层，以跨越领域模型与表示层之间的鸿沟，此外还将介绍开发人员如何测试UI。这一部分几乎都是邀请其他作者编写的。

- 第10章，博采其他设计技术

这一章首先简单讨论限界上下文（bounded context），然后讨论现在和将来都应关注的一些设计技术，例如面向服务、依赖注入/控制反转，以及面向方面。

- 第11章，关注UI

这一章主要讨论如何将UI连接到领域模型，以及在使用领域模型时如何提高富客户端应用程序和Web应用程序用户界面的可测试性。

### 附录

本书有两个附录，附录A包含更多领域模型风格的示例，附录B提供了简明的模式目录。

## 关于 C#示例

本书绝不是一本讲述C#的书。但我仍需要一种用于编写示例的语言（语言有多种，但我必须从中选择一种），这样就选择了C#。

选择C#的主要原因是我当前经常使用它，而且大多数VB.NET和Java开发人员都很容易看懂C#代码。

至于版本方面，本书大部分示例代码可以同时在C# 1.1和2.0中运行，个别章节的代码只能在C# 2.0中运行。

## 未涵盖的主题

本书中未涵盖的主题有很多，最主要的是分布式和高级建模。

### 分布式

本书最早的写作计划中是包含分布式方面的完整讨论的，但后来我发现这样本书的重点就过于分散了。但一些章节中仍然穿插了一些这方面的内容。

### 高级建模

本书的书名可能暗示书中会包含特定问题领域的一些高级和有趣的建模示例，但事实并不是这样。我们示例的重点更多地放在了应用TDD以及为DDD添加基础架构上。

## 最后的感言

相信读者能够理解，像本书这样的项目几乎从来不是一个人就能完成的。它是很多人共同努

力的结果。请参见致谢部分的人员名单，但还有更多参与人员未在这个名单中列出，特别是那些负责本书印制的人。虽然如此，但任何在印刷之前未发现的错误都应归咎于我一个人。

我将把读者感兴趣的一些信息发布到网上：www.jnsk.se/adddp。

再回头说说跨越鸿沟，厄勒海峡大桥的照片是我的朋友Magnus von Schenck在他的一次海上旅行中拍摄的。

尽管本书的创作不像第一本书那样艰辛，但其中也凝聚了无数的心血、汗水和泪水。我希望通过我的这些付出，能够减少读者的付出。最后愿读者从阅读本书中获得乐趣，祝你们好运！

Jimmy Nilsson
www.jnsk.se/weblog/
2005年9月于瑞典Listerby

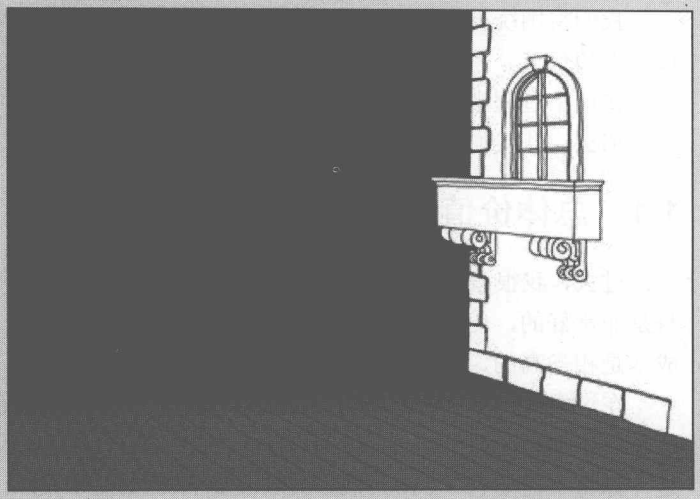
# 目　录

## 第二部分 应用DDD

# Part 1 第一部分

# 背景知识

背景知识这一部分概括地讨论架构和过程，重点讲解领域模型和 DDD（领域驱动设计）[Evans DDD]，还将介绍模式和 TDD（测试驱动开发）。



第一部分是有关布置场景的。《罗密欧与朱丽叶》需要有一个场景。

## 本部分内容