

Join the discussion @ [p2p.wrox.com](http://p2p.wrox.com)



Wrox Programmer to Programmer™



# Professional Test Driven Development with C#

*Developing Real-World Applications with TDD*

James Bender, Jeff McWherter



# PROFESSIONAL TEST-DRIVEN DEVELOPMENT WITH C#

---

INTRODUCTION.....	xxv
► <b>PART I    GETTING STARTED</b>	
CHAPTER 1    The Road to Test-Driven Development .....	3
CHAPTER 2    An Introduction to Unit Testing .....	19
CHAPTER 3    A Quick Review of Refactoring .....	41
CHAPTER 4    Test-Driven Development: Let the Tests Be Your Guide .....	73
CHAPTER 5    Mocking External Resources .....	97
► <b>PART II    PUTTING BASICS INTO ACTION</b>	
CHAPTER 6    Starting the Sample Application .....	117
CHAPTER 7    Implementing the First User Story .....	137
CHAPTER 8    Integration Testing .....	169
► <b>PART III    TDD SCENARIOS</b>	
CHAPTER 9    TDD on the Web .....	197
CHAPTER 10    Testing Windows Communication Foundation Services .....	227
CHAPTER 11    Testing WPF and Silverlight Applications .....	245
► <b>PART IV    REQUIREMENTS AND TOOLS</b>	
CHAPTER 12    Dealing with Defects and New Requirements .....	267
CHAPTER 13    The Great Tool Debate .....	279
CHAPTER 14    Conclusions .....	299
APPENDIX    TDD Katas .....	307
INDEX.....	311



PROFESSIONAL

# **Test-Driven Development with C#**



PROFESSIONAL

# **Test-Driven Development with C#**

DEVELOPING REAL WORLD APPLICATIONS WITH TDD

James Bender  
Jeff McWherter



WILEY

Wiley Publishing, Inc.

## **Professional Test-Driven Development with C#: Developing Real World Applications with TDD**

Published by  
Wiley Publishing, Inc.  
10475 Crosspoint Boulevard  
Indianapolis, IN 46256  
[www.wiley.com](http://www.wiley.com)

Copyright © 2011 by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-64320-4  
ISBN: 978-1-118-10210-7 (ebk)  
ISBN: 978-1-118-10211-4 (ebk)  
ISBN: 978-1-118-10212-1 (ebk)

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, (201) 748-6011, fax (201) 748-6008, or online at <http://www.wiley.com/go/permissions>.

**Limit of Liability/Disclaimer of Warranty:** The publisher and the author make no representations or warranties with respect to the accuracy or completeness of the contents of this work and specifically disclaim all warranties, including without limitation warranties of fitness for a particular purpose. No warranty may be created or extended by sales or promotional materials. The advice and strategies contained herein may not be suitable for every situation. This work is sold with the understanding that the publisher is not engaged in rendering legal, accounting, or other professional services. If professional assistance is required, the services of a competent professional person should be sought. Neither the publisher nor the author shall be liable for damages arising herefrom. The fact that an organization or website is referred to in this work as a citation and/or a potential source of further information does not mean that the author or the publisher endorses the information the organization or website may provide or recommendations it may make. Further, readers should be aware that Internet websites listed in this work may have changed or disappeared between when this work was written and when it is read.

For general information on our other products and services please contact our Customer Care Department within the United States at (877) 762-2974, outside the United States at (317) 572-3993 or fax (317) 572-4002.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

**Library of Congress Control Number: 2011924919**

**Trademarks:** Wiley, the Wiley logo, Wrox, the Wrox logo, Programmer to Programmer, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. Wiley Publishing, Inc., is not associated with any product or vendor mentioned in this book.



*For Gayle. Thank you for being so awesome!*

—JAMES

*To everyone who has believed in me.*

—JEFF

*To my wonderful wife Courtney and my  
two amazing kids, Katie and Jacob.*

—MICHAEL



# ABOUT THE AUTHORS

**JAMES BENDER** is Vice Present of Technology for Improving Enterprises and has been involved in software development and architecture for 17 years. He has worked as a developer and architect on everything from small, single-user applications to Enterprise-scale, multi-user systems. His specialties are .NET development and architecture, SOA, WCF, WF, cloud computing, and agile development methodologies. He is an experienced mentor and author.

James has spent his career pushing the envelope of software development and pursuing new and better ways of building applications. He began his career developing credit card processing applications in C++ on SCO Unix based systems. In the late 90's James began exploring web development with both Java based JSP pages and Microsoft's ASP technologies. He was an early adopter of .NET starting with the first public beta. He continued exploring the .NET technology stack, focusing on the distributed computing paradigm made possible by .NET web services, which naturally evolved into a somewhat obsessive interest in Microsoft's Windows Communication Foundation (WCF).

James has been practicing agile-based methodologies since 2003, including Scrum and eXtreme Programming (XP). At part of this interest in agile methodologies, James began exploring test-driven development at the same time. He was instrumental in introducing the concepts and techniques used in agile software development and test-driven development to many developers at his clients and in the software development community in general.

James is a Microsoft MVP for Visual C#. James is an active member of the development community. He is the current president of the Central Ohio .NET Developers Group ([www.condg.org](http://www.condg.org)) and continues to lead the Columbus Architects Group ([www.colarc.org](http://www.colarc.org)) and is the senior editor of first-party content for [nplus1.org](http://nplus1.org), an educational website aimed toward architects and aspiring architects. His blog can be found at [www.jamescbender.com](http://www.jamescbender.com).

**JEFF MCWHERTER** is a partner and director of development at Gravity Works Design and Development, based in a historic office in Lansing Michigan's Old Town District. A graduate of Michigan State University with over 12 years of professional software development experience, Jeff holds numerous certifications from Microsoft including Microsoft Certified Solutions Developer (MCSD), Microsoft Certified Database Administrator (MCDBA), Microsoft Certified Application Developer (MCAD), and Microsoft Technology Specialist (MCTS).

In 2010 Jeff was awarded with the Microsoft Most Valuable Professional (MVP) for the third year in a row. Also in 2010, Jeff received the Ten Over The Next Ten award presented by the Lansing Regional Chamber of Commerce, which recognizes 10 young professionals to "watch" over the next 10 years. Jeff is also a published author, with *Testing ASP.NET Web Applications* published by Wrox Press.

Along with being an author and software developer, Jeff is very active in developing programming communities across the country by speaking at conferences and organizing events such as the Lansing Give Camp, which pairs developers with non-profit organizations for volunteer projects.

**MICHAEL EATON** has been developing awesome solutions using Microsoft tools and technologies since 1994, but in 2001 he broke free from the confines of the cube farm to go out on his own. While he lives in the middle-of-nowhere Michigan, he serves clients throughout the Midwest. Well known for his dislike of web development and box lunches, his focus over the past few years has been on XAML-based technologies like WPF and Silverlight. He speaks at regional events and user groups, runs the Kalamazoo X Conference and helps with the Ann Arbor Give Camp. He is also a C# MVP. When not working on projects or spending time with his family, he treats his World of Warcraft addiction with ample doses of time on his Xbox 360.

# ABOUT THE TECHNICAL EDITOR

**MITCHEL SELLERS** specializes in software development using Microsoft technologies. He is the CEO of IowaComputerGurus Inc., a Microsoft C# MVP, a Microsoft Certified Professional, has served as an author on two books, and served as technical editor on many other books. You will often find Mitchel interacting with the greater software development community either at events/conferences or in online discussion forums. To obtain additional information on Mitchel's professional experience, certifications, and publications refer to his resume at [MitchelSellers.com](http://MitchelSellers.com).



# CREDITS

**ACQUISITIONS EDITOR**

Paul Reese

**PROJECT EDITOR**

Sydney Jones

**TECHNICAL EDITORS**

Jeff McWherter

Mitchell Sellers

**PRODUCTION EDITOR**

Rebecca Anderson

**COPY EDITOR**

Gayle Johnson

**EDITORIAL DIRECTOR**

Robyn B. Siesky

**EDITORIAL MANAGER**

Mary Beth Wakefield

**FREELANCER EDITORIAL MANAGER**

Rosemarie Graham

**ASSOCIATE DIRECTOR OF MARKETING**

David Mayhew

**PRODUCTION MANAGER**

Tim Tate

**VICE PRESIDENT AND EXECUTIVE GROUP****PUBLISHER**

Richard Swadley

**VICE PRESIDENT AND EXECUTIVE PUBLISHER**

Barry Pruett

**ASSOCIATE PUBLISHER**

Jim Minatel

**PROJECT COORDINATOR, COVER**

Katie Crocker

**PROOFREADER**

Carrie Hunter, Word One New York

**INDEXER**

J & J Indexing

**COVER DESIGNER**

Michael E. Trent

**COVER IMAGE**

© iStock / technotr





# ACKNOWLEDGMENTS

**I WANT TO START BY THANKING MY GIRLFRIEND** (with any luck, fiancé by the time you are reading this) Gayle. She has been very supportive and EXTREMELY understanding during the process of writing this book. More than she should have had to be. Thank you.

I want to thank my parents for making this book possible by making me possible. My mother is so proud she'll read every page of this book. Bless her heart. I hope she's still proud when she realizes I lied and this is NOTHING like a Stephen King novel.

In the understanding and supporting department I would also like to thank Daniel Grey, Mark Kovacevich, Jeff Perry and everyone else at Improving Enterprises. I'd also like to thank Pete Klassen. We miss you man!

I'd like to thank Jeff McWherter and Michael Eaton for their contributions to this book. Jeff; thanks for taking some of the load off. Mike; thank you for pushing me to include the “non-web” people. I'd also like to thank my editor Sydney for making this book look like I know how to write.

Brian Prince; thank you for pushing me to get involved in the development community. I was going to write something funny here, but I couldn't think of anything. I'll getcha in the next one.

When I was presented with the opportunity to write this book, I almost said no. I want to thank Ted Neward for talking me into it. So, this is kinda your fault too.

I want to thank my partners in NPlus1.org Mike Wood and Chris Woodruff for picking up my slack of the past several months while I worked on this.

Long list of general thanks: Brahma Ghosh, Brian Sherwin, Bill Sempf, Jeff Blankenburg, Carey Payette, Caleb Jenkins, Jennifer Marsman, Sarah & Kevin Dutkiewicz, Steve Harman, Josh Holmes. Thanks to Matt Groves for pimping this book almost as much as I did. I'm sure I forgot someone, so I apologize.

—JAMES

**FIRST AND FOREMOST I WOULD LIKE TO THANK** my very patient wife Carla. Thank you for all the support, patience, and understanding you have provided to me for all of my endeavors. Thank you to the staff at Gravity Works — Amelia Marschall, Lauren Colton, Scott Gowell and Dave Smith — for answering my random questions that appeared to come out of nowhere. And lastly I would like to thank James for his hard work, dedication, and friendship.

—JEFF



# CONTENTS

## INTRODUCTION

xxv

## PART I: GETTING STARTED

<b>CHAPTER 1: THE ROAD TO TEST-DRIVEN DEVELOPMENT</b>	<b>3</b>
<b>The Classical Approach to Software Development</b>	<b>4</b>
A Brief History of Software Engineering	4
From Waterfall to Iterative and Incremental	5
<b>A Quick Introduction to Agile Methodologies</b>	<b>6</b>
A Brief History of Agile Methodologies	6
The Principles and Practices of Test-Driven Development	7
<b>The Concepts Behind TDD</b>	<b>8</b>
TDD as a Design Methodology	8
TDD as a Development Practice	8
<b>The Benefits of TDD</b>	<b>9</b>
<b>A Quick Example of the TDD Approach</b>	<b>10</b>
<b>Summary</b>	<b>17</b>
<b>CHAPTER 2: AN INTRODUCTION TO UNIT TESTING</b>	<b>19</b>
<b>What Is a Unit Test?</b>	<b>19</b>
Unit Test Definition	20
What Is Not a Unit Test?	20
Other Types of Tests	22
<b>A Brief Look at NUnit</b>	<b>24</b>
What Is a Unit Test Framework?	24
The Basics of NUnit	25
<b>Decoupling with Mock Objects</b>	<b>28</b>
Why Mocking Is Important	28
Dummy, Fake, Stub, and Mock	29
Best and Worst Practices	35
<b>A Brief Look at Moq</b>	<b>36</b>
What Does a Mocking Framework Do?	36
A Bit About Moq	36
Moq Basics	36
<b>Summary</b>	<b>40</b>

<b>CHAPTER 3: A QUICK REVIEW OF REFACTORING</b>	<b>41</b>
<b>Why Refactor?</b>	<b>42</b>
A Project's Lifecycle	42
Maintainability	43
Code Metrics	43
<b>Clean Code Principles</b>	<b>45</b>
OOP Principles	45
Encapsulation	45
Inheritance	46
Polymorphism	48
The SOLID Principles	49
The Single Responsibility Principle	50
The Open/Close Principle	50
The Liskov Substitution Principle	51
The Interface Segregation Principle	51
The Dependency Inversion Principle	52
<b>Code Smells</b>	<b>52</b>
What Is a Code Smell?	52
Duplicate Code and Similar Classes	53
Big Classes and Big Methods	54
Comments	55
Bad Names	56
Feature Envy	57
Too Much If/Switch	58
Try/Catch Bloat	59
<b>Typical Refactoring</b>	<b>60</b>
Extract Classes or Interfaces	60
Extract Methods	62
Rename Variables, Fields, Methods, and Classes	66
Encapsulate Fields	67
Replace Conditional with Polymorphism	68
Allow Type Inference	71
<b>Summary</b>	<b>71</b>
<b>CHAPTER 4: TEST-DRIVEN DEVELOPMENT: LET THE TESTS             BE YOUR GUIDE</b>	<b>73</b>
<b>It Starts with the Test</b>	<b>74</b>
<b>Red, Green, Refactor</b>	<b>76</b>
The Three Phases of TDD	77
The Red Phase	77
The Green Phase	78

The Refactoring Phase	79
Starting Again	79
<b>A Refactoring Example</b>	<b>79</b>
The First Feature	80
Making the First Test Pass	83
The Second Feature	83
Refactoring the Unit Tests	85
The Third Feature	87
Refactoring the Business Code	88
Correcting Refactoring Defects	91
The Fourth Feature	93
<b>Summary</b>	<b>94</b>
 <b>CHAPTER 5: MOCKING EXTERNAL RESOURCES</b>	 <b>97</b>
<b>The Dependency Injection Pattern</b>	<b>98</b>
Working with a Dependency Injection Framework	99
<b>Abstracting the Data Access Layer</b>	<b>108</b>
Moving the Database Concerns Out of the Business Code	108
Isolating Data with the Repository Pattern	108
Injecting the Repository	109
Mocking the Repository	112
<b>Summary</b>	<b>113</b>
 <b>PART II: PUTTING BASICS INTO ACTION</b>	
 <b>CHAPTER 6: STARTING THE SAMPLE APPLICATION</b>	 <b>117</b>
<b>Defining the Project</b>	<b>118</b>
Developing the Project Overview	118
Defining the Target Environment	119
Choosing the Application Technology	120
<b>Defining the User Stories</b>	<b>120</b>
Collecting the Stories	120
Defining the Product Backlog	122
<b>The Agile Development Process</b>	<b>123</b>
Estimating	124
Working in Iterations	124
Communication Within Your Team	126
Iteration Zero: Your First Iteration	127
Testing in Iteration Zero	127
Ending an Iteration	128

<b>Creating the Project</b>	<b>129</b>
Choosing the Frameworks	129
Defining the Project Structure	131
Organizing Project Folders	131
Creating the Visual Studio Solution	132
<b>Summary</b>	<b>134</b>
<b>CHAPTER 7: IMPLEMENTING THE FIRST USER STORY</b>	<b>137</b>
<b>The First Test</b>	<b>138</b>
Choosing the First Test	138
Naming the Test	139
Writing the Test	140
<b>Implementing the Functionality</b>	<b>148</b>
Writing the Simplest Thing That Could Possibly Work	148
Running the Passing Test	157
Writing the Next Test	158
<b>Improving the Code by Refactoring</b>	<b>165</b>
<b>Triangulation of Tests</b>	<b>166</b>
<b>Summary</b>	<b>166</b>
<b>CHAPTER 8: INTEGRATION TESTING</b>	<b>169</b>
<b>Integrate Early; Integrate Often</b>	<b>170</b>
<b>Writing Integration Tests</b>	<b>171</b>
How to Manage the Database	171
How to Write Integration Tests	172
Reviewing the IRepository	173
Adding Ninject for Dependency Injection	174
Creating the Fluent NHibernate Configuration	177
Creating the Fluent NHibernate Mapping	179
Creating the Integration Test	183
End-to-End Integration Tests	191
Keeping Various Types of Tests Apart	191
<b>When and How to Run Integration Tests</b>	<b>191</b>
<b>Summary</b>	<b>192</b>
<b>PART III: TDD SCENARIOS</b>	
<b>CHAPTER 9: TDD ON THE WEB</b>	<b>197</b>
<b>ASP.NET Web Forms</b>	<b>197</b>
Web Form Organization	198
ASPX Files	198

Code-Behind Files	198
Implementing Test-Driven Development with MVP and Web Forms	199
<b>Working with the ASP.NET MVC</b>	<b>210</b>
MVC 101	211
Microsoft ASP.NET MVC 3.0	212
Creating an ASP.NET MVC Project	212
Creating Your First Test	213
Making Your First Test Pass	215
Creating Your First View	216
Gluings Everything Together	217
Using the MVC Contrib Project	220
ASP.NET MVC Summarized	220
<b>Working with JavaScript</b>	<b>220</b>
JavaScript Testing Frameworks	221
<b>Summary</b>	<b>226</b>
 <b>CHAPTER 10: TESTING WINDOWS COMMUNICATION FOUNDATION SERVICES</b>	 <b>227</b>
<b>WCF Services in Your Application</b>	<b>228</b>
Services Are Code Too	228
<b>Testing WCF Services</b>	<b>228</b>
Refactoring for Testability	229
Introducing Dependency Injection to Your Service	231
Writing the Test	236
Stubbing the Dependencies	239
Verifying the Results	243
Trouble Spots to Watch	244
<b>Summary</b>	<b>244</b>
 <b>CHAPTER 11: TESTING WPF AND SILVERLIGHT APPLICATIONS</b>	 <b>245</b>
<b>The Problem with Testing the User Interface</b>	<b>246</b>
The MVVM Pattern	246
How MVVM Makes WPF/Silverlight Applications Testable	248
Bringing It All Together	261
<b>Summary</b>	<b>263</b>
 <b>PART IV: REQUIREMENTS AND TOOLS</b>	
 <b>CHAPTER 12: DEALING WITH DEFECTS AND NEW REQUIREMENTS</b>	 <b>267</b>
<b>Handling Change</b>	<b>268</b>
Change Happens	268

Adding New Features	268
Addressing Defects	269
Starting with a Test	270
Changing the Code	272
Keeping the Tests Passing	276
<b>Summary</b>	<b>276</b>
<b>CHAPTER 13: THE GREAT TOOL DEBATE</b>	<b>279</b>
<b>Test Runners</b>	<b>279</b>
TestDriven.NET	280
Developer Express Test Runner	280
Gallio	281
<b>Unit Testing Frameworks</b>	<b>282</b>
MSTest	282
MbUnit	283
xUnit	284
<b>Mocking Frameworks</b>	<b>285</b>
Rhino Mocks	285
Type Mock	287
<b>Dependency Injection Frameworks</b>	<b>289</b>
Structure Map	289
Unity	291
Windsor	293
Autofac	294
<b>Miscellaneous Useful Tools</b>	<b>295</b>
nCover	295
PEX	295
<b>How to Introduce TDD to Your Team</b>	<b>296</b>
Working in Environments That Are Resistant to Change	297
Working in Environments That Are Accepting of Change	297
<b>Summary</b>	<b>297</b>
<b>CHAPTER 14: CONCLUSIONS</b>	<b>299</b>
<b>What You Have Learned</b>	<b>299</b>
You Are the Client of Your Code	300
Find the Solutions Step by Step	300
Use the Debugger as a Surgical Instrument	300
<b>TDD Best Practices</b>	<b>301</b>
Use Significant Names	301
Write at Least One Test for One Unit of Functionality	301
Keep Your Mocks Simple	302



The Benefits of TDD	302
How to Introduce TDD in Your Team	303
Summary	304
<b>APPENDIX: TDD KATAS</b>	<b>307</b>
Working with TDD Katas	307
Share Your Work	308
OSIM User Stories	308
<b><i>INDEX</i></b>	<b><i>311</i></b>



# INTRODUCTION

**AS A CONSULTANT, I WORK WITH MANY DEVELOPERS.** At each client I get to meet a new team and see how they develop software. I've seen great teams, and I've seen teams that are so broken they have never had a successful project. Over the years I've noticed that different teams along this success continuum have different traits. And I've started to formulate an idea of what makes a development team able to develop and deploy applications that are high-quality and deliver value to the business.

The observation that most people expect me to make is that the successful teams had smarter, more competent people, and certainly they did. But the teams that failed had plenty of smart people as well. Clearly intelligence is not a key factor in success.

What I observed about the successful teams was that they had a passion for technology and pride in the work they produced. They were always learning about new tools and techniques, with the aim of developing software faster and with fewer bugs. On the other hand, the less successful teams were content to stick with their old ways of doing things and never took an interest in the changes that were going on around them.

Not all those successful, passionate development teams were practicing test-driven development (TDD) when I first found them. However, most of them quickly and eagerly latched on to it when introduced to the concept. These teams have found that adding the practice of test-driven development to their process of building software produced immediate, measurable results by increasing quality and reducing the number of defects in the delivered application.

Passion is difficult to create but easy to kill. In teams that lack passion, the introduction of test-driven development has, in many cases, reignited passion in developers. This is particularly true of developers who have grown tired of doing the same kind of development day in and day out.

Passion aside, there is another very compelling reason to investigate test-driven development. Arguably the two biggest changes in recent years with the potential to reach the largest number of developers are the rise of agile methodologies and test-driven development. Often the two go hand in hand. I don't believe that an agile methodology can succeed in the long term without the use of test-driven development, and I have great difficulty seeing how test-driven development could work in a waterfall environment.

Agile is here to stay. It's no longer a "crazy cowboy coding" way of working practiced by small development shops. Large companies that have made huge investments in structuring their IT departments around waterfalls are starting to build more and more projects with an agile methodology. Even the most bureaucratic organization in existence, government, is starting to investigate agile with great success. These developments spell out a clear reality: Developers who can work in agile environments, including the practice of test-driven development, soon will be more valuable than those who can't.

Test-driven development has not existed in a vacuum. In the past several years, many groups and movements have been aimed at raising the quality of the software being developed and bringing business into the process. New principles and ways of doing things have been advanced to help developers build maintainable applications that serve the needs of the business. Terms such as software craftsmanship and SOLID have made their way into the lexicon of passionate developers all over the world. Some developers have even gone so far as to call themselves software artisans or craftsmen.

Many books, websites, and workshops have appeared to feed the need to learn test-driven development and all its supporting pieces. Many of these are very good. But others are nothing more than commercials for a common and transportable way of doing things that is dressed up as an expensive and proprietary solution. Many smart, passionate developers talk about and evangelize test-driven development. However, no “one-stop shopping” resource has been able to take a developer — specifically, a .NET developer — from neophyte to, well, still a neophyte, but a neophyte with some information.

The fact that you are reading this book indicates that you have some interest in test-driven development. Maybe you’re a developer who’s heard a lot about test-driven development but never really had an opportunity to explore it. Perhaps you’re an experienced test-driven developer who is curious to see how this book is different from all the other books on the subject. In either case, the fact that you are reading this book indicates that test-driven development has become mainstream and is worthy of your time to learn, practice, and promote.

## WHO THIS BOOK IS FOR

Test-driven development is an effective way to build quality into your application from the start. The supporting principles and practices of test-driven development will enable you and your development team to quickly write maintainable software that is more aligned with the needs of the business. If you are a developer interested in improving your skills, this book is for you.

If you’re new to test-driven development, start with Chapter 1. Doing so will give you a good background in why test-driven development has become such a compelling practice. It will also introduce you to the concepts of object-oriented programming, the SOLID Principles, and refactoring. These skills are a crucial foundation for the practice of test-driven development.

If you’ve dabbled in test-driven development, you might want to start with Chapter 3, which provides a refresher on object-oriented development, the SOLID Principles, and refactoring. Even seasoned developers sometimes need a reminder of how these concepts relate to application development. The rest of the book, starting with Chapter 4, provides form and structure for test-driven development for these developers.

Developers who are experienced with test-driven development will probably want to start with Part III. Doing so assumes that you have a high degree of skill with test-driven development, object-oriented programming (OOP), and SOLID. This part focuses on specific scenarios that .NET developers face. It covers how to practice test-driven development in web-based applications (including web forms, ASP.NET MVC, and JavaScript), applications built on Windows Presentation

Foundation (WPF) with the Model-View-ViewModel (MVVM) pattern, and service applications built using Microsoft's Windows Communication Foundation (WCF). The most difficult part of an application to test is the edge. These chapters will show you how to make the edges around your application as thin as possible and therefore more testable.

## WHAT THIS BOOK COVERS

This book starts by covering the conditions that brought the software industry to the point where test-driven development could flourish. It's important to understand this history and the conditions that brought software development to its current state. Avoiding the mistakes of the past is important. But identifying these antipatterns in your current development practice is even more important.

To support your practice of test-driven development, this book also includes extensive coverage of object-oriented programming, agile methodologies, and the SOLID software design and coding principles.

Of course, this book covers the concepts inherent in and necessary to test-driven development. The first tests you will be exposed to are simple and easy to understand. You'll see how the NUnit unit-testing framework can be used to write unit tests in Visual Studio.

Later, the dependency injection pattern is introduced. You will see how this pattern is implemented and how dependency injection frameworks such as Ninject can help manage the dependencies in your application. The practice of mocking and mocking frameworks also are covered, including an introduction to the mocking framework Moq.

The basics of behavior-driven development are covered, but a deep discussion of this topic is not included. This book explains the idea behind behavior-driven development and showcases the business-driven development style of naming tests. This book also introduces the NBehave testing framework. NBehave has many features, but this book simply uses it to provide syntactic sugar for the tests.

## HOW THIS BOOK IS STRUCTURED

A great deal of effort has been expended to structure the information in this book so that each chapter builds upon the lessons in the previous one. The first chapters are designed to provide a foundation built on the importance of test-driven development and the underlying skills needed to effectively practice it. Each chapter and section build on a concept such as dependency injection and mocking until you've been exposed to all the necessary tools and techniques to practice test-driven development.

Incorporating the test-driven development skills taught in the previous chapters, Part III demonstrates how to practice test-driven development with several of Microsoft's frameworks aimed at developing interfaces for applications, including ASP.NET MVC, WPF, and WCF.

The book ends with an appendix that lists some alternative tools that can help you develop applications using test-driven development. It also lists potential user stories to use as practice if you are not in a position to use test-driven development in your everyday work.

## WHAT YOU NEED TO USE THIS BOOK

To follow along with the examples in this book and use the demonstration application available for download at [www.wrox.com](http://www.wrox.com), you need the following tools:

- Visual Studio 2010 (any version)
- NUnit version 2.5.2.9222 or later, available at [nunit.org](http://nunit.org)
- Moq version 4 beta 4 (build 4.0.10827.0) or later, available at [code.google.com/p/moq](http://code.google.com/p/moq)
- Ninject version 2 (build 2.1.0.91) or later, available at [ninject.org](http://ninject.org)
- NBehave version 0.4.5.183 or later, available at [nbehave.org](http://nbehave.org)
- Fluent NHibernate version 1.1 or later, available at [fluentnhibernate.org](http://fluentnhibernate.org)
- A Database Management System (DBMS) is required for the sample applications. The examples in this book use Microsoft SQL Server Developer, but any relational database system will suffice.

## CONVENTIONS

To help you get the most from the text and keep track of what's happening, we use a number of conventions throughout the book: As for styles in the text:

- We *italicize* new terms and important words when we introduce them.
- We show keyboard strokes like this: Ctrl+A.
- We show filenames, URLs, and code within the text like so: `persistence.properties`.

We present code in two different ways:

We use a monospace type with no highlighting for most code examples.  
We use bold to emphasize code that's particularly important in the present context.



*The pencil icon indicates notes, tips, hints, tricks, or asides to the current discussion.*