Extracted from:

Domain-Driven Design

using Naked Objects

The Pragmatic Programmers

Domain-Driven Design Using Naked Objects

Dan Haywood

Edited by Susannah Davidson Pfalzer

Our Pragmatic courses, workshops, and other products can help you and your team create better software and have more fun. For more information, as well as the latest Pragmatic titles, please visit us at

http://www.pragprog.com

Printed in the United States of America.

ISBN-10: 1-934356-44-1

ISBN-13: 978-1-934356-44-9

Printed on acid-free paper.

P1.0 printing, December 2009

Version: 2010-1-16

Preface

There's no doubt that writing enterprise applications is hard. To bring the thing together, you must master numerous technologies: frameworks to handle presentation, APIs to deal with persistence, yet more technology for remoting and authentication. . . the list goes on. Yet none of this matters to the business users asking for the system in the first place.

Domain-driven design is an approach to building application software that focuses on the bit that matters in enterprise applications: the core business domain. Rather than putting all your effort into technical concerns, you work to identify the key concepts that you want the application to handle, you try to figure out how they relate, and you experiment with allocating responsibilities (functionality). Those concepts might be easy to see (Customers, Products, Orders, and so on), but often there are more subtle ones (Payable, ShippingRecipient, and RepeatingOrder) that won't get spotted the first time around. So, you can use a team that consists of business domain experts and developers, and you work to make sure that each understands the other by using the common ground of the domain itself.

Even with your domain experts' help, building these domain models isn't always that easy. You need to be precise as to what the domain concepts actually mean, but that's easier said than done, especially if you're automating previously manual business processes. But if you can develop a deep and insightful model, then the benefits can be enormous. Not only will it deliver benefits to the business (reduced operational costs and the like), it will also be easy to extend and maintain (reducing time to market for future requirements).

So, there's your challenge: how do you bring the domain model to life so that it can be checked, verified, and refined? Static UML diagrams don't hack it; they are too technical for business users to understand.

No, what a nontechnical audience needs to see is rapidly developed working prototypes. And Naked Objects, an open source Java framework, gives you this capability. You write the classes that make up the domain model as plain old Java objects (pojos). Naked Objects then automatically renders those domain classes in an object-oriented user interface, running either as a rich client app or (with no extra coding) as a web application. You run the application using whichever user interface the business users find most comfortable.

With Naked Objects taking care of the presentation layer for you, you can focus solely on the domain. If you make a change to a domain class, you'll see the change immediately when you run the application, and that gives you the feedback loop you need for experimentation. You work with the business's domain experts trying out new ideas; if they come to naught, then it will have cost you little in terms of development effort, and your compensation will be a deeper knowledge of the domain.

This book is about using Naked Objects for domain-driven design, but it's not just about building prototypes. You will also see how Naked Objects defines a straightforward architecture for your domain-driven applications, making it easy to get started. You use Naked Objects' integration with the FitNesse acceptance testing framework, allowing you and your domain experts to work together to specify your application's functionality through scenario tests. And you see how to deploy a domain model as a fully working application using either Naked Objects' own object-oriented user interfaces or a custom user interface.

Who This Book Is For

This book is mostly targeted at developers working on enterprise applications, especially those working on line-of-business operational applications. When I say "developer," I use a broad brush. Naked Objects uses Java as its language, so if you have a Java background, then you'll be right at home. However, even if you typically use only the .NET languages or another object-oriented language, you'll find nothing too complex in the code here.[1]

1. If you are a .NET developer, you might like to know there is a commercial version of Naked Objects offering broadly equivalent functionality. See http://www.nakedobjects.net for further details.

More specifically, this book is for developers who also do business analysis, or at least work with business analysts; after all, domain-driven design requires an interest in thinking about domains. If your job title is plain business analyst, then you'll get a lot out of this book too—so long as the thought of some code doesn't scare you. And if you are a database specialist, you'll also find something of interest; there's a lot of common ground between logical data modeling and domain modeling.

If you are a project manager, you'll find this book opens up some new approaches for your development process. In particular, you might be interested in the ability to rapidly prototype (which takes in requirements capture), in writing scenario tests, and in the various deployment options that Naked Objects makes available. And if you were originally from a development background, you might also fancy the chance to do a bit of coding again!

How the Book Is Organized

This book is not a theoretical discourse on how to practice domain-driven design; instead, it's a "wade in there and get your hands mucky" sort of book. It breaks down into three parts.

In the first part, *Tools*, you'll learn the building blocks for developing domain-driven applications. You'll use Naked Objects to build up a single domain application, learning the conventions that make up the Naked Objects programming model as you go.

In the second part, *Techniques*, you'll look at ways to develop deeper insight within your domain models and to ensure that your domain applications are maintainable. You will lean on existing "prior art" for building object-oriented applications, but always from our domain-driven design/Naked Objects perspective. You'll also learn about two different ways to test your domain-driven applications.

In the third part, *Practices*, you'll explore the development process and deployment options. At one end of the spectrum, you'll look at using Naked Objects just as a design tool within development; at the other, you'll look in detail at how to integrate and deploy Naked Objects applications into production (taking in custom web interfaces, persistence, and enterprise integration). We finish up by looking at the various ways to deploy your domain-driven application and briefly discuss one of the largest domain-driven applications (600+ users), built with and deployed on Naked Objects.

There are also several reference appendixes, covering the Naked Objects programming model and other resources.

The book has an accompanying website from which you can download (as a single ZIP) the various code snippets scattered throughout the book.[2] The code download also includes the various versions of the example case study (discussed in the next section) as we build it up through the chapters.

By the end of the book your hands should be well and truly dirty! You will have seen the power of the Naked Objects approach for building domain-driven applications firsthand and will be ready to put it into use for your own domain applications.

Case Study and Exercises

Throughout the book you'll be using a single case study called CarServ, a vehicle-servicing application for a garage. CarServ suits a domain-driven approach because it's an enterprise system supporting core operational business requirements. It also suits our purposes as an example to learn from because it's a domain that most can relate to.

As each new idea is introduced, you'll immediately apply it to CarServ; and because Naked Objects generates the user interface for you, you'll also be able to *see* it in use. In the first two parts of the book, all the changes you need are fully described, so you can follow along either at your computer or in your head. As already mentioned, though, multiple versions of CarServ are available for download from the book's website. Each of these are self-contained and runnable without reference to previous versions, so you can pick up the story at any point. In Part III, because we are integrating with some other technologies, the CarServ downloads already have the necessary changes made, so you can download them and try them out; all the relevant code snippets are also in the book if you're not at your computer.

CarServ also acts as a good basis for structuring your own applications. One of the things that domain-driven design requires is a strict layered architecture, but figuring out how to put this together can be a challenge. CarServ demonstrates how Naked Objects provides a standard approach for the domain layer. As mentioned previously, the chapters

2. http://www.pragprog.com/titles/dhnako

in Part III explain how to integrate the domain layer with the other layers of the architecture.

At the end of every chapter are a number of exercises. Many of these ask you to enhance CarServ—for example, by adding an additional type of business rule. But the point of the book isn't about developing CarServ; it's so you can learn the skills to develop your own domain applications. Therefore, an ongoing exercise is for you to select an application and build it up alongside the case study; I guarantee it'll be worth your time.

Conventions

You'll see a few conventions throughout this book:

- Short sidebars (with a "DDD" icon) briefly explain domain-driven design terminology. Since this is a practical book rather than theoretical, use these sidebars as jumping-off points to further reading. I've used the same terms for domain-driven design as Eric Evans does in his book, *Domain-Driven Design* [Eva03].
- You'll occasionally see sidebars with a "Joe Asks" icon. Joe is the Pragmatic Bookshelf's mythical developer, and he likes to ask the odd question related to the text.
- ***Bold italics*** indicate when there is a version of the CarServ case study. You'll find these in the code download ZIP under the casestudy directory.
- Each code snippet appears with a little gray lozenge above it, specifying its location in the code download ZIP. This should save you typing if you are following along. And if you have bought the PDF version of the book, clicking the link goes directly to the code.

Before we start, let me just provide you with some further pointers.

Further Resources

This book will give you a great start toward writing fully fledged domain-driven applications. However, it's not the be-all and end-all; you should know about some other places. Since this is a "PragProg" book, there are the usual resources:

- The book's website is at http://www.pragprog.com/titles/dhnako.
- The book's errata is at http://www.pragprog.com/titles/dhnako/errata.
- The discussion group is at http://forums.pragprog.com/forums/106.

In addition, I've put together a blog that aims to supplement and extend the ideas in this book:

- http://danhaywood.com, "the blog of the book"

Finally, there are the websites for Naked Objects and its sister projects:

- Naked Objects is hosted via http://nakedobjects.org.
- Scimpi is hosted at http://scimpi.org.
- The sister projects that we use in some of the later chapters are collectively signposted from http://starobjects.org.

So, that's about it. I'm eager to get started, and I hope you are too. In Chapter 1 we're going to look at some of the key concepts of domain-driven design and see them in action by running our first Naked Objects application. See you there!

The Pragmatic Bookshelf

The Pragmatic Bookshelf features books written by developers for developers. The titles continue the well-known Pragmatic Programmer style and continue to garner awards and rave reviews. As development gets more and more difficult, the Pragmatic Programmers will be there with more titles and products to help you stay on top of your game.

Visit Us Online

Domain-Driven Design using Naked Objects Home Page
http://pragprog.com/titles/dhnako
Source code from this book, errata, and other resources. Come give us feedback, too!

Register for Updates
http://pragprog.com/updates
Be notified when updates and new books become available.

Join the Community
http://pragprog.com/community
Read our weblogs, join our online discussions, participate in our mailing list, interact with our wiki, and benefit from the experience of other Pragmatic Programmers.

New and Noteworthy
http://pragprog.com/news
Check out the latest pragmatic developments, new titles and other offerings.

Buy the Book

If you liked this eBook, perhaps you'd like to have a paper copy of the book. It's available for purchase at our store: pragprog.com/titles/dhnako.

Contact Us

Online Orders:	www.pragprog.com/catalog
Customer Service:	support@pragprog.com
Non-English Versions:	translations@pragprog.com
Pragmatic Teaching:	academic@pragprog.com
Author Proposals:	proposals@pragprog.com
Contact us:	1-800-699-PROG (+1 919 847 3884)