Professional

Test Driven Development with C#

Developing Real-World Applications with TDD

James Bender, Jeff McWherter

PROFESSIONAL TEST-DRIVEN DEVELOPMENT WITH C#

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

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

Library of Congress Control Number: 2011924919

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`) and continues to lead the Columbus Architects Group (`www.colarc.org`) and is the senior editor of first-party content for nplus1.org, an educational website aimed toward architects and aspiring architects. His blog can be found at `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`.

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

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`, you need the following tools:

- Visual Studio 2010 (any version)
- NUnit version 2.5.2.9222 or later, available at `nunit.org`
- Moq version 4 beta 4 (build 4.0.10827.0) or later, available at `code.google.com/p/moq`
- Ninject version 2 (build 2.1.0.91) or later, available at `ninject.org`
- NBehave version 0.4.5.183 or later, available at `nbehave.org`
- Fluent NHibernate version 1.1 or later, available at `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 monofont 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.