

Jimmy Nilsson



# Applying Domain-Driven Design and Patterns

---

With Examples in C# and .NET

Forewords by Martin Fowler and Eric Evans

# Praise for *Applying Domain-Driven Design and Patterns*

*“I don’t know what it was I professed to doing before I had added Domain-Driven Design and Test-Driven Development to my toolkit, but from my present perspective, I’m reticent to call it anything but chaotic hacking. Domain-Driven Design and Test-Driven Development are two approaches that have consistently guided me toward a practical application of software design principles, and brought my projects to fruition with healthy, sustainable software. This is a book that puts its money where its mouth is in terms of concretely communicating Agile software development practice. It’s potentially one of the most impactful guides to software development success practices yet offered to the .NET software development community.”*

—Scott Bellware, Microsoft MVP for C#, DDD and TDD Blogger

*“Jimmy Nilsson does his readers a great service by showing them how to apply the foundational principles of enterprise application design and development taught by Evans, Fowler, and other thought leaders. The book uses a worked example not only to explain, but also to demonstrate Domain-Driven Design, Patterns of Enterprise Application Architecture, and Test-Driven Development. Jimmy’s insight and experience make reading it a pleasure, and leave the reader with the certainty that they have learned from a master practitioner. Enterprise developers looking to master these principles will find the book a valuable guide.”*

—Jack Greenfield, Enterprise Tools architect, Visual Studio Team System, Microsoft

*“Good software architects reserve the right to get smarter. Beyond the goal of shipping their current system, they’re on a quest to discover better ways to design and build software. This book is travelogue of sorts in which Jimmy documents his journey through a wide range of patterns, practices, and technologies, explaining how his thinking about enterprise systems has evolved along the way. If you’re traveling the same road, this book is a good companion.”*

—Tim Ewald, principal architect at Foliage Software Systems and author of *Transactional COM+: Building Scalable Applications*

# Applying Domain-Driven Design and Patterns: With Examples in C# and .NET

## Table of Contents

Contents

About the Author

Forewords

Preface: Bridging Gaps

PART I: BACKGROUND

Chapter 1: Values to Value

Overall Values

Architecture Styles to Value

Process Ingredients to Value

Continuous Integration

Don't Forget About Operations

Summary

Chapter 2: A Head Start on Patterns

A Little Bit About Patterns

Design Patterns

Architectural Patterns

Design Patterns for Specific Types of Applications

Domain Patterns

Summary

Chapter 3: TDD and Refactoring

Test-Driven Development (TDD)

Mocks and Stubs

Refactoring

# **Table of Contents**

Summary

## **PART II: APPLYING DDD**

### **Chapter 4: A New Default Architecture**

The Basis of the New Default Architecture

A First Sketch

Making a First Attempt at Hooking the UI to the Domain Model

Yet Another Dimension

Summary

### **Chapter 5: Moving Further with Domain-Driven Design**

Refining the Domain Model Through Simple TDD Experimentation

Fluent Interface

Summary

### **Chapter 6: Preparing for Infrastructure**

POCO as a Lifestyle

Dealing with Save Scenarios

Let's Build the Fake Mechanism

Database Testing

Querying

Summary

### **Chapter 7: Let the Rules Rule**

Categorization of Rules

Principles for Rules and Their Usage

Starting to Create an API

Requirements for a Basic Rules API Related to Persistence

Focus on Domain-Related Rules

Extending the API

Refining the Implementation

Binding to the Persistence Abstraction

Generics and Anonymous Methods to the Rescue

# **Table of Contents**

What Others Have Done

Summary

## **PART III: APPLYING PoEAA**

### **Chapter 8: Infrastructure for Persistence**

Requirements on the Persistence Infrastructure

Where to Store Data

Approach

Classification

Another Classification: Infrastructure Patterns

Summary

### **Chapter 9: Putting NHibernate into Action**

Why NHibernate?

A Short Introduction to NHibernate

Requirements of the Persistence Infrastructure

Classification

Another Classification: Infrastructure Patterns

NHibernate and DDD

Summary

## **PART IV: WHAT'S NEXT?**

### **Chapter 10: Design Techniques to Embrace**

Context Is King

An Introduction to SOA

Inversion of Control and Dependency Injection

Aspect-Oriented Programming (AOP)

Summary

### **Chapter 11: Focus on the UI**

A Prelogue

The Model-View-Controller Pattern

Test-Driving a Web Form

# **Table of Contents**

Mapping and Wrapping

Summary

Epilogue

## **PART V: APPENDICES**

### **Appendix A: Other Domain Model Styles**

Object-Oriented Data Model, Smart Service Layer, and Documents

The Database Model Is the Domain Model

Pragmatism and the Nontraditional Approach

Summary

### **Appendix B: Catalog of Discussed Patterns**

**References**

**Index**